Scyld ClusterWare Documentation

Release 7.9.16

Penguin Computing
1 Release Notes

1.1 Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000

1.1.1 About This Release

1.1.1.1 Important: Recommend using /usr/sbin/install-scyld script

1.1.1.2 Important for clusters using 3rd-party drivers or applications

1.1.1.3 Important for clusters using Panasas storage

1.1.2 First Installation of Scyld ClusterWare 7 On A Server

1.1.3 Upgrading An Earlier Release of Scyld ClusterWare 7 to 7.9

1.1.4 Post-Installation Configuration Issues

1.1.4.1 Resolve *.rpmnew and *.rpmsave configuration file differences

1.1.4.2 Disable SELinux and NetworkManager

1.1.4.3 Edit /etc/beowulf/conf.d/sysctl.conf as needed

1.1.4.4 Optionally reduce size of /usr/lib/locale/locale-archive

1.1.4.5 Optionally configure and enable compute node CPU speed/power management

1.1.4.6 Optionally install a different TORQUE package

1.1.4.7 Optionally enable job manager

1.1.4.8 Optionally enable TORQUE scheduler

1.1.4.9 Optionally enable Ganglia monitoring tool

1.1.4.10 Optionally enable NFS locking

1.1.4.11 Optionally adjust the size limit for locked memory

1.1.4.12 Optionally increase the max number of processes per user

1.1.4.13 Optionally enable SSHD on compute nodes

1.1.4.14 Optionally allow IP Forwarding

1.1.4.15 Optionally increase the nf_conntrack table size

1.1.4.16 Optionally configure vm.zone_reclaim_mode on compute nodes

1.1.4.17 Optionally configure automount on compute nodes

1.1.4.18 Optionally reconfigure node names

1.1.5 Post-Installation Configuration Issues For Large Clusters

1.1.5.1 Optionally increase the number of nfsd threads

1.1.5.2 Optionally increase the max number of processID values

1.1.5.3 Optionally increase the max number of open files

1.1.5.4 Issues with Ganglia

1.1.6 Post-Installation Release of Updated Packages

1.1.7 Notable Feature Enhancements And Bug Fixes

1.1.7.1 v7.9.16 - August 19, 2022

1.1.7.2 v7.9.15 - July 8, 2022

1.1.7.3 v7.9.14 - June 1, 2022

1.1.7.4 v7.9.13 - April 13, 2022

1.1.7.5 v7.9.12 - March 2, 2022
<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>v7.9.11</td>
<td>January 20, 2022</td>
<td>14</td>
</tr>
<tr>
<td>v7.9.10</td>
<td>December 6, 2021</td>
<td>14</td>
</tr>
<tr>
<td>v7.9.9</td>
<td>October 20, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.8</td>
<td>September 15, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.7</td>
<td>July 27, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.6</td>
<td>June 21, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.5</td>
<td>May 26, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.4</td>
<td>April 27, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.3</td>
<td>April 7, 2021</td>
<td>15</td>
</tr>
<tr>
<td>v7.9.2</td>
<td>February 11, 2021</td>
<td>16</td>
</tr>
<tr>
<td>v7.9.1</td>
<td>January 6, 2021</td>
<td>16</td>
</tr>
<tr>
<td>v7.9.0</td>
<td>December 9, 2020</td>
<td>16</td>
</tr>
<tr>
<td>v7.8.5</td>
<td>September 22, 2020</td>
<td>16</td>
</tr>
<tr>
<td>v7.8.4</td>
<td>August 5, 2020</td>
<td>17</td>
</tr>
<tr>
<td>v7.8.3</td>
<td>June 17, 2020</td>
<td>17</td>
</tr>
<tr>
<td>v7.8.2</td>
<td>May 6, 2020</td>
<td>17</td>
</tr>
<tr>
<td>v7.8.1</td>
<td>April 1, 2020</td>
<td>18</td>
</tr>
<tr>
<td>v7.7.4</td>
<td>February 20, 2020</td>
<td>18</td>
</tr>
<tr>
<td>v7.7.3</td>
<td>December 13, 2019</td>
<td>18</td>
</tr>
<tr>
<td>v7.7.2</td>
<td>November 27, 2019</td>
<td>18</td>
</tr>
<tr>
<td>v7.7.1</td>
<td>October 18, 2019</td>
<td>19</td>
</tr>
<tr>
<td>v7.7.0</td>
<td>September 24, 2019</td>
<td>19</td>
</tr>
<tr>
<td>v7.6.4</td>
<td>August 16, 2019</td>
<td>19</td>
</tr>
<tr>
<td>v7.6.3</td>
<td>July 10, 2019</td>
<td>20</td>
</tr>
<tr>
<td>v7.6.2</td>
<td>June 11, 2019</td>
<td>20</td>
</tr>
<tr>
<td>v7.6.1</td>
<td>April 8, 2019</td>
<td>20</td>
</tr>
<tr>
<td>v7.6.0</td>
<td>December 14, 2018</td>
<td>20</td>
</tr>
<tr>
<td>v7.5.4</td>
<td>October 2, 2018</td>
<td>21</td>
</tr>
<tr>
<td>v7.5.3</td>
<td>July 27, 2018</td>
<td>22</td>
</tr>
<tr>
<td>v7.5.2</td>
<td>July 14, 2018</td>
<td>22</td>
</tr>
<tr>
<td>v7.5.1</td>
<td>June 4, 2018</td>
<td>23</td>
</tr>
<tr>
<td>v7.5.0</td>
<td>June 4, 2018</td>
<td>23</td>
</tr>
<tr>
<td>v7.4.6</td>
<td>October 3, 2018</td>
<td>24</td>
</tr>
<tr>
<td>v7.4.5</td>
<td>March 23, 2018</td>
<td>25</td>
</tr>
<tr>
<td>v7.4.4</td>
<td>February 8, 2018</td>
<td>25</td>
</tr>
<tr>
<td>v7.4.3</td>
<td>January 11, 2018</td>
<td>26</td>
</tr>
<tr>
<td>v7.4.2</td>
<td>December 26, 2017</td>
<td>26</td>
</tr>
<tr>
<td>v7.4.1</td>
<td>October 31, 2017</td>
<td>27</td>
</tr>
<tr>
<td>v7.4.0</td>
<td>October 11, 2017</td>
<td>27</td>
</tr>
<tr>
<td>v7.4.0</td>
<td>October 6, 2017</td>
<td>27</td>
</tr>
<tr>
<td>v7.3.7</td>
<td>October 13, 2017</td>
<td>28</td>
</tr>
<tr>
<td>v7.3.6</td>
<td>August 7, 2017</td>
<td>28</td>
</tr>
<tr>
<td>v7.3.5</td>
<td>June 28, 2017</td>
<td>28</td>
</tr>
<tr>
<td>v7.3.4</td>
<td>June 16, 2017</td>
<td>28</td>
</tr>
<tr>
<td>v7.3.3</td>
<td>April 27, 2017</td>
<td>29</td>
</tr>
<tr>
<td>v7.3.2</td>
<td>April 3, 2017</td>
<td>29</td>
</tr>
<tr>
<td>v7.3.1</td>
<td>March 8, 2017</td>
<td>29</td>
</tr>
<tr>
<td>v7.3.0</td>
<td>January 20, 2017</td>
<td>30</td>
</tr>
<tr>
<td>v7.2.0</td>
<td>November 14, 2016</td>
<td>30</td>
</tr>
</tbody>
</table>

1.1.8 Known Issues And Workarounds

1.1.8.1 Issues with bootmodule firmware

1.1.8.2 Kernel panic using non-\textit{invpcid} old Intel nodes
2.8.1.1 bpstat ................................................................. 61
2.8.1.2 BeoStatus ............................................................. 61
2.8.2 Running Jobs Across the Cluster ........................................ 63
  2.8.2.1 Directed Execution with bps ................................. 63
  2.8.2.2 Dynamic Execution with beorun and mpprun ................. 63
2.9 Troubleshooting ClusterWare ........................................ 63
  2.9.1 Failing PXE Network Boot ....................................... 63
  2.9.2 Mixed Uni-Processor and SMP Cluster Nodes .................. 65
  2.9.3 IP Forwarding .................................................... 65
  2.9.4 SSH Traffic ...................................................... 65
  2.9.5 Device Driver Updates ......................................... 65
  2.9.6 Finding Further Information .................................. 66
2.10 Compute Node Disk Partitioning .................................. 66
  2.10.1 Architectural Overview ...................................... 66
  2.10.2 Operational Overview ....................................... 66
  2.10.3 Disk Partitioning Procedures ................................ 67
    2.10.3.1 Typical Partitioning .................................. 67
    2.10.3.2 Default Partitioning .................................. 68
    2.10.3.3 Generalized, User-Specified Partitions ............... 68
    2.10.3.4 Unique Partitions ..................................... 68
  2.10.4 Mapping Compute Node Partitions ................................ 69
2.11 Changes to Configuration Files ................................. 69
  2.11.1 Changes to Red Hat Configuration Files .................... 69
  2.11.2 Possible Changes to ClusterWare Configuration Files .... 70
3 Eval Installation Guide ............................................. 71
  3.1 Evaluation Installation Instructions ............................ 71
  3.2 Scyld ClusterWare System Overview .............................. 71
  3.3 Hardware Requirements .......................................... 72
  3.4 BIOS Requirements .............................................. 72
  3.5 Software Requirements ......................................... 73
  3.6 Install Yum Configuration File and License File .............. 73
  3.7 Update Base Distribution and Install Scyld Clusterware Software .... 73
  3.8 Configure the Private and Public Networks ..................... 73
  3.9 Start Cluster Operations ........................................ 74
  3.10 Documentation and Support ..................................... 75
  3.11 Purchasing Scyld ClusterWare ................................... 75
  3.12 Uninstalling Scyld ClusterWare ................................. 75
4 Administrator’s Guide .............................................. 76
  4.1 Preface .................................................................. 76
  4.2 Scyld ClusterWare Design Overview ............................. 76
    4.2.1 System Architecture ......................................... 77
      4.2.1.1 System Hardware Context ................................ 77
      4.2.1.2 Network Topologies .................................... 79
      4.2.1.3 System Data Flow ...................................... 81
      4.2.1.4 System Software Context ............................... 83
      4.2.1.5 System Level Files ..................................... 84
    4.2.2 Technical Description ...................................... 85
      4.2.2.1 Compute Node Boot Procedure ......................... 85
      4.2.2.2 BProc Distributed Process Space ..................... 87
      4.2.2.3 Compute Node Categories ............................... 88
      4.2.2.4 Compute Node States .................................. 89
      4.2.2.5 Miscellaneous Components ............................. 91
4.2.3 Software Components .................................. 93
  4.2.3.1 BeoBoot Tools ..................................... 93
  4.2.3.2 BProc Daemons .................................... 94
  4.2.3.3 BProc Clients ..................................... 94
  4.2.3.4 ClusterWare Utilities .............................. 94
4.3 Monitoring the Status of the Cluster ........................ 94
  4.3.1 Monitoring Utilities .................................. 95
  4.3.1.1 Cluster Monitoring Interfaces ..................... 95
  4.3.1.2 Monitoring Daemons ............................... 95
  4.3.2 Using the Data ....................................... 95
  4.3.3 beostatus ............................................. 96
  4.3.3.1 beostatus File Menu ............................... 97
  4.3.3.2 beostatus Modes ................................... 97
  4.3.4 beostat ................................................ 98
  4.3.5 bpstat ................................................ 100
  4.3.6 Ganglia .............................................. 101
  4.3.7 beoweb ............................................... 101
4.4 Configuring the Cluster .................................. 102
  4.4.1 Configuring the Cluster Manually .................... 103
    4.4.1.1 Configuration Files .............................. 103
    4.4.1.2 Command Line Tools ............................. 110
    4.4.1.3 Configuring CPU speed/power for Compute Nodes 111
    4.4.1.4 Adding New Kernel Modules ..................... 112
    4.4.1.5 Accessing External License Servers ............... 113
    4.4.1.6 Configuring SSH for Remote Job Execution ...... 113
    4.4.1.7 Interconnects ................................. 114
  4.4.2 Configuring Users ................................... 115
    4.4.2.1 Adding New Users ............................... 115
    4.4.2.2 Adding a User to a Group ....................... 116
    4.4.2.3 Removing a Group ................................ 117
  4.4.3 Configuring Groups .................................. 117
    4.4.3.1 What Node Ownership Means ...................... 117
    4.4.3.2 Checking Node Ownership ......................... 117
    4.4.3.3 Setting Node Ownership .......................... 118
4.5 Remote Administration and Monitoring ..................... 115
  4.5.1 Command Line Tools ................................ 115
  4.5.2 X Forwarding ....................................... 115
4.6 Managing Users on the Cluster .......................... 115
  4.6.1 Managing User Accounts ............................. 115
    4.6.1.1 Adding New Users ............................... 115
    4.6.1.2 Removing Users ................................. 116
  4.6.2 Managing User Groups ................................ 116
    4.6.2.1 Creating a Group ............................... 116
    4.6.2.2 Adding a User to a Group ....................... 116
    4.6.2.3 Removing a Group ................................ 117
  4.6.3 Controlling Access to Cluster Resources ............ 117
    4.6.3.1 What Node Ownership Means ...................... 117
    4.6.3.2 Checking Node Ownership ......................... 117
    4.6.3.3 Setting Node Ownership .......................... 118
4.7 Job Batching ........................................... 119
  4.7.1 Enabling TORQUE or Slurm ......................... 119
4.8 Managing Non-Scyld Nodes ................................ 120
  4.8.1 DHCP IP address assignment to devices .............. 120
  4.8.2 Simple provisioning using PXE ...................... 121
  4.8.3 Simple provisioning using the class directive .... 121
  4.8.4 Booting a node from the local harddrive ........... 122
  4.8.5 Provisioning a non-Scyld node ...................... 122
  4.8.6 Integrating a non-Scyld node into the cluster .... 123
4.9 Managing Multiple Master Nodes ......................... 124
  4.9.1 Active-Passive Masters ............................ 124
  4.9.2 Active-Active Masters ............................ 124
4.10 Managing Node Failures .................................. 125
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.10.1 Protecting an Application from Node Failure</td>
<td>125</td>
</tr>
<tr>
<td>4.10.2 Compute Node Failure</td>
<td>126</td>
</tr>
<tr>
<td>4.10.2.1 When Compute Nodes Fail</td>
<td>126</td>
</tr>
<tr>
<td>4.10.2.2 Compute Node Data</td>
<td>126</td>
</tr>
<tr>
<td>4.10.3 Master Node Failure</td>
<td>127</td>
</tr>
<tr>
<td>4.10.3.1 When Master Nodes Fail - Without Run-to-Completion</td>
<td>127</td>
</tr>
<tr>
<td>4.10.3.2 When Master Nodes Fail - With Run-to-Completion</td>
<td>127</td>
</tr>
<tr>
<td>4.11 Compute Node Boot Options</td>
<td>128</td>
</tr>
<tr>
<td>4.11.1 Compute Node Boot Media</td>
<td>128</td>
</tr>
<tr>
<td>4.11.1.1 PXE</td>
<td>128</td>
</tr>
<tr>
<td>4.11.1.2 Local Disk</td>
<td>129</td>
</tr>
<tr>
<td>4.11.1.3 Linux BIOS</td>
<td>129</td>
</tr>
<tr>
<td>4.11.1.4 Flash Disk</td>
<td>129</td>
</tr>
<tr>
<td>4.11.2 Changing Boot Settings</td>
<td>129</td>
</tr>
<tr>
<td>4.11.2.1 Adding Steps to the node_up Script</td>
<td>129</td>
</tr>
<tr>
<td>4.11.2.2 Per-Node Parameters</td>
<td>129</td>
</tr>
<tr>
<td>4.11.2.3 Other Per-Node Config Options</td>
<td>130</td>
</tr>
<tr>
<td>4.11.3 Error Logs</td>
<td>130</td>
</tr>
<tr>
<td>4.12 Disk Partitioning</td>
<td>130</td>
</tr>
<tr>
<td>4.12.1 Disk Partitioning Concepts</td>
<td>130</td>
</tr>
<tr>
<td>4.12.2 Disk Partitioning with ClusterWare</td>
<td>131</td>
</tr>
<tr>
<td>4.12.2.1 Master Node</td>
<td>131</td>
</tr>
<tr>
<td>4.12.2.2 Compute Nodes</td>
<td>131</td>
</tr>
<tr>
<td>4.12.3 Default Partitioning</td>
<td>131</td>
</tr>
<tr>
<td>4.12.3.1 Master Node</td>
<td>131</td>
</tr>
<tr>
<td>4.12.3.2 Compute Nodes</td>
<td>131</td>
</tr>
<tr>
<td>4.12.4 Partitioning Scenarios</td>
<td>132</td>
</tr>
<tr>
<td>4.12.4.1 Applying the Default Partitioning</td>
<td>132</td>
</tr>
<tr>
<td>4.12.4.2 Specifying Manual Partitioning</td>
<td>133</td>
</tr>
<tr>
<td>4.13 File Systems</td>
<td>134</td>
</tr>
<tr>
<td>4.13.1 File Systems on a Cluster</td>
<td>134</td>
</tr>
<tr>
<td>4.13.1.1 Local File Systems</td>
<td>134</td>
</tr>
<tr>
<td>4.13.1.2 Network/Cluster File Systems</td>
<td>134</td>
</tr>
<tr>
<td>4.13.2 NFS</td>
<td>134</td>
</tr>
<tr>
<td>4.13.2.1 NFS on Clusters</td>
<td>134</td>
</tr>
<tr>
<td>4.13.2.2 Configuration of NFS</td>
<td>134</td>
</tr>
<tr>
<td>4.13.3 ROMIO</td>
<td>137</td>
</tr>
<tr>
<td>4.13.3.1 Reasons to Use ROMIO</td>
<td>137</td>
</tr>
<tr>
<td>4.13.3.2 Installation and Configuration of ROMIO</td>
<td>137</td>
</tr>
<tr>
<td>4.13.4 Other Cluster File Systems</td>
<td>137</td>
</tr>
<tr>
<td>4.14 Load Balancing</td>
<td>137</td>
</tr>
<tr>
<td>4.14.1 Load Balancing in a Scyld Cluster</td>
<td>138</td>
</tr>
<tr>
<td>4.14.1.1 Mapping Policy</td>
<td>138</td>
</tr>
<tr>
<td>4.14.1.2 Queuing Policy</td>
<td>138</td>
</tr>
<tr>
<td>4.14.2 Implementing a Scheduling Policy</td>
<td>138</td>
</tr>
<tr>
<td>4.15 IPMI</td>
<td>139</td>
</tr>
<tr>
<td>4.15.1 IPMITool</td>
<td>139</td>
</tr>
<tr>
<td>4.16 Updating Software On Your Cluster</td>
<td>139</td>
</tr>
<tr>
<td>4.16.1 What Can’t Be Updated</td>
<td>140</td>
</tr>
<tr>
<td>4.17 Special Directories, Configuration Files, and Scripts</td>
<td>140</td>
</tr>
<tr>
<td>4.17.1 What Resides on the Master Node</td>
<td>140</td>
</tr>
<tr>
<td>4.17.1.1 /etc/beowulf/ directory</td>
<td>140</td>
</tr>
<tr>
<td>4.17.1.2 /usr/lib/beoboot directory</td>
<td>141</td>
</tr>
<tr>
<td>4.17.1.3 /var/beowulf directory</td>
<td>141</td>
</tr>
</tbody>
</table>
5 User’s Guide
5.1 Preface ................................................. 148
5.2 Scyld ClusterWare Overview .............................. 148
5.3 What Is a Beowulf Cluster? ................................. 149
  5.3.1 A Brief History of the Beowulf ...................... 149
  5.3.2 First-Generation Beowulf Clusters .................. 150
  5.3.3 Scyld ClusterWare: A New Generation of Beowulf .. 151
5.4 Scyld ClusterWare Technical Summary ................. 151
  5.4.1 Top-Level Features of Scyld ClusterWare .......... 151
  5.4.2 Process Space Migration Technology .............. 152
  5.4.3 Compute Node Provisioning ....................... 152
  5.4.4 Compute Node Categories ......................... 153
  5.4.5 Compute Node States ............................ 153
  5.4.6 Major Software Components ...................... 153
5.5 Typical Applications of Scyld ClusterWare .......... 154
5.6 Interacting With the System ......................... 154
  5.6.1 Verifying the Availability of Nodes .............. 155
  5.6.2 Monitoring Node Status .......................... 155
    5.6.2.1 The BeoStatus GUI Tool .................... 155
    5.6.2.2 The bpstat Command Line Tool .............. 157
    5.6.2.3 The beostat Command Line Tool .............. 158
  5.6.3 Issuing Commands ................................ 160
    5.6.3.1 Commands on the Master Node .............. 160
    5.6.3.2 Commands on the Compute Node .............. 160
  5.6.4 Copying Data to the Compute Nodes ............. 163
    5.6.4.1 Sharing Data via NFS ....................... 163
    5.6.4.2 Copying Data via bpcp ...................... 163
    5.6.4.3 Programmatic Data Transfer ................. 164
    5.6.4.4 Data Transfer by Migration .................. 164
  5.6.5 Monitoring and Controlling Processes ............ 164
5.7 Running Programs ................................... 165
  5.7.1 Program Execution Concepts ...................... 166
    5.7.1.1 Stand-Alone Computer vs. Scyld Cluster .... 166
    5.7.1.2 Traditional Beowulf Cluster vs. Scyld Cluster . 166
    5.7.1.3 Program Execution Examples ............... 167
  5.7.2 Environment Modules ............................. 169
  5.7.3 Running Programs That Are Not Parallelized .... 169
    5.7.3.1 Starting and Migrating Programs to Compute Nodes (bpsh) .. 169
    5.7.3.2 Copying Information to Compute Nodes (bpcp) .... 170
  5.7.4 Running Parallel Programs ....................... 171
    5.7.4.1 An Introduction to Parallel Programming APIs .. 171
    5.7.4.2 Mapping Jobs to Compute Nodes .............. 173
    5.7.4.3 Running MPICH and MVAPICH Programs ....... 174
    5.7.4.4 Running OpenMPI Programs ................. 175
    5.7.4.5 Running MPICH2 and MVAPICH2 Programs ...... 176
    5.7.4.6 Running PVM-Aware Programs ............... 177
    5.7.4.7 Porting Other Parallelized Programs .......... 177
  5.7.5 Running Serial Programs in Parallel ............ 178
5.7.5.1 mpprun ................................................. 178
5.7.5.2 beorun ................................................. 179
5.7.6 Job Batching .............................................. 179
  5.7.6.1 Job Batching Options for ClusterWare .............. 179
  5.7.6.2 Job Batching with TORQUE ...................... 180
  5.7.6.3 Job Batching with POD Tools .................... 183
5.7.7 Using Singularity ....................................... 183
5.7.8 File Systems ............................................ 184
5.8 Glossary of Parallel Computing Terms .................... 184
5.9 TORQUE and Maui Release Information .................... 186
5.10 OpenMPI Release Information ............................ 186
5.11 MPICH2 Release Information ............................. 215
  5.12.1 CHANGELOG .......................................... 221
  5.12.2 Release Notes ....................................... 255
5.13 MVAPICH2 Release Information .......................... 257
5.14 SLURM Release Information .............................. 294
6 Programmer’s Guide .......................................... 352
  6.1 Under Revision ........................................... 352
7 Reference Guide .............................................. 353
  7.1 Preface ................................................... 353
  7.2 Scyld ClusterWare Commands ............................ 353
  7.2.1 beoboot ............................................... 353
    7.2.1.1 Name ............................................ 353
    7.2.1.2 Synopsis ........................................ 354
    7.2.1.3 Description ..................................... 354
    7.2.1.4 Options ......................................... 354
    7.2.1.5 Examples ........................................ 354
  7.2.2 beoconfig ............................................. 355
    7.2.2.1 Name ............................................ 355
    7.2.2.2 Synopsis ........................................ 355
    7.2.2.3 Description ..................................... 355
    7.2.2.4 Options ......................................... 355
    7.2.2.5 Examples ........................................ 356
  7.2.3 beomap ................................................ 357
    7.2.3.1 Name ............................................ 357
    7.2.3.2 Synopsis ........................................ 357
    7.2.3.3 Description ..................................... 357
    7.2.3.4 Options ......................................... 358
    7.2.3.5 Examples ........................................ 359
    7.2.3.6 Special Notes ................................... 359
  7.2.4 beonpc ................................................ 360
    7.2.4.1 Name ............................................ 360
    7.2.4.2 Synopsis ........................................ 360
    7.2.4.3 Description ..................................... 360
    7.2.4.4 Options ......................................... 360
    7.2.4.5 Example .......................................... 360
  7.2.5 beorun ................................................ 360
    7.2.5.1 Name ............................................ 360
    7.2.5.2 Synopsis ........................................ 361
    7.2.5.3 Description ..................................... 361
    7.2.5.4 Options ......................................... 361
7.2.5.5 Examples ................................................................. 362
7.2.6 beosi ................................................................. 362
  7.2.6.1 Name ............................................................. 362
  7.2.6.2 Synopsis .......................................................... 363
  7.2.6.3 Description ...................................................... 363
  7.2.6.4 Options .......................................................... 363
  7.2.6.5 Examples .......................................................... 364
7.2.7 beostat ............................................................... 365
  7.2.7.1 Name ............................................................. 365
  7.2.7.2 Synopsis .......................................................... 365
  7.2.7.3 Description ...................................................... 365
  7.2.7.4 Options .......................................................... 365
  7.2.7.5 FILTERING OPTIONS ........................................... 366
  7.2.7.6 Examples .......................................................... 366
7.2.8 beostat ............................................................... 367
  7.2.8.1 Name ............................................................. 367
  7.2.8.2 Synopsis .......................................................... 367
  7.2.8.3 Description ...................................................... 367
  7.2.8.4 Options .......................................................... 367
7.2.9 bpcp ................................................................. 368
  7.2.9.1 Name ............................................................. 368
  7.2.9.2 Synopsis .......................................................... 368
  7.2.9.3 Description ...................................................... 368
  7.2.9.4 Options .......................................................... 368
  7.2.9.5 Examples .......................................................... 369
7.2.10 bpdate .............................................................. 369
  7.2.10.1 Name ............................................................. 369
  7.2.10.2 Synopsis .......................................................... 369
  7.2.10.3 Description ...................................................... 369
  7.2.10.4 Options .......................................................... 370
  7.2.10.5 Examples .......................................................... 370
7.2.11 bpsh ............................................................... 370
  7.2.11.1 Name ............................................................. 370
  7.2.11.2 Synopsis .......................................................... 370
  7.2.11.3 Description ...................................................... 370
  7.2.11.4 Options .......................................................... 371
  7.2.11.5 Examples .......................................................... 371
7.2.12 bpstat .............................................................. 372
  7.2.12.1 Name ............................................................. 372
  7.2.12.2 Synopsis .......................................................... 372
  7.2.12.3 Description ...................................................... 373
  7.2.12.4 Options .......................................................... 373
  7.2.12.5 Examples .......................................................... 374
7.2.13 mpprun ............................................................. 375
  7.2.13.1 Name ............................................................. 375
  7.2.13.2 Synopsis .......................................................... 375
  7.2.13.3 Description ...................................................... 375
  7.2.13.4 Options .......................................................... 375
  7.2.13.5 Examples .......................................................... 376
7.3 Scyld ClusterWare Maintenance Commands ...................................... 377
  7.3.1 beofdisk ........................................................... 377
    7.3.1.1 Name .......................................................... 377
    7.3.1.2 Synopsis .......................................................... 377
    7.3.1.3 Description ...................................................... 377
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Description</td>
</tr>
<tr>
<td>7.3.4</td>
<td>Options</td>
</tr>
<tr>
<td>7.3.5</td>
<td>Examples</td>
</tr>
<tr>
<td>7.3.6</td>
<td>Name</td>
</tr>
<tr>
<td>7.3.7</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.3.8</td>
<td>Description</td>
</tr>
<tr>
<td>7.3.9</td>
<td>Options</td>
</tr>
<tr>
<td>7.3.10</td>
<td>Examples</td>
</tr>
<tr>
<td>Section</td>
<td>Topic</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>7.4</td>
<td>Scyld ClusterWare Special Files</td>
</tr>
<tr>
<td>7.4.1</td>
<td>beowulf-config</td>
</tr>
<tr>
<td>7.4.1.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.4.1.2</td>
<td>Description</td>
</tr>
<tr>
<td>7.4.1.3</td>
<td>Config File Format</td>
</tr>
<tr>
<td>7.4.1.4</td>
<td>Keywords</td>
</tr>
<tr>
<td>7.4.1.5</td>
<td>Examples</td>
</tr>
<tr>
<td>7.4.2</td>
<td>beowulf-fstab</td>
</tr>
<tr>
<td>7.4.2.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.4.2.2</td>
<td>Description</td>
</tr>
<tr>
<td>7.4.2.3</td>
<td>Syntax</td>
</tr>
<tr>
<td>7.4.2.4</td>
<td>Examples</td>
</tr>
<tr>
<td>7.4.2.5</td>
<td>Files</td>
</tr>
<tr>
<td>7.4.3</td>
<td>beowulf-nsswitch.conf</td>
</tr>
<tr>
<td>7.4.3.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.4.3.2</td>
<td>Description</td>
</tr>
<tr>
<td>7.4.3.3</td>
<td>See Also</td>
</tr>
<tr>
<td>7.5</td>
<td>Scyld ClusterWare Beostat Libraries</td>
</tr>
<tr>
<td>7.5.1</td>
<td>beostat_count_idle_cpus</td>
</tr>
<tr>
<td>7.5.1.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.5.1.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.5.1.3</td>
<td>Arguments</td>
</tr>
<tr>
<td>7.5.1.4</td>
<td>Description</td>
</tr>
<tr>
<td>7.5.1.5</td>
<td>Examples</td>
</tr>
<tr>
<td>7.5.1.6</td>
<td>Return Value</td>
</tr>
<tr>
<td>7.5.1.7</td>
<td>Errors</td>
</tr>
<tr>
<td>7.5.2</td>
<td>beostat_count_idle_cpus_on_node</td>
</tr>
<tr>
<td>7.5.2.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.5.2.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.5.2.3</td>
<td>Arguments</td>
</tr>
<tr>
<td>7.5.2.4</td>
<td>Description</td>
</tr>
<tr>
<td>7.5.2.5</td>
<td>Examples</td>
</tr>
<tr>
<td>7.5.2.6</td>
<td>Return Value</td>
</tr>
<tr>
<td>7.5.2.7</td>
<td>Errors</td>
</tr>
<tr>
<td>7.5.3</td>
<td>beostat_get_avail_nodes_by_id</td>
</tr>
<tr>
<td>7.5.3.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.5.3.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.5.3.3</td>
<td>Arguments</td>
</tr>
<tr>
<td>7.5.3.4</td>
<td>Description</td>
</tr>
<tr>
<td>7.5.3.5</td>
<td>Examples</td>
</tr>
<tr>
<td>7.5.3.6</td>
<td>Return Value</td>
</tr>
<tr>
<td>7.5.3.7</td>
<td>Errors</td>
</tr>
<tr>
<td>7.5.4</td>
<td>beostat_get_cpu_count</td>
</tr>
<tr>
<td>7.5.4.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.5.4.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.5.4.3</td>
<td>Arguments</td>
</tr>
<tr>
<td>7.5.4.4</td>
<td>Description</td>
</tr>
<tr>
<td>7.5.4.5</td>
<td>Return Value</td>
</tr>
<tr>
<td>7.5.4.6</td>
<td>Errors</td>
</tr>
<tr>
<td>7.5.5</td>
<td>beostat_get_cpu_percent</td>
</tr>
<tr>
<td>7.5.5.1</td>
<td>Name</td>
</tr>
<tr>
<td>7.5.5.2</td>
<td>Synopsis</td>
</tr>
<tr>
<td>7.5.5.3</td>
<td>Arguments</td>
</tr>
<tr>
<td>7.5.5.4</td>
<td>Description</td>
</tr>
</tbody>
</table>
7.6.17 bproc_setnodestatus ................................. 434
  7.6.17.1 Name ........................................ 434
  7.6.17.2 Synopsis ..................................... 434
  7.6.17.3 Arguments ................................... 434
  7.6.17.4 Description .................................. 434
  7.6.17.5 Return Value ................................ 435
  7.6.17.6 Errors ....................................... 435

8 Feedback ................................. 436
1.1 Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000

1.1.1 About This Release

Scyld ClusterWare Release v7.9.16-7916g0000 is the latest update to Scyld ClusterWare 7.

Scyld ClusterWare v7.9.16 expects to execute in a Red Hat RHEL7 Update 9 or CentOS 7.9 base distribution environment, each having been updated to the latest RHEL/CentOS 7 errata (https://rhn.redhat.com/errata/rhel-server-7-errata.html) as of the Scyld ClusterWare v7.9.16 release date. Any compatibility issues between Scyld ClusterWare v7.9.16 and RHEL7 are documented on the Penguin Computing Support Portal at https://www.penguincomputing.com/support.


For the most up-to-date product documentation and other helpful information, visit the Penguin Computing Support Portal.

1.1.1.1 Important: Recommend using /usr/sbin/install-sclyld script

Penguin Computing highly recommends using the /usr/sbin/install-sclyld script to guide the initial installation of Scyld ClusterWare (including updating the RHEL/CentOS base distribution software) and using the /usr/sbin/update-sclyld script (which is equivalent to install-sclyld -u) to update base distribution and ClusterWare software.

Before continuing, make sure you are reading the most recent Scyld ClusterWare Release Notes, which can be found on the Penguin Computing Support Portal at https://www.penguincomputing.com/support/documentation. The most recent version will accurately reflect the current state of the Scyld ClusterWare yum repository of rpms that you are about to install. You may consult the Installation Guide for its more generic and expansive details about the installation process. The Release Notes document more specifically describes how to upgrade an earlier version of Scyld ClusterWare 7 to 7.9 (see Upgrading An Earlier Release of Scyld ClusterWare 7 to 7.9), or how to install Scyld ClusterWare v7.9.16 as a fresh install (see First Installation of Scyld ClusterWare 7 On A Server).
1.1.1.2 Important for clusters using 3rd-party drivers or applications

Before installing or updating Scyld ClusterWare, if your cluster uses any 3rd-party drivers (e.g., Ethernet, InfiniBand, GPU, parallel storage) and if an install or update includes a new kernel, then verify that those 3rd-party drivers can be rebuilt or relinked to the new kernel. If an install or update involves upgrading to a new RHEL/CentOS base distribution, then verify that your cluster’s 3rd-party applications are all supported by that new base distribution.

1.1.1.3 Important for clusters using Panasas storage

If the cluster uses Panasas storage, then you must ensure that the appropriate Panasas kernel module is installed. See the Notable Feature Enhancements And Bug Fixes section for the specific Scyld ClusterWare version you intend to use to determine the name of that kernel’s matching Panasas rpm.

If that Panasas rpm is not already installed, then login to your Panasas account at https://my.panasas.com/portal, click on the Downloads tab, then click on DirectFLOW Client, then click on Search DirectFLOW Release, then do a Keywords search naming the specific rpm to download. Install that rpm after you install the associated ClusterWare kernel. If you do not find the appropriate Panasas rpm, then do not install or upgrade to the desired ClusterWare kernel.

1.1.2 First Installation of Scyld ClusterWare 7 On A Server

When installing Scyld ClusterWare 7 on a system that does not yet contain Scyld ClusterWare, you should perform the following steps:

1. The directory /etc/yum.repos.d/ must contain active repo config files bearing a suffix of .repo. If there is no ClusterWare repo file, then you should download clusterware.repo that gives your cluster access to the customer-facing Scyld ClusterWare yum repos.

   To download a yum repo file that is customized to your cluster:
   2. Click on the tab labeled Assets, and then select a specific Asset Name in the list.
   3. In the Asset Detail section, click on YUM Repo File, which downloads an asset-specific clusterware.repo file, and move that file to the /etc/yum.repos.d/ directory.
   4. Set the permissions: chmod 644 /etc/yum.repos.d/clusterware.repo
   5. The clusterware.repo file contains three sections, labeled cw-core, cw-updates, and cw-next. Generally, the cw-next repo should not be enabled unless so directed by Penguin Computing Support.

2. Examine /etc/yum.repos.d/clusterware.repo to ensure that it specifies the desired yum repository release version. Employ $releasever or 7 to use rpms from the latest Scyld ClusterWare release, which currently is 7.9. Alternatively, a more specific major-minor pair, e.g., 7.2, limits the rpms to just that version, even as ClusterWare releases march forward to newer versions.

3. If updating using a RHEL7 yum repo, then your RHEL7 yum configuration file should also look in the RHEL7 Server Optional repo to find rpms such as compat-dapl-devel and sharutils. The regular CentOS7 yum repo contains these rpms.

4. Install a useful Scyld ClusterWare script that simplifies installing (and later updating) software, then execute that script:

   ```
   yum install install-scyld
   install-scyld
   ```

5. If the cluster uses Panasas storage, then you should have already downloaded the Panasas rpm that matches the Scyld ClusterWare 7 kernel you have just installed. Now install the Panasas rpm using rpm -i.
6. Configure the network for Scyld ClusterWare: edit `/etc/beowulf/config` to specify the cluster interface, the maximum number of compute nodes, and the beginning IP address of the first compute node. See the Installation Guide for more details.

7. Compute nodes must support the PXE network boot protocol. Each node’s BIOS must be configured to prioritize PXE network booting ahead of booting from the local hard drive.

8. If the private cluster network switch uses Spanning Tree Protocol (STP), then either reconfigure the switch to disable STP, or if that is not feasible because of network topology, then enable Rapid STP or portfast on the compute node and edge ports. See Issues with Spanning Tree Protocol and portfast for details.

9. Reboot the master node.

10. After rebooting the new kernel, and after installing any new kernel modules, you should rebuild the master node’s list of modules and dependencies using `depmod`. See Issues with kernel modules for details.

1.1.3 Upgrading An Earlier Release of Scyld ClusterWare 7 to 7.9

If you wish to upgrade a RHEL6/CentOS6 or earlier base distribution to RHEL7/CentOS7, then we recommend you accomplish this with a full install of Release 7, rather than attempt to update from an earlier major release to Release 7. Visit https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux for the Red Hat Enterprise Linux 7 Installation Guide for details. If you already have installed Scyld ClusterWare 6 (or earlier) on the physical hardware that you intend to convert to RHEL7/CentOS7, then we recommend that you backup your master node prior to the new installation of RHEL7/CentOS7, as some of the Scyld ClusterWare configuration files may be a useful reference for Release 7, especially files in `/etc/beowulf/`.

When upgrading from an earlier Scyld ClusterWare 7 version to a newer Scyld ClusterWare 7, you should perform the following steps:

1. Examine `/etc/yum.repos.d/clusterware.repo` to ensure that it specifies the desired yum repository release version. Employ `$releasever` or 7 to use rpms from the latest Scyld ClusterWare release, which currently is 7.9. Alternatively, a more specific major-minor pair, e.g., 7.2, limits the rpms to just that version, even as ClusterWare releases march forward to newer versions.

2. Consider whether or not to stop the cluster prior to updating software. Most updates can be made to a running cluster, although some updates (e.g., those affecting daemons that execute on the master node) require a subsequent restart of the ClusterWare service. Other updates require rebooting the master node, in particular when updating to a new kernel, and this obviously restarts the cluster nodes, too. The safest approach is to stop the cluster before updating the master node, and restart the cluster after the update completes.

   ```bash
   systemctl stop clusterware
   ```

3. Update the software on the master node using the `install-scyld` script that guides you through the process, step by step. If this script doesn’t exist on your system, then install it.

   ```bash
   yum install install-scyld  # if not already installed
   install-scyld -u
   ```

4. The script first determines if it needs to update itself. If that self-update occurs, then the script exits and you should re-execute it.

5. If the cluster uses Panasas storage, then you should have already downloaded the Panasas rpm that matches the Scyld ClusterWare v7.9.16 kernel you have just installed. Now install the Panasas rpm using `rpm -i`.

6. Compare `/etc/beowulf/config`, which remains untouched by the Scyld ClusterWare update, with the new `config.rpmnew` (if that file exists), examine the differences:
and carefully merge the **config.rpmnew** differences into */etc/beowulf/config*. See **Resolve *.rpmnew and *.rpmsave configuration file differences** for details.

Similarly, the preexisting */etc/beowulf/fstab* may have been saved as */fstab.rpmsave* if it was locally modified. If so, merge those local changes back into */etc/beowulf/fstab*.

7. If a new kernel has been installed, then reboot the master node. Otherwise, simply reboot the ClusterWare service:

```bash
systemctl restart clusterware
```

8. After rebooting a new kernel, and after installing any new kernel modules, you should rebuild the master node’s list of modules and dependencies using `depmod`. See **Issues with kernel modules** for details.

### 1.1.4 Post-Installation Configuration Issues

Following a successful update or install of Scyld ClusterWare, you may need to make one or more configuration changes, depending upon the local requirements of your cluster. Larger cluster configurations have additional issues to consider; see **Post-Installation Configuration Issues For Large Clusters**.

#### 1.1.4.1 Resolve *.rpmnew and *.rpmsave configuration file differences

As with every Scyld ClusterWare upgrade, after the upgrade you should locate any Scyld ClusterWare *.rpmsave* and *.rpmnew* files and perform merges, as appropriate, to carry forward the local changes. Sometimes an upgrade will save the locally modified version as *.rpmsave* and overwrite the basic file with a new version. Other times the upgrade will keep the locally modified version untouched, installing the new version as *.rpmnew*.

For example,

```bash
cd /etc/beowulf
find . -name \*rpmnew
find . -name \*rpmsave
```

and examine each such file to understand how it differs from the configuration file that existed prior to the update. You may need to merge new lines from the newer *.rpmnew* file into the existing file, or perhaps replace existing lines with new modifications. For instance, this is commonly done with */etc/beowulf/config* and */config.rpmnew*. Or you may need to merge older local modifications in */config.rpmsave* into the newly installed pristine version of the file. For instance, this is occasionally done with */etc/beowulf/fstab.rpmsave*.

Generally speaking, be careful when making changes to */etc/beowulf/config*, as mistakes may leave your cluster in a non-working state. In particular, take care when modifying the keyword entries for `interface`, `nodes`, `iprange`, and `nodeassgin`. The `kernelimage` and `node` entries are automatically managed by ClusterWare services and should not be merged.

The remaining differences are candidates for careful merging. Pay special attention to merge additions to the `bootmod-ule`, `modarg`, `server`, `libraries`, and `prestage` keyword entries. New `nodename` entries for `infiniband` or `ipmi` are offsets to each node’s IP address on the private cluster network, and these offsets may need to be altered to be compatible with your local network subnet. Also, be sure to merge differences in */config.rpmnew* comments, as those are important documentation information for future reference.

Contact Penguin Computing Customer Support if you are unsure about how to resolve particular differences, especially with */etc/beowulf/config*. 

---

1.1. Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000 21
1.1.4.2 Disable SELinux and NetworkManager

Scyld ClusterWare execution currently requires that SELinux and NetworkManager services be disabled. The install-scyld script assists in performing this disabling. Cluster administrators are strongly encouraged to always use that script to install or update ClusterWare.

1.1.4.3 Edit /etc/beowulf/conf.d/sysctl.conf as needed

The /etc/beowulf/sysctl.conf.rebuild.sh script simplifies the building of the /etc/beowulf/conf.d/sysctl.conf file, which gets pushed to each compute node's /etc/sysctl.conf at node boot time to configure the node's sysctl command behavior. Prior to Scyld ClusterWare v7.4.2, /etc/beowulf/conf.d/sysctl.conf was automatically built (if it did not currently exist) at node boot time by copying just the master node's /etc/sysctl.conf. In ClusterWare v7.4.2 and beyond, sysctl.conf.rebuild.sh instead performs the rebuild by aggregating all the *.conf files that reside in the various sysctl configuration directories. See man sysctl.conf for a list of those directories.

The script executes automatically (if /etc/beowulf/conf.d/sysctl.conf does not currently exist) when installing or updating the ClusterWare nodesscripts package. The Cluster Administrator can also manually execute the script at any time to rebuild the file from the latest contents of the master node’s various sysctl *.conf files.

After the script executes, the newly built /etc/beowulf/conf.d/sysctl.conf will subsequently be seen on node $NODE when $NODE reboots, or by executing:

```
bpcp /etc/beowulf/conf.d/sysctl.conf $NODE:/etc/sysctl.conf
bpsh $NODE sysctl -q -e -p /etc/sysctl.conf
```

NOTE: Because the script rebuilds /etc/beowulf/conf.d/sysctl.conf from the master node's sysctl *.conf files, the newly rebuilt file may contain some configuration lines that are appropriate for the master node but not for compute nodes, or there may be lines that should be added that are desired for compute nodes but are unwanted in a master node's sysctl *.conf file. Therefore, the Cluster Administrator should review the contents of /etc/beowulf/conf.d/sysctl.conf after it gets rebuilt to ensure that it contains the desired configuration lines for compute nodes. Once the file is built - whether it is subsequently modified or not - then the file is never modified by ClusterWare until and unless the Cluster Administrator manually executes sysctl.conf.rebuild.sh. If the Cluster Administrator manually deletes /etc/beowulf/conf.d/sysctl.conf, then the file gets automatically rebuilt the first time any node reboots.

1.1.4.4 Optionally reduce size of /usr/lib/locale/locale-archive

Glibc applications silently open the file /usr/lib/locale/locale-archive, which means it gets downloaded by each compute node early in a node’s startup sequence. The default RHEL7 locale-archive is about 100 MBytes in size, thus consuming significant network bandwidth and potentially causing serialization delays if numerous compute nodes attempt to concurrently boot, and consuming significant RAM filesystem space on each node. It is likely that a cluster’s users and applications do not require all the international locale data that is present in the default file. With care, the cluster administrator may choose to rebuild locale-archive with a greatly reduced set of locales and thus create a significantly smaller file. See the Administrator's Guide for details.
1.1.4.5 Optionally configure and enable compute node CPU speed/power management

Modern motherboards and processors support a degree of administrator management of CPU frequency within a range defined by the motherboard’s BIOS. Scyld ClusterWare provides the /etc/beowulf/init.d/30cpuspeed script and its associated /etc/beowulf/conf.d/cpuspeed.conf configuration file to implement this management for compute nodes. The local cluster administrator is encouraged to review the ClusterWare Administrator’s Guide Configuring CPU speed/power for Compute Nodes for details.

1.1.4.6 Optionally install a different TORQUE package

TORQUE is available in several versions: torque-4-scyld (which is the current default) and torque-4-nocupuset-scyld provide version 4, torque-5-scyld and torque-5-nocupuset-scyld provide version 5, and torque-6-scyld and torque-6-nocgroup-scyld provide version 6.

The nocpuset packages specifically disable the default cpuset functionality that optionally allows an application to constrain the movement of software threads between CPUs within a node in order to achieve optimal performance. See http://docs.adaptivecomputing.com/torque/4-1-4/help.htm#topics/3-nodes/linuxCpusetSupport.htm for details.

One, and only one, TORQUE must be installed at any one time. Since each TORQUE package specifies a list of package dependencies that should not be removed when uninstalling the existing TORQUE package, care must be taken to retain those dependencies when switching from one version of TORQUE to another. For example, to switch from torque-4-scyld to torque-4-nocupuset-scyld:

```
rpm -e --nodeps torque-4-scyld
yum install torque-4-nocupuset-scyld
```

1.1.4.7 Optionally enable job manager

The default Scyld ClusterWare installation includes two job managers: TORQUE and Slurm. TORQUE is available in several versions. See Optionally install a different TORQUE package for important details. Both Slurm and one, and only one, of these TORQUE versions must be installed on the master node, although only Slurm or one of the TORQUE versions may be enabled and executing at any one time.

To enable TORQUE: after all compute nodes are up and running, you disable Slurm (if it is currently enabled), then enable and configure TORQUE, then reboot all the compute nodes:

```
slurm-scyld.setup cluster-stop
beochkconfig 98slurm off
slurm-scyld.setup disable
beochkconfig 98torque on
torque-scyld.setup reconfigure   # when needed
torque-scyld.setup enable
torque-scyld.setup cluster-start
torque-scyld.setup status
bpctl -S all -R
```

To enable Slurm: after all compute nodes are up and running, you disable TORQUE (if it is currently enabled), then enable and configure Slurm, then reboot all the compute nodes:

```
torque-scyld.setup cluster-stop
beochkconfig 98torque off
torque-scyld.setup disable
beochkconfig 98slurm on
```

(continues on next page)
slurm-scyld.setup reconfigure # when needed
slurm-scyld.setup enable
slurm-scyld.setup cluster-start
slurm-scyld.setup status
bpctl -S all -R

See the **Administrator’s Guide** for more details about TORQUE configuration, and the **User’s Guide** for details about how to use TORQUE.

Each Slurm user must set up the PATH and LD_LIBRARY_PATH environment variables to properly access the Slurm commands. This is done automatically for users who login when the `slurm` service is running and the `pbs_server` is not running, via the `/etc/profile.d/scyld.slurm.sh` script. Alternatively, each Slurm user can manually execute `module load slurm` or can add that command line to (for example) the user’s `.bash_profile`.

See the **Administrator’s Guide** for more details about TORQUE and Slurm configuration.

### 1.1.4.8 Optionally enable TORQUE scheduler

The Scyld ClusterWare TORQUE package includes the `pbs_sched` job scheduler, which can coexist with an optionally licensed Moab job scheduler installation. The cluster administrator must choose one, and only one, job scheduler to activate. Manage the selection of `pbs_sched` vs. Moab with the `systemctl` operations: `enable`, `start`, `stop`, `disable`.

**Note:** ClusterWare has discontinued distribution of the Maui job scheduler, and cluster administrators should transition to using either `pbs_sched` or Moab.

### 1.1.4.9 Optionally enable Ganglia monitoring tool

To enable the Ganglia cluster monitoring tool,

```bash
chkconfig beostat on
systemctl enable xinetd
systemctl enable httpd
systemctl enable gmetad
```

then either reboot the master node, which automatically restarts these system services; or without rebooting, manually restart `xinetd` then start the remaining services that are not already running:

```bash
systemctl restart xinetd
systemctl start httpd
systemctl start gmetad
```

See the **Administrator’s Guide** for more details.
1.1.4.10 Optionally enable NFS locking

To enable cluster-wide NFS locking for compute node clients, edit /etc/beowulf/fstab (or the appropriate node-specific /etc/beowulf/fstab.N file(s)) to remove the default option nolock for that mountpoint. See the Administrator's Guide for more details.

1.1.4.11 Optionally adjust the size limit for locked memory

OpenIB, MVAPICH, and MVAPICH2 require an override to the limit of how much memory can be locked. Scyld ClusterWare adds a memlock override entry to /etc/security/limits.conf during a Scyld ClusterWare upgrade (if the override entry does not already exist in that file), regardless of whether or not Infiniband is present in the cluster. The new override line,

```
* - memlock unlimited
```

raises the limit to unlimited. If Infiniband is not present, then this new override line is unnecessary and may be deleted. If Infiniband is present, we recommend leaving the new unlimited line in place. If you choose to experiment with a smaller discrete value, then understand that Scyld ClusterWare MVAPICH requires a minimum of 16,384 KBytes, which means changing unlimited to 16384. If your new discrete value is too small, then MVAPICH reports a “CQ Creation” or “QP Creation” error.

1.1.4.12 Optionally increase the max number of processes per user

RHEL7 defaults to a maximum of 4096 processes per user, as specified in /etc/security/limits.d/20-nproc.conf, which contrasts with the RHEL5 default of 16,384 and the RHEL6 default of 1024. If this RHEL7 value is too low, then override the nproc entry in that file, as appropriate for your cluster workload needs. Use a discrete value, not unlimited.

1.1.4.13 Optionally enable SSHD on compute nodes

If you wish to allow users to execute MVAPICH2 applications, or to use /usr/bin/ssh or /usr/bin/scp from the master to a compute node, or from one compute node to another compute node, then you must enable sshd on compute nodes by enabling the script:

```
beochkconfig 81sshd on
```

The cluster is preconfigured to allow user root ssh access to compute nodes. The cluster administrator may wish to configure the cluster to allow ssh access for non-root users. See the Administrator's Guide for details.

1.1.4.14 Optionally allow IP Forwarding

By default, the master node does not allow IP Forwarding from compute nodes on the private cluster network to external IP addresses on the public network. If IP Forwarding is desired, then edit /etc/beowulf/config to enable the directive ipforward yes, and ensure that the file /etc/sysconfig/iptables eliminates or comments-out the default entry:

```
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```
1.1.4.15 Optionally increase the nf_conntrack table size

Certain workloads may trigger a syslog message nf_conntrack: table full, dropping packet. At cluster startup, Scyld ClusterWare insures a NAT table max size of at least 524,288. However, this max value may still be inadequate for local workloads, and the table full, dropping packet syslog messages may still occur. Use:

```
sysctl -n net.nf_conntrack_max
```

to view the current max size, then keep manually increasing the max until the syslog messages stop occurring, e.g., use:

```
sysctl -w net.nf_conntrack_max=Nmax
```

to try new Nmax values. Make this value persist across master node reboots by adding:

```
net.nf_conntrack_max=Nmax
```

to /etc/sysctl.conf.

1.1.4.16 Optionally configure vm.zone_reclaim_mode on compute nodes

Because Scyld ClusterWare compute nodes are predominantly used for High Performance Computing, versus (for example) used as file servers, we suggest that the /etc/beowulf/conf.d/sysctl.conf file contain the line:

```
vm.zone_reclaim_mode=1
```

for optimal NUMA performance. Scyld ClusterWare’s node_up script adds this line if it doesn’t already exist, but will not alter an existing vm.zone_reclaim_mode declaration in that file. If the file /etc/beowulf/conf.d/sysctl.conf does not exist, then node_up creates it by replicating the master node’s /etc/sysctl.conf, which may contain a vm.zone_reclaim_mode=N declaration that is perhaps not =1 and thus not optimal for compute nodes, even if the value is optimal for the master node. In this case, the cluster administrator should consider manually editing /etc/beowulf/conf.d/sysctl.conf to change the line to vm.zone_reclaim_mode=1.

1.1.4.17 Optionally configure automount on compute nodes

If you wish to run automount from compute nodes, you must first set up all the necessary configuration files in /etc/beowulf/conf.d/autofs/ before enabling the /etc/beowulf/init.d/50autofs script. These config files are similar to those normally found on a server in /etc/, such as /etc/auto.master, as the 50autofs script copies the files in /etc/beowulf/conf.d/autofs/ to each compute node’s /etc/.

A default /etc/beowulf/conf.d/autofs/auto.master must exist. All automount config files that are listed in that master.conf, such as /etc/auto.misc, /etc/auto.net, etc., should also reside in /etc/beowulf/conf.d/autofs/.

Node-specific config files (auto.master and related auto.* config files) may reside in /etc/beowulf/conf.d/autofs/$NODE/. Those files override the default top level /etc/beowulf/conf.d/auto.master, etc., for the specific $NODE.

The 50autofs script parses the config files as mentioned above. It creates mount point directories, installs the autofs4 kernel module, and starts automount on each booting compute node. The script exits with a warning if there are missing config files.

NOTE: This script does not validate the correctness of potential future automount mount requests (i.e., those described in the various auto.* config files). The cluster administrator should set up the config files, then enable 50autofs and reboot one or a limited number of nodes and ensure that each potential automount will function properly prior to...
rebooting all compute nodes. Common failures include naming an unknown server or attempting to mount a directory that has not been properly exported by the server. Mount failures will be syslogged in /var/log/messages.

1.1.4.18 Optionally reconfigure node names

You may declare site-specific alternative node names for cluster nodes by adding entries to /etc/beowulf/config. The syntax for a node name entry is:

```
nodename format-string [IPv4offset] [netgroup]
```

For example,

```
nodename node%N
```

allows the user to refer to node 4 using the traditional .4 name, or alternatively using names like node4 or node004. See man beowulf-config and the Administrator’s Guide for details.

1.1.5 Post-Installation Configuration Issues For Large Clusters

Larger clusters have additional issues that may require post-installation adjustments.

1.1.5.1 Optionally increase the number of nfsd threads

The default count of 8 nfsd NFS daemons may be insufficient for large clusters. One symptom of an insufficiency is a syslog message, most commonly seen when you currently boot all the cluster nodes:

```
nfsd: too many open TCP sockets, consider increasing the number of nfsd threads
```

Scyld ClusterWare automatically increases the nfsd thread count to at least one thread per compute node, with a lower-bound of eight (for =64 nodes). If this increase is insufficient, then increase the thread count (e.g., to 16) by executing: echo 16 > /proc/fs/nfsd/threads Ideally, the chosen thread count should be sufficient to eliminate the syslog complaints, but not significantly higher, as that would unnecessarily consume system resources. One approach is to repeatedly double the thread count until the syslog error messages stop occurring, then make the satisfactory value N persistent across master node reboots by creating the file /etc/sysconfig/nfs, if it does not already exist, and adding to it an entry of the form: RPCNFSDCOUNT=N A value N of 1.5x to 2x the number of nodes is probably adequate, although perhaps excessive. See the Administrator’s Guide for a more detailed discussion of NFS configuration.

1.1.5.2 Optionally increase the max number of processID values

The kernel defaults to using a maximum of 32,768 processID values. Scyld ClusterWare automatically increases this default to 98,304 [= 3*32768], which likely is adequate for small- to medium-size clusters and which keeps pid values at a familiar 5-column width maximum. Because BProc manages a common process space across the cluster, even the increase to 98,304 may be insufficient for very large clusters and/or workloads that create large numbers of concurrent processes. The cluster administrator can increase the value further by using the sysctl command, e.g.,

```
sysctl -w kernel.pid_max=N
```

directs the kernel to use pid values up to N. The kernel (and BProc) supports an upperbound of 4,194,304 [= (4*1024*1024)]. To set a value N that persists across master node reboots, add an entry

```
kernels.pid_max=N
```

1.1. Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000
to /etc/sysctl.conf. NOTE: Even though /etc/beowulf/conf.d/sysctl.conf is referenced by the sysctl command that executes at boot time on each node, any kernel.pid_max entry in that file is ignored. The master node's kernel.pid_max value prevails cluster-wide for Scyld ClusterWare nodes.

1.1.5.3 Optionally increase the max number of open files

RHEL7 defaults to a maximum of 1024 concurrently open files. This value may be too low for large clusters. The cluster administrator can add a nofile override entry to /etc/security/limits.conf to specify a larger value. Caution: for nofile, use only a numeric upperbound value, never unlimited, as that will result in being unable to login.

1.1.5.4 Issues with Ganglia

The Ganglia cluster monitoring tool may fail for large clusters. If the /var/log/httpd/error_log shows a fatal error of the form PHP Fatal error: Allowed memory size of 8388608 bytes exhausted, then edit the file /etc/php.ini to increase the memory_limit parameter. The default is memory_limit = 8M can be safely doubled and re-doubled until the error goes away.

1.1.6 Post-Installation Release of Updated Packages

From time to time, Penguin Computing releases updated Scyld ClusterWare 7 rpms to track Red Hat kernel security or bug fix errata, or to fix Scyld ClusterWare problems or to introduce enhancements. Download the latest version of the Scyld ClusterWare 7 Release Notes from https://www.penguincomputing.com/support/documentation to ensure you have the latest guidance before updating your cluster.

First check for the availability of updated rpms:

```
yum check-update
```

and ascertain if the base distribution and/or Scyld ClusterWare would update to a newer kernel, or even more significantly to a new major-minor release. Upgrading the kernel will require updating, perhaps even rebuilding, any 3rd-party drivers that are installed and linked against the current kernel, and you should be prepared to do that if you proceed with the updates. Updating to a newer major-minor release may also affect 3rd-party applications that are validated only for the current base distribution release.

In general, if you choose to update software, then you should use:

```
install-scyld -u
```

and update all available packages.

If your cluster uses Panasas storage, then before updating Scyld ClusterWare you must ensure that a Panasas kernel module is available that matches the Scyld ClusterWare kernel that will be installed. See Important for clusters using Panasas storage for more information.
1.1.7 Notable Feature Enhancements And Bug Fixes

1.1.7.1 v7.9.16 - August 19, 2022

1. The base kernel updates to 3.10.0-1160.76.1.el7.7916g0000. See https://access.redhat.com/errata/RHSA-2022:5937 for details.

1.1.7.2 v7.9.15 - July 8, 2022

1. The base kernel updates to 3.10.0-1160.71.1.el7.7915g0000. See https://access.redhat.com/errata/RHSA-2022:5232 for details.

1.1.7.3 v7.9.14 - June 1, 2022

1. The base kernel updates to 3.10.0-1160.66.1.el7.7914g0000. See https://access.redhat.com/errata/RHSA-2022:4642 for details.

2. The bproc-libs-libc package updates /lib64/scyld/files to glibc-2.17-326.e17_9 to maintain bproc command compatibility with the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHBA-2022:4641.html for details about the base distribution’s glibc.

1.1.7.4 v7.9.13 - April 13, 2022

1. The base kernel updates to 3.10.0-1160.62.1.el7.7913g0000. See https://access.redhat.com/errata/RHSA-2022:1198 for details.

1.1.7.5 v7.9.12 - March 2, 2022

1. The base kernel updates to 3.10.0-1160.59.1.el7.7912g0000. See https://access.redhat.com/errata/RHSA-2022:0620 for details.

1.1.7.6 v7.9.11 - January 20, 2022

1. The base kernel updates to 3.10.0-1160.53.1.el7.7911g0000. See https://access.redhat.com/errata/RHSA-2022:0063 for details.

1.1.7.7 v7.9.10 - December 6, 2021

1. The base kernel updates to 3.10.0-1160.49.1.el7.7910g0000. See https://access.redhat.com/errata/RHSA-2021:4777 for details.

2. The bproc-libs-libc package updates /lib64/scyld/files to glibc-2.17-325.e17_9 to maintain bproc command compatibility with the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHBA-2021:3803.html for details about the base distribution’s glibc.
1.1.7.8 v7.9.9 - October 20, 2021

1. The base kernel updates to 3.10.0-1160.45.1.el7.799g0000. See https://access.redhat.com/errata/RHSA-2021:3801 for details.

1.1.7.9 v7.9.8 - September 15, 2021

1. The base kernel updates to 3.10.0-1160.42.2.el7.798g0000. See https://access.redhat.com/errata/RHSA-2021:3327 and https://access.redhat.com/errata/RHSA-2021:3438 for details.

1.1.7.10 v7.9.7 - July 27, 2021

1. The base kernel updates to 3.10.0-1160.36.2.el7.797g0000. See https://access.redhat.com/errata/RHSA-2021:2734 for details.

1.1.7.11 v7.9.6 - June 21, 2021

1. The base kernel updates to 3.10.0-1160.31.1.el7.796g0000. See https://access.redhat.com/errata/RHSA-2021:2314 for details.

1.1.7.12 v7.9.5 - May 26, 2021

1. The base kernel updates to 3.10.0-1160.25.1.el7.795g0000. See https://access.redhat.com/errata/RHBA-2021:1397 for details.

2. The bproc-libs-libc package updates /lib64/scyld/ files to glibc-2.17-324.el7_9 to maintain bproc command compatibility with the the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHSA-2021:1392.html for details about the base distribution’s glibc.

1.1.7.13 v7.9.4 - April 27, 2021

1. The base kernel updates to 3.10.0-1160.24.1.el7.794g0000. See https://access.redhat.com/errata/RHSA-2021:1071 for details.

1.1.7.14 v7.9.3 - April 7, 2021

1. The base kernel updates to 3.10.0-1160.21.1.el7.793g0000. See https://access.redhat.com/errata/RHSA-2021:0856 for details.

2. The bproc-libs-libc package updates /lib64/scyld/ files to glibc-2.17-323.el7_9 to maintain bproc command compatibility with the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHSA-2021:0439.html for details about the base distribution’s glibc.

3. The Slurm job manager updates to version 20.11.5, derived from https://slurm.schedmd.com. See the ClusterWare User's Guide SLURM Release Information for details. The Scyld openmpi packages are compatible with this Slurm /opt/scyld/slurm/lib64/libslurm.so.36 library.
1.1.7.15 v7.9.2 - February 11, 2021

1. The base kernel updates to 3.10.0-1160.15.2.el7.792g0000. See https://access.redhat.com/errata/RHSA-2021:0336 for details.

2. The bproc-libs-libc package updates /lib64/scyld/libc to glibc-2.17-322.el7_9 to maintain bproc command compatibility with the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHSA-2021:0348.html for details about the base distribution’s glibc.

1.1.7.16 v7.9.1 - January 6, 2021

1. The base kernel updates to 3.10.0-1160.11.1.el7.791g0000. See https://access.redhat.com/errata/RHSA-2020:5437 for details.

2. ClusterWare now distributes openmpi-4.1-scyld packages, which are initially version 4.1.0. Installation of openmpi-4.1 does not affect any other OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-44, Intel version 2020.1.217, and PGI version 20.4. See the ClusterWare User’s Guide OpenMPI Release Information for details.

1.1.7.17 v7.9.0 - December 9, 2020

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 9 and CentOS 7.9 base distribution environments. Cluster administrators and users are encouraged to visit https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux and read the RHEL 7.9 Release Notes in order to understand the differences between the 7.9 base distribution versus earlier base distributions.


3. The Slurm job manager updates to version 20.11.0, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details. The Scyld openmpi packages are compatible with the new Slurm 20.11.0 library.


5. ClusterWare no longer distributes openmpi-1.8 and mpich2 because they no longer work with PGI version 20.

6. All other Scyld openmpi, mpich, and mvapich2 packages have been rebuilt for 790g0000 using Gnu version 4.8.5-44, Intel version 2020.1.217, and PGI version 20.4.

1.1.7.18 v7.8.5 - September 22, 2020

1. The base kernel updates to 3.10.0-1127.19.1.el7.785g0000. See https://access.redhat.com/errata/RHBA-2020:3528 for details.

2. The Slurm job manager updates to version 19.05.7, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.


4. The openmpi-4.0-scyld packages update to version 4.0.5. Updating openmpi-4.0 only affects the version 4.0.x series. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-39, Intel version 19.1.1.217, and PGI version 20.4 compiler families.

1. Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000
5. MVAPICH2 updates to version 2.3.4 for the mvapich2-psm-scyld and mvapich2-scyld packages. See *Installing and managing concurrent versions of packages* for general issues about supporting multiple concurrent versions. This software suite derives from [http://mvapich.cse.ohio-state.edu/](http://mvapich.cse.ohio-state.edu/). The libraries were built with Gnu version 4.8.5-39, Intel version 19.1.1.217, and PGI version 20.4 compiler families. See the ClusterWare User’s Guide *MVAPICH2 Release Information* for details.

**1.1.7.19 v7.8.4 - August 5, 2020**

1. The base kernel updates to 3.10.0-1127.18.2.el7.784g0000. See [https://access.redhat.com/errata/RHSA-2020:3220](https://access.redhat.com/errata/RHSA-2020:3220) for details.

2. Enhance install-sclyld to version 1.45 for improved detection of an install or update that leaves the master node with no ClusterWare kernel and associated bproc kernel modules; automatic restart of install-sclyld after a runtime update to a newer version; and terser output.

**1.1.7.20 v7.8.3 - July 7, 2020**


**1.1.7.21 v7.8.2 - June 17, 2020**

1. The base kernel updates to 3.10.0-1127.10.1.el7.782g0000. See [https://access.redhat.com/errata/RHBA-2020:2355](https://access.redhat.com/errata/RHBA-2020:2355) for details.

2. Update beoserv to version 2.9.5: fix a problem with config file ‘kernelcommandline’ specifying node number or range.

**1.1.7.22 v7.8.1 - May 27, 2020**

1. The base kernel updates to 3.10.0-1127.8.2.el7.781g0000. See [https://access.redhat.com/errata/RHSA-2020:2082](https://access.redhat.com/errata/RHSA-2020:2082) for details.

**1.1.7.23 v7.8.0 - May 6, 2020**

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 8 and CentOS 7.8 base distribution environments. Cluster administrators and users are encouraged to visit [https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux](https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux) and read the RHEL 7.8 Release Notes in order to understand the differences between the 7.8 base distribution versus earlier base distributions.

2. The base kernel updates to 3.10.0-1127.el7.780g0000. See [https://access.redhat.com/errata/RHSA-2020:1016](https://access.redhat.com/errata/RHSA-2020:1016) for details.

3. Update bproc-libs-libc updates to include the /lib64/scyld/libc-2.17.so (i.e., libc.so.6) that is compatible with the base distribution /lib64/libc-2.17.so.

4. The Slurm job manager updates to version 20.02.1, derived from [https://slurm.schedmd.com](https://slurm.schedmd.com), and is available if desired in the ClusterWare updates.next repo, together in that repo with the various openmpi-*-scyld packages that have been rebuilt for consistency with the new Slurm library for v20.02.1. See the ClusterWare User’s Guide *SLURM Release Information* for details.
5. The openmpi-4.0-sclyld packages update to version 4.0.3. Updating openmpi-4.0 only affects the version 4.0.x series. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-39, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.

1.1.7.24 v7.7.5 - April 1, 2020

1. The base kernel updates to 3.10.0-1062.18.1.el7.775g0000. See https://access.redhat.com/errata/RHSA-2020: 0834 for details.
2. Update beosi to version 1.64: capture all modified files in packages distributed by ClusterWare.
3. Update beobootutils to version 1.4.44: add /sbin/ldconfig to the rootfs to support installing Python v3.6 directly on compute nodes.

1.1.7.25 v7.7.4 - February 20, 2020

1. The base kernel updates to 3.10.0-1062.12.1.el7.774g0000. See https://access.redhat.com/errata/RHSA-2020: 0374 for details.
2. Update beoserv to version 2.9.4: fix a problem doing UEFI DHCP of Gigabyte motherboards.
3. Update beobootutils to version 1.4.43: broaden permissions of /var/beowulf/boot/pxelinux.cfg/default to allow beoserv v2.9.4 to read as a non-root user.
4. Update beostat to version 0.9.4: fix a problem sending periodic stats from compute nodes to master nodes for large per-node CPU counts and mismatched MTU values.
5. The Slurm job manager updates to version 19.05.5, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.

1.1.7.26 v7.7.3 - December 13, 2019


1.1.7.27 v7.7.2 - November 27, 2019

2. Update beoserv to version 2.9.3, which aborts at ClusterWare startup if it detects a duplicate MAC address in the /etc/beowulf/config list of node entries.
3. The Slurm job manager updates to version 19.05.3-2, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.
4. The openmpi-4.0-sclyld packages update to version 4.0.2. Updating openmpi-4.0 does not affect any other OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-39, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.
1.1.7.28 v7.7.1 - October 18, 2019

1. The base kernel updates to 3.10.0-1062.1.2.el7.771g0000. See https://access.redhat.com/errata/RHSA-2019:2829 for details.

2. Update beoserv to version 2.9.0 to support UEFI client requests for four additional files distributed in the base distribution’s grub2-efi-x64-modules package, which is now an additional beoserv dependency at install/update time.

1.1.7.29 v7.7.0 - September 24, 2019

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 7 and CentOS 7.7 base distribution environments. Cluster administrators and users are encouraged to visit https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux and read the RHEL 7.7 Release Notes in order to understand the differences between the 7.7 base distribution versus earlier base distributions.


3. The bproc-libs-libc package updates /lib64/scyld/files to glibc-2.17-292.el7 to maintain bproc command compatibility with the the latest base distribution RHEL/CentOS 7.9 glibc. See Issues with Scyld ClusterWare process migration in heterogeneous clusters for more details about why this is done, and see https://access.redhat.com/errata/RHSA-2019:2118.html for details about the base distribution’s glibc.

4. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-957.1.3.el7.760g0000.x86_64 kernel. This panfs works properly in a 3.10.0-1062.1.1.el7.770g0000 environment (and in future v7.7.x environments, until noted otherwise) using weak-updates linking.

5. Scyld ClusterWare v7.9.16 discontinues distribution of the Maui job scheduler for TORQUE, and TORQUE instead includes the pbs_sched scheduler. Alternatively, the local cluster may use the Moab scheduler, which is separately licensed by Adaptive Computing, Inc.

1.1.7.30 v7.6.4 - August 16, 2019

1. The base kernel is 3.10.0-957.27.2.el7.764g0000. See https://access.redhat.com/errata/RHSA-2019:1873 for details.


4. The openmpi-3.1-scyld packages update to version 3.1.4, openmpi-3.0-scyld updates to version 3.0.4, and openmpi-2.1-scyld updates to version 2.1.6. The remaining openmpi-*-scyld packages have been rebuilt for compatibility with the updated Slurm.
1.1.7.31 v7.6.3 - July 10, 2019


2. Update beoserv to version 2.8.10 to tweak the multi-master behavior of boot ordering.

1.1.7.32 v7.6.2 - June 11, 2019


2. Update beoserv to version 2.8.9 to fix two PXEboot problems:
   - The pxelinux.0 loader v6.03 could not TFTP /ldlinux.c32.
   - beoserv could not find the UEFI grubx64.efi loader in a Red Hat RHEL (i.e., not CentOS) base distribution environment.


1.1.7.33 v7.6.1 - April 8, 2019


2. Update beoserv to version 2.8.7 to fix proper boot ordering in a multi-master environment.

3. The Slurm job manager updates to version 18.08.6-2, derived from https://slurm.schedmd.com. See the ClusterWare User's Guide SLURM Release Information for details.

4. ClusterWare now distributes openmpi-4.0-scycld packages, which are initially version 4.0.1. Installation of openmpi-4.0 does not affect any other OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-36, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User's Guide OpenMPI Release Information for details.

5. MVAPICH2 updates to version 2.3.1 for the mvapich2-psm-scycld and mvapich2-scycld packages. See Installing and managing concurrent versions of packages for general issues about supporting multiple concurrent versions. This software suite derives from http://mvapich.cse.ohio-state.edu/. See the ClusterWare User's Guide MVAPICH2 Release Information for details.

1.1.7.34 v7.6.0 - December 14, 2018

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 6 and CentOS 7.6 base distribution environments. Cluster administrators and users are encouraged to visit https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux and read the RHEL 7.6 Release Notes in order to understand the differences between the 7.6 base distribution versus earlier base distributions.


3. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.11.6.el7.754g0000.x86_64 kernel. This panfs works properly in a 3.10.0-957.1.3.el7.760g0000 environment using weak-updates linking.
4. The Slurm job manager updates to version 18.08.4, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details. (Note: the openmpi-* packages have been rebuilt for compatibility with the new Slurm.)

5. The openmpi-3.1-scylld packages update to version 3.1.3, which by default update and replace only earlier version 3.1 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-36, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.


1.1.7.35 v7.5.4 - October 2, 2018


2. This kernel (and newer) builds-in firmware for various network controllers: bnx2, cxgb3, hfi1. See Issues with bootmodule firmware for details. (Enhancement backported to v7.4.6)

3. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.11.6.el7.754g0000.x86_64 kernel. This panfs works properly in a 3.10.0-862.14.4.el7.754g0001 environment using weak-updates linking.

4. Intel-processor nodes that do not support invpcid suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.

5. IMPORTANT: ClusterWare v7.5.4 includes OpenMPI psm2 PGI rpms, despite an incompatibility in the combination of PGI compilers, OpenMPI psm2 software, and Intel(r) Omni-Path Architecture (OPA) software. ClusterWare v7.5.x OpenMPI psm2 gnu and intel rpms do work for OPA. Execute openmpi psm2 PGI programs using execstack -c <binary> as a workaround for this incompatibility.

6. Fix bpsh command hangs and misbehavior that was occasionally seen on large clusters (e.g., >300 nodes). (Fix backported to v7.4.6)

7. Enhance the beoserv daemon to log more information for a TFTP client download hang infrequently seen during an EFI PXEboot. (Enhancement backported to v7.4.6)


10. The openmpi-2.1-scylld packages update to version 2.1.5, which by default update and replace only earlier version 2.1 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-28, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.

11. The openmpi-3.1-scylld packages update to version 3.1.2, which by default update and replace only earlier version 3.1 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-28, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.
12. Installing or updating the v7.5.4 (and later) `beonss` rpm relocates `/etc/beowulf/nsswitch.conf` to `/etc/beowulf/conf.d/nsswitch.conf`, thus moving it to where the other ClusterWare .conf files reside. The `node_up` script similarly relocates any optional existing compute node-specific `/etc/beowulf/nsswitch.conf.<nodenum>` file to `/etc/beowulf/conf.d/` if encountered when booting node `<nodenum>`.

### 1.1.7.36 v7.5.3 - July 27, 2018

1. The base kernel is 3.10.0-862.9.1.el7.753g0000. See https://access.redhat.com/errata/RHBA-2018:2198.html for details.
2. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.9.1.el7.753g0000 kernel.
3. Intel-processor nodes that do not support `invpcid` suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.
4. **IMPORTANT**: ClusterWare v7.5.0 to v7.5.3 do not include OpenMPI psm2 PGI rpms because there is an incompatibility in the combination of PGI compilers, OpenMPI psm2 software, and Intel(r) Omni-Path Architecture (OPA) software. ClusterWare v7.5.x OpenMPI psm2 gnu and intel rpms do work for OPA. If OpenMPI psm2 PGI support for OPA hardware is required for your cluster workloads, then you should not update beyond ClusterWare v7.4.6, or update to v7.5.4 and execute openmpi psm2 PGI programs using `execstack -c <binary>`.
5. The Slurm job manager updates to version 17.11.8, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.
6. MVAPICH2 updates to version 2.3 for the mvapich2-psm-scyld and mvapich2-scyld packages. See Installing and managing concurrent versions of packages for general issues about supporting multiple concurrent versions. This software suite derives from http://mvapich.cse.ohio-state.edu/. See the ClusterWare User’s Guide MVAPICH2 Release Information for details.
7. Fix a bproc problem with the setup of the `/bpfs` pseudo filesystem that caused (for example) the `beomap` command to not detect any compute nodes.

### 1.1.7.37 v7.5.2 - July 14, 2018

2. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.6.3.el7.752g0000 kernel.
3. Intel-processor nodes that do not support `invpcid` suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.
4. **IMPORTANT**: ClusterWare v7.5.0 to v7.5.3 do not include OpenMPI psm2 PGI rpms because there is an incompatibility in the combination of PGI compilers, OpenMPI psm2 software, and Intel(r) Omni-Path Architecture (OPA) software. ClusterWare v7.5.x OpenMPI psm2 gnu and intel rpms do work for OPA. If OpenMPI psm2 PGI support for OPA hardware is required for your cluster workloads, then you should not update beyond ClusterWare v7.4.6, or update to v7.5.4 and execute openmpi psm2 PGI programs using `execstack -c <binary>`.
5. **IMPORTANT**: ClusterWare v7.5.2 does not yet include a new version of the optional `beoweb` rpm. If an earlier beoweb is currently installed and you are updating to ClusterWare v7.5.2, then beoweb will continue to work. However, beoweb is not currently available for a fresh install of ClusterWare v7.5.2.
6. The bproc `filecache` functionality now properly downloads files for 32-bit applications. ClusterWare still does not support `bprsh` of a 32-bit executable binary, although `bprsh` does work properly to execute a binary that already exists (or is accessible via NFS) on the target node.
7. The Slurm job manager updates to version 17.11.6, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.

8. ClusterWare now distributes openmpi-3.1-scyld packages, which are initially version 3.1.1. Installation of openmpi-3.1 does not affect any other OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-28, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.

9. The openmpi-3.0-scyld packages update to version 3.0.2, which by default update and replace only earlier version 3.0 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. The libraries were built with Gnu version 4.8.5-28, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. See the ClusterWare User’s Guide OpenMPI Release Information for details.

10. Singularity updates to version 2.5.2. See https://www.sylabs.io/docs/ and the ClusterWare User’s Guide Using Singularity for details.

1.1.7.38 v7.5.1 - June 4, 2018

1. The base kernel is 3.10.0-862.3.2.el7.751g0000. See https://access.redhat.com/errata/RHSA-2018:1629.html for details.

2. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.3.2.el7.751g0000 kernel.

3. Intel-processor nodes that do not support invpcid suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.

4. IMPORTANT: ClusterWare v7.5.0 to v7.5.3 do not include OpenMPI psm2 PGI rpms because there is an incompatibility in the combination of PGI compilers, OpenMPI psm2 software, and Intel(r) Omni-Path Architecture (OPA) software. ClusterWare v7.5.x OpenMPI psm2 gnu and intel rpms do work for OPA. If OpenMPI psm2 PGI support for OPA hardware is required for your cluster workloads, then you should not update beyond ClusterWare v7.4.6, or update to v7.5.4 and execute openmpi psm2 PGI programs using execstack -c <binary>.

5. IMPORTANT: ClusterWare v7.5.1 does not yet include a new version of the optional beoweb rpm. If an earlier beoweb is currently installed and you are updating to ClusterWare v7.5.1, then beoweb will continue to work. However, beoweb is not currently available for a fresh install of ClusterWare v7.5.1.

1.1.7.39 v7.5.0 - June 4, 2018

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 5 and CentOS 7.5 base distribution environments. Cluster administrators and users are encouraged to visit https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux and read the RHEL 7.5 Release Notes in order to understand the differences between the 7.5 base distribution versus earlier base distributions.


3. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-862.2.3.el7.750g0000 kernel.

4. Intel-processor nodes that do not support invpcid suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.

5. IMPORTANT: ClusterWare v7.5.0 to v7.5.3 do not include OpenMPI psm2 PGI rpms because there is an incompatibility in the combination of PGI compilers, OpenMPI psm2 software, and Intel(r) Omni-Path Architecture (OPA) software. ClusterWare v7.5.x OpenMPI psm2 gnu and intel rpms do work for OPA. If OpenMPI psm2
PGI support for OPA hardware is required for your cluster workloads, then you should not update beyond ClusterWare v7.4.6, or update to v7.5.4 and execute openmpi psm2 PGI programs using execstack -c <binary>.

6. **IMPORTANT:** ClusterWare v7.5.0 does not yet include a new version of the optional beoweb rpm. If an earlier beoweb is currently installed and you are updating to ClusterWare v7.5.0, then beoweb will continue to work. However, beoweb is not currently available for a fresh install of ClusterWare v7.5.0.

7. Update /lib64/scyld/ files for compatibility with the base distribution’s /lib64/ files.

8. Fix a bpmaster segfault that occurs when the /etc/beowulf/config file’s nodes and imprange upperbound IP address contradict each other. (Fix backported to v7.4.6)

9. The bpcp command now supports a new -a option, which specifies to copy the local source file(s) to every up compute node. (Enhancement backported to v7.4.6)


11. The openmpi-3.0-scyld packages update to version 3.0.1, which by default update and replace only earlier version 3.0 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. See the ClusterWare User’s Guide OpenMPI Release Information for details.

12. Distribute a new clusterware-docs rpm that replaces the earlier scyld-doc rpms. It installs the ClusterWare documentation in the form of a single combined clusterware-docs.pdf PDF file and a single combined HTML package to /var/www/html for local access, and installs ClusterWare manpages. The https://www.penguincomputing.com/support/documentation web page now contains the same PDF and HTML choices for viewing ClusterWare documentation, vs. the earlier set of individual PDF files. A combined document simplifies searching and allows for full cross-referencing between the individual documents.

13. The net-snmp-scyld package is deprecated and removed from the ClusterWare distribution.

### 1.1.7.40 v7.4.6 - October 3, 2018

1. The base kernel is 3.10.0-693.21.1.el7.746g0000. See https://access.redhat.com/errata/RHSA-2018:0395.html for details.

2. This kernel builds-in firmware for various network controllers: bnx2, cxgb3, hfi1. See Issues with bootmodule firmware for details. (Backport from v7.5.4)

3. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-693.21.1.el7.745g0000 kernel. If this rpm is not yet installed, then download and install. The v7.4.5 panfs.ko is compatible with the v7.4.6 kernel. Then copy the panfs kernel module to v7.4.6 after installing the v7.4.6 kernel:

   ```bash
cd /lib/modules/3.10.0-693.21.1.el7.746g0000.x86_64
mkdir -p extra/panfs
 cp -a .../3.10.0-693.21.1.el7.745g0000.x86_64/extra/panfs/panfs.ko 
    extra/panfs/panfs.ko
depmod -a
```

4. Intel-processor nodes that do not support invpcid will suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.

5. Fix bpsh command hangs and misbehavior that was occasionally seen on large clusters (e.g., >300 nodes). (Backport from v7.5.4)

6. Fix a bpmaster segfault that occurs when the /etc/beowulf/config file’s nodes and imprange upperbound IP address contradict each other. (Backport from v7.5.0)

39
7. The `bpcp` command now supports a new `-a` option, which specifies to copy the local source file(s) to every *up* compute node. (Backport from v7.5.0)

8. The `bproc filecache` functionality now properly downloads files for 32-bit applications. ClusterWare still does not support `bpsh` of a 32-bit executable binary, although `bprsh` does work properly to execute a binary that already exists (or is accessible via NFS) on the target node. (Backport from v7.5.2)

9. Enhance the `beoserv` daemon to log more information for a TFTP client download hang infrequently seen during an EFI PXE boot. (Backport from v7.5.4)

1.1.7.41 v7.4.5 - March 23, 2018

1. The base kernel is 3.10.0-693.21.1.el7.745g0000. See https://access.redhat.com/errata/RHSA-2018:0395.html for details.

2. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-693.21.1.el7.745g0000 kernel.

3. Intel-processor nodes that do not support *invpcid* will suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-*invpcid* old Intel nodes for a workaround.

4. Fix the `beoserv` daemon, which executes on the master node, to properly service DHCP requests from motherboard BMCs (Baseboard Management Controllers) and CMCs (Chassis Management Controllers). This functionality broke beginning with the v7.3.6 release. Also fixes problems for some nodes attempting to PXE boot using EFI mode.

5. Fix the `beoclient` daemon, which executes on a booting compute node as the *init* process, to properly generate and print to the node’s console a message that clearly explains that the daemon cannot find an appropriate network driver for communication back to the master node. The most common reason for that failure is because that driver has not been mentioned as a “bootmodule” in `/etc/beowulf/config`.

6. The `openmpi-2.1-scyld` packages update to version 2.1.3, which by default update and replace only earlier version 2.1 packages and do not affect any other installed OpenMPI version. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. See the ClusterWare User’s Guide OpenMPI Release Information for details.


8. Singularity updates to version 2.4.5. See https://www.sylabs.io/docs/ and the ClusterWare User’s Guide Using Singularity for details.

9. The `install-scyld` script now appends its logging messages to `/etc/beowulf/install-scyld.log`, instead of writing a logging file to the current working directory. This file is now being backed up into `/etc/beowulf/backups/` for every clusterware service start, restart, and reload, just as various other `/etc/beowulf/` configuration files have been saved.

1.1.7.42 v7.4.4 - February 8, 2018

1. The base kernel is 3.10.0-693.17.1.el7.744g0000. See https://access.redhat.com/errata/RHSA-2018:0151.html for details.

2. Intel-processor nodes that do not support *invpcid* will suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-*invpcid* old Intel nodes for a workaround.

3. A `systemctl start clusterware`, `systemctl restart clusterware`, or `systemctl reload clusterware` rebuilds the initrd (initial root directory) file for compute nodes. That file now contains
the latest Intel and AMD CPU microcode files found in /lib/firmware/intel-ucode/ and /lib/firmware/amd-ucode/. A booting node’s kernel chooses an appropriate microcode file (if available) to dynamically reload every CPU’s microcode.


5. Singularity updates to version 2.4.2. See https://www.sylabs.io/docs/ and the ClusterWare User’s Guide Using Singularity for details.

1.1.7.43 v7.4.3 - January 11, 2018

1. The base kernel is 3.10.0-693.11.6.el7.743g0000. See https://access.redhat.com/errata/RHSA-2018:0007.html for details.

2. Intel-processor nodes that do not support invpcid will suffer a kernel panic when used as either a compute node or a master node. See Kernel panic using non-invpcid old Intel nodes for a workaround.

3. This kernel fixes the security issues noted in https://access.redhat.com/security/cve/CVE-2017-5753 and https://access.redhat.com/security/cve/CVE-2017-5715, which affects both Intel and AMD x86_64, and https://access.redhat.com/security/cve/CVE-2017-5754, which affects only Intel x86_64.

These fixes may result in performance degradation, especially for applications that perform high rates of system calls and interrupts. See https://access.redhat.com/security/vulnerabilities/speculativeexecution for more information, and https://access.redhat.com/articles/3311301 for extraordinary methods to disable these security fixes and thereby expose the cluster to security vulnerabilities.

1.1.7.44 v7.4.2 - December 26, 2017

1. The base kernel is 3.10.0-693.11.1.el7.742g0000. See https://access.redhat.com/errata/RHSA-2017:3315.html for details.

2. Fix a bproc problem involving a master node that employs the latest Intel microarchitecture (codenamed “Skylake”, succeeding “Broadwell”) resulting in a kernel panic in task_packer_save_cpu() when booting a compute node.


4. Fix a flaw in the install-scyld script that left /etc/yum.conf with too-restrictive permissions: 0600 instead of the proper 0644. Improper permissions breaks commands such as yum grouplist when executed by non-root users.

5. The /etc/beowulf/init.d/95sudo script updates to the new /usr/libexec/sudo/ location of sudoers. so.

6. The Slurm job manager updates to version 17.11.0, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.

7. The openmpi-2.0-scyld packages update to version 2.0.4, which by default update and replace only earlier version 2.0 packages and do not affect any installed OpenMPI version 1.10 and earlier packages. See Installing and managing concurrent versions of packages for general issues about supporting multiple concurrent versions. The libraries were built with Gnu version 4.8.5-16, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. See the ClusterWare User’s Guide OpenMPI Release Information for details.


1.1. Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000
9. The podtools and beoweb packages are now optional and thus not installed by default. `install-scyld -u` will continue to update already-installed packages.

10. Introduce the `/etc/beowulf/sysctl.conf.rebuild.sh` script that rebuilds `/etc/beowulf/conf.d/sysctl.conf`. See `Edit /etc/beowulf/conf.d/sysctl.conf as needed` for details.

**1.1.7.45 v7.4.1 - October 31, 2017**

1. The base kernel is 3.10.0-693.5.2.el7.740g0000. See [https://access.redhat.com/errata/RHSA-2017:2930.html](https://access.redhat.com/errata/RHSA-2017:2930.html) for details.


**1.1.7.46 v7.4.0 - October 11, 2017**

1. Fix a bproc problem that was introduced in ClusterWare release v7.3.5 that breaks bpcp when attempting to copy a file or files from a compute node to another compute node or to the master node.

**1.1.7.47 v7.4.0 - October 6, 2017**

1. This is the first ClusterWare release that is compatible with the Red Hat RHEL7 Update 5 and CentOS 7.4 base distribution environments. Cluster administrators and users are encouraged to visit [https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux](https://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux) and read the RHEL 7.4 Release Notes in order to understand the differences between the 7.4 base distribution versus earlier base distributions.


3. Fix a master node kernel panic that occurs when the master reboots or occasionally when executing `sync`.

4. ClusterWare now supports client PXEboot for EFI mode, in addition to the current Legacy mode. See the Administrator’s Guide for details.

5. Fix a problem with bpcp and `bdate` aborting with a syslog message `trap invalid opcode` when executing on a “newer” CPU model and interacting with an “older” model that doesn’t support the full instruction set used by the “newer” model, e.g., “newer” supports string instructions and “older” does not.

6. The various openmpi, mpich2, and mpich-scyld packages’ libraries are built with Gnu version 4.8.5-16 from the base distribution, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families.

7. A new `/etc/beowulf/init.d/60lvmluks` script starts the `lvmetad` daemon on compute nodes to manage encrypted logical volumes.

8. The Slurm job manager updates to version 17.02.7, derived from [https://slurm.schedmd.com](https://slurm.schedmd.com). See the ClusterWare User’s Guide SLURM Release Information for details.

9. Scyld ClusterWare now distributes openmpi-3.0-scyld packages, which are initially version 3.0.0. Installation of openmpi-3.0 does not affect any earlier OpenMPI version. The openmpi-2.1-scyld packages update to version 2.1.2, which by default update and replace only earlier version 2.1 packages and do not affect any installed OpenMPI version 2.0 and earlier packages. ClusterWare releases of OpenMPI derive from [https://www.open-mpi.org](https://www.open-mpi.org). See the ClusterWare User’s Guide OpenMPI Release Information for details.
10. Each ClusterWare compute node employs a custom syslog server daemon that forwards the node’s syslog messages to the central syslog server – typically the master node’s rsyslogd daemon – which writes those messages to /var/log/messages. Previously, many compute node syslog messages were written to /var/log/messages containing a redundant date-time string, which is both unnecessary and violates the RFC 3164 format standard. The ClusterWare compute node server daemon now strips out that redundant date-time string before forwarding a message to the master node’s rsyslogd. If for some reason a local cluster administrator wishes to revert to the previous behavior, then edit the /etc/beowulf/config’s kernelcommandline directive to add legacy_syslog=1.

1.1.7.48 v7.3.7 - October 13, 2017

1. Fix a bproc problem that was introduced in ClusterWare release v7.3.5 that breaks bpcp when attempting to copy a file or files from a compute node to another compute node or to the master node.

1.1.7.49 v7.3.6 - August 7, 2017


2. For Panasas support, search the Panasas website (see Important for clusters using Panasas storage for details) for an rpm that matches the 3.10.0-514.26.2.el7.736g0000 kernel.

1.1.7.50 v7.3.5 - June 28, 2017


2. Fix a memory leak that occurs when executing bpsh, bprsh, and bpcp commands. This memory leak could eventually lead to an out-of-memory (OOM) kernel panic.


5. The Slurm job manager updates to version 17.02.5, derived from https://slurm.schedmd.com. See the ClusterWare User’s Guide SLURM Release Information for details.

1.1.7.51 v7.3.4 - June 16, 2017


4. The openmpi-2.1-scyld packages update to version 2.1.1, which by default update and replace only earlier version 2.1 packages and do not affect any installed OpenMPI version 2.0 and earlier packages. The openmpi-2.0-scyld packages update to version 2.0.3, which by default update and replace only earlier version 2.0 packages and do not affect any installed OpenMPI version 1.10 and earlier packages. The openmpi-1.10-scyld packages update...
to version 1.10.7, which by default update and replace only earlier version 1.10 packages and do not affect any installed OpenMPI version 1.8 and earlier packages. See *Installing and managing concurrent versions of packages* for details.


### 1.1.7.52 v7.3.3 - April 27, 2017

1. The base kernel is 3.10.0-514.16.1.el7.733g0000. See https://access.redhat.com/errata/RHSA-2017:0933.html for details. The Scyld ClusterWare kernel now includes built-in firmware to properly boot some compute node server models that employ bnx2, bnx2x, or cxgb3 Ethernet controllers.

2. The bproc filecache functionality now properly downloads files from the master node that were previously rejected because the files have restricted read access permissions. Now all files are downloaded to compute nodes - and, as always, downloaded files are given access permissions that are replicated from the master node.

3. The /etc/yum.repos.d/clusterware.repo.template file is updated to remove the # prefix characters in the cw-next section, thereby exposing that section, albeit with the default enabled=0 line. The cluster administrator should make a similar removal of those # characters in the local /etc/yum.repos.d/clusterware*repo file(s) currently in active use.

### 1.1.7.53 v7.3.2 - April 4, 2017

1. The base kernel is 3.10.0-514.10.2.el7.732g0000. See https://rhn.redhat.com/errata/RHSA-2017-0386.html for details.

2. Fix a problem of compute nodes too slowly reporting process CPU time stats to the master node, e.g., reported by the top command.


4. Scyld ClusterWare now distributes openmpi-2.1-scyld packages, which are initially version 2.1.0. Installation of openmpi-2.1 does not affect any earlier OpenMPI version. The libraries were built with Gnu version 4.8.5-11, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. ClusterWare releases of OpenMPI derive from https://www.open-mpi.org. See the ClusterWare *User’s Guide OpenMPI Release Information* for details.

5. Scyld ClusterWare now distributes Singularity, which is initially version 2.2.1. See https://www.sylabs.io/docs/ and the ClusterWare *User’s Guide Using Singularity* for details.

### 1.1.7.54 v7.3.1 - March 8, 2017

1. The base kernel is 3.10.0-514.6.1.el7.731g0000. See https://rhn.redhat.com/errata/RHSA-2017-0086.html for details.

2. The Slurm job manager updates to version 17.02.0, derived from https://slurm.schedmd.com. See the ClusterWare *User’s Guide SLURM Release Information* for details. (Note: the openmpi-* packages have been rebuilt for compatibility with the new Slurm.)

3. The openmpi-2.0-scyld packages update to version 2.0.2, which by default update and replace only earlier version 2.0 packages. The openmpi-1.10-scyld packages update to version 1.10.6, which by default update and replace only earlier version 1.10 packages. The libraries were built with Gnu version 4.8.5-11, Intel version 2013_sp1.3.174, and PGI version 14.6 compiler families. ClusterWare releases of OpenMPI derive from
https://www.open-mpi.org. See the ClusterWare User’s Guide OpenMPI Release Information for details, and Installing and managing concurrent versions of packages for general issues about supporting multiple concurrent versions of OpenMPI.

4. Fix a timing bug in beonsst that exhibits itself as a syslog warning (in /var/log/messages) involving the parsing of /etc/beowulf/config nodes and iprange directives.

1.1.7.55 v7.3.0 - January 20, 2017


2. The Slurm job manager updates to version 16.05.8, derived from https://slurm.schedmd.com See the ClusterWare User’s Guide SLURM Release Information for details.

3. Various scripts in /etc/beowulf/init.d/ have been renamed with different numeric prefixes in order to adjust the execution ordering: 95sudo, 98slurm, and 98torque. If any of these scripts has been copied and modified locally (see Caution when modifying Scyld ClusterWare scripts for details), then you should rename the local copy to match the new numeric prefix.

4. TORQUE 6 now supports cgroups instead of cpusets.

5. Optional openmpi-psm2-1.10-scyld and openmpi-psm2-2.0-scyld packages are now available to download and install. These psm2 packages work in conjunction with Intel(r) Omni-Path Architecture and its libpsm2.so.

1.1.7.56 v7.2.0 - November 14, 2016

1. This is the first release of Scyld ClusterWare 7.

2. The base kernel is 3.10.0-327.36.3.el7.720g0000. See https://rhn.redhat.com/errata/RHSA-2016-2098.html for details.

1.1.8 Known Issues And Workarounds

The following are known issues of significance with the latest version of Scyld ClusterWare v7.9.16 and suggested workarounds.

1.1.8.1 Issues with bootmodule firmware

A Scyld ClusterWare compute node needs a functional network connection to the master node and perhaps to other storage devices or nodes. Some controllers need device-specific firmware that is installed by the controller’s software driver at modprobe time. This firmware is commonly found in /lib/firmware/ file(s), which is populated by the RHEL7 base distribution’s kernel-firmware package and possibly also by additional 3rd-party software distributions that have been installed on the master node.

Device firmware files are commonly read by a controller’s software driver with the assistance of the systemd daemon. However, because systemd is not executing on compute nodes, these firmware files must be built-in to the ClusterWare kernel, and the software driver then reads the built-in file(s) without needing systemd. The latest ClusterWare kernel includes built-in firmware for various commonly seen non-Penguin Computing network controllers and devices, currently bnx2, cxgb3, and hfi1. This is certainly not all the firmware present in /lib/firmware/. Contact Penguin Computing Support if you need kernel support for a non-booting compute node or if experiencing any /var/log/messages error that mentions not finding a specific firmware file.
1.1.8.2 Kernel panic using non-*invpcid* old Intel nodes

Intel-processor nodes that do not support *invpcid* will suffer a kernel panic when used as a ClusterWare compute node or a master node. Examples of such processors are “Westmere” and “Sandy Bridge”, dating back to the 2010-2012 time frame. Currently, the only available workaround is to disable the kernel’s Spectre/Meltdown “independent page table” fixes that were introduced in the ClusterWare v7.4.3 kernel.

If all compute nodes are affected, then add the keyword *nopti* to the “kernelcommandline” directive in /etc/beowulf/config. For a cluster with a mix of affected and unaffected compute nodes, then you need only add an additional “kernelcommandline [nodes] [options]” line that specifies just the affected nodes. The [nodes] argument can be comma-separated no-blanks list of individual node number(s) and/or node number range(s), e.g., *kernelcommandline 128 or kernelcommandline 128,129 or kernelcommandline 48-52,60,72-70.*

For an affected master node, edit /etc/default/grub to add *nopti* to the GRUB_CMDLINE_LINUX variable, then execute `grub2-mkconfig -o /boot/grub2/grub.cfg` and reboot the master node.

1.1.8.3 Managing environment modules .version files

Several Scyld ClusterWare packages involve the use of environment modules. This functionality allows for users to dynamically set up a shell’s user environment for subsequent compilations and executions of applications, and for viewing the manpages for commands that are associated with those compilations and executions.

The ClusterWare packages are found in the various */opt/scyld/package/* subdirectories, and for each package there are subdirectories organized by package version number, compiler suite type, and per-version per-compiler subdirectories containing the associated scripts, libraries, executable binaries, and manpages for building and executing applications for that package. The */opt/scyld/modulefiles/package/* subdirectories contain per-package per-version per-compiler files that contain various pathname strings that are prepended to the shell’s $PATH, SLD_LIBRARY_PATH, and $MANPATH variables that properly find those */opt/scyld/package/* scripts, libraries, executable files, and manpages.

For example, `module load mpich2/intel/1.5` sets up the environment so that the mpicc and mpirun commands build and execute MPI applications using the Intel compiler suite and the mpich2 libraries specifically crafted for mpich2 version 1.5. The *module load* command also understands defaults. For example, `module load mpich2-gnu` defaults to use the gnu compiler and the mpich2 version specified by the contents of the file */opt/scyld/modulefiles/mpich2-gnu*.version* (if that file exists). Similarly, `module load mpich2` first looks at the contents of */opt/scyld/modulefiles/mpich2/.version* to determine the default compiler suite, then (supposing *gnu* is that default) looks at the contents of */opt/scyld/modulefiles/mpich2/gnu*.version* to determine which mpich2 software version to use.

As a general rule, after updating one of these ClusterWare packages that employs environment modules, the associated */opt/scyld/modulefiles/package’s* subdirectories’ .version files remain untouched. The responsibility for updating any .version file remains with the cluster administrator, presumably after consulting with users. If the contents of a .version points to a compiler suite or to a package version number that no longer exists, then a subsequent module load for that package which expects to use a default selection will fail with a message of the form:

```
ERROR:105: Unable to locate a modulefile
```

The user must then perform `module load` commands that avoid any reference to the offending .version, e.g., use the explicit `module load mpich2/intel/1.5`, until the cluster administrator resets the .version contents to the desired default. Each module-employing ClusterWare package installs sample files with the name .version.*versionNumber.*

The openmpi packages manage defaults differently. Suppose `openmpi-2.0-sclyld` is currently version 2.0.1 and is updating to 2.0.2. Just as the default update behavior is to replace all 2.0.1 packages with the newer 2.0.2 packages, this openmpi-2.0 update also silently changes the gnu, intel, and pgi .version files which happen to specify the same major-minor version, e.g., those that specify version 2.0.1 are silently updated to the newer 2.0.2. If, however, the
current .version files specify an older major-minor release, e.g., 1.10.4, then updating openmpi-2.0-scyld does not change any of these older major-minor .version specifiers.

Additionally, each set of openmpi-x.y-scyld packages maintain a major-minor symlink that points to the newest major-minor-release module file. For example, when openmpi-2.0-scyld version 2.0.1 is currently installed, then the /opt/scyld/modulefiles/openmpi/gnu/2.0 symlink changes to the 2.0.1 module file. When openmpi-2.0-scyld updates to 2.0.2, then /opt/scyld/modulefiles/openmpi/gnu/2.0 changes that symlink to point to the 2.0.2 module file. This convenient symlink allows for users to maintain job manager scripts that simply specify a major-minor number, e.g., module load openmpi/intel/2.0, that survives updates from openmpi-2.0-scyld 2.0.1 to 2.0.2 to 2.0.3, etc., versus using scripts that contain the more specific module load openmpi/intel/2.0.1 that break when 2.0.1 packages update to 2.0.2.

Note that each compiler suite can declare a different default package version, although most commonly the cluster administrator edits the /opt/scyld/modulefiles/package/compiler/.version files so that for a given package, all compiler suites reference the same default version number.

One method to check the current package defaults is to execute:

```bash
cd /opt/scyld/modulefiles
module purge
module avail
for m in $(ls); do module load $m; done
module list
module purge
```

and then verify each loaded default against the module avail available alternatives.

### 1.1.8.4 Installing and managing concurrent versions of packages

Scyld ClusterWare distributes various repackaged Open Source software suites, including several variations of “MPI”, e.g., openmpi, mpich-scyld, mpich2-scyld, mvapich2-scyld. Users manage the selection of which software stack to use via the module load command. See Managing environment modules .version files for details.

By default, install-scyld -u updates each existing package with the newest version of that package by installing the newest version and removing all earlier (i.e., lower-numbered) versions, thereby retaining only a single version of each software suite. For example, the openmpi-2.0-scyld packages update to the latest 2.0.x version (major 2, minor 0, version x), and the openmpi-1.10-scyld packages update to the latest 1.10.y (major 1, minor 10, version y). Thus, a default update of package openmpi-2.0 installs the newest version 2.0.x and removes earlier versions of 2.0, leaving versions 1.10.x, 1.8.x, 1.7.x, etc. untouched.

Because Scyld ClusterWare installs a package’s files into unique /opt/scyld/package/version version-specific directories, this permits multiple versions of each major-minor package to potentially co-exist on the master node, e.g., openmpi versions 2.0.2 and 2.0.1. Each such package/version subdirectory contains one or more compiler suite subdirectories, e.g., gnu, intel, and pgi, and each of those contain scripts, libraries, executable binaries, and manpages associated with that particular package, version, and compiler suite.

Some customers (albeit rarely) may wish to install multiple concurrent x.y.z versions for a given x.y major-minor because specific applications might only work properly when linked to a specific version, or applications might perform differently for different versions. For example, to retain openmpi version 2.0.1 prior to using install-scyld -u or yum update, which might replace those 2.0.1 packages with a newer 2.0.2 version, first edit /etc/yum.conf to add the line:

```bash
exclude=openmpi-2.0-scyld*
```

which blocks yum from updating any and all currently installed openmpi-2.0-scyld packages. If the cluster administrator wishes to install (for example) the 2.0.2 packages and not disturb the 2.0.1 installation, then temporarily comment-out that exclude=openmpi-2.0-scyld* line and execute:

### 1.1. Release Notes: Scyld ClusterWare Release v7.9.16-7916g0000

47
and then re-enable the exclude= line to again protect against any inadvertent openmpi-2.0-scyld updates. Manually install these additional downloaded rpms using rpm -iv and not use rpm -Uv or even yum install, as both of those commands will remove older openmpi-2.0-scyld packages.

### 1.1.8.5 Issues with OpenMPI

Scyld ClusterWare distributes repackaged releases of the Open Source OpenMPI, derived from https://www.open-mpi.org. The Scyld ClusterWare distributions consist of a `openmpi-x.y-scyld` base package for the latest OpenMPI version x.y.z, plus several compiler-environment-specific packages for gnu, intel, and pgi. For example, the distribution of OpenMPI non-psm2 version 2.0.1 consists of the base rpm `openmpi-2.0-scyld-2.0.1` and the various compiler-specific rpms: `openmpi-2.0-scyld-gnu-2.0.1`, `openmpi-2.0-scyld-intel-2.0.1`, and `openmpi-2.0-scyld-pgi-2.0.1`.

Scyld ClusterWare distributes versions `openmpi-2.0-scyld`, `openmpi-1.10-scyld`, and `openmpi-1.8-scyld`, as well as `openmpi-psm2-2.0-scyld` and `openmpi-psm2-1.10-scyld` for clusters using the Intel Omni-Path Architecture (OPA) networking (which also requires hfi1-psm rpms from the Intel OPA software bundle).


The modulefiles append the current shell's $PATH, $LD_LIBRARY_PATH, and $MANPATH with pathnames that point to the associated compiler-specific version-specific `/opt/scyld/openmpi/version/compiler/` subdirectories. This permits multiple versions to co-exist on the master node, with each variation being user-selectable at runtime using the `module load` command.

Many customers support multiple OpenMPI versions because some applications might only work properly when linked to specific OpenMPI versions. Sometimes an application needs only to be recompiled and relinked against a newer version of the libraries. Other applications may have a dependency upon a particular OpenMPI version that a simple recompilation won't fix. The cluster administrator can specify which compiler and version is the default by manipulating the contents of the various `version` files in the `/opt/scyld/modulefiles/openmpi/` (or `/openmpi-psm2/`) subdirectories. For example, a module load `openmpi` might default to specify version 1.10.4 of the gnu libraries, while module load `openmpi-psm2` might default to specify version 2.0.1 of the intel libraries, while at the same time a version-specific module load `openmpi-psm2/gnu/1.10.4` or module load `openmpi/pgi/1.8.8` allows the use of different compilers and libraries for different OpenMPI versions.

The latest Open Source release of `openmpi-2.0-scyld` is a “mandatory” install, and `openmpi-1.10-scyld`, `openmpi-1.8-scyld`, `openmpi-psm2-2.0-scyld`, and `openmpi-psm2-1.10-scyld` are “optional” and can be manually installed by the cluster administrator using (for example) `yum install openmpi-psm2-1.10-scyld-*.` A subsequent `yum update` will update each and every installed `openmpi-x.y-scyld` and installed `openmpi-psm2-x.y-scyld` to the latest available version x.y.z. If the cluster administrator wishes to retain additional x.y.z releases within an x.y family, then instead of doing `yum update`, the administrator should `yum update --exclude=openmpi*scyld-*`, then download specific rpms from the yum repo as desired using `yumdownloader`, and then manually install (not update) the rpms using rpm -i. Note that the use of `yumdownloader` and `rpm -i` is necessary because doing a simple (for example) `yum install openmpi-1.10-scyld-1.10.4` will not, in fact, execute a simple `install` and retain older 1.10.z packages. Rather, it actually executes an `update` and removes any and all older installed versions of `openmpi-1.10-scyld-1.10.z` rpms.
1.1.8.6 Issues with Scyld ClusterWare process migration in heterogeneous clusters

In a homogeneous cluster, all nodes (master and compute) are identical server models, including having identical amounts of RAM. In a heterogeneous cluster, the nodes are not all identical. The advantage of a homogeneous cluster is simplicity in scheduling work on the nodes, since every node is identical and interchangeable. However, in the real world, many if not most clusters are heterogeneous. Some nodes may have an attached GPU or different amounts of available RAM, or may even be different server models with different x86_64 processor technologies.

Scyld ClusterWare users have always needed to be aware of potential problems running applications on heterogeneous clusters. For example, applications expecting to employ a GPU have needed to take care to execute only on nodes with an attached GPU, and an application that is specifically compiled or linked to libraries that employ newer x86_64 instructions that are not universally understood by every x86_64 processor must ensure that the application only execute on the nodes with processors that understand those newer instructions.

However, RHEL7 heterogeneous clusters present a new set of challenges to users. The essence of the issue is this: when a software thread begins execution, some libraries (e.g., libc) make a one-time determination of which processor model is being used, and the library self-configures certain routines (e.g., strcmp) to use implementations that exploit processor model-specific instructions for optimal performance. However, if the software thread subsequently migrates to a different node in the cluster, then the thread’s one-time determination state migrates to the destination node. If the destination node does not support the same x86_64 instructions that are supported by the original node, then the software thread will likely suffer a fatal “invalid opcode” trap if it attempts to execute one of these optimized library routines. Scyld ClusterWare performs such a thread migration through the use of the bproc_move() or bproc_rfork() library routines found in libbproc. These bproc routines are employed by the MPICH and MVAPICH libraries and by the bpcp command.

The bpcp command links with a special ClusterWare /lib64/scyld/libc-2.17.scyld.so that uses only generic, universally acceptable x86_64 instructions. Users may similarly link applications to this special library by adding:

```plaintext
Xlinker -rpath=/lib64/scyld
```
as a linker option.

1.1.8.7 Issues with MVAPICH2 and mpirun_rsh or mpispawn

Scyld ClusterWare has applied a workaround to mpiexec to fix a problem with MPICH2 and MVAPICH2 exec’ing the application executable binary across NFS. The problem is not fixed for launching the application using mpirun_rsh or mpispawn, which likely will result in the application hanging as it attempts to execve() the application. We strongly encourage using only mpiexec to launch MPICH2 and MVAPICH2 applications.

1.1.8.8 Issues with Singularity

Singularity versions 3.2.1 (and earlier) supports running programs with the --containall option, which directs Singularity to execute programs inside a fully self-contained container, which in part involves allocating local PID values doing process fork/clone using the CLONE_NEWPID option. However, CLONE_NEWPID is not supported by bproc, and therefore the --containall option has been removed from the Singularity v3.2.1 distributed by ClusterWare. Since Singularity versions 3.3.0 and beyond use fully self-contained contains by default, ClusterWare does not support Singularity beyond v3.2.1.

Singularity containers that employ a root directory structure of separate /bin/ and /usr/bin/ subdirectories (e.g., Ubuntu), versus the Scyld ClusterWare 7 scheme of /bin being a symlink to /usr/bin/, may fail to execute a simple command. For example, when using an Ubuntu container on a compute node, singularity exec test.simg ls / fails with an error of the form “ls relocation error” because ClusterWare improperly resolves the location of the ls executable outside the container as /usr/bin/ls, and not inside the container where it resides as /bin/ls, and that results in a confusion of which libraries to use. The workaround is to use fully qualified pathnames for executables that are valid inside the container, e.g., do singularity exec test.simg /bin/ls /.
1.1.8.9 Issues with ptrace

Cluster-wide ptrace functionality is not yet supported in Scyld ClusterWare 7. For example, you cannot use a debugger running on the master node to observe or manipulate a process that is executing on a compute node, e.g., using `gdb -p procID`, where `procID` is a process ID of a compute node process. `strace` does function in its basic form, although you cannot use the `-f` or `-F` options to trace forked children if those children move away from the parent’s node.

1.1.8.10 Issues with IP Forwarding

If the `clusterware` service has started, then a subsequent `systemctl stop iptables` (or `restart`) will hang because it attempts to unload the `ipt_MASQUERADE` kernel module while the `clusterware` service is using (and not releasing) that module. For a workaround, edit `/etc/sysconfig/iptables-config` to change:

```
IPTABLES_MODULES_UNLOAD="yes"
```

to:

```
IPTABLES_MODULES_UNLOAD="no"
```

1.1.8.11 Issues with kernel modules

The `modprobe` command uses `/usr/lib/`uname -r`/modules.dep.bin` to determine the pathnames of the specified kernel module and that module’s dependencies. The `depmod` command builds the human-readable `modules.dep` and the binary `module.dep.bin` files, and it should be executed on the master node after installing any new kernel module.

Executing `modprobe` on a compute node requires additional caution. The first use of `modprobe` retrieves the current `modules.dep.bin` from the master node using `bproc`’s `filecache` functionality. Since any subsequent `depmod` on the master node rebuilds `modules.dep.bin`, then a subsequent `modprobe` on a compute node will only see the new `modules.dep.bin` if that file is copied to the node using `bpcp`, or if the node is rebooted and thereby silently retrieves the new file.

In general, you should not execute `depmod` on a compute node, since that command will only see those few kernel modules that have previously been retrieved from the master node, which means the node’s newly built `modules.dep.bin` will only be a sparse subset of the master node’s full `module.dep.bin`. `Bproc`’s `filecache` functionality will always properly retrieve a kernel module from the master node, as long as the node’s `module.dep.bin` properly specifies the pathname of that module, so the key is to have the node’s `module.dep.bin` be a current copy of the master’s file.

1.1.8.12 Issues with port numbers

Scyld ClusterWare employs several daemons that execute in cooperating pairs: a server daemon that executes on the master node, and a client daemon that executes on compute nodes. Each daemon pair communicates using TCP or UDP through a presumably unique port number. By default, Scyld ClusterWare uses ports 932 (beofs2), 933 (bproc), 3045 (beonss), and 5545 (beostats). In the event that one or more of these port numbers collides with a non-Scyld ClusterWare daemon using the same port number, the cluster administrator can override Scyld ClusterWare default port numbers to use different, non-colliding unused ports using the `/etc/beowulf/config` file’s `server` directive. See `man beowulf-config` and `/etc/beowulf/config` file’s `server` directive for a discussion of the `server` directive.

The official list of assigned ports and their associated services is [http://www.iana.org/assignments/port-numbers](http://www.iana.org/assignments/port-numbers), and `/etc/services` is a list shipped with your base distribution. However, the absence in either list of a specific port number is no guarantee that the port will not be used by some software on your cluster. Use `lsof -i :portNumber` to determine if a particular port number is in active use.
A common collision is with `beofs2` port 932 or `bproc` port 933, since the `rpc.statd` or `rpc.mountd` daemons may randomly grab either of those ports before ClusterWare can grab them. However, ClusterWare automatically recognizes the conflict and tries alternative ports until it finds an unused port. If this flexible search causes problems with other daemons, you can edit `/etc/beowulf/config` to specify a tentative override value using the `server beofs2` or `server bproc` directive, as appropriate.

Less common are collisions with `beonss` port 3045 or `beostats` port 5545. The `server beonss` and `server beostats` override values are used as-specified and not adjusted by ClusterWare at runtime.

### 1.1.8.13 Issues with Spanning Tree Protocol and portfast

Network switches with Spanning Tree Protocol (STP) enabled will block packets received on a port for the first 30 seconds after the port comes online, giving the switch and the Spanning Tree algorithm time to determine if the device on the new link is a switch, and to determine if Spanning Tree will block or forward packets from this port. This is done to prevent “loops” which can cause packets to be endlessly repeated at a high rate and consume all network bandwidth. Each time the link goes down and comes back up, another 30-second blocking delay occurs. This delay can prevent PXE/DHCP from obtaining an IP address, or can prevent the node’s initial kernel from downloading its initial root filesystem, which results in the node endlessly iterating in the early boot sequence, or can delay the node’s ongoing `filecache` provisioning of libraries to the node.

We recommend disabling STP if feasible. If not feasible, then we recommend reconfiguring the switch to use Rapid STP or portfast, which avoids the 30-second delay, or employing some other port mode that will forward packets as a port comes up. There is no generic procedure for enabling these options. For Cisco switches, see [http://www.cisco.com/en/US/products/hw/switches/ps700/products_tech_note09186a00800b1500.shtml](http://www.cisco.com/en/US/products/hw/switches/ps700/products_tech_note09186a00800b1500.shtml). For other switch models, see the model-specific documentation.

If that reconfiguration is also not possible, you may need to increase the default Scyld ClusterWare timeout used by the node to a value safely greater than the STP delay: e.g., add `rootfs_timeout=120` `getfile_timeout=120` to the `/etc/beowulf/config` `kernelcommandline` entry to increase the timeouts to 120 seconds.

### 1.1.8.14 Issues with Gdk

If you access a cluster master node using `ssh -X` from a workstation, some graphical commands or program may fail with:

```
Gdk-ERROR **: BadMatch (invalid parameter attributes)
  serial 798 error_code 8 request_code 72 minor_code 0
Gdk-ERROR **: BadMatch (invalid parameter attributes)
  serial 802 error_code 8 request_code 72 minor_code 0
```

Remedy this by doing:

```
export XLIB_SKIP_ARGB_VISUALS=1
```

prior to running the failing program. If this workaround is successful, then consider adding this line to `/etc/bashrc` or to `~/.bashrc`. See [https://bugs.launchpad.net/ubuntu/+source/xmms/+bug/58192](https://bugs.launchpad.net/ubuntu/+source/xmms/+bug/58192) for details.
1.1.8.15 Caution when modifying Scyld ClusterWare scripts

Scyld ClusterWare installs various scripts in /etc/beowulf/init.d/ that node_up executes when booting each node in the cluster. Any site-local modification to one of these scripts will be lost when a subsequent Scyld ClusterWare update overwrites the file with a newer version. If a cluster administrator believes a local modification is necessary, we suggest:

Copy the to-be-edited original script to a file with a unique name, e.g.:

```
cd /etc/beowulf/init.d
cp 20ipmi 20ipmi_local
```

Remove the executable state of the original:

```
beochkconfig 20ipmi off
```

Edit 20ipmi_local as desired.

Thereafter, subsequent Scyld ClusterWare updates may install a new 20ipmi, but that update will not re-enable the non-executable state of that script. The locally modified 20ipmi_local remains untouched. However, keep in mind that the newer Scyld ClusterWare version of 20ipmi may contain fixes or other changes that need to be reflected in 20ipmi_local because that edited file was based upon an older Scyld ClusterWare version.

1.1.8.16 Caution using tools that modify config files touched by Scyld ClusterWare

Software tools exist that might make modifications to various system configuration files that Scyld ClusterWare also modifies. These tools do not have knowledge of the Scyld ClusterWare specific changes and therefore may undo or cause damage to the changes or configuration. Care must be taken when using such tools. One such example is /usr/sbin/authconfig, which manipulates /etc/nsswitch.conf.

Scyld ClusterWare modifies these system configuration files at install time:

```
/etc/exports
/etc/nsswitch.conf
/etc/security/limits.conf
/etc/sysconfig/syslog
```

Additionally, Scyld ClusterWare uses chkconfig to enable nfs.

1.1.8.17 Running nscd service on master node may cause kickbackdaemon to misbehave

The nscd (Name Service Cache Daemon) service executes by default on the master node, and /usr/sbin/nscd executes by default on each compute node via /etc/beowulf/init.d/09nscd. However, if this service is also enabled and executes on the master node, then it may cause the Scyld ClusterWare name service kickbackdaemon to misbehave.

Accordingly, when the ClusterWare service starts, if it detects that the nscd service is running on the master node, then ClusterWare automatically stops that service. ClusterWare does not permanently disable that service on the master node. To do that:

```
systemct1 disable nscd
```
1.1.8.18 Beofdisk does not support local disks without partition tables

Currently, beofdisk only supports disks that already have partition tables, even if those tables are empty. Compute nodes with preconfigured hardware RAID, where partition tables have been created on the LUNs, should be configurable. Contact Customer Service for assistance with a disk without partition tables.

1.1.8.19 Issues with bproc and the getpid() syscall

BProc interaction with getpid() may return incorrect processID values.

Details: The Red Hat’s glibc implements the getpid() syscall by asking the kernel once for the current processID value, then caching that value for subsequent calls to getpid(). If a program calls getpid() before calling bproc_rfork() or bproc_vrfork(), then bproc silently changes the child’s processID, but a subsequent getpid() continues to return the former cached processID value.

Workaround: do not call getpid() prior to calling bproc_rfork or bproc_vrfork.
CHAPTER TWO

INSTALLATION GUIDE

2.1 Preface

Congratulations on purchasing Scyld ClusterWare, the most scalable and configurable Linux Cluster Software on the market. This guide describes how to install Scyld ClusterWare using Penguin’s installation repository. You should read this document in its entirety, and should perform any necessary backups of the system before installing this software. You should pay particular attention to keeping a copy of any local configuration files.

The Scyld ClusterWare documentation set consists of:

- The Installation Guide containing detailed information for installing and configuring your cluster.
- The Release Notes containing release-specific details, potentially including information about installing or updating the latest version of Scyld ClusterWare.

These product guides are available in two formats, HTML and PDF. You can browse the documentation on the Penguin Computing Support Portal at https://www.penguincomputing.com/support/documentation.

Once you have completed the Scyld ClusterWare installation, you can view the HTML and PDF documentations in /var/www/html/, or visit http://localhost/clusterware-docs/ and http://localhost/clusterware-docs.pdf in a web browser. Note that if you are visiting the web page from a computer other than the cluster’s master node, then you must change localhost to the master node’s hostname. For example, if the hostname is “iceberg”, then you may need to use its fully qualified name, such as http://iceberg.penguincomputing.com/clusterware-docs/ and http://iceberg.penguincomputing.com/clusterware-docs.pdf.

Note: If your reseller pre-installed Scyld ClusterWare on your cluster, you may skip these installation instructions and visit the User’s Guide and Reference Guide for helpful insights about how to use Scyld ClusterWare.

2.2 Scyld ClusterWare System Overview

Scyld ClusterWare streamlines the processes of configuring, running, and maintaining a Linux cluster using a group of commodity off-the-shelf (COTS) computers connected through a private network.

The front-end “master node” in the cluster is configured with a full Linux installation, distributing computing tasks to the other “compute nodes” in the cluster. Nodes communicate across a private network and share a common process execution space with common, cluster-wide process ID values.

A compute node is commonly diskless, as its kernel image is downloaded from the master node at node startup time using the Preboot eXecution Environment (PXE), and libraries and executable binaries are transparently transferred.
from the master node as needed. A compute node may access data files on locally attached storage or across NFS from an NFS server managed by the master node or some other accessible server.

In order for the master node to communicate with an outside network, it needs two network interface controllers (NICs): one for the private internal cluster network, and the other for the outside network. It is suggested that the master node be connected to an outside network so multiple users can access the cluster from remote locations.

![Cluster Layout](image)

**Figure 1. Cluster Configuration**

### 2.3 Recommended Components

Hardware selection for a ClusterWare system is based on the price/performance ratio. ClusterWare recommends the components listed below:

- **Processors**: 64-bit Intel® or AMD™ x86_64 architecture **required**, single-core or multi-core
- **Architecture**: 1 or multiple sockets per motherboard
- **Physical Memory**: 4096 MBytes (4 GBytes) or more preferred, minimum 2048 MBytes (2 GBytes)
- **Operating System**: Red Hat Enterprise Linux 7 (RHEL7) or CentOS7 **required**

The Scyld ClusterWare **Release Notes** state the specific version and update of Red Hat or CentOS required to support the ClusterWare release you are installing.

- **Network Interface Controllers (NIC)**: Gigabit Ethernet (Fast Ethernet at a minimum) PCI-X or PCI-Express adapters (with existing Linux driver support) in each node for the internal private IP network.

The master node typically employs an additional NIC for connecting the cluster to the external network. This NIC should be selected based on the network infrastructure (e.g., Fast Ethernet if the external network you are connecting the cluster to is Fast Ethernet).
Network Switch. The master node private network NIC and all compute nodes should be connected to a non-blocking Gigabit Ethernet switch for the internal private network. At a minimum, the network switch should match the speed of the network cards.

The switch is a critical component for correct operation and performance of the cluster. In particular, the switch must be able to handle all network traffic over the private interconnect, including cluster management traffic, process migration, library transfer, and storage traffic. It must also properly handle DHCP and PXE.

**Tip**

It is sometimes confusing to identify which NIC is connected to the private network. Take care to connect the master node to the private switch through the NIC with the same or higher speed than the NICs in the compute nodes.

Disk Drives. For the master node, we recommend using either Serial ATA (SATA) or SCSI disks in a RAID 1 (mirrored) configuration. The operating system on the master node requires approximately 3 GB of disk space. We recommend configuring the compute nodes without local disks (disk-less).

If local disks are required on the compute nodes, we recommend using them for storing data that can be easily re-created, such as scratch storage or local copies of globally-available data.

In the default configuration, /home on the master node is exported to the compute nodes; other file systems may be exported as well. After installing Scyld ClusterWare, see the file /etc/beowulf/fstab for the full list of default mounts for compute nodes. If you expect heavy file system traffic, we recommend that you provide a second pair of disks in a RAID 1 (mirrored) configuration for these exported file systems. Otherwise, it is possible for accesses to the exported file systems to interfere with the master node accessing its system files, thus affecting the master node’s ability to launch new processes and manage the cluster.

Optional Hardware Components. Gigabit Ethernet with a non-blocking switch serves most users. However, some applications benefit from a lower-latency interconnect.

Infiniband is an industry standard interconnect providing low-latency messaging, IP, and storage support. Infiniband can be configured as a single universal fabric serving all of the cluster’s interconnect needs.

More information about Infiniband may be found at the Infiniband Trade Association web site at http://www.infinibandta.org. ClusterWare supports Infiniband as a supplemental messaging interconnect in addition to Ethernet for cluster control communications.

2.4 Assembling the Cluster

The full Scyld ClusterWare Cluster Virtualization Software and the underlying Linux operating system are installed only on the master node.

Most recent hardware supports network boot (PXE boot), which ClusterWare requires for booting the compute nodes.

2.5 Software Components

The following are integral components of Scyld ClusterWare:

- beostatus: A graphic utility for monitoring the status of a ClusterWare cluster.
- Scyld ClusterWare: Allows processes to be started on compute nodes in the cluster and tracked in the process table on the master node. Scyld ClusterWare also provides process migration mechanisms to help in creating remote processes, and removes the need for most binaries on the remote nodes.
- MPICH2, MVAPICH2, and OpenMPI: Message Passing Interfaces, customized to work with Scyld ClusterWare.

For more detailed information on these software components, see the Administrator’s Guide and the User’s Guide.
2.6 Quick Start Installation

2.6.1 Introduction

Scyld ClusterWare is supported on Red Hat Enterprise Linux 7 (RHEL7) and CentOS7. This document describes installing on Red Hat, though installing on CentOS will be identical, except where explicitly noted. Scyld ClusterWare is installed on the master node after installing a RHEL7 or CentOS7 base distribution. You must configure your network interface and network security settings to support Scyld ClusterWare.

The compute nodes join the cluster without any explicit installation. Having obtained a boot image via PXE, the nodes are converted to a Scyld-developed network boot system and seamlessly appear as part of a virtual parallel computer.

This chapter introduces you to the Scyld ClusterWare installation procedures, highlights the important steps in the Red Hat installation that require special attention, and then steps you through the installation process. Installation is done using the `/usr/bin/yum` command, installing from a repository of rpms, typically across a network connection. See Detailed Installation Instructions for more detailed instructions. Refer to the Red Hat documentation for information on installing RHEL7.

2.6.2 Network Interface Configuration

Tip

To begin, you must know which interface is connected to the public network and which is connected to the private network. Typically, the public interface is eth0 and the private interface is eth1.

It is important to properly configure the network interfaces to support Scyld ClusterWare. The Network Configuration screen is presented during the RHEL7 installation; it can be accessed post-installation via the Applications -> System Settings -> Network menu options.

2.6.2.1 Cluster Public Network Interface

For the public network interface (typically eth0), the following settings are typical, but can vary depending on your local needs:

- DHCP is the default, and is recommended for the public interface.

- If your external network is set up to use static IP addresses, then you must configure the public network interface manually. Select and edit this interface, setting the IP address and netmask as provided by your Network Administrator.

- If you use a static IP address, the subnet must be different from that chosen for the private interface. You must set the hostname manually and also provide gateway and primary DNS IP addresses.

  Tip

  When configuring the network security settings (see Network Security Settings), Scyld recommends setting a firewall for the public interface.
2.6.2.2 Cluster Private Network Interface

Caution

For the private network interface (typically eth1), DHCP is shown as default, but this option cannot be used. You must configure the network interface manually and assign a static IP address and netmask.

Caution

The cluster will not run correctly unless the private network interface is trusted. You can set this interface as a “trusted device” when configuring the network security settings post-installation; see Trusted Devices.

For the cluster private interface (typically eth1), the following settings are required for correct operation of Scyld ClusterWare:

- Do not configure this interface using DHCP. You must select this interface in the Network Configuration screen and edit it manually in the Edit Interface dialog.
- Set this interface to “activate on boot” to initialize the specific network device at boot-time.
- Specify a static IP address. We recommend using a non-routable address (such as 192.168.x.x, 172.16.x.x to 172.30.x.x, or 10.x.x.x).
- If the public subnet is non-routable, then use a different non-routable range for the private subnet (e.g., if the public subnet is 192.168.x.x, then use 172.16.x.x to 172.30.x.x or 10.x.x.x for the private subnet).
• Once you have specified the IP address, set the subnet mask based on this address. The subnet mask must accommodate a range large enough to contain all of your compute nodes.

Figure 2. Private Network Interface Configuration

Tip
You must first select the private interface in the Network Configuration screen, then click Edit to open the Edit Interface dialog box.

Tip
Although you can edit the private interface manually during the Red Hat installation, making this interface a “trusted device” must be done post-installation.

2.6.3 Network Security Settings

Caution
The security features provided with this system do not guarantee a completely secure system.

The Firewall Configuration screen presented during the RHEL7 installation applies to the public network interface and should be set according to your local standards.

The RHEL7 installer allows you to select some, but not all, of the security settings needed to support Scyld ClusterWare. The remaining security settings must be made post-installation; see Trusted Devices.
Scyld has the following recommendations for configuring the firewall:

- Set a firewall for the public network interface (typically eth0).

- If you chose to install a firewall, you must make the private network interface (typically eth1) a “trusted device” to allow all traffic to pass to the internal private cluster network; otherwise, the cluster will not run correctly. This setting must be made post-installation.

- The Red Hat installer configures the firewall with most services disabled. If you plan to use SSH to connect to the master node, be sure to select SSH from the list of services in the Firewall Configuration screen to allow SSH traffic to pass through the firewall.

### 2.6.4 Red Hat RHEL7 or CentOS7 Installation

Scyld ClusterWare depends on the prior installation of certain RHEL7 or CentOS7 packages from the base distribution. Ideally, each Scyld ClusterWare rpm names every dependency, which means that when you use /usr/bin/yum to install Scyld ClusterWare, yum attempts to gracefully install those dependencies if the base distribution yum repository (or repositories) are accessible and the dependencies are found. If a dependency cannot be installed, then the Scyld installation will fail with an error message that describes what rpm(s) or file(s) are needed.

**Caution**

Check the Scyld ClusterWare Release Notes for your release to determine whether you must update your Red Hat or CentOS base installation. If you are not familiar with the yum command, see Updating Red Hat or CentOs Installation for details on the update procedures.

### 2.6.5 Scyld ClusterWare Installation

Scyld ClusterWare is installed using the Penguin Yum repository http://updates.penguincomputing.com/clusterware/. Each Scyld ClusterWare release is continuously tested with the latest patches from Red Hat and CentOS7. Before installing or updating your master node, be sure to visit the Support Portal to determine if any patches should be excluded due to incompatibility with ClusterWare. Such incompatibilities should be rare. Then, update RHEL7 or CentOS7 on your master node before proceeding (excluding incompatible packages if necessary) with installing or updating your Scyld ClusterWare.

#### 2.6.5.1 Configure Yum To Support ClusterWare

The Yum repo configuration file for Scyld ClusterWare must be downloaded from the Penguin Computing Support Portal and properly configured:


Click on Download your Yum repo file to download this clusterware.repo file and place the it in the /etc/yum.repos.d/ directory.

Set the permissions:

```
[root@scyld ~]# chmod 644 /etc/yum.repos.d/clusterware.repo
```

With this setup complete, your master node is ready to retrieve Scyld ClusterWare installations and updates.
2.6.5.2 Install ClusterWare

You can use Yum to install ClusterWare and all updates up to and including the latest ClusterWare release, assuming you have updated your RHEL7 or CentOS7 base distribution as prescribed in the ClusterWare Release Notes.

1. Verify the version you are running with the following:

   [root@scyld ~]# cat /etc/redhat-release

   This should return a string similar to “Red Hat Enterprise Linux Server release 7.9” or “CentOS Linux release 7.9.2009 (Core)”.

2. Install the Scyld ClusterWare script that simplifies installing (and later updating) software, then execute that script:

   [root@scyld ~]# yum install install-scyld
   [root@scyld ~]# install-scyld

3. Configure the network for Scyld ClusterWare: edit /etc/beowulf/config to specify the cluster interface, the maximum number of compute nodes, and the beginning IP address of the first compute node. See the remainder of this guide and the Administrator’s Guide for details.

4. Reboot your system.

5. To verify that ClusterWare was installed successfully, do the following:

   [root@scyld ~]# uname -r

   The result should match the specific ClusterWare kernel version noted in the Release Notes.

2.6.6 Trusted Devices

If you chose to install a firewall, you must make the private network interface (typically eth1) a “trusted device” to enable all traffic on this interface to pass through the firewall; otherwise, the cluster will not run properly. This must be done post-installation.

After you have installed Red Hat and Scyld ClusterWare, reboot the system and log in as “root”.

Access the security settings through the Red Hat Applications -> System Settings -> Security Level menu options.

In the Security Level Configuration dialog box, make sure the private interface is checked in the “trusted devices” list, then click OK.

**Tip**

If you plan to use SSH to connect to the master node, be sure that SSH is checked in the “trusted services” list.
You are now ready to boot and configure the compute nodes, as described in the next section.

### 2.6.7 Compute Nodes

In a Scyld cluster, the master node controls booting, provisioning, and operation of the compute nodes. You do not need to explicitly install Scyld ClusterWare on the compute nodes.

Scyld requires configuring your compute nodes to boot via PXE and using the auto-activate node options, so that each node automatically joins the cluster as it powers on. Nodes do not need to be added manually.

If you are not already logged in as root, log into the master node using the root username and password.

Use the command `bpstat -U` in a terminal window on the master node to view a continuously updated table of node status information.

Set the BIOS on each compute node to boot via PXE. Using the auto-activate option with PXE booting allows each node to automatically boot and join the cluster as it powers on.

Node numbers are initially assigned in order of connection with the master node. Boot the compute nodes by powering them up in the order you want them to be numbered, typically one-by-one from the top of a rack downwards (or from the bottom up). You can reorder nodes later as desired; see the *Administrator's Guide*.

The nodes transition through the boot phases. As the nodes join the cluster and are ready for use, they will be shown as “Up” by the `bpstat -U` command.
The cluster is now fully operational with disk-less compute nodes. See Cluster Verification Procedures for more about bpstat and node states.

2.7 Detailed Installation Instructions

This chapter provides detailed instructions for installing Scyld ClusterWare. This software installation is intended for the first computer (“node”) of the cluster, which functions as the “master node” to control and monitor other nodes and distribute jobs.

Scyld ClusterWare is installed on the master node that is running with a base distribution of RHEL7 or CentOS7.

It is assumed that you are familiar with the concepts outlined in the previous chapters, and that you have correctly assembled the hardware for your cluster. If this is not the case, please refer to the previous chapters to acquaint yourself with Scyld ClusterWare, and then verify that your hardware configuration is set up properly.

2.7.1 Red Hat Installation Specifics

During a RHEL7 installation, you have the option to configure various aspects of the installation to support Scyld ClusterWare. Important points include the following:

- **Disk partitioning** — Scyld recommends letting the installer automatically partition the disk; refer to the Red Hat documentation if you plan to manually partition instead.

- **Network interface configuration** — To support your Scyld cluster, you need to configure one interface dedicated to the external public network (typically eth0) and one to your internal private cluster network (typically eth1). Detailed instructions are provided in the section on Network Interface Configuration.

- **Network security settings** — You can configure some of your firewall settings during a RHEL7 installation. Other settings needed to support a Scyld cluster must be made post-installation. Detailed instructions are provided in the sections on Network Security Settings and Trusted Devices.

- **Package group selection** — Scyld recommends installing all Red Hat packages. See Package Group Selection.

The following sections provide instructions and/or recommendations for specific portions of the RHEL7 installation that are relevant to an optimal Scyld ClusterWare installation. This guide does not cover all steps in the RHEL7 installation; you should refer to the Red Hat documentation for more complete information.

2.7.1.1 Network Interface Configuration

**Tip**

To begin, you must know which interface is connected to the public network and which is connected to the private network. Typically, the public interface is eth0 and the private interface is eth1.

A typical Scyld cluster has one interface dedicated to the external public network (typically eth0) and one dedicated to your internal private cluster network (typically eth1). It is important to properly to configure both of these interfaces to support your Scyld ClusterWare installation.

The network interface configuration screen will be presented to you during a RHEL7 installation. For an existing Red Hat installation, you can access the network configuration screens through the Red Hat Applications -> System Settings -> Network menu options.
Cluster Public Network Interface

DHCP is selected by default for all network devices, as shown below in the Red Hat Network Configuration Screen. For the public network interface (typically eth0), this option is recommended.

![Network Configuration Screen](image)

Figure 1. Public Network Interface (DHCP Default is Recommended)

However, if your external network is set up to use static IP addresses, then follow these steps to manually configure the interface:

1. In the Network Configuration screen, select the public network interface (typically eth0) in the Network Devices list, then click *Edit* to open the Edit Interface dialog box.
2. In the Edit Interface dialog box:
   a. Select the Activate on boot checkbox to initialize the specific network device at boot-time.
   b. Specify the IP address and netmask provided by your network administrator.

When you have completed these settings, click OK to return to the Network Configuration screen.

3. In the Set the hostname area of the Network Configuration screen, select the manually radio button and provide a host name.

4. In the Miscellaneous Settings area of the screen, enter the gateway and primary DNS IP addresses provided by your Network Administrator.
Network Configuration

Any network devices you have on the system are automatically detected by the installation program and shown in the Network Devices list.

To configure the network device, first select the device and then click Edit. In the Edit Interface screen, you can choose to have the IP and Netmask information configured by DHCP or you can enter it manually. You can also choose to make the device active at boot time.

If you do not have DHCP client access or are unsure as to

Cluster Private Network Interface

Caution

For the private network interface (typically eth1), DHCP is shown as default, but this option cannot be used. The configuration tool Beonetconf requires a static IP address for the private interface. Therefore, you must configure the network interface manually and assign a static IP address and netmask.

The cluster will not run correctly unless the private network interface is trusted. You can set this interface as a “trusted device” when configuring the network security settings post-installation; see Trusted Devices.

1. In the Network Configuration screen, select the private network interface (typically eth1) in the Network Devices list, then click Edit to open the Edit Interface dialog box.
2. In the Edit Interface dialog box:
   
   a. Select the Activate on boot checkbox to initialize the specific network device at boot-time.
   
   b. Specify a static IP address. We recommend using a non-routable address (such as 192.168.x.x, 172.16.x.x to 172.30.x.x, or 10.x.x.x).
   
   c. If the public subnet is non-routable, then use a different non-routable range for the private subnet (e.g., if the public subnet is 192.168.x.x, then use 172.16.x.x to 172.30.x.x or 10.x.x.x for the private subnet).
   
   d. Once you have specified the IP address, set the subnet mask based on this address. The subnet mask must accommodate a range large enough to contain all of your compute nodes.
   
   When you have completed these settings, click OK to return to the Network Configuration screen.
   
3. In the Set the hostname area of the Network Configuration screen, you have the option to set the hostname automatically via the DHCP server or to provide one manually; this can be done according to your local standards. The following figure illustrates a completed typical configuration for both the public and private network interfaces.
2.7.1.2 Network Security Settings

**Caution**

The security features provided with this system do not guarantee a completely secure system.

The Firewall Configuration screen presented during the RHEL 7 installation applies to the public network interface and should be set according to your local standards. This screen allows you to customize several aspects of the firewall that protects your cluster from possible network security violations.

The RHEL 7 installer allows you to select some, but not all, of the security settings needed to support Scyld ClusterWare. The remaining security settings must be made post-installation; see *Trusted Devices.*
Scyld recommends setting a firewall for the public network interface (typically eth0). You can configure the following security settings during the Red Hat install:

Select from the following firewall options:

a. **No Firewall** — Allows all connections to your system and does no security checking. This option is not recommended unless you plan to configure your firewall after the installation.

b. **Enable Firewall** — Blocks any connections to your system that are not defaults or explicitly defined by you. By default, connections are allowed in response to outbound requests, such as DNS replies or DHCP requests.

Select services for which you want to allow possible connections. You can select any combination of the services listed.

**Tip**

If you plan to use SSH to connect to the master node, be sure that SSH is checked in the Trusted Services list.

Set the Enable SELinux? dropdown to “Disabled”.

If you chose to install a firewall, you must make the private network interface (typically eth1) a “trusted device” to enable all traffic on this interface to pass through the firewall. See *Trusted Devices*.
2.7.1.3 Package Group Selection

Caution

Scyld ClusterWare depends on certain Red Hat packages, and the Scyld installation may fail if the necessary Red Hat packages are not installed. Therefore, Scyld recommends that you install all Red Hat packages.

The Red Hat package selection screens enable you to select the particular software packages that you wish to install.

1. In the Package Installation Defaults screen, select the Customize… option.

![Package Installation Defaults](image)

The default installation environment includes our recommended package selection, including:

- Desktop shell (GNOME)
- Administration Tools
- Server Configuration Tools
- Web Server
- Windows File Server (SMB)

After installation, additional software can be added or removed using the ‘system-config-packages’ tool.

If you are familiar with Red Hat Enterprise Linux AS, you may have specific packages you would like to install or avoid installing. Check the box below to customize your installation.

- Install default software packages
- Customize software packages to be installed

Figure 7. Customize Package Installation

2. In the Package Group Selection screen, scroll down to the Miscellaneous section. Select the Everything checkbox, then continue the installation process.
Tip

To update an existing Red Hat installation to include all packages, insert the first Red Hat CD and invoke the Red Hat update program. Check the Everything box in the Package Group Selection screen, then continue with the update process.

2.7.2 Updating Red Hat or CentOS Installation

Update RHEL7 or CentOS7 either using yum, or using Red Hat or CentOS distribution media. Note that Penguin continually tests ClusterWare with new patches from Red Hat and CentOS. Visit the Penguin Computing Support Portal at https://www.penguincomputing.com/support to see the most recent errata fix tested with ClusterWare, and see any cautions about updated packages which may cause problems with ClusterWare.
2.7.2.1 Updating Using Yum

Use the following command:

```
[root@scyld ~]# yum update --disablerepo=cw*
```

(`--disablerepo=cw*` is used above in case the ClusterWare repo is already installed in `/etc/yum.repos.d`, you must exclude it during the `yum update`). You can also exclude other packages using the `--exclude=$package` parameter. See the `yum` man page for instructions on using `yum`. The CentOS web site also provides an online manual for `yum` at http://www.centos.org/docs/4/html/yum/.

2.7.2.2 Updating Using Media

If you update your system via distribution media, be sure to select an “upgrade install” rather than a “full install”, then follow the instructions provided with the media.

Tip

The just-installed newest base distribution kernel becomes the default in `/etc/grub.conf`. However, the Scyld ClusterWare includes a customized kernel that must be the kernel that is booted when running Scyld ClusterWare.

2.7.3 Scyld ClusterWare Installation

Scyld ClusterWare is installed using the Penguin Yum repository http://updates.penguincomputing.com/clusterware/. Each Scyld ClusterWare release is continuously tested with the latest patches from Red Hat and CentOS7. Before installing or updating your master node, be sure to visit the Support Portal to determine if any patches should be excluded due to incompatibility with ClusterWare. Such incompatibilities should be rare. Then, update RHEL7 or CentOS7 on your master node before proceeding (excluding incompatible packages if necessary) with installing or updating your Scyld ClusterWare.

2.7.3.1 Configure Yum To Support ClusterWare

The Yum repo configuration file for Scyld ClusterWare must be downloaded from the Penguin Computing Support Portal and properly configured:


Click on Download your Yum repo file to download this `clusterware.repo` file and place the it in the `/etc/yum.repos.d/` directory.

Set the permissions:

```
[root@scyld ~]# chmod 644 /etc/yum.repos.d/clusterware.repo
```

With this setup complete, your master node is ready to retrieve Scyld ClusterWare installations and updates.
2.7.3.2 Install ClusterWare

You can use Yum to install ClusterWare and all updates up to and including the latest ClusterWare release, assuming you have updated your RHEL7 or CentOS7 base distribution as prescribed in the ClusterWare Release Notes.

1. Verify the version you are running with the following:

   [root@scyld ~]# cat /etc/redhat-release

   This should return a string similar to “Red Hat Enterprise Linux Server release 7.9” or “CentOS Linux release 7.9.2009 (Core)”.

2. Install the Scyld ClusterWare script that simplifies installing (and later updating) software, then execute that script:

   [root@scyld ~]# yum install install-scyld
   [root@scyld ~]# install-scyld

3. Configure the network for Scyld ClusterWare: edit /etc/beowulf/config to specify the cluster interface, the maximum number of compute nodes, and the beginning IP address of the first compute node. See the remainder of this guide and the Administrator’s Guide for details.

4. Reboot your system.

5. To verify that ClusterWare was installed successfully, do the following:

   [root@scyld ~]# uname -r

   The result should match the specific ClusterWare kernel version noted in the Release Notes.

2.7.4 Trusted Devices

If you chose to install a firewall, you must make the private network interface (typically eth1) a “trusted device” to enable all traffic on this interface to pass through the firewall; otherwise, the cluster will not run properly. This must be done post-installation.

After you have installed Red Hat and Scyld ClusterWare, reboot the system and log in as “root”.

Access the security settings through the Red Hat Applications -> System Settings -> Security Level menu options.

In the Security Level Configuration dialog box, make sure the private interface is checked in the “trusted devices” list, then click OK.

Tip

If you plan to use SSH to connect to the master node, be sure that SSH is checked in the “trusted services” list.
2.7.5 Enabling Access to External License Servers

1. Enable ipforward in the /etc/beowulf/config file. The line should read as follows:
   
   ipforward yes

2. Restart the cluster services as “root”:

   :: [root@scyld ~]# systemctl restart clusterware

2.7.6 Post-Installation Configuration

Following a successful update or install of Scyld ClusterWare, you may need to make one or more configuration changes, depending upon the local requirements of your cluster. Larger cluster configurations have additional issues to consider. Accordingly, review the Release Notes sections titled Post-Installation Configuration Issues and Post-Installation Configuration Issues For Large Clusters for important detailed information.
2.7.7 Scyld ClusterWare Updates

You can use Yum update to update ClusterWare once you have upgraded your RHEL7 or CentOS7 base distribution. See Updating Red Hat or CentOS Installation or details on updating your base distribution, and Scyld ClusterWare Installation for how to set up the Yum repo configuration files.

To verify which distribution you are currently running, do the following:

```
[root@scyld ~]# cat /etc/redhat-release
```

2.7.7.1 Updating ClusterWare

1. It is advisable to update the base distribution prior to updating Scyld ClusterWare, taking care to exclude the base distribution’s kernel-* packages to avoid potentially updating to a newer kernel than is currently available in the Scyld ClusterWare yum repos:

```
[root@scyld ~]# yum --disablerepo=cw* --exclude=kernel-* update
```

2. Update the Scyld ClusterWare package that contains a useful script that simplifies updating ClusterWare, then execute that script:

```
[root@scyld ~]# yum update install-scyld
[root@scyld ~]# install-scyld -u
```

3. Compare /etc/beowulf/config, which remains untouched by the Scyld ClusterWare update, with the new config.rpmnew (if that file exists), and examine the differences:

```
[root@scyld ~]# cd /etc/beowulf
[root@scyld ~]# diff config config.rpmnew
```

and carefully merge the config.rpmnew differences into /etc/beowulf/config. Similarly, the preexisting /etc/beowulf/fstab may have been saved as fstab.rpmsave if it was locally modified. If so, merge those local changes back into /etc/beowulf/fstab

4. Reboot your system.

5. To verify that ClusterWare was installed successfully, do the following:

```
[root@scyld ~]# uname -r
```

The result should match the ClusterWare kernel version noted in the Release Notes.

6. Restart the compute nodes.

2.8 Cluster Verification Procedures

Once the master node and compute nodes have been configured and rebooted, you should run through the cluster verification to identify common software and hardware configuration problems. This chapter describes the Scyld ClusterWare tools for monitoring cluster status and running jobs across the cluster.

Cluster verification is generally required by reseller technical support when starting on a new issue. When you call your reseller for support, they will require that you have completed the cluster verification procedures outlined in this chapter, and that you capture information using the beosi script.

Also see the Administrator’s Guide and the User’s Guide for more detailed information.

2.8. Cluster Verification Procedures 75
2.8.1 Monitoring Cluster Status

You can monitor the status of the nodes in your cluster using the `bpstat` or `beostatus` commands.

2.8.1.1 bpstat

The `bpstat` command, run at a shell prompt on the master node, shows a table of status information for each node in the cluster. You do not need to be a privileged user to use this command.

Following is an example of the outputs from `bpstat` for a cluster with 10 compute nodes.

<table>
<thead>
<tr>
<th>Node(s)</th>
<th>Status</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-9</td>
<td>down</td>
<td>----------</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>4</td>
<td>up</td>
<td>---x--x--x</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>0-3</td>
<td>up</td>
<td>---x--x--x</td>
<td>root</td>
<td>root</td>
</tr>
</tbody>
</table>

Some things to keep in mind for `bpstat`:

- Ensure that each node is listed as `up`. The node count is based upon the `nodes` and `iprange` entries in the `/etc/beowulf/config` configuration file.
- Nodes that have not yet been configured are marked as `down`.
- Nodes currently booting are temporarily shown with a status of `boot`.
- An `error` status indicates a node initialization problem. Check for error messages in the log file `/var/log/beowulf/node.N` (where N is the node number). Typical problems are failing network connections, unpartitioned harddrives, or unavailable network file systems.

2.8.1.2 BeoStatus

The BeoStatus tool is a graphical user interface (GUI) program. You can start it by clicking the BeoStatus icon on the desktop.

Alternatively, type the command `beostatus` in a terminal window on the master node; you do not need to be a privileged user to use this command.

You can also view the status of the cluster in text mode by typing the command `beostatus -c` at a terminal window on the master node.

The default BeoStatus GUI mode (shown below) is a tabular format known as the “Classic” display. Each row corresponds to a different node, with specific state and resource usage information displayed per node.
Figure 1. BeoStatus in the “Classic” Display Mode

You should sanity-check the information shown in the BeoStatus window. The configured nodes that are powered up (those with a green checkmark in the “Up” column) should show expected values in the associated usage columns. When there are no active jobs on your cluster, the CPU and Network columns should be fairly close to zero. The memory usage columns (Memory, Swap, and Disk) should be showing reasonable values.

- **Node** — The node’s assigned node number, starting at zero. Node -1, if shown, is the master node. The total number of node entries shown is set by the “iprange” or “nodes” keywords in the file /etc/beowulf/config, rather than the number of detected nodes. The entry for an inactive node displays the last reported data in a grayed-out row.

- **Up** — A graphical representation of the node’s status. A green checkmark is shown if the node is up and available. Otherwise, a red “X” is shown.

- **State** — The node’s last known state. This should agree with the state reported by both the bpstat and beostatus commands.

- **CPU “X”** — The CPU loads for the node’s processors; at minimum, this indicates the CPU load for the first processor in each node. Since it is possible to mix uni-processor and multi-processor machines in a Scyld cluster, the number of CPU load columns is equal to the maximum number of processors for any node in your cluster. The label “N/A” will be shown for nodes with less than the maximum number of processors.

- **Memory** — The node’s current memory usage.

- **Swap** — The node’s current swap space (virtual memory) usage.

- **Disk** — The node’s harddrive usage. If a RAM disk is used, the maximum value shown is one-half the amount of physical memory. As the RAM disk competes with the kernel and application processes for memory, not all the RAM may be available.

- **Network** — The node’s network bandwidth usage. The total amount of bandwidth available is the sum of all network interfaces for that node.
2.8.2 Running Jobs Across the Cluster

Jobs can be executed on a Scyld cluster using either “directed execution” with the `bpsh` command or “dynamic execution” with the `beorun` or `mpprun` commands.

2.8.2.1 Directed Execution with bpsh

In the directed execution mode, the user explicitly defines which node (or nodes) will run a particular job. This mode is invoked using the `bpsh` command, the ClusterWare shell command analogous in functionality to both the `rsh` (remote shell) and `ssh` (secure shell) commands. Following are some examples of using `bpsh`:

- This example runs `hostname` on the compute node and writes the output back to the user’s screen from compute node 0:

  ```
  [user@cluster user]$ bpsh 0 /bin/hostname
  .0
  ```

- The following example runs the uptime utility on node 0, assuming it is installed in `/usr/bin`:

  ```
  [user@cluster user]$ bpsh 0 /usr/bin/uptime
  12:56:44 up 4:57, 5 users, load average: 0.06, 0.09, 0.03
  ```

2.8.2.2 Dynamic Execution with beorun and mpprun

In the dynamic execution mode, Scyld decides which node is the most capable of executing the job at that moment in time. Scyld includes two parallel execution tools that dynamically select nodes, `beorun` and `mpprun`. They differ only in that `beorun` runs the job on the selected nodes concurrently, while `mpprun` runs the job sequentially on one node at a time.

The following example shows the difference in the amount of time the system uses to run a command with `beorun` vs. `mpprun`:

```
[user@cluster user]$ date;beorun -np 8 sleep 1;date
Fri Aug 18 11:48:30 PDT 2006
Fri Aug 18 11:48:31 PDT 2006

[user@cluster user]$ date;mpprun -np 8 sleep 1;date
Fri Aug 18 11:48:46 PDT 2006
Fri Aug 18 11:48:54 PDT 2006
```

2.9 Troubleshooting ClusterWare

2.9.1 Failing PXE Network Boot

If a compute node fails to join the cluster when booted via PXE network boot, there are several places to look, as discussed below.

**Rule out physical problems.** Check for disconnected Ethernet cables, malfunctioning network equipment, etc.

**Check the system logs.** There are several log files:
• The master node’s /var/log/messages file combines rsyslog output from the master node and each compute node. The master node’s Scyld ClusterWare beoserv daemon serves as the cluster’s DHCP server, and it logs the basic PXEboot interactions with each compute node. If a compute node shows no PXEboot logging, then the beoserv daemon is not seeing the initial PXEboot or DHCP request. Verify that the master node’s private cluster network firewall is not blocking incoming requests.

• If the syslog shows a compute node is making repeated PXEboot responses without ever reaching boot, error, or up state, then the Scyld ClusterWare beoclient daemon on the compute node is unable to start up the node.

Commonly, beoclient is failing to load the appropriate kernel binary module for the Ethernet interface. Ensure that /etc/beowulf/config specifies a bootmodule for the Ethernet controller hardware used by that specific compute node server, and that any modarg module options are valid for that particular kernel driver. Scyld ClusterWare distributes bootmodule entries for all Penguin Computing servers. If your compute node is not a Penguin Computing server, then verify that the necessary kernel driver is named as a bootmodule.

Definitive diagnosis may require viewing the compute node’s console output, either by attaching a graphical monitor to the console port, attaching a serial cable from the compute node’s serial console output to another server and using /usr/bin/minicom to capture the output, or capturing the compute node’s serial console output using the IPMI serial console functionality.

• If a compute node reaches boot state, then example the node’s individual /var/log/beowulf/node.log file, where N is the node number.

Check for the correct DHCP server. If a node fails to appear initially (on power-up), or appears then subsequently disappears, then the node may be unable to find the master node’s DHCP server. Another DHCP server may be answering and supplying IP addresses.

To check whether the master is seeing the compute node’s DHCP requests, or whether another server is answering, use the Linux tcpdump utility. The following example shows a correct dialog between compute node 0 (10.10.100.100) and the master node.

```
[root@cluster ~]# tcpdump -i eth1 -c 10
Listening on eth1, link-type EN10MB (Ethernet),
capture size 96 bytes
18:22:07.901571 IP master.bootpc > 255.255.255.255.bootps:
   BOOTP/DHCP, Request from .0, length: 548
18:22:07.902579 IP .-1.bootps > 255.255.255.255.bootpc:
   BOOTP/DHCP, Reply, length: 430
18:22:09.974536 IP master.bootpc > 255.255.255.255.bootps:
   BOOTP/DHCP, Request from .0, length: 548
18:22:09.974882 IP .-1.bootps > 255.255.255.255.bootpc:
   BOOTP/DHCP, Reply, length: 430
18:22:09.977268 arp who-has .-1 tell 10.10.100.100
18:22:09.977285 arp reply .-1 is-at 00:0c:29:3b:4e:50
18:22:09.977565 IP 10.10.100.100.2070 > .-1.tftp: 32 RRQ
   "bootimg::loader" octet tsize 0
18:22:09.978299 IP .-1.32772 > 10.10.100.100.2070:
   UDP, length 14
10 packets captured
32 packets received by filter
0 packets dropped by kernel
```

Check the network interface. Verify that the master node’s network interface is properly set up. Then check the network interface settings using beonetconf. Reconfigure as needed, and restart cluster services again.

Verify that ClusterWare services are running. Check the status of ClusterWare services by entering the following command in a terminal window:
[root@cluster ~]# systemctl status clusterware

Restart ClusterWare services from the command line using:

[root@cluster ~]# systemctl restart clusterware

**Check the switch configuration.** If the compute nodes fail to boot immediately on power-up but successfully boot later, the problem may lie with the configuration of a managed switch.

Some Ethernet switches delay forwarding packets for approximately one minute after link is established, attempting to verify that no network loop has been created (“spanning tree”). This delay is longer than the PXE boot timeout on some servers.

Disable the spanning tree check on the switch; the parameter is typically named “fast link enable”. See the *Administrator’s Guide* for more details.

### 2.9.2 Mixed Uni-Processor and SMP Cluster Nodes

The Scyld ClusterWare system architecture eliminates the problem of unintentionally running different versions of a program over the cluster’s compute nodes.

The cluster nodes are required to run the same kernel version, typically with the same features and optimization enabled. Uni-processor machines can run the SMP kernel. The best choice for a mixed cluster is to run the SMP kernel. Beginning with CW4.1.1, support for uniprocessor kernels was dropped.

### 2.9.3 IP Forwarding

If IP forwarding is enabled in `/etc/beowulf/config` but is still not working, then check `/etc/sysctl.conf` to see if it is disabled.

Check for the line “net.ipv4.ip_forward = 1”. If the value is set to 0 (zero) instead of 1, then IP forwarding will be disabled, even if it is enabled in `/etc/beowulf/config`.

### 2.9.4 SSH Traffic

The Red Hat installer configures the firewall with most services disabled. If SSH traffic isn’t passing through the firewall, then check your firewall settings to make sure SSH is selected as a trusted service.

To do this, log in as a root user and choose the Red Hat *Applications -> System Settings -> Security Level* menu option to open the Security Level Configuration window. Then make sure that SSH is checked in the list of trusted services.

### 2.9.5 Device Driver Updates

Scyld ClusterWare releases are tested on many different machine configurations, but it is impossible to provide device drivers for hardware unknown at release time.

Most problems with unsupported hardware or device-specific problems are resolved by updating to a newer device driver. Some devices may not yet be supported under Linux. Check with your hardware vendor.

The Scyld ClusterWare architecture makes most driver updates simple. Drivers are installed and updated on the master node exactly as with a single machine installation. The new drivers are immediately available to compute nodes, although already-loaded drivers are not replaced.
There are two irregular device driver types that require special actions: disk drivers and network drivers, both of which apply to the compute nodes. In both cases, the drivers must be available to load additional drivers and programs, and are thus packaged in initial RAM disk images.

Another irregular instance is where drivers must execute scripts when they load; one example is Infiniband. Contact the hardware vendor or Scyld support if you have difficulty with the script that loads the driver.

2.9.6 Finding Further Information

If you encounter a problem installing your Scyld cluster and find that this Installation Guide cannot help you, the following are sources for more information:

- See the Release Notes for special installation or upgrade procedures that must be taken for your particular version of ClusterWare. It is available on the master node or on the documentation CD included in the Scyld installation kit.
- See the Administrator's Guide, which includes descriptions of more advanced administration and setup options. It is available on the master node or on the documentation CD included in the Scyld installation kit.
- See the Reference Guide, a complete technical reference to Scyld ClusterWare. It is available on the master node or on the documentation CD included in the Scyld installation kit.

For the most up-to-date product documentation and other helpful information about Scyld ClusterWare, visit the Scyld Customer Support website at https://www.penguincomputing.com/support. and online documentation at https://www.penguincomputing.com/support/documentation.

2.10 Compute Node Disk Partitioning

2.10.1 Architectural Overview

The Scyld ClusterWare system uses a “disk-less administration” model for compute nodes. This means that the compute nodes boot and operate without the need for mounting any file system, either on a local disk or a network file system. By using this approach, the cluster system does not depend on the storage details or potential misconfiguration of the compute nodes, instead putting all configuration information and initialization control on the master.

This does not mean that the cluster cannot or does not use local disk storage or network file systems. Instead it allows the storage to be tailored to the needs of the application rather than the underlying cluster system.

The first operational issue after installing a cluster is initializing and using compute node storage. While the concept and process is similar to configuring the master machine, the “disk-less administration” model makes it much easier to change the storage layout on the compute nodes.

2.10.2 Operational Overview

Compute node hard disks are used for three primary purposes:

- **Swap Space** — Expands the Virtual Memory of the local machine.
- **Application File Storage** — Provides scratch space and persistent storage for application output.
- **System Caching** — Increases the size and count of executables and libraries cached by the local node.

In addition, a local disk may be used to hold a cluster file system (used when the node acts as a file server to other nodes). To make this possible, Scyld provides programs to create disk partitions, a system to automatically create and check file systems on those partitions, and a mechanism to mount file systems.
2.10.3 Disk Partitioning Procedures

Deciding on a partitioning schema for the compute node disks is no easier than with the master node, but it can be changed more easily.

Compute node hard disks may be remotely partitioned from the master using beofdisk. This command automates the partitioning process, allowing all compute node disks with a matching hard drive geometry (cylinders, heads, sectors) to be partitioned simultaneously.

If the compute node hard disks have not been previously partitioned, you can use beofdisk to generate default partition tables for the compute node hard disks. The default partition table allocates three partitions, as follows:

- A BeoBoot partition equal to 2 MB (currently unused)
- A swap partition equal to 2 times the node’s physical memory
- A single root partition equal to the remainder of the disk

The partition table for each disk geometry is stored in the directory /etc/beowulf/fdisk on the master node, with the filename specified in nomenclature that reflects the disk type, position, and geometry. Example filenames are hda:2495:255:63, hdb:3322:255:63, and sda:2495:255:63.

The beofdisk command may also be used to read an existing partition table on a compute node hard disk, as long as that disk is properly positioned in the cluster. The command captures the partition table of the first hard disk of its type and geometry (cylinder, heads, sectors) in each position on a compute node’s controller (e.g., sda or hdb). The script sequentially queries the compute nodes numbered 0 through N-1, where N is the number of nodes currently in the cluster.

2.10.3.1 Typical Partitioning

While it is not possible to predict every configuration that might be desired, the typical procedure to partition node disks is as follows:

1. From the master node, capture partition tables for the compute nodes:

   ```
   [root@cluster ~]# beofdisk -q
   ```

   With the `-q` parameter, beofdisk queries all compute nodes. For the first drive found with a specific geometry (cylinders, heads, sectors), it reads the partition table and records it in a file. If the compute node disk has no partition table, this command creates a default partition set and reports the activity to the console.

   If the partition table on the disk is empty or invalid, it is captured and recorded as described, but no default partition set is created. You must create a default partition using the `beofdisk -d` command; see Default Partitioning.

2. Based on the specific geometry of each drive, write the appropriate partition table to each drive of each compute node:

   ```
   [root@cluster ~]# beofdisk -w
   ```

   This technique is useful, for example, when you boot a single compute node with a local hard disk that is already partitioned, and you want the same partitioning applied to all compute nodes. You would boot the prototypical compute node, capture its partition table, boot the remaining compute nodes, and write that prototypical partition table to all nodes.

3. Reboot all compute nodes to make the partitioning effective.

4. If needed, update the file /etc/beowulf/fstab on the master node to record the mapping of the partitions on the compute node disks to the file systems.
2.10.3.2 Default Partitioning

To apply the recommended default partitioning to each disk of each compute node, follow these steps:

1. Generate default partition maps to /etc/beowulf/fdisk:

   ```
   [root@cluster ~]# beofdisk -d
   ```

2. Write the partition maps out to the nodes:

   ```
   [root@cluster ~]# beofdisk -w
   ```

3. You must reboot the compute nodes before the new partitions are usable.

2.10.3.3 Generalized, User-Specified Partitions

To create a unique partition table for each disk type/position/geometry triplet, follow these steps:

1. Remotely run the fdisk command on each compute node where the disk resides:

   ```
   [root@cluster ~]# bpsh n fdisk device
   ```

   where n is the node number or the first compute node with the drive geometry you want to partition, and device is the device you wish to partition (e.g., /dev/sda, /dev/hdb).

2. Once you have created the partition table and written it to the disk using fdisk, capture it and write it to all disks with the same geometry using:

   ```
   [root@cluster ~]# beofdisk -w
   ```

3. You must reboot the compute nodes before the new partitioning will be effective.

4. You must then map file systems to partitions as described later in this chapter.

2.10.3.4 Unique Partitions

To generate a unique partition for a particular disk, follow these steps:

1. Partition your disks using either default partitioning or generalized partitions as described above.

2. From the master node, remotely run the fdisk command on the appropriate compute node to re-create a unique partition table using:

   ```
   [root@cluster ~]# bpsh n fdisk device
   ```

   where n is the compute node number for which you wish to create a unique partition table and device is the device you wish to partition (e.g., /dev/sda).

3. You must then map file systems to partitions as described below.
### 2.10.4 Mapping Compute Node Partitions

If your compute node hard disks are already partitioned, edit the file `/etc/beowulf/fstab` on the master node to record the mapping of the partitions on your compute node disks to your file systems. This file contains example lines (commented out) showing the mapping of file systems to drives; read the comments in the `fstab` file for guidance.

1. Query the disks on the compute nodes to determine how they are partitioned:

   ```
   [root@cluster ~]# beofdisk -q
   ```

   This creates a partition file in `/etc/beowulf/fdisk`, with a name similar to `sda:512:128:32` and containing lines similar to the following:

   ```
   [root@cluster root]# cat sda:512:128:32
   /dev/sda1 : start= 32, size= 8160, id=89, bootable
   /dev/sda2 : start= 8192, size= 1048576, Id=82
   /dev/sda3 : start= 1056768, size= 1040384, Id=83
   /dev/sda4 : start= 0, size= 0, Id=0
   ```

2. Read the comments in `/etc/beowulf/fstab`. Add the lines to the file to use the devices named in the `sda` file:

   ```
   # This is the default setup from beofdisk
   #/dev/hda2 swap swap defaults 0 0
   #/dev/hda3 / ext2 defaults 0 0
   /dev/sda1 /boot ext2 defaults 0 0
   /dev/sda2 swap /scratch ext3 defaults 0 0
   /dev/sda3 /scratch ext3 defaults 0 0
   ```

3. After saving `fstab`, you must reboot the compute nodes for the changes to take affect.

### 2.11 Changes to Configuration Files

#### 2.11.1 Changes to Red Hat Configuration Files

An installation of Red Hat sets a default configuration optimized for a stand-alone server. Installing ClusterWare on a Red Hat installation changes some of these default configuration parameters to better support a cluster. The following sections describe the changes the ClusterWare installation automatically makes to the Red Hat configuration. Any of these may be reversed; however, reversing them may adversely affect the operation of the ClusterWare cluster.

1. `/etc/grub.conf` has been modified.

   After ClusterWare has been installed, the default boot becomes the newest ClusterWare kernel.

2. NFS Services default configuration has been modified.

   By default, Red Hat configures NFS to “off” for security reasons. However, most cluster applications require that at least the home directory of the master node be accessible to the compute nodes. NFS services on the master are set with the default to “on” for run levels 3, 4, and 5.

   The default out-of-box `chkconfig` for NFS on RHEL7 is as follows:

   ```
   [root@scyld ~]# chkconfig --list nfs
   nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
   ```

   ClusterWare has changed the default to the following:
To get NFS to mount directories from the master to the compute nodes, the file `/etc/exports` needs one entry per line for each file system to export from the master to the compute nodes (the RHEL-MAJOR default is a blank/non-existent file). ClusterWare creates this file if it didn’t already exist, and adds several new entries of the form:

```
ExportedDirectoryPathname @cluster(accessMode,syncMode,no_root_squash)
```

The export for `/home` from the master is configured with an `accessMode` of `rw` (read-write) and a `syncMode` of `sync` by default for data reliability reasons, and the non-/home directories are exported `ro` (read-only) for security reasons and `async` for performance reasons.

See the ClusterWare Release Notes for details about which directories are added by Scyld.

3. `/etc/sysconfig/syslog` has been modified.
   
   Compute nodes will forward messages to the master node’s syslogd daemon, which places them in `/var/log/messages`. In order for this to function correctly, ClusterWare modifies the `/etc/sysconfig/syslog` file by adding the `-r` option to the `SYSLOGD_OPTIONS` line:

```
SYSLOGD_OPTIONS="-m 0 -r"
```

### 2.11.2 Possible Changes to ClusterWare Configuration Files

A clean install of ClusterWare introduces various ClusterWare configuration files that include default settings that a local sysadmin may choose to modify. A subsequent upgrade from one ClusterWare release to a newer release will avoid replacing these potentially modified files. Instead, an update installs a new version of the default file as a file of the form `CWconfigFile.rpmnew`. Therefore, after a ClusterWare upgrade, the sysadmin is encouraged to compare each such existing `CWconfigFile` with the new default version to ascertain which of the new default entries are appropriate to manually merge into the preexisting `CWconfigFile` file.

1. `/etc/beowulf/config` and `config.rpmnew`
   
   ClusterWare specifies additional libraries for compute nodes that may help various applications and scripts execute out-of-the-box.

2. `/etc/beowulf/fstab` and `fstab.rpmnew`
   
   ClusterWare specifies additional `/dev` devices and NFS-mounted directories for compute nodes that may help various applications and scripts execute out-of-the-box.
3.1 Evaluation Installation Instructions

Thank you for your interest in evaluating Scyld ClusterWare. This guide describes how to install an evaluation copy of Scyld ClusterWare using Penguin’s installation repository. You should perform any necessary backups of the system before installing this software, and should pay particular attention to keeping a copy of any local configuration files.

To proceed with the evaluation, you will only need this document, together with two files which you should have received by email: `clusterware.repo` and `scyld.lic`.

3.2 Scyld ClusterWare System Overview

Scyld ClusterWare streamlines the processes of configuring, running, and maintaining a Linux cluster using a group of commodity off-the-shelf (COTS) computers connected through a private network.

The front-end “master node” in the cluster is configured with a full Linux installation, distributing computing tasks to the other “compute nodes” in the cluster. Nodes communicate across a private network and share a common process execution space with common, cluster-wide process ID values.

A compute node is commonly diskless, as its kernel image is downloaded from the master node at node startup time using the Preboot eXecution Environment (PXE), and libraries and executable binaries are transparently transferred from the master node as needed. A compute node may access data files on locally attached storage or across NFS from an NFS server managed by the master node or some other accessible server.

In order for the master node to communicate with an outside network, it needs two network interface controllers (NICs): one for the private internal cluster network, and the other for the outside network. It is suggested that the master node be connected to an outside network so multiple users can access the cluster from remote locations.
3.3 Hardware Requirements

64-bit Intel® or AMD™ x86_64 processor architecture.

1024 MBytes (1 GByte) main memory, with 2048 MBytes (2 GBytes) or more preferred.

At least one Gigabit Ethernet Network Interface Controller on each compute node.

Preferably two Gigabit Ethernet Network Interface Controllers on the master node.

A Gigabit network switch for the private cluster network.

Optional: An Infiniband network infrastructure for compute nodes. Infiniband controllers must use the Mellanox™ chipset.

3.4 BIOS Requirements

Compute nodes must support the PXE network boot protocol. If an operating system has already been installed on a compute node’s local disk, the node’s BIOS must be configured to prioritize PXE network booting ahead of booting from the local disk.
3.5 Software Requirements

Scyld ClusterWare should be installed on a system running a base distribution of Red Hat Enterprise Linux (RHEL) or CentOS version 6 or 7.

The Infiniband interconnects (if present) must be supported by the mthca driver. When in doubt, you should contact your Infiniband hardware vendor to determine if your hardware is supported by this driver.

Scyld ClusterWare includes a customized version of the base distribution kernel that can co-exist with the kernel(s) currently installed on your master node.

NOTE: The Scyld ClusterWare kernel packages contain only those kernel modules that are included in the base distribution. This means that if your non-ClusterWare kernel is using a 3rd-party kernel module, e.g., for Panasas storage, or an Infiniband controller not supported by the mthca driver found in the base distribution, then that 3rd-party module (and whatever hardware it controls) is unavailable in the Scyld ClusterWare kernel environment.

3.6 Install Yum Configuration File and License File

You will have received two files by email that need to be installed in the appropriate places on your master node:

Install the clusterware.repo yum configuration file as /etc/yum.repos.d/clusterware.repo. This contains the credentials to identify your master node to Penguin Computing and to access Scyld ClusterWare software for download.

Install the scyld.lic evaluation license file as /etc/scyld.lic. This grants free use of ClusterWare for a month beginning from the date the license file was generated.

3.7 Update Base Distribution and Install Scyld Clusterware Software

We recommend that you visit https://www.penguincomputing.com/support/documentation to examine the Release Notes for the version you intend to install.

Once your master node’s /etc/yum.repos.d/ directory contains working yum repo files for both the base distribution (RHEL or CentOS) and for Scyld ClusterWare, then install and execute the very useful install-scyld script that guides you through the updating (if necessary) of software from the RHEL or CentOS base distribution and Scyld ClusterWare:

```
yum install install-scyld
install-scyld
```

Ideally, the script should have prompted you to accept the End User License Agreement (EULA).

3.8 Configure the Private and Public Networks

On the master node, execute `ip addr` to view the available Ethernet devices. These are typically named eth0 and eth1, although the names may vary for your master node. The controller that is connected to the private cluster network must have a static IP address and be able to communicate with all the compute nodes, which themselves will be assigned a dynamic IP address when booting. The controller that is connected to the public cluster network can have a dynamic IP address or a static IP address, although the latter is preferable for consistency in accessing the master node from some other machine in the public network space.
Edit the `/etc/beowulf/config` configuration file to specify the private network interface details. Find the “interface” directive, and change the initially undefined “none” name to the actual interface name, and change the “iprange” directive to be the base address of the first compute node, which is typically node n0. Then change the “nodes” directive to specify the maximum number of compute nodes that are connected to the private cluster network.

For example, suppose the interface name is “eth1”, the master node’s IP address is 10.20.0.1, and there are currently eight compute nodes connected, with a plan to add an additional eight later. The `config` file may then specify:

```
interface eth1
nodes 16
iprange 10.20.0.4
```

Note that the above “iprange” allows room for three additional master nodes, if desired, each with an unique IP address in the range 10.20.0.0 to 10.20.0.3. The 16 compute nodes span 10.20.0.4 to 10.20.0.19.

### 3.9 Start Cluster Operations

Reboot the master node:

```
[root@scyld ~]# reboot
```

After rebooting, run:

```
[root@scyld ~]# uname -r
```

and confirm that the master node is running a Scyld ClusterWare kernel.

Normally, Scyld ClusterWare services automatically start whenever the master node reboots. However, Scyld ClusterWare requires that you have read and accepted an End User License Agreement (EULA). You should have done this when executing the `install-scyld` script. Additionally, any error in the `/etc/beowulf/config` configuration file will result in a clusterware service startup error.

Test for a functional ClusterWare by executing the simple command to view the cluster status: `bpstat`. A successful first output line should begin with `Node(s)`. If that does not appear, then attempt to start the ClusterWare service manually and look for an error message:

```
[root@scyld ~]# systemctl start clusterware
```

Once the `clusterware` service is up and running, the master node can PXE boot as many compute nodes into the cluster as were defined by the “nodes” directive in the `/etc/beowulf/config` file. You can monitor the cluster status with the graphical `beostatus`, with the text-based `beostatus -c`, or with a simple `bpstat -U`.

Note: Depending upon BIOS settings, the compute nodes’ DHCP requests may timeout because the master node hadn’t been ready to respond, and compute nodes would then revert to a BIOS prompt waiting for human input. If `bpstat` continues to show that compute nodes are `down`, then ‘reset’ or powercycle each compute node, either manually or using an already configured `ipmitool`.

---

**3.9. Start Cluster Operations**

89
3.10 Documentation and Support

For a complete reference, the Scyld ClusterWare documentation set consists of:

- The full Installation Guide containing broader, more detailed information for installing and configuring the cluster.
- The Administrator’s Guide describing how to configure, maintain, and update the cluster.
- The User’s Guide and Reference Guide describing the commands, architecture, and programming interface for the cluster, including sample programs.
- The Release Notes containing release-specific details, including information about known issues and workarounds.

These product guides are available in two formats, HTML and PDF.

- Visit the Penguin Computing Support Portal at https://www.penguincomputing.com/support using the username and password you received for this evaluation, and click on Application Notes for information about running specific applications.
- Note: If you ssh -X from a remote system that executes a more recent version of X11 to the master node, some graphical programs may fail with an error of the form:

  Gdk-ERROR **: BadMatch (invalid parameter attributes)
  serial 798 error_code 8 request_code 72 minor_code 0

  For a workaround, try setting the ssh client host X11 depth to 8:

  [root@scyld ~]# export XLIB_SKIP_ARGB_VISUALS=1

  on the master node before running the failing program. If that is successful, then consider adding that export to /etc/bashrc or to an individual’s ~/.bashrc.

- For additional support, contact Customer Support at scyldEval@penguincomputing.com.

3.11 Purchasing Scyld ClusterWare

To license Scyld ClusterWare, please email support@penguincomputing.com

3.12 Uninstalling Scyld ClusterWare

1. Reboot the master node into a non Scyld ClusterWare kernel.
2. Use yum groupremove to uninstall ClusterWare:

   [root@scyld ~]# yum groupremove Scyld-ClusterWare

3. Restore any base distribution packages that were explicitly removed when you installed Scyld ClusterWare, e.g., openmpi* mvapich*.
4.1 Preface

Welcome to the Scyld ClusterWare Administrator’s Guide. It is written for use by Scyld ClusterWare administrators and advanced users. This document covers cluster configuration, maintenance, and optimization. As is typical for any Linux-based system, the administrator must have root privileges to perform the administrative tasks described in this document.

The beginning of this guide describes the Scyld ClusterWare system architecture and design; it is critical to understand this information in order to properly configure and administer the cluster. The guide then provides specific information about tools and methods for setting up and maintaining the cluster, the cluster boot process, ways to control cluster usage, methods for batching jobs and controlling the job queue, how load balancing is handled in the cluster, and optional tools that can be useful in administrating your cluster. Finally, the an appendix covers the important files and directories that pertain to operation of Scyld ClusterWare.

This guide is written with the assumption that the administrator has a background in a Unix or Linux operating environment; therefore, the document does not cover basic Linux system administration. If you do not have sufficient knowledge for using or administering a Linux system, we recommend that you first consult Linux in a Nutshell and other useful books published by O’Reilly and Associates.

When appropriate, this document refers the reader to other parts of the Scyld documentation set for more detailed explanations of the topic at hand. For information on the initial installation of Scyld ClusterWare, refer to the Installation Guide, which provides explicit detail on setting up the master node and booting the compute nodes. For administrators who are new to the ClusterWare concept, we recommend reading the User’s Guide first, as it introduces ClusterWare computing concepts.

4.2 Scyld ClusterWare Design Overview

This chapter discusses the design behind Scyld ClusterWare, beginning with a high-level description of the system architecture for the cluster as a whole, including the hardware context, network topologies, data flows, software context, and system level files. From there, the discussion moves into a technical description that includes the compute node boot procedure, the process migration technology, compute node categories and states, and miscellaneous components. Finally, the discussion focuses on the ClusterWare software components, including tools, daemons, clients, and utilities.

As mentioned in the preface, this document assumes a certain level of knowledge from the reader and therefore, it does not cover any system design decisions related to a basic Linux system. In addition, it is assumed the reader has a general understanding of Linux clustering concepts and how the second generation Scyld ClusterWare system differs from the traditional Beowulf. For more information on these topics, see the User’s Guide.
4.2.1 System Architecture

Scyld ClusterWare provides a software infrastructure designed specifically to streamline the process of configuring, administering, running, and maintaining commercial production Linux cluster systems. Scyld ClusterWare installs on top of a standard Linux distribution on a single node, allowing that node to function as the control point or “master node” for the entire cluster of “compute nodes”.

This section discusses the Scyld ClusterWare hardware context, network topologies, system data flow, system software context, and system level files.

4.2.1.1 System Hardware Context

A Scyld cluster has three primary components:

- The master node
- Compute nodes
- The cluster private network interface

These components are illustrated in the following block diagram. The remaining element in the diagram is the public/building network interface connected to the master node. This network connection is not required for the cluster to operate properly, and may not even be connected (for example, for security reasons).

Figure 1. Cluster Configuration

The master node and compute nodes have different roles in Scyld ClusterWare, and thus they have different hardware requirements. The master node is the central administration console for the cluster; it is the machine that all users of the cluster log into for starting their jobs. The master node is responsible for sending these jobs out to the appropriate compute node(s) for execution. The master node also performs all the standard tasks of a Linux machine, such as queuing print jobs or running shells for individual users.
Master Node

Given the role of the master node, it is easy to see why its hardware closely resembles that of a standard Linux machine. The master node will typically have the standard human user interface devices such as a monitor, keyboard, and mouse. It may have a fast 3D video card, depending on the cluster’s application.

The master is usually equipped with two network interface cards (NICs). One NIC connects the master to the cluster’s compute nodes over the private cluster network, and the other NIC connects the master to the outside world.

The master should be equipped with enough hard disk space to satisfy the demands of its users and the applications it must execute. The Linux operating system and Scyld ClusterWare together use about 7 GB of disk space. We recommend at least a 20 GB hard disk for the master node.

The master node should contain a minimum of 2 GB of RAM, or enough RAM to avoid swap during normal operations; a minimum of 4 GB is recommended. Having to swap programs to disk will degrade performance significantly, and RAM is relatively cheap.

Any network attached storage should be connected to both the private cluster network and the public network through separate interfaces.

In addition, if you plan to create boot CDs for your compute nodes, the master node requires a CD-RW or writeable DVD drive.

Compute Nodes

In contrast to the master node, the compute nodes are single-purpose machines. Their role is to run the jobs sent to them by the master node. If the cluster is viewed as a single large-scale parallel computer, then the compute nodes are its CPU and memory resources. They don’t have any login capabilities, other than optionally accepting ssh connections from the master node, and aren’t running many of the daemons typically found on a standard Linux box. These nodes don’t need a monitor, keyboard, or mouse.

Video cards aren’t required for compute nodes either (but may be required by the BIOS). However, having an inexpensive video card installed may prove cost effective when debugging hardware problems.

To facilitate debugging of hardware and software configuration problems on compute nodes, Scyld ClusterWare provides forwarding of all kernel log messages to the master’s log, and all messages generated while booting a compute node are also forwarded to the master node. Another hardware debug solution is to use a serial port connection back to the master node from the compute nodes. The kernel command line options for a compute node can be configured to display all boot information to the serial port. See Compute node command-line options or details about the console= configuration setting.

Compute node RAM requirements are dependent upon the needs of the jobs that execute on the node. Compute node physical memory is shared between its RAM-based root filesystem (rootfs) and the runtime memory needs of user applications and the kernel itself. As more space is consumed by the root filesystem for files, less physical memory is available to applications’ virtual memory and kernel physical memory, a shortage of which leads to Out-Of-Memory (OOM) events that result in application failure(s) and potentially total node failure.

Various remedies exist if the workloads fill the root filesystem or trigger Out-Of-Memory events, including adding RAM to the node and/or adding a local harddrive, which can be configured to add adequate swap space (which expands the available virtual memory capacity) and/or to add local filesystems (to reduce the demands on the RAM-based root filesystem). Even if local swap space is available and sufficient to avoid OOM events, optimal performance will only be achieved when there is sufficient physical memory to avoid swapping in the first place. See Compute Node Failure for a broader discussion of node failures, and Compute node command-line options for a discussion of the rootfs_size= configuration setting that limits the maximum size of the root filesystem.

A harddrive is not a required component for a compute node. If employed, we recommend using such local storage for data that can be easily re-created, such as swap space, scratch storage, or local copies of globally-available data.

4.2. Scyld ClusterWare Design Overview 93
If the compute nodes do not support PXE boot, a bootable CD-ROM drive is required.

### 4.2.1.2 Network Topologies

For many applications that will be run on Scyld ClusterWare, an inexpensive Ethernet network is all that is needed. Other applications might require multiple networks to obtain the best performance; these applications generally fall into two categories, “message intensive” and “server intensive”. The following sections describe a minimal network configuration, a performance network for “message intensive” applications, and a server network for “server intensive” applications.

#### Minimal Network Configuration

Scyld ClusterWare requires that at least one IP network be installed to enable master and compute node communications. This network can range in speed from 10 Mbps (Fast Ethernet) to over 1 Gbps, depending on cost and performance requirements.

![Minimal Network Configuration Diagram](image)

*Figure 2. Minimal Network Configuration*
Performance Network Configuration

The performance network configuration is intended for applications that can benefit from the low message latency of proprietary networks like Infiniband, TOE Ethernet, or RDMA Ethernet. These networks can optionally run without the overhead of an IP stack with direct memory-to-memory messaging. Here the lower bandwidth requirements of the Scyld software can be served by a standard IP network, freeing the other network from any OS-related overhead completely.

It should be noted that these high performance interfaces may also run an IP stack, in which case they may also be used in the other configurations as well.

Server Network Configuration

The server network configuration is intended for web, database, or application servers. In this configuration, each compute node has multiple network interfaces, one for the private control network and one or more for the external public networks.

The Scyld ClusterWare security model is well-suited for this configuration. Even though the compute nodes have a public network interface, there is no way to log into them. There is no /etc/passwd file or other configuration files to hack. There are no shells on the compute nodes to execute user programs. The only open ports on the public network interface are the ones your specific application opened.

To maintain this level of security, you may wish to have the master node on the internal private network only. The setup for this type of configuration is not described in this document, because it is very dependent on your target deployment. Contact Scyld’s technical support for help with a server network configuration.
4.2.1.3 System Data Flow

The following data flow diagram shows the primary messages sent over the private cluster network between the master node and compute nodes in a Scyld cluster. Data flows in three ways:

- From the master node to the compute nodes
- From the compute nodes to the master node
- From the compute nodes to other compute nodes

The job control commands and cluster admin commands shown in the data flow diagram represent inputs to the master from users and administrators.
Following is a list of the data items sent from the master node to a compute node, as depicted in the data flow diagram.

- **Cluster control commands** — These are the commands sent from the master to the compute node telling it to perform such tasks as rebooting, halting, powering off, etc.

- **Files to be cached** — The master node sends the files to be cached on the compute nodes under Scyld JIT provisioning.

- **Jobs, processes, signals, and app data** — These include the process snapshots captured by Beowulf for migrating processes between nodes, as well as the application data sent between jobs. Beowulf is the collection of software that makes up Scyld, including beoserv for PXE/DHCP, BProc, beomap, beonss, and beostat.

- **Final boot images** — The final boot image (formerly called the Phase 2 boot image) is sent from the master to a compute node in response to its Dynamic Host Configuration Protocol (DHCP) requests during its boot procedure.

### Compute Node to Master Node

Following is a list of the data items sent from a compute node to the master node, as depicted in the data flow diagram.

- **DHCP and PXE requests** — These requests are sent to the master from a compute node while it is booting. In response, the master replies back with the node’s IP address and the final boot image.

- **Jobs, processes, signals, and app data** — These include the process snapshots captured by Beowulf for migrating processes between nodes, as well as the application data sent between jobs.

- **Performance metrics and node status** — All the compute nodes in a Scyld cluster send periodic status information back to the master.
Compute Node to Compute Node

Following is a list of the data items sent between compute nodes, as depicted in the data flow diagram.

- **Jobs, processes, app data** — These include the process snapshots captured by Beowulf for migrating processes between nodes, as well as the application data sent between jobs.

4.2.1.4 System Software Context

The following diagram illustrates the software components available on the nodes in a Scyld cluster.

![Diagram of System Software Context](image)

**Master Node Software Components**

The master node runs the `bmaster`, `beoserv`, and `recvstats` daemons. This node also stores the Scyld-specific libraries `libbproc` and `libbeostat`, as well as Scyld-modified versions of utilities such as MPICH, LAM, and PVM. The commands and utilities are a small subset of all the software tools available on the master node.
Compute Node Software Components

The compute nodes run the beoclient daemon, which serves as the init process on the compute nodes, and run the Scyld beoklogd, bpslave, and sendstats daemons:

- **beoklogd** is run as soon as the compute node establishes a network connection to the master node, ensuring that the master node begins capturing compute node kernel messages as early as possible.
- **bpslave** is the compute node component of BProc, and is necessary for supporting the unified process space and for migrating processes.
- **sendstats** is necessary for monitoring the load on the compute node and for communicating the data to the master node's recvstats daemon.
- **kickbackproxy** communicates with the master node’s kickbackdaemon daemon to retrieve Name Service (NSS) information from the master node, e.g., hostnames and user names.

In general, minimal binaries reside on compute nodes, thus minimizing space consumed in a node’s RAM filesystem. By default, the directories that contain common commands (e.g., /usr/bin) are NFS-mounted. User applications are migrated from the master node to a compute node at run-time, using a command such as bpsh, or are accessed using an NFS mount. Libraries are pulled to a compute node on demand, as needed.

4.2.1.5 System Level Files

The following sections briefly describe the system level files found on the master node and compute nodes in a Scyld cluster.

Master Node Files

The file layout on the master node is the layout of the base Linux distribution. For those who are not familiar with the file layout that is commonly used by Linux distributions, here are some things to keep in mind:

- /bin, /usr/bin — directories with user level command binaries
- /sbin, /usr/sbin — directories with administrator level command binaries
- /lib, /usr/lib — directories with static and shared libraries
- /usr/include — directory with include files
- /etc — directory with configuration files
- /var/log — directory with system log files
- /var/beowulf — directory with various ClusterWare image and status files
- /usr/share/doc — directory with various documentation files

Scyld ClusterWare also has some special directories and files on the master node that are useful to know about. The per-node boot logs are stored in /var/log/beowulf/node.N, where N is the node number. The master node’s kernel and syslog messages are received by the syslog or rsyslog service, which appends these log messages to the master’s /var/log/messages file. By default, each compute node’s kernel and syslog messages are forwarded to the master node’s logging service and are also appended to the same /var/log/messages. However, the compute node logging can be optionally forwarded to the syslog or rsyslog service on another server. See the syslog_server= option in Compute node command-line options for details.

The legacy behavior of the compute node’s syslog handling has been to introduce a date-time string to the message text, then forward the message to the syslog server (typically on the master node), which would add its own date-time string. This redundant timestamp violates the RFC 3164 format standard, and recent ClusterWare releases strips the...
compute node's timestamp before sending the text to the master server. If for some reason a local cluster administrator wishes to revert to the previous behavior, then edit the /etc/beowulf/config's kernelcommandline directive to add legacy_syslog=1.

Configuration files for Scyld ClusterWare are found in /etc/beowulf/. The directory /usr/lib/beoboot/bin/ contains various scripts that are used to configure compute nodes during boot, including the node_up and setup_fs scripts.

For more information on the special directories, files, and scripts used by Scyld ClusterWare, see Special Directories, Configuration Files, and Scripts. Also see the Reference Guide.

Compute Node Files

Only a very few files exist on the compute nodes. For the most part, these files are all dynamic libraries; there are almost no actual binaries. For a detailed list of exactly what files exist on the compute nodes, see Special Directories, Configuration Files, and Scripts.

4.2.2 Technical Description

The following sections discuss some of the technical details of a Scyld cluster, such as the compute node boot procedure, the BProc distributed process space and Beowulf process migration software, compute node categories and states, and miscellaneous components.

4.2.2.1 Compute Node Boot Procedure

The Scyld cluster architecture is designed around light-weight provisioning of compute nodes using the master node's kernel and Linux distribution. Network booting ensures that what is provisioned to each compute node is properly version-synchronized across the cluster.

Earlier Scyld distributions supported a 2-phase boot sequence. Following PXE boot of a node, a fixed Phase 1 kernel and initial RAM disk (initrd) were copied to the node and installed. Alternatively, this Phase 1 kernel and initrd were used to boot from local hard disk or removable media. This Phase 1 boot package then built the node root filesystem rootfs in RAM disk, requested the run-time (Phase 2) kernel and used 2-Kernel-Monte to switch to it, then loaded the Scyld daemons and initialized the BProc system. Means were provided for installing the Phase 1 boot package on local hard disk and on removable floppy and CD media.

Beginning with Scyld 30-series, PXE is the supported method for booting nodes into the cluster. For some years, all servers produced have supported PXE booting. For servers that cannot support PXE booting, Scyld ClusterWare provides the means to easily produce Etherboot media on CD to use as compute node boot media. ClusterWare can also be configured to boot a compute node from a local disk. See Special Directories, Configuration Files, and Scripts.

The Boot Package

The compute node boot package consists of the kernel, initrd, and rootfs for each compute node. The beoboot command builds this boot package.

By default, the kernel is the one currently running on the master node. However, other kernels may be specified to the beoboot command and recorded on a node-by-node basis in the Beowulf configuration file. This file also includes the kernel command line parameters associated with the boot package. This allows each compute node to potentially have a unique kernel, initrd, rootfs, and kernel command lines.

Caution
Note that if you specify a different kernel to boot specific compute nodes, these nodes cannot be part of the BProc unified process space.

The path to the initrd and rootfs are passed to the compute node on the kernel command line, where it is accessible to the booting software.

Each time the ClusterWare service restarts on the master node, the beoboot command is executed to recreate the default compute node boot package. This ensures that the package contains the same versions of the components as are running on the master node.

**Booting a Node**

A compute node begins the boot process by sending a PXE request over the cluster private network. This request is handled by the beoserv daemon on the master node, which provides the compute node with an IP address and (based on the contents of the Beowulf configuration file) a kernel and initrd. If the cluster config file does not specify a kernel and initrd for a particular node, then the defaults are used.

The cluster config file specifies the path to the kernel, the initrd, and the rootfs. The initrd contains the minimal set of programs for the compute node to establish a connection to the master and request additional files. The rootfs is an archive of the root filesystem, including the filesystem directory structure and certain necessary files and programs, such as the bproc, filecache, and task_packer kernel modules and bpslave daemon.

The beoserv daemon logs its dialog with the compute node, including its MAC address, all of the node’s requests, and the responses. This facilitates debugging of compute node booting problems.

**The initrd and beoclient**

Once the initrd is loaded, control is transferred to the kernel. Within the Scyld architecture, booting is tightly controlled by the compute node’s beoclient daemon, which also serves as the compute node’s init process. The beoclient daemon uses configuration files and executable binaries in the initrd and initial root filesystem to load the the necessary kernel modules to establish the TCP/IP connection back to the master node and basic access to local harddrives, and starts various other daemons, such as beoklogd, which serves as the node’s local system log server to forward kernel and syslog messages (prefixed with the identify of the compute node) to the cluster’s syslog server, and the bpslave daemon. Once beoclient has initialized this basic BProc functionality, then the remaining boot sequence is directed by and controlled by the master node through the node_up and setup_fs scripts and various configuration files, bootstrapping on top of the BProc functionality now executing on the node.

The beoklogd daemon normally forwards the kernel and syslog messages from the compute node to the master node’s syslog or rsyslog service. However, this compute node logging can be optionally directed to an alternate server. See the syslog_server= option in Compute node command-line options for details. To facilitate debugging node booting problems, the kernel logging daemon on a compute node is started as soon as the network driver is loaded and the network connection to the syslog server is established.

**The rootfs**

Once the network connection to the master node is established and kernel logging has been started, beoclient requests the rootfs archive, using the path passed on the kernel command line. beoserv provides the rootfs tarball, which is then uncompressed and expanded into a RAM disk.
The **bpslave** daemon establishes a connection to **bpmaster** on the master node, and indicates that the compute node is ready to begin accepting work. **bpmaster** then launches the **node_up** script, which runs on the master node but completes initialization of the compute node using the **BProc** commands (**bpsh**, **bpcp**, and **bpct1**).

### 4.2.2.2 BProc Distributed Process Space

**Scyld Beowulf** is able to provide a single system image through its use of **BProc**, the Scyld process space management kernel enhancement. **BProc** enables the processes running on cluster compute nodes to be visible and manageable on the master node. Processes start on the master node and are migrated to the appropriate compute node by **BProc** process migration code. Process parent-child relationships and UNIX job control information are maintained with the migrated jobs, as follows:

- All processes appear in the master node’s process table.
- All standard UNIX signals (kill, suspend, resume, etc.) can be sent to any process on a compute node from the master.
- The **stdin**, **stdout** and **stderr** output from jobs is redirected back to the master through a socket.

**BProc** is one of the primary features that makes a Scyld cluster different from a traditional Beowulf cluster. It is the key software component that makes compute nodes appear as attached computational resources to the master node. The figure below depicts the role **BProc** plays in a Scyld cluster.

---

**Figure 7. BProc Data Flows in a Scyld Cluster**

**BProc** itself is divided into three components:
• **bpmaster** — a daemon program that runs on the master node at all times
• **bpslave** — a daemon program that runs on each of the compute nodes
• **libbproc** — a library that provides a user programming interface to BProc runtime intrinsics.

The user of a Scyld cluster will never need to directly run or interact with these daemons. However, their presence greatly simplifies the task of running parallel jobs with Scyld ClusterWare.

The **bpmaster** daemon uses a process migration module (**VMADump** in older Scyld systems or **TaskPacker** in newer Scyld systems) to freeze a running process so that it can be transferred to a remote node. The same module is also used by the **bpslave** daemon to thaw the process after it has been received. In a nutshell, the process migration module saves or restores a process’s memory space to or from a stream. In the case of **BProc**, the stream is a TCP socket connected to the remote machine.

**VMADump** and **TaskPacker** implement an optimization that greatly reduces the size of the memory space required for storing a frozen process. Most programs on the system are dynamically linked; at run-time, they will use **mmap** to map copies of various libraries into their memory spaces. Since these libraries are demand paged, the entire library is always mapped even if most of it will never be used. These regions must be included when copying a process's memory space and included again when the process is restored. This is expensive, since the C library dwarfs most programs in size.

For example, the following is the memory space for the program **sleep**. This is taken directly from /proc/pid/maps.

```
08048000-08049000  r-xp 00000000 03:01 288816 /bin/sleep
08049000-0804a000  rw-p 00000000 03:01 288816 /bin/sleep
40000000-40012000  r-xp 00000000 03:01 911381 /lib/ld-2.1.2.so
40012000-40013000  rw-p 00012000 03:01 911381 /lib/ld-2.1.2.so
40017000-40102000  r-xp 00000000 03:01 911434 /lib/libc-2.1.2.so
40102000-40106000  rw-p 000ea000 03:01 911434 /lib/libc-2.1.2.so
40106000-4010a000  rw-p 00000000 00:00 0
bffe0000-c0000000  rwxp fffff000 00:00 0
```

The total size of the memory space for this trivial program is 1,089,536 bytes; all but 32K of that comes from shared libraries. **VMADump** and **TaskPacker** take advantage of this; instead of storing the data contained in each of these regions, they store a reference to the regions. When the image is restored, **mmap** will map the appropriate files to the same memory locations.

In order for this optimization to work, **VMADump** and **TaskPacker** must know which files to expect in the location where they are restored. The **bplib** utility is used to manage a list of files presumed to be present on remote systems.

### 4.2.2.3 Compute Node Categories

Each compute node in the cluster is classified into one of three categories by the master node: “configured”, “ignored”, or “unknown”. The classification of a node is dictated by whether or where it is listed in one of the following files:

- The cluster config file **/etc/beowulf/config** (includes both “configured” and “ignored nodes”)
- The unknown addresses file **/var/beowulf/unknown_addresses** (includes “unknown” nodes only)

When a compute node completes its initial boot process, it begins to send out DHCP requests on all the network interface devices that it finds. When the master node receives a DHCP request from a new node, the new node will automatically be added to the cluster as “configured” until the maximum configured node count is reached. After that, new nodes will be classified as “ignored”. Nodes will be considered “unknown” only if the cluster isn’t configured to auto-insert or auto-append new nodes.

The cluster administrator can change the default node classification behavior by manually editing the **/etc/beowulf/config** file (discussed in *Configuring the Cluster Manually*). The classification of any specific node can also be changed manually by the cluster administrator. Also see *Special Directories, Configuration Files, and Scripts* to learn about special directories, configuration files, and scripts.
Following are definitions of the node categories.

**Configured** A “configured” node is one that is listed in the cluster config file `/etc/beowulf/config` using the `node` tag. These are nodes that are formally part of the cluster, and are recognized as such by the master node. When running jobs on your cluster, the “configured” nodes are the ones actually used as computational resources by the master.

**Ignored** An “ignored” node is one that is listed in the cluster config file `/etc/beowulf/config` using the `ignore` tag. These nodes are not considered part of the cluster, and will not receive the appropriate responses from the master during their boot process. New nodes that attempt to join the cluster after it has reached its maximum configured node count will be automatically classified as “ignored”.

The cluster administrator can also classify a compute node as “ignored” if for any reason you’d like the master node to simply ignore that node. For example, you may choose to temporarily reclassify a node as “ignored” while performing hardware maintenance activities when the node may be rebooting frequently.

**Unknown** An “unknown” node is one not formally recognized by the cluster as being either “configured” or “ignored”. When the master node receives a DHCP request from a node not already listed as “configured” or “ignored” in the cluster configuration file, and the cluster is not configured to auto-insert or auto-append new nodes, it classifies the node as “unknown”. The node will be listed in the `/var/beowulf/unknown_addresses` file.

### 4.2.2.4 Compute Node States

Cluster compute nodes may be in any of several functional states, such as *down, up, or unavailable*. Some of these states are transitional (*boot* or *shutdown*); some are informational variants of the *up state* (*unavailable* and *error*). BProc actually handles only 3 node operational variations:

- The node is not communicating — *down*. Variations of the *down* state may record the reason, such as *halted* (known to be halted by the master) or *reboot* (the master shut down the node with a reboot command).

- The node is communicating — *up, [up], alive, unavailable, or error*. Here the strings indicate different levels of usability.

- The node is transitioning — *boot*. This state has varying levels of communication, operating on scripted sequence.

During a normal power-on sequence, the user will see the node state change from *down* to *boot* to *up*. Depending on the machine speed, the *boot* phase may be very short and may not be visible due to the update rate of the cluster monitoring tools. All state information is reset to *down* whenever the `bpmaster` daemon is started/restarted.

In the following diagram, note that these states can also be reached via imperative commands such as `bpctl`. This command can be used to put the node into the *error* state, such as in response to an error condition detected by a script.
Following are definitions of the compute node states:

**down**  From the master node’s view, *down* means only that there is no communication with the compute node. A node is *down* when it is powered off, has been halted, has been rebooted, has a network link problem, or has some other hardware problem that prevents communication.

**boot**  This is a transitional state, during which the node will not accept user commands. The *boot* state is set when the node_up script has started and will transition to *up* or *error* when the script has completed. While in the *boot* state, the node will respond to administrator commands, but indicates that the node is still being configured for normal operation. The duration of this state varies with the complexity of the node_up script.

**up**  This is a functional state, set when the node_up script has completed without encountering any errors. BProc checks the return status of the script and sets the node state to *up* if the script was successful. This is the only state where the node is available to non-administrative users, as BProc checks this before moving any program to a node; administrator programs bypass this check. This state may also be commanded when the previous state was *unavailable* or *error*.

**error**  This is an informational state, set when the node_up script has exited with errors. The administrator may access the node, or look in the 
/var/log/beowulf/node.x (where x is a node number) file to determine the problem. If a problem is seen to be non-critical, the administrator may then set the node to *up*.

**unavailable**  This is a functional state. The node is not available for non-administrative users; however, it is completely available to the administrator. Currently running jobs will not be affected by a transition to this state. With respect to job control, this state comes into play only when attempting to run new jobs, as new jobs will fail to migrate to a node marked *unavailable*. This state is intended to allow node maintenance without having to bring the node offline.

**[up]**  This Scyld ClusterWare node is *up* and is being actively managed by another master node, which for now is the node’s primary master. The secondary master node(s) see the node as *[up]*. A secondary master can ssh to the node (if ssh is enabled), but the node only responds to BProc commands from its primary master (e.g., bpsh and bpcp). See Managing Multiple Master Nodes for details.

**alive**  This non-Scyld node is alive to the extent that it is running the sendstats daemon to report various /proc statistics about the node state, and it is integrated as a compute node in the cluster. For example, the Job Manager may be able to run jobs on this node. See Managing Non-Scyld Nodes for details.
4.2.2.5 Miscellaneous Components

Scyld ClusterWare includes several miscellaneous components, such as name lookup functionality (beonss), IP communications ports, library caching, and external data access.

beonss

beonss provides name service lookup functionality for Scyld ClusterWare. The information it provides includes hostnames, netgroups, and user information. In general, whatever name service information is available to the master node, using whatever query methods available to the master node (e.g., NIS, LDAP), is also transparently available to the compute nodes through the beonss functionality. The Scyld ClusterWare installation automatically (and silently) configures beonss.

Hostnames  beonss provides dynamically generated hostnames for all the nodes in the cluster. The hostnames are of the form .<nodenumber>, so the hostname for node 0 would be .0, the hostname for node 50 would be .50, and the hostname for the master node would be .-1.

The nodename entries in the /etc/beowulf/config file allow for the declaration of additional hostname aliases for compute nodes. For instance,

```
nodename n%N
```
declares aliases for nodes, e.g., n4 is an alias for node .4. For another example, suppose the IP address of node 4 is 10.0.0.4, and suppose that node 4 has its IPMI interface configured to respond to the IP address 10.1.0.4. Then the line:

```
nodename n%N-ipmi 0.1.0.0 ipmi
```
declares aliases for the hostnames in the group called ipmi. The hostname n4-ipmi is the arithmetic sum of n4’s IP address 10.1.0.4 plus the offset 0.1.0.0, forming the IP address 10.1.0.4. See man beowulf-config and the comments in the file /etc/beowulf/config for details and other examples.

beonss also provides the hostname master, which always points to the IP of the master node on the cluster’s internal network. The hostnames .-1 and master always point to the same IP.

These hostnames will always point to the right IP address based on the configuration of your IP range. You don’t need to do anything special for these hostnames to work. Also, these hostnames will work on the master node or any of the compute nodes.

Note that beonss does not know the hostname and IP address that the master node uses for the outside network. Suppose your master node has the public name mycluster and uses the IP address 1.2.3.4 for the outside network. By default, a compute node on the private network will be unable to open a connection to mycluster or to 1.2.3.4. However, by enabling IP forwarding in both the /etc/beowulf/config file and the /etc/sysctl.conf file, compute nodes can resolve hostnames and access hosts that are accessible by the master through the master’s public network interface, provided you have your DNS services working and available on the compute nodes.

Tip

When you enable IP forwarding, the master node will set up NAT routing between your compute nodes and the outside world, so your compute nodes will be able to make outbound connections. However, this does not enable outsiders to access or “see” your compute nodes.

Caution

On compute nodes the NFS directories must be mounted using either the NFS server’s IP address or the “$MASTER” keyword, as is specified in the /etc/beowulf/fstab file. Hostnames cannot be used because the compute node’s NFS mounting is performed before the node’s name service is active, which would otherwise be able to translate a hostname to its IP address.
**Netgroups** Netgroups are a concept from NIS. They make it easy to specify an arbitrary list of machines, then treat all those machines the same when carrying out an administrative procedure (for example, specifying what machines to export NFS filesystems to).

`beosss` creates one netgroup called `cluster`, which includes all of the nodes in the cluster. This is used in the default `/etc/exports` file in order to easily export `/home` to all of the compute nodes.

**User Information** When jobs are running on the compute nodes, `beosss` allows the standard `getpwnam()` and `getpwuid()` functions to successfully retrieve information (such as username, home directory, shell, and uid), as long as these functions are retrieving information on the user that is running the program. All other information that `getpwnam()` and `getpwuid()` would normally retrieve will be set to “NULL”.

**IP Communications Ports**

Scyld ClusterWare uses a few TCP/IP and UDP/IP communication ports when sending information between nodes. Normally, this should be completely transparent to the user. However, if the cluster is using a switch that blocks various ports, it may be important to know which ports are being used and for what.

Following are key components of Scyld ClusterWare and the ports they use:

- **beoserv** — This daemon is responsible for replying to the DHCP request from a compute node when it is booting. The reply includes a new kernel, the kernel command line options, and a small final boot RAM disk. The daemon supports both multi-cast and uni-cast file serving.

  By default, `beoserv` uses TCP port 932. This can be overridden by changing the value of the `server beofs2` directive (formerly `server tcp`, which is deprecated but continues to be accepted) in the `/etc/beowulf/config` file to the desired port number.

- **BProc** — This ClusterWare component provides unified process space, process migration, and remote execution of commands on compute nodes. By default, `BProc` uses TCP port 933. This can be overridden by changing the value of the `server bproc` directive in the `/etc/beowulf/config` file to the desired port number.

- **BeoStat** — This service is composed of compute node daemons (`sendstats`), a master node daemon (`recvstats`), and a master node library (`libbeostat`) that collects performance metrics and status information from compute nodes and transmits this information to the master node for caching and for distribution to the various cluster monitoring display tools. The daemons use UDP port 5545 by default.

**Library Caching**

One of the features Scyld ClusterWare uses to improve the performance of transferring jobs to and from compute nodes is to cache libraries. When `BProc` needs to migrate a job between nodes, it uses the process migration code (`VMADump` or `TaskPacker`) to take a snapshot of all the memory the process is using, including the binary and shared libraries. This memory snapshot is then sent across the private cluster network during process migration.

`VMADump` and `TaskPacker` take advantage of the fact that libraries are being cached on the compute nodes. The shared library data is not included in the snapshot, which reduces the amount of information that needs to be sent during process migration. By not sending over the libraries with each process, Scyld ClusterWare is able to reduce network traffic, thus speeding up cluster operations.
External Data Access

There are several common ways for processes running on a compute node to access data stored externally to the cluster, as discussed below.

Transfer the data. You can transfer the data to the master node using a protocol such as scp or ftp, then treat it as any other file that resides on the master node.

Access the data through a network filesystem, such as NFS or AFS. Any remote filesystem mounted on the master node can’t be re-exported to the compute node. Therefore, you need to use another method to access the data on the compute nodes. There are two options:

• Use bpsh to start your job, and use shell redirection on the master node to send the data as stdin for the job
• Use MPI and have the rank 0 job read the data, then use MPI’s message passing capabilities to send the data.

If you have a job that is natively using Beowulf functions, you can also have your job read the data on the master node before it moves itself to the compute nodes.

NFS mount directories from external file servers. There are two options:

• For file servers directly connected to the cluster private network, this can be done directly, using the file server’s IP address. Note that the server name cannot be used, because the name service is not yet up when /etc/beowulf/fstab is evaluated.
• For file servers external to the cluster, setting up IP forwarding on the master node allows the compute nodes to mount exported directories using the file server’s IP address.

Use a cluster filesystem. If you have questions regarding the use of any particular cluster filesystem with Scyld ClusterWare, contact Scyld Customer Support for assistance.

4.2.3 Software Components

The following sections describe the various software packages in Scyld ClusterWare, along with their individual components. For additional information, see the Reference Guide.

4.2.3.1 BeoBoot Tools

The following tools are associated with the beoboot package. For additional information, see the Reference Guide.

BeoBoot This utility is used to generate boot images for the compute nodes in the cluster. Earlier versions of Scyld used two types of images, initial (Phase 1) and final (Phase 2). The initial images were placed on the hard disk or a floppy disk, and were used to boot the nodes. The final image was downloaded from the master node by the initial image. Currently, only the final image is used by Scyld ClusterWare; support for initial images has been dropped.

By default, the final image is stored on the master node in the /var/beowulf/boot.img file; this is where the beoserv daemon expects to find it. Where initial images were used to begin the network boot process for systems that lacked PXE support, Scyld now provides PXELinux for this purpose. Bootable PXELinux media may be created for CD-ROM booting.

beoserv This is the BeoBoot daemon. It responds to DHCP requests from the compute nodes in the cluster and serves them their final boot images over the private cluster network.
4.2.3.2 BProc Daemons

The following daemons are associated with BProc. For additional information, see the Reference Guide.

bpmaster  This is the BProc master daemon. It runs on the master node, listening on a TCP port and accepting connections from bpslave daemons. Configuration information comes from the /etc/beowulf/config file.

bpslave  This is the BProc compute daemon. It runs on a compute node to accept jobs from the master, and connects to the master through a TCP port.

4.2.3.3 BProc Clients

The following command line utilities are closely related to BProc. For additional information, see the Reference Guide.

bpsh  This is a replacement for rsh (remote shell). It runs a specified command on an individually referenced node. The “nodenum” parameter may be a single node number, a comma delimited list of nodes, “-a” for all nodes that are up, or “-A” for all nodes that are not down.

bpsh will forward standard input, standard output, and standard error for the remote processes it spawns. Standard output and error are forwarded subject to specified options; standard input will be forwarded to the remote processes. If there is more than one remote process, standard input will be duplicated for every remote node. For a single remote process, the exit status of bpsh will be the exit status of the remote process.

bpctl  This is the BProc control utility. It can be used to apply various commands to individually referenced nodes. bpctl can be used to change the user and group ownership settings for a node; it can also be used to set a node’s state. Finally, this utility can be used to query such information as the node’s IP address.

bpcp  This utility can be used to copy files between machines in the cluster. Each file (f1...fn) or directory argument (dir) is either a remote filename of the form node:path, or a local filename (containing no colon “:” characters).

bpstat  This command displays various pieces of status information about the compute nodes. The display is formatted in columns specifying node number, node status, node permission, user access, and group access. This program also includes a number of options intended to be useful for scripts.

4.2.3.4 ClusterWare Utilities

Following are various command line and graphical user interface (GUI) utilities that are part of Scyld ClusterWare. For additional information, see the Reference Guide.

beostat  The beostat command line tool is a text-based utility used to monitor cluster status and performance. This tool provides a text listing of the information from the /proc structure on each node. See Monitoring the Status of the Cluster for a discussion of this tool.

beostatus  The beostatus GUI tool is used to monitor cluster status and performance. See Monitoring the Status of the Cluster for a discussion of this tool.

4.3 Monitoring the Status of the Cluster

Scyld ClusterWare provides several methods to monitor cluster performance and health, with a Web browser, a GUI, the command line, and “C” language interfaces. In general, these tools provide easy access to the information available through the Linux /proc filesystem, as well as BProc information for each of the cluster nodes. The monitoring programs are available to both administrators and regular users, since they provide no cluster command capabilities.
4.3.1 Monitoring Utilities

4.3.1.1 Cluster Monitoring Interfaces

Scyld ClusterWare provides several cluster monitoring interfaces. Following is brief summary of these interfaces; more detailed information is provided in the sections that follow:

- **libbeostat** — The libbeostat library, together with the compute nodes' sendstats daemons and the master node's recvstats daemon, provides the underpinning for the various display tools. Users can also create custom displays or create more sophisticated resource scheduling software by interfacing directly to libbeostat.

- **beostat** — The beostat command provides a detailed command-line display using the underlying libbeostat library. With no options, beostat lists information for the master node and all compute nodes that is retrieved from /proc/cpuinfo, /proc/meminfo, /proc/loadavg, /proc/net/dev, and /proc/stat. Alternatively, you can use the arguments to select any combination of those statistics.

- **beostatus** — The beostatus cluster monitoring utility uses the underlying libbeostat functionality to display CPU utilization, memory usage, swap usage, disk usage, and network utilization. It defaults to a bar graph X-window GUI, but can display the information in several text formats. For large clusters, a small footprint GUI can be selected, with colored dots depicting the overall status on each node.

- **bpstat** — This displays a text-only snapshot of the current cluster state. The bpstat utility only reports nodes that are part of the BProc unified process space, vs. beostat and beostatus, which report on all nodes (BProc and non-BProc) that execute a sendstats daemon.

- **Ganglia** — Scyld installs the popular Ganglia monitoring package by default, but does not configure it to execute by default. For information on configuring Ganglia, see Ganglia.

- **beoweb** — Beoweb is an optional Web service that can execute on the cluster's master node. Built with Pylons (a Python-based Web framework), beoweb exposes an API for cluster status and remote job submission and monitoring.

4.3.1.2 Monitoring Daemons

Underlying the libbeostat monitoring facility are two daemons: sendstats and recvstats. The recvstats daemon is started by the /etc/rc.d/init.d/clusterware script and only executes on the master node. A sendstats daemon executes on each compute node and sends status information at regular intervals (currently once per second) to the master's recvstats daemon. For more information on the daemon options, see man recvstats and man sendstats, or the Reference Guide.

The optional beoweb service employs the paster daemon on the master node. See beoweb for details.

4.3.2 Using the Data

The outputs from the monitoring utilities can provide insights into obtaining the best performance from your cluster. If you are new to cluster computing, you will want to note the relationship between the different machine resources, including CPU utilization, swap usage, and network utilization. Following are some useful guidelines:

- Low CPU usage with high network traffic might indicate that your system is I/O bound and could benefit from faster network components.

- Low network load and high CPU usage indicate that your system performance could improve with faster CPUs.

- Medium to high swap usage is always bad. This indicates that memory is oversubscribed, and application pieces must be moved to the much slower disk sub-system. This can be a substantial bottleneck, and is a sure sign that additional RAM is needed.
Any of these issues could be helped with application optimization, but sometimes it is more economical to add resources than to change working software.

For best performance of a computational workload, make sure your compute nodes have ample memory for the application and problem set. Also, use diskless compute nodes or configure local disks for scratch file space rather than swap space.

### 4.3.3 beostatus

The `beostatus` GUI display is a Gnome X-window that supports four different types of display generation, all of which can be operated simultaneously. Output in bar graph mode (also known as “Classic” mode) is the default, and is provided by a Gnome/GTK+ GUI display. This display is updated once every 5 seconds by default, but the update rate may be changed using the `-u` option.

You can start `beostatus` by clicking the “blocks” icon on the desktop.

Alternatively, type the command `beostatus` in a terminal window on the master node; you do not need to be a privileged user to use this command.

![beostatus GUI Display (in “Classic” Mode)](image-url)

**Figure 1. beostatus GUI Display (in “Classic” Mode)**
4.3.3.1 beostatus File Menu

The File menu in the beostatus GUI display includes two options, Preferences and Quit, as described below.

Preferences

Selecting Preferences from the File menu displays the Options dialog box (shown below). You can change the values for update rate, master node maximum bandwidth, slave (compute node) maximum bandwidth, and the default display mode.

![Options dialog box](image)

Figure 2. beostatus Preference Options

Quit

Selecting Quit from the File menu closes the beostatus GUI display.

4.3.3.2 beostatus Modes

The Mode menu in the beostatus GUI display allows you to choose between the various display options. Some display options can also be accessed using command line options, including Dots mode, Curses mode, and Text mode. These display options are described in the following sections.

beostatus Dots Mode

Output in Dots mode (beostatus -d) provides a Gnome/GTK+ GUI display. Each node is represented by a colored dot. This output provides a small “footprint”, intended for quick overviews and for situations where the screen size needed for the full display for large clusters is unavailable.

![Dots mode output](image)

Figure 3. beostatus GUI Display in Dots Mode
Following are the color indicators used in Dots mode:

- Red — No access; node statedown
- Yellow — Admin access only; node state unavailable, boot, or error
- Green — Ready; node state up and node load less than/equal 48%
- Blue — Busy; node state up and node load greater than 48%

Note that SMP is considered for node load calculation as load(CPU1) + load(CPU2) > 48%.

**beostatus Curses Mode**

Output in Curses mode (beostatus -c) prints a column header and a line for each node without a linefeed. This continuous output provides a method to monitor the system over text-only connections, such as the installed ssh server. Following is an example of the output in Curses mode:

```
Beostatus - 3.0
Node State CPU 0 Memory Swap Disk Network
-1 up 2.5% 91.7% 0.0% 9.2% 1 kbps
 0 up 0.2% 20.5% 0.0% 25.0% 1 kbps
 1 up 0.1% 20.5% 0.0% 25.0% 1 kbps
 2 up 0.1% 20.5% 0.0% 25.0% 1 kbps
 3 up 0.2% 20.4% 0.0% 25.0% 1 kbps
 4 up 0.1% 20.3% 0.0% 25.0% 1 kbps
 5 up 0.1% 20.3% 0.0% 25.0% 1 kbps
 6 up 0.2% 20.6% 0.0% 25.0% 1 kbps
 7 up 0.1% 20.4% 0.0% 25.0% 1 kbps
```

### 4.3.4 beostat

The beostat utility is a command-line program that provides a text listing of the information from /proc on each node. Following is example output from a single node.

```
============== Node: .0 (index 0) ==============

*** /proc/cpuinfo *** Tue Sep 12 14:38:04 2006
disk processors : 2
vendor_id : AuthenticAMD
cpu family : 15
model : 5
model name : AMD Opteron(tm) Processor 248
stepping : 10
cpu MHz : 2211.355
cache size : 1024 KB
fdiv_bug : no
hlt_bug : no
sep_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 1
```
(continues on next page)
The libbeostat library contains the “C” language functions listed below. You compile with the header files sys/bproc.h and sys/beostat.h, adding the linker commands -lbproc -lbeostat.

beostat-get-cpu-count
beostat-get-name
beostat-get-time
beostat-get-cpuinfo-x86
beostat-get-meminfo
beostat-get-loadavg

(continues on next page)
beostat-get-net-dev
beostat-get-stat-cpu
beostat-get-MHz
beostat-get-statfs-p
beostat-get-last-multicast
beostat-set-last-multicast
beostat-get-cpu-percent
beostat-get-net-rate
beostat-get-disk-usage
beostat-count-idle-cpus
beostat-count-idle-cpus-on-node
beostat-get-avail-nodes-by-id
beostat-is-node-available

4.3.5 bpstat

bpstat displays a text-only snapshot of the current cluster state/configuration:

```
[root@cluster ~] # bpstat
Node(s) Status Mode User Group
16-31 down ---------- root root
0-15 up  ---x--x--x root root
```

You can include the master node in the display, which is especially useful if the master node has non-default access permissions:

```
[root@cluster ~] # bpstat
Node(s) Status Mode User Group
16-31 down ---------- root root
-1 up  ---x--x--x root root
0-15 up  ---x--x--x root root
```

Using the `-p` option, you can view the PID for each user process running on the nodes. You can then pipe the `ps` command into `grep` to get the command string associated with it, such as `ps -aux |grep 8370`. Normal process signaling will work with these PIDs, such as `kill -9 8369`.

```
PID  Node  Ghost
  8367  0  -1
  8368  1  -1
  8369  2  -1
  8370  3  -1
```

See the *Reference Guide* for more details on the command options.
4.3.6 Ganglia

Ganglia is an open source distributed monitoring technology for high-performance computing systems, such as clusters and grids. In current versions of Scyld ClusterWare, Ganglia provides network metrics for the master node, time and string metrics (boottime, machine_type, os_release, and sys_clock), and constant metrics (cpu_num and mem_total). Ganglia uses a web server to display these statistics; thus, to use Ganglia, you must run a web server on the cluster’s master node.

When installing Scyld ClusterWare, make sure the Ganglia package is selected among the package groups to be installed. Once you have completed the Scyld installation and configured your compute nodes, you will need to configure Ganglia as follows:

1. Name your cluster.

   By default, Ganglia will name your cluster “my cluster”. You should change this to match the master node’s hostname. In the file /etc/gmetad.conf, and on or about line 39, change:

   ```
data_source "my cluster" localhost
   ```

   to replace my_cluster with the master’s hostname. Note that Ganglia will not collect or display statistics without at least one entry for data_source.

2. Enable and start the Ganglia Data Collection Service.

   ```
   [root@cluster ~] # chkconfig beostat on
   [root@cluster ~] # systemctl enable xinetd
   [root@cluster ~] # systemctl enable httpd
   [root@cluster ~] # systemctl enable gmetad
   [root@cluster ~] # systemctl restart xinetd
   [root@cluster ~] # systemctl start httpd
   [root@cluster ~] # systemctl start gmetad
   ```


   Note that if you are visiting the web page from a computer other than the cluster’s master node, then you must change localhost to the master node’s hostname. For example, if the hostname is “iceberg”, then you may need to use its fully qualified name, such as http://iceberg.penguincomputing.com/ganglia.

   **Caution**

   The Ganglia graphs that track load (1-, 5-, and 15-minute), the number of CPUs, and the number of processes may appear inaccurate. These graphs are in fact reporting correct statistics, but for the system as a whole rather than just user processes. Scyld draws its statistics directly from system data structures and /proc. It does not take any further steps to interpret or post-process the metrics reported by these data structures.

4.3.7 beoweb

The beoweb service does not execute by default. To enable it:

```
chkconfig beoweb on
```

and then it will start automatically the next time the master node boots. It can be started immediately by doing:

```
[root@cluster ~] # service beoweb start
```
Beoweb exposes an API for cluster status monitoring and remote job submission and monitoring. In its current state, beoweb is best used when paired with PODTools to enable remote job submission. (See the User’s Guide for details about PODTools.) Beoweb does not yet support being viewed with a web browser; rather, it merely provides a web service accessible through APIs. Beoweb supports job submission using the TORQUE resource manager or SGE.

Beoweb is installed in /opt/scyld/beoweb, and the main configuration file, beoweb.ini, is located there. Some key settings to inspect are:

- **host = 0.0.0.0**
  This specifies the interface on which Beoweb will bind/listen. 0.0.0.0 specifies all available interfaces. Use an actual IP address to limit this to a single interface.

- **port = 5000**
  The port number on which beoweb listens. Change to a different port number as needed.

- **ssl_pem = %(here)s/data/beoweb.pem**
  The ssl_pem parameter controls whether or not beoweb uses SSL/TLS encryption for communication. It is strongly encouraged that you use SSL. When beoweb is installed, a temporary PEM file will be created at %(here)s/data/beoweb.pem. This certificate is good for 365 days.

- **auth.use_system_shadow = True**
  The value defaults to True. Unless explicitly disabled, beoweb will read /etc/shadow for user authentication. If this is set to False, you must use auth.auth_file to specify a different list of authorized users.

- **auth.auth_file = %(here)s/data/shadow**
  This file allows for user passwords to be stored independently from the master node’s /etc/shadow file. Currently, beoweb only supports shadow-type login accounts. For example, if you put user credentials in %(here)s/data/shadow and not in /etc/shadow, then that user can access the master node’s beoweb services without being allowed to actually login to the master node. The format for this file is identical to /etc/shadow.

- **stage.jobs_dir = podsh_jobs**
  This names a folder that will be created and used in a user’s home directory for job scripts uploaded through PODTools.

- **stage.port_range = 10000-11000**
  When file uploads and downloads are requested through beoweb using PODTools, the files are transferred through a TCP socket connection. Beoweb opens a socket on the port in the range given in this entry, then sends that port number back to PODTools for use. This range should be chosen such that it does not conflict with other services on your system.

### 4.4 Configuring the Cluster

The Scyld ClusterWare configuration is defined by the contents of several flat ASCII files. Most of these files reside in the /etc/beowulf/ directory. Various ClusterWare scripts (which mostly reside in /usr/lib/beoboot/bin), daemons, and commands read (and some occasionally update) these flat files.

The root user can manipulate the configuration manually using a text editor.
4.4.1 Configuring the Cluster Manually

This section discusses how to configure a cluster. Penguin Computing strongly recommends that the administrator use Manual editing of configuration files, especially the centerpiece /etc/beowulf/config file, should only be done with care, together with sufficient understanding of the ramifications of the manual manipulations.

Caution

If manual edits are made to the config file for a running cluster, then after saving the file, but sure to execute systemctl reload clusterware, which will immediately send a SIGHUP signal to the bpmaster and beoserv daemons that notifies each to re-read the config file.

4.4.1.1 Configuration Files

/etc/beowulf/config

The file /etc/beowulf/config is the principal configuration file for the cluster. The config file is organized using keywords and values, which are used to control most aspects of running the cluster, including the following:

- The name, IP address and netmask of the network interface connected to the private cluster network
- The network port numbers used by ClusterWare for various services
- The IP address range to assign to the compute nodes
- The MAC (hardware) address of each identified node accepted into the cluster
- The node number and IP address assigned to each hardware address
- The default kernel and kernel command line to use when creating a boot file
- A list of kernel modules to be available for loading on compute nodes at runtime
- A list of shared library directories to cache on the compute nodes
- A list of files to prestage on the compute nodes
- Compute node filesystem startup policy
- The name of the final boot file to send to the compute nodes at boot time
- The hostname and hostname aliases of compute nodes
- Compute node policies for handling local disks and filesystems, responding to master node failure, etc.

The following sections briefly discuss some key aspects of the configuration file. See the Reference Guide (or man beowulf-config) for details on the specific keywords and values in /etc/beowulf/config.

Setting the IP Address Range

The IP address range should be kept to a minimum, as all the cluster utilities will loop through this range. Having a few spare addresses is a good idea to allow for growth in the cluster. However, having a large number of addresses that will never be used will be an unnecessary waste of resources.
Identifying New Nodes

When a new node boots, it issues a DHCP request to the network in order to get an IP address assigned to it. The master’s beoserv detects these DHCP packets, and its response is dependent upon the current nodeassign policy. With a default append policy, beoserv appends a new node entry to the end of the /etc/beowulf/config file. This new entry identifies the node’s MAC address(es), and the relative ordering of the node entry defines the node’s number and what IP address is assigned to it. With a manual policy, beoserv appends the new node’s MAC address to the file /var/beowulf/unknown_addresses, and then assigns a temporary IP address to the node that is outside the iprange address range and which does not integrate this new node into the cluster. It is expected that the cluster administrator will eventually assign this new MAC address to a cluster node, giving it a node entry with an appropriate position and node number. Upon cluster restart, when the node reboots (after a manual reset or an IPMI powercycle), the node will assume its assigned place in the cluster. With a locked policy, the new node gets ignored completely: no recording of its MAC address, and no IP address assignment.

Assigning Node Numbers and IP Addresses

Two config file keywords control the assignment of IP addresses to compute nodes on the private cluster network: nodes and iprange. The nodes keyword specifies the max number of compute nodes, and the iprange specifies the range of IP addresses that are assigned to those compute nodes.

By default and in general practice, node numbers and IP addresses are assigned to the compute nodes in the order that their node entries appear in the config file, beginning with node 0 and the first IP address specified by the iprange entry in the config file. For example, the config file entries:

```
nodes 8
iprange 10.20.30.100 10.20.30.107
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```

specify a network that contains a maximum of eight nodes, with four nodes currently known, and with an IP address range that falls between the 10.20.30.100 lower bound and the 10.20.30.107 upper bound. Here the node with MAC address 00:01:02:03:04:1C is node 2 and will be assigned an IP address 10.20.30.102.

ClusterWare treats the upperbound IP address as optional, so all that is necessary to specify is:

```
nodes 8
iprange 10.20.30.100
```

and ClusterWare calculates the upperbound IP address. This is especially useful when dealing with large nodes counts, e.g.:

```
nodes 1357
iprange 10.20.30.100
```

when it becomes increasingly clumsy for the cluster administrator to accurately calculate the upperbound address.

An optional node number can explicitly specify an override node number:

```
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```
explicitly (and redundantly) specifies the node 2 numbering. Alternatively:

```
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```

explicitly names that node as node 5 with IP address 10.20.30.105, and the next node (with MAC address
```
00:01:02:03:04:1D
```
will now be node 6 with IP address 10.20.30.106.

In another variation, commenting-out the MAC address(es) leaves a node numbering gap for node 2, and MAC address
```
00:01:02:03:04:1D
```
continues to be known as node 3:

```
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node # 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```

However, if the node with that commented-out MAC address 00:01:02:03:04:1C does attempt to PXE boot, then
```
beoserv
```
assigns a new node number (4) to that physical node and automatically appends a new node entry to the list
(assuming the nodeassign policy is append, and assuming the iprange and nodes entries allow room for expansion).
This appending results in:

```
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node # 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```

If you want to have beoserv ignore that physical node and keep the remaining nodes numbered without change, then
use the keyword off:

```
node 00:01:02:03:04:1A 00:01:02:03:05:2A
node 00:01:02:03:04:1B 00:01:02:03:05:2B
node off 00:01:02:03:04:1C 00:01:02:03:05:2B
node 00:01:02:03:04:1D 00:01:02:03:05:2B
```

A node entry can identify itself as a non-Scyld node and can direct beoserv to respond to the node in a variety of
ways, including telling the node to boot from a local harddrive, or provisioning the node with specific kernel and initrd
images. See Managing Non-Scyld Nodes for details.

**Caching Shared Libraries**

To add a shared library to the list of libraries cached on the compute nodes, specify the pathname of the individual file
or the pathname of the entire directory in which the file resides using the libraries directive. An open() syscall on a
compute node to open a file thus named, or to open a file that resides in a named directory, will cause that file to be
pulled from the master node to the compute node and saved in the local RAM filesystem.

The prestage directive names specific files to be pulled onto each compute node at node boot time. If a file pathname
resides in one of the libraries directories, then BProc’s filecache functionality pulls the file from the master node.
Otherwise, the specified file is pushed from the master to the compute node at startup, with directories created as
needed.
Specifying node names and aliases

The `nodename` keyword in the master's `/etc/beowulf/config` affects the behavior of the ClusterWare NSS. Using the `nodename` keyword, one may redefine the primary host-name of the cluster, define additional hostname aliases for compute nodes, and define additional hostname (and hostname aliases) for entities loosely associated with the compute node’s cluster position.

```
nodename [name-format] <IPv4 Offset or base> <netgroup>
```

The presence of the optional IPv4 argument defines if the entry is for “compute nodes” (i.e. the entry will resolve to the ‘dot-number’ name) or if the entry is for non-cluster entities that are loosely associated with the compute node. In the case where there is an IPv4 argument, the `nodename` keyword defines an additional hostname name that maps to an IPv4 address loosely associated with the node number. In case where IPv4 argument is present, the `nodename` keyword defines hostname and hostname aliases for the clustering interface (i.e. the compute nodes). Subsequent `nodename` entries without an IPv4 argument specify additional hostname aliases for compute nodes. In either case, the format string must contain a conversion specification for node number substitution. The conversion specification is introduced by a ‘%’. An optional following digit in the range 1..5 specifies a zero-padded minimum field width. The specification is completed with an ‘N’. An unspecified or zero field width allows numeric interpretation to match compute node host names. For example, `n%N` will match `n23`, `n+23`, and `n0000023`. By contrast, `n%3N` will only match `n001` or `node023`, but not `n1` or `n23`.

Compute node command-line options

The `kernelcommandline` directive is a method of passing various options to the compute node’s kernel and to Beowulf on the node. There are a large number of different command line options that you can employ. This section covers some of them.

Some options are interpreted by the kernel on the compute node and ignored by Beowulf:

- **apic**  This option turns on APIC support on the compute node. APIC is the newer of two different mechanisms Intel provides for invoking interrupts. It works better with SMP systems than the older mechanism, called XT-PIC. However, not every motherboard and chipset works correctly with APIC, so this option is disabled by default to avoid problems for those machines that do not support it.

  If you find that your cluster nodes kernel panic or crash immediately upon boot, you probably want to turn off APIC by specifying `noapic` in the command line options. If you have many devices that generate interrupts (such as hard disk controllers, network adapters, etc.) you may want to try turning on APIC to see if there is any performance advantage for your cluster.

- **panic=**<seconds>  This option allows you to specify how many seconds the kernel should wait to reboot after a kernel panic. For example, if you specify `panic=60`, then the kernel will wait 60 seconds before rebooting. Note that Beowulf automatically adds `panic=30` to final boot images.

- **apm=**<action>  This option allows you to specify APM options on the compute node. Acceptable <action> values are `on` (to turn APM completely on), `off` (to turn it completely off), `debug` (to turn on debugging), and `power-off` (to turn on only the power-off part of APM).

  APM is not SMP-safe in the kernel; it will auto-disable itself if turned completely on for an SMP box. However, the `power-off` part of APM is SMP safe; thus, if you want to be able to power-off SMP boxes, you can do so by specifying `apm=power-off`. Note that `apm=power-off` is specified in the default `kernelcommandline` directive.

- **console=**<device>,<options>  This option is used to select which device(s) to use for console output. For `device` use `tty0` for the foreground virtual console, `ttyX` (e.g., `tty1`) for any other virtual console, and `ttyXs` (e.g., `ttyS0`) for a serial port.

  For the serial port, `<options>` defines the baud rate/parity/bits of the port in the format “BBBBP”, where “BBBB” is the speed, “P” is parity (n/o/e), and “N” is bits. The default setting is 9600n8, and the maximum baud
rate is 115200. For example, to use the serial port at the maximum baud rate, specify `console=ttyS0,115200n8r`

Other options are interpreted by Beowulf on the compute node:

**rootfs_size=<size>** A compute node employs a RAM-based root filesystem for local non-persistent storage, typically used to contain BProc’s filecache libraries and other files, the `/tmp` directory, and other directories that are not mounted using some variety of global storage (e.g., NFS or Panfs) or on local harddrives. This `tmpfs` root filesystem consumes physical memory only as needed, which commonly is about 100- to 200-MBytes unless user workloads impose greater demands on (for example) `/tmp` space. However, by default the rootfs is allowed to grow to consume a maximum of 50% of physical memory, which has the potential of allowing users to consume (perhaps inadvertently) an excessive amount of RAM that would otherwise be available to applications’ virtual memory needs.

This 50% default can be overridden by the judicious use of the `<size>` option, where `<size>` can be expressed as numeric bytes, megabytes (appending “m” or “M”), or gigabytes (appending “g” or “G”), or as a percentage of total physical memory (appending numeric value and “%”). Examples:

```
rootfs_size=2048m
rootfs_size=1G
rootfs_size=15%
```

Note that this override is rarely needed, and it must be utilized with care. An inappropriately constrained root filesystem will cripple the node, just as an inadequate amount of physical memory that is available for virtual memory will trigger Out-Of-Memory failures. The cluster administrator is encouraged to limit user filesystem usage in other ways, such as declaring `/etc/security/limits.conf` limits on the max number of open files and/or the maximum filesize.

**rootfs_timeout=<seconds>; getfile_timeout=<seconds>** The `beoclient` daemon on each compute node manages the early boot process, such as using tftp to read the kernel image and initrd files from the master node’s `beoserv` daemon, and using tcp to read the initial root filesystem image (`rootfs`) from beoserv. After the node boots, BProc’s filecache functionality on the compute node also uses tcp to read files from the master, as needed by applications.

The default timeout for these tcp reads is 30 seconds. If this timeout is too short, then add one of these options to the `kernelcommandline` to override the default. The option `getfile_timeout` overrides the timeout for all beoclient tcp read operations. The option `rootfs_timeout` overrides the timeout only for the tcp read of the root filesystem at node boot time.

**syslog_server=<IPaddress>** By default, a compute node forwards its kernel messages and syslog messages back to the master node’s `syslog` or `rsyslog` service, which then appends these log messages to the master’s `/var/log/messages` file. Alternatively, the cluster administrator may choose to instead forward these compute node log messages to another server by using the `syslog_server` option to identify the <IPaddress> of that server. This should be an IPv4 address, e.g., `syslog_server=10.20.30.2`.

Scyld ClusterWare automatically configures the master node’s log service to handle incoming log messages from remote compute nodes. However, the cluster administrator must manually configure the alternate syslog server:

For the `syslog` service (Scyld ClusterWare 4 and 5), edit `/etc/sysconfig/syslog` on the alternate server to add “-r -x” to the variable SYSLOGD_OPTIONS.

For the `rsyslog` service (Scyld ClusterWare 6), edit `/etc/sysconfig/rsyslog` on the alternate server to add “-x” to the variable SYSLOGD_OPTIONS, and edit `/etc/rsyslog.conf` to un-comment the following lines to expose them, i.e., just as Scyld ClusterWare has done in the master node’s `/etc/rsyslog.conf` file:

```
$ModLoad imudp.so
$UDPServerRun 514
```

Finally, restart the service on both the master node and the alternate syslog server before restarting the cluster.
**legacy_syslog=<num>**  The legacy behavior of the the compute node’s syslog handling has been to introduce a date-time string to the message text, then forward the message to the syslog server (typically on the master node), which would add its own date-time string. This redundant timestamp violates the RFC 3164 format standard, and recent ClusterWare releases strips the compute node’s timestamp before sending the text to the master server. If for some reason a local cluster administrator wishes to revert to the previous behavior, then add `legacy_syslog=1`. The default is `legacy_syslog=0`.

### Specifying kernel modules for use on compute nodes

Each `bootmodule` entry identifies a kernel module to be added to the initrd that is passed to each compute node at boot time. These entries typically name possible Ethernet drivers used by nodes supplied by Penguin Computing. If the cluster contains nodes not supplied by Penguin Computing, then the cluster administrator should examine the default list and add new `bootmodule` entries as needed.

At boot time, Beowulf scans the node’s PCI bus to determine what devices are present and what driver is required for each device. If the specified driver is named by a `bootmodule` entry, then Beowulf loads the module and all its dependencies. However, some needed modules are not found by this PCI scan, e.g., those used to manage specific filesystem types. These modules require adding an additional configuration file entry: `modprobe`. For example:

```
modprobe xfs
```

Note that each named `modprobe` module must also be named as a `bootmodule`.

You may also specify module-specific arguments to be applied at module load time, e.g.,

```
modarg forcedeth optimization_mode=1
```

RHEL7 introduced externally visible discrete firmware files that are associated with specific kernel software drivers. When `modprobe` attempts to load a kernel module that contains such a software driver, and that driver determines that the controller hardware needs one or more specific firmware images (which are commonly found in `/lib/firmware`), then the kernel first looks at its list of built-in firmware files. If the desired file is not found in that list, then the kernel sends a request to the `udevd` daemon to locate the file and to pass its contents back to the driver, which then downloads the contents to the controller. This functionality is problematic if the kernel module is an `/etc/beowulf/config bootmodule` and is an Ethernet driver that is necessary to boot a particular compute node in the cluster. The number of `/lib/firmware/` files associated with every possible `bootmodule` module is too large to embed into the initrd image common to all compute nodes, as that burdens every node with a likely unnecessarily oversized initrd to download. Accordingly, the cluster administrator must determine which specific firmware file(s) are actually required for a particular cluster and are not yet built-in to the kernel, then add `firmware` directive(s) for those files.

A `bootmodule` firmware problem exhibits itself as a compute node which does not boot because the needed Ethernet driver cannot be `modprobe`d because it cannot load a specific firmware file. After a timeout waiting for `udevd` to unsuccessfully find the file, the compute node typically reboots - endlessly, as it continues to be unable to load the needed firmware file.

The cluster administrator can use the `firmware` directive to add specific firmware files to the compute node initrd, as needed. The compute node kernel writes the relevant firmware filename information to its console, e.g. a line of the form:

```
Failed to load firmware "bnx2/bnx2-mips-06-6.2.1.fw"
```

Ideally, the administrator gains access to the node’s console to see the specific filename, then adds a directive to `/etc/beowulf/config`:

```
firmware bnx2/bnx2-mips-06-6.2.1.fw
```

and rebuilds the initrd:

---

4.4. Configuring the Cluster 123
[root@cluster ~] # systemctl reload clusterware

(Note: *reload*, not *restart*)

If the node continues to fail to boot, then the failure is likely due to another missing firmware file. Check the node’s console output again, and add the specified file to the *firmware* directive.

If the cluster administrator cannot easily see the node’s console output to determine what firmware files are needed, then if the administrator knows the likely *bootmodule* module culprit, then the administrator can brute-force every known firmware file for that module using a directive of the form:

```
firmware bnx2
```

that names an entire */lib/firmware/* subdirectory. This will likely create a huge *initrd* that will (if the correct *bootmodule* module is specified) successfully boot the compute node. The administrator should then examine the node’s syslog output, which is typically seen in */var/log/messages*, to determine the specific individual firmware filenames that were actually needed, and then the administrator replaces the subdirectory name with the now-known specific firmware filenames. Subsequently, the cluster administrator should contact Penguin Computing Support to inform us what those needed firmware files are, so that we can build-in these files into future kernel images and thus allow the cluster administrator to remove the *firmware* directives and thus reduce the *initrd* size, which contains not only the firmware images, but additionally includes various executable binaries and libraries that are only needed for this dynamic *udevd* functionality.

/\etc/beowulf/fdisk

The */etc/beowulf/fdisk* directory is created by the *beofdisk* utility when it evaluates local disks on individual compute nodes and creates partition tables for them. For each unique drive geometry discovered among the local disks on the compute nodes, *beofdisk* creates a file within this directory. The file naming convention is “head:ccc;hhh;sss”, where “ccc” is the number of cylinders on the disk, “hhh” is the number of heads, and “sss” is the number of sectors per track.

These files contain the partition table information as read by *beofdisk*. Normally, these files should not be edited by hand.

You may create separate versions of this directory that end with the node number (for example, */etc/beowulf/fdisk.3*). The master’s BeoBoot software will look for these directories before using the general */etc/beowulf/fdisk* directory.

For more information, see the section on *beofdisk* in the *Reference Guide*.

/\etc/beowulf/fstab

This is the filesystem table for the mount points of the partitions on the compute nodes. It should be familiar to anyone who has dealt with an */etc/fstab* file in a standard Linux system, though with a few Scyld ClusterWare extensions. For details, see the *Reference Guide* or execute `man beowulf-fstab`.

You may create separate node-specific versions by appending the node number, e.g., */etc/beowulf/fstab.3* for node 3. The master’s beoboot *node_up* script looks first for a node_specific *fstab*.\(N\) file, then if no such file exists will use the default */etc/beowulf/fstab* file.

**Caution**

On compute nodes, NFS directories must be mounted using either the IP address or the `$MASTER` keyword; the master node’s hostname cannot be used. This is because */etc/beowulf/fstab* is evaluated before the Scyld ClusterWare name service is initialized, which means hostnames cannot be resolved on a compute node at that point.
/etc/beowulf/backups/

This directory contains time-stamped backups of older versions of various configuration files, e.g., /etc/beowulf/config and /etc/beowulf/fstab, to assist in the recovery of a working configuration after an invalid edit.

/etc/beowulf/conf.d/

This directory contains various configuration files that are involved when booting a compute node. In particular, the node_up script pushes the master node’s /etc/beowulf/conf.d/limits.conf to each compute node as /etc/security/limits.conf, and pushes /etc/beowulf/conf.d/sysctl.conf to each compute node as /etc/sysctl.conf. If /etc/beowulf/conf.d/limits.conf does not exist, then node_up creates an initial file as a concatenation of the master node’s /etc/security/limits.conf plus all files in the directory /etc/security/limits.d/. Similarly, node_up creates an initial /etc/beowulf/conf.d/sysctl.conf (if it doesn’t already exist) as a copy of the master’s /etc/sysctl.conf. The cluster administrator may subsequently modify these initial “best guess” configuration files as needed for compute nodes.

4.4.1.2 Command Line Tools

bpstat

The command bpstat can be used to quickly check the status of the cluster nodes and/or see what processes are running on the compute nodes. See the Reference Guide for details on usage.

bpctl

To reboot or set the state of a node via the command line, one can use the bpctl command. For example, to reboot node 5:

[root@cluster ~] # bpctl -S 5 -R

As the administrator, you may at some point have reason to prevent other users from running new jobs on a specific node, but you do not want to shut it down. For this purpose we have the unavailable state. When a node is set to unavailable non-root users will be unable to start new jobs on that node, but existing jobs will continue running. To do this, set the state to unavailable using the bpctl command. For example, to set node 5 to unavailable:

[root@cluster ~] # bpctl -S 5 -s unavailable

node_down

If you are mounting local filesystems on the compute nodes, you should shut down the node cleanly so that the filesystems on the harddrives stay in a consistent state. The node_down script in /usr/lib/beoboot/bin does exactly this. It takes two arguments; the first is the node number, and the second is the state to which you want the node to go. For example, to cleanly reboot node 5:

[root@cluster ~] # /usr/lib/beoboot/bin/node_down 5 reboot

Alternatively, to cleanly power-off node 5:

[root@cluster ~] # /usr/lib/beoboot/bin/node_down 5 pwroff
The node_down script works by first setting the node’s state to unavailable, then remounting the filesystems on the compute node read-only, then calling bpctl to change the node state. This can all be done by hand, but the script saves some keystrokes.

To configure node_down to use IPMI, set the ipmi value in /etc/beowulf/config to enabled as follows:

```
[root@cluster ~] # beoconfig ipmi enabled
```

### 4.4.1.3 Configuring CPU speed/power for Compute Nodes

Modern motherboards and processors support a degree of administrator management of CPU frequency within a range defined by the motherboard’s BIOS. Scyld ClusterWare provides the /etc/beowulf/init.d/30cpuspeed script and its associated /etc/beowulf/conf.d/cpuspeed.conf configuration file to implement this management for compute nodes. The local cluster administrator is encouraged to review the cpuspeed.conf config file’s section labeled Scaling governor values and potentially adjust the environment variable SCALINGGOV as desired, and then to enable the 30cpuspeed script:

```
[root@cluster ~] # beochkconfig 30cpuspeed on
```

The administrator should also ensure that no other cpuspeed or cpupower script is enabled for compute nodes.

In brief, the administrator can choose among four CPU scaling governor settings:

- **performance**, which directs the CPUs to execute at the maximum frequency supported by the motherboard and processor, as specified by the motherboard BIOS.

- **powersave**, which directs the CPUs to execute at the minimum frequency supported by the motherboard and processor.

- **ondemand**, which directs the kernel to adjust the CPU frequency between the minimum and maximum. An idle CPU executes at the minimum. As a load appears, the frequency increases relatively quickly to the maximum, and if and when the load subsides, then the frequency decreases back to the minimum. This is the default setting.

- **conservative**, which similarly directs the kernel to adjust the CPU frequency between the minimum and maximum, albeit making those adjustments with somewhat longer latency than is done for ondemand.

The upside of the performance scaling governor is that applications running on compute nodes always enjoy the maximum CPU frequencies that are supported by the node hardware. The downside is that even idle CPUs consume that same maximum power and thus generate maximum heat. For the scaling governors performance, ondemand, and conservative, a computebound workload drives the CPU frequencies (and power and heat) to the maximum, and thus computebound application performance will exhibit little or no difference among those governors. However, a workload of rapid context switching and frequent idle time will show perhaps 10-20% lower performance for ondemand versus performance, and possibly an even larger decline with conservative. The powersave governor is typically only employed when a need to minimize the cluster power consumption and/or minimize thermal levels outweighs a need to achieve maximum performance.

A broader discussion can be found in the /usr/share/doc/kernel-doc-2.6.32/Documentation/cpu-freq/documents, e.g., governors.txt. Install the RHEL7 or CentOS7 base distribution’s kernel-doc package to access these documents.
4.4.1.4 Adding New Kernel Modules

The modprobe command uses /usr/lib/`uname -r`/modules.dep.bin to determine the pathnames of the specified kernel module and that module's dependencies. The depmod command builds the human-readable modules.dep and the binary module.dep.bin files, and it should be executed on the master node after installing any new kernel module.

Executing modprobe on a compute node requires additional caution. The first use of modprobe retrieves the current modules.dep.bin from the master node using bproc's filecache functionality. Since any subsequent depmod on the master node rebuilds modules.dep.bin, then a subsequent modprobe on a compute node will only see the new modules.dep.bin if that file is copied to the node using bpcp, or if the node is rebooted and thereby silently retrieves the new file.

In general, you should not execute depmod on a compute node, since that command will only see those few kernel modules that have previously been retrieved from the master node, which means the node's newly built modules.dep.bin will only be a sparse subset of the master node's full module.dep.bin. Bproc's filecache functionality will always properly retrieve a kernel module from the master node, as long as the node's module.dep.bin properly specifies the pathname of that module, so the key is to have the node's module.dep.bin be a current copy of the master's file.

Many device drivers are included with Scyld ClusterWare and are supported out-of-the-box for both the master and the compute nodes. If you find that a device, such as your Ethernet adapter, is not supported and a Linux source code driver exists for it, then you will need to build the driver modules for the master.

To do this, you will need to install the RPM of kernel source code (if you haven't already done so). Next, compile the source code using the following extra GNU C Compiler (gcc) options.

```
-D__BOOT_KERNEL_SMP=1 -D__BOOT_KERNEL_UP=0
```

The compiled modules must be installed in the appropriate directories under /lib/modules. For example, if you are currently running under the 2.6.9-67.0.4.ELsmp kernel version, the compiled module for an Ethernet driver would be put in the following directory:

```
/lib/modules/2.6.9-67.0.4.ELsmp/kernel/drivers/net
```

Any kernel module that is required to boot a compute node, e.g., most commonly the Ethernet driver(s) used by compute nodes, needs special treatment. Edit the config file /etc/beowulf/config to add the name of the driver to the bootmodule list; you can add more bootmodule lines if needed. See Compute Node Boot Options.

Next, you need to configure how the device driver gets loaded. You can set it up so that the device driver only loads if the specific device is found on the compute node. To do this, you need to add the PCI vendor/device ID pair to the PCI table information in the /usr/share/hwdata/pcitable file. You can figure out what these values are by using a combination of lspci and lspci -n.

So that your new kernel module is always loaded on the compute nodes, include the module in the initial RAM disk by adding a modprobe line to /etc/beowulf/config. The line should look like the following:

```
modprobe <module>
```

where <module> is the kernel module in question.

Finally, you can regenerate the BeoBoot images by running systemctl reload clusterware. For more details, see Compute Node Boot Options.
4.4.1.5 Accessing External License Servers

To configure the firewall for accessing external license servers, enable `ipforward` in the `/etc/beowulf/config` file. The line should read as follows:

```
ipforward yes
```

You must then reboot the compute nodes and restart the cluster services. To do so, run the following two commands as root in quick succession:

```
[root@cluster ~] # bpctl -S all -R
[root@cluster ~] # systemctl restart clusterware
```

Tip

If IP forwarding is enabled in `/etc/beowulf/config` but is still not working, then check `/etc/sysctl.conf` to see if it is disabled.

Check for the line “net.ipv4.ip_forward = 1”. If the value is set to 0 (zero) instead of 1, then IP forwarding will be disabled, even if it is enabled in `/etc/beowulf/config`.

4.4.1.6 Configuring SSH for Remote Job Execution

Most applications that leverage `/usr/bin/ssh` on compute nodes can be configured to use `/usr/bin/rsh`. In the event that your application requires SSH access to compute nodes, ClusterWare provides this ability through `/etc/beowulf/init.d/81sshd`. To start `sshd` on compute nodes, enable the `81sshd` script and reboot your nodes:

```
[root@cluster ~] # beochkconfig 81sshd on
[root@cluster ~] # bpctl -S all -R
```

When each node boots, 81sshd starts sshd on the node, and the master’s root user will be able to SSH to a compute node without a password, e.g.:

```
[root@cluster ~] # ssh n0 ls
```

By default, compute node sshd daemons do not allow for password-based authentication – only key-based authentication is available – and only the root user’s SSH keys have been configured.

If a non-root user needs SSH access to compute nodes, the user’s SSH keys will need to be configured. For example, create a DSA key using `ssh-keygen`, and hit Enter when prompted for a password if you want password-less authentication:

```
[user1@cluster ~] $ ssh-keygen -t dsa
```

Since the master’s `/home` directory is mounted (by default) as `/home` on the compute nodes, just copy the public key to `~/.ssh/authorized_keys`:

```
[user1@cluster ~] $ cp -a ~/.ssh/id_dsa.pub ~/.ssh/authorized_keys
```

Now the user can run commands over SSH to any node using shared key authentication:

```
[user1@cluster ~] $ ssh n0 date
```

If you wish to modify `sshd`’s settings, you can edit `/etc/beowulf/conf.d/sshd_config` and then reboot the nodes. Node-specific sshd configuration settings can be saved as `/etc/beowulf/conf.d/sshd_config.$NODE`. 
Client behavior for SSH on the nodes can be adjusted by editing the global /etc/beowulf/conf.d/ssh_config or a node-specific /etc/beowulf/conf.d/ssh_config.$NODE. This SSH client configuration will only be useful when using SSH from node to node. For example:

```
[user1@cluster ~] $ ssh n0 ssh n1 ls
```

Note that /etc/beowulf/conf.d/sshd_config and ssh_config only affect SSH behavior on compute nodes. The master’s SSH configuration will not be affected.

### 4.4.1.7 Interconnects

There are many different types of network fabric one can use to interconnect the nodes of your cluster. The least expensive and most common is Fast (100Mbps) and Gigabit (1000Mbps) Ethernet. Other cluster-specific network types, such Infiniband, offer lower latency, higher bandwidth and features such as RDMA (Remote Direct Memory Access).

**Ethernet**

Switching fabric is always the most important (and expensive) part of any interconnected sub-system. Ethernet switches with up to 48 ports are extremely cost effective; however, anything larger becomes expensive quickly. Intelligent switches (those with software monitoring and configuration) can be used effectively to partition sets of nodes into separate clusters using VLANs; this allows nodes to be easily reconfigured between clusters if necessary.

**Adding a New Ethernet Driver**

Drivers for most Ethernet adapters are included with the Linux distribution, and are supported out of the box for both the master and the compute nodes. If you find that your card is not supported, and a Linux source code driver exists for it, you need to compile it against the master’s kernel, and then add it to the cluster config file using the `bootmodule` keyword. See the Reference Guide for a discussion on the cluster config file.

For details on adding new kernel modules, see *Adding New Kernel Modules*.

**Gigabit Ethernet vs. Specialized Cluster Interconnects**

Surprisingly, the packet latency for Gigabit Ethernet is approximately the same as for Fast Ethernet. In some cases, the latency may even be slightly higher, as the network is tuned for high bandwidth with low system impact utilization. Thus Gigabit Ethernet will not give significant improvement over Fast Ethernet to fine-grained communication-bound parallel applications, where specialized interconnects have a significant performance advantage.

However, Gigabit Ethernet can be very efficient when doing large I/O transfers, which may dominate the overall runtime of a system.

4.4. Configuring the Cluster 129
Other Interconnects

Infiniband is a new, standardized interconnect for system area networking. While the hardware interface is an industry standard, the details of the hardware device interface are vendor specific and change rapidly. Contact Scyld Customer Support for details on which Infiniband host adapters and switches are currently supported.

With the exception of unique network monitoring tools for each, the administrative and end user interaction is unchanged from the base Scyld ClusterWare system.

4.5 Remote Administration and Monitoring

Scyld ClusterWare provides a variety of tools for remotely monitoring and administering clusters. These include traditional shell and X window based tools, along with web based tools. Some utilities are available for users to monitor the system, while others are for administrators to configure the cluster.

4.5.1 Command Line Tools

The RHEL CentOS base distribution includes openssh. This tool allows you to securely ssh into your master node and manually edit /etc/beowulf/config, and modify /etc/beowulf/init.d/ scripts, /etc/beowulf/conf.d/ configuration files, and base distribution configuration files on the master node. For more information on the configuration files and command line utilities, see Configuring the Cluster Manually.

4.5.2 X Forwarding

SSH can also be configured to do X forwarding, which allows the GUI applications to be run on a remote machine. This allows you to use the full functionality of the convenient graphics tools, but can be slow, especially if the connection to the cluster is not via a local area network. In order to activate X forwarding, you may need to use the -X option to ssh from your client. Once the X forwarding is setup, you can use any of the GUI tools described throughout this manual.

4.6 Managing Users on the Cluster

In order for someone to gain access to a cluster, s/he must first be given a user account. The cluster administrator can manage user accounts with the same tools that are available with most Linux distributions. User access to cluster resources can also be controlled by the cluster administrator.

This chapter discusses the tools and commands for managing user accounts and controlling access to cluster resources.

4.6.1 Managing User Accounts

4.6.1.1 Adding New Users

The useradd command enables you to add a new user to the system. This command takes a single argument, which is the new user’s login name:

```
[root@cluster ~] # useradd <username>
```
This command also creates a home directory named /home/<username>.

After you add the user, give them a default password using the `passwd` command so that they will be able to log in. This command takes a single argument, which is the username:

```
[root@cluster ~] # passwd <username>
```

**Tip**

It is good practice to give each user their own unique home directory.

### 4.6.1.2 Removing Users

To remove a user from your cluster, use the `userdel` command. This command takes a single argument, which is the username:

```
[root@cluster ~] # userdel <username>
```

By default, `userdel` does not remove the user’s home directory. To remove the home directory, include the `-r` option in the command:

```
[root@cluster ~] # userdel -r <username>
```

**Tip**

The `userdel` command will never remove any files that are not in the user’s home directory. To fully remove all of a user’s files, remove the user’s mail file from `/var/spool/mail/`, as well as any files the user may have in `/tmp/`, `/var/tmp/`, and any other directories to which the user had write permissions.

### 4.6.2 Managing User Groups

In addition to user accounts, you can also create user groups. Groups can be very powerful, as they allow you to assign resources to an arbitrary set of users. Groups are typically used for file permissions. However, you can also utilize groups to assign nodes to a specific set of users, thereby limiting which users have access to certain nodes. This section covers creating and modifying groups.

#### 4.6.2.1 Creating a Group

Before you can add users to a group, you must first create the group. Groups can be created with the `groupadd` command. This command takes a single argument, which represents the name of the group:

```
[root@cluster ~] # groupadd <groupname>
```

#### 4.6.2.2 Adding a User to a Group

Use the `usermod` command To add a user to a group. This command requires you to list all the groups the user should be a member of. To avoid accidentally removing any of the user’s groups, first use the `groups` command to get a list of the user’s current groups. The following example shows how to find the groups for a user named Smith:

```
[root@cluster ~] # groups smith
smith : smith src
```

After getting a list of the user’s current groups, you can then add them to new groups, for example:
4.6.2.3 Removing a Group

To remove a group, run the `groupdel` command with the groupname as an argument:

```
[root@cluster ~] # groupdel <groupname>
```

4.6.3 Controlling Access to Cluster Resources

By default, anyone who can log into the master node of the cluster can send a job to any compute node. This is not always desirable. You can use *node ownership* and *mode* to restrict the use of each node to a certain user or group, including restricting compute node access to the master node.

4.6.3.1 What Node Ownership Means

Each node (including the master node) has *user*, *group* and *mode* bits assigned to it; these indicate who is allowed to run jobs on that node. The *user* and *group* bits can be set to any user ID or group ID on your system. In addition, the use of a node can be unrestricted by setting the *user* and *group* to “root”.

For the BProc unified process space, the node permissions “root” and “any” are equivalent. Node user access follows the normal Linux convention, i.e., the most restrictive access rule is the one used. Some examples:

- user “root”, group “test”, mode 101 (u=1, g=0, o=1) — Users in the group “test” will not be able to access the node.
- user “tester”, group “root”, and mode 011 (u=0, g=1, o=1) — The user “tester” will not be able to access the node.
- user “tester”, group “test”, and mode 110 (u=0, g=1, o=1) — The user “tester” and users in the group “test” are the only non-root users able to access the node.

**Tip**

In Linux systems, “other” is defined as anyone not listed in the user or group.

4.6.3.2 Checking Node Ownership

Display the current node access state by running the `bpstat` command:

```
[root@cluster ~] # bpstat -M
Node(s) Status Mode User Group
16-31 down ---------- root root
-1 up ---x--x--x root root
0-15 up ---x--x--x root root
```

The “User” column shows the user for each node and the “Group” column shows the group for each node. This display shows a cluster with default access permissions.
4.6.3.3 Setting Node Ownership

You can set node ownership with the `bpctl` command. Use the `-S` option to specify which node to change. Use either the `-u` option to change the user, `-g` option to change the group, or `-m` to change the mode. The only bit utilized for the mode is the `execute` bit. Following are some examples.

- The following sets the user for node 5 to `root`:

  ```
  [root@cluster ~] # bpctl -S 5 -u root
  ```

- The following sets all the compute nodes to be in the group `beousers`:

  ```
  [root@cluster ~] # bpctl -S all -g beousers
  ```

- The following allows only the group `beousers` to access the compute nodes:

  ```
  [root@cluster ~] # bpctl -S all -m 010 -g beousers
  ```

- The following disallows non-root users to execute on the master:

  ```
  [root@cluster ~] # bpctl -M -m 0110
  ```

For example:

```
[root@cluster ~] # bpctl -M -m 0110
[root@cluster ~] # bpctl -S 0-3 -g physics
[root@cluster ~] # bpstat -M

<table>
<thead>
<tr>
<th>Node(s)</th>
<th>Status</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-31</td>
<td>down</td>
<td>------</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>5</td>
<td>up</td>
<td>-----X-</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>1-3</td>
<td>up</td>
<td>----X---</td>
<td>root</td>
<td>physics</td>
</tr>
<tr>
<td>4-15</td>
<td>up</td>
<td>----X---X</td>
<td>root</td>
<td>root</td>
</tr>
</tbody>
</table>
```

See the *Reference Guide* for additional details on `bpctl`.

Using `bpctl` does not permanently change the node ownership settings. Whenever the master node reboots or `systemctl restart clusterware` reboots the cluster, the node ownership settings revert to the default of full, unrestricted access, or to the optional override settings specified by the `nodeaccess` directive(s) in the `/etc/beowulf/config` file. To make permanent changes to these settings, you must edit this file. For example, to make the above setting persistent, add the `nodeaccess` entries:

```
nodeaccess -M -m 0110
nodeaccess -S 0-3 -g physics
```

The *Reference Guide* and `man beowulf-config` provides details for the `/etc/beowulf/config` file.
4.7 Job Batching

For Scyld ClusterWare, the default installation includes both the TORQUE resource manager and the Slurm workload manager, each providing users with an intuitive interface for remotely initiating and managing batch jobs on distributed compute nodes.

ClusterWare TORQUE is a customized redistribution of Open Source software that derives from Adaptive Computing Enterprises, Inc. https://www.adaptivecomputing.com/products/opensource/torque. TORQUE is an Open Source tool based on standard OpenPBS. ClusterWare Slurm is a redistribution of Open Source software that derives from https://slurm.schedmd.com, and the associated Munge package derives from http://dun.github.io/munge/.

Both TORQUE and Slurm are installed by default, although only one job manager can be enabled at any one time. See Enabling TORQUE or Slurm below, for details. See the User’s Guide for general information about using TORQUE or Slurm. See Managing Multiple Master Nodes for details about how to configure TORQUE for high availability using multiple master nodes.

Scyld also redistributes the Scyld Maui jobscheduler, also derived from Adaptive Computing, that functions in conjunction with the TORQUE job manager. The alternative Moab job scheduler is also available from Adaptive Computing with a separate license, giving customers additional job scheduling, reporting, and monitoring capabilities.

In addition, Scyld provides support for most popular Open Source and commercial schedulers and resource managers, including SGE, LSF, and PBSPro. For the latest information, see the Penguin Computing Support Portal at https://www.penguincomputing.com/support.

4.7.1 Enabling TORQUE or Slurm

To enable TORQUE: after all compute nodes are up and running, you disable Slurm (if it is currently enabled), then enable and configure TORQUE, then reboot all the compute nodes:

```
slurm-scyld.setup cluster-stop
beochkconfig 98slurm off
slurm-scyld.setup disable
beochkconfig 98torque on
torque-scyld.setup reconfigure     # when needed
torque-scyld.setup enable
torque-scyld.setup cluster-start
torque-scyld.setup status
bpctl -S all -R
```

To enable Slurm: after all compute nodes are up and running, you disable TORQUE (if it is currently enabled), then enable and configure Slurm, then reboot all the compute nodes:

```
torque-scyld.setup cluster-stop
beochkconfig 98torque off
torque-scyld.setup disable
beochkconfig 98slurm on
slurm-scyld.setup reconfigure     # when needed
slurm-scyld.setup enable
slurm-scyld.setup cluster-start
slurm-scyld.setup status
bpctl -S all -R
```

Note: slurmdbd uses mysql to create a database defined by /etc/slurm/slurmdbd.conf, and expects mysql to be configured with no password.
Each Slurm user must setup the PATH and LD_LIBRARY_PATH environment variables to properly access the Slurm commands. This is done automatically for users who login when the slurm service is running and the pbs_server is not running, via the /etc/profile.d/scyld.slurm.sh script. Alternatively, each Slurm user can manually execute module load slurm or can add that command line to (for example) the user’s .bash_profile.

4.8 Managing Non-Scyld Nodes

A ClusterWare cluster typically consists of a Scyld master node and one or more Scyld compute nodes, integrated and communicating across the private cluster network interface. However, ClusterWare also supports additional devices and nodes that may reside on that private cluster network. This section describes how these Scyld and non-Scyld nodes are configured using entries in the /etc/beowulf/config file.

4.8.1 DHCP IP address assignment to devices

The private cluster network may have one or more devices attached to it that issue a DHCP request to obtain a dynamic IP address, vs. the device being configured with a static IP address. Typically, only the master node (or nodes - see Managing Multiple Master Nodes) owns a static IP address.

**Caution**

Care must be taken with static IP addresses to guarantee there are no address collisions.

Examples of such devices are managed switches and storage servers. The beoserv DHCP service for such devices is configured using the host directive, together with an associated hostname directive. For example,

```plaintext
nodes 32
iprange 10.20.30.100 10.20.30.131 # IPaddr range of compute nodes
... 
hostrange 10.20.30.4 10.20.30.9  # IPaddr range of devices for DHCP
hostrange 10.20.30.90 10.20.30.99  # IPaddr range of PDUs for DHCP
... 
host 00:A0:D1:E9:87:CA 10.20.30.5 smartswitch
host 00:A0:D1:E3:FC:E2 10.20.30.90 pdu1
host 00:A0:D1:E3:FD:4A 10.20.30.91 pdu2
```

The host keyword affects both the beoserv DHCP server and how the ClusterWare NSS responds to hostname lookups. The host keyword associates a non-cluster entity, identified by its MAC address, to an IP address that should be delivered to that client entity, if and when it makes a DHCP request to the master node, together with one or more optional hostnames to be associated with this IP address.

If the hostname is provided, then normal NSS functionality is available. Using the above example, then:

```bash
[user1@cluster ~] $ getent hosts smartswitch
```

returns:

```
10.20.30.5 smartswitch
```

and

```bash
[user1@cluster ~] $ getent ethers 00:A0:D1:E9:87:CA
```

returns:
Each host IP address must fall within a defined hostrange range of IP addresses. Moreover, each of the potentially multiple hostranges must not overlap any other range, must not overlap the cluster compute nodes range that is defined by the iprange directive, and must not collide with IP address(es) of master node(s) on the private network.

### 4.8.2 Simple provisioning using PXE

A default node entry, such as:

```
node 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F
```

or an explicitly numbered node entry, such as one for node15:

```
node 15 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F
```

is assumed to be a Scyld node, and a PXE request from one of these MAC addresses results in beoserv provisioning the node with the kernel image, initrd image, and kernel command-line arguments that are specified in /etc/beowulf/config file entries, e.g.:

```
kerneldir /boot/vmlinuz-2.6.18-164.2.1.el5.540g0000
initrdimage /var/beowulf/boot/computenode.initrd
kerneldirrw root=/dev/ram0 image=/var/beowulf/boot/computenode rootfs
```

ClusterWare automatically maintains the config file’s default kerneldir to specify the same kernel that currently executes on the master node. A Scyld node integrates into the BProc unified process space.

Enhanced syntax allows for custom booting of different kernel and initrd images. For example, specific nodes can boot a standalone RAM memory test in lieu of booting a full Linux kernel:

```
kerneldir 15 /var/beowulf/boot/memtest86+-4.00.bin
initrdimage 15 none
kerneldir 15 none
```

Thus when node15 makes a PXE request, it gets provisioned with the specified binary image that performs a memory test. In the above example, the initrdimage of none means that no initrd image is provisioned to the node because that particular memory test binary doesn't need an initrd. Moreover, the node number specifier of 15 can be a range of node numbers, each of which would be provisioned with the same memory test.

### 4.8.3 Simple provisioning using the class directive

An optional config file class directive assigns a name to a set of image and kernel command-line arguments. The previous example can be alternatively accomplished with:

```
class memtest kerneldir /var/beowulf/boot/memtest86+-4.00.bin
class memtest initrdimage none
class memtest kernelcommandline none
...
node 15 memtest 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F
```

which results in the same memory test provisioning of node15 as seen earlier.

Similarly, the default Scyld node provisioning can be expressed as:
The first `pxe` is termed the `boot-sequence`, and the second `pxe` is termed the `boot-stage`. The `boot-stage` describes how `beoserv` should respond to a node’s PXE request. In the example above, the `boot-stage` of `pxe` instructs `beoserv` to respond to the node’s first PXE request with the kernel image, initrd image, and kernel command-line specified in the class `scyld`.

### 4.8.4 Booting a node from the local harddrive

The `node` entry’s `boot-sequence` and `boot-stage` have more powerful capabilities. For example, suppose node 15 is installed with a full distribution of CentOS 4.8 on a local harddrive, and suppose the master node’s `config` file contains entries:

```plaintext
class genericboot kernelimage none
class genericboot initrdimage none
class genericboot kernelcommandline none
...
node 15 genericboot pxe+local local 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F
```

When node 15 boots, it first makes a DHCP request to join the private cluster network, then it attempts to boot, abiding by the specific sequence of boot devices named in its BIOS. ClusterWare expects that the first boot device is PXE over Ethernet, and the second boot device is a local harddrive. When node 15 initiates its PXE request to the master node, `beoserv` sees the `boot-stage` of `local` and thus directs node 15 to “boot next”, i.e., to boot from the local harddrive.

### 4.8.5 Provisioning a non-Scyld node

In the previous example, we assumed that node 15 already had a functioning, bootable operating system already installed on the node. Having a preexisting installation is not a requirement. Suppose the `config` file contains entries:

```plaintext
class centos5u4 kernelimage /var/beowulf/boot/vmlinuz-centos5u4_amd64
class centos5u4 initrdimage /var/beowulf/boot/initrd-centos5u4_amd64.img
class centos5u4 kernelcommandline initrd=initrd-centos5u4_amd64.img
    ks=nfs:10.1.1.1:/home/os/kickstarts/n5-ks.cfg ksdevice=eth0
...
node 15 centos5u4 pxe+local pxe 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F
```

(where the `kernelcommandline` has been broken into two lines for readability, although in reality it must be a single line in the `config` file). This time node 15’s PXE request arrives, and the `boot-stage` of `pxe` directs `beoserv` to respond with the `class centos5u4` kernel image, initrd image, and kernel command-line arguments. The latter’s `ks` arguments informs node 15’s kernel to initiate a kickstart operation, which is a Red Hat functionality that provisions the requester with rpms and other configuration settings as specified in the `/home/os/kickstarts/n5-ks.cfg` kickstart configuration file found on the master node. It is the responsibility of the cluster administrator to create this kickstart file. See Special Directories, Configuration Files, and Scripts for a sample configuration file.

After this initial PXE response (i.e., the `pxe` step of the `pxe+local` `boot-sequence`), `beoserv` rewrites the `node` entry to change the `boot-stage` to the `local` second step of the `pxe+local` `boot-sequence`. For example,
node 15 centos5u4 pxe+local pxe 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F

gets automatically changed to:

node 15 centos5u4 pxe+local local 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F

What this accomplishes is: the first PXE request is met with a directive to boot a kernel on node15 that initiates the kickstart provisioning, and then any subsequent PXE request from node15 (presumably from a now-fully provisioned node) results in a beoserv directive to node15 to “boot next”, i.e., to boot from the local harddrive.

If the cluster administrator wishes to reprovision the node and start fresh, then simply change the boot-stage from local back to pxe, and execute systemctl reload clusterware to instruct beoserv to re-read the config file to see your manual changes.

If you want the node to kickstart reprovision on every boot (albeit an unlikely scenario, but presented here for completeness), then you would configure this using:

node 15 centos5u4 pxe pxe 00:A0:D1:E5:C4:6E 00:A0:D1:E5:C4:6F

### 4.8.6 Integrating a non-Scyld node into the cluster

A non-Scyld node that locally boots a full distribution operating system environment may have an assigned IP address in the private cluster network iprange, but it is initially invisible to the master node’s monitoring tools and job manager. The bpstat tool only knows about Scyld nodes, and the more general beostatus is ignorant of the non-Scyld node’s presence in the cluster. The non-Scyld node is itself ignorant about the names and IP addresses of other nodes in the cluster, whether they be Scyld or non-Scyld nodes, until and unless the cluster administrator adds each and every node into the non-Scyld node’s local /etc/hosts file.

This shortcoming can be remedied by installing two special ClusterWare packages onto the non-Scyld node: beostat-sendstats and beonss-kickbackclient. These packages contain the client-side pieces of beostat and beonss. They are available in the standard ClusterWare yum repository and are compatible with non-Scyld RHEL and CentOS distributions - and perhaps with other distributions. One way to judge compatibility is to determine what libraries the ClusterWare daemons need to find on the non-Scyld compute node. (The daemons are known to execute in recent RHEL and CentOS environments.) Examine the daemons that were installed on the master node when ClusterWare was installed:

```
ldd /usr/sbin/sendstats
ldd /usr/sbin/kickbackproxy
```

and then determine if the libraries that these binaries employ are present on the target non-Scyld node. If the libraries do so exist, then the special ClusterWare packages can be downloaded and installed on a non-Scyld node.

First, you should download the packages from the ClusterWare yum repo. A useful downloader is the /usr/bin/yumdownloader utility, which can be installed from the CentOS extras yum repository if it is not already installed on your master node:

```
[root@cluster ~] # yum install yum-utils
```

Then use the utility to download the special Penguin ClusterWare rpms:

```
[root@cluster ~] # yumdownloader --destdir=<localdir> beostat-sendstats beonss-kickbackclient
```

retrieves the rpms and stores them into the directory <localdir>, e.g., /var/www/html or /etc/beowulf/nonsclyld.
These special packages can be installed manually on the non-Scyld node, or can be installed as part of the kickstart procedure (see Provisioning a non-Scyld node). Each package includes a /etc/init.d/ script that must be edited by the cluster administrator. Examine /etc/init.d/beostat-sendstats and /etc/init.d/beonss-kickbackclient, which contain comments that instruct the administrator about how to configure each script. Additionally, the non-Scyld node’s /etc/nsswitch.conf must be configured to invoke the kickback service for the databases that the administrator wishes to involve beonss and the master node. See the master node’s /etc/beowulf/nsswitch.conf for a guide to which databases are supported, e.g., hosts, passwd, shadow, and group. Finally, on the non-Scyld node, enable the scripts to start at node startup:

```
[root@cluster ~] # chkconfig beostat-sendstats on
[root@cluster ~] # chkconfig beonss-kickbackclient on
```

### 4.9 Managing Multiple Master Nodes

ClusterWare supports up to four master nodes on the same private cluster network. Every master node of a given cluster typically references a common /etc/beowulf/config file, which means all master nodes share a common understanding of all compute nodes that are attached to the private cluster network. That is, each master node knows a given physical node (denoted by its MAC addresses) by a common IP address and hostname. The config file’s masterorder directive configures which master node controls which compute nodes. Additionally, every master node should share a common understanding of userID (/etc/passwd) and groupID (/etc/group) values.

#### 4.9.1 Active-Passive Masters

In a simple active-passive configuration, all compute nodes are “owned” by one and only one master node at any one time, and the secondary master node (or nodes) comes into play only if and when the primary master fails. A compute node’s self-reassignment of ownership is called “cold re-parenting”, as it only occurs when a node reboots.

For example, for a cluster with two master nodes and 32 compute nodes, the /etc/beowulf/config file on each master node contains the entry:

```
masterorder 0-31 10.1.1.1 10.1.1.2
```

or alternatively, an entry that uses a hyphen to avoid using explicit node numbers:

```
masterorder - 10.1.1.1 10.1.1.2
```

where the IP addresses are the static addresses assigned to the two masters. When a compute node boots, each master node interprets the same masterorder directive and knows that master 10.1.1.1 is the primary master and nominally “owns” all the nodes, and 10.1.1.2 is the secondary master which only steps in if the primary master is unresponsive.

#### 4.9.2 Active-Active Masters

Many labs and workgroups today have several compute clusters, where each one is dedicated to a different research team or engineering group, or is used to run different applications. When an unusually large job needs to execute, it may be useful to combine most or all of the nodes into a single larger cluster, and then afterwards split up the cluster when the job is completed. Also, the overall demand for particular applications may change over time, requiring changes in the allocation of nodes to applications.

The downside to this approach of using multiple discrete clusters, each with their separate private cluster network, is that the compute node reconfiguration requires physically rewiring the network cabling, or requires reprogramming a smart switch to move nodes from one discrete network to another.
However, with an active-active configuration, the cluster’s master nodes and compute nodes reside on the same common private cluster network. The nodes are divided into subsets, and each subset is actively “owned” by a different master node and perhaps dedicated to separate users and applications. Additionally, each subset is passively associated with other master nodes.

For example, suppose each master node’s `/etc/beowulf/config` contains:

```
masterorder  0-15 10.1.1.1 10.1.1.2 10.1.1.3
masterorder  16-31 10.1.1.1 10.1.1.2 10.1.1.3
```

which divides the 32 compute nodes into two subsets of 16, with one subset owned by master 10.1.1.1 and the other subset owned by 10.1.1.2. To add complexity to this example, we introduce a passive third master node, 10.1.1.3, which becomes active only if both master nodes fail. This configuration provides for several advantages over two discrete 16-node clusters. One advantage is the same as provided by an active-passive configuration: in the event of a failure of one master node, that master’s compute nodes automatically reboot and “cold re-parent” to another master node, which now becomes the active “owner” of all 32 compute nodes.

Another advantage is that the cluster administrator can easily respond to changing demands for computing resources through a controlled and methodical migration of nodes between masters. For example, the administrator can shift eight nodes, n16 to n23, from one master to the other by changing the `masterorder` entries to be:

```
masterorder  0-23 10.1.1.1 10.1.1.2 10.1.1.3
masterorder  24-31 10.1.1.1 10.1.1.2 10.1.1.3
```

and replicating this same change to all other master nodes. Then the administrator executes on every master node the command `systemctl reload clusterware`, which instructs `beoserv` and `bpmaster` daemons to re-read the changed `/etc/beowulf/config`. Finally, on the currently “owning” master the administrator executes the command `bpctl -S 16-23 -R`, which reboots those shifted eight nodes and thereby causes them to cold re-parent to a different master node.

Reversing this reconfiguration, or performing any other reconfiguration, is equally simple:

1. Edit `/etc/beowulf/config` on one master to change the `masterorder` entries,
2. Replicate these same changes (or copy the same `config` file) to every affected master node,
3. Execute `systemctl reload clusterware` on each master node to re-read the `config` file, and
4. Execute `bpctl -S <noderange> -R` on the current “owning” master node, where `<noderange>` is the range of affected nodes, which tells the affected node(s) to reboot and re-parent to their new active master.

### 4.10 Managing Node Failures

Node failures are an unfortunate reality of any computer system, and failures in a Scyld ClusterWare cluster are inevitable and hopefully rare. Various strategies and techniques are available to lessen the impact of node failures.

#### 4.10.1 Protecting an Application from Node Failure

There is only one good solution for protecting your application from node failure, and that is checkpointing. Checkpointing is where at regular intervals your application writes to disk what it has done so far, and at startup checks the file on disk so that it can start off where it was when it last wrote the file.

The way to checkpoint that gives you the highest chance of recovering is to send the data back to the master node and have it checkpoint there, and also make regular backups of your files on the master node.

When setting up checkpointing, it is important to think carefully about how often you want to checkpoint. Some jobs that don’t have much data that needs to be saved can checkpoint as often as every 5 minutes, whereas if you have a large
data set, it might be smarter to checkpoint every hour, day, week, or longer. It depends a lot on your application. If you have a lot of data to checkpoint, you don’t want to do it often as that will drastically increase your run time. However, you also want to make sure that if you only checkpoint once every two days, that you can live with losing two days worth of work if there is ever a problem.

4.10.2 Compute Node Failure

A compute node can fail for any of a variety of reasons, e.g., broken node hardware, a broken network, software bugs, or inadequate hardware resources. A common example of the latter is a condition known as Out Of Memory, or OOM, which occurs when one or more applications on the node have consumed all available RAM memory and no swap space is available. The Linux kernel detects an OOM condition, attempts to report what is happening to the cluster’s syslog server, and begins to kill processes on the node in an attempt to eliminate the process that is triggering the problem. While this kernel response may occasionally be successful, more commonly it will kill one or more processes that are important for proper node behavior (e.g., a job manager daemon, the crucial Scyld bpslave daemon, or even a daemon that is required for the kernel’s syslog messages to get communicated to the cluster’s syslog server). When that happens, the node may still remain up in a technical sense, but the node is useless and must be rebooted.

4.10.2.1 When Compute Nodes Fail

When a compute node fails, all jobs running on that node will fail. If there was an MPI job running that was using that node, the entire job will fail on all the nodes on which the MPI program was running.

Even though the running jobs running on that node failed, jobs running on other nodes that weren’t communicating with jobs on the failed node will continue to run without a problem.

If the problem with the node is easily fixed and you want to bring the node back into the cluster, then you can try to reboot it using bpctl -S nodenumber -R. If the compute node has failed in a more catastrophic way, then such a graceful reboot will not work, and you will need to powercycle or manually reset the hardware. When the node returns to the up state, new jobs can be spawned that will use it.

If you wish to switch out the node for a new physical machine, then you must replace the broken node’s MAC addresses with the new machine’s MAC addresses. When you boot the new machine, it either appears as a new cluster node that is appended to the end of the list of nodes (if the config file says nodeassign append and there is room for new nodes), or else the node’s MAC addresses get written to the /var/beowulf/unknown_addresses file. Alternatively, manually edit the config to change the MAC addresses of the broken node to the MAC addresses of the new machine, followed by the command systemctl reload clusterware. Reboot this node, or use IPMI to powercycle it, and the new machine reboots in the correct node order.

4.10.2.2 Compute Node Data

What happens to data on a compute node after the node goes down depends on how you have set up the file system on the node. If you are only using a RAMdisk on your compute nodes, then all data stored on your compute node will be lost when it goes down.

If you are using the harddrive on your compute nodes, there are a few more variables to take into account. If you have your cluster configured to run mke2fs on every compute node boot, then all data that was stored on ext2 file systems on the compute nodes will be destroyed. If mke2fs does not execute, then fsck will try to recover the ext2 file systems; however, there are no guarantees that the file system will be recoverable.

Note that even if fsck is able to recover the file system, there is a possibility that files you were writing to at the moment of node failure may be in a corrupt or unstable state.
### 4.10.3 Master Node Failure

A master node can fail for the same reasons a compute node can fail, i.e., hardware faults or software faults. An Out-Of-Memory condition is more rare on a master node because the master node is typically configured with more physical RAM, more swap space, and is less commonly a participant in user application execution than is a compute node. However, in a Scyld ClusterWare cluster the master node plays an important role in the centralized management of the cluster, so the loss of a master node for any reason has more severe consequences than the loss of a single compute node. One common strategy for reducing the impact of a master node failure is to employ multiple master nodes in the cluster. See *Managing Multiple Master Nodes* for details.

Another moderating strategy is to enable *Run-to-Completion*. If the `bpslave` daemon that runs on each compute node detects that its master node has become unresponsive, then the compute node becomes an *orphan*. What happens next depends upon whether or not the compute nodes have been configured for *Run-to-Completion*.

#### 4.10.3.1 When Master Nodes Fail - Without Run-to-Completion

The default behavior of an orphaned `bpslave` is to initiate a reboot. All currently executing jobs on the compute node will therefore fail. The reboot generates a new DHCP request and a PXEboot. If multiple master nodes are available, then eventually one master node will respond. The compute node reconnects to this master - perhaps the same master that failed and has itself restarted, or perhaps a different master - and the compute node will be available to accept new jobs.

Currently, Scyld only offers *Cold Re-parenting* of a compute node, in which a compute node must perform a full reboot in order to “fail-over” and reconnect to a master. See *Managing Multiple Master Nodes* for details.

#### 4.10.3.2 When Master Nodes Fail - With Run-to-Completion

You can enable *Run-to-Completion* by enabling the ClusterWare script: `beochkconfig 85run2complete on`. When enabled, if the compute node becomes orphaned because its `bpslave` daemon has lost contact with its master node’s `bmpmaster` daemon, then the compute node does not immediately reboot. Instead, the `bpslave` daemon keeps the node up and running as best it can without the cooperation of an active master node. In an ideal world, most or all jobs running on that compute node will continue to execute until they complete or until they require some external resource that causes them to hang indefinitely.

Run-to-Completion enjoys greatest success when the private cluster network uses file server(s) that require no involvement of any compute node’s active master node. In particular, this means not using the master node as an NFS server, and not using a file server that is accessed using IP-forwarding through the master node. Otherwise, an unresponsive master also means an unresponsive file server, and that circumstance is often fatal to a job. Keep in mind that the default `/etc/beowulf/fstab` uses `$MASTER` as the NFS server. You should edit `/etc/beowulf/fstab` to change `$MASTER` to the IP address of the dedicated (and hopefully long-lived) non-master NFS server.

Stopping or restarting the clusterware service, or just rebooting the compute nodes doing `bpctl -S all -R`, will not put the compute nodes into an orphan state. These actions instruct each compute node to perform an immediate graceful shutdown and to restart with a PXEboot request to its active master node. Similarly, rebooting the master node will also stop the service with a `systemctl stop clusterware` as part of the master shutdown, and the compute nodes will immediately reboot and attempt to PXEboot before the master node has fully rebooted and thus ready to service the nodes. This will be a problem unless another master node is running on the private cluster network that will respond to the PXEboot request, or unless the nodes’ BIOS have been configured to perpetually retry the PXEboot, or unless you explicitly force all the compute nodes to immediately become orphans prior to rebooting the master with `bpctl -S all -O`, thereby delaying the nodes’ reboots until the master has time to reboot.

Once a compute node has become orphaned, it can only rejoin the cluster by rebooting, i.e., a so-called *Cold Re-parenting*. There are two modes that `bpslave` can employ:

- **1. No automatic reboot.** The cluster administrator must reboot each orphaned node using IPMI or by manually powering cycling the server(s).
2. Reboot the node after being “effectively idle” for a span of \( N \) seconds. This is the default mode. The default \( N \) is 300 seconds, and the default “effectively idle” is cpu usage below 1% of one cpu’s available cpu cycles.

Edit the `85run2complete` script to change the defaults. Alternatively, the `bpctl` can set (or reset) the run-to-completion modes and values. See `man bpctl`.

The term “effectively idle” means a condition wherein the cpu usage on the compute node is so small as to be interpreted as insignificant, e.g., attributed to various daemons such as `bpslave`, `sendstats`, and `pbs_mom`, which periodically awaken, check fruitlessly for pending work, and quickly go back to sleep. An orphaned node’s `bpslave` periodically computes cpu usage across short time intervals. If the cpu usage is below a threshold percentage \( P \) (default 1%) of one cpu’s total available cpu cycles, then the node is deemed “effectively idle” across that short time interval. If and when the “effectively idle” condition persists for the full \( N \) seconds time span (default 300 seconds), then the node reboots. If the cpu usage exceeds that threshold percentage during any one of those short time intervals, then the time-until-reboot is reset back to the full \( N \) seconds.

If the cluster uses TORQUE as a job manager, Run-to-Completion works best if TORQUE is configured for High Availability. Refer to PBS TORQUE documentation for details.

RHEL/CentOS 7 has deprecated the earlier `Heartbeat` software in preference to `Corosync`.

### 4.11 Compute Node Boot Options

One of the unique advantages of Scyld ClusterWare is the fast and flexible boot procedure for compute nodes. The Scyld `BeoBoot` system is a combination of unified booting and a carefully designed light-weight compute node environment. The `BeoBoot` system allows compute nodes to initialize with a very small boot image that may be stored on a wide range of boot media. This small boot image never has to change; however, Scyld ClusterWare’s boot setup allows you to change the kernel the compute nodes run, the modules that are loaded, and every aspect of the application environment by changing a few files on the master node.

This chapter gives instructions for setting up different types of boot media for the compute nodes, changing various settings that control the boot process, and checking for boot error messages. A detailed description of the boot process is included in the ClusterWare technical description in `Scyld ClusterWare Design Overview`.

#### 4.11.1 Compute Node Boot Media

There are several ways to boot a compute node with Scyld ClusterWare, as discussed in the following sections. The methods described are all interchangeable, and they work seamlessly with each other. Thus, you can have some of your compute nodes boot using one method and other nodes boot with a different method.

##### 4.11.1.1 PXE

PXE is a protocol that defines a standard way to netboot x86-based machines. In order for PXE to work, your compute nodes must have support for it in both the network adapters as well as the BIOS. The option to PXE boot must also be turned on in the BIOS. This is the preferred method of booting nodes in a Scyld cluster.
4.11.1.2 Local Disk

You can configure a node to boot from its local harddrive. See Managing Non-Scyld Nodes for details.

4.11.1.3 Linux BIOS

Linux BIOS is a project to replace the BIOS of a machine with Linux. This greatly speeds up the boot process as most of the actual work done by the BIOS is designed to make things like DOS work, but which aren't really needed by Linux.

There has been work done by third parties so that it is a Scyld ClusterWare initial image that replaces the BIOS. This has the advantage that all you need for a compute node is a motherboard with ram, processor, built-in network adapter, and a power supply.

Linux BIOS is not supported by Penguin Computing, Inc., however you can see http://www.linuxbios.org/ for more information if you are interested.

4.11.1.4 Flash Disk

Although not Scyld specific, using a flash disk is mentioned as it can increase cluster reliability. A flash disk is a solid state device using an Electrical Erasable PROM (EEPROM). The devices are seen by the BIOS as an IDE or SCSI harddrive, and support all normal drive operations, including running beofdisk and installing the initial boot image. This allows a node cluster configuration with no moving parts other than cooling fans, and is an alternative to using the Linux BIOS. These devices are faster and cheaper than harddrives, and are currently limited to 4 MB to 512 MB. But, for booting, less than 2 MB would be needed.

4.11.2 Changing Boot Settings

4.11.2.1 Adding Steps to the node_up Script

If you wish to add more steps to be executed during the node_up script, you can do it without actually editing the script. Instead, you create a script in the /etc/beowulf/init.d/ directory. All scripts in this directory will be executed for each node that boots up. This script will be sourced by the node_up script when the specified node boots, therefore it must be written in standard sh. When your script is sourced, the variable $NODE will be set to the node number that is booting. See Special Directories, Configuration Files, and Scripts for more details.

4.11.2.2 Per-Node Parameters

Starting with Scyld Series 30, support is provided for specifying kernel image and kernel command line parameters on a per-node basis in the cluster config file /etc/beowulf/config. This enables one set of nodes to boot with a particular initrd image, while another group boots with a different one.

The utility of this feature can be illustrated by the use of the memtest86 memory testing utility. For example, if you had just expanded your cluster with 5 new nodes (nodes 16 through 20), and you wanted to test their memory before putting them into production, you could have them all boot into memtest86 rather than the usual Scyld initrd with the following entry in /etc/beowulf/config:

```
kernalimage 16-20 /var/beowulf/boot/memtest86.bin
initrdimage 16-20 none
kernalcommandline 16-20 none
```
4.11.3 Error Logs

There are a number of ways to check for errors that occur during the compute node boot process, as follows:

- During the compute node boot process, any error messages are sent to the console of the compute node and forwarded to the cluster’s syslog server’s /var/log/messages file by the node’s beoklogd daemon. By default, the syslog server is the master node. See the syslog_server= option in Compute node command-line options for details about how to direct these compute node logging messages to an alternate server. Messages can be viewed by manually editing this file read-only or by running the standard Linux System Logs tool: Select System Tools -> System Logs from the desktop menu to open the System Logs window, then select the System Log from the list of logs in the left panel, then scroll near the end to see errors.

- During each node’s boot, the node_up script writes node-specific output to a log file /var/log/beowulf/node.<nodenumber>, where <nodenumber> is the node number. If the compute node ends up in the error state, or if it remains in the boot state for an extended length of time, then you should examine this node log.

4.12 Disk Partitioning

Partitioning allows disk storage space to be broken up into segments that are then accessible by the operating system. This chapter discusses disk partitioning concepts, the default partitioning used by Scyld ClusterWare, and some useful partitioning scenarios.

Scyld ClusterWare creates a RAM disk on the compute node by default during the initial boot process. This RAM disk is used to hold the final boot image downloaded from the master node. If you have diskless nodes, then this chapter does not pertain to you.

4.12.1 Disk Partitioning Concepts

Disk partitioning on a cluster is essentially no different than partitioning on any stand-alone computer, with a few exceptions.

On a stand-alone computer or server, the disk drive’s file system(s) divide the storage available on the disk into different sections that are configured in ways and sizes to meet your particular needs. Each partition is a segment that can be accessed independently, like a separate disk drive. The partitions are configured and determined by the partition table contained on each disk.

Each partition table entry contains information about the locations on the disk where the partition starts and ends, the state of the partition (active or not), and the partition’s type. Many partition types exist, such as Linux native, AIX, DOS, etc. The cluster administrator can determine the appropriate partition types for his/her own system.

Disk partitioning on a cluster is very much determined by the cluster system hardware and the requirements of the application(s) that will be running on the cluster, for instance:

- Some applications are very process intensive but not very data intensive. In such instances, the cluster may best utilize a RAM disk in the default partitioning scheme. The speed of the RAM will provide better performance, and not having a harddrive will provide some cost savings.

- Some applications are very data intensive but not very process intensive. In these cases, a hard disk is either required (given the size of the data set the application is working with) and/or is a very inexpensive solution over purchasing an equivalent amount of memory.
The harddrive partitioning scheme is very dependent on the application needs, the other tools that will interface with the data, and the preferences of the end-user.

4.12.2 Disk Partitioning with ClusterWare

This section briefly describes the disk partitioning process for the master node and compute nodes in a Scyld cluster.

4.12.2.1 Master Node

On the master node of a Scyld cluster, the disk partitioning administration is identical to that on any stand-alone Linux server. As part of installing Red Hat Linux, you are requested to select how you would like to partition the master node’s hard disk. After installation, the disk partitioning can be modified, checked, and utilized via traditional Linux tools such as `fdisk`, `sfdisk`, `cfdisk`, `mount`, etc.

4.12.2.2 Compute Nodes

The compute nodes of a Scyld cluster are slightly different from a traditional, stand-alone Linux server. Each compute node hard disk needs to be formatted and partitioned to be useful to the applications running on the cluster. However, not too many people would enjoy partitioning 64 or more nodes manually.

To simplify this task, Scyld ClusterWare provides the `beofdisk` tool, which allows remote partitioning of the compute node hard disks. It is very similar in operation to `fdisk`, but allows many nodes to be partitioned at once. The use of `beofdisk` for compute node partitioning is covered in more detail in `Partitioning Scenarios`.

4.12.3 Default Partitioning

This section addresses the default partitioning schemes used by Scyld ClusterWare.

4.12.3.1 Master Node

The default Scyld partition table allocates 4 partitions:
  - `/boot` partition
  - `/home` partition
  - `/` partition
  - Swap partition = 2 times physical memory

Most administrators will want to change this to meet the requirements of their particular cluster.

4.12.3.2 Compute Nodes

The default partition table allocates three partitions for each compute node:
  - BeoBoot partition = 2 MB
  - Swap partition = half the compute node’s physical memory or half the disk, whichever is smaller
  - Single root partition = remainder of disk
For diskless operation, the default method of configuring the compute nodes at boot time is to run off a RAM disk. This “diskless” configuration is appropriate for many applications, but not all. Typical usage requires configuration and partitioning of the compute node hard disks, which is covered in the partitioning scenarios discussed in the following section.

4.12.4 Partitioning Scenarios

This section discusses how to implement two of the most common partitioning scenarios in Scyld ClusterWare:

- Apply the default partitioning to all disks in the cluster
- Specify your own manual but homogeneous partitioning to all disks in the cluster

The Scyld beofdisk tool can read an existing partition table on a compute node. It sequentially queries compute nodes beginning with node 0. For each new type/position/geometry it finds, it looks for an existing partition table file in /etc/beowulf/fdisk. If no partition table is present, a new one is generated that uses the default scheme. For each device/drive geometry it finds, beofdisk creates a file in /etc/beowulf/fdisk/. These files can then be modified by hand. Whether modified or using the default options, the files can be written back to the harddrives.

**Caution**

If you attempt to boot a node with an unpartitioned harddrive that is specified in /etc/beowulf/fstab (or a node-specific fstab.N for node N), then that node boots to an error state unless the fstab entry includes the “nonfatal” option. See the Reference Guide or man beowulf-fstab for details.

4.12.4.1 Applying the Default Partitioning

To apply the default disk partitioning scheme (as recommended by the Scyld beofdisk tool) to the compute nodes, following these steps:

Query all the harddrives on the compute nodes and write out partition table files for them that contain the suggested partitioning:

```
[root@cluster ~] # beofdisk -d
Creating a default partition table for hda:2495:255:63
Creating a default partition table for hda:1222:255:63
```

Read the partition table files, and partition the harddrives on the compute nodes so that they match:

```
[root@cluster ~] # beofdisk -w
```

To use the new partitions you created, modify the /etc/beowulf/fstab file to specify how the partitions on the compute node should be mounted. The contents of /etc/beowulf/fstab should be in the standard fstab format.

To format the disk(s) on reboot, change “mkfs never” to “mkfs always” in the cluster config file /etc/beowulf/config.

To try out the new partitioning, reboot the compute nodes with the following:

```
[root@cluster ~] # bpctl -S all -R
```

**Caution**

To prevent disks from being reformatted on subsequent reboots, change "mkfs always" back to "mkfs never" in `~/etc/beowulf/config` after the nodes have booted.
4.12.4.2 Specifying Manual Partitioning

You can manually apply your own homogeneous partitioning scheme to the partition tables, instead of taking the suggested defaults. There are two methods for doing this:

- The recommended method involves running `fdisk` on the first node (node 0) of the cluster, and then on every first node that has a unique type of hard disk.
- The other method is to manually edit the partition table text file retrieved by the `beofdisk` query.

For example, assume that your cluster has 6 compute nodes, and that all disks have 255 heads and 63 sectors (this is the most common). Nodes 0, 1, and 5 have a single IDE hard disk with 2500 cylinders. Nodes 2, 3, and 4 have a first IDE disk with 2000 cylinders, and node 4 has a SCSI disk with 5000 cylinders. This cluster could be partitioned as follows:

1. Partition the disk on node 0:
   ```
   [root@cluster ~] # bpsh 0 fdisk /dev/hda
   ```
   Follow the steps through the standard `fdisk` method of partitioning the disk.

2. Manually partition the disk on node 2 with `fdisk`:
   ```
   [root@cluster ~] # bpsh 2 fdisk /dev/hda
   ```
   Again, follow the steps through the standard `fdisk` method of partitioning the disk.

3. Manually partition the SCSI disk on node 4 with `fdisk`:
   ```
   [root@cluster ~] # bpsh 4 fdisk /dev/sda
   ```
   Again, follow the steps through the standard `fdisk` method of partitioning the disk.

4. Next, query the compute nodes to get all the partition table files written for their harddrives by using the command `"beofdisk -q"`.

   At this point, the 3 partition tables will be translated into text descriptions, and 3 files will be put in the directory `/etc/beowulf/fdisk`. The file names will be `hda:2500:255:63`, `hda:2000:255:63`, and `sda:5000:255:63`. These file names represent the way the compute node harddrives are currently partitioned.

   You have the option to skip the `fdisk` command and just edit these files manually. The danger is that there are lots of rules about what combinations of values are allowed, so it is easy to make an invalid partition table. Most of these rules are explained as comments at the top of the file.

5. Now write out the partitioning scheme using the command `beofdisk -w`.

   When specifying unique partitioning for certain nodes, you must also specify a unique `fstab` for each node that has a unique partition table. To do this, create the file `/etc/beowulf/fstab.<nodenumber>`. If this file exists, the `node_up` script will use that as the `fstab` for the compute node; otherwise, it will default to `/etc/beowulf/fstab`. Each instance of `/etc/beowulf/fstab.<nodenumber>` should be in the same format as `/etc/beowulf/fstab`.

6. To format the disk(s) on reboot, change “mkfs never” to “mkfs always” in the cluster config file `/etc/beowulf/config`.

7. To try out the new partitioning, reboot the compute nodes with the following:

   ```
   [root@cluster ~] # bpctl -S all -R
   ```

Caution

To prevent disks from being reformatted on subsequent reboots, change the “mkfs always” back to “mkfs never” in `/etc/beowulf/config` after the nodes have booted.
4.13 File Systems

4.13.1 File Systems on a Cluster

File systems on a cluster consist of two types of file systems, local file systems and network file systems. The file /etc/fstab describes the filesystems mounted on the master node, and the file /etc/beowulf/fstab describes the filesystems mounted on each compute node. You may also create node-specific /etc/beowulf/fstab.N files, where N is a node number.

4.13.1.1 Local File Systems

Local file systems are the file systems that exist locally on each machine. In the Scyld ClusterWare setup, the master node has a local file system, typically ext3, and each node also has a local file system. The local file systems are used for storing data that is local to the machines.

4.13.1.2 Network/Cluster File Systems

Network file systems are used so that files can be shared across the cluster and every node in the cluster can see the exact same set of files. The default network file system for Scyld ClusterWare is NFS. NFS allows the contents of a directory on the server (by default the master node) to be accessed by the clients (the compute nodes). The default Scyld ClusterWare setup has the /home directory exported through NFS so that all the user home directories can be accessed on the compute nodes. Additionally, various other directories are mounted by default, as specified by /etc/beowulf/fstab or by a node-specific fstab.N.

Note that root’s home directory is not in /home, and thus cannot access its home directory on the compute nodes. This should not be a problem, as normal compute jobs should not be run as “root”.

4.13.2 NFS

NFS is the standard way to have files stored on one machine, yet be able to access them from other machines on the network as if they were stored locally.

4.13.2.1 NFS on Clusters

NFS in clusters is typically used so that if all the nodes need the same file, or set of files, they can access the file(s) through NFS. This way, if one changes the file, every node sees the change, and there is only one copy of the file that needs to be backed up.

4.13.2.2 Configuration of NFS

The Network File System (NFS) is what Scyld ClusterWare uses to allow users to access their home directories and other remote directories from compute nodes. (The User’s Guide has a small discussion on good and bad ways to use NFS.) Two files control what directories are NFS mounted on the compute nodes. The first is /etc/exports. This tells the nfs daemon on the master node what directories it should allow to be mounted and who can access them. Scyld ClusterWare adds various commonly useful entries to /etc/exports. For example:

| /home   @cluster(rw) |
The /home says that /home can be nfs mounted, and @cluster(rw) says who can mount it and what forms of access are allowed. @cluster is a netgroup. It uses one word to represent several machines. In this case, it represents all your compute nodes. cluster is a special netgroup that is setup by beonss that automatically maps to all of your compute nodes. This makes it easy to specify something can be mounted by your compute nodes. The (rw) part specifies what permissions the compute node has when it mounts /home. In this case, all user processes on the compute nodes have read-write access to /home. There are more options that can go here, and you can find them detailed in man exports.

The second file is /etc/beowulf/fstab. (Note that it is possible to set up an individual fstab.N for a node. For this discussion, we will assume that you are using a global fstab for all nodes.) For example, one line in the default /etc/beowulf/fstab is the following:

```
$MASTER:/home  /home  nfs  nolock,nonfatal  0  0
```

This is the line that tells the compute nodes to try to mount /home when they boot:

- The $MASTER is a variable that will automatically be expanded to the IP of the master node.
- The first /home is the directory location on the master node.
- The second /home is where it should be mounted on the compute node.
- The nfs specifies that this is an nfs file system.
- The nolock specifies that locking should be turned off with this nfs mount. We turn off locking so that we don’t have to run daemons on the compute nodes. (If you need locking, see File Locking Over NFS for details.)
- The nonfatal tells ClusterWare’s /usr/lib/beoboot/bin/setup_fs script to treat a mount failure as a nonfatal problem. Without this nonfatal option, any mount failure leaves the compute node in an error state, thus making it unavailable to users.
- The two 0’s on the end are there to make the fstab like the standard fstab in /etc.

To add an nfs mount of /foo to all your compute nodes, first add the following line to the end of the /etc/exports file:

```
/foo  @cluster(rw)
```

Then execute exportfs -a as root. For the mount to take place the next time your compute nodes reboot, you must add the following line to the end of /etc/beowulf/fstab:

```
$MASTER:/foo  /foo  nfs  nolock  0  0
```

You can then reboot all your nodes to make the nfs mount happen. If you wish to mount the new exported filesystem without rebooting the compute nodes, you can issue the following two commands:

```
[root @cluster ~] # bpsh -a mkdir -p /foo
[root @cluster ~] # bpsh -a mount -t nfs -o nolock master:/foo /foo
```

Note that /foo will need to be adjusted for the directory you actually want.

If you wish to stop mounting a certain directory on the compute nodes, you can either remove the line from /etc/beowulf/fstab or just comment it out by inserting a ‘#’ at the beginning of the line. You can leave untouched the entry referring to the filesystem in /etc/exports, or you can delete the reference, whichever you feel more comfortable with.

If you wish to umount that directory on all the compute nodes without rebooting them, you can then run the following:

```
[root @cluster ~] # bpsh -a umount /foo
```

where /foo is the directory you no longer wish to have NFS mounted.
Caution

On compute nodes, NFS directories must be mounted using either a specific IP address or the $MASTER keyword; the hostname cannot be used. This is because \texttt{fstab} is evaluated before node name resolution is available.

File Locking Over NFS

By default, the compute nodes mount NFSv3 filesystems with locking turned off. If you have a program that requires locking, first ensure that the \texttt{nfslock} service is enabled on the master node and is executing:

\begin{verbatim}
[root @cluster ~] # systemctl enable nfslock
[root @cluster ~] # systemctl start nfslock
\end{verbatim}

Next, edit /etc/beowulf/fstab to remove the \texttt{nolock} keyword from the NFS mount entries.

Finally, reboot the cluster nodes to effect the NFS remounting with locking enabled.

NFSD Configuration

By default, when the master node reboots, the /etc/init.d/nfs script launches 8 NFS daemon threads to service client NFS requests. For large clusters this count may be insufficient. One symptom of an insufficiency is a syslog message, most commonly seen when you boot all the cluster nodes:

\begin{verbatim}
nfsd: too many open TCP sockets, consider increasing the number of nfsd threads
\end{verbatim}

To increase the thread count (e.g., to 16):

\begin{verbatim}
[root @cluster ~] # echo 16 > /proc/fs/nfsd/threads
\end{verbatim}

Ideally, the chosen thread count should be sufficient to eliminate the syslog complaints, but not significantly higher, as that would unnecessarily consume system resources. To make the new value persistent across master node reboots, create the file /etc/sysconfig/nfs, if it does not already exist, and add to it an entry of the form:

\begin{verbatim}
RPCNFSDCOUNT=16
\end{verbatim}

A value of 1.5x to 2x the number of nodes is probably adequate, although perhaps excessive.

A more refined analysis starts with examining NFSD statistics:

\begin{verbatim}
[root @cluster ~] # grep th /proc/net/rpc/nfsd
\end{verbatim}

which outputs thread statistics of the form:

\begin{verbatim}
th 16 10 26.774 5.801 0.035 0.000 0.019 0.008 0.003 0.011 0.000 0.040
\end{verbatim}

From left to right, the \texttt{16} is the current number of NFSD threads, and the \texttt{10} is the number of times that all threads have been simultaneously busy. (Not all circumstances of all threads being busy results in that syslog message, but a high all-busy count does suggest that adding more threads may be beneficial.)

The remaining 10 numbers are histogram buckets that show how many accumulated seconds a percentage of the total number of threads have been simultaneously busy. In this example, 0-10\% of the threads were busy 26.744 seconds, 10-20\% of the threads were busy 5.801 seconds, and 90-100\% of the threads were busy 0.040 seconds. High numbers at the end indicate that most or all of the threads are simultaneously busy for significant periods of time, which suggests that adding more threads may be beneficial.
4.13.3 ROMIO

ROMIO is a high-performance, portable implementation of MPI-IO, the I/O chapter in MPI-2: Extensions to the Message Passing Interface, and is included in the Scyld ClusterWare distribution. ROMIO is optimized for noncontiguous access patterns, which are common in parallel applications. It has an optimized implementation of collective I/O, an important optimization in parallel I/O.

4.13.3.1 Reasons to Use ROMIO

ROMIO gives you an abstraction layer on top of high performance input/output. The details for the file system may be implemented in various ways, but ROMIO prevents you from caring. Your binary code will run on an NFS file system here and a different file system there, without changing a line or recompiling. Although POSIX open(), read(), ... calls already do this, the virtual file system code to handle this abstraction is deep in the kernel.

You may need to use ROMIO to take advantage of new special and experimental file systems. It is easier and more portable to implement a ROMIO module for a new file system than a Linux-specific VFS kernel layer.

Since ROMIO is an abstraction layer, it has the freedom to be implemented arbitrarily. For example, it could be implemented on top of the POSIX Asynchronous and List I/O calls for real-time performance reasons. The end-user application is shielded from caring, and benefits from careful optimization of the I/O details by experts.

4.13.3.2 Installation and Configuration of ROMIO

ROMIO Over NFS

To use ROMIO on NFS, file locking with fcntl must work correctly on the NFS installation. First, since file locking is turned off by default, you need to turn on NFSv3 locking. See File Locking Over NFS. Now, to get the fcntl locks to work, you must mount the NFS file system with the noac option (no attribute caching). This is done by modifying the line for mounting /home in /etc/beowulf/fstab to look like the following:

```
$MASTER:/home /home nfs noac,nonfatal 0 0
```

Turning off attribute caching may reduce performance, but it is necessary for correct behavior.

4.13.4 Other Cluster File Systems

There are variety of network file systems that can be used on a cluster. If you have questions regarding the use of any particular cluster file system with Scyld ClusterWare, contact Scyld Customer Support for assistance.

4.14 Load Balancing

You have made some rather significant investment in your cluster. It is also evident that it depreciates at a rather frightening rate. Given these two facts it should be obvious you want your cluster busy 100% of the time if possible.

However, timely results of output are also important. If the memory requirements of programs running on the cluster exceed the available physical memory, swap memory (hard disk) will be used severely reducing performance. Even if the memory requirements of many processes still fit within the physical memory, results of any one of the programs may take significantly longer to achieve if many jobs are running on the same nodes simultaneously.

Thus we come to concept of the “load balancing”, which maintain a delicate balance between overburdened and idle. Load balancing is when multiple servers can perform the same task, and which server performs the task is based on which server is currently doing the least amount of work. This helps to spread a heavy work load across several
machines, and does it intelligently: if one machine is more heavily loaded than the others, new requests will not be sent to it. By doing this, a job is always run on a machine that has the most resources to devote to it, and therefore gets finished sooner.

Generally, it is believed that a constant load of one 100% CPU bound process per CPU is ideal. However, not all processes are CPU bound; many are I/O bound on either the harddrive or the network. The act of load balancing is often described as “scheduling”.

Optimal load balancing is almost never achieved; hence, it is a subject of study for many researchers. The optimal algorithm for scheduling the programs running on your cluster is probably not the same as it might be for others, so you may want to spend time on your own load balancing scheme.

4.14.1 Load Balancing in a Scyld Cluster

Scyld ClusterWare supplies a general load balancing and job scheduling scheme via the beomap subsystem in conjunction with job queuing utilities. Mapping is the assignment of processes to nodes based on current CPU load. Queuing is the holding of jobs until the cluster is idle enough to let the jobs run. Both of these are covered in detail in other sections of this guide and in the User's Guide. In this section, we’ll just discuss the scheduling policy that is used.

4.14.1.1 Mapping Policy

The current default mapping policy consists of the following steps:

- Run on nodes that are idle
- Run on CPUs that are idle
- Minimize the load per CPU

Each proceeding step is only performed if the number of desired processes (NP) is not yet satisfied. The information required to perform these steps comes from the BeoStat sub-system of daemons and libbeostat library.

4.14.1.2 Queuing Policy

The current default queuing policy is to attempt to determine the desired number of processes (NP) and other mapping parameters from the job script. Next, the beomap command is run to determine which nodes would be used if it ran immediately. If every node in the returned map is below 0.8 CPU usage the job is released for execution.

4.14.2 Implementing a Scheduling Policy

The queuing portion of the schedule policy depends on which scheduling and resource management tool you are using. The mapping portions, however, are already modularized. There are a number of ways to override the default, including

- Substitute a different program for the beomap command and use mpirun to start jobs (which uses beomap).
- Create a shared library that defines the function get_beowulf_job_map() and use the environment variable LD_PRELOAD to force the pre-loading of this shared library.
- Create the shared library and replace the default /usr/lib/libbeomap.so file.

These methods are in order of complexity. We can’t actually highly recommend the first method as your mileage may vary. The second method is the most recommended followed by the third method of replacing the Scyld source code when you’re happy that your scheduler is better.

It is highly recommended that you get the source code for the beomap package. It will give you a head start on writing your own mappers. For more information on developing your own mapper, see the Programmer's Guide.
4.15 IPMI

Included in the RHEL/CentOS base distribution are tools that may be of interest to users, including the ipmitool command for monitoring and managing compute node hardware.

4.15.1 IPMI Tool

ipmitool is a hardware management utility that supports the Intelligent Platform Management Interface (IPMI) specification v1.5 and v2.0.

IPMI is an open standard that defines the structures and interfaces used for remote monitoring and management of a computer motherboard (baseboard). IPMI defines a micro-controller, called the “baseboard management controller” (BMC), which is accessed locally through the managed computer’s bus or through an out-of-band network interface connection (NIC).

Scyld ClusterWare supports ipmitool as the primary way to monitor and manage compute node hardware. The Scyld ClusterWare distribution includes /etc/beowulf/init.d/20ipmi, a script that executes at compute node boot time that enables IPMI on a compute node.

The root can use ipmitool for a variety of tasks, such as:

- Inventory a node’s baseboards to determine what sensors are present
- Monitor sensors (fan status, temperature, power supply voltages, etc.)
- Read and display values from the Sensor Data Repository (SDR)
- Read and set the BMC’s LAN configuration
- Remotely control chassis power
- Display the contents of the System Event Log (SEL), which records events detected by the BMC as well as events explicitly logged by the operating system
- Print Field Replaceable Unit (FRU) information, such as vendor ID, manufacturer, etc.
- Configure and emulate a serial port to the baseboard using the out-of-band network connection known as serial over LAN (SOL)

Several dozen companies support IPMI, including many leading manufacturers of computer hardware. You can learn more about OpenIPMI from the OpenIPMI project page at http://openipmi.sourceforge.net, which includes links to documentation and downloads.

4.16 Updating Software On Your Cluster

From time to time, Scyld may release updates and add-ons to Scyld ClusterWare. Customers on active support plans for Scyld software products can access these updates on the Penguin Computing website. Visit https://www.penguincomputing.com/support for details. This site offers answers to common technical questions and provides access to application notes, software updates, product documentation, and Release Notes.

The Release Notes for each software update will include instructions for installation, along with information on why the update was released and what bug(s) it fixes. Be sure to thoroughly read the Release Notes, as they may discuss specific requirements and potential conflicts with other software.
4.16.1 What Can’t Be Updated

Some packages provided with Scyld ClusterWare, such as Ganglia, are specifically optimized to take advantage of the BProc unified process space, which is added to the standard Linux distributions that Scyld supports. Other packages, such as MPICH2, MVAPICH2, MPICH3, and OpenMPI, take advantage of features of the Scyld ClusterWare TORQUE distribution. Although there are generally available versions of these packages that you can download from other sources, you should use the versions provided by Scyld for best performance with BProc and ClusterWare. Contact Scyld Customer Support if you have questions about specific packages that you would like to use with Scyld ClusterWare.

Users may also choose to use commercially available MPIS, such as Intel, HP, Scali, or Verari. These require specific configuration on Scyld ClusterWare. See the Penguin Computing Support Portal at https://www.penguincomputing.com/support, or contact Scyld Customer Support.

4.17 Special Directories, Configuration Files, and Scripts

Scyld ClusterWare adds some special files and directories on top of the standard Linux install that help control the behavior of the cluster. This appendix contains a summary of those files and directories, and what is in them.

4.17.1 What Resides on the Master Node

4.17.1.1 /etc/beowulf/ directory

All the config files for controlling how BProc and Beoboot behave are stored here.

/etc/beowulf/config

This file contains the settings that control the bpmaster daemon for BProc, and the beoserv daemon that is part of beoboot. It also contains part of the configuration for how to make beoboot boot images.

/etc/beowulf/fdisk/

This directory is used by beofdisk to store files detailing the partitioning of the compute nodes’ harddrives, and is also read from when it rewrites the partition tables on the compute nodes. See ?

/etc/beowulf/fstab

Refer to Disk Partitioning for details on using node-specific fstab.N files.
/etc/beowulf/backups/ directory

Contains time-stamped backups of older versions of various configuration files, e.g., /etc/beowulf/config and /etc/beowulf/fstab, to assist in the recovery of a working configuration after an invalid edit.

/etc/beowulf/init.d/ directory

Contains various scripts that are executed on the master node by the node_up script when booting a compute node.

/etc/beowulf/conf.d/ directory

Contains various configuration files that are needed when booting a compute node.

4.17.1.2 /usr/lib/beoboot directory

This directory contains files that are used by beoboot for booting compute nodes.

/usr/lib/beoboot/bin

This directory contains the node_up script and several smaller scripts that it calls.

4.17.1.3 /var/beowulf directory

This directory contains compute node boot files and static information, as well as the list of unknown MAC addresses. It includes three subdirectories.

/var/beowulf/boot

This is the default location for files essential to booting compute nodes. Once a system is up and running, you will typically find three files in this directory:

- computenode — the boot sector used for bootstrapping the kernel on the compute node.
- computenode.initrd — the kernel image and initial ramdisk used to boot the compute node.
- computenode.rootfs — the root file system for the compute node.

/var/beowulf/statistics

This directory contains a cached copy of static information from the compute nodes. At a minimum, it includes a copy of /proc/cpuinfo.
This file contains a list of Ethernet hardware (MAC) addresses for nodes considered unknown by the cluster. See Compute Node Categories for more information.

4.17.1.4 /var/log/beowulf directory

This directory contains the boot logs from compute nodes. These logs are the output of what happens when the node_up script runs. The files are named node.<number>, where <number> is the actual node number.

4.17.2 What Gets Put on the Compute Nodes at Boot Time

- Generally speaking, the /dev directory contains a subset of devices present in the /dev directory on the master node. The /usr/lib/beoboot/bin/mknoderootfs script creates most of the /dev/ entries (e.g., zero, null, and random). /etc/beowulf/init.d/20ipmi creates ipmi0. /usr/lib/beoboot/bin/setup_fs creates shm and pts (as directed by /etc/beowulf/fstab). The harddrive devices (e.g., sda) are created at compute node bootup time, if local drives are discovered. If Infiniband hardware is present on the compute node, /etc/beowulf/init.d/15openib creates various device entries in /dev/infiniband/.
- The /etc directory contains the ld.so.cache, localtime, mtab, and nsswitch.conf files. The node_up script creates a simple hosts file.
- The /home directory exists as a read-write NFS mount of the /home directory from the master node. Thus, all the home directories can be accessed by jobs running on the compute nodes.
- Additionally, other read-only NFS mounts exist by default, to better assist out-of-the-box application and script execution: /bin, /usr/bin, /opt, /usr/lib64/python2.3, /usr/lib/perl5, and /usr/lib64/perl5.
- The node_up script mounts pseudo-filesystems as directed by /etc/beowulf/fstab: /proc, /sys, and /btrfs.
- mknoderootfs creates /var and several of its subdirectories.
- The /tmp directory is world-writeable and can be used as temporary space for compute jobs.
- /etc/beowulf/config names various libraries directories that are managed by the compute node’s library cache. Run beoconfig libraries to see the current list of library directories. Caching shared libraries, done automatically as needed on a compute node, speeds up the transfer process when you are trying to run jobs, eliminates the need to NFS-mount the various common directories that contain libraries, and minimizes the space consumed by libraries in the compute node’s RAM filesystem.
- Typically, when the loader starts up an application, it opens the needed shared libraries. Each open() causes the compute node to pull the shared library from the master node and save it in the library cache, which typically resides in the node’s RAM filesystem. However, some applications and scripts reference a shared library or other file that, although it resides in one of those libraries directories, the reference does not use open() to access the file, and so the file does not get automatically pulled into the library cache. For example, an application or script might first use stat() to determine if a specific file exists, and then use open() if the stat() is successful, otherwise continue on to stat() an alternative file. The stat() on the compute node will fail until an open() pulls the file from the master. The application or script thus fails to execute, and the missing library or file name is typically displayed as an error.

To remedy this type of failure, you should use a prestage directive in /etc/beowulf/config to explicitly name files that should be pulled to each compute node at node startup time. Run beoconfig prestage for the current list of prestaged files.
4.17.3 /usr/lib/locale/locale-archive Internationalization

Glibc applications silently open the file /usr/lib/locale/locale-archive, which means it gets downloaded by each compute node early in a node’s startup sequence via the BProc filecache functionality. The default locale-archive is 95 MBytes in RHEL6 and over 100 MBytes in RHEL7. This download consumes significant network bandwidth and thus causes serialization delays if numerous compute nodes attempt to concurrently boot, and thereafter this large file consumes significant RAM filesystem space on each node. It is likely that a cluster’s users and applications do not require all the international locale data that is present in the default file. With care, the cluster administrator may choose to rebuild locale-archive with a greatly reduced set of locales and thus create a significantly smaller file that is less impactful on cluster performance.

Rebuilding and replacing locale-archive should be done on a quiescent master node, as the file typically is mmapped by a process (e.g., crond, bash), and the appearance of a replacement version may perturb shells and other programs, such as aborting the shell that executes the rebuild or having that shell issue an immediate warning message about an undefined environment variable. In the event that a problem does appear, you should reboot the master node. Otherwise, newly executing programs on the master node will use the updated locale-archive, and compute nodes will employ the new file only after the node reboots.

In a RHEL5 environment, the glibc-common RPM installs the /usr/lib/locale/ directory containing the full set of locale definition files and a full locale-archive binary file. The build-locale-archive command rebuilds the locale-archive with every individual locale data file that is found in that directory. Thus, to reduce the size of locale-archive, you must first reduce the number of locale data files in that directory - but only after saving the default locale data files in a safe place, so you can later rebuild the locale-archive with a different set of locale data files as the cluster’s needs change. Beginning with the default /usr/lib/locale/ directory with its full set of locale data files:

```
[root@cluster ~] # cd /usr/lib
[root@cluster ~] # cp -a locale locale.default
[root@cluster ~] # (cd locale ; rm -fr _*)
```
saves all the locale data files in a new directory and produces a stripped-down /usr/lib/locale/, leaving only the locale-archive file. Now reintroduce a smaller set of locale data files. For example, to include the U.S.-English and U.S.-Great Britain locale files:

```
[root@cluster ~] # cp -a locale.default/en_US* locale
[root@cluster ~] # cp -a locale.default/en_GB* locale
```

When /usr/lib/locale/ contains the desired locale data files, perform the rebuild:

```
[root@cluster ~] # build-locale-archive
```
and reboot the master node and/or the compute nodes as needed.

In a RHEL6 environment, the glibc-common RPM installs just the default locale-archive binary file. The default /usr/lib/locale/ directory contains no locale data files. Scyld ClusterWare has saved the default locale-archive as locale-archive.default and has created locale-archive.default.list as a text file containing a list of all the locales in that default file. To generate a smaller file, you start with the full default locale-archive, then eliminate locales from the full list using localedef --delete-from-archive, then execute build-locale-archive to finalize the new locale-archive file. To assist in this procedure, Scyld ClusterWare installs helper scripts and some sample locale lists. For example, to rebuild with just the U.S.-English locales:

```
[root@cluster ~] # cd /usr/lib/locale
[root@cluster ~] # ./rebuild-archive.sh locales.English_US
```

Or to include all the English language locales:

```
[root@cluster ~] # cd /usr/lib/locale
[root@cluster ~] # ./rebuild-archive.sh locales.English
```
When executing `rebuild-archive.sh`, this helper script prints details of what is being requested and asks for permission to proceed.

Several other sample `locales.*` files have been provided. The local cluster administrator can use one of these files, or can create a new custom file, as desired. Each such `locales.*` file should contain a list of one or more specific locales (e.g., `en_US.uts8`), or contain patterns that match a locale or locales (e.g., `en_US`), one per line. For example, the `locales.English` file contains:

```bash
# All English language locales
en_
```

which is a pattern that matches every `en_*` locale.

Additionally, Scyld ClusterWare provides `reset-archive.sh`, which is a script that returns `locale-archive` to its original default state.

**Caution**

Note that for both RHEL6 and RHEL7, we recommend always including `en_US*` locales, just to be safe, as the default RHEL/CentOS distributions reference the `LANG=en_US.uts8` locale in several `/etc/` configuration files. Each Scyld ClusterWare 6-supplied `locales.*` file contains the suggested `en_US` locale pattern.

### 4.17.4 Site-Local Startup Scripts

Local, homegrown scripts to be executed at node boot time can be placed in `/etc/beowulf/init.d/`. The conventions for this are as follows:

- Scripts should live in `/etc/beowulf/init.d/`
- Scripts should be numbered in the order in which they are to be executed (e.g., `20raid`, `30startsan`, `45mycustom_hw`)
- Any scripts going into `/etc/beowulf/init.d/` should be cluster aware. That is, they should contain the appropriate `bpsh` and/or `bpcp` commands to make the script work on the compute node rather than on the master node. Examine the Scyld ClusterWare distributed scripts for examples.

Any local modifications to Scyld ClusterWare distributed scripts in `/etc/beowulf/init.d` will be lost across subsequent Scyld ClusterWare updates. If a local sysadmin believes a local modification is necessary, we suggest:

1. Copy the to-be-edited original script to a file with a unique name, e.g.:

   ```bash
   cd /etc/beowulf/init.d
   cp 37some_script 37some_script_local
   ```

2. Remove the executable state of the original:

   ```bash
   beochkconfig 37some_script off
   ```

3. Edit `37some_script_local` as desired.

4. Thereafter, subsequent ClusterWare updates may install a new `37some_script`, but the update will not re-enable the non-executable state of that script. The local `37some_script_local` remains untouched. However, keep in mind that the newer ClusterWare version of `37some_script` may contain fixes or other changes that need to be reflected in `37some_script_local` because that edited file was based upon an older ClusterWare version.
4.17.5 Sample Kickstart Script

Non-Scyld nodes can be provisioned using the Red Hat kickstart utility. The following is a sample kickstart configuration script, which should be edited as appropriate for your local cluster:

```
# centos 5u3 (amd64) hybrid example kickstart
install
reboot
# point to NFS server that exports a directory containing the iso images of centOS 5.3
nfs --server=192.168.5.30 --dir=/eng_local/nfs-install/centos5u3_amd64
lang en_US.UTF-8
keyboard us
xconfig --startxonboot
network --device eth0 --bootproto dhcp --onboot yes
#network --device eth1 --onboot no --bootproto dhcp
rootpw --iscrypted $1$DC2r9BD4$Y1QsTSuL6K9ESdVkJ8eJT0
firewall --disabled
selinux --disabled
authconfig --enablesyslog --enablemd5
timezone --utc America/Los_Angeles
bootloader --location=mbr
key --skip

# The following is commented-out so nobody uses this by accident and
# overwrites their local harddisks on a compute node.
#
# In order to enable using this kickstart script to install an operating system
# on /dev/sda of your compute node and thereby erasing all prior content,
# remove the comment character in front of the next 4 lines:
#
# clearpart --linux --drives=sda
# part /boot --fstype ext3 --size=100 --ondisk=sda
# part swap --fstype swap --size=2040 --ondisk=sda
# part / --fstype ext3 --size=1024 --grow

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%packages
@ ruby
@ system-tools
@ MySQL Database
@ Editors
@ System Tools
@ Text-based Internet
@ Legacy Network Server
@ DNS Name Server
@ FTP Server
@ Network Servers
@ Web Server
@ Server Configuration Tools
@ Sound and Video
@ Administration Tools
@ Graphical Internet
```
@ Engineering and Scientific
@ Development Libraries
@ GNOME Software Development
@ X Software Development
@ Authoring and Publishing
@ Legacy Software Development
@ Emacs
@ Legacy Software Support
@ Ruby
@ KDE Software Development
@ Horde
@ PostgreSQL Database
@ Development Tools
@ Yum Utilities
@ FreeNX and NX
kernel-devel
OpenIPMI-tools
openmpi-devel
sg3_utils

%pre
# any thing you want to happen before the install process starts

%post
#!/bin/bash
# anything you want to happen after the install process finishes
masterip=10.56.10.1
wget http://$masterip/sendstats
chmod +x sendstats
mv sendstats /usr/local/sbin/
echo "/usr/local/sbin/sendstats" >> /etc/rc.local

# If you get the blinking cursor of death and no OS post, then uncomment this.
# grub-install --root-directory=/boot hd0
# grub-install --root-directory=/boot hd1
# grub-install --root-directory=/boot hd2

# Removes rhgb and quiet from grub.conf
sed -i /boot/grub/grub.conf -e 's/rhgb/!/g;s/quiet//g'

# Sets up the serial console in grub.conf
# TODO

# turns off cpuspeed
chkconfig cpuspeed --level 123456 off

# changes xorg.conf from mga to vesa
sed -i /etc/X11/xorg.conf -e 's/mga/vesa/'

(continues on next page)
# turns on ipmi
chkconfig ipmi on
chkconfig sshd on
wget http://10.56.10.1/done
5.1 Preface

Welcome to the Scyld ClusterWare User’s Guide. This manual is for those who will use ClusterWare to run applications, so it presents the basics of ClusterWare parallel computing — what ClusterWare is, what you can do with it, and how you can use it. The manual covers the ClusterWare architecture and discusses the unique features of Scyld ClusterWare. It will show you how to navigate the ClusterWare environment, how to run programs, and how to monitor their performance.

Because this manual is for the user accessing a ClusterWare system that has already been configured, it does not cover how to install, configure, or administer your Scyld cluster. You should refer to other parts of the Scyld documentation set for additional information, specifically:


• If you have not yet built your cluster or installed Scyld ClusterWare, refer to the latest Release Notes and the Installation Guide.

• If you are looking for information on how to administer your cluster, refer to the Administrator’s Guide.

• If you plan to write programs to use on your Scyld cluster, refer to the Programmer’s Guide.

Also not covered is use of the Linux operating system, on which Scyld ClusterWare is based. Some of the basics are presented here, but if you have not used Linux or Unix before, a book or online resource will be helpful. Books by O’Reilly and Associates are good sources of information.

This manual will provide you with information about the basic functionality of the utilities needed to start being productive with Scyld ClusterWare.

5.2 Scyld ClusterWare Overview

Scyld ClusterWare is a Linux-based high-performance computing system. It solves many of the problems long associated with Linux Beowulf-class cluster computing, while simultaneously reducing the costs of system installation, administration, and maintenance. With Scyld ClusterWare, the cluster is presented to the user as a single, large-scale parallel computer.

This chapter presents a high-level overview of Scyld ClusterWare. It begins with a brief history of Beowulf clusters, and discusses the differences between the first-generation Beowulf clusters and a Scyld cluster. A high-level technical summary of Scyld ClusterWare is then presented, covering the top-level features and major software components of Scyld. Finally, typical applications of Scyld ClusterWare are discussed.

Additional details are provided throughout the Scyld ClusterWare documentation set.
5.3 What Is a Beowulf Cluster?

The term “Beowulf” refers to a multi-computer architecture designed for executing parallel computations. A “Beowulf cluster” is a parallel computer system conforming to the Beowulf architecture, which consists of a collection of commodity off-the-shelf computers (COTS) (referred to as “nodes”), connected via a private network running an open-source operating system. Each node, typically running Linux, has its own processor(s), memory storage, and I/O interfaces. The nodes communicate with each other through a private network, such as Ethernet or Infiniband, using standard network adapters. The nodes usually do not contain any custom hardware components, and are trivially reproducible.

One of these nodes, designated as the “master node”, is usually attached to both the private and public networks, and is the cluster’s administration console. The remaining nodes are commonly referred to as “compute nodes”. The master node is responsible for controlling the entire cluster and for serving parallel jobs and their required files to the compute nodes. In most cases, the compute nodes are configured and controlled by the master node. Typically, the compute nodes require neither keyboards nor monitors; they are accessed solely through the master node. From the viewpoint of the master node, the compute nodes are simply additional processor and memory resources.

In conclusion, Beowulf is a technology of networking Linux computers together to create a parallel, virtual supercomputer. The collection as a whole is known as a “Beowulf cluster”. While early Linux-based Beowulf clusters provided a cost-effective hardware alternative to the supercomputers of the day, allowing users to execute high-performance computing applications, the original software implementations were not without their problems. Scyld ClusterWare addresses — and solves — many of these problems.

5.3.1 A Brief History of the Beowulf

Cluster computer architectures have a long history. The early network-of-workstations (NOW) architecture used a group of standalone processors connected through a typical office network, their idle cycles harnessed by a small piece of special software, as shown below.

![Network-of-Workstations Architecture](image)

The NOW concept evolved to the Pile-of-PCs architecture, with one master PC connected to the public network, and
the remaining PCs in the cluster connected to each other and to the master through a private network as shown in the following figure. Over time, this concept solidified into the Beowulf architecture.

![Figure 2. A Basic Beowulf Cluster](image)

For a cluster to be properly termed a “Beowulf”, it must adhere to the “Beowulf philosophy”, which requires:

- Scalable performance
- The use of commodity off-the-shelf (COTS) hardware
- The use of an open-source operating system, typically Linux

Use of commodity hardware allows Beowulf clusters to take advantage of the economies of scale in the larger computing markets. In this way, Beowulf clusters can always take advantage of the fastest processors developed for high-end workstations, the fastest networks developed for backbone network providers, and so on. The progress of Beowulf clustering technology is not governed by any one company’s development decisions, resources, or schedule.

### 5.3.2 First-Generation Beowulf Clusters

The original Beowulf software environments were implemented as downloadable add-ons to commercially-available Linux distributions. These distributions included all of the software needed for a networked workstation: the kernel, various utilities, and many add-on packages. The downloadable Beowulf add-ons included several programming environments and development libraries as individually-installable packages.

With this first-generation Beowulf scheme, every node in the cluster required a full Linux installation and was responsible for running its own copy of the kernel. This requirement created many administrative headaches for the maintainers of Beowulf-class clusters. For this reason, early Beowulf systems tended to be deployed by the software application developers themselves (and required detailed knowledge to install and use). Scyld ClusterWare reduces and/or eliminates these and other problems associated with the original Beowulf-class clusters.
5.3.3 Scyld ClusterWare: A New Generation of Beowulf

Scyld ClusterWare streamlines the process of configuring, administering, running, and maintaining a Beowulf-class cluster computer. It was developed with the goal of providing the software infrastructure for commercial production cluster solutions.

Scyld ClusterWare was designed with the differences between master and compute nodes in mind; it runs only the appropriate software components on each compute node. Instead of having a collection of computers each running its own fully-installed operating system, Scyld creates one large distributed computer. The user of a Scyld cluster will never log into one of the compute nodes nor worry about which compute node is which. To the user, the master node is the computer, and the compute nodes appear merely as attached processors capable of providing computing resources.

With Scyld ClusterWare, the cluster appears to the user as a single computer. Specifically,

- The compute nodes appear as attached processor and memory resources
- All jobs start on the master node, and are migrated to the compute nodes at runtime
- All compute nodes are managed and administered collectively via the master node

The Scyld ClusterWare architecture simplifies cluster setup and node integration, requires minimal system administration, provides tools for easy administration where necessary, and increases cluster reliability through seamless scalability. In addition to its technical advances, Scyld ClusterWare provides a standard, stable, commercially-supported platform for deploying advanced clustering systems. See the next section for a technical summary of Scyld ClusterWare.

5.4 Scyld ClusterWare Technical Summary

Scyld ClusterWare presents a more uniform system view of the entire cluster to both users and applications through extensions to the kernel. A guiding principle of these extensions is to have little increase in both kernel size and complexity and, more importantly, negligible impact on individual processor performance.

In addition to its enhanced Linux kernel, Scyld ClusterWare includes libraries and utilities specifically improved for high-performance computing applications. For information on the Scyld libraries, see the Reference Guide. Information on using the Scyld utilities to run and monitor jobs is provided in Interacting With the System and Running Programs. If you need to use the Scyld utilities to configure and administer your cluster, see the Administrator’s Guide.

5.4.1 Top-Level Features of Scyld ClusterWare

The following list summarizes the top-level features of Scyld ClusterWare.

Security and Authentication. With Scyld ClusterWare, the master node is a single point of security administration and authentication. The authentication envelope is drawn around the entire cluster and its private network. This obviates the need to manage copies or caches of credentials on compute nodes or to add the overhead of networked authentication. Scyld ClusterWare provides simple permissions on compute nodes, similar to Unix file permissions, allowing their use to be administered without additional overhead.

Easy Installation. Scyld ClusterWare is designed to augment a full Linux distribution, such as Red Hat Enterprise Linux (RHEL) or CentOS. The installer used to initiate the installation on the master node is provided on an auto-run CD-ROM. You can install from scratch and have a running Linux HPC cluster in less than an hour. See the Installation Guide for full details.

Install Once, Execute Everywhere. A full installation of Scyld ClusterWare is required only on the master node. Compute nodes are provisioned from the master node during their boot process, and they dynamically cache any additional parts of the system during process migration or at first reference.

Single System Image. Scyld ClusterWare makes a cluster appear as a multi-processor parallel computer. The master node maintains (and presents to the user) a single process space for the entire cluster, known as the BProc Distributed
Process Space. *BProc* is described briefly later in this chapter, and more details are provided in the Administrator’s Guide.

**Execution Time Process Migration.** Scyld ClusterWare stores applications on the master node. At execution time, *BProc* migrates processes from the master to the compute nodes. This approach virtually eliminates both the risk of version skew and the need for hard disks on the compute nodes. More information is provided in the section on process space migration later in this chapter. Also refer to the *BProc* discussion in the Administrator’s Guide.

**Seamless Cluster Scalability.** Scyld ClusterWare seamlessly supports the dynamic addition and deletion of compute nodes without modification to existing source code or configuration files.

**Administration Tools.** Scyld ClusterWare includes simplified tools for performing cluster administration and maintenance. Both graphical user interface (GUI) and command line interface (CLI) tools are supplied. See the Administrator’s Guide for more information.

**Web-Based Administration Tools.** Scyld ClusterWare includes web-based tools for remote administration, job execution, and monitoring of the cluster. See the Administrator’s Guide for more information.

**Additional Features.** Additional features of Scyld ClusterWare include support for cluster power management (IPMI and Wake-on-LAN, easily extensible to other out-of-band management protocols); runtime and development support for MPI and PVM; and support for the LFS and NFS3 file systems.

**Fully-Supported.** Scyld ClusterWare is fully-supported by Penguin Computing, Inc.

### 5.4.2 Process Space Migration Technology

Scyld ClusterWare is able to provide a single system image through its use of the *BProc* Distributed Process Space, the Beowulf process space management kernel enhancement. *BProc* enables the processes running on compute nodes to be visible and managed on the master node. All processes appear in the master node’s process table, from which they are migrated to the appropriate compute node by *BProc*. Both process parent-child relationships and Unix job-control information are maintained with the migrated jobs. The *stdout* and *stderr* streams are redirected to the user’s *ssh* or terminal session on the master node across the network.

The *BProc* mechanism is one of the primary features that makes Scyld ClusterWare different from traditional Beowulf clusters. For more information, see the system design description in the Administrator’s Guide.

### 5.4.3 Compute Node Provisioning

Scyld ClusterWare utilizes light-weight provisioning of compute nodes from the master node’s kernel and Linux distribution. For Scyld Series 30 and Scyld ClusterWare, PXE is the supported method for booting nodes into the cluster; the 2-phase boot sequence of earlier Scyld distributions is no longer used.

The master node is the DHCP server serving the cluster private network. PXE booting across the private network ensures that the compute node boot package is version-synchronized for all nodes within the cluster. This boot package consists of the kernel, *initrd*, and *rootfs*. If desired, the boot package can be customized per node in the Beowulf configuration file */etc/beowulf/config*, which also includes the kernel command line parameters for the boot package.

For a detailed description of the compute node boot procedure, see the system design description in the Administrator’s Guide. Also refer to the chapter on compute node boot options in that document.
5.4.4 Compute Node Categories

Compute nodes seen by the master over the private network are classified into one of three categories by the master node, as follows:

- **Unknown** — A node not formally recognized by the cluster as being either a *Configured* or *Ignored* node. When bringing a new compute node online, or after replacing an existing node’s network interface card, the node will be classified as *unknown*.

- **Ignored** — Nodes which, for one reason or another, you’d like the master node to ignore. These are not considered part of the cluster, nor will they receive a response from the master node during their boot process.

- **Configured** — Those nodes listed in the cluster configuration file using the “node” tag. These are formally part of the cluster, recognized as such by the master node, and used as computational resources by the cluster.

For more information on compute node categories, see the system design description in the *Administrator’s Guide*.

5.4.5 Compute Node States

*BProc* maintains the current condition or “node state” of each configured compute node in the cluster. The compute node states are defined as follows:

- **down** — Not communicating with the master, and its previous state was either *down*, *up*, *error*, *unavailable*, or *boot*.

- **unavailable** — Node has been marked *unavailable* or “off-line” by the cluster administrator; typically used when performing maintenance activities. The node is useable only by the user *root*.

- **error** — Node encountered an error during its initialization; this state may also be set manually by the cluster administrator. The node is useable only by the user *root*.

- **up** — Node completed its initialization without error; node is online and operating normally. This is the only state in which non-*root* users may access the node.

- **reboot** — Node has been commanded to reboot itself; node will remain in this state until it reaches the *boot* state, as described below.

- **halt** — Node has been commanded to halt itself; node will remain in this state until it is reset (or powered back on) and reaches the *boot* state, as described below.

- **pwroff** — Node has been commanded to power itself off; node will remain in this state until it is powered back on and reaches the *boot* state, as described below.

- **boot** — Node has completed its *stage 2* boot but is still initializing. After the node finishes booting, its next state will be either *up* or *error*.

For more information on compute node states, see the system design description in the *Administrator’s Guide*.

5.4.6 Major Software Components

The following is a list of the major software components included with Scyld ClusterWare. For more information, see the relevant sections of the Scyld ClusterWare documentation set, including the *Installation Guide*, *Administrator’s Guide*, *User’s Guide*, *Reference Guide*, and *Programmer’s Guide*.

- **BProc** — The process migration technology; an integral part of Scyld ClusterWare.

- **BeoSetup** — A GUI for configuring the cluster.

- **BeoStatus** — A GUI for monitoring cluster status.

- **beostat** — A text-based tool for monitoring cluster status.
• beoboot — A set of utilities for booting the compute nodes.
• beofdisk — A utility for remote partitioning of hard disks on the compute nodes.
• beoserv — The cluster’s DHCP, PXE and dynamic provisioning server; it responds to compute nodes and serves the boot image.
• BPmaster — The BProc master daemon; it runs on the master node.
• BPslave — The BProc compute daemon; it runs on each of the compute nodes.
• bpstat — A BProc utility that reports status information for all nodes in the cluster.
• bpctl — A BProc command line interface for controlling the nodes.
• bpsl — A BProc utility intended as a replacement for rsh (remote shell).
• bpcp — A BProc utility for copying files between nodes, similar to rcp (remote copy).
• MPI — The Message Passing Interface, optimized for use with Scyld ClusterWare.
• PVM — The Parallel Virtual Machine, optimized for use with Scyld ClusterWare.
• mpируn — A parallel job-creation package for Scyld ClusterWare.

5.5 Typical Applications of Scyld ClusterWare

Scyld clustering provides a facile solution for anyone executing jobs that involve either a large number of computations or large amounts of data (or both). It is ideal for both large, monolithic, parallel jobs and for many normal-sized jobs run many times (such as Monte Carlo type analysis).

The increased computational resource needs of modern applications are frequently being met by Scyld clusters in a number of domains, including:

• **Computationally-Intensive Activities** — Optimization problems, stock trend analysis, financial analysis, complex pattern matching, medical research, genetics research, image rendering
• **Scientific Computing / Research** — Engineering simulations, 3D-modeling, finite element analysis, computational fluid dynamics, computational drug development, seismic data analysis, PCB / ASIC routing
• **Large-Scale Data Processing** — Data mining, complex data searches and results generation, manipulating large amounts of data, data archival and sorting
• **Web / Internet Uses** — Web farms, application serving, transaction serving, data serving

These types of jobs can be performed many times faster on a Scyld cluster than on a single computer. Increased speed depends on the application code, the number of nodes in the cluster, and the type of equipment used in the cluster. All of these can be easily tailored and optimized to suit the needs of your applications.

5.6 Interacting With the System

This chapter discusses how to verify the availability of the nodes in your cluster, how to monitor node status, how to issue commands and copy data to the compute nodes, and how to monitor and control processes. For information on running programs across the cluster, see *Running Programs*. 
5.6.1 Verifying the Availability of Nodes

In order to use a Scyld cluster for computation, at least one node must be available or up. Thus, the first priority when interacting with a cluster is ascertaining the availability of nodes. Unlike traditional Beowulf clusters, Scyld ClusterWare provides rich reporting about the availability of the nodes.

You can use either the BeoStatus GUI tool or the bpstat command to determine the availability of nodes in your cluster. These tools, which can also be used to monitor node status, are described in the next section.

If fewer nodes are up than you think should be, or some nodes report an error, check with your Cluster Administrator.

5.6.2 Monitoring Node Status

You can monitor the status of nodes in your cluster with the BeoStatus GUI tool or with either of two command line tools, bpstat and beostat. These tools are described in the sections that follow. Also see the Reference Guide for information on the various options and flags supported for these tools.

5.6.2.1 The BeoStatus GUI Tool

The BeoStatus graphical user interface (GUI) tool is the best way to check the status of the cluster, including which nodes are available or up. There are two ways to open the BeoStatus GUI as a Gnome X window, as follows.

Click the BeoStatus icon in the tool tray or in the applications pulldown.

Alternatively, type the command beostatus in a terminal window on the master node; you do not need to be a privileged user to use this command.

The default BeoStatus GUI mode is a tabular format known as the “Classic” display (shown in the following figure). You can select different display options from the Mode menu.

![Figure 1. BeoStatus in the “Classic” Display Mode](image-url)

Chapter 5. User’s Guide
BeoStatus Node Information

Each row in the BeoStatus display reports information for a single node, including the following:

- **Node** — The node’s assigned node number, starting at zero. Node -1, if shown, is the master node. The total number of node entries shown is set by the “iprange” or “nodes” keywords in the file /etc/beowulf/config, rather than the number of detected nodes. The entry for an inactive node displays the last reported data in a grayed-out row.

- **Up** — A graphical representation of the node’s status. A green checkmark is shown if the node is up and available. Otherwise, a red “X” is shown.

- **State** — The node’s last known state. This should agree with the state reported by both the bpstat command and in the BeoSetup window.

- **CPU “X”** — The CPU loads for the node’s processors; at minimum, this indicates the CPU load for the first processor in each node. Since it is possible to mix uni-processor and multi-processor machines in a Scyld cluster, the number of CPU load columns is equal to the maximum number of processors for any node in your cluster. The label “N/A” will be shown for nodes with less than the maximum number of processors.

- **Memory** — The node’s current memory usage.

- **Swap** — The node’s current swap space (virtual memory) usage.

- **Disk** — The node’s hard disk usage. If a RAM disk is used, the maximum value shown is one-half the amount of physical memory. As the RAM disk competes with the kernel and application processes for memory, not all the RAM may be available.

- **Network** — The node’s network bandwidth usage. The total amount of bandwidth available is the sum of all network interfaces for that node.

BeoStatus Update Intervals

Once running, BeoStatus is non-interactive; the user simply monitors the reported information. The display is updated at 4-second intervals by default. You can modify this default using the command beostatus -u secs (where secs is the number of seconds) in a terminal window or an ssh session to the master node with X-forwarding enabled.

**Tip**

Each update places load on the master and compute nodes, as well as the interconnection network. Too-frequent updates can degrade the overall system performance.

BeoStatus in Text Mode

In environments where use of the Gnome X window system is undesirable or impractical, such as when accessing the master node through a slow remote network connection, you can view the status of the cluster as curses text output (shown in the following figure). Do do this, enter the command beostatus -c in a terminal window on the master node or an ssh session to the master node.

BeoStatus in text mode reports the same node information as reported by the “Classic” display, except for the graphical indicator of node up (green checkmark) or node down (red X). The data in the text display is updated at 4-second intervals by default.
5.6.2.2 The bpstat Command Line Tool

You can also check node status with the bpstat command. When run at a shell prompt on the master node without options, bpstat prints out a listing of all nodes in the cluster and their current status. You do not need to be a privileged user to use this command.

Following is an example of the outputs from bpstat for a cluster with 10 compute nodes.

```
[user@cluster user]$ bpstat

Node Status Mode User Group
-1 down ---------- root root
0 up ---x--x--x any any
1 up 0.0% 0.0% 0.6% 0.0% 2.9% 0 kEps
2 up 0.0% 0.0% 1.0% 0.0% 2.8% 0 kEps
3 up 0.0% 0.0% 0.5% 0.0% 2.8% 0 kEps
4 up 0.0% 0.0% 1.2% 0.0% 2.9% 0 kEps
5 up 0.0% 0.0% 1.2% 0.0% 2.9% 0 kEps
6 down 53.5% 80.0% 1.5% 0.0% 2.9% 4854 kEps
```

bpstat will show one of the following indicators in the “Status” column:

- A node marked *up* is available to run jobs. This status is the equivalent of the green checkmark in the BeoStatus GUI.
- Nodes that have not yet been configured are marked as *down*. This status is the equivalent of the red X in the BeoStatus GUI.
- Nodes currently booting are temporarily shown with a status of *boot*. Wait 10-15 seconds and try again.
- The “error” status indicates a node initialization problem. Check with your Cluster Administrator.

For additional information on bpstat, see the section on monitoring and controlling processes later in this chapter. Also see the Reference Guide for details on using bpstat and its command line options.
5.6.2.3 The beostat Command Line Tool

You can use the beostat command to display raw status data for cluster nodes. When run at a shell prompt on the master node without options, beostat prints out a listing of stats for all nodes in the cluster, including the master node. You do not need to be a privileged user to use this command.

The following example shows the beostat output for the master node and one compute node:

```
[user@cluster user] $ beostat
model    : 5
model name : AMD Opteron(tm) Processor 248
stepping  : 10
cpu MHz   : 2211.352
cache size : 1024 KB
fdiv_bug : no
hlt_bug   : no
sep_bug   : no
f00f_bug : no
coma_bug  : no
fpu       : yes
fpu_exception : yes
cpuid level : 1
wp        : yes
bogomips  : 4422.05

*** /proc/meminfo *** Sun Sep 17 10:46:33 2006
     total: used: free: shared: buffers: cached:
Mem: 4217454592 318734336 3898720256 0 60628992 0
Swap: 2089209856 0 2089209856
MemTotal: 4118608 kB
MemFree: 3807344 kB
MemShared: 0 kB
Buffers: 59208 kB
Cached: 0 kB
SwapTotal: 2040244 kB
SwapFree: 2040244 kB

*** /proc/loadavg *** Sun Sep 17 10:46:33 2006
3.00 2.28 1.09 178/178 0

*** /proc/net/dev *** Sun Sep 17 10:46:33 2006
          Inter- | Receive | Transmit
face | bytes packets errs drop fifo frame compressed multicast | bytes packets errs |
      |drop fifo colls carrier compressed
eth0: 85209660 615362 0 0 0 0 703311290 |
drop fifo colls carrier compressed
   559376 0 0 0 0 0 0
eth1: 4576500575 13507271 0 0 0 0 0 0 0 0 0 0 0 0
   943033982 13220730 0 0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

*** /proc/stat ***
cpu0 15040 0 466102 25629625  Sun Sep 17 10:46:33 2006
cpu1 17404 0 1328475 24751544  Sun Sep 17 10:46:33 2006
```

(continues on next page)
*** statfs (“/”) *** Sun Sep 17 10:46:33 2006
path: /
f_type: 0xef53
f_bsize: 4096
f_blocks: 48500104
f_bfree: 41439879
f_bavail: 38976212
f_files: 24641536
f_ffree: 24191647
f_fsid: 000000 000000
f_namelen: 255

============== Node: .0 (index 0) ===============

*** /proc/cpuinfo *** Sun Sep 17 10:46:34 2006
num processors : 2
vendor_id : AuthenticAMD
cpu family : 15
model : 5
model name : AMD Opteron(tm) Processor 248
stepping : 10
cpu MHz : 2211.386
cache size : 1024 KB
fdiv_bug : no
hlt_bug : no
sep_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
bogomips : 4422.04

*** /proc/meminfo *** Sun Sep 17 10:46:34 2006

Mem: 4216762368 99139584 4117622784 0 0 0
MemTotal: 4117932 kB
MemFree: 4021116 kB
MemShared: 0 kB
Buffers: 0 kB
Cached: 0 kB
SwapTotal: 0 kB
SwapFree: 0 kB

*** /proc/loadavg *** Sun Sep 17 10:46:34 2006
0.99 0.75 0.54 36/36 0

*** /proc/net/dev *** Sun Sep 17 10:46:34 2006

(continues on next page)
The Reference Guide provides details for using beostat and its command line options.

5.6.3 Issuing Commands

5.6.3.1 Commands on the Master Node

When you log into the cluster, you are actually logging into the master node, and the commands you enter on the command line will execute on the master node. The only exception is when you use special commands for interacting with the compute nodes, as described in the next section.

5.6.3.2 Commands on the Compute Node

Scyld ClusterWare provides the bpsh command for running jobs on the compute nodes. bpsh is a replacement for the traditional Unix utility rsh, used to run a job on a remote computer. Like rsh, the bpsh arguments are the node on which to run the command and the command. bpsh allows you to run a command on more than one node without having to type the command once for each node, but it doesn’t provide an interactive shell on the remote node like rsh does.

bpsh is primarily intended for running utilities and maintenance tasks on a single node or a range of nodes, rather than for running parallel programs. For information on running parallel programs with Scyld ClusterWare, see Running Programs.

bpsh provides a convenient yet powerful interface for manipulating all (or a subset of) the cluster’s nodes simultaneously. bpsh provides you the flexibility to access a compute node individually, but removes the requirement to access each node individually when a collective operation is desired. A number of examples and options are discussed in the sections that follow. For a complete reference to all the options available for bpsh, see the Reference Guide.
Examples for Using bpsh

Example 1. Checking for a File

You can use bpsh to check for specific files on a compute node. For example, to check for a file named output in the /tmp directory of node 3, you would run the following command on the master node:

```
[user@cluster user] $ bpsh 3 ls /tmp/output
```

The command output would appear on the master node terminal where you issued the command.

Example 2. Running a Command on a Range of Nodes

You can run the same command on a range of nodes using bpsh. For example, to check for a file named output in the /tmp directory of nodes 3 through 5, you would run the following command on the master node:

```
[user@cluster user] $ bpsh 3,4,5 ls /tmp/output
```

Example 3. Running a Command on All Available Nodes

Use the -a flag to indicate to bpsh that you wish to run a command on all available nodes. For example, to check for a file named output in the /tmp directory of all nodes currently active in your cluster, you would run the following command on the master node:

```
[user@cluster user] $ bpsh -a ls /tmp/output
```

Note that when using the -a flag, the results are sorted by the response speed of the compute nodes, and are returned without node identifiers. Because this command will produce output for every currently active node, the output may be hard to read if you have a large cluster. For example, if you ran the above command on a 64-node cluster in which half of the nodes have the file being requested, the results returned would be 32 lines of /tmp/output and another 32 lines of ls: /tmp/output: no such file or directory. Without node identifiers, it is impossible to ascertain the existence of the target file on a particular node.

See the next section for bpsh options that enable you to format the results for easier reading.

Formatting bpsh Output

The bpsh command has a number of options for formatting its output to make it more useful for the user, including the following:

- The -L option makes bpsh wait for a full line from a compute node before it prints out the line. Without this option, the output from your command could include half a line from node 0 with a line from node 1 tacked onto the end, then followed by the rest of the line from node 0.

- The -p option prefixes each line of output with the node number of the compute node that produced it. This option causes the functionality for -L to be used, even if not explicitly specified.

- The -s option forces the output of each compute node to be printed in sorted numerical order, rather than by the response speed of the compute nodes. With this option, all the output for node 0 will appear before any of the output for node 1. To add a divider between the output from each node, use the -d option.

- Using -d generates a divider between the output from each node. This option causes the functionality for -s to be used, even if not explicitly specified.
For example, if you run the command `bpsh -a -d -p ls /tmp/output` on an 8-node cluster, the output would make it clear which nodes do and do not have the file `output` in the `/tmp` directory, for example:

```
 0: /tmp/output
 1: ls: /tmp/output: No such file or directory
 2: ls: /tmp/output: No such file or directory
 3: /tmp/output
 4: /tmp/output
 5: /tmp/output
 6: ls: /tmp/output: No such file or directory
 7: ls: /tmp/output: No such file or directory
```

**bpsh and Shell Interaction**

Special shell features, such as piping and input/output redirection, are available to advanced users. This section provides several examples of shell interaction, using the following conventions:

- The command running will be `cmda`.
- If it is piped to anything, it will be piped to `cmdb`.
- If an input file is used, it will be `/tmp/input`.
- If an output file is used, it will be `/tmp/output`.
- The node used will always be node 0.

**Example 4. Command on Compute Node, Output on Master Node**

The easiest case is running a command on a compute node and doing something with its output on the master node, or giving it input from the master. Following are a few examples:

```
[user@cluster user] $ bpsh 0 cmda | cmdb
[user@cluster user] $ bpsh 0 cmda > /tmp/output
[user@cluster user] $ bpsh 0 cmda < /tmp/input
```
Example 5. Command on Compute Node, Output on Compute Node

A bit more complex situation is to run the command on the compute node and do something with its input (or output) on that same compute node. There are two ways to accomplish this.

The first solution requires that all the programs you run be on the compute node. For this to work, you must first copy the `cmda` and `cmdb` executable binaries to the compute node. Then you would use the following commands:

```
[user@cluster user] $ bpsh 0 sh -c "cmda | cmdb"
[user@cluster user] $ bpsh 0 sh -c "cmda > /tmp/output"
[user@cluster user] $ bpsh 0 sh -c "cmda < /tmp/input"
```

The second solution doesn’t require any of the programs to be on the compute node. However, it uses a lot of network bandwidth as it takes the output and sends it to the master node, then sends it right back to the compute node. The appropriate commands are as follows:

```
[user@cluster user] $ cmda | bpsh 0 cmdb
[user@cluster user] $ cmda | bpsh 0 dd of=/tmp/output
[user@cluster user] $ bpsh 0 cat /tmp/input | cmda
```

Example 6. Command on Master Node, Output on Compute Node

You can also run a command on the master node and do something with its input or output on the compute nodes. The appropriate commands are as follows:

```
[user@cluster user] $ cmda | bpsh 0 cmdb
[user@cluster user] $ cmda | bpsh 0 dd of=/tmp/output
[user@cluster user] $ bpsh 0 cat /tmp/input | cmda
```

5.6.4 Copying Data to the Compute Nodes

There are several ways to get data from the master node to the compute nodes. This section describes using NFS to share data, using the Scyld ClusterWare command `bpcp` to copy data, and using programmatic methods for data transfer.

5.6.4.1 Sharing Data via NFS

The easiest way to transfer data to the compute nodes is via NFS. All files in your `/home` directory are shared by default to all compute nodes via NFS. Opening an NFS-shared file on a compute node will, in fact, open the file on the master node; no actual copying takes place.

5.6.4.2 Copying Data via bpcp

To copy a file, rather than changing the original across the network, you can use the `bpcp` command. This works much like the standard Unix file-copying command `cp`, in that you pass it a file to copy as one argument and the destination as the next argument. Like the Unix `scp`, the file paths may be qualified by a computer host name.

With `bpcp`, you can indicate the node number for the source file, destination file, or both. To do this, prepend the node number with a colon before the file name, to specify that the file is on that node or should be copied to that node. For example, to copy the file `/tmp/foo` to the same location on node 1, you would use the following command:
5.6.4.3 Programmatic Data Transfer

The third method for transferring data is to do it programmatically. This is a bit more complex than the methods described in the previous section, and will only be described here only conceptually.

If you are using an MPI job, you can have your Rank 0 process on the master node read in the data, then use MPI’s message passing capabilities to send the data over to a compute node.

If you are writing a program that uses BProc functions directly, you can have the process first read the data while it is on the master node. When the process is moved over to the compute node, it should still be able to access the data read in while on the master node.

5.6.4.4 Data Transfer by Migration

Another programmatic method for file transfer is to read a file into memory prior to calling BProc to migrate the process to another node. This technique is especially useful for parameter and configuration files, or files containing the intermediate state of a computation. See the Reference Guide for a description of the BProc system calls.

5.6.5 Monitoring and Controlling Processes

One of the features of Scyld ClusterWare that isn’t provided in traditional Beowulf clusters is the BProc Distributed Process Space. BProc presents a single unified process space for the entire cluster, run from the master node, where you can see and control jobs running on the compute nodes. This process space allows you to use standard Unix tools, such as top, ps, and kill. See the Administrator’s Guide for more details on BProc.

Scyld ClusterWare also includes a tool called bpstat that can be used to determine which node is running a process. Using the command option bpstat -p will list all processes currently running by processID (PID), with the number of the node running each process. The following output is an example:

```
[user@cluster user] $ bpstat -p

PID  Node
6301  0
6302  1
6303  0
6304  2
6305  1
6313  2
6314  3
6321  3
```

Using the command option bpstat -P (with an uppercase “P” instead of a lowercase “p”) tells bpstat to take the output of the ps and reformat it, pre-pending a column showing the node number. The following two examples show the difference in the outputs from ps and from bpstat -P.

Example output from ps:

```
[user@cluster user] $ ps xf

PID  TTY  STAT  TIME  COMMAND
6503  pts/2  S  0:00  bash
6665  pts/2  R  0:00  ps  xf
```

(continues on next page)
Example of the same `ps` output when run through `bpstat -P` instead:

```
[user@cluster user]$ ps xf | bpstat -P

  NODE   PID   TTY  STAT  TIME COMMAND
6503 pts/2 S  0:00 bash
6666 pts/2 R  0:00 ps xf
6667 pts/2 R  0:00 bpstat -P
6471 pts/3 S  0:00 bash
6538 pts/3 S  0:00 /bin/sh /usr/bin/linpack
6553 pts/3 S  0:00 _ /bin/sh /usr/bin/mpirun -np 5 /tmp/xhpl
6654 pts/3 R  0:06 _ /tmp/xhpl -p4pg /tmp/PI6553 -p4wd /tmp
6655 pts/3 S  0:00 _ /tmp/xhpl -p4pg /tmp/PI6553 -p4wd /tmp
6656 pts/3 RW 0:06 _ [xhpl]
6658 pts/3 SW 0:00 | _ [xhpl]
6657 pts/3 RW 0:06 _ [xhpl]
6660 pts/3 SW 0:00 | _ [xhpl]
6659 pts/3 RW 0:06 _ [xhpl]
6662 pts/3 SW 0:00 | _ [xhpl]
6661 pts/3 SW 0:00 _ [xhpl]
6663 pts/3 SW 0:00 _ [xhpl]
```

For additional information on `bpstat`, see the section on monitoring node status earlier in this chapter. For information on the `bpstat` command line options, see the Reference Guide.

### 5.7 Running Programs

This chapter describes how to run both serial and parallel jobs with Scyld ClusterWare, and how to monitor the status of the cluster once your applications are running. It begins with a brief discussion of program execution concepts, including some examples. The discussion then covers running programs that aren’t parallelized, running parallel programs (including MPI-aware and PVM-aware programs), running serial programs in parallel, job batching, and file systems.
5.7.1 Program Execution Concepts

This section compares program execution on a stand-alone computer and a Scyld cluster. It also discusses the differences between running programs on a traditional Beowulf cluster and a Scyld cluster. Finally, it provides some examples of program execution on a Scyld cluster.

5.7.1.1 Stand-Alone Computer vs. Scyld Cluster

On a stand-alone computer running Linux, Unix, and most other operating systems, executing a program is a very simple process. For example, to generate a list of the files in the current working directory, you open a terminal window and type the command `ls` followed by the `[return]` key. Typing the `[return]` key causes the command shell — a program that listens to and interprets commands entered in the terminal window — to start the `ls` program (stored at `/bin/ls`). The output is captured and directed to the standard output stream, which also appears in the same window where you typed the command.

A Scyld cluster isn’t simply a group of networked stand-alone computers. Only the master node resembles the computing system with which you are familiar. The compute nodes have only the minimal software components necessary to support an application initiated from the master node. So for instance, running the `ls` command on the master node causes the same series of actions as described above for a stand-alone computer, and the output is for the master node only.

However, running `ls` on a compute node involves a very different series of actions. Remember that a Scyld cluster has no resident applications on the compute nodes; applications reside only on the master node. So for instance, to run the `ls` command on compute node 1, you would enter the command `bpsh 1 ls` on the master node. This command sends `ls` to compute node 1 via Scyld’s BProc software, and the output stream is directed to the terminal window on the master node, where you typed the command.

Some brief examples of program execution are provided in the last section of this chapter. Both BProc and bpsh are covered in more detail in the Administrator’s Guide.

5.7.1.2 Traditional Beowulf Cluster vs. Scyld Cluster

A job on a Beowulf cluster is actually a collection of processes running on the compute nodes. In traditional clusters of computers, and even on earlier Beowulf clusters, getting these processes started and running together was a complicated task. Typically, the cluster administrator would need to do all of the following:

- Ensure that the user had an account on all the target nodes, either manually or via a script.
- Ensure that the user could spawn jobs on all the target nodes. This typically entailed configuring a `hosts.allow` file on each machine, creating a specialized PAM module (a Linux authentication mechanism), or creating a server daemon on each node to spawn jobs on the user’s behalf.
- Copy the program binary to each node, either manually, with a script, or through a network file system.
- Ensure that each node had available identical copies of all the dependencies (such as libraries) needed to run the program.
- Provide knowledge of the state of the system to the application manually, through a configuration file, or through some add-on scheduling software.

With Scyld ClusterWare, most of these steps are removed. Jobs are started on the master node and are migrated out to the compute nodes via BProc. A cluster architecture where jobs may be initiated only from the master node via BProc provides the following advantages:

- Users no longer need accounts on remote nodes.
- Users no longer need authorization to spawn jobs on remote nodes.
• Neither binaries nor libraries need to be available on the remote nodes.

• The BProc system provides a consistent view of all jobs running on the system.

With all these complications removed, program execution on the compute nodes becomes a simple matter of letting BProc know about your job when you start it. The method for doing so depends on whether you are launching a parallel program (for example, an MPI job or PVM job) or any other kind of program. See the sections on running parallel programs and running non-parallelized programs later in this chapter.

5.7.1.3 Program Execution Examples

This section provides a few examples of program execution with Scyld ClusterWare. Additional examples are provided in the sections on running parallel programs and running non-parallelized programs later in this chapter.

Example 1. Directed Execution with bpsh

In the directed execution mode, the user explicitly defines which node (or nodes) will run a particular job. This mode is invoked using the bpsh command, the ClusterWare shell command analogous in functionality to both the rsh (remote shell) and ssh (secure shell) commands. Following are two examples of using bpsh.

The first example runs hostname on compute node 0 and writes the output back from the node to the user’s screen:

```
[user@cluster user] $ bpsh 0 /bin/hostname
n0
```

If /bin is in the user’s $PATH, then the bpsh does not need the full pathname:

```
[user@cluster user] $ bpsh 0 hostname
n0
```

The second example runs the /usr/bin/uptime utility on node 1. Assuming /usr/bin is in the user’s $PATH:

```
[user@cluster user] $ bpsh 1 uptime
12:56:44 up 4:57, 5 users, load average: 0.06, 0.09, 0.03
```

Example 2. Dynamic Execution with beorun and mpprun

In the dynamic execution mode, Scyld decides which node is the most capable of executing the job at that moment in time. Scyld includes two parallel execution tools that dynamically select nodes: beorun and mpprun. They differ only in that beorun runs the job concurrently on the selected nodes, while mpprun runs the job sequentially on one node at a time.

The following example shows the difference in the elapsed time to run a command with beorun vs. mpprun:

```
[user@cluster user] $ date;beorun -np 8 sleep 1;date
Fri Aug 18 11:48:30 PDT 2006
Fri Aug 18 11:48:31 PDT 2006
```

```
[user@cluster user] $ date;mpprun -np 8 sleep 1;date
Fri Aug 18 11:48:46 PDT 2006
Fri Aug 18 11:48:54 PDT 2006
```
Example 3. Binary Pre-Staged on Compute Node

A needed binary can be “pre-staged” by copying it to a compute node prior to execution of a shell script. In the following example, the shell script is in a file called test.sh:

```
######
#! /bin/bash
hostname.local
######

[user@cluster user] $ bpsh 1 mkdir -p /usr/local/bin
[user@cluster user] $ bpcp /bin/hostname 1:/usr/local/bin/hostname.local
[user@cluster user] $ bpsh 1 ./test.sh
```

This makes the `hostname` binary available on compute node 1 as `/usr/local/bin/hostname.local` before the script is executed. The shell’s $PATH contains `/usr/local/bin`, so the compute node searches locally for `hostname.local` in $PATH, finds it, and executes it.

Note that copying files to a compute node generally puts the files into the RAM filesystem on the node, thus reducing main memory that might otherwise be available for programs, libraries, and data on the node.

Example 4. Binary Migrated to Compute Node

If a binary is not “pre-staged” on a compute node, the full path to the binary must be included in the script in order to execute properly. In the following example, the master node starts the process (in this case, a shell) and moves it to node 1, then continues execution of the script. However, when it comes to the `hostname.local2` command, the process fails:

```
######
#! /bin/bash
hostname.local2
######

[user@cluster user] $ bpsh 1 ./test.sh
./test.sh: line 2: hostname.local2: command not found
```

Since the compute node does not have `hostname.local2` locally, the shell attempts to resolve the binary by asking for the binary from the master. The problem is that the master has no idea which binary to give back to the node, hence the failure.

Because there is no way for `Bproc` to know which binaries may be needed by the shell, `hostname.local2` is not migrated along with the shell during the initial startup. Therefore, it is important to provide the compute node with a full path to the binary:

```
######
#! /bin/bash
/tmp/hostname.local2
######

[user@cluster user] $ cp /bin/hostname /tmp/hostname.local2
[user@cluster user] $ bpsh 1 ./test.sh
```

5.7. Running Programs
With a full path to the binary, the compute node can construct a proper request for the master, and the master knows which exact binary to return to the compute node for proper execution.

**Example 5. Process Data Files**

Files that are opened by a process (including files on disk, sockets, or named pipes) are not automatically migrated to compute nodes. Suppose the application BOB needs the data file 1.dat:

```
er@cluster user] $ bpsh 1 /usr/local/BOB/bin/BOB 1.dat
```

1.dat must be either pre-staged to the compute node, e.g., using bpcp to copy it there; or else the data files must be accessible on an NFS-mounted file system. The file `/etc/beowulf/fstab` (or a node-specific `fstab.nodeNumber`) specifies which filesystems are NFS-mounted on each compute node by default.

**Example 6. Installing Commercial Applications**

Through the course of its execution, the application BOB in the example above does some work with the data file 1.dat, and then later attempts to call `/usr/local/BOB/bin/BOB.helper.bin` and `/usr/local/BOB/bin/BOB.cleanup.bin`. If these binaries are not in the memory space of the process during migration, the calls to these binaries will fail. Therefore, `/usr/local/BOB` should be NFS-mounted to all of the compute nodes, or the binaries should be pre-staged using bpcp to copy them by hand to the compute nodes. The binaries will stay on each compute node until that node is rebooted.

Generally for commercial applications, the administrator should have `$APP_HOME` NFS-mounted on the compute nodes that will be involved in execution. A general best practice is to mount a general directory such as `/opt`, and install all of the applications into `/opt`.

### 5.7.2 Environment Modules

The RHEL/CentOS environment-modules package provides for the dynamic modification of a user’s environment via modulefiles. Each modulefile contains the information needed to configure the shell for an application, allowing a user to easily switch between applications with a simple `module switch` command that resets environment variables like `PATH` and `LD_LIBRARY_PATH`. A number of modules are already installed that configure application builds and execution with OpenMPI, MPICH2, and MVAPICH2. Execute the command `module avail` to see a list of available modules. See specific sections, below, for examples of how to use modules.

For more information about creating your own modules, see [http://modules.sourceforge.net](http://modules.sourceforge.net), or view the manpages `man module` and `man modulefile`.

### 5.7.3 Running Programs That Are Not Parallelized

#### 5.7.3.1 Starting and Migrating Programs to Compute Nodes (bpsh)

There are no executable programs (binaries) on the file system of the compute nodes. This means that there is no getty, no login, nor any shells on the compute nodes.

Instead of the remote shell (`rsh`) and secure shell (`ssh`) commands that are available on networked stand-alone computers (each of which has its own collection of binaries), Scyld ClusterWare has the `bpsh` command. The following example shows the standard `ls` command running on node 2 using `bpsh`:
At startup time, by default Scyld ClusterWare exports various directories, e.g., /bin and /usr/bin, on the master node, and those directories are NFS-mounted by compute nodes.

However, an NFS-accessible /bin/ls is not a requirement for bpsh 2 ls to work. Note that the /sbin directory also exists on the compute node. It is not exported by the master node by default, and thus it exists locally on a compute node in the RAM-based filesystem. bpsh 2 ls /sbin usually shows an empty directory. Nonetheless, bpsh 2 modprobe bproc executes successfully, even though which modprobe shows the command resides in /sbin/modprobe and bpsh 2 which modprobe fails to find the command on the compute node because its /sbin does not contain modprobe.

bpsh 2 modprobe bproc works because the bpsh initiates a modprobe process on the master node, then forms a process memory image that includes the command's binary and references to all its dynamically linked libraries. This process memory image is then copied (migrated) to the compute node, and there the references to dynamic libraries are remapped in the process address space. Only then does the modprobe command begin real execution.

bpsh is not a special version of sh, but a special way of handling execution. This process works with any program. Be aware of the following:

- All three standard I/O streams — stdin, stdout, and stderr — are forwarded to the master node. Since some programs need to read standard input and will stop working if they're run in the background, be sure to close standard input at invocation by using the bpsh -n flag when you run a program in the background on a compute node.

- Because shell scripts expect executables to be present, and because compute nodes don't meet this requirement, shell scripts should be modified to include the bpsh commands required to affect the compute nodes and run on the master node.

- The dynamic libraries are cached separately from the process memory image, and are copied to the compute node only if they are not already there. This saves time and network bandwidth. After the process completes, the dynamic libraries are unloaded from memory, but they remain in the local cache on the compute node, so they won't need to be copied if needed again.

For additional information on the BProc Distributed Process Space and how processes are migrated to compute nodes, see the Administrator's Guide.

### 5.7.3.2 Copying Information to Compute Nodes (bpcp)

Just as traditional Unix has copy (cp), remote copy (rcp), and secure copy (scp) to move files to and from networked machines, Scyld ClusterWare has the bpcp command.

Although the default sharing of the master node’s home directories via NFS is useful for sharing small files, it is not a good solution for large data files. Having the compute nodes read large data files served via NFS from the master node will result in major network congestion, or even an overload and shutdown of the NFS server. In these cases, staging data files on compute nodes using the bpcp command is an alternate solution. Other solutions include using dedicated NFS servers or NAS appliances, and using cluster file systems.

Following are some examples of using bpcp.

This example shows the use of bpcp to copy a data file named foo2.dat from the current directory to the /tmp directory on node 6:

```
[user@cluster user] $ bpcp foo2.dat 6:/tmp
```
The default directory on the compute node is the current directory on the master node. The current directory on the compute node may already be NFS-mounted from the master node, but it may not exist. The example above works, since /tmp exists on the compute node, but will fail if the destination does not exist. To avoid this problem, you can create the necessary destination directory on the compute node before copying the file, as shown in the next example.

In this example, we change to the /tmp/foo directory on the master, use bpsh to create the same directory on the node 6, then copy foo2.dat to the node:

```
[user@cluster user]$ cd /tmp/foo
[user@cluster user]$ bpsh 6 mkdir /tmp/foo
[user@cluster user]$ bpcp foo2.dat 6:
```

This example copies foo2.dat from node 2 to node 3 directly, without the data being stored on the master node. As in the first example, this works because /tmp exists:

```
[user@cluster user]$ bpcp 2:/tmp/foo2.dat 3:/tmp
```

5.7.4 Running Parallel Programs

5.7.4.1 An Introduction to Parallel Programming APIs

Programmers are generally familiar with serial, or sequential, programs. Simple programs — like “Hello World” and the basic suite of searching and sorting programs — are typical of sequential programs. They have a beginning, an execution sequence, and an end; at any time during the run, the program is executing only at a single point.

A thread is similar to a sequential program, in that it also has a beginning, an execution sequence, and an end. At any time while a thread is running, there is a single point of execution. A thread differs in that it isn’t a stand-alone program; it runs within a program. The concept of threads becomes important when a program has multiple threads running at the same time and performing different tasks.

To run in parallel means that more than one thread of execution is running at the same time, often on different processors of one computer; in the case of a cluster, the threads are running on different computers. A few things are required to make parallelism work and be useful: The program must migrate to another computer or computers and get started; at some point, the data upon which the program is working must be exchanged between the processes.

The simplest case is when the same single-process program is run with different input parameters on all the nodes, and the results are gathered at the end of the run. Using a cluster to get faster results of the same non-parallel program with different inputs is called parametric execution.

A much more complicated example is a simulation, where each process represents some number of elements in the system. Every few time steps, all the elements need to exchange data across boundaries to synchronize the simulation. This situation requires a message passing interface or MPI.

To solve these two problems — program startup and message passing — you can develop your own code using POSIX interfaces. Alternatively, you could utilize an existing parallel application programming interface (API), such as the Message Passing Interface (MPI) or the Parallel Virtual Machine (PVM). These are discussed in the sections that follow.
MPI

The Message Passing Interface (MPI) application programming interface is currently the most popular choice for writing parallel programs. The MPI standard leaves implementation details to the system vendors (like Scyld). This is useful because they can make appropriate implementation choices without adversely affecting the output of the program.

A program that uses MPI is automatically started a number of times and is allowed to ask two questions: How many of us (size) are there, and which one am I (rank)? Then a number of conditionals are evaluated to determine the actions of each process. Messages may be sent and received between processes.

The advantages of MPI are that the programmer:

- Doesn’t have to worry about how the program gets started on all the machines
- Has a simplified interface for inter-process messages
- Doesn’t have to worry about mapping processes to nodes
- Abstracts the network details, resulting in more portable hardware-agnostic software

Also see the section on running MPI-aware programs later in this chapter. Scyld ClusterWare includes several implementations of MPI:

MPICH. Scyld ClusterWare 6 (and earlier releases) includes MPICH, a freely-available implementations of the MPI standard, and a project that is managed by Argonne National Laboratory. NOTE: MPICH is deprecated and removed from ClusterWare 7 and later releases, and supplanted by MPICH2 and beyond. Visit https://www.mpich.org for more information. Scyld MPICH is modified to use BProc and Scyld job mapping support; see the section on job mapping later in this chapter.

MVAPICH. MVAPICH is an implementation of MPICH for Infiniband interconnects. NOTE: MVAPICH is deprecated and removed from ClusterWare 7 and later releases, and supplanted by MVAPICH2 and beyond. Visit http://mvapich.cse.ohio-state.edu/ for more information. Scyld MVAPICH is modified to use BProc and Scyld job mapping support; see the section on job mapping later in this chapter.

MPICH2. Scyld ClusterWare includes MPICH2, a second generation MPICH. Visit https://www.mpich.org for more information. Scyld MPICH2 is customized to use environment modules. See MPICH2 Release Information for details.

MVAPICH2. MVAPICH2 is second generation MVAPICH. Visit http://mvapich.cse.ohio-state.edu/ for more information. Scyld MVAPICH2 is customized to use environment modules. See MVAPICH2 Release Information for details.

OpenMPI. OpenMPI is an open-source implementation of the Message Passing Interface 2 (MPI-2) specification. The OpenMPI implementation is an optimized combination of several other MPI implementations. Visit https://www.open-mpi.org for more information. Also see OpenMPI Release Information for details.

Other MPI Implementations. Various commercial MPI implementations run on Scyld ClusterWare. Visit the Penguin Computing Support Portal at https://www.penguincomputing.com/support for more information. You can also download and build your own version of MPI, and configure it to run on Scyld ClusterWare.

PVM

Parallel Virtual Machine (PVM) was an earlier parallel programming interface. Unlike MPI, it is not a specification but a single set of source code distributed on the Internet. PVM reveals much more about the details of starting your job on remote nodes. However, it fails to abstract implementation details as well as MPI does.

PVM is deprecated, but is still in use by legacy code. We generally advise against writing new programs in PVM, but some of the unique features of PVM may suggest its use.

Also see the section on running PVM-aware programs later in this chapter.

5.7. Running Programs
Custom APIs

As mentioned earlier, you can develop your own parallel API by using various Unix and TCP/IP standards. In terms of starting a remote program, there are programs written:

- Using the `rexec` function call
- To use the `rexec` or `rsh` program to invoke a sub-program
- To use Remote Procedure Call (RPC)
- To invoke another sub-program using the `inetd` super server

These solutions come with their own problems, particularly in the implementation details. What are the network addresses? What is the path to the program? What is the account name on each of the computers? How is one going to load-balance the cluster?

Scyld ClusterWare, which doesn’t have binaries installed on the cluster nodes, may not lend itself to these techniques. We recommend you write your parallel code in MPI. That having been said, we can say that Scyld has some experience with getting `rexec()` calls to work, and that one can simply substitute calls to `rsh` with the more cluster-friendly `bpsh`.

5.7.4.2 Mapping Jobs to Compute Nodes

Running programs specifically designed to execute in parallel across a cluster requires at least the knowledge of the number of processes to be used. Scyld ClusterWare uses the `NP` environment variable to determine this. The following example will use 4 processes to run an MPI-aware program called `a.out`, which is located in the current directory.

```
[user@cluster user] $ NP=4 ./a.out
```

Note that each kind of shell has its own syntax for setting environment variables; the example above uses the syntax of the Bourne shell (`/bin/sh` or `/bin/bash`).

What the example above does not specify is which specific nodes will execute the processes; this is the job of the mapper. Mapping determines which node will execute each process. While this seems simple, it can get complex as various requirements are added. The mapper scans available resources at the time of job submission to decide which processors to use.

Scyld ClusterWare includes `beomap`, a mapping API (documented in the `Programmer’s Guide` with details for writing your own mapper). The mapper’s default behavior is controlled by the following environment variables:

- `NP` — The number of processes requested, but not the number of processors. As in the example earlier in this section, `NP=4 ./a.out` will run the MPI program `a.out` with 4 processes.
- `ALL_CPUS` — Set the number of processes to the number of CPUs available to the current user. Similar to the example above, `--all-cpus=1 ./a.out` would run the MPI program `a.out` on all available CPUs.
- `ALL_NODES` — Set the number of processes to the number of nodes available to the current user. Similar to the `ALL_CPUS` variable, but you get a maximum of one CPU per node. This is useful for running a job per node instead of per CPU.
- `ALL_LOCAL` — Run every process on the master node; used for debugging purposes.
- `NO_LOCAL` — Don’t run any processes on the master node.
- `EXCLUDE` — A colon-delimited list of nodes to be avoided during node assignment.
- `BEOWULF_JOB_MAP` — A colon-delimited list of nodes. The first node listed will be the first process (MPI Rank 0) and so on.
You can use the beomap program to display the current mapping for the current user in the current environment with the current resources at the current time. See the Reference Guide for a detailed description of beomap and its options, as well as examples for using it.

5.7.4.3 Running MPICH and MVAPICH Programs

NOTE: MPICH and MVAPICH (version 1) are deprecated and removed from Scyld ClusterWare

MPI-aware programs are those written to the MPI specification and linked with Scyld MPI libraries. NOTE: MPICH and MVAPICH are deprecated and have been supplanted by MPICH2 and MVAPICH2 (and newer versions of those packages). Applications that use MPICH (Ethernet “p4”) or MVAPICH (Infiniband “vapi”) are compiled and linked with common MPI/MVAPICH implementation libraries, plus specific compiler family (e.g., gnu, Intel, PGI) libraries. The same application binary can execute either in an Ethernet interconnection environment or an Infiniband interconnection environment that is specified at run time. This section discusses how to run these programs and how to set mapping parameters from within such programs.

For information on building MPICH/MVAPICH programs, see the Programmer’s Guide.

mpirun

Almost all implementations of MPI have an mpirun program, which shares the syntax of mpprun, but which boasts of additional features for MPI-aware programs.

In the Scyld implementation of mpirun, all of the options available via environment variables or flags through directed execution are available as flags to mpirun, and can be used with properly compiled MPI jobs. For example, the command for running a hypothetical program named my-mpi-prog with 16 processes:

```
[user@cluster user] $ mpirun -np 16 my-mpi-prog arg1 arg2
```

is equivalent to running the following commands in the Bourne shell:

```
[user@cluster user] $ export NP=16
[user@cluster user] $ my-mpi-prog arg1 arg2
```

Setting Mapping Parameters from Within a Program

A program can be designed to set all the required parameters itself. This makes it possible to create programs in which the parallel execution is completely transparent. However, it should be noted that this will work only with Scyld ClusterWare, while the rest of your MPI program should work on any MPI platform.

Use of this feature differs from the command line approach, in that all options that need to be set on the command line can be set from within the program. This feature may be used only with programs specifically designed to take advantage of it, rather than any arbitrary MPI program. However, this option makes it possible to produce turn-key application and parallel library functions in which the parallelism is completely hidden.

Following is a brief example of the necessary source code to invoke mpirun with the -np 16 option from within a program, to run the program with 16 processes:

```c
/* Standard MPI include file */
#include <mpi.h>

main(int argc, char **argv) {
    setenv("NP","16",1); // set up mpirun env vars
    (continues on next page)
}
MPI_Init(&argc,&argv);
MPI_Finalize();
}

More details for setting mapping parameters within a program are provided in the *Programmer's Guide*.

**Examples**

The examples in this section illustrate certain aspects of running a hypothetical MPI-aware program named `my-mpi-prog`.

**Example 7. Specifying the Number of Processes**

This example shows a cluster execution of a hypothetical program named `my-mpi-prog` run with 4 processes:

```
[user@cluster user] $ NP=4 ./my-mpi-prog
```

An alternative syntax is as follows:

```
[user@cluster user] $ NP=4
[user@cluster user] $ export NP
[user@cluster user] $ ./my-mpi-prog
```

Note that the user specified neither the nodes to be used nor a mechanism for migrating the program to the nodes. The mapper does these tasks, and jobs are run on the nodes with the lowest CPU utilization.

In addition to specifying the number of processes to create, you can also exclude specific nodes as computing resources. In this example, we run `my-mpi-prog` again, but this time we not only specify the number of processes to be used (NP=6), but we also exclude of the master node (NO_LOCAL=1) and some cluster nodes (EXCLUDE=2:4:5) as computing resources.

```
[user@cluster user] $ NP=6 NO_LOCAL=1 EXCLUDE=2:4:5 ./my-mpi-prog
```

**5.7.4.4 Running OpenMPI Programs**

OpenMPI programs are those written to the MPI-2 specification. This section provides information needed to use programs with OpenMPI as implemented in Scyld ClusterWare.

**Pre-Requisites to Running OpenMPI**

A number of commands, such as `mpirun`, are duplicated between OpenMPI and other MPI implementations. The environment-modules package gives users a convenient way to switch between the various implementations. Each module bundles together various compiler-specific environment variables to configure your shell for building and running your application, and for accessing compiler-specific manpages. Be sure that you are loading the proper module to match the compiler that built the application you wish to run. For example, to load the OpenMPI module for use with the Intel compiler, do the following:

```
[user@cluster user] $ module load openmpi/intel
```

Currently, there are modules for the GNU, Intel, and PGI compilers. To see a list of all of the available modules:
Using OpenMPI

OpenMPI does not honor the Scyld ClusterWare job mapping environment variables. You must either specify the list of hosts on the command line or inside a hostfile. To specify the list of hosts on the command line, use the -H option. The argument following -H is a comma separated list of hostnames, not node numbers. For example, to run a two process job, with one process running on node 0 and one on node 1:

```
[user@cluster user] $ mpirun -H n0,n1 -np 2 ./mpiprog
```

Support for running jobs over Infiniband using the OpenIB transport is included with OpenMPI distributed with Scyld ClusterWare. Much like running a job with MPICH over Infiniband, one must specifically request the use of OpenIB. For example:

```
[user@cluster user] $ mpirun --mca btl openib,sm,self -H n0,n1 -np 2 ./myprog
```

Read the OpenMPI `mpirun` man page for more information about, using a hostfile, and using other tunable options available through `mpirun`.

### 5.7.4.5 Running MPICH2 and MVAPICH2 Programs

MPICH2 and MVAPICH2 programs are those written to the MPI-2 specification. This section provides information needed to use programs with MPICH2 or MVAPICH2 as implemented in Scyld ClusterWare.

**Pre-Requisites to Running MPICH2/MVAPICH2**

As with Scyld OpenMPI, the Scyld MPICH2 and MVAPICH2 distributions are repackaged Open Source MPICH2 and MVAPICH2 that utilize environment modules to build and to execute applications. Each module bundles together various compiler-specific environment variables to configure your shell for building and running your application, and for accessing implementation- and compiler-specific manpages. You must use the same module to both build the application and to execute it. For example, to load the MPICH2 module for use with the Intel compiler, do the following:

```
[user@cluster user] $ module load mpich2/intel
```

Currently, there are modules for the GNU, Intel, and PGI compilers. To see a list of all of the available modules:

```
[user@cluster user] $ module avail mpich2 mvapich2
```

For more information about creating your own modules, see [http://modules.sourceforge.net](http://modules.sourceforge.net) and the manpages `man module` and `man modulefile`.
Using MPICH2

Unlike the Scyld ClusterWare MPICH implementation, MPICH2 does not honor the Scyld ClusterWare job mapping environment variables. Use mpiexec to execute MPICH2 applications. After loading an mpich2 module, see the man mpiexec manpage for specifics, and visit https://www.mpich.org for full documentation.

Using MVAPICH2

MVAPICH2 does not honor the Scyld ClusterWare job mapping environment variables. Use mpirun_rsh to execute MVAPICH2 applications. After loading an mvapich2 module, use mpirun_rsh --help to see specifics, and visit http://mvapich.cse.ohio-state.edu/ for full documentation.

5.7.4.6 Running PVM-Aware Programs

Parallel Virtual Machine (PVM) is an application programming interface for writing parallel applications, enabling a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. Scyld has developed the Scyld PVM library, specifically tailored to allow PVM to take advantage of the technologies used in Scyld ClusterWare. A PVM-aware program is one that has been written to the PVM specification and linked against the Scyld PVM library.

A complete discussion of cluster configuration for PVM is beyond the scope of this document. However, a brief introduction is provided here, with the assumption that the reader has some background knowledge on using PVM.

You can start the master PVM daemon on the master node using the PVM console, pvm. To add a compute node to the virtual machine, issue an add . command, where # is replaced by a node’s assigned number in the cluster.

Tip

You can generate a list of node numbers using bpstat command.

Alternately, you can start the PVM console with a hostfile filename on the command line. The hostfile should contain a .# for each compute node you want as part of the virtual machine. As with standard PVM, this method automatically spawns PVM slave daemons to the specified compute nodes in the cluster. From within the PVM console, use the conf command to list your virtual machine’s configuration; the output will include a separate line for each node being used. Once your virtual machine has been configured, you can run your PVM applications as you normally would.

5.7.4.7 Porting Other Parallelized Programs

Programs written for use on other types of clusters may require various levels of change to function with Scyld ClusterWare. For instance:

- Scripts or programs that invoke rsh can instead call bpsh.
- Scripts or programs that invoke rcp can instead call bpcp.
- beomap can be used with any script to load balance programs that are to be dispatched to the compute nodes.

For more information on porting applications, see the Programmer’s Guide.
5.7.5 Running Serial Programs in Parallel

For jobs that are not “MPI-aware” or “PVM-aware”, but need to be started in parallel, Scyld ClusterWare provides the parallel execution utilities `mpprun` and `beorun`. These utilities are more sophisticated than `bpsh`, in that they can automatically select ranges of nodes on which to start your program, run tasks on the master node, determine the number of CPUs on a node, and start a copy on each CPU. Thus, `mpprun` and `beorun` provide you with true “dynamic execution” capabilities, whereas `bpsh` provides “directed execution” only.

`mpprun` and `beorun` are very similar, and have similar parameters. They differ only in that `mpprun` runs jobs sequentially on the selected processors, while `beorun` runs jobs concurrently on the selected processors.

5.7.5.1 `mpprun`

`mpprun` is intended for applications rather than utilities, and runs them sequentially on the selected nodes. The basic syntax of `mpprun` is as follows:

```
$ mpprun [options] app arg1 arg2...
```

where `app` is the application program you wish to run; it need not be a parallel program. The `arg` arguments are the values passed to each copy of the program being run.

**Options**

`mpprun` includes options for controlling various aspects of the job, including the ability to:

- Specify the number of processors on which to start copies of the program
- Start one copy on each node in the cluster
- Start one copy on each CPU in the cluster
- Force all jobs to run on the master node
- Prevent any jobs from running on the master node

The most interesting of the options is the `--map` option, which lets the user specify which nodes will run copies of a program; an example is provided in the next section. This argument, if specified, overrides the mapper’s selection of resources that it would otherwise use.

See the *Reference Guide* for a complete list of options for `mpprun`.

**Examples**

Run 16 tasks of program `app`:

```
[user@cluster user] $ mpprun -np 16 app infile outfile
```

Run 16 tasks of program `app` on any available nodes except nodes 2 and 3:

```
[user@cluster user] $ mpprun -np 16 --exclude 2:3 app infile outfile
```

Run 4 tasks of program `app` with task 0 on node 4, task 1 on node 2, task 2 on node 1, and task 3 on node 5:

```
[user@cluster user] $ mpprun --map 4:2:1:5 app infile outfile
```
5.7.5.2 beorun

beorun is intended for applications rather than utilities, and runs them concurrently on the selected nodes. The basic syntax of beorun is as follows:

[user@cluster user] $ beorun [options] app arg1 arg2...

where app is the application program you wish to run; it need not be a parallel program. The arg arguments are the values passed to each copy of the program being run.

Options

beorun includes options for controlling various aspects of the job, including the ability to:

- Specify the number of processors on which to start copies of the program
- Start one copy on each node in the cluster
- Start one copy on each CPU in the cluster
- Force all jobs to run on the master node
- Prevent any jobs from running on the master node

The most interesting of the options is the --map option, which lets the user specify which nodes will run copies of a program; an example is provided in the next section. This argument, if specified, overrides the mapper's selection of resources that it would otherwise use.

See the Reference Guide for a complete list of options for beorun.

Examples

Run 16 tasks of program app:

[user@cluster user] $ beorun -np 16 app infile outfile

Run 16 tasks of program app on any available nodes except nodes 2 and 3:

[user@cluster user] $ beorun -np 16 --exclude 2:3 app infile outfile

Run 4 tasks of program app with task 0 on node 4, task 1 on node 2, task 2 on node 1, and task 3 on node 5:

[user@cluster user] $ beorun --map 4:2:1:5 app infile outfile

5.7.6 Job Batching

5.7.6.1 Job Batching Options for ClusterWare

For Scyld ClusterWare, the default installation includes both the TORQUE resource manager and the Slurm workload manager, each providing users an intuitive interface for remotely initiating and managing batch jobs on distributed compute nodes. TORQUE is an Open Source tool based on standard OpenPBS. Slurm is another Open Source tool, employing the Open Source Munge for authentication and mysql (for ClusterWare 6) or mariadb (for ClusterWare 7 and beyond) for managing a database. Basic instructions for using TORQUE are provided in the next section. For more general product information, see http://www.adaptivecomputing.com/ for Adaptive Computing’s TORQUE information and https://slurm.schedmd.com for Slurm information.
Only one job manager can be enabled at any one time. See the Scyld ClusterWare Administrator’s Guide for details about how to enable either TORQUE or Slurm. If Slurm is the chosen job manager, then users must setup the PATH and LD_LIBRARY_PATH environment variables to properly access the Slurm commands. This is done automatically for users who login when the slurm service is running and the pbs_server is not running, via the /etc/profile.d/scyld.slurm.sh script. Alternatively, each Slurm user can manually execute module load slurm or can add that command line to (for example) the user’s .bash_profile.

The https://slurm.schedmd.com Slurm website also provides an optional TORQUE wrapper to minimize the syntactic differences between TORQUE and Slurm commands and scripts. See https://slurm.schedmd.com/rosetta.pdf for a discussion of the differences between TORQUE and Slurm, and https://slurm.schedmd.com/faq.html#torque provides useful information about how to switch from PBS or TORQUE to Slurm.

Scyld also redistributes the Scyld Maui job scheduler, also derived from Adaptive Computing, that functions in conjunction with the TORQUE job manager. The alternative Moab job scheduler is also available from Adaptive Computing with a separate license, giving customers additional job scheduling, reporting, and monitoring capabilities.

In addition, Scyld provides support for most popular open source and commercial schedulers and resource managers, including SGE, LSF, and PBSPro. For the latest information, visit the Penguin Computing Support Portal at https://www.penguincomputing.com/support.

### 5.7.6.2 Job Batching with TORQUE

The default installation is configured as a simple job serializer with a single queue named batch.

You can use the TORQUE resource manager to run jobs, check job status, find out which nodes are running your job, and find job output.

#### Running a Job

To run a job with TORQUE, you can put the commands you would normally use into a job script, and then submit the job script to the cluster using qsub. The qsub program has a number of options that may be supplied on the command line or as special directives inside the job script. For the most part, these options should behave exactly the same in a job script or via the command line, but job scripts make it easier to manage your actions and their results.

#### Example 9. Starting a Job with a Job Script Using One Node

Following are some examples of running a job using qsub. For more detailed information on qsub, see the qsub man page.

The following script declares a job with the name “myjob”, to be run using one node. The script uses the PBS -N directive, launches the job, and finally sends the current date and working directory to standard output.

```
#!/bin/sh

## Set the job name
#PBS -N myjob
#PBS -l nodes=1

# Run my job
/path/to/myjob

echo Date: $

echo Dir: $PWD
```
You would submit “myjob” as follows:

```
[bjosh@iceberg]$ qsub -l nodes=1 myjob
```

**Example 10. Starting a Job from the Command Line**

This example provides the command line equivalent of the job run in the example above. We enter all of the `qsub` options on the initial command line. Then `qsub` reads the job commands line-by-line until we type `^D`, the end-of-file character. At that point, `qsub` queues the job and returns the Job ID.

```
[bjosh@iceberg]$ qsub -N myjob -l nodes=1:ppn=1 -j oe
cd $PBS_0_WORKDIR
echo Date: $
echo Dir: $PWD
^D
```

**Example 11. Starting an MPI Job with a Job Script**

The following script declares an MPI job named “mpijob”. The script uses the `PBS -N` directive, prints out the nodes that will run the job, launches the job using `mpirun`, and finally prints out the current date and working directory. When submitting MPI jobs using TORQUE, it is recommended to simply call `mpirun` without any arguments. `mpirun` will detect that it is being launched from within TORQUE and assure that the job will be properly started on the nodes TORQUE has assigned to the job. In this case, TORQUE will properly manage and track resources used by the job.

```
## Set the job name
#PBS -N mpijob

# RUN my job
mpirun /path/to/mpijob

echo Date: $
echo Dir: $PWD
```

To request 8 total processors to run “mpijob”, you would submit the job as follows:

```
[bjosh@iceberg]$ qsub -l nodes=8 mpijob
```

To request 8 total processors, using 4 nodes, each with 2 processors per node, you would submit the job as follows:

```
[bjosh@iceberg]$ qsub -l nodes=4:ppn=2 mpijob
```
Checking Job Status

You can check the status of your job using `qstat`. The command line option `qstat -n` will display the status of queued jobs. To watch the progression of events, use the `watch` command to execute `qstat -n` every 2 seconds by default; type `[CTRL]-C` to interrupt `watch` when needed.

Example 12. Checking Job Status

This example shows how to check the status of the job named “myjob”, which we ran on 1 node in the first example above, using the option to watch the progression of events.

```
[bjosh@iceberg]$ qsub myjob && watch qstat -n
iceberg:
JobID   Username   Queue  Jobname    SessID  NDS  TSK  ReqdMemory    ReqdTime  S  ElapTime
15.iceberg bjosh default myjob  --  1  --  --   00:01  Q  --
```

Table 1. Useful Job Status Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>`ps -ef</td>
<td>bpstat -P`</td>
</tr>
<tr>
<td><code>qstat -Q</code></td>
<td>Display status of all queues</td>
</tr>
<tr>
<td><code>qstat -n</code></td>
<td>Display status of queued jobs</td>
</tr>
<tr>
<td><code>qstat -f JOBID</code></td>
<td>Display very detailed information about Job ID</td>
</tr>
<tr>
<td><code>pbsnodes -a</code></td>
<td>Display status of all nodes</td>
</tr>
</tbody>
</table>

Finding Out Which Nodes Are Running a Job

To find out which nodes are running your job, use the following commands:

- To find your Job Ids: `qstat -an`
- To find the Process IDs of your jobs: `qstat -f`
- To find the number of the node running your job: `ps -ef | bpstat -P | grep`  
  
The number of the node running your job will be displayed in the first column of output.

Finding Job Output

When your job terminates, TORQUE will store its output and error streams in files in the script’s working directory.

- Default output file: `.o`
  
  You can override the default using `qsub` with the ``-o`` option on the command line, or use the ```#PBS -o``` directive in your job script.

- Default error file: `.e`
  
  You can override the default using `qsub` with the ```-e``` option on the command line, or use the ```#PBS -e``` directive in your job script.
• To join the output and error streams into a single file, use `qsub` with the `-j oe` option on the command line, or use the `#PBS -j oe` directive in your job script.

### 5.7.6.3 Job Batching with POD Tools

**POD Tools** is a collection of tools for submitting TORQUE jobs to a remote cluster and for monitoring them. POD Tools is useful for, but not limited to, submitting and monitoring jobs to a remote Penguin On Demand cluster. POD Tools executes on both Scyld and non-Scyld client machines, and the Tools communicate with the `beoweb` service that must be executing on the target cluster.

The primary tool in POD Tools is **POD Shell (podsh)**, which is a command-line interface that allows for remote job submission and monitoring. POD Shell is largely self-documented. Enter `podsh --help` for a list of possible commands and their formats.

The general usage is `podsh [OPTIONS] [FILE/ID]`. The action specifies what type of action to perform, such as `submit` (for submitting a new job) or `status` (for collecting status on all jobs or a specific job).

POD Shell can upload a TORQUE job script to the target cluster, where it will be added to the job queue. Additionally, POD Shell can be used to stage data in and out of the target cluster. Staging data in (i.e. copying data to the cluster) is performed across an unencrypted TCP socket. Staging data out (i.e. from the cluster back to the client machine) is performed using `scp` from the cluster to the client. In order for this transfer to be successful, password-less authentication must be in place using SSH keys between the cluster’s master node and the client.

POD Shell uses a configuration file that supports both site-wide and user-local values. Site-wide values are stored in entries in `/etc/podtools.conf`. These settings can be overridden by values in a user’s `~/.podtools/podtools.conf` file. These values can again be overridden by command-line arguments passed to `podsh`. The template for `podtools.conf` is found at `/opt/scyld/podtools/podtools.conf.template`.

### 5.7.7 Using Singularity

Scyld ClusterWare 7 distributes Singularity, a powerful Linux container platform designed by Lawrence Berkeley National Laboratory.

Singularity enables users to have full control of their environment, allowing a non-privileged user to “swap out” the operating system on the host by executing a lightweight Singularity container environment and an application that executes within that environment. For example, Singularity can provide a user with the ability to create an Ubuntu image of their application, and run the containerized application on a RHEL7 or CentOS7 ClusterWare system in its native Ubuntu environment.

Refer to the Singularity documentation at [https://www.sylabs.io/docs/](https://www.sylabs.io/docs/) for instructions on how to create and use Singularity containers.

When running MPI-enabled applications with Singularity on Scyld ClusterWare, follow these additional instructions:

- Always compile MPI applications inside a container image with the same MPI implementation and version you plan to use on your Scyld ClusterWare system. Refer to the Singularity documentation for currently supported MPI implementations.

- Be aware of the MPI transports which are compatible with your containerized binary, and ensure that you use the same MPI transport when executing MPI applications through Singularity. For example, Scyld ClusterWare’s OpenMPI packages support TCP, Verbs, PSM and PSM2 MPI transports, but not all operating systems will support this gamut of options. Adjust your `mpirun` accordingly on Scyld ClusterWare to use the MPI transport supported by your containerized application.

For example, after building a container image and an OpenMPI executable binary that was built for that image:
5.7.8 File Systems

Data files used by the applications processed on the cluster may be stored in a variety of locations, including:

- On the local disk of each node
- On the master node’s disk, shared with the nodes through a network file system
- On disks on multiple nodes, shared with all nodes through the use of a parallel file system

The simplest approach is to store all files on the master node, as with the standard Network File System. Any files in your /home directory are shared via NFS with all the nodes in your cluster. This makes management of the files very simple, but in larger clusters the performance of NFS on the master node can become a bottleneck for I/O-intensive applications. If you are planning a large cluster, you should include disk drives that are separate from the system disk to contain your shared files; for example, place /home on a separate pair of RAID1 disks in the master node. A more scalable solution is to utilize a dedicated NFS server with a properly configured storage system for all shared files and programs, or a high performance NAS appliance.

Storing files on the local disk of each node removes the performance problem, but makes it difficult to share data between tasks on different nodes. Input files for programs must be distributed manually to each of the nodes, and output files from the nodes must be manually collected back on the master node. This mode of operation can still be useful for temporary files created by a process and then later reused on that same node.

5.8 Glossary of Parallel Computing Terms

- **Bandwidth.** A measure of the total amount of information delivered by a network. This metric is typically expressed in millions of bits per second (Mbps) for data rate on the physical communication media or megabytes per second (MBps) for the performance seen by the application.

- **Backplane Bandwidth.** The total amount of data that a switch can move through it in a given time, typically much higher than the bandwidth delivered to a single node.

- **Bisection Bandwidth.** The amount of data that can be delivered from one half of a network to the other half in a given time, through the least favorable halving of the network fabric.

- **Boot Image.** The file system and kernel seen by a compute node at boot time; contains enough drivers and information to get the system up and running on the network.

- **Cluster.** A collection of nodes, usually dedicated to a single purpose.

- **Compute Node.** Nodes attached to the master through an interconnection network, used as dedicated attached processors. With Scyld, users never need to directly log into compute nodes.

- **Data Parallel.** A style of programming in which multiple copies of a single program run on each node, performing the same instructions while operating on different data.

- **Efficiency.** The ratio of a program’s actual speed-up to its theoretical maximum.

- **FLOPS.** Floating-point operations per second, a key measure of performance for many scientific and numerical applications.
Grain Size, Granularity. A measure of the amount of computation a node can perform in a given problem between communications with other nodes, typically defined as “coarse” (large amount of computation) or “fine” (small amount of computation). Granularity is a key in determining the performance of a particular process on a particular cluster.

High Availability. Refers to level of reliability; usually implies some level of fault tolerance (ability to operate in the presence of a hardware failure).

Hub. A device for connecting the NICs in an interconnection network. Only one pair of ports (a bus) can be active at any time. Modern interconnections utilize switches, not hubs.

Isoefficiency. The ability of a process to maintain a constant efficiency if the size of the process scales with the size of the machine.

Jobs. In traditional computing, a job is a single task. A parallel job can be a collection of tasks, all working on the same problem but running on different nodes.

Kernel. The core of the operating system, the kernel is responsible for processing all system calls and managing the system’s physical resources.

Latency. The length of time from when a bit is sent across the network until the same bit is received. Can be measured for just the network hardware (wire latency) or application-to-application (includes software overhead).

Local Area Network (LAN). An interconnection scheme designed for short physical distances and high bandwidth, usually self-contained behind a single router.

MAC Address. On an Ethernet NIC, the hardware address of the card. MAC addresses are unique to the specific NIC, and are useful for identifying specific nodes.

Master Node. Node responsible for interacting with users, connected to both the public network and interconnection network. The master node controls the compute nodes.

Message Passing. Exchanging information between processes, frequently on separate nodes.

Middleware. A layer of software between the user’s application and the operating system.

MPI. The Message Passing Interface, the standard for producing message passing libraries.

MPICH. A commonly used MPI implementation, built on the chameleon communications layer.

Network Interface Card (NIC). The device through which a node connects to the interconnection network. The performance of the NIC and the network it attaches to limit the amount of communication that can be done by a parallel program.

Node. A single computer system (motherboard, one or more processors, memory, possibly a disk, network interface).

Parallel Programming. The art of writing programs that are capable of being executed on many processors simultaneously.

Process. An instance of a running program.

Process Migration. Moving a process from one computer to another after the process begins execution.

PVM. The Parallel Virtual Machine, a common message passing library that predates MPI.

Scalability. The ability of a process to maintain efficiency as the number of processors in the parallel machine increases.

Single System Image. All nodes in the system see identical system files, including the same kernel, libraries, header files, etc. This guarantees that a program that will run on one node will run on all nodes.

Socket. A low-level construct for creating a connection between processes on a remote system.

Speedup. A measure of the improvement in the execution time of a program on a parallel computer vs. a serial computer.

Switch. A device for connecting the NICs in an interconnection network so that all pairs of ports can communicate simultaneously.
Version Skew. The problem of having more than one version of software or files (kernel, tools, shared libraries, header files) on different nodes.

5.9 TORQUE and Maui Release Information

TORQUE software downloads from Adaptive Computing: https://www.adaptivecomputing.com/products/opensource/torque


Adaptive Computing's TORQUE release notes are found at https://www.adaptivecomputing.com/support/documentation-index/torque-resource-manager-documentation

5.10 OpenMPI Release Information

The following is reproduced essentially verbatim from files contained within the OpenMPI tarball downloaded from https://www.open-mpi.org.

Copyright (c) 2004-2010 The Trustees of Indiana University and Indiana University Research and Technology Corporation. All rights reserved.

Copyright (c) 2004-2006 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2004-2008 High Performance Computing Center Stuttgart, University of Stuttgart. All rights reserved.

Copyright (c) 2004-2006 The Regents of the University of California. All rights reserved.

Copyright (c) 2006-2017 Cisco Systems, Inc. All rights reserved.

Copyright (c) 2006 Voltaire, Inc. All rights reserved.

Copyright (c) 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

Copyright (c) 2006-2017 Los Alamos National Security, LLC. All rights reserved.

Copyright (c) 2010-2017 IBM Corporation. All rights reserved.

Copyright (c) 2012 Oak Ridge National Labs. All rights reserved.

Copyright (c) 2012-2017 Sandia National Laboratories. All rights reserved.

Copyright (c) 2012 University of Houston. All rights reserved.

Copyright (c) 2013 NVIDIA Corporation. All rights reserved.

Copyright (c) 2013-2017 Intel, Inc. All rights reserved.

Copyright (c) 2017 Research Organization for Information Science and Technology (RIST). All rights reserved.

Copyright (c) 2018 Amazon.com, Inc. or its affiliates. All Rights reserved.

Additional copyrights may follow.

As more fully described in the "Software Version Number" section in the README file, Open MPI typically releases two separate version series simultaneously. Since these series have different goals and

(continues on next page)
are semi-independent of each other, a single NEWS-worthy item may be introduced into different series at different times. For example, feature F was introduced in the vA.B series at version vA.B.C, and was later introduced into the vX.Y series at vX.Y.Z.

The first time feature F is released, the item will be listed in the vA.B.C section, denoted as:

(*** also to appear: X.Y.Z) -- indicating that this item is also likely to be included in future release version vX.Y.Z.

When vX.Y.Z is later released, the same NEWS-worthy item will also be included in the vX.Y.Z section and be denoted as:

(*** also appeared: A.B.C) -- indicating that this item was previously included in release version vA.B.C.

4.0.1 -- March, 2019

-------------------

- Update embedded PMIx to 3.1.2.
- Fix an issue with Vader (shared-memory) transport on OS-X. Thanks to Daniel Vollmer for reporting.
- Fix a problem with the usNIC BTL Makefile. Thanks to George Marselis for reporting.
- Fix an issue when using --enable-visibility configure option and older versions of hwloc. Thanks to Ben Menadue for reporting and providing a fix.
- Fix an issue with MPI_WIN_CREATE_DYNAMIC and MPI_GET from self. thanks to Bart Janssens for reporting.
- Fix an issue of excessive compiler warning messages from mpi.h when using newer C++ compilers. Thanks to @Shadow-fax for reporting.
- Fix a problem when building Open MPI using clang 5.0.
- Fix a problem with MPI_WIN_CREATE when using UCX. Thanks to Adam Simpson for reporting.
- Fix a memory leak encountered for certain MPI datatype destructor operations. Thanks to Axel Huebl for reporting.
- Fix several problems with MPI RMA accumulate operations. Thanks to Jeff Hammond for reporting.
- Fix possible race condition in closing some file descriptors during job launch using mpirun. Thanks to Jason Williams for reporting and providing a fix.
- Fix a problem in OMPIO for large individual write operations. Thanks to Axel Huebl for reporting.
- Fix a problem with parsing of map-by ppr options to mpirun. Thanks to David Rich for reporting.
- Fix a problem observed when using the mpool hugepage component. Thanks to Hunter Easterday for reporting and fixing.
- Fix valgrind warning generated when invoking certain MPI Fortran data type creation functions. Thanks to @rtoijala for reporting.
- Fix a problem when trying to build with a PMIX 3.1 or newer release. Thanks to Alastair McKinstry for reporting.
- Fix a problem encountered with building MPI F08 module files. Thanks to Igor Andriyash and Axel Huebl for reporting.
- Fix two memory leaks encountered for certain MPI-RMA usage patterns. Thanks to Joseph Schuchart for reporting and fixing.
- Fix a problem with the ORTE rmaps_base_oversubscribe MCA paramater. Thanks to @iassiour for reporting.
- Fix a problem with UCX PML default error handler for MPI communicators. Thanks to Marcin Krotkiewski for reporting.
- Fix various issues with OMPIO uncovered by the testmpio test suite.

4.0.0 -- September, 2018
------------------------
- OSHMEM updated to the OpenSHMEM 1.4 API.
- Do not build OpenSHMEM layer when there are no SPMLs available. Currently, this means the OpenSHMEM layer will only build if a MXM or UCX library is found.
- A UCX BTL was added for enhanced MPI RMA support using UCX
- With this release, OpenIB BTL now only supports iWarp and RoCE by default.
- Updated internal HWLOC to 2.0.2
- Updated internal PMIx to 3.0.2
- Change the priority for selecting external verses internal HWLOC and PMIx packages to build. Starting with this release, configure by default selects available external HWLOC and PMIx packages over the internal ones.
- Updated internal ROMIO to 3.2.1.
- Removed support for the MXM MTL.
- Removed support for SCIF.
- Improved CUDA support when using UCX.
- Enable use of CUDA allocated buffers for OMPIO.
- Improved support for two phase MPI I/O operations when using OMPIO.
- Added support for Software-based Performance Counters, see https://github.com/davideberius/ompi/wiki/How-to-Use-Software-Based-Performance-Counters-(SPCs)-in-Open-MPI
- Change MTL OFI from opting-IN on "psm,psm2,gni" to opting-OUT on "shm,sockets,tcp,udp,rstream"
- Various improvements to MPI RMA performance when using RDMA capable interconnects.
- Update memkind component to use the memkind 1.6 public API.
- Fix a problem with javadoc builds using OpenJDK 11. Thanks to Siegmart Gross for reporting.
- Fix a memory leak using UCX. Thanks to Charles Taylor for reporting.
- Fix hangs in MPI_FINALIZE when using UCX.
- Fix a problem with building Open MPI using an external PMIx 2.1.2 library. Thanks to Marcin Krotkiewski for reporting.
- Fix race conditions in Vader (shared memory) transport.
- Fix problems with use of newer map-by mpirun options. Thanks to Tony Reina for reporting.
- Fix rank-by algorithms to properly rank by object and span
- Allow for running as root of two environment variables are set.
Requested by Axel Huebl.
- Fix a problem with building the Java bindings when using Java 10. Thanks to Bryce Glover for reporting.
- Fix a problem with ORTE not reporting error messages if an application terminated normally but exited with non-zero error code. Thanks to Emre Brookes for reporting.

3.1.3 -- October, 2018
----------------------
- Fix race condition in MPI_THREAD_MULTIPLE support of non-blocking send/receive path.
- Fix error handling SIGCHLD forwarding.
- Add support for CHARACTER and LOGICAL Fortran datatypes for MPI_SIZEOF.
- Fix compile error when using OpenJDK 11 to compile the Java bindings.
- Fix crash when using a hostfile with a 'user@host' line.
- Numerous Fortran '08 interface fixes.
- TCP BTL error message fixes.
- OFI MTL now will use any provider other than shm, sockets, tcp, udp, or rstream, rather than only supporting gni, psm, and psm2.
- Disable async receive of CUDA buffers by default, fixing a hang on large transfers.
- Support the BCM57XXX and BCM58XXX Broadcom adapters.
- Fix minmax datatype support in ROMIO.
- Bug fixes in vader shared memory transport.
- Support very large buffers with MPI_TYPE_VECTOR.
- Fix hang when launching with mpirun on Cray systems.

3.1.2 -- August, 2018
---------------------
- A subtle race condition bug was discovered in the "vader" BTL (shared memory communications) that, in rare instances, can cause MPI processes to crash or incorrectly classify (or effectively drop) an MPI message sent via shared memory. If you are using the "ob1" PML with "vader" for shared memory communication (note that vader is the default for shared memory communication with ob1), you need to upgrade to v3.1.2 or later to fix this issue. You may also upgrade to the following versions to fix this issue:
  - Open MPI v2.1.5 (expected end of August, 2018) or later in the v2.1.x series
  - Open MPI v3.0.1 (released March, 2018) or later in the v3.0.x series
- Assorted Portals 4.0 bug fixes.
- Fix for possible data corruption in MPI_BSEND.
- Move shared memory file for vader btl into /dev/shm on Linux.
- Fix for MPI_ISCATTER/MPI_ISCATTERV Fortran interfaces with MPI_IN_PLACE.
- Upgrade PMIx to v2.1.3.
- Numerous One-sided bug fixes.
- Fix for race condition in uGNI BTL.
- Improve handling of large number of interfaces with TCP BTL.
- Numerous UCX bug fixes.
3.1.1 -- June, 2018
-------------------
- Fix potential hang in UCX PML during MPI_FINALIZE
- Update internal PMIx to v2.1.2rc2 to fix forward version compatibility.
- Add new MCA parameter osc_sm_backing_store to allow users to specify where in the filesystem the backing file for the shared memory one-sided component should live. Defaults to /dev/shm on Linux.
- Fix potential hang on non-x86 platforms when using builds with optimization flags turned off.
- Disable osc/pt2pt when using MPI_THREAD_MULTIPLE due to numerous race conditions in the component.
- Fix dummy variable names for the mpi and mpi_f08 Fortran bindings to match the MPI standard. This may break applications which use name-based parameters in Fortran which used our internal names rather than those documented in the MPI standard.
- Revamp Java detection to properly handle new Java versions which do not provide a javah wrapper.
- Fix RMA function signatures for use-mpi-f08 bindings to have the asynchronous property on all buffers.
- Improved configure logic for finding the UCX library.

3.1.0 -- May, 2018
------------------
- Various OpenSHMEM bug fixes.
- Properly handle array_of_commands argument to Fortran version of MPI_COMM_SPAWN_MULTIPLE.
- Fix bug with MODE_SEQUENTIAL and the sharedfp MPI-IO component.
- Use "javac -h" instead of "javah" when building the Java bindings with a recent version of Java.
- Fix mis-handling of jostepid under SLURM that could cause problems with PathScale/OmniPath NICs.
- Disable the POWER 7/BE block in configure. Note that POWER 7/BE is still not a supported platform, but it is no longer automatically disabled. See https://github.com/open-mpi/ompi/issues/4349#issuecomment-374970982 for more information.
- The output-filename option for mpirun is now converted to an absolute path before being passed to other nodes.
- Add monitoring component for PML, OSC, and COLL to track data movement of MPI applications. See ompi/mca/common/monitoring/HowTo_pml_monitoring.tex for more information about the monitoring framework.
- Add support for communicator assertions: mpi_assert_no_any_tag, mpi_assert_no_any_source, mpi_assert_exact_length, and mpi_assert_allow_overtaking.
- Update PMIx to version 2.1.1.
- Update hwloc to 1.11.7.
- Many one-sided behavior fixes.
- Improved performance for Reduce and Allreduce using Rabenseifner's algorithm.

(continues on next page)
- Revamped mpirun --help output to make it a bit more manageable.
- Portals4 MTL improvements: Fix race condition in rendezvous protocol and retry logic.
- UCX OSC: initial implementation.
- UCX PML improvements: add multi-threading support.
- Yalla PML improvements: Fix error with irregular contiguous datatypes.
- Openib BTL: disable XRC support by default.
- TCP BTL: Add check to detect and ignore connections from processes that aren't MPI (such as IDS probes) and verify that source and destination are using the same version of Open MPI, fix issue with very large message transfer.
- ompi_info parsable output now escapes double quotes in values, and also quotes values can contains colons. Thanks to Lev Givon for the suggestion.
- CUDA-aware support can now handle GPUs within a node that do not support CUDA IPC. Earlier versions would get error and abort.
- Add a mca parameter ras_base_launch_orted_on_hn to allow for launching MPI processes on the same node where mpirun is executing using a separate orte daemon, rather than the mpirun process. This may be useful to set to true when using SLURM, as it improves interoperability with SLURM's signal propagation tools. By default it is set to false, except for Cray XC systems.
- Remove LoadLeveler RAS support.
- Remove IB XRC support from the OpenIB BTL due to lack of support.
- Add functionality for IBM s390 platforms. Note that regular regression testing does not occur on the s390 and it is not considered a supported platform.
- Remove support for big endian PowerPC.
- Remove support for XL compilers older than v13.1.
- Remove support for atomic operations using MacOS atomics library.

3.0.2 -- June, 2018

-------------

- Disable osc/pt2pt when using MPI_THREAD_MULTIPLE due to numerous race conditions in the component.
- Fix dummy variable names for the mpi and mpi_f08 Fortran bindings to match the MPI standard. This may break applications which use name-based parameters in Fortran which used our internal names rather than those documented in the MPI standard.
- Fixed MPI_SIZEOF in the "mpi" Fortran module for the NAG compiler.
- Fix RMA function signatures for use-mpi-f08 bindings to have the asynchronous property on all buffers.
- Fix Fortran MPI_COMM_SPAWN_MULTIPLE to properly follow the count length argument when parsing the array_of_commands variable.
- Revamp Java detection to properly handle new Java versions which do not provide a javah wrapper.
- Improved configure logic for finding the UCX library.
- Add support for HDR InfiniBand link speeds.
- Disable the POWER 7/BE block in configure. Note that POWER 7/BE is still not a supported platform, but it is no longer automatically disabled. See https://github.com/open-mpi/ompi/issues/4349#issuecomment-374970982
for more information.

3.0.1 -- March, 2018
---------------------
- Fix ability to attach parallel debuggers to MPI processes.
- Fix a number of issues in MPI I/O found by the HDF5 test suite.
- Fix (extremely) large message transfers with shared memory.
- Fix out of sequence bug in multi-NIC configurations.
- Fix stdin redirection bug that could result in lost input.
- Disable the LSF launcher if CSM is detected.
- Plug a memory leak in MPI_Mem_free(). Thanks to Philip Blakely for reporting.
- Fix the tree spawn operation when the number of nodes is larger than the radix.
  Thanks to Carlos Eduardo de Andrade for reporting.
- Fix Fortran 2008 macro in MPI extensions. Thanks to Nathan T. Weeks for reporting.
- Add UCX to list of interfaces that OpenSHMEM will use by default.
- Add --{enable|disable}-show-load-errors-by-default to control default behavior of the load errors option.
- OFI MTL improvements: handle empty completion queues properly, fix incorrect error message around fi_getinfo(), use default progress option for provider by default, Add support for reading multiple CQ events in ofi_progress.
- PSM2 MTL improvements: Allow use of GPU buffers, thread fixes.
- Numerous corrections to memchecker behavior.
- Add a mca parameter ras_base_launch_orted_on_hn to allow for launching MPI processes on the same node where mpirun is executing using a separate orted daemon, rather than the mpirun process. This may be useful to set to true when using SLURM, as it improves interoperability with SLURM's signal propagation tools. By default it is set to false, except for Cray XC systems.
- Fix a problem reported on the mailing separately by Kevin McGrattan and Stephen Guzik about consistency issues on NFS file systems when using OMPI. This fix also introduces a new mca parameter fs_ufs_lock_algorithm which allows to control the locking algorithm used by ompio for read/write operations. By default, ompio does not perform locking on local UNIX file systems, locks the entire file per operation on NFS file systems, and selective byte-range locking on other distributed file systems.
- Add an mca parameter pmix_server_usock_connections to allow mpirun to support applications statically built against the Open MPI v2.x release, or installed in a container along with the Open MPI v2.x libraries. It is set to false by default.

3.0.0 -- September, 2017
------------------------
Major new features:
- Use UCX allocator for OSHMEM symmetric heap allocations to optimize intra-node data transfers. UCX SPML only.
- Use UCX multi-threaded API in the UCX PML. Requires UCX 1.0 or later.
- Added support for Flux PMI
- Update embedded PMIx to version 2.1.0

(continues on next page)
- Update embedded hwloc to version 1.11.7

Changes in behavior compared to prior versions:

- Per Open MPI's versioning scheme (see the README), increasing the major version number to 3 indicates that this version is not ABI-compatible with prior versions of Open MPI. In addition, there may be differences in MCA parameter names and defaults from previous releases. Command line options for mpirun and other commands may also differ from previous versions. You will need to recompile MPI and OpenSHMEM applications to work with this version of Open MPI.
- With this release, Open MPI supports MPI_THREAD_MULTIPLE by default.
- New configure options have been added to specify the locations of libnl and zlib.
- A new configure option has been added to request Flux PMI support.
- The help menu for mpirun and related commands is now context based.
  "mpirun --help compatibility" generates the help menu in the same format as previous releases.

Removed legacy support:
- AIX is no longer supported.
- Loadleve is no longer supported.
- OpenSHMEM currently supports the UCX and MXM transports via the ucx and ikrit SPMLs respectively.
- Remove IB XRC support from the OpenIB BTL due to lack of support.
- Remove support for big endian PowerPC.
- Remove support for XL compilers older than v13.1

Known issues:

- MPI_Connect/accept between applications started by different mpirun commands will fail, even if ompi-server is running.

2.1.5 -- August 2018
--------------

- A subtle race condition bug was discovered in the "vader" BTL (shared memory communications) that, in rare instances, can cause MPI processes to crash or incorrectly classify (or effectively drop) an MPI message sent via shared memory. If you are using the "obl" PML with 'vader' for shared memory communication (note that vader is the default for shared memory communication with obl), you need to upgrade to v2.1.5 to fix this issue. You may also upgrade to the following versions to fix this issue:
  - Open MPI v3.0.1 (released March, 2018) or later in the v3.0.x series
  - Open MPI v3.1.2 (expected end of August, 2018) or later
- A link issue was fixed when the UCX library was not located in the linker-default search paths.

2.1.4 -- August, 2018
------------------
Bug fixes/minor improvements:
- Disable the POWER 7/BE block in configure. Note that POWER 7/BE is still not a supported platform, but it is no longer automatically disabled. See https://github.com/open-mpi/ompi/issues/4349#issuecomment-374970982 for more information.
- Fix bug with request-based one-sided MPI operations when using the "rdma" component.
- Fix issue with large data structure in the TCP BTL causing problems in some environments. Thanks to @lgarithm for reporting the issue.
- Minor Cygwin build fixes.
- Minor fixes for the openib BTL:
  - Support for the QLogic RoCE HCA
  - Support for the Boradcom Cumulus RoCE HCA
  - Enable support for HDR link speeds
- Fix MPI_FINALIZED hang if invoked from an attribute destructor during the MPI_COMM_SELF destruction in MPI_FINALIZE. Thanks to @AndrewGaspar for reporting the issue.
- Java fixes:
  - Modernize Java framework detection, especially on OS X/MacOS. Thanks to Bryce Glover for reporting and submitting the fixes.
  - Prefer "javac -h" to "javah" to support newer Java frameworks.
- Fortran fixes:
  - Use conformant dummy parameter names for Fortran bindings. Thanks to Themos Tsikas for reporting and submitting the fixes.
  - Build the MPI_SIZEOF() interfaces in the "TKR"-style "mpi" module whenever possible. Thanks to Themos Tsikas for reporting the issue.
  - Fix array of argv handling for the Fortran bindings of MPI_COMM SPAWN_MULTIPLE (and its associated man page).
  - Make NAG Fortran compiler support more robust in configure.
  - Disable the "pt2pt" one-sided MPI component when MPI_THREAD_MULTIPLE is used. This component is simply not safe in MPI_THREAD_MULTIPLE scenarios, and will not be fixed in the v2.1.x series.
  - Make the "external" hwloc component fail gracefully if it is tries to use an hwloc v2.x.y installation. hwloc v2.x.y will not be supported in the Open MPI v2.1.x series.
  - Fix "vader" shared memory support for messages larger than 2GB. Thanks to Heiko Bauke for the bug report.
  - Configure fixes for external PMI directory detection. Thanks to Davide Vanzo for the report.

2.1.3 -- March, 2018
-------------------
Bug fixes/minor improvements:
- Update internal PMIx version to 1.2.5.
- Fix a problem with ompi_info reporting using param option. Thanks to Alexander Pozdneev for reporting.
- Correct PMPI_Aint_{add|diff} to be functions (not subroutines) in the Fortran mpi_f08 module.
- Fix a problem when doing MPI I/O using data types with large extents in conjunction with MPI_TYPE_CREATE_SUBARRAY. Thanks to Christopher Brady for reporting.
- Fix a problem when opening many files using MPI_FILE_OPEN. Thanks to William Dawson for reporting.
- Fix a problem with debuggers failing to attach to a running job. Thanks to Dirk Schubert for reporting.
- Fix a problem when using madvise and the OpenIB BTL. Thanks to Timo Bingmann for reporting.
- Fix a problem in the Vader BTL that resulted in failures of IMB under certain circumstances. Thanks to Nicolas Morey-Chaisemartin for reporting.
- Fix a problem preventing Open MPI from working under Cygwin. Thanks to Marco Atzeri for reporting.
- Reduce some verbosity being emitted by the USNIC BTL under certain circumstances. Thanks to Peter Forai for reporting.
- Fix a problem with misdirection of SIGKILL. Thanks to Michael Fern for reporting.
- Replace use of posix_memalign with malloc for small allocations. Thanks to Ben Menaude for reporting.
- Fix a problem with Open MPI's out of band TCP network for file descriptors greater than 32767. Thanks to Wojtek Wasko for reporting and fixing.
- Plug a memory leak in MPI_Mem_free(). Thanks to Philip Blakely for reporting.

2.1.2 -- September, 2017
------------------------

Bug fixes/minor improvements:
- Update internal PMIx version to 1.2.3.
- Fix some problems when using the NAG Fortran compiler to build Open MPI and when using the compiler wrappers. Thanks to Neil Carlson for reporting.
- Fix a compilation problem with the SM BTL. Thanks to Paul Hargrove for reporting.
- Fix a problem with MPI_IALLTOALLW when using zero-length messages. Thanks to Dahai Guo for reporting.
- Fix a problem with C11 generic type interface for SHMEM_G. Thanks to Nick Park for reporting.
- Switch to using the lustreapi.h include file when building Open MPI with Lustre support.
- Fix a problem in the OB1 PML that led to hangs with OSU collective tests.
- Fix a progression issue with MPI_WIN_FLUSH_LOCAL. Thanks to Joseph Schuchart for reporting.
- Fix an issue with recent versions of PBSPro requiring libcrypto. Thanks to Petr Hanousek for reporting.
- Fix a problem when using MPI_ANY_SOURCE with MPI_SENDRECV.
- Fix an issue that prevented signals from being propagated to ORTE daemons.
- Ensure that signals are forwarded from ORTE daemons to all processes in the process group created by the daemons. Thanks to Ted Sussman for reporting.
- Fix a problem with launching a job under a debugger. Thanks to Greg Lee for reporting.
- Fix a problem with Open MPI native I/O MPI_FILE_OPEN when using a communicator having an associated topology. Thanks to Wei-keng Liao for reporting.
- Fix an issue when using MPI_ACCUMULATE with derived datatypes.
- Fix a problem with Fortran bindings that led to compilation errors for user defined reduction operations. Thanks to Nathan Weeks for reporting.
- Fix ROMIO issues with large writes/reads when using NFS file systems.
- Fix definition of Fortran MPI_ARGV_NULL and MPI_ARGVS_NULL.
- Enable use of the head node of a SLURM allocation on Cray XC systems.
- Fix a problem with synchronous Sends when using the UCX PML.
- Use default socket buffer size to improve TCP BTL performance.
- Add a mca parameter ras_base_launch_orted_on_hn to allow for launching MPI processes on the same node where mpirun is executing using a separate orte daemon, rather than the mpirun process. This may be useful to set to true when using SLURM, as it improves interoperability with SLURM's signal propagation tools. By default it is set to false, except for Cray XC systems.
- Fix --without-lsf when lsf is installed in the default search path.
- Remove support for big endian PowerPC.
- Remove support for XL compilers older than v13.1.
- Remove IB XRC support from the OpenIB BTL due to loss of maintainer.

2.1.1 -- April, 2017
-------------------

Bug fixes/minor improvements:

- Fix a problem with one of Open MPI's fifo data structures which led to hangs in a make check test. Thanks to Nicolas Morey-Chaisemartin for reporting.
- Add missing MPI_AINT_ADD/MPI_AINT_DIFF function definitions to mpif.h. Thanks to Aboorva Devarajan for reporting.
- Fix the error return from MPI_WIN_LOCK when rank argument is invalid. Thanks to Jeff Hammond for reporting and fixing this issue.
- Fix a problem with mpirun/orterun when started under a debugger. Thanks to Gregory Leff for reporting.
- Add configury option to disable use of CMA by the vader BTL. Thanks to Sascha Hunold for reporting.
- Add configury check for MPI_DOUBLE_COMPLEX datatype support. Thanks to Alexander Klein for reporting.
- Fix memory allocated by MPI_WIN_ALLOCATE_SHARED to be 64 bit aligned. Thanks to Joseph Schuchart for reporting.
- Update MPI_WTICK man page to reflect possibly higher resolution than 10e-6. Thanks to Mark Dixon for reporting.
- Add missing MPI_T_PVAR_SESSION_NULL definition to mpi.h include file. Thanks to Omri Mor for this contribution.
- Enhance the Open MPI spec file to install modulefile in /opt if installed in a non-default location. Thanks to Kevin Buckley for reporting and supplying a fix.
- Fix a problem with conflicting PMI symbols when linking statically.
Known issues (to be addressed in v2.1.2):
- See the list of fixes slated for v2.1.2 here:
  https://github.com/open-mpi/ompi/milestone/28

2.1.0 -- March, 2017
--------------------

Major new features:
- The main focus of the Open MPI v2.1.0 release was to update to PMIx v1.2.1. When using PMIx (e.g., via mpirun-based launches, or via direct launches with recent versions of popular resource managers), launch time scalability is improved, and the run time memory footprint is greatly decreased when launching large numbers of MPI / OpenSHMEM processes.
- Update OpenSHMEM API conformance to v1.3.
- The usnic BTL now supports MPI_THREAD_MULTIPLE.
- General/overall performance improvements to MPI_THREAD_MULTIPLE.
- Add a summary message at the bottom of configure that tells you many of the configuration options specified and/or discovered by Open MPI.

Changes in behavior compared to prior versions:
- None.

Removed legacy support:
- The ptmalloc2 hooks have been removed from the Open MPI code base. This is not really a user-noticable change; it is only mentioned here because there was much rejoicing in the Open MPI developer community.

Bug fixes/minor improvements:
- New MCA parameters:
  - iof_base_redirect_app_stderr_to_stdout: as its name implies, it combines MPI / OpenSHMEM applications' stderr into its stdout stream.
  - opal_event_include: allow the user to specify which FD selection mechanism is used by the underlying event engine.
  - opal_stacktrace_output: indicate where stacktraces should be sent upon MPI / OpenSHMEM process crashes ("none", "stdout", "stderr", "file:filename").
  - orte_timeout_for_stack_trace: number of seconds to wait for stack traces to be reported (or <=0 to wait forever).
  - mtl_ofi_control_prog_type/ml_ofi_data_prog_type: specify libfabric progress model to be used for control and data.
- Fix MPI_WTICK regression where the time reported may be inaccurate
on systems with processor frequency scaling enabled.
- Fix regression that lowered the memory maximum message bandwidth for
large messages on some BTL network transports, such as openib, sm,
and vader.
- Fix a name collision in the shared file pointer MPI IO file locking
scheme. Thanks to Nicolas Joly for reporting the issue.
- Fix datatype extent/offset errors in MPI_PUT and MPI_RACCUMULATE
when using the Portals 4 one-sided component.
- Add support for non-contiguous datatypes to the Portals 4 one-sided
component.
- Various updates for the UCX PML.
- Updates to the following man pages:
  - mpirun(1)
  - MPI_COMM_CONNECT(3)
  - MPI_WIN_GET_NAME(3). Thanks to Nicolas Joly for reporting the
typo.
  - MPI_INFO_GET_[NKEYS|NTHKEY](3). Thanks to Nicolas Joly for
reporting the typo.
- Fixed a problem in the TCP BTL when using MPI_THREAD_MULTIPLE.
Thanks to Evgueni Petrov for reporting.
- Fixed external32 representation in the romio314 module. Note that
for now, external32 representation is not correctly supported by the
ompio module. Thanks to Thomas Gastine for bringing this to our
attention.
- Add note how to disable a warning message about when a high-speed
MPI transport is not found. Thanks to Susan Schwarz for reporting
the issue.
- Ensure that sending SIGINT when using the rsh/ssh launcher does not
orphan children nodes in the launch tree.
- Fix the help message when showing deprecated MCA param names to show
the correct (i.e., deprecated) name.
- Enable support for the openib BTL to use multiple different
InfiniBand subnets.
- Fix a minor error in MPI_AINT_DIFF.
- Fix bugs with MPI_IN_PLACE handling in:
  - MPI_ALLGATHER[V]
  - MPI_[I][GATHER|SCATTER][V]
  - MPI_IREDUCE[_SCATTER]
  - Thanks to all the users who helped diagnose these issues.
- Allow qrsh to tree spawn (if the back-end system supports it).
- Fix MPI_T_PVAR_GET_INDEX to return the correct index.
- Correctly position the shared file pointer in append mode in the
OMPIO component.
- Add some deprecated names into shmem.h for backwards compatibility
with legacy codes.
- Fix MPI_MODE_NOCHECK support.
- Fix a regression in PowerPC atomics support. Thanks to Orion
Poplawski for reporting the issue.
- Fixes for assembly code with aggressively-optimized compilers on
x86_64/AMD64 platforms.
- Fix one more place where configure was mangling custom CFLAGS.
Thanks to Phil Tooley (@Telemin) for reporting the issue.
- Better handle builds with external installations of hwloc.
- Fixed a hang with MPI_PUT and MPI_WIN_LOCK_ALL.
- Fixed a bug when using MPI_GET on non-contiguous datatypes and
  MPI_LOCK/MPI_UNLOCK.
- Fixed a bug when using POST/START/COMPLETE/WAIT after a fence.
- Fix configure portability by cleaning up a few uses of "==" with
  "test". Thanks to Kevin Buckley for pointing out the issue.
- Fix bug when using darrays with lib and extent of darray datatypes.
- Updates to make Open MPI binary builds more bit-for-bit
  reproducible. Thanks to Alastair McKinstry for the suggestion.
- Fix issues regarding persistent request handling.
- Ensure that shmempx.h is a standalone OpenSHMEM header file. Thanks
  to Nick Park (@nspark) for the report.
- Ensure that we always send SIGTERM prior to SIGKILL. Thanks to Noel
  Rycroft for the report.
- Added ConnectX-5 and Chelsio T6 device defaults for the openib BTL.
- OpenSHMEM no longer supports MMX less than v2.0.
- Plug a memory leak in ompi_osc_sm_free. Thanks to Joseph Schuchart
  for the report.
- The "self" BTL now uses less memory.
- The vader BTL is now more efficient in terms of memory usage when
  using XPMEM.
- Removed the --enable-openib-failover configure option. This is not
  considered backwards-incompatible because this option was stale and
  had long-since stopped working, anyway.
- Allow jobs launched under Cray aprun to use hyperthreads if
  opal_hwloc_base_hwthreads_as_cpus MCA parameter is set.
- Add support for 32-bit and floating point Cray Aries atomic
  operations.
- Add support for network AMOs for MPI_ACCUMULATE, MPI_FETCH_AND_OP,
  and MPI_COMPARE_AND_SWAP if the "ompi_single_intrinsic" info key is
  set on the window or the "acc_single_intrinsic" MCA param is set.
- Automatically disqualify RDMA CM support in the openib BTL if
  MPI_THREAD_MULTIPLE is used.
- Make configure smarter/better about auto-detecting Linux CMA
  support.
- Improve the scalability of MPI_COMM_SPLIT_TYPE.
- Fix the mixing of C99 and C++ header files with the MPI C++
  bindings. Thanks to Alastair McKinstry for the bug report.
- Add support for ARM v8.
- Several MCA parameters now directly support MPI_T enumerator
  semantics (i.e., they accept a limited set of values -- e.g., MCA
  parameters that accept boolean values).
- Added --with-libmpi-name=STRING configure option for vendor releases
  of Open MPI. See the README for more detail.
- Fix a problem with Open MPI's internal memory checker. Thanks to Yvan
  Fournier for reporting.
- Fix a multi-threaded issue with MPI_WAIT. Thanks to Pascal Deveze for
  reporting.

Known issues (to be addressed in v2.1.1):
- See the list of fixes slated for v2.1.1 here:
  https://github.com/open-mpi/ompi/milestone/26

2.0.4 -- November, 2017
--------------------

Bug fixes/minor improvements:
- Fix an issue with visibility of functions defined in the built-in PMIx.
  Thanks to Siegmar Gross for reporting this issue.
- Add configure check to prevent trying to build this release of
  Open MPI with an external hwloc 2.0 or newer release.
- Add ability to specify layered providers for OFI MTL.
- Fix a correctness issue with Open MPI's memory manager code
  that could result in corrupted message data. Thanks to
  Valentin Petrov for reporting.
- Fix issues encountered when using newer versions of PBS Pro.
  Thanks to Petr Hanousek for reporting.
- Fix a problem with MPI_GET when using the vader BTL. Thanks
  to Dahai Guo for reporting.
- Fix a problem when using MPI_ANY_SOURCE with MPI_SENDRECV_REPLACE.
  Thanks to Dahai Guo for reporting.
- Fix a problem using MPI_FILE_OPEN with a communicator with an
  attached cartesian topology. Thanks to Wei-keng Liao for reporting.
- Remove IB XRC support from the OpenIB BTL due to lack of support.
- Remove support for big endian PowerPC.
- Remove support for XL compilers older than v13.1

2.0.3 -- June 2017
------------------

Bug fixes/minor improvements:
- Fix a problem with MPI_IALLTOALLW when zero size messages are present.
  Thanks to @mathbird for reporting.
- Add missing MPI_USER_FUNCTION definition to the mpi_f08 module.
  Thanks to Nathan Weeks for reporting this issue.
- Fix a problem with MPI_WIN_LOCK not returning an error code when
  a negative rank is supplied. Thanks to Jeff Hammond for reporting and
  providing a fix.
- Fix a problem with make check that could lead to hangs. Thanks to
  Nicolas Morey-Chaisemartin for reporting.
- Resolve a symbol conflict problem with PMI-1 and PMI-2 PMIx components.
  Thanks to Kilian Cavalotti for reporting this issue.
- Insure that memory allocations returned from MPI_WIN_ALLOCATE_SHARED are
  64 byte aligned. Thanks to Joseph Schuchart for reporting this issue.
- Make use of DOUBLE_COMPLEX, if available, for Fortran bindings. Thanks
  to Alexander Klein for reporting this issue.
- Add missing MPI_T_PVAR_SESSION_NULL definition to Open MPI mpi.h include
  file. Thanks to Omri Mor for reporting and fixing.
- Fix a problem with use of MPI shared file pointers when accessing
  a file from independent jobs. Thanks to Nicolas Joly for reporting
  this issue.
- Optimize zero size MPI_IALLTOALL{V,W} with MPI_IN_PLACE. Thanks to Lisandro Dalcin for the report.
- Fix a ROMIO buffer overflow problem for large transfers when using NFS filesystems.
- Fix type of MPI_ARGV[S]_NULL which prevented it from being used properly with MPI_COMM_SPAWN[_MULTIPLE] in the mpi_f08 module.
- Ensure to add proper linker flags to the wrapper compilers for dynamic libraries on platforms that need it (e.g., RHEL 7.3 and later).
- Get better performance on TCP-based networks 10Gbps and higher by using OS defaults for buffer sizing.
- Fix a bug with MPI_[R][GET_]ACCUMULATE when using DARRAY datatypes.
- Fix handling of --with-lustre configure command line argument. Thanks to Prentice Bisbal and Tim Mattox for reporting the issue.
- Added MPI_AINT_ADD and MPI_AINT_DIFF declarations to mpif.h. Thanks to Aboorva Devarajan (@AboorvaDevarajan) for the bug report.
- Fix a problem in the TCP BTL when Open MPI is initialized with MPI_THREAD_MULTIPLE support. Thanks to Evgueni Petro for analyzing and reporting this issue.
- Fix yalla PML to properly handle underflow errors, and fixed a memory leak with blocking non-contiguous sends.
- Restored ability to run autogen.pl on official distribution tarballs (although this is still not recommended for most users!).
- Fix accuracy problems with MPI_WTIME on some systems by always using either clock_gettime(3) or gettimeofday(3).
- Fix a problem where MPI_WTICK was not returning a higher time resolution when available. Thanks to Mark Dixon for reporting this issue.
- Restore SGE functionality. Thanks to Kevin Buckley for the initial report.
- Fix external hwloc compilation issues, and extend support to allow using external hwloc installations as far back as v1.5.0. Thanks to Orion Poplawski for raising the issue.
- Added latest Mellanox Connect-X and Chelsio T-6 adapter part IDs to the openib list of default values.
- Do a better job of cleaning up session directories (e.g., in /tmp).
- Update a help message to indicate how to suppress a warning about no high performance networks being detected by Open MPI. Thanks to Susan Schwarz for reporting this issue.
- Fix a problem with mangling of custom CFLAGS when configuring Open MPI. Thanks to Phil Tooley for reporting.
- Fix some minor memory leaks and remove some unused variables. Thanks to Joshua Gerrard for reporting.
- Fix MPI_ALLGATHERV bug with MPI_IN_PLACE.

Known issues (to be addressed in v2.0.4):

- See the list of fixes slated for v2.0.4 here:
  https://github.com/open-mpi/ompi/milestone/29

2.0.2 -- 26 January 2017

---------------------------------
Bug fixes/minor improvements:

- Fix a problem with MPI_FILE_WRITE_SHARED when using MPI MODE_APPEND and Open MPI's native MPI-I0 implementation. Thanks to Nicolas Joly for reporting.
- Fix a typo in the MPI_WIN_GET_NAME man page. Thanks to Nicolas Joly for reporting.
- Fix a race condition with ORTE's session directory setup. Thanks to @tbj900 for reporting this issue.
- Fix a deadlock issue arising from Open MPI's approach to catching calls to munmap. Thanks to Paul Hargrove for reporting and helping to analyze this problem.
- Fix a problem with PPC atomics which caused make check to fail unless built-in atomics configure option was enabled. Thanks to Orion Poplawski for reporting.
- Fix a problem with use of x86_64 cpuid instruction which led to segmentation faults when Open MPI was configured with -O3 optimization. Thanks to Mark Santcroos for reporting this problem.
- Fix a problem when using built-in atomics configure options on PPC platforms when building 32 bit applications. Thanks to Paul Hargrove for reporting.
- Fix a problem with building Open MPI against an external hwloc installation. Thanks to Orion Poplawski for reporting this issue.
- Remove use of DATE in the message queue version string reported to debuggers to insure bit-wise reproducibility of binaries. Thanks to Alastair McKinstry for help in fixing this problem.
- Fix a problem with early exit of a MPI process without calling MPI_FINALIZE or MPI_ABORT that could lead to job hangs. Thanks to Christof Koehler for reporting.
- Fix a problem with forwarding of SIGTERM signal from mpirun to MPI processes in a job. Thanks to Noel Rycroft for reporting this problem.
- Plug some memory leaks in MPI_WIN_FREE discovered using Valgrind. Thanks to Joseph Schuchart for reporting.
- Fix a problems MPI_NEIGHBOR_ALLTOALL when using a communicator with an empty topology graph. Thanks to Daniel Ibanez for reporting.
- Fix a typo in a PMIx component help file. Thanks to @njoly for reporting this.
- Fix a problem with Valgrind false positives when using Open MPI's internal memchecker. Thanks to Yvan Fournier for reporting.
- Fix a problem with MPI_FILE_DELETE returning MPI_SUCCESS when deleting a non-existent file. Thanks to Wei-keng Liao for reporting.
- Fix a problem with MPI_IMPROBE that could lead to hangs in subsequent MPI point to point or collective calls. Thanks to Chris Pattison for reporting.
- Fix a problem when configure Open MPI for powerpc with --enable-mpi-cxx enabled. Thanks to Alastair McKinstry for reporting.
- Fix a problem using MPI_IALLTOALL with MPI_IN_PLACE argument. Thanks to Chris Ward for reporting.
- Fix a problem using MPI_RACUMULATE with the Portals4 transport. Thanks to @PDeveze for reporting.
- Fix an issue with static linking and duplicate symbols arising from PMIx Slurm components. Thanks to Limin Gu for reporting.
- Fix a problem when using MPI dynamic memory windows. Thanks to Christoph Niethammer for reporting.
- Fix a problem with Open MPI's pkgconfig files. Thanks to Alastair McKinstry for reporting.
- Fix a problem with MPI_IREDUCE when the same buffer is supplied for the send and recv buffer arguments. Thanks to Valentin Petrov for reporting.
- Fix a problem with atomic operations on PowerPC. Thanks to Paul Hargrove for reporting.

Known issues (to be addressed in v2.0.3):

- See the list of fixes slated for v2.0.3 here: https://github.com/open-mpi/ompi/milestone/23

2.0.1 -- 2 September 2016
-------------------------

Bug fixes/minor improvements:

- Short message latency and message rate performance improvements for all transports.
- Fix shared memory performance when using RDMA-capable networks. Thanks to Tetsuya Mishima and Christoph Niethammer for reporting.
- Fix bandwith performance degredation in the yalla (MXM) PML. Thanks to Andreas Kempf for reporting the issue.
- Fix OpenSHMEM crash when running on non-Mellanox MXM-based networks. Thanks to Debendra Das for reporting the issue.
- Fix a crash occurring after repeated calls to MPI_FILE_SET_VIEW with predefined datatypes. Thanks to Eric Chamberland and Matthew Knepley for reporting and helping chase down this issue.
- Fix stdin propagation to MPI processes. Thanks to Jingchao Zhang for reporting the issue.
- Fix various runtime and portability issues by updating the PMIx internal component to v1.1.5.
- Fix process startup failures on Intel MIC platforms due to very large entries in /proc/mounts.
- Fix a problem with use of relative path for specifying executables to mpirun/oshrun. Thanks to David Schneider for reporting.
- Various improvements when running over portals-based networks.
- Fix thread-based race conditions with GNI-based networks.
- Fix a problem with MPI_FILE_CLOSE and MPI_FILE_SET_SIZE. Thanks to Cihan Altinay for reporting.
- Remove all use of rand(3) from within Open MPI so as not to perturb applications use of it. Thanks to Matias Cabral and Noel Rycroft for reporting.
- Fix crash in MPI_COMM_SPAWN.
- Fix types for MPI_UNWEIGHTED and MPI_WEIGHTS_EMPTY. Thanks to Lisandro Dalcin for reporting.
- Correctly report the name of MPI_INTEGER16.
- Add some missing MPI constants to the Fortran bindings.
- Fixed compile error when configuring Open MPI with --enable-timing.
- Correctly set the shared library version of libompitrace.so. Thanks to Alastair McKinstry for reporting.
- Fix errors in the MPI_RPUT, MPI_RGET, MPI_RACCUMULATE, and MPI_RGET_ACCUMULATE Fortran bindings. Thanks to Alfio Lazzaro and Joost VandeVondele for tracking this down.
- Fix problems with use of derived datatypes in non-blocking collectives. Thanks to Yuki Matsumoto for reporting.
- Fix problems with OpenSHMEM header files when using CMake. Thanks to Paul Kapinos for reporting the issue.
- Fix problem with use of non-zero lower bound datatypes in collectives. Thanks to Hristo Iliev for reporting.
- Fix a problem with memory allocation within MPI_GROUP_INTERSECTION. Thanks to Lisandro Dalcin for reporting.
- Fix an issue with MPI_ALLGATHER for communicators that don't consist of two ranks. Thanks to David Love for reporting.
- Various fixes for collectives when used with esoteric MPI datatypes.
- Fixed corner cases of handling DARRAY and HINDEXED_BLOCK datatypes.
- Fix a problem with filesystem type check for OpenBSD. Thanks to Paul Hargrove for reporting.
- Fix some debug input within Open MPI internal functions. Thanks to Durga Choudhury for reporting.
- Fix a typo in a configury help message. Thanks to Paul Hargrove for reporting.
- Correctly support MPI_IN_PLACE in MPI_[I]ALLTOALL[|V|W] and MPI_[I]EXSCAN.
- Fix alignment issues on SPARC platforms.

Known issues (to be addressed in v2.0.2):

- See the list of fixes slated for v2.0.2 here:
  https://github.com/open-mpi/ompi/milestone/20, and
  https://github.com/open-mpi/ompi-release/milestone/19
  (note that the "ompi-release" Github repo will be folded/absorbed into the "ompi" Github repo at some point in the future)

2.0.0 -- 12 July 2016
---------------------

**********************************************************************
* Open MPI is now fully MPI-3.1 compliant                           *
**********************************************************************

Major new features:

- Many enhancements to MPI RMA. Open MPI now maps MPI RMA operations on to native RMA operations for those networks which support this capability.
- Greatly improved support for MPI_THREAD_MULTIPLE (when configured with --enable-mpi-thread-multiple).
- Enhancements to reduce the memory footprint for jobs at scale. A new MCA parameter, "mpi_add_procs_cutoff", is available to set the threshold for using this feature.
- Completely revamped support for memory registration hooks when using OS-bypass network transports.
- Significant OMPIO performance improvements and many bug fixes.
- Add support for PMIx - Process Management Interface for Exascale. Version 1.1.2 of PMIx is included internally in this release.
- Add support for PLFS file systems in Open MPI I/O.
- Add support for UCX transport.
- Simplify build process for Cray XC systems. Add support for using native SLURM.
- Add a --tune mpiexec command line option to simplify setting many environment variables and MCA parameters.
- Add a new MCA parameter "orte_default_dash_host" to offer an analogue to the existing "orte_default_hostfile" MCA parameter.
- Add the ability to specify the number of desired slots in the mpiexec --host option.

Changes in behavior compared to prior versions:

- In environments where mpiexec cannot automatically determine the number of slots available (e.g., when using a hostfile that does not specify "slots", or when using --host without specifying a ":N" suffix to hostnames), mpiexec now requires the use of "-np N" to specify how many MPI processes to launch.
- The MPI C++ bindings -- which were removed from the MPI standard in v3.0 -- are no longer built by default and will be removed in some future version of Open MPI. Use the --enable-mpi-cxx-bindings configure option to build the deprecated/removed MPI C++ bindings.
- ompi_info now shows all components, even if they do not have MCA parameters. The prettyprint output now separates groups with a dashed line.
- OMPIO is now the default implementation of parallel I/O, with the exception for Lustre parallel filesystems (where ROMIO is still the default). The default selection of OMPI vs. ROMIO can be controlled via the "--mca io ompi|romio" command line switch to mpiexec.
- Per Open MPI's versioning scheme (see the README), increasing the major version number to 2 indicates that this version is not ABI-compatible with prior versions of Open MPI. You will need to recompile MPI and OpenSHMEM applications to work with this version of Open MPI.
- Removed checkpoint/restart code due to loss of maintainer. :-(
- Change the behavior for handling certain signals when using PSM and PSM2 libraries. Previously, the PSM and PSM2 libraries would trap certain signals in order to generate tracebacks. The mechanism was found to cause issues with Open MPI's own error reporting mechanism. If not already set, Open MPI now sets the IPATH_NO_BACKTRACE and HFI_NO_BACKTRACE environment variables to disable PSM/PSM2's handling these signals.

Removed legacy support:

- Removed support for OS X Leopard.
- Removed support for Cray XT systems.
- Removed VampirTrace.
- Removed support for Myrinet/MX.
- Removed legacy collective module:ML.
- Removed support for Alpha processors.
- Removed --enable-mpi-profiling configure option.
Known issues (to be addressed in v2.0.1):

- See the list of fixes slated for v2.0.1 here:
  https://github.com/open-mpi/ompi/milestone/16, and
  https://github.com/open-mpi/ompi-release/milestone/16
  (note that the "ompi-release" Github repo will be folded/absorbed
  into the "ompi" Github repo at some point in the future)

- **ompi-release#986**: Fix data size counter for large ops with fcoll/static
- **ompi-release#987**: Fix OMPIO performance on Lustre
- **ompi-release#1013**: Fix potential inconsistency in btl/openib default settings
- **ompi-release#1014**: Do not return MPI_ERR_PENDING from collectives
- **ompi-release#1056**: Remove dead profile code from oshmem
- **ompi-release#1081**: Fix MPI_IN_PLACE checking for IALLTOALL{V|W}
- **ompi-release#1081**: Fix memchecker in MPI_IALLTOALLW
- **ompi-release#1081**: Support MPI_IN_PLACE in MPI_(I)ALLTOALL and MPI_(I)EXSCAN
- **ompi-release#1107**: Allow future PMIx support for RM spawn limits
- **ompi-release#1108**: Fix sparse group process reference counting
- **ompi-release#1109**: If specified to be oversubscribed, disable binding
- **ompi-release#1122**: Allow NULL arrays for empty datatypes
- **ompi-release#1123**: Make max hostname length uniform across code base
- **ompi-release#1127**: Fix MPI_Compare_and_swap
- **ompi-release#1127**: Fix MPI_Win_lock when used with MPI_Win_fence
- **ompi-release#1132**: Fix typo in help message for --enable-mca-no-build
- **ompi-release#1154**: Ensure pairwise coll algorithms disqualify themselves properly
- **ompi-release#1165**: Fix typos in debugging/verbose message output
- **ompi-release#1178**: Fix ROMIO filesystem check on OpenBSD 5.7
- **ompi-release#1197**: Fix Fortran pthread configure check
- **ompi-release#1205**: Allow using external PMIx 1.1.4 and 2.0
- **ompi-release#1215**: Fix configure to support the NAG Fortran compiler
- **ompi-release#1220**: Fix combiner args for MPI_HINDEXED_BLOCK
- **ompi-release#1225**: Fix combiner args for MPI_DARRAY
- **ompi-release#1226**: Disable old memory hooks with recent gcc versions
- **ompi-release#1231**: Fix new "patcher" support for some XLC platforms
- **ompi-release#1244**: Fix Java error handling
- **ompi-release#1250**: Ensure TCP is not selected for RDMA operations
- **ompi-release#1252**: Fix verbose output in coll selection
- **ompi-release#1253**: Set a default name for user-defined MPI_Op
- **ompi-release#1254**: Add count==0 checks in some non-blocking colls
- **ompi-release#1258**: Fix "make distclean" when using external pmix/hwloc/libevent
- **ompi-release#1260**: Clean up/uniform mca/coll/base memory management
- **ompi-release#1261**: Remove "patcher" warning message for static builds
- **ompi-release#1263**: Fix IO MPI_Request for 0-size read/write
- **ompi-release#1264**: Add blocking fence for SLURM operations

Bug fixes / minor enhancements:

- Updated internal/embedded copies of third-party software:
  - Update the internal copy of ROMIO to that which shipped in MPICH
    3.1.4.
- Update internal copy of libevent to v2.0.22.
- Update internal copy of hwloc to v1.11.2.
- Notable new MCA parameters:
  - opal_progress_lp_call_ration: Control how often low-priority callbacks are made during Open MPI's main progress loop.
  - opal_common_verbs_want_fork_support: This replaces the btl_openib_want_fork_support parameter.
- Add --with-platform-patches-dir configure option.
- Add --with-pmi-libdir configure option for environments that install PMI libs in a non-default location.
- Various configure-related compatibility updates for newer versions of libibverbs and OFED.
- Numerous fixes/improvements to orte-dvm. Special thanks to Mark Santcroos for his help.
- Fix a problem with timer code on ia32 platforms. Thanks to Paul Hargrove for reporting this and providing a patch.
- Fix a problem with use of a 64 bit atomic counter. Thanks to Paul Hargrove for reporting.
- Fix a problem with singleton job launching. Thanks to Lisandro Dalcin for reporting.
- Fix a problem with use of MPI_UNDEFINED with MPI_COMM_SPLIT_TYPE. Thanks to Lisandro Dalcin for reporting.
- Silence a compiler warning in PSM MTL. Thanks to Adrian Reber for reporting this.
- Properly detect Intel TrueScale and OmniPath devices in the ACTIVE state. Thanks to Durga Choudhury for reporting the issue.
- Fix detection and use of Solaris Studio 12.5 (beta) compilers. Thanks to Paul Hargrove for reporting and debugging.
- Fix various small memory leaks.
- Allow NULL arrays when creating empty MPI datatypes.
- Replace use of alloca with malloc for certain datatype creation functions. Thanks to Bogdan Sataric for reporting this.
- Fix use of MPI_LB and MPI_UB in creation of of certain MPI datatypes. Thanks to Gus Correa for helping to fix this.
- Implement a workaround for a GNU Libtool problem. Thanks to Eric Schnetter for reporting and fixing.
- Improve hcoll library detection in configure. Thanks to David Shrader and Ake Sandgren for reporting this.
- Miscellaneous minor bug fixes in the hcoll component.
- Miscellaneous minor bug fixes in the ugni component.
- Fix problems with XRC detection in OFED 3.12 and older releases. Thanks to Paul Hargrove for his analysis of this problem.
- Update (non-standard/experimental) Java MPI interfaces to support MPI-3.1 functionality.
- Fix an issue with MCA parameters for Java bindings. Thanks to Takahiro Kawashima and Siegmar Gross for reporting this issue.
- Fix a problem when using persistent requests in the Java bindings. Thanks to Nate Chambers for reporting.
- Fix problem with Java bindings on OS X 10.11. Thanks to Alexander Daryin for reporting this issue.
- Fix a performance problem for large messages for Cray XC systems. Thanks to Jerome Vienne for reporting this.
- Fix an issue with MPI_WIN_LOCK_ALL. Thanks to Thomas Jahns for reporting.
- Fix an issue with passing a parameter to configure multiple times. Thanks to QuesarVII for reporting and supplying a fix.
- Add support for ALPS resource allocation system on Cray CLE 5.2 and later. Thanks to Mark Santcroos.
- Corrections to the HACKING file. Thanks to Maximilien Levesque.
- Fix an issue with user supplied reduction operator functions. Thanks to Rupert Nash for reporting this.
- Fix an issue with an internal list management function. Thanks to Adrian Reber for reporting this.
- Fix a problem with MPI-RMA PSCW epochs. Thanks to Berk Hess for reporting this.
- Fix a problem in neighborhood collectives. Thanks to Lisandro Dalcin for reporting.
- Fix MPI_IREDUCE_SCATTER_BLOCK for a one-process communicator. Thanks to Lisandro Dalcin for reporting.
- Add (Open MPI-specific) additional flavors to MPI_COMM_SPLIT_TYPE. See MPI_Comm_split_type(3) for details. Thanks to Nick Andersen for supplying this enhancement.
- Improve closing of file descriptors during the job launch phase. Thanks to Piotr Lesnicki for reporting and providing this enhancement.
- Fix a problem in MPI_GET_ACCUMULATE and MPI_RGET_ACCUMULATE when using Portals4. Thanks to Nicolas Chevalier for reporting.
- Use correct include file for lstat prototype in ROMIO. Thanks to William Throwe for finding and providing a fix.
- Add missing Fortran bindings for MPI_WIN_ALLOCATE. Thanks to Christoph Niethammer for reporting and fixing.
- Fortran related fixes to handle Intel 2016 compiler. Thanks to Fabrice Roy for reporting this.
- Fix a Fortran linkage issue. Thanks to Macro Atzeri for finding and suggesting a fix.
- Fix problem with using BIND(C) for Fortran bindings with logical parameters. Thanks to Paul Romano for reporting.
- Fix an issue with use of DL-related macros in opal library. Thanks to Scott Atchley for finding this.
- Fix an issue with parsing mpirun command line options which contain colons. Thanks to Lev Given for reporting.
- Fix a problem with Open MPI's package configury files. Thanks to Christoph Junghans for reporting.
- Fix a typo in the MPI_INTERCOMM_MERGE man page. Thanks To Harald Servat for reporting and correcting.
- Update man pages for non-blocking sends per MPI 3.1 standard. Thanks to Alexander Pozdneev for reporting.
- Fix problem when compiling against PVFS2. Thanks to Dave Love for reporting.
- Fix problems with MPI_NEIGHBOR_ALLTOALL{V,W}. Thanks to Willem Vermin for reporting this issue.
- Fix various compilation problems on Cygwin. Thanks to Marco Atzeri for supplying these fixes.
- Fix problem with resizing of subarray and darray data types. Thanks
to Keith Bennett and Dan Garmann for reporting.
- Fix a problem with MPI_COMBINER_RESIZED. Thanks to James Ramsey for
  the report.
- Fix an hwloc binding issue. Thanks to Ben Menadue for reporting.
- Fix a problem with the shared memory (sm) BTL. Thanks to Peter Wind
  for the report.
- Fixes for heterogeneous support. Thanks to Siegmar Gross for reporting.
- Fix a problem with memchecker. Thanks to Clinton Simpson for reporting.
- Fix a problem with MPI_UNWEIGHTED in topology functions. Thanks to
  Jun Kudo for reporting.
- Fix problem with a MCA parameter base filesystem types. Thanks to
  Siegmar Gross for reporting.
- Fix a problem with some windows info argument types. Thanks to
  Alastair McKinstry for reporting.

1.10.7 -- 16 May 2017
---------------------
- Fix bug in TCP BTL that impacted performance on 10GbE (and faster)
  networks by not adjusting the TCP send/recv buffer sizes and using
  system default values
- Add missing MPI_AINT_ADD and MPI_AINT_DIFF function declarations in
  mpif.h
- Fixed time reported by MPI_WTIME; it was previously reported as
  dependent upon the CPU frequency.
- Fix platform detection on FreeBSD
- Fix a bug in the handling of MPI_TYPE_CREATE_DARRAY in
  MPI_(R)_(GET_)ACCUMULATE
- Fix openib memory registration limit calculation
- Add missing MPI_T_PVAR_SESSION_NULL in mpi.h
- Fix "make distcheck" when using external hwloc and/or libevent packages
- Add latest ConnectX-5 vendor part id to OpenIB device params
- Fix race condition in the UCX PML
- Fix signal handling for rsh launcher
- Fix Fortran compilation errors by removing MPI_SIZEOF in the Fortran
  interfaces when the compiler does not support it
- Fixes for the pre-ignore-TKR "mpi" Fortran module implementation
  (i.e., for older Fortran compilers -- these problems did not exist
  in the "mpi" module implementation for modern Fortran compilers):
  - Add PMPI_* interfaces
  - Fix typo in MPI_FILE_WRITE_AT_ALL_BEGIN interface name
  - Fix typo in MPI_FILE_READ_ORDERED_BEGIN interface name
  - Fixed the type of MPI_DISPLACEMENT_CURRENT in all Fortran interfaces
    to be an INTEGER(KIND=MPI_OFFSET_KIND).
  - Fixed typos in MPI_INFO_GET_* man pages. Thanks to Nicolas Joly for
    the patch
  - Fix typo bugs in wrapper compiler script

1.10.6 -- 17 Feb 2017
---------------------
- Fix bug in timer code that caused problems at optimization settings

(continues on next page)
greater than 2
- OSHMEM: make mmap allocator the default instead of sysv or verbs
- Support MPI_Dims_create with dimension zero
- Update USNIC support
- Prevent 64-bit overflow on timer counter
- Add support for forwarding signals
- Fix bug that caused truncated messages on large sends over TCP BTL
- Fix potential infinite loop when printing a stacktrace

1.10.5 -- 19 Dec 2016
---------------------
- Update UCX APIs
- Fix bug in darray that caused MPI/I0 failures
- Use a MPI_Get_library_version() like string to tag the debugger DLL.
  Thanks to Alastair McKinstry for the report
- Fix multi-threaded race condition in coll/libnbc
- Several fixes to OSHMEM
- Fix bug in UCX support due to uninitialized field
- Fix MPI_Ialltoallv with MPI_IN_PLACE and without MPI param check
- Correctly reset receive request type before init. Thanks Chris Pattison
  for the report and test case.
- Fix bug in iallgather[v]
- Fix concurrency issue with MPI_Comm_accept. Thanks to Pieter Noordhuis
  for the patch
- Fix omni_coll_base_{gather,scatter}_intra_binomial
- Fixed an issue with MPI_Type_get_extent returning the wrong extent
  for distributed array datatypes.
- Re-enable use of rdtsc instruction as a monotonic clock source if
  the processor has a core-invariant tsc. This is a partial fix for a
  performance regression introduced in Open MPI v1.10.3.

1.10.4 -- 01 Sept 2016
---------------------
- Fix assembler support for MIPS
- Improve memory handling for temp buffers in collectives
- Fix [all]reduce with non-zero lower bound datatypes
  Thanks Hristo Iliev for the report
- Fix non-standard ddt handling. Thanks Yuki Matsumoto for the report
- Various libnbc fixes. Thanks Yuki Matsumoto for the report
- Fix typos in request RMA bindings for Fortran. Thanks to @alazzaro
  and @vondele for the assist
- Various bug fixes and enhancements to collective support
- Fix predefined types mapping in hcoll
- Revive the coll/sync component to resolve unexpected message issues
  during tight loops across collectives
- Fix typo in wrapper compiler for Fortran static builds

1.10.3 -- 15 June 2016
(continues on next page)
- Fix zero-length datatypes. Thanks to Wei-keng Liao for reporting the issue.
- Minor manpage cleanups
- Implement atomic support in OSHMEM/UCX
- Fix support of MPI_COMBINER_RESIZED. Thanks to James Ramsey for the report
- Fix computation of #cpus when --use-hwthread-cpus is used
- Add entry points for Allgatherv, iAllgatherv, Reduce, and iReduce for the HCOLL library
- Fix an HCOLL integration bug that could signal completion of request while still being worked
- Fix computation of cores when SMT is enabled. Thanks to Ben Menadue for the report
- Various USNIC fixes
- Create a datafile in the per-proc directory in order to make it unique per communicator. Thanks to Peter Wind for the report
- Fix zero-size malloc in one-sided pt-to-pt code. Thanks to Lisandro Dalcin for the report
- Fix MPI_Get_address when passed MPI_BOTTOM to not return an error. Thanks to Lisandro Dalcin for the report
- Fix MPI_TYPE_SET_ATTR with NULL value. Thanks to Lisandro Dalcin for the report
- Fix various Fortran08 binding issues
- Fix memchecker no-data case. Thanks to Clinton Stimpson for the report
- Fix CUDA support under OS-X
- Fix various OFI/MTL integration issues
- Add MPI_T man pages
- Fix one-sided pt-to-pt issue by preventing communication from happening before a target enters a fence, even in the no-precede case
- Fix a bug that disabled Totalview for MPMD use-case
- Correctly support MPI_UNWEIGHTED in topo-graph-neighbors. Thanks to Jun Kudo for the report
- Fix singleton operations under SLURM when PMI2 is enabled
- Do not use MPI_IN_PLACE in neighborhood collectives for non-blocking collectives (libnbc). Thanks to Jun Kudo for the report
- Silence autogen deprecation warnings for newer versions of Perl
- Do not return MPI_ERR_PENDING from collectives
- Use type int* for MPI_WIN_DISP_UNIT, MPI_WIN_CREATE_FLAVOR, and MPI_WIN_MODEL. Thanks to Alastair McKinstry for the report
- Fix register_datarep stub function in IO/OMPIO. Thanks to Eric Chamberland for the report
- Fix a bus error on MPI_WIN_[POST,START] in the shared memory one-sided component
- Add several missing MPI_WIN_FLAVOR constants to the Fortran support
- Enable connecting processes from different subnets using the openib BTL
- Fix bug in basic/barrier algorithm in OSHMEM
- Correct process binding for the --map-by node case
- Include support for subnet-to-subnet routing over InfiniBand networks
- Fix usnic resource check
- AUTHORS: Fix an errant reference to Subversion IDs
- Fix affinity for MPMD jobs running under LSF
- Fix many Fortran binding bugs
- Fix `MPI_IN_PLACE`-related bugs
- Fix PSM/PSM2 support for singleton operations
- Ensure MPI transports continue to progress during RTE barriers
- Update HWLOC to 1.9.1 end-of-series
- Fix a bug in the Java command line parser when the
  `-Djava.library.path` options was given by the user
- Update the MTL/OFI provider selection behavior
- Add support for clock_gettime on Linux.
- Correctly detect and configure for Solaris Studio 12.5 beta compilers
- Correctly compute #slots when `-host` is used for MPMD case
- Fix a bug in the hcoll collectives due to an uninitialized field
- Do not set a binding policy when oversubscribing a node
- Fix hang in intercommunicator operations when oversubscribed
- Speed up process termination during MPI_Abort
-Disable backtrace support by default in the PSM/PSM2 libraries to
  prevent unintentional conflicting behavior.

1.10.2 -- 26 Jan 2016
---------------------

**************************************************************************************
* OSHMEM is now 1.2 compliant
**************************************************************************************

- Fix NBC_Copy for legitimate zero-size messages
- Fix multiple bugs in OSHMEM
- Correctly handle mpirun --host <user>@<ip-address>
- Centralize two MCA params to avoid duplication between OMPI and
  OSHMEM layers: opal_abort_delay and opal_abort_print_stack
- Add support for Fujitsu compilers
- Add UCX support for OMPI and OSHMEM
- Correctly handle oversubscription when not given directives
  to permit it. Thanks to @ammore1 for reporting it
- Fix rpm spec file to not include the `/usr` directory
- Add Intel HFI1 default parameters for the openib BTL
- Resolve symbol conflicts in the PSM2 library
- Add ability to empty the rgpusm cache when full if requested
- Fix another libtool bug when `-L` requires a space between it
  and the path. Thanks to Eric Schnetter for the patch.
- Add support for OSHMEM v1.2 APIs
- Improve efficiency of oshmem_preconnect_all algorithm
- Fix bug in buffered sends support
- Fix double free in edge case of mpirun. Thanks to @jsharpe for
  the patch
- Multiple one-sided support fixes
- Fix integer overflow in the tuned "reduce" collective when
  using buffers larger than INT_MAX in size
- Fix parse of user environment variables in mpirun. Thanks to
  Stefano Garzarella for the patch

(continues on next page)
- Performance improvements in PSM2 support
- Fix NBS iBarrier for inter-communicators
- Fix bug in vader BTL during finalize
- Improved configure support for Fortran compilers
- Fix rank_file mapper to support default --slot-set. Thanks to Matt Thompson for reporting it
- Update MPI_Testsome man page. Thanks to Eric Schnetter for the suggestion
- Fix missing resize of the returned type for subarray and darray types. Thanks to Keith Bennett and Dan Garmann for reporting it
- Fix Java support on OSX 10.11. Thanks to Alexander Daryin for reporting the problem
- Fix some compilation issues on Solaris 11.2. Thanks to Paul Hargrove for his continued help in such areas

1.10.1 -- 4 Nov 2015
-------------------
- Workaround an optimization problem with gcc compilers >= 4.9.2 that causes problems with memory registration, and forced mpi_leave_pinned to default to 0 (i.e., off). Thanks to @oere for the fix.
- Fix use of MPI_LB and MPI_UB in subarray and darray datatypes. Thanks to Gus Correa and Dimitar Pashov for pointing out the issue.
- Minor updates to mpi_show_mpi_alloc_mem_leaks and ompi_debug_show_handle_leaks functionality.
- Fix segv when invoking non-blocking reductions with a user-defined operation. Thanks to Rupert Nash and Georg Geiser for identifying the issue.
- No longer probe for PCI topology on Solaris (unless running as root).
- Fix for Intel Parallel Studio 2016 ifort partial support of the !GCCS pragma. Thanks to Fabrice Roy for reporting the problem.
- Bunches of Coverity / static analysis fixes.
- Fixed ROMIO to look for lstat in <sys/stat.h>. Thanks to William Throwe for submitting the patch both upstream and to Open MPI.
- Fixed minor memory leak when attempting to open plugins.
- Fixed type in MPI_IBARRIER C prototype. Thanks to Harald Servat for reporting the issue.
- Add missing man pages for MPI_WIN_CREATE_DYNAMIC, MPI_WIN_ATTACH, MPI_WIN_DETACH, MPI_WIN_ALLOCATE, MPI_WIN_ALLOCATE_SHARED.
- When mpirun-launched new applications, only close file descriptors that are actually open (resulting in a faster launch in some environments).
- Fix "test ==" issues in Open MPI's configure script. Thank to Kevin Buckley for pointing out the issue.
- Fix performance issue in usnic BTL: ensure progress thread is throttled back to not aggressively steal CPU cycles.
- Fix cache line size detection on POWER architectures.
- Add missing #include in a few places. Thanks to Orion Poplawski for supplying the patch.
- When OpenSHMEM building is disabled, no longer install its header files, help files, or man pages. Add man pages for oshrun, oshcc, and oshfort.
- Fix mpi_f08 implementations of MPI_COMM_SET_INFO, and profiling versions of MPI_BUFFER_DETACH, MPI_WIN_ALLOCATE, MPI_WIN_ALLOCATE_SHARED, MPI_WTICK, and MPI_WTIME.
- Add orte_rmaps_dist_device MCA param, allowing users to map near a specific device.
- Various updates/fixes to the openib BTL.
- Add missing defaults for the Mellanox ConnectX 3 card to the openib BTL.
- Minor bug fixes in the OFI MTL.
- Various updates to Mellanox's MXM, hcoll, and FCA components.
- Add OpenSHMEM man pages. Thanks to Tony Curtis for sharing the man pages files from openshmem.org.
- Add missing "const" attributes to MPICOMPARE_AND_SWAP, MPI_FETCH_AND_OP, MPI_RACUMULATE, and MPI_WIN_DETECT prototypes. Thanks to Michael Knobloch and Takahiro Kawashima for bringing this to our attention.
- Fix linking issues on some platforms (e.g., SLES 12).
- Fix hang on some corner cases when MPI applications abort.
- Add missing options to mpirun man page. Thanks to Daniel Letai for bringing this to our attention.
- Add new --with-platform-patches-dir configure option
- Adjust relative selection priorities to ensure that MTL support is favored over BTL support when both are available
- Use CUDA IPC for all sized messages for performance

1.10.0 -- 25 Aug 2015
--------------------

** NOTE: The v1.10.0 release marks the transition to Open MPI's new version numbering scheme. The v1.10.x release series is based on the v1.8.x series, but with a few new features. v2.x will be the next series after the v1.10.x series, and complete the transition to the new version numbering scheme. See README for more details

** NOTE: In accordance with OMPI version numbering, the v1.10 is *not* API compatible with the v1.8 release series.

- Added libfabric support (see README for more details):
  - usNIC BTL updated to use libfabric.
  - Added OFI MTL (usable with PSM in libfabric v1.1.0).
  - Added Intel Omni-Path support via new PSM2 MTL.
  - Added "yalla" PML for faster MXM support.
  - Removed support for MX
  - Added persistent distributed virtual machine (pDVM) support for fast workflow executions.
  - Fixed typo in GCC inline assembly introduced in Open MPI v1.8.8. Thanks to Paul Hargrove for pointing out the issue.
  - Add missing man pages for MPI_Win_get/set_info(3).
- Ensure that session directories are cleaned up at the end of a run.
- Fixed linking issues on some OSs where symbols of dependent libraries are not automatically publicly available.
- Improve hcoll and fca config library detection. Thanks to David Shrader for helping track down the issue.
- Removed the LAMA mapper (for use in setting affinity). Its functionality has been largely superseded by other mpirun CLI options.
- CUDA: Made the asynchronous copy mode be the default.
- Fix a malloc(0) warning in MPI_IREDUCE_ScATTER_BLOCK. Thanks to Lisandro Dalcin for reporting the issue.
- Fix typo in MPI_Scatter(3) man page. Thanks to Akshay Venkatesh for noticing the mistake.
- Add rudimentary protection from TCP port scanners.
- Fix typo in Open MPI error handling. Thanks to Ake Sandgren for pointing out the error.
- Increased the performance of the CM PML (i.e., the Portals, PSM, PSM2, MXM, and OFI transports).
- Restored visibility of blocking send requests in message queue debuggers (e.g., TotalView, DDT).
- Fixed obscure IPv6-related bug in the TCP BTL.
- Add support for the "no_locks" MPI_Info key for one-sided functionality.
- Fixed ibv_fork support for verbs-based networks.
- Fixed a variety of small bugs in OpenSHMEM.
- Fixed MXM configure with additional CPPFLAGS and LDFLAGS. Thanks to David Shrader for the patch.
- Fixed incorrect memalign threshold in the openib BTL. Thanks to Xavier Besseron for pointing out the issue.

### 5.11 MPICH2 Release Information

The following is reproduced essentially verbatim from files contained within the MPICH2 tarball downloaded from http://www.mpich.org/downloads/.

NOTE: MPICH-2 has been effectively deprecated by the Open Source Community in favor of MPICH-3, which Scyld ClusterWare distributes as a set of mpich-scylD RPMs. Scyld ClusterWare continues to distribute mpich2-scylD, although we encourage users to migrate to MPICH-3, which enjoys active support by the Community.

<table>
<thead>
<tr>
<th>Changes in 1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td># OVERALL: Nemesis now supports an &quot;--enable-yield=...&quot; configure option for better performance/behavior when oversubscribing processes to cores. Some form of this option is enabled by default on Linux, Darwin, and systems that support sched_yield().</td>
</tr>
<tr>
<td># OVERALL: Added support for Intel Many Integrated Core (MIC) architecture: shared memory, TCP/IP, and SCIF based communication.</td>
</tr>
</tbody>
</table>
OVERALL: Added support for IBM BG/Q architecture. Thanks to IBM for the contribution.

MPI-3: const support has been added to mpi.h, although it is disabled by default. It can be enabled on a per-translation unit basis with "#define MPICH2_CONST const".

MPI-3: Added support for MPIX_Type_create_hindexed_block.

MPI-3: The new MPI-3 nonblocking collective functions are now available as "MPIX_" functions (e.g., "MPIX_Ibcast").

MPI-3: The new MPI-3 neighborhood collective routines are now available as "MPIX_" functions (e.g., "MPIX_Neighbor_allgather").

MPI-3: The new MPI-3 MPI_Comm_split_type function is now available as an "MPIX_" function.

MPI-3: The new MPI-3 tools interface is now available as "MPIX_T_" functions. This is a beta implementation right now with several limitations, including no support for multithreading. Several performance variables related to CH3’s message matching are exposed through this interface.

MPI-3: The new MPI-3 matched probe functionality is supported via the new routines MPIX_Mprobe, MPIX_Iprobe, MPIX_Mrecv, and MPIX_Imrecv.

MPI-3: The new MPI-3 nonblocking communicator duplication routine, MPIX_Comm_idup, is now supported. It will only work for single-threaded programs at this time.

MPI-3: MPIX_Comm_reenable_anysource support

MPI-3: Native MPIX_Comm_create_group support (updated version of the prior MPIX_Group_comm_create routine).

MPI-3: MPI_Intercomm_create's internal communication no longer interferes with point-to-point communication, even if point-to-point operations on the parent communicator use the same tag or MPI_ANY_TAG.

MPI-3: Eliminated the possibility of interference between MPI_Intercomm_create and point-to-point messaging operations.

Build system: Completely revamped build system to rely fully on autotools. Parallel builds ("make -j8" and similar) are now supported.

Build system: rename "./maint/updatefiles" --> "./autogen.sh" and "configure.in" --> "configure.ac"

JUMPSHOT: Improvements to Jumpshot to handle thousands of...
timelines, including performance improvements to slog2 in such cases.

# JUMPSHOT: Added navigation support to locate chosen drawable's ends when viewport has been scrolled far from the drawable.

# PM/PMI: Added support for memory binding policies.

# PM/PMI: Various improvements to the process binding support in Hydra. Several new pre-defined binding options are provided.

# PM/PMI: Upgraded to hwloc-1.5

# PM/PMI: Several improvements to PBS support to natively use the PBS launcher.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

```
```

... or at the following link:


Changes in 1.4.1

# OVERALL: Several improvements to the ARMCI API implementation within MPICH2.

# Build system: Added beta support for DESTDIR while installing MPICH2.

# PM/PMI: Upgrade hwloc to 1.2.1rc2.

# PM/PMI: Initial support for the PBS launcher.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

```
svn log -r8675:HEAD \
https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.4.1
```

... or at the following link:

# OVERALL: Improvements to fault tolerance for collective operations. Thanks to Rui Wang @ ICT for reporting several of these issues.

# OVERALL: Improvements to the universe size detection. Thanks to Yauheni Zelenko for reporting this issue.

# OVERALL: Bug fixes for Fortran attributes on some systems. Thanks to Nicolai Stange for reporting this issue.

# OVERALL: Added new ARMCI API implementation (experimental).

# OVERALL: Added new MPIX_Group_comm_create function to allow non-collective creation of sub-communicators.

# FORTRAN: Bug fixes in the MPI_DIST_GRAPH_ Fortran bindings.

# PM/PMI: Support for a manual "none" launcher in Hydra to allow for higher-level tools to be built on top of Hydra. Thanks to Justin Wozniak for reporting this issue, for providing several patches for the fix, and testing it.

# PM/PMI: Bug fixes in Hydra to handle non-uniform layouts of hosts better. Thanks to the MVAPICH group at OSU for reporting this issue and testing it.

# PM/PMI: Bug fixes in Hydra to handle cases where only a subset of the available launchers or resource managers are compiled in. Thanks to Satish Balay @ Argonne for reporting this issue.

# PM/PMI: Support for a different username to be provided for each host; this only works for launchers that support this (such as SSH).

# PM/PMI: Bug fixes for using Hydra on AIX machines. Thanks to Kitrick Sheets @ NCSA for reporting this issue and providing the first draft of the patch.

# PM/PMI: Bug fixes in memory allocation/management for environment variables that was showing up on older platforms. Thanks to Steven Sutphen for reporting the issue and providing detailed analysis to track down the bug.

# PM/PMI: Added support for providing a configuration file to pick the default options for Hydra. Thanks to Saurabh T. for reporting the issues with the current implementation and working with us to improve this option.
# PM/PMI: Improvements to the error code returned by Hydra.

# PM/PMI: Bug fixes for handling "=" in environment variable values in hydra.

# PM/PMI: Upgrade the hwloc version to 1.2.

# COLLECTIVES: Performance and memory usage improvements for MPI_Bcast in certain cases.

# VALGRIND: Fix incorrect Valgrind client request usage when MPICH2 is built for memory debugging.

# BUILD SYSTEM: "--enable-fast" and "--disable-error-checking" are once again valid simultaneous options to configure.

# TEST SUITE: Several new tests for MPI RMA operations.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

```
svn log -r7838:HEAD \
    https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.4
...
```

... or at the following link:

```
```

------------------------------------------------------------------------

KNOWN ISSUES
------------------------------------------------------------------------

### Known runtime failures

* MPI_Alltoall might fail in some cases because of the newly added fault-tolerance features. If you are seeing this error, try setting the environment variable MPICH_ENABLE_COLL_FT_RET=0.

### Threads

* ch3:sock does not (and will not) support fine-grained threading.

* MPI-I0 APIs are not currently thread-safe when using fine-grained threading (--enable-thread-cs=per-object).

* ch3:nemesis:tcp fine-grained threading is still experimental and may have correctness or performance issues. Known correctness issues include dynamic process support and generalized request support.

(continues on next page)
### Lacking channel-specific features

* ch3 does not presently support communication across heterogeneous platforms (e.g., a big-endian machine communicating with a little-endian machine).

* ch3:nemesis:mx does not support dynamic processes at this time.

* Support for "external32" data representation is incomplete. This affects the MPI_Pack_external and MPI_Unpack_external routines, as well the external data representation capabilities of ROMIO.

* ch3 has known problems in some cases when threading and dynamic processes are used together on communicators of size greater than one.

### Build Platforms

* Builds using the native "make" program on OpenSolaris fail unknown reasons. A workaround is to use GNU Make instead. See the following ticket for more information:

  http://trac.mcs.anl.gov/projects/mpich2/ticket/1122

* Build fails with Intel compiler suite 13.0, because of weak symbol issues in the compiler. A workaround is to disable weak symbol support by passing --disable-weak-symbols to configure. See the following ticket for more information:

  https://trac.mcs.anl.gov/projects/mpich2/ticket/1659

* The sctp channel is fully supported for FreeBSD and Mac OS X. As of the time of this release, bugs in the stack currently existed in the Linux kernel, and will hopefully soon be resolved. It is known to not work under Solaris and Windows. For Solaris, the SCTP API available in the kernel of standard Solaris 10 is a subset of the standard API used by the sctp channel. Cooperation with the Sun SCTP developers to support ch3:sctp under Solaris for future releases is currently ongoing. For Windows, no known kernel-based SCTP stack for Windows currently exists.

### Process Managers

* The MPD process manager can only handle relatively small amounts of data on stdin and may also have problems if there is data on stdin that is not consumed by the program.

* The SMPD process manager does not work reliably with threaded MPI processes. MPI_Comm_spawn() does not currently work for >= 256 arguments with smpd.
### Performance issues

* SMP-aware collectives do not perform as well, in select cases, as non-SMP-aware collectives, e.g. MPI_Reduce with message sizes larger than 64KiB. These can be disabled by the configure option "--disable-smpcoll".

* MPI_Irecv operations that are not explicitly completed before MPI_Finalize is called may fail to complete before MPI_Finalize returns, and thus never complete. Furthermore, any matching send operations may erroneously fail. By explicitly completed, we mean that the request associated with the operation is completed by one of the MPI_Test or MPI_Wait routines.

### C++ Binding:

* The MPI datatypes corresponding to Fortran datatypes are not available (e.g., no MPI::DOUBLE_PRECISION).

* The C++ binding does not implement a separate profiling interface, as allowed by the MPI-2 Standard (Section 10.1.10 Profiling).

* MPI::ERRORS_RETURN may still throw exceptions in the event of an error rather than silently returning.

---

### 5.12 MPICH-3 Release Information

The following is reproduced essentially verbatim from files contained within the MPICH-3 tarball downloaded from https://www.mpich.org. See https://www.mpich.org/documentation/guides for various user guides.

#### 5.12.1 CHANGELOG

```
Changes in 3.2.1

# Fixes for platforms with strict memory alignment requirements.

# Fixes for MPI_Win info management.

# Fixed a progress bug with MPI generalized requests.

# Fixed multiple integer overflow bugs in CH3 and ROMIO.

# Improved detection for Fortran 2008 binding support.
```
# Enhanced support for libfabric (OFI) netmod.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.2..v3.2.1

===============================================================================

Changes in 3.2

===============================================================================

# Added support for MPI-3.1 features including nonblocking collective I/O, address manipulation routines, thread-safety for MPI initialization, pre-init functionality, and new MPI_T routines to look up variables by name.

# Fortran 2008 bindings are enabled by default and fully supported.

# Added support for the Mellanox MXM InfiniBand interface. (thanks to Mellanox for the code contribution).

# Added support for the Mellanox HCOLL interface for collectives. (thanks to Mellanox for the code contribution).

# Significant stability improvements to the MPICH/portals4 implementation.

# Completely revamped RMA infrastructure including several scalability improvements, performance improvements, and bug fixes.

# Added experimental support for Open Fabrics Interfaces (OFI) version 1.0.0. https://github.com/ofiwg/libfabric (thanks to Intel for code contribution)

# The Myrinet MX network module, which had a life cycle from 1.1 till 3.1.2, has now been deleted.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.1.3..v3.2rc1

A full list of bugs that have been fixed is available at the following link:

https://trac.mpich.org/projects/mpich/
  query?status=closed&group=resolution&milestone=mpich-3.2

===============================================================================

(continues on next page)
Changes in 3.1.4
===============================================================================
# Bug fixes to MPI-3 shared memory functionality.
# Fixed a bug that prevented Fortran programs from being profiled by PMPI libraries written in C.
# Fixed support for building MPICH on OSX with Intel C/C++ and Fortran compilers.
# Several bug fixes in ROMIO.
# Enhancements to the testsuite.
# Backports support for the Mellanox MXM InfiniBand interface.
# Backports support for the Mellanox HCOLL interface for collectives.
# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.1.3..v3.1.4

Changes in 3.1.3
===============================================================================
# Several enhancements to Portals4 support.
# Several enhancements to PAMI (thanks to IBM for the code contribution).
# Several enhancements to the CH3 RMA implementation.
# Several enhancements to ROMIO.
# Fixed deadlock in multi-threaded MPI_Comm_idup.
# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.1.2..v3.1.3

A full list of bugs that have been fixed is available at the following link:

https://trac.mpich.org/projects/mpich/ \query?status=closed&group=resolution&milestone=mpich-3.1.3
Changes in 3.1.2

# Significant enhancements to the BG/Q device, especially for RMA and shared memory functionality.

# Several enhancements to ROMIO.

# Upgraded to hwloc-1.9.

# Added more Fortran 2008 (F08) tests and fixed a few F08 binding bugs. Now all MPICH F90 tests have been ported to F08.

# Updated weak alias support to align with gcc-4.x

# Minor enhancements to the CH3 RMA implementation.

# Better implementation of MPI_Allreduce for intercommunicator.

# Added environment variables to control memory tracing overhead.

# Added flags to enable C99 mode with Solaris compilers.

# Updated implementation of MPI-T CVARs of type MPI_CHAR, as interpreted in MPI-3.0 Errata.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.1.1..v3.1.2

A full list of bugs that have been fixed is available at the following link:

https://trac.mpich.org/projects/mpich/\query?status=closed&group=resolution&milestone=mpich-3.1.2

Changes in 3.1.1

# Blue Gene/Q implementation supports MPI-3. This release contains a functional and compliant Blue Gene/Q implementation of the MPI-3 standard. Instructions to build on Blue Gene/Q are on the mpich.org wiki:


# Fortran 2008 bindings (experimental). Build with --enable-fortran=all. Must have a Fortran 2008 + TS 29113 capable compiler.
# Significant rework of MPICH library management and which symbols go into which libraries. Also updated MPICH library names to make them consistent with Intel MPI, Cray MPI and IBM PE MPI. Backward compatibility links are provided for older mpich-based build systems.

# The ROMIO "Blue Gene" driver has seen significant rework. We have separated "file system" features from "platform" features, since GPFS shows up in more places than just Blue Gene

# New ROMIO options for aggregator selection and placement on Blue Gene

# Optional new ROMIO two-phase algorithm requiring less communication for certain workloads

# The old ROMIO optimization "deferred open" either stopped working or was disabled on several platforms.

# Added support for powerpcle compiler. Patched libtool in MPICH to support little-endian powerpc linux host.

# Fixed the prototype of the Reduce_local C++ binding. The previous prototype was completely incorrect. Thanks to Jeff Squyres for reporting the issue.

# The mpd process manager, which was deprecated and unsupported for the past four major release series (1.3.x till 3.1), has now been deleted. RIP.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.1..v3.1.1

A full list of bugs that have been fixed is available at the following link:

https://trac.mpich.org/projects/mpich/ \ query?status=closed&group=resolution&milestone=mpich-3.1.1

===============================================================================

Changes in 3.1
===============================================================================

# Implement runtime compatibility with MPICH-derived implementations as per the ABI Compatibility Initiative (see http://www.mpich.org/abi for more information).

# Integrated MPICH-PAMI code base for Blue Gene/Q and other IBM platforms.

(continues on next page)
# Several improvements to the SCIF netmod. (code contribution from Intel).

# Major revamp of the MPI_T interface added in MPI-3.

# Added environment variables to control a lot more capabilities for collectives. See the README.envvar file for more information.

# Allow non-blocking collectives and fault tolerance at the same time. The option MPIR_PARAM_ENABLE_COLL_FT_RET has been deprecated as it is no longer necessary.

# Improvements to MPI_WIN_ALLOCATE to internally allocate shared memory between processes on the same node.

# Performance improvements for MPI RMA operations on shared memory for MPI_WIN_ALLOCATE and MPI_WIN_ALLOCATE_SHARED.

# Enable shared library builds by default.

# Upgraded hwloc to 1.8.

# Several improvements to the Hydra-SLURM integration.

# Several improvements to the Hydra process binding code. See the Hydra wiki page for more information: http://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager

# MPICH now supports operations on very large datatypes (those that describe more than 32 bits of data). This work also allows MPICH to fully support MPI-3’s introduction of MPI_Count.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.0.4..v3.1

A full list of bugs that have been fixed is available at the following link:

https://trac.mpich.org/projects/mpich/ \ query?status=closed&group.resolution=milestone=mpich-3.1

===============================================================================
Changes in 3.0.4
===============================================================================

# BUILD SYSTEM: Reordered the default compiler search to prefer Intel and PG compilers over GNU compilers because of the performance
difference.

WARNING: If you do not explicitly specify the compiler you want through CC and friends, this might break ABI for you relative to the previous 3.0.x release.

# OVERALL: Added support to manage per-communicator eager-rendezvous thresholds.

# PM/PMI: Performance improvements to the Hydra process manager on large-scale systems by allowing for key/value caching.

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.0.3..v3.0.4

===============================================================================
Changes in 3.0.3
===============================================================================

# RMA: Added a new mechanism for piggybacking RMA synchronization operations, which improves the performance of several synchronization operations, including Flush.

# RMA: Added an optimization to utilize the MPI_MODE_NOCHECK assertion in passive target RMA to improve performance by eliminating a lock request message.

# RMA: Added a default implementation of shared memory windows to CH3. This adds support for this MPI 3.0 feature to the ch3:sock device.

# RMA: Fix a bug that resulted in an error when RMA operation request handles where completed outside of a synchronization epoch.

# PM/PMI: Upgraded to hwloc-1.6.2rc1. This version uses libpciaccess instead of libpci, to workaround the GPL license used by libpci.

# PM/PMI: Added support for the Cobalt process manager.

# BUILD SYSTEM: allow MPI_LONG_DOUBLE_SUPPORT to be disabled with a configure option.

# FORTRAN: fix MPI_WEIGHTS_EMPTY in the Fortran bindings

# MISC: fix a bug in MPI_Get_elements where it could return incorrect values

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.0.2..v3.0.3
Changes in 3.0.2

# PM/PMI: Upgrade to hwloc-1.6.1

# RMA: Performance enhancements for shared memory windows.

# COMPILER INTEGRATION: minor improvements and fixes to the clang static type checking annotation macros.

# MPI-I0 (ROMIO): improved error checking for user errors, contributed by IBM.

# MPI-3 TOOLS INTERFACE: new MPI_T performance variables providing information about nemesis communication behavior and and CH3 message matching queues.

# TEST SUITE: "make testing" now also outputs a "summary.tap" file that can be interpreted with standard TAP consumer libraries and tools. The "summary.xml" format remains unchanged.

# GIT: This is the first release built from the new git repository at git.mpich.org. A few build system mechanisms have changed because of this switch.

# BUG FIX: resolved a compilation error related to LLONG_MAX that affected several users (ticket #1776).

# BUG FIX: nonblocking collectives now properly make progress when MPICH is configured with the ch3:sock channel (ticket #1785).

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available at the following link:

http://git.mpich.org/mpich.git/shortlog/v3.0.1..v3.0.2

Changes in 3.0.1

# PM/PMI: Critical bug-fix in Hydra to work correctly in multi-node tests.

# A full list of changes is available using:

svn log -r10790:HEAD \
   https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich-3.0.1

... or at the following link:
Changes in 3.0

# MPI-3: All MPI-3 features are now implemented and the MPI_VERSION bumped up to 3.0.

# OVERALL: Added support for ARM-v7 native atomics

# MPE: MPE is now separated out of MPICH and can be downloaded/used as a separate package.

# PM/PMI: Upgraded to hwloc-1.6

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available using:

```
svn log -r10344:HEAD \n   https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich-3.0
```

... or at the following link:


Changes in 1.5

# OVERALL: Nemesis now supports an "--enable-yield=..." configure option for better performance/behavior when oversubscribing processes to cores. Some form of this option is enabled by default on Linux, Darwin, and systems that support sched_yield().

# OVERALL: Added support for Intel Many Integrated Core (MIC) architecture: shared memory, TCP/IP, and SCIF based communication.

# OVERALL: Added support for IBM BG/Q architecture. Thanks to IBM for the contribution.

# MPI-3: const support has been added to mpi.h, although it is disabled by default. It can be enabled on a per-translation unit basis with "#define MPICH2_CONST const".

# MPI-3: Added support for MPIX_Type_create_hindexed_block.

# MPI-3: The new MPI-3 nonblocking collective functions are now (continues on next page)
available as "MPIX_" functions (e.g., "MPIX_Ibcast").

# MPI-3: The new MPI-3 neighborhood collective routines are now available as "MPIX_" functions (e.g., "MPIX_Neighbor_allgather").

# MPI-3: The new MPI-3 MPI_Comm_split_type function is now available as an "MPIX_" function.

# MPI-3: The new MPI-3 tools interface is now available as "MPIX_T_" functions. This is a beta implementation right now with several limitations, including no support for multithreading. Several performance variables related to CH3's message matching are exposed through this interface.

# MPI-3: The new MPI-3 matched probe functionality is supported via the new routines MPIX_Mprobe, MPIX_Improbe, MPIX_Mrecv, and MPIX_Irecv.

# MPI-3: The new MPI-3 nonblocking communicator duplication routine, MPIX_Comm_idup, is now supported. It will only work for single-threaded programs at this time.

# MPI-3: MPIX_Comm_reenable_anysource support

# MPI-3: Native MPIX_Comm_create_group support (updated version of the prior MPIX_Group_comm_create routine).

# MPI-3: MPIX_Comm_reenable_anysource support

# MPI-3: MPIX_Comm_reenable_anysource support

# Build system: Completely revamped build system to rely fully on autotools. Parallel builds ("make -j8" and similar) are now supported.

# Build system: rename "./maint/updatefiles" --> "/autogen.sh" and "configure.in" --> "configure.ac"

# JUMPSHOT: Improvements to Jumpshot to handle thousands of timelines, including performance improvements to slog2 in such cases.

# JUMPSHOT: Added navigation support to locate chosen drawable's ends when viewport has been scrolled far from the drawable.

# PM/PMI: Added support for memory binding policies.

# PM/PMI: Various improvements to the process binding support in Hydra. Several new pre-defined binding options are provided.

(continues on next page)
# PM/PMI: Upgraded to hwloc-1.5

# PM/PMI: Several improvements to PBS support to natively use the PBS launcher.

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available using:

```
```
... or at the following link:

```
```

Changes in 1.4.1

# OVERALL: Several improvements to the ARMCi API implementation within MPICH2.

# Build system: Added beta support for DESTDIR while installing MPICH2.

# PM/PMI: Upgrade hwloc to 1.2.1rc2.

# PM/PMI: Initial support for the PBS launcher.

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available using:

```
svn log -r8675:HEAD \\n  https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.4.1
```
... or at the following link:

```
```

Changes in 1.4

# OVERALL: Improvements to fault tolerance for collective operations. Thanks to Rui Wang @ ICT for reporting several of these issues.

# OVERALL: Improvements to the universe size detection. Thanks to
Yauheni Zelenko for reporting this issue.

# OVERALL: Bug fixes for Fortran attributes on some systems. Thanks to Nicolai Stange for reporting this issue.

# OVERALL: Added new ARMCII API implementation (experimental).

# OVERALL: Added new MPIX_Group_comm_create function to allow non-collective creation of sub-communicators.

# FORTRAN: Bug fixes in the MPI_DIST_GRAPH_Fortran bindings.

# PM/PMI: Support for a manual "none" launcher in Hydra to allow for higher-level tools to be built on top of Hydra. Thanks to Justin Wozniak for reporting this issue, for providing several patches for the fix, and testing it.

# PM/PMI: Bug fixes in Hydra to handle non-uniform layouts of hosts better. Thanks to the MVAPICH group at OSU for reporting this issue and testing it.

# PM/PMI: Bug fixes in Hydra to handle cases where only a subset of the available launchers or resource managers are compiled in. Thanks to Satish Balay @ Argonne for reporting this issue.

# PM/PMI: Support for a different username to be provided for each host; this only works for launchers that support this (such as SSH).

# PM/PMI: Bug fixes for using Hydra on AIX machines. Thanks to Kitrick Sheets @ NCSA for reporting this issue and providing the first draft of the patch.

# PM/PMI: Bug fixes in memory allocation/management for environment variables that was showing up on older platforms. Thanks to Steven Sutphen for reporting the issue and providing detailed analysis to track down the bug.

# PM/PMI: Added support for providing a configuration file to pick the default options for Hydra. Thanks to Saurabh T. for reporting the issues with the current implementation and working with us to improve this option.

# PM/PMI: Improvements to the error code returned by Hydra.

# PM/PMI: Bug fixes for handling "=" in environment variable values in hydra.

# PM/PMI: Upgrade the hwloc version to 1.2.

# COLLECTIVES: Performance and memory usage improvements for MPI_Bcast in certain cases.
# VALGRIND: Fix incorrect Valgrind client request usage when MPICH2 is built for memory debugging.

# BUILD SYSTEM: "--enable-fast" and "--disable-error-checking" are once again valid simultaneous options to configure.

# TEST SUITE: Several new tests for MPI RMA operations.

# Several other minor bug fixes, memory leak fixes, and code cleanup.
A full list of changes is available using:

```
svn log -r7838:HEAD \
   https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.4
```

... or at the following link:

https://trac.mcs.anl.gov/projects/mpich2/log/mpich2/tags/release/ \ 
   mpich2-1.4?action=follow_copy&rev=HEAD&stop_rev=7838&mode=follow_copy

Changes in 1.3.2

# OVERALL: MPICH2 now recognizes the OSX mach_absolute_time as a native timer type.

# OVERALL: Performance improvements to MPI_Comm_split on large systems.

# OVERALL: Several improvements to error returns capabilities in the presence of faults.

# PM/PMI: Several fixes and improvements to Hydra's process binding capability.

# PM/PMI: Upgrade the hwloc version to 1.1.1.

# PM/PMI: Allow users to sort node lists allocated by resource managers in Hydra.

# PM/PMI: Improvements to signal handling. Now Hydra respects Ctrl-Z signals and passes on the signal to the application.

# PM/PMI: Improvements to STDOUT/STDERR handling including improved support for rank prepending on output. Improvements to STDIN handling for applications being run in the background.

# PM/PMI: Split the bootstrap servers into "launchers" and "resource managers", allowing the user to pick a different resource manager from the launcher. For example, the user can now pick the "SLURM"
resource manager and "SSH" as the launcher.

# PM/PMI: The MPD process manager is deprecated.

# PM/PMI: The PLPA process binding library support is deprecated.

# WINDOWS: Adding support for gfortran and 64-bit gcc libs.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

    svn log -r7457:HEAD \n    https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.3.2

... or at the following link:


Changes in 1.3.1

# OVERALL: MPICH2 is now fully compliant with the CIFTS FTB standard MPI events (based on the draft standard).

# OVERALL: Major improvements to RMA performance for long lists of RMA operations.

# OVERALL: Performance improvements for Group_translate_ranks.

# COLLECTIVES: Collective algorithm selection thresholds can now be controlled at runtime via environment variables.

# ROMIO: PVFS error codes are now mapped to MPI error codes.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

    svn log -r7350:HEAD \n    https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.3.1

... or at the following link:


Changes in 1.3

(continues on next page)
# OVERALL: Initial support for fine-grained threading in ch3:nemesis:tcp.

# OVERALL: Support for Asynchronous Communication Progress.

# OVERALL: The ssm and shm channels have been removed.

# OVERALL: Checkpoint/restart support using BLCR.

# OVERALL: Improved tolerance to process and communication failures when error handler is set to MPI_ERRORS_RETURN. If a communication operation fails (e.g., due to a process failure) MPICH2 will return an error, and further communication to that process is not possible. However, communication with other processes will still proceed normally. Note, however, that the behavior collective operations on communicators containing the failed process is undefined, and may give incorrect results or hang some processes.

# OVERALL: Experimental support for inter-library dependencies.

# PM/PMI: Hydra is now the default process management framework replacing MPD.

# PM/PMI: Added dynamic process support for Hydra.

# PM/PMI: Added support for LSF, SGE and POE in Hydra.

# PM/PMI: Added support for CPU and memory/cache topology aware process-core binding.

# DEBUGGER: Improved support and bug fixes in the Totalview support.

# Build system: Replaced F90/F90FLAGS by FC/FCFLAGS. F90/F90FLAGS are not longer supported in the configure.

# Multi-compiler support: On systems where C compiler that is used to build mpich2 libraries supports multiple weak symbols and multiple aliases, the Fortran binding built in the mpich2 libraries can handle different Fortran compilers (than the one used to build mpich2). Details in README.

# Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

```
svn log -r5762:HEAD \\
    https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.3
```

... or at the following link:

```
    mpich2-1.3?action=follow_copy&rev=HEAD&stop_rev=5762&mode=follow_copy
```
Changes in 1.2.1

# OVERALL: Improved support for fine-grained multithreading.

# OVERALL: Improved integration with Valgrind for debugging builds of MPICH2.

# PM/PMI: Initial support for hwloc process-core binding library in Hydra.

# PM/PMI: Updates to the PMI-2 code to match the PMI-2 API and wire-protocol draft.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available using:

    svn log -r5425:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.2.1

... or at the following link:


Changes in 1.2

# OVERALL: Support for MPI-2.2

# OVERALL: Several fixes to Nemesis/MX.

# WINDOWS: Performance improvements to Nemesis/windows.

# PM/PMI: Scalability and performance improvements to Hydra using PMI-1.1 process-mapping features.

# PM/PMI: Support for process-binding for hyperthreading enabled systems in Hydra.

# PM/PMI: Initial support for PBS as a resource management kernel in Hydra.

# PM/PMI: PMI2 client code is now officially included in the release.

# TEST SUITE: Support to run the MPICH2 test suite through valgrind.

# Several other minor bug fixes, memory leak fixes, and code cleanup.

A full list of changes is available using:
svn log -r5025:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.2
... or at the following link:

https://trac.mcs.anl.gov/projects/mpich2/log/mpich2/tags/release/mpich2-1.2?\ 
action=follow_copy&rev=HEAD&stop_rev=5025&mode=follow_copy

===============================================================================
Changes in 1.1.1p1
===============================================================================

- OVERALL: Fixed an invalid read in the dataloop code for zero count types.
- OVERALL: Fixed several bugs in ch3:nemesis:mx (tickets #744,#760; also change r5126).
- BUILD SYSTEM: Several fixes for functionality broken in 1.1.1 release, including MPICH2LIB_xFLAGS and extra libraries living in $LIBS instead of $LDFLAGS. Also, '-lpthread' should no longer be duplicated in link lines.
- BUILD SYSTEM: MPICH2 shared libraries are now compatible with glibc versioned symbols on Linux, such as those present in the MX shared libraries.
- BUILD SYSTEM: Minor tweaks to improve compilation under the nvcc CUDA compiler.
- PM/PMI: Fix mpd incompatibility with python2.3 introduced in mpich2-1.1.1.
- PM/PMI: Several fixes to hydra, including memory leak fixes and process binding issues.
- TEST SUITE: Correct invalid arguments in the coll2 and coll3 tests.
- Several other minor bug fixes, memory leak fixes, and code cleanup. A full list of changes is available using:

  svn log -r5032:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.1.1p1

... or at the following link:

https://trac.mcs.anl.gov/projects/mpich2/log/mpich2/tags/release/mpich2-1.1.1p1?\ 
action=follow_copy&rev=HEAD&stop_rev=5032&mode=follow_copy

===============================================================================
Changes in 1.1.1
===============================================================================

# OVERALL: Improved support for Boost MPI.
# PM/PMI: Significantly improved time taken by MPI_Init with Nemesis and MPD on large numbers of processes.

# PM/PMI: Improved support for hybrid MPI-UPC program launching with Hydra.

# PM/PMI: Improved support for process-core binding with Hydra.

# PM/PMI: Preliminary support for PMI-2. Currently supported only with Hydra.

# Many other bug fixes, memory leak fixes and code cleanup. A full list of changes is available using:

svn log -r4655:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.1.1

... or at the following link:


===============================================================================
Changes in 1.1
===============================================================================

- OVERALL: Added MPI 2.1 support.

- OVERALL: Nemesis is now the default configuration channel with a completely new TCP communication module.

- OVERALL: Windows support for nemesis.

- OVERALL: Added a new Myrinet MX network module for nemesis.

- OVERALL: Initial support for shared-memory aware collective communication operations. Currently MPI_Bcast, MPI_Reduce, MPI_Allreduce, and MPI_Scan.

- OVERALL: Improved handling of MPI Attributes.

- OVERALL: Support for BlueGene/P through the DCMF library (thanks to IBM for the patch).

- OVERALL: Experimental support for fine-grained multithreading

- OVERALL: Added dynamic processes support for Nemesis.

- OVERALL: Added automatic as well as statically runtime configurable receive timeout variation for MPD (thanks to OSU for the patch).
- OVERALL: Improved performance for MPI_Allgatherv, MPI_Gatherv, and MPI_Alltoall.

- PM/PMI: Initial support for the new Hydra process management framework (current support is for ssh, rsh, fork and a preliminary version of slurm).

- ROMIO: Added support for MPI_Type_create_resized and MPI_Type_create_indexed_block datatypes in ROMIO.

- ROMIO: Optimized Lustre ADIO driver (thanks to Weikuan Yu for initial work and Sun for further improvements).

- Many other bug fixes, memory leak fixes and code cleanup. A full list of changes is available using:

  svn log -r813:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/tags/release/mpich2-1.1

  ... or at the following link:


Changes in 1.0.7

- OVERALL: Initial ROMIO device for BlueGene/P (the ADI device is also added but is not configurable at this time).

- OVERALL: Major clean up for the propagation of user-defined and other MPICH2 flags throughout the code.

- OVERALL: Support for STI Cell Broadband Engine.

- OVERALL: Added datatype free hooks to be used by devices independently.

- OVERALL: Added device-specific timer support.

- OVERALL: make uninstall works cleanly now.

- ROMIO: Support to take hints from a config file

- ROMIO: more tests and bug fixes for nonblocking I/O

- PM/PMI: Added support to use PMI Clique functionality for process managers that support it.

- PM/PMI: Added SLURM support to configure to make it transparent to users.
- PM/PMI: SMPD Singleton Init support.
- WINDOWS: Fortran 90 support added.
- SCTP: Added MPICH_SCTP_NAGLE_ON support.
- MPE: Updated MPE logging API so that it is thread-safe (through global mutex).
- MPE: Added infrastructure to piggyback argument data to MPI states.
- DOCS: Documentation creation now works correctly for VPATH builds.
- Many other bug fixes, memory leak fixes and code cleanup. A full list of changes is available using:
  
  svn log -r100:HEAD https://svn.mcs.anl.gov/repos/mpi/mpich2/branches/release/MPICH2_1_0_7

===============================================================================

Changes in 1.0.6
===============================================================================

- Updates to the ch3:nemesis channel including preliminary support for thread safety.
- Preliminary support for dynamic loading of ch3 channels (sock, ssm, shm). See the README file for details.
- Singleton init now works with the MPD process manager.
- Fixes in MPD related to MPI-2 connect-accept.
- Improved support for MPI-2 generalized requests that allows true nonblocking I/O in ROMIO.

- MPE changes:
  * Enabled thread-safe MPI logging through global mutex.
  * Enhanced Jumpshot to be more thread friendly
    + added simple statistics in the Legend windows.
  * Added backtrace support to MPE on Solaris and glibc based systems, e.g. Linux. This improves the output error message from the Collective/Datatype checking library.
  * Fixed the CLOG2 format so it can be used in serial (non-MPI) logging.

- Performance improvements for derived datatypes (including packing and communication) through in-built loop-unrolling and buffer alignment.

- Performance improvements for MPI_Gather when non-power-of-two processes are used, and when a non-zero ranked root is performing the gather.
- MPI_Comm_create works for intercommunicators.

- Enabled -O2 and equivalent compiler optimizations for supported compilers by default (including GNU, Intel, Portland, Sun, Absoft, IBM).

- Many other bug fixes, memory leak fixes and code cleanup. A full list of changes is available at www.mcs.anl.gov/mpi/mpich2/mpich2_1_0_6changes.htm.

===============================================================================
Changes in 1.0.5
===============================================================================

- An SCTP channel has been added to the CH3 device. This was implemented by Brad Penoff and Mike Tsai, Univ. of British Columbia. Their group's webpage is located at http://www.cs.ubc.ca/labs/dsg/mpi-sctp/.

- Bugs related to dynamic processes have been fixed.

- Performance-related fixes have been added to derived datatypes and collective communication.

- Updates to the Nemesis channel

- Fixes to thread safety for the ch3:sock channel

- Many other bug fixes and code cleanup. A full list of changes is available at www.mcs.anl.gov/mpi/mpich2/mpich2_1_0_5changes.htm.

===============================================================================
Changes in 1.0.4
===============================================================================

- For the ch3:sock channel, the default build of MPICH2 supports thread safety. A separate build is not needed as before. However, thread safety is enabled only if the user calls MPI_Init_thread with MPI_THREAD_MULTIPLE. If not, no thread locks are called, so there is no penalty.

- A new low-latency channel called Nemesis has been added. It can be selected by specifying the option --with-device=ch3:nemesis. Nemesis uses shared memory for intranode communication and various networks for internode communication. Currently available networks are TCP, GM and MX. Nemesis is still a work in progress. See the README for more information about the channel.

- Support has been added for providing message queues to debuggers. Configure with --enable-debuginfo to make this information available.
This is still a "beta" test version and has not been extensively tested.

- For systems with firewalls, the environment variable MPICH_PORT_RANGE can be used to restrict the range of ports used by MPICH2. See the documentation for more details.

- Withdrew obsolete modules, including the ib and rdma communication layers. For Infiniband and MPICH2, please see http://nowlab.cse.ohio-state.edu/projects/mpi-iba/ For other interconnects, please contact us at mpich2-maint@mcs.anl.gov.

- Numerous bug fixes and code cleanup. A full list of changes is available at www.mcs.anl.gov/mpi/mpich2/mpich2_1_0_4changes.htm.

- Numerous new tests in the MPICH2 test suite.

- For developers, the way in which information is passed between the top level configure and configures in the device, process management, and related modules has been cleaned up. See the comments at the beginning of the top-level configure.in for details. This change makes it easier to interface other modules to MPICH2.

Changes in 1.0.3

- There are major changes to the ch3 device implementation. Old and unsupported channels (essm, rdma) have been removed. The internal interface between ch3 and the channels has been improved to simplify the process of adding a new channel (sharing existing code where possible) and to improve performance. Further changes in this internal interface are expected.

- Numerous bug fixes and code cleanup

  Creation of intercommunicators and intracommunicators from the intercommunicators created with Spawn and Connect/Accept

  The computation of the alignment and padding of items within structures now handles additional cases, including systems where the alignment an padding depends on the type of the first item in the structure

  MPD recognizes wdir info keyword

  gforker's mpiexec supports -env and -genv arguments for controlling which environment variables are delivered to created processes

- While not a bug, to aid in the use of memory trace packages, MPICH2 tries to free all allocated data no later than when MPI_Finalize returns.
- Support for DESTDIR in install targets

- Enhancements to SMPD

- In order to support special compiler flags for users that may be different from those used to build MPICH2, the environment variables MPI_CFLAGS, MPI_FFLAGS, MPI_CXXFLAGS, and MPI_F90FLAGS may be used to specify the flags used in mpicc, mpif77, mpicxx, and mpif90 respectively. The flags CFLAGS, FFLAGS, CXXFLAGS, and F90FLAGS are used in the building of MPICH2.

- Many enhancements to MPE

- Enhanced support for features and idiosyncracies of Fortran 77 and Fortran 90 compilers, including gfortran, g95, and xlf

- Enhanced support for C++ compilers that do not fully support abstract base classes

- Additional tests in the mpich2/tests/mpi

- New FAQ included (also available at http://www.mcs.anl.gov/mpi/mpich2/faq.htm)

- Man pages for mpiexec and mpif90

- Enhancements for developers, including a more flexible and general mechanism for inserting logging and information messages, controllable with --mpich-dbg-xxx command line arguments or MPICH_DBG_XXX environment variables.

- Note to developers:
  This release contains many changes to the structure of the CH3 device implementation (in src/mpid/ch3), including significant reworking of the files (many files have been combined into fewer files representing logical grouping of functions). The next release of MPICH2 will contain even more significant changes to the device structure as we introduce a new communication implementation.

===============================================================================
Changes in 1.0.2
===============================================================================

- Optimizations to the MPI-2 one-sided communication functions for the sshm (scalable shared memory) channel when window memory is allocated with MPI_Alloc_mem (for all three synchronization methods).

- Numerous bug fixes and code cleanup.

- Fixed memory leaks.
- Fixed shared library builds.

- Fixed performance problems with MPI_Type_create_subarray/darray

- The following changes have been made to MPE2:

  - MPE2 now builds the MPI collective and datatype checking library by default.

  - SLOG-2 format has been upgraded to 2.0.6 which supports event drawables and provides count of real drawables in preview drawables.

  - New slog2 tools, slog2filter and slog2updater, which both are logfile format convertors. slog2filter removes undesirable categories of drawables as well as alters the slog2 file structure. slog2updater is a slog2filter that reads in older logfile format, 2.0.5, and writes out the latest format 2.0.6.

- The following changes have been made to MPD:

  - Nearly all code has been replaced by new code that follows a more object-oriented approach than before. This has not changed any fundamental behavior or interfaces.

  - There is info support in spawn and spawn_multiple for providing parts of the environment for spawned processes such as search-path and current working directory. See the Standard for the required fields.

  - mpdccheck has been enhanced to help users debug their cluster and network configurations.

  - CPickle has replaced marshal as the source module for dumps and loads.

  - The mpiexec command has been replaced by mpiexec -gdb.

  - Alternate interfaces can be used. See the Installer's Guide.

===============================================================================
Changes in 1.0.1
===============================================================================

- Copyright statements have been added to all code files, clearly identifying that all code in the distribution is covered by the extremely flexible copyright described in the COPYRIGHT file.

- The MPICH2 test suite (mpich2/test) can now be run against any MPI implementation, not just MPICH2.

- The send and receive socket buffers sizes may now be changed by setting MPICH_SOCKET_BUFFER_SIZE. Note: the operating system may impose a maximum
socket buffer size that prohibits MPICH2 from increasing the buffers to the desire size. To raise the maximum allowable buffer size, please contact your system administrator.

- Error handling throughout the MPI routines has been improved. The error handling in some internal routines has been simplified as well, making the routines easier to read.

- MPE (Jumpshot and CLOG logging) is now supported on Microsoft Windows.

- C applications built for Microsoft Windows may select the desired channels at runtime.

- A program not started with mpiexec may become an MPI program by calling MPI_Init. It will have an MPI_COMM_WORLD of size one. It may then call other MPI routines, including MPI_COMM_SPAWN, to become a truly parallel program. At present, the use of MPI_COMM_SPAWN and MPI_COMM_SPAWN_MULTIPLE by such a process is only supported by the MPD process manager.

- Memory leaks in communicator allocation and the C++ binding have been fixed.

- Following GNU guidelines, the parts of the install step that checked the installation have been moved to an installcheck target. Much of the installation now supports the DESTDIR prefix.

- Microsoft Visual Studio projects have been added to make it possible to build x86-64 version

- Problems with compilers and linkers that do not support weak symbols, which are used to support the PMPI profiling interface, have been corrected.

- Handling of Fortran 77 and Fortran 90 compilers has been improved, including support for g95.

- The Fortran stdcall interface on Microsoft Windows now supports character*.

- A bug in the OS X implementation of poll() caused the sock channel to hang. A workaround has been put in place.

- Problems with installation under OS/X are now detected and corrected. (Install breaks libraries that are more than 10 seconds old!)

- The following changes have been made to MPD:

  - Sending a SIGINT to mpiexec/mpdrun, such as by typing control-C, now causes SIGINT to be sent to the processes within the job. Previously, SIGKILL was sent to the processes, preventing applications from catching the signal and performing their own signal processing.

  - The process for merging output has been improved.

  - A new option, -ifhn, has been added to the machine file, allowing the user
to select the destination interface to be used for TCP communication. See the User's Manual for details.

- The user may now select, via the "-s" option to mpiexec/mpdrun, which processes receive input through stdin. stdin is immediately closed for all processes not in set receiving input. This prevents processes not in the set from hanging should they attempt to read from stdin.

- The MPICH2 Installer's Guide now contains an appendix on troubleshooting problems with MPD.

- The following changes have been made to SMPD:
  
  - On Windows machines, passwordless authentication (via SSPI) can now be used to start processes on machines within a domain. This feature is a recent addition, and should be considered experimental.

  - On Windows machines, the -localroot option was added to mpiexec, allowing processes on the local machines to perform GUI operations on the local desktop.

  - On Windows machines, network drive mapping is now supported via the -map option to mpiexec.

  - Three new GUI tools have been added for Microsoft Windows. These tools are wrappers to the command line tools, mpiexec.exe and smpd.exe. wmmpiexec allows the user to run a job much in the way they with mpiexec. wmpiconfig provides a means of setting various global options to the SMPD process manager environment. wmpiregister encrypts the user's credentials and saves them to the Windows Registry.

- The following changes have been made to MPE2:

  - MPE2 no longer attempt to compile or link code during 'make install' to validate the installation. Instead, 'make installcheck' may now be used to verify that the MPE installation.

  - MPE2 now supports DESTDIR.

  - The sock channel now has preliminary support for MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE on both UNIX and Microsoft Windows. We have performed rudimentary testing; and while overall the results were very positive, known issues do exist. ROMIO in particular experiences hangs in several places. We plan to correct that in the next release. As always, please report any difficulties you encounter.

  - Another channel capable of communicating with both over sockets and shared memory has been added. Unlike the ssm channel which waits for new data to arrive by continuously polling the system in a busy loop, the essm channel waits by blocking on an operating system event object. This channel is experimental, and is only available for Microsoft Windows.
- The topology routines have been modified to allow the device to override the
default implementation. This allows the device to export knowledge of the
underlying physical topology to the MPI routines (Dims_create and the
reorder == true cases in Cart_create and Graph_create).

- New memory allocation macros, MPIU_CHK[PL]MEM_*(), have been added to help
prevent memory leaks. See mpich2/src/include/mpimem.h.

- New error reporting macros, MPIU_ERR_*, have been added to simplify the error
handling throughout the code, making the code easier to read. See
mpich2/src/include/mpierrs.h.

- Interprocess communication using the Sock interface (sock and ssm channels)
may now be bound to a particular destination interface using the environment
variable MPICH_INTERFACE_HOSTNAME. The variable needs to be set for each
process for which the destination interface is not the default interface.
(Other mechanisms for destination interface selection will be provided in
future releases.) Both MPD and SMPD provide a more simplistic mechanism for
specifying the interface. See the user documentation.

- Too many bug fixes to describe. Much thanks goes to the users who reported
bugs. Their patience and understanding as we attempted to recreate the
problems and solve them is greatly appreciated.

===============================================================================
Changes in 1.0
===============================================================================

- MPICH2 now works on Solaris.

- The User's Guide has been expanded considerably. The Installation Guide has
been expanded some as well.

- MPI_COMM.Join has been implemented; although like the other dynamic process
routines, it is only supported by the Sock channel.

- MPI_COMM.CONNECT and MPI_COMM.ACCEPT are now allowed to connect with remote
process to which they are already connected.

- Shared libraries can now be built (and used) on IA32 Linux with the GNU
compilers (--enable-sharedlibs=gcc), and on Solaris with the native Sun
Workshop compilers (--enable-sharedlibs=solaris). They may also work on
other operating systems with GCC, but that has not been tested. Previous
restrictions disallowing C++ and Fortran bindings when building shared
libraries have been removed.

- The dataloop and datatype contents code has been improved to address
alignment issues on all platforms.

- A bug in the datatype code, which handled zero block length cases
incorrectly, has been fixed.
- An segmentation fault in the datatype memory management, resulting from freeing memory twice, has been fixed.

- The following changes were made to the MPD process manager:

  - MPI_SPAWN_MULTIPLE now works with MPD.

  - The arguments to the 'mpiexec' command supplied by the MPD have changed. First, the -default option has been removed. Second, more flexible ways to pass environment variables have been added.

  - The commands 'mpdcheck' and 'testconfig' have been to installations using MPD. These commands test the setup of the machines on which you wish to run MPICH2 jobs. They help to identify misconfiguration, firewall issues, and other communication problems.

  - Support for MPI_APPNUM and MPI_UNIVERSE_SIZE has been added to the Simple implementation of PMI and the MPD process manager.

  - In general, error detection and recovery in MPD has improved.

  - A new process manager, gforker, is now available. Like the forker process manager, gforker spawns processes using fork(), and thus is quite useful on SMPs machines. However, unlike forker, gforker supports all of the features of a standard mpiexec, plus some. Therefore, it should be used in place of the previous forker process manager, which is now deprecated.

- The following changes were made to ROMIO:

  - The amount of duplicated ROMIO code in the close, resize, preallocate, read, write, asynchronous I/O, and sync routines has been substantially reduced.

  - A bug in flattening code, triggered by nested datatypes, has been fixed.

  - Some small memory leaks have been fixed.

  - The error handling has been abstracted allowing different MPI implementations to handle and report error conditions in their own way. Using this abstraction, the error handling routines have been made consistent with rest of MPICH2.

  - AIO support has been cleaned up and unified. It now works correctly on Linux, and is properly detected on old versions of AIX.

  - A bug in MPI_File_seek code, and underlying support code, has been fixed.

  - Support for PVFS2 has improved.

  - Several dead file systems have been removed. Others, including HFS, SFS, PIOFS, and Paragon, have been deprecated.
- MPE and CLOG have been updated to version 2.1. For more details, please see src/mpe2/README.

- New macros for memory management were added to support function local allocations (alloca), to rollback pending allocations when error conditions are detected to avoid memory leaks, and to improve the conciseness of code performing memory allocations.

- New error handling macros were added to make internal error handling code more concise.

Changes in 0.971

- Code restricted by copyrights less flexible than the one described in the COPYRIGHT file has been removed.

- Installation and User Guides have been added.

- The SMPD PMI Wire Protocol Reference Manual has been updated.

- To eliminate portability problems, common blocks in mpif.h that spanned multiple lines were broken up into multiple common blocks each described on a single line.

- A new command, mpich2version, was added to allow the user to obtain information about the MPICH2 installation. This command is currently a simple shell script. We anticipate that the mpich2version command will eventually provide additional information such as the patches applied and the date of the release.

- The following changes were made to MPD2:

  - Support was added for MPI's "singleton init", in which a single process started in the normal way (i.e., not by mpiexec or mpirun) becomes an MPI process with an MPI_COMM_WORLD of size one by calling MPI_Init. After this the process can call other MPI functions, including MPI_Comm_spawn.

  - The format for some of the arguments to mpiexec have changed, especially for passing environment variables to MPI processes.

  - In addition to miscellaneous hardening, better error checking and messages have been added.

  - The install process has been improved. In particular, configure has been updated to check for a working install program and supply it's own installation script (install.sh) if necessary.

  - A new program, mpdcheck, has been added to help diagnose machine
configurations that might be erroneous or at least confusing to
mpd.

- Runtime version checking has been added to insure that the Simple
  implementation of PMI linked into the application and the MPD
  process manager being used to run that application are compatible.

- Minor improvements have been made to mpdboot.

- Support for the (now deprecated) BNR interface has been added to
  allow MPICH1 programs to also be run via MPD2.

- Shared libraries are now supported on Linux systems using the GNU compilers
  with the caveat that C++ support must be disabled (--disable-cxx).

- The CH3 interface and device now provide a mechanism for using RDMA (remote
direct memory access) to transfer data between processes.

- Logging capabilities for MPI and internal routines have been readded. See
  the documentation in doc/logging for details.

- A "meminit" option was added to --enable-g to force all bytes associated with
  a structure or union to be initialized prior to use. This prevents programs
  like Valgrind from complaining about uninitialized accesses.

- The dist-with-version and snap targets in the top-level Makefile.sm now
  properly produce mpich2-<ver>/maint/Version instead of mpich2-<ver>/Version.
  In addition, they now properly update the VERSION variable in Makefile.sm
  without clobbering the sed line that performs the update.

- The dist and snap targets in the top-level Makefile.sm now both use the
  dist-with-version target to avoid inconsistencies.

- The following changes were made to simplemake:

  - The environment variables DEBUG, DEBUG_DIRS, and DEBUG_CONFDIR can now be
    used to control debugging output.

  - Many fixes were made to make simplemake so that it would run cleanly with
    perl -w.

  - Installation of *all* files from a directory is now possible (example,
    installing all of the man pages).

  - The clean targets now remove the cache files produced by newer versions of
    autoconf.

  - For files that are created by configure, the determination of the
    location of that configure has been improved, so that make of those
    files (e.g., make Makefile) is more likely to work. There is still
    more to do here.
- Short loops over subdirectories are now unrolled.

- The maintainerclean target has been renamed to maintainer-clean to match GNU guidelines.

- The distclean and maintainer-clean targets have been improved.

- An option was added to perform one ar command per directory instead of one per file when creating the profiling version of routines (needed only for systems that do not support weak symbols).

===============================================================================
Changes in 0.97
===============================================================================

- MPI-2 one-sided communication has been implemented in the CH3 device.

- mpigdb works as a simple parallel debugger for MPI programs started with mpd. New since MPICH1 is the ability to attach to running parallel programs. See the README in mpich2/src/pm/mpd for details.

- MPI_Type_create_darray() and MPI_Type_create_subarray() implemented including the right contents and envelope data.

- ROMIO flattening code now supports subarray and darray combiners.

- Improve scalability and performance of some ROMIO PVFS and PVFS2 routines

- An error message string parameter was added to MPID_Abort(). If the parameter is non-NULL this string will be used as the message with the abort output. Otherwise, the output message will be base on the error message associated with the mpi_errno parameter.

- MPID_Segment_init() now takes an additional boolean parameter that specifies if the segment processing code is to produce/consume homogeneous (FALSE) or heterogeneous (TRUE) data.

- The definitions of MPID_VCR and MPID_VCRT are now defined by the device.

- The semantics of MPID_Progress_{Start,Wait,End}() have changed. A typical blocking progress loop now looks like the following.

```c
if (req->cc != 0)
{
    MPID_Progress_state progress_state;

    MPID_Progress_start(&progress_state);
    while (req->cc != 0)
    {
        mpi_errno = MPID_Progress_wait(&progress_state);
        if (mpi_errno != MPI_SUCCESS)  
```
NOTE: each of these routines now takes a single parameter, a pointer to a thread local state variable.

- The CH3 device and interface have been modified to better support MPI_COMM_{SPAWN,SPAWN_MULTIPLE,CONNECT,ACCEPT,DISCONNECT}. Channels writers will notice the following. (This is still a work in progress. See the note below.)

- The introduction of a process group object (MPIDI_PG_t) and a new set of routines to manipulate that object.

- The renaming of the MPIDI_VC object to MPIDI_VC_t to make it more consistent with the naming of other objects in the device.

- The process group information in the MPIDI_VC_t moved from the channel specific portion to the device layer.

- MPIDI_CH3_Connection_terminate() was added to the CH3 interface to allow the channel to properly shutdown a connection before the device deletes all associated data structures.

- A new upcall routine, MPIDI_CH3_Handle_connection(), was added to allow the device to notify the device when a connection related event has completed. A present the only event is MPIDI_CH3_VC_EVENT_TERMINATED, which notify the device that the underlying connection associated with a VC has been properly shutdown. For every call to MPIDI_CH3_Connection_terminate() that the device makes, the channel must make a corresponding upcall to MPIDI_CH3_Handle_connection(). MPID_Finalize() will likely hang if this rule is not followed.

- MPIDI_CH3_Get_parent_port() was added to provide MPID_Init() with the port name of the the parent (spawner). This port name is used by MPID_Init() and MPID_Comm_connect() to create an intercommunicator between the parent (spawner) and child (spawnee). Eventually, MPID_Comm_spawn_multiple() will be update to perform the reverse logic; however, the logic is presently still in the sock channel.

Note: the changes noted are relatively fresh and are the beginning to a set of future changes. The goal is to minimize the amount of code required by a channel to support MPI dynamic process functionality. As such, portions of the device will change dramatically in a future release. A few more changes to the CH3 interface are also quite likely.
- MPIDI_CH3_{iRead,iWrite}() have been removed from the CH3 interface.
  MPIDI_CH3U_Handle_recv_pkt() now returns a receive request with a populated
  iovec to receive data associated with the request.
  MPIDU_CH3U_Handle_{recv,send}_req() reload the iovec in the request and
  return and set the complete argument to TRUE if more data is to read or
  written. If data transfer for the request is complete, the complete argument
  must be set to FALSE.

Changes in 0.96p2

The shm and ssm channels have been added back into the distribution.
Officially, these channels are supported only on x86 platforms using the gcc
compiler. The necessary assembly instructions to guarantee proper ordering of
memory operations are lacking for other platforms and compilers. That said, we
have seen a high success rate when testing these channels on unsupported
systems.

This patch release also includes a new unsupported channel. The scalable
shared memory, or sshm, channel is similar to the shm channel except that it
allocates shared memory communication queues only when necessary instead of
preallocating N-squared queues.

Changes in 0.96p1

This patch release fixes a problem with building MPICH2 on Microsoft Windows
platforms. It also corrects a serious bug in the poll implementation of the
Sock interface.

Changes in 0.96

The 0.96 distribution is largely a bug fix release. In addition to the many
bug fixes, major improvements have been made to the code that supports the
dynamic process management routines (MPI_Comm_{connect,accept,spawn,...}()).
Additional changes are still required to support MPI_Comm_disconnect().

We also added an experimental (and thus completely unsupported) rdma device.
The internal interface is similar to the CH3 interface except that it contains
a couple of extra routines to inform the device about data transfers using the
rendezvous protocol. The channel can use this extra information to pin memory
and perform a zero-copy transfer. If all goes well, the results will be rolled
back into the CH3 device.
Due to last minute difficulties, this release does not contain the shm or ssm channels. These channels will be included in a subsequent patch release.

Changes in 0.94

Active target one-sided communication is now available for the ch3:sock channel. This new functionality has undergone some correctness testing but has not been optimized in terms of performance. Future release will include performance enhancements, passive target communication, and availability in channels other than just ch3:sock.

The shared memory channel (ch3:shm), which performs communication using shared memory on a single machine, is now complete and has been extensively tested. At present, this channel only supports IA32 based machines (excluding the Pentium Pro which has a memory ordering bug). In addition, this channel must be compiled with gcc. Future releases with support additional architectures and compilers.

A new channel has been added that performs inter-node communication using sockets (TCP/IP) and intra-node communication using shared memory. This channel, ch3:ssm, is ideal for clusters of SMPs. Like the shared memory channel (ch3:shm), this channel only supports IA32 based machines and must be compiled with gcc. In future releases, the ch3:ssm channel will support additional architectures and compilers.

The two channels that perform commutation using shared memory, ch3:shm and ch3:ssm, now support the allocation of shared memory using both the POSIX and System V interfaces. The POSIX interface will be used if available; otherwise, the System V interface is used.

In the interest of increasing portability, many enhancements have been made to both the code and the configure scripts.

And, as always, many bugs have been fixed :-).

***** INTERFACE CHANGES ****

The parameters to MPID_Abort() have changed. MPID_Abort() now takes a pointer to communicator object, an MPI error code, and an exit code.

MPIDI_CH3_Progress() has been split into two functions:
  MPIDI_CH3_Progress_wait() and MPIDI_CH3_Progress_test().

Changes in 0.93

(continues on next page)
Version 0.93 has undergone extensive changes to provide better error reporting. Part of these changes involved modifications to the ADI3 and CH3 interfaces. The following routines now return MPI error codes:

MPID_Cancel_send()
MPID_Cancel_recv()
MPID_Progress_poke()
MPID_Progress_test()
MPID_Progress_wait()
MPIDI_CH3_Cancel_send()
MPIDI_CH3_Progress()
MPIDI_CH3_Progress_poke()
MPIDI_CH3_iRead()
MPIDI_CH3_iSend()
MPIDI_CH3_iSendv()
MPIDI_CH3_iStartmsg()
MPIDI_CH3_iStartmsgv()
MPIDI_CH3_iWrite()
MPIDI_CH3U_Handle_recv_pkt()
MPIDI_CH3U_Handle_recv_req()
MPIDI_CH3U_Handle_send_req()

******************************************************************************
Of special note are MPID_Progress_test(), MPID_Progress_wait() and MPIDI_CH3_Progress() which previously returned an integer value indicating if one or more requests had completed. They no longer return this value and instead return an MPI error code (also an integer). The implication being that while the semantics changed, the type signatures did not.
******************************************************************************

The function used to create error codes, MPIR_Err_create_code(), has also changed. It now takes additional parameters, allowing it create a stack of errors and making it possible for the reporting function to indicate in which function and on which line the error occurred. It also allows an error to be designated as fatal or recoverable. Fatal errors always result in program termination regardless of the error handler installed by the application.

A RDMA channel has been added and includes communication methods for shared memory and shmem. This is recent development and the RDMA interface is still in flux.

5.12.2 Release Notes

KNOWLEDGE ISSUES

### Fine-grained thread safety

* ch3:sock does not (and will not) support fine-grained threading.
* MPI-I0 APIs are not currently thread-safe when using fine-grained threading (--enable-thread-cs=per-object).

* ch3:nemesis:tcp fine-grained threading is still experimental and may have correctness or performance issues. Known correctness issues include dynamic process support and generalized request support.

### Lacking channel-specific features

* ch3 does not presently support communication across heterogeneous platforms (e.g., a big-endian machine communicating with a little-endian machine).

* ch3:nemesis:mx does not support dynamic processes at this time.

* Support for "external32" data representation is incomplete. This affects the MPI_Pack_external and MPI_Unpack_external routines, as well as the external data representation capabilities of ROMIO. In particular: noncontiguous user buffers could consume egregious amounts of memory in the MPI library and any types which vary in width between the native representation and the external32 representation will likely cause corruption. The following ticket contains some additional information:

  http://trac.mpich.org/projects/mpich/ticket/1754

* ch3 has known problems in some cases when threading and dynamic processes are used together on communicators of size greater than one.

### Process Managers

* Hydra has a bug related to stdin handling:

  https://trac.mpich.org/projects/mpich/ticket/1782

### Performance issues

* SMP-aware collectives do not perform as well, in select cases, as non-SMP-aware collectives, e.g. MPI_Reduce with message sizes larger than 64KiB. These can be disabled by the configure option "--disable-smpcoll".

* MPI_Irecv operations that are not explicitly completed before MPI_Finalize is called may fail to complete before MPI_Finalize returns, and thus never complete. Furthermore, any matching send operations may erroneously fail. By explicitly completed, we mean that the request associated with the operation is completed by one
of the MPI_Test or MPI_Wait routines.

5.13 MVAPICH2 Release Information

The following is reproduced essentially verbatim from files contained within the MVAPICH2 tarball downloaded from http://mvapich.cse.ohio-state.edu/

The MVAPICH2 User Guide is available at http://mvapich.cse.ohio-state.edu/support/.

MVAPICH2-2.1 introduces an algorithm to determine CPU topology on the node, and this new algorithm does not work properly for older Mellanox controllers and firmware, resulting in software threads not spreading out across a node's cores by default. This problem has been fixed in MVAPICH-2.2 and beyond.

Prior to updating to MVAPICH2-2.1, the cluster administrator should determine the potential vulnerability to this problem. For each node that contains an Infiniband controller, execute `ibstat`, and if the first output line is:

```
CA 'mthca0'
```

then that node may exhibit the problem. The cluster administrator has two choices: either avoid updating the mvapich2-sclyld packages (keeping in mind that the mvapich2-psm-sclyld packages can be updated, as those packages are only used by QLogic Infiniband controllers, which don't have the problem); or update mvapich2-sclyld, execute tests to determine if the problem exists for those Mellanox mthca nodes, and if the problem does exist, then instruct users to employ explicit CPU Mapping. See http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.1-userguide.html#x1-540006.5 for details.

MVAPICH2 Changelog
------------------
This file briefly describes the changes to the MVAPICH2 software package. The logs are arranged in the "most recent first" order.

MVAPICH2 2.3.1 (03/01/2019)

* Features and Enhancements (since 2.3):
  - Add support for JSM and Flux resource managers
  - Architecture detection, enhanced point-to-point and collective tuning for AMD Epyc system
  - Enhanced point-to-point and collective tuning for IBM POWER9 and ARM systems
  - Add support of DDN Infinite Memory Engine (IME) to ROMIO
    - Thanks to Sylvain Didelot @DDN for the patch
  - Optimize performance of MPI_Wait operation
  - Update to hwloc 1.11.11

* Bug Fixes (since 2.3):
  - Fix autogen error with Flang compiler on ARM systems
    - Thanks to Nathan Sircombe @ARM for the patch
  - Fix issues with shmem collectives on ARM architecture
    - Thanks to Pavel Shamis @ARM for the patch
  - Fix issues with MPI-3 shared memory windows for PSM-CH3 and PSM2-CH3 channel
    - Thanks to Adam Moody @LLNL for the report
  - Fix segfault in MPI_Reduce

(continues on next page)
- Thanks to Samuel Khuvis @OSC for the report
- Fix compilation issues with IBM XLC compiler
  - Thanks to Ken Raﬀenetti and Yanfei Guo @ANL for the patch
- Fix issues with MPI_Mprobe/Improbe and MPI_Mrecv/Imrecv for PSM-CH3 and
  PSM2-CH3 channel
  - Thanks to Adam Moody @LLNL for the report
- Fix compilation issues with PGI compilers for CUDA-enabled builds
- Fix potential hangs in MPI_Finalize
- Fix issues in handling very large messages with RGET protocol
- Fix issues with handling GPU buffers
- Fix issue with hardware multicast based Allreduce
- Fix build issue with TCP/IP-CH3 channel
- Fix memory leaks exposed by TotalView
  - Thanks to Adam Moody @LLNL for the report
- Fix issues with cleaning up temporary ﬁles generated in CUDA builds
- Fix compilation warnings

MVAPICH2 2.3 (07/23/2018)

* Features and Enhancements (since 2.3rc2):
  - Add point-to-point and collective tuning for IBM POWER9 CPUs
  - Enhanced collective tuning for IBM POWER8, Intel Skylake, Intel KNL, Intel
    Broadwell architectures

* Bug Fixes (since 2.3rc2):
  - Fix issues in CH3-TCP/IP channel
  - Fix build and runtime issues with CUDA support
  - Fix error when XRC and RoCE were enabled at the same time
  - Fix issue with XRC connection establishment
  - Fix for failure at finalize seen on iWARP enabled devices
  - Fix issue with MPI_IN_PLACE-based communication in MPI_Reduce and
    MPI_Reduce_scatter
  - Fix issue with allocating large number of shared memory based MPI3-RMA
    windows
  - Fix failure in mpirun_rsh with large number of nodes
  - Fix singleton initialization issue with SLURM/PMI2 and PSM/Omni-Path
    - Thanks to Adam Moody @LLNL for the report
  - Fix build failure with when enabling GPFS support in ROMIO
    - Thanks to Doug Johnson @OHTech for the report
  - Fix issues with architecture detection in PSM-CH3 and PSM2-CH3 channels
  - Fix failures with CMA read at very large message sizes
  - Fix failures with MV2_SHOW_HCA_BINDING on single-node jobs
  - Fix compilation warnings and memory leaks

MVAPICH2 2.3rc2 (04/30/2018)

* Features and Enhancements (since 2.3rc1):
  - Based on MPICH v3.2.1
  - Enhanced small message performance for MPI_Alltoallv
  - Improve performance for host-based transfers when CUDA is enabled
  - Add architecture detection for IBM POWER9 CPUs
  - Enhance architecture detection for Intel Skylake CPUs

(continues on next page)
- Enhance MPI initialization to gracefully handle RDMA_CM failures
- Improve algorithm selection of several collectives
- Enhance detection of number and IP addresses of IB devices
- Tested with CLANG v5.0.0

* Bug Fixes (since 2.3rc1):
- Fix issue in autogen step with duplicate error messages
- Fix issue with XRC connection establishment
- Fix build issue with SLES 15 and Perl 5.26.1
  - Thanks to Matias A Cabral @Intel for the report and patch
- Fix segfault when manually selecting collective algorithms
- Fix cleanup of preallocated RDMA_FP regions at RDMA_CM finalize
- Fix compilation warnings and memory leaks

MVAPICH2 2.3rc1 (02/19/2018)

* Features and Enhancements (since 2.3b):
- Enhanced performance for Allreduce, Reduce_scatter_block, Allgather, Allgatherv through new algorithms
  - Thanks to Danielle Sikich and Adam Moody @ LLNL for the patch
- Enhance support for MPI_T PVARs and CVARs
- Improved job startup time for OFA-IB-CH3, PSM-CH3, and PSM2-CH3
- Support to automatically detect IP address of IB/RoCE interfaces when RDMA_CM is enabled without relying on mv2.conf file
- Enhance HCA detection to handle cases where node has both IB and RoCE HCAs
- Automatically detect and use maximum supported MTU by the HCA
- Added logic to detect heterogeneous CPU/HFI configurations in PSM-CH3 and PSM2-CH3 channels
  - Thanks to Matias Cabral@Intel for the report
- Enhanced intra-node and inter-node tuning for PSM-CH3 and PSM2-CH3 channels
- Enhanced HFI selection logic for systems with multiple Omni-Path HFIs
- Enhanced tuning and architecture detection for OpenPOWER, Intel Skylake and Cavium ARM (ThunderX) systems
- Added 'SPREAD', 'BUNCH', and 'SCATTER' binding options for hybrid CPU binding policy
- Rename MV2_THREADS_BINDING_POLICY to MV2_HYBRID_BINDING_POLICY
- Added support for MV2_SHOW_CPU_BINDING to display number of OMP threads
- Update to hwloc version 1.11.9

* Bug Fixes (since 2.3b):
- Fix issue with RDMA_CM in multi-rail scenario
- Fix issues in nullpscw RMA test.
- Fix issue with reduce and allreduce algorithms for large message sizes
- Fix hang issue in hydra when no SLURM environment is present
  - Thanks to Vaibhav Sundriyal for the report
- Fix issue to test Fortran KIND with FFLAGS
  - Thanks to Rob Latham@mcs.anl.gov for the patch
- Fix issue in parsing environment variables
- Fix issue in displaying process to HCA binding
- Enhance CPU binding logic to handle vendor specific core mappings
- Fix compilation warnings and memory leaks
MVAPICH2 2.3b (08/10/2017)

* Features and Enhancements (since 2.3a):
  - Enhance performance of point-to-point operations for CH3-Gen2 (InfiniBand),
    CH3-PSM, and CH3-PSM2 (Omni-Path) channels
  - Improve performance for MPI-3 RMA operations
  - Introduce support for Cavium ARM (ThunderX) systems
  - Improve support for process to core mapping on many-core systems
    - New environment variable MV2_THREADS_BINDING_POLICY for
      multi-threaded MPI and MPI-OpenMP applications
    - Support `linear' and `compact' placement of threads
    - Warn user if oversubcription of core is detected
  - Improve launch time for large-scale jobs with mpirun_rsh
  - Add support for non-blocking Allreduce using Mellanox SHARP
  - Efficient support for different Intel Knight's Landing (KNL) models
  - Improve performance for Intra- and Inter-node communication for OpenPOWER architecture
  - Improve support for large processes per node and hugepages on SMP systems
  - Enhance collective tuning for Intel Knight's Landing and Intel Omni-Path based systems
  - Enhance collective tuning for Bebop@ANL, Bridges@PSC, and Stampede2@TACC systems
  - Enhance large message intra-node performance with CH3-IB-Gen2 channel on Intel Knight's Landing
  - Enhance support for MPI_T PVARs and CVARs

* Bug Fixes (since 2.3a):
  - Fix issue with bcast algorithm selection
  - Fix issue with large message transfers using CMA
  - Fix issue in Scatter and Gather with large messages
  - Fix tuning tables for various collectives
  - Fix issue with launching single-process MPI jobs
  - Fix compilation error in the CH3-TCP/IP channel
    - Thanks to Isaac Carroll@Lightfleet for the patch
  - Fix issue with memory barrier instructions on ARM
    - Thanks to Pavel (Pasha) Shamis@ARM for reporting the issue
  - Fix compilation warnings and memory leaks

MVAPICH2 2.3a (03/29/2017)

* Features and Enhancements (since 2.2):
  - Based on and ABI compatible with MPICH 3.2
  - Support collective offload using Mellanox's SHArP for Allreduce
    - Enhance tuning framework for Allreduce using SHArP
  - Introduce capability to run MPI jobs across multiple InfiniBand subnets
  - Introduce basic support for executing MPI jobs in Singularity
  - Enhance collective tuning for Intel Knight's Landing and Intel Omni-path
  - Enhance process mapping support for multi-threaded MPI applications
    - Introduce MV2_CPU_BINDING_POLICY=hybrid
    - Introduce MV2_THREADS_PER_PROCESS
  - On-demand connection management for PSM-CH3 and PSM2-CH3 channels
  - Enhance PSM-CH3 and PSM2-CH3 job startup to use non-blocking PMI calls

(continues on next page)
- Enhance debugging support for PSM-CH3 and PSM2-CH3 channels
- Improve performance of architecture detection
- Introduce run time parameter MV2_SHOW_HCA_BINDING to show process to HCA bindings
- Enhance MV2_SHOW_CPU_BINDING to enable display of CPU bindings on all nodes
- Deprecate OFA-IB-Nemesis channel
- Update to hwloc version 1.11.6

* Bug Fixes (since 2.2):
  - Fix issue with ring startup in multi-rail systems
  - Fix startup issue with SLURM and PMI-1
    - Thanks to Manuel Rodriguez for the report
  - Fix startup issue caused by fix for bash `shellshock' bug
  - Fix issue with very large messages in PSM
  - Fix issue with singleton jobs and PMI-2
    - Thanks to Adam T. Moody@LLNL for the report
  - Fix incorrect reporting of non-existing files with Luster ADIO
    - Thanks to Wei Kang@NWU for the report
  - Fix hang in MPI_Probe
    - Thanks to John Westlund@Intel for the report
  - Fix issue while setting affinity with Torque Cgroups
    - Thanks to Doug Johnson@OSC for the report
  - Fix runtime errors observed when running MVAPICH2 on aarch64 platforms
    - Thanks to Sreenidhi Bharathkar Ramesh@Broadcom for posting the original patch
    - Thanks to Michal Schmidt@RedHat for reposting it
  - Fix failure in mv2_show_cpu_affinity with affinity disabled
    - Thanks to Carlos Rosales-Fernandez@TACC for the report
  - Fix mpirun_rsh error when running short-lived non-MPI jobs
    - Thanks to Kevin Manalo@OSC for the report
  - Fix comment and spelling mistake
    - Thanks to Maksym Planeta for the report
  - Ignore cpusets and cgroups that may have been set by resource manager
    - Thanks to Adam T. Moody@LLNL for the report and the patch
  - Fix reduce tuning table entry for 2ppn 2node
  - Fix compilation issues due to inline keyword with GCC 5 and newer
  - Fix compilation warnings and memory leaks

MVAPICH2 2.2 (09/07/2016)

* Features and Enhancements (since 2.2rc2):
  - Single node collective tuning for Bridges@PSC, Stampede@TACC and other architectures
  - Enable PSM builds when both PSM and PSM2 libraries are present
    - Thanks to Adam T. Moody@LLNL for the report and patch
  - Add support for HCAs that return result of atomics in big endian notation
  - Establish loopback connections by default if HCA supports atomics

* Bug Fixes (since 2.2rc2):
  - Fix minor error in use of communicator object in collectives
  - Fix missing u_int64_t declaration with PGI compilers
    - Thanks to Adam T. Moody@LLNL for the report and patch
  - Fix memory leak in RMA rendezvous code path

(continues on next page)
- Thanks to Min Si@ANL for the report and patch

MVAPICH2 2.2rc2 (08/08/2016)

* Features and Enhancements (since 2.2rc1):
  - Enhanced performance for MPI_Comm_split through new bitonic algorithm
    - Thanks to Adam T. Moody@LLNL for the patch
  - Enable graceful fallback to Shared Memory if LIMIC2 or CMA transfer fails
  - Enable support for multiple MPI initializations
  - Unify process affinity support in Gen2, PSM and PSM2 channels
  - Remove verbs dependency when building the PSM and PSM2 channels
  - Allow processes to request MPI_THREAD_MULTIPLE when socket or NUMA node level affinity is specified
  - Point-to-point and collective performance optimization for Intel Knights Landing
  - Automatic detection and tuning for InfiniBand EDR HCAs
  - Warn user to reconfigure library if rank type is not large enough to represent all ranks in job
  - Collective tuning for Opal@LLNL, Bridges@PSC, and Stampede-1.5@TACC
  - Tuning and architecture detection for Intel Broadwell processors
  - Add ability to avoid using --enable-new-dtags with ld
    - Thanks to Adam T. Moody@LLNL for the suggestion
  - Add LIBVTMPICH specific CFLAGS and LDFLAGS
    - Thanks to Adam T. Moody@LLNL for the suggestion

* Bug Fixes (since 2.2rc1):
  - Disable optimization that removes use of calloc in ptmalloc hook detection code
    - Thanks to Karl W. Schulz@Intel
  - Fix weak alias typos (allows successful compilation with CLANG compiler)
    - Thanks to Min Dong@Old Dominion University for the patch
  - Fix issues in PSM large message gather operations
    - Thanks to Adam T. Moody@LLNL for the report
  - Enhance error checking in collective tuning code
    - Thanks to Jan Bierbaum@Technical University of Dresden for the patch
  - Fix issues with UD based communication in RoCE mode
  - Fix issues with PMI2 support in singleton mode
  - Fix default binding bug in hydra launcher
  - Fix issues with Checkpoint Restart when launched with mpirun_rsh
  - Fix fortran binding issues with Intel 2016 compilers
  - Fix issues with socket/NUMA node level binding
  - Disable atomics when using Connect-IB with RDMA_CM
  - Fix hang in MPI_Finalize when using hybrid channel
  - Fix memory leaks

MVAPICH2 2.2rc1 (03/29/2016)

* Features and Enhancements (since 2.2b):
  - Support for OpenPower architecture
  - Optimized inter-node and intra-node communication
  - Support for Intel Omni-Path architecture
    - Thanks to Intel for contributing the patch

(continues on next page)
- Introduction of a new PSM2 channel for Omni-Path
- Support for RoCEv2
- Architecture detection for PSC Bridges system with Omni-Path
- Enhanced startup performance and reduced memory footprint for storing InfiniBand end-point information with SLURM
  - Support for shared memory based PMI operations
  - Availability of an updated patch from the MVAPICH project website with this support for SLURM installations
- Optimized pt-to-pt and collective tuning for Chameleon InfiniBand systems at TACC/UoC
- Enable affinity by default for TrueScale(PSM) and Omni-Path(PSM2) channels
- Enhanced tuning for shared-memory based MPI_Bcast
- Enhanced debugging support and error messages
- Update to hwloc version 1.11.2

* Bug Fixes (since 2.2b):
  - Fix issue in some of the internal algorithms used for MPI_Bcast, MPI_Alltoall and MPI_Reduce
  - Fix hang in one of the internal algorithms used for MPI_Scatter
    - Thanks to Ivan Raikov@Stanford for reporting this issue
  - Fix issue with rdma_connect operation
  - Fix issue with Dynamic Process Management feature
  - Fix issue with de-allocating InfiniBand resources in blocking mode
  - Fix build errors caused due to improper compile time guards
    - Thanks to Adam Moody@LLNL for the report
  - Fix finalize hang when running in hybrid or UD-only mode
    - Thanks to Jerome Vienne@TACC for reporting this issue
  - Fix issue in MPI_Win_flush operation
    - Thanks to Nenad Vukicevic for reporting this issue
  - Fix out of memory issues with non-blocking collectives code
    - Thanks to Phanisri Pradeep Pratapa and Fang Liu@GaTech for reporting this issue
  - Fix fall-through bug in external32 pack
    - Thanks to Adam Moody@LLNL for the report and patch
  - Fix issue with on-demand connection establishment and blocking mode
    - Thanks to Maksym Planeta@TU Dresden for the report
  - Fix memory leaks in hardware multicast based broadcast code
  - Fix memory leaks in TrueScale(PSM) channel
  - Fix compilation warnings

MVAPICH2 2.2b (11/12/2015)

* Features and Enhancements (since 2.2a):
  - Enhanced performance for small messages
  - Enhanced startup performance with SLURM
    - Support for PMIX_Iallgather and PMIX_Ifence
  - Support to enable affinity with asynchronous progress thread
  - Enhanced support for MPIT based performance variables
  - Tuned VBUF size for performance
  - Improved startup performance for QLogic PSM-CH3 channel
    - Thanks to Maksym Planeta@TU Dresden for the patch
* Bug Fixes (since 2.2a):
  - Fix issue with MPI_Get_count in QLogic PSM-CH3 channel with very large messages (>2GB)
  - Fix issues with shared memory collectives and checkpoint-restart
  - Fix hang with checkpoint-restart
  - Fix issue with unlinking shared memory files
  - Fix memory leak with MPIT
  - Fix minor typos and usage of inline and static keywords
    - Thanks to Maksym Planeta@TU Dresden for the patch and suggestions
  - Fix missing MPIDI_FUNC_EXIT
    - Thanks to Maksym Planeta@TU Dresden for the patch
  - Remove unused code
    - Thanks to Maksym Planeta@TU Dresden for the patch
  - Continue with warning if user asks to enable XRC when the system does not support XRC

MVAPICH2 2.2a (08/17/2015)

* Features and Enhancements (since 2.1 GA):
  - Based on MPICH 3.1.4
  - Support for backing on-demand UD CM information with shared memory for minimizing memory footprint
  - Reorganized HCA-aware process mapping
  - Dynamic identification of maximum read/atomic operations supported by HCA
  - Enabling support for intra-node communications in RoCE mode without shared memory
  - Updated to hwloc 1.11.0
  - Updated to sm_20 kernel optimizations for MPI Datatypes
  - Automatic detection and tuning for 24-core Haswell architecture

* Bug Fixes (since 2.1 GA):
  - Fix for error with multi-vbuf design for GPU based communication
  - Fix bugs with hybrid UD/RC/XRC communications
  - Fix for MPICH putfence/getfence for large messages
  - Fix for error in collective tuning framework
  - Fix validation failure with Alltoall with IN_PLACE option
    - Thanks for Mahidhar Tatineni @SDSC for the report
  - Fix bug with MPI_Reduce with IN_PLACE option
    - Thanks to Markus Geimer for the report
  - Fix for compilation failures with multicast disabled
    - Thanks to Devesh Sharma @Emulex for the report
  - Fix bug with MPI_Bcast
  - Fix IPC selection for shared GPU mode systems
  - Fix for build time warnings and memory leaks
  - Fix issues with Dynamic Process Management
    - Thanks to Neil Spruit for the report
  - Fix bug in architecture detection code
    - Thanks to Adam Moody @LLNL for the report

(continues on next page)
MVAPICH2-2.1 (04/03/2015)

* Features and Enhancements (since 2.1rc2):
  - Tuning for EDR adapters
  - Optimization of collectives for SDSC Comet system

* Bug-Fixes (since 2.1rc2):
  - Relocate reading environment variables in PSM
    - Thanks to Adam Moody@LLNL for the suggestion
  - Fix issue with automatic process mapping
  - Fix issue with checkpoint restart when full path is not given
  - Fix issue with Dynamic Process Management
  - Fix issue in CUDA IPC code path
  - Fix corner case in CMA runtime detection

MVAPICH2-2.1rc2 (03/12/2015)

* Features and Enhancements (since 2.1rc1):
  - Based on MPICH-3.1.4
  - Enhanced startup performance with mpirun_rsh
  - Checkpoint-Restart Support with DMTCP (Distributed MultiThreaded CheckPointing)
    - Thanks to the DMTCP project team (http://dmtcp.sourceforge.net/)
  - Support for handling very large messages in RMA
  - Optimize size of buffer requested for control messages in large message transfer
  - Enhanced automatic detection of atomic support
  - Optimized collectives (bcast, reduce, and allreduce) for 4K processes
  - Introduce support to sleep for user specified period before aborting
    - Thanks to Adam Moody@LLNL for the suggestion
  - Disable PSM from setting CPU affinity
    - Thanks to Adam Moody@LLNL for providing the patch
  - Install PSM error handler to print more verbose error messages
    - Thanks to Adam Moody@LLNL for providing the patch
  - Introduce retry mechanism to perform psm_ep_open in PSM channel
    - Thanks to Adam Moody@LLNL for providing the patch

* Bug-Fixes (since 2.1rc1):
  - Fix failures with shared memory collectives with checkpoint-restart
  - Fix failures with checkpoint-restart when using internal communication buffers of different size
  - Fix undeclared variable error when --disable-cxx is specified with configure
    - Thanks to Chris Green from FANL for the patch
  - Fix segfault seen during connect/accept with dynamic processes
    - Thanks to Neil Spruit for the fix
  - Fix errors with large messages pack/unpack operations in PSM channel
  - Fix for bcast collective tuning
  - Fix assertion errors in one-sided put operations in PSM channel
  - Fix issue with code getting stuck in infinite loop inside ptmalloc
    - Thanks to Adam Moody@LLNL for the suggested changes
  - Fix assertion error in shared memory large message transfers
- Thanks to Adam Moody@LLNL for reporting the issue
- Fix compilation warnings

MVAPICH2-2.1rc1 (12/18/2014)

* Features and Enhancements (since 2.1a):
  - Based on MPICH-3.1.3
  - Flexibility to use internal communication buffers of different size for improved performance and memory footprint
  - Improve communication performance by removing locks from critical path
  - Enhanced communication performance for small/medium message sizes
  - Support for linking Intel Trace Analyzer and Collector
  - Increase the number of connect retry attempts with RDMA CM
  - Automatic detection and tuning for Haswell architecture

* Bug-Fixes (since 2.1a):
  - Fix automatic detection of support for atomics
  - Fix issue with void pointer arithmetic with PGI
  - Fix deadlock in ctxidup MPICH test in PSM channel
  - Fix compile warnings

MVAPICH2-2.1a (09/21/2014)

* Features and Enhancements (since 2.0):
  - Based on MPICH-3.1.2
  - Support for PMI-2 based startup with SLURM
  - Enhanced startup performance for Gen2/UD-Hybrid channel
  - GPU support for MPI_Scan and MPI_Exscan collective operations
  - Optimize creation of 2-level communicator
  - Collective optimization for PSM-CH3 channel
  - Tuning for IvyBridge architecture
  - Add -export-all option to mpirun_rsh
  - Support for additional MPI-T performance variables (PVARs) in the CH3 channel
  - Link with libstdc++ when building with GPU support (required by CUDA 6.5)

* Bug-Fixes (since 2.0):
  - Fix error in large message (>2GB) transfers in CMA code path
  - Fix memory leaks in OFA-IB-CH3 and OFA-IB-Nemesis channels
  - Fix issues with optimizations for broadcast and reduce collectives
  - Fix hang at finalize with Gen2-Hybrid/UD channel
  - Fix issues for collectives with non power-of-two process counts
  - Thanks to Evren Yurtesen for identifying the issue
  - Make ring startup use HCA selected by user
  - Increase counter length for shared-memory collectives

MVAPICH2-2.0 (06/20/2014)

* Features and Enhancements (since 2.0rc2):
  - Consider CMA in collective tuning framework

(continues on next page)
* Bug-Fixes (since 2.0rc2):
  - Fix bug when disabling registration cache
  - Fix shared memory window bug when shared memory collectives are disabled
  - Fix mpirun_rsh bug when running mpmd programs with no arguments

MVAPICH2-2.0rc2 (05/25/2014)

* Features and Enhancements (since 2.0rc1):
  - CMA support is now enabled by default
  - Optimization of collectives with CMA support
  - RMA optimizations for shared memory and atomic operations
  - Tuning RGET and Atomics operations
  - Tuning RDMA FP-based communication
  - MPI-T support for additional performance and control variables
  - The --enable-mpit-pvars=yes configuration option will now enable only MVAPICH2 specific variables
  - Large message transfer support for PSM interface
  - Optimization of collectives for PSM interface
  - Updated to hwloc v1.9

* Bug-Fixes (since 2.0rc1):
  - Fix multicast hang when there is a single process on one node and more than one process on other nodes
  - Fix non-power-of-two usage of scatter-doubling-allgather algorithm
  - Fix for bcastzero type hang during finalize
  - Enhanced handling of failures in RDMA_CM based connection establishment
  - Fix for a hang in finalize when using RDMA_CM
  - Finish receive request when RDMA READ completes in RGET protocol
  - Always use direct RDMA when flush is used
  - Fix compilation error with --enable-g=all in PSM interface
  - Fix warnings and memory leaks

MVAPICH2-2.0rc1 (03/24/2014)

* Features and Enhancements (since 2.0b):
  - Based on MPICH-3.1
  - Enhanced direct RDMA based designs for MPI_Put and MPI_Get operations in OFA-IB-CH3 channel
  - Optimized communication when using MPI_Win_allocate for OFA-IB-CH3 channel
  - MPI-3 RMA support for CH3-PSM channel
  - Multi-rail support for UD-Hybrid channel
  - Optimized and tuned blocking and non-blocking collectives for OFA-IB-CH3, OFA-IB-Nemesis, and CH3-PSM channels
  - Improved hierarchical job startup performance
  - Optimized sub-array data-type processing for GPU-to-GPU communication
  - Tuning for Mellanox Connect-IB adapters
  - Updated hwloc to version 1.8
  - Added options to specify CUDA library paths
  - Deprecation of uDAPL-CH3 channel
* Bug-Fixes (since 2.0b):
  - Fix issues related to MPI-3 RMA locks
  - Fix an issue related to MPI-3 dynamic window
  - Fix issues related to MPI_Win_allocate backed by shared memory
  - Fix issues related to large message transfers for OFA-IB-CH3 and OFA-IB-Nemesis channels
  - Fix warning in job launch, when using DPM
  - Fix an issue related to MPI atomic operations on HCAs without atomics support
  - Fixed an issue related to selection of compiler. (We prefer the GNU, Intel, PGI, and Ekopath compilers in that order).
    - Thanks to Uday R Bondhugula from IISc for the report
  - Fix an issue in message coalescing
  - Prevent printing out inter-node runtime parameters for pure intra-node runs
    - Thanks to Jerome Vienne from TACC for the report
  - Fix an issue related to ordering of messages for GPU-to-GPU transfers
  - Fix a few memory leaks and warnings

MVAPICH2-2.0b (11/08/2013)

* Features and Enhancements (since 2.0a):
  - Based on MPICH-3.1b
  - Multi-rail support for GPU communication
  - Non-blocking streams in asynchronous CUDA transfers for better overlap
  - Initialize GPU resources only when used by MPI transfer
  - Extended support for MPI-3 RMA in OFA-IB-CH3, OFA-IWARP-CH3, and OFA-RoCE-CH3
  - Additional MPIT counters and performance variables
  - Updated compiler wrappers to remove application dependency on network and other extra libraries
    - Thanks to Adam Moody from LLNL for the suggestion
  - Capability to checkpoint CH3 channel using the Hydra process manager
  - Optimized support for broadcast, reduce and other collectives
  - Tuning for IvyBridge architecture
  - Improved launch time for large-scale mpirun_rsh jobs
  - Introduced retry mechanism in mpirun_rsh for socket binding
  - Updated hwloc to version 1.7.2

* Bug-Fixes (since 2.0a):
  - Consider list provided by MV2_IBA_HCA when scanning device list
  - Fix issues in Nemesis interface with --with-ch3-rank-bits=32
  - Better cleanup of XRC files in corner cases
  - Initialize using better defaults for ibv_modify_qp (initial ring)
  - Add unconditional check and addition of pthread library
  - MPI_Get_library_version updated with proper MVAPICH2 branding
    - Thanks to Jerome Vienne from the TACC for the report

MVAPICH2-2.0a (08/24/2013)

* Features and Enhancements (since 1.9):
  - Based on MPICH-3.0.4

(continues on next page)
- Dynamic CUDA initialization. Support GPU device selection after MPI_Init
- Support for running on heterogeneous clusters with GPU and non-GPU nodes
- Supporting MPI-3 RMA atomic operations and flush operations with CH3-Gen2 interface
- Exposing internal performance variables to MPI-3 Tools information interface (MPIT)
- Enhanced MPI_Bcast performance
- Enhanced performance for large message MPI_Scatter and MPI_Gather
- Enhanced intra-node SMP performance
- Tuned SMP eager threshold parameters
- Reduced memory footprint
- Improved job-startup performance
- Warn and continue when ptmalloc fails to initialize
- Enable hierarchical SSH-based startup with Checkpoint-Restart
- Enable the use of Hydra launcher with Checkpoint-Restart

* Bug-Fixes (since 1.9):
  - Fix data validation issue with MPI_Bcast
    - Thanks to Claudio J. Margulis from University of Iowa for the report
  - Fix buffer alignment for large message shared memory transfers
  - Fix a bug in One-Sided shared memory backed windows
  - Fix a flow-control bug in UD transport
    - Thanks to Benjamin M. Auer from NASA for the report
  - Fix bugs with MPI-3 RMA in Nemesis IB interface
  - Fix issue with very large message (>2GB bytes) MPI_Bcast
    - Thanks to Lu Qiyue for the report
  - Handle case where $HOME is not set during search for MV2 user config file
    - Thanks to Adam Moody from LLNL for the patch
  - Fix a hang in connection setup with RDMA-CM

MVAPICH2-1.9 (05/06/2013)

* Features and Enhancements (since 1.9rc1):
  - Updated to hwloc v1.7
  - Tuned Reduce, AllReduce, Scatter, Reduce-Scatter and Allgatherv Collectives

* Bug-Fixes (since 1.9rc1):
  - Fix cuda context issue with async progress thread
    - Thanks to Osuna Escamilla Carlos from env.ethz.ch for the report
  - Overwrite pre-existing PSM environment variables
    - Thanks to Adam Moody from LLNL for the patch
  - Fix several warnings
    - Thanks to Adam Moody from LLNL for some of the patches

MVAPICH2-1.9RC1 (04/16/2013)

* Features and Enhancements (since 1.9b):
  - Based on MPICH-3.0.3
  - Updated SCR to version 1.1.8
  - Install utility scripts included with SCR
  - Support for automatic detection of path to utilities used by mpirun_rsh
during configuration
- Utilities supported: rsh, ssh, xterm, totalview
- Support for launching jobs on heterogeneous networks with mpirun_rsh
- Tuned Bcast, Reduce, Scatter Collectives
- Tuned MPI performance on Kepler GPUs
- Introduced MV2_RDMA_CM_CONF_FILE_PATH parameter which specifies path to mv2.conf

* Bug-Fixes (since 1.9b):
  - Fix autoconf issue with LiMIC2 source-code
    - Thanks to Doug Johnson from OH-TECH for the report
  - Fix build errors with --enable-thread-cs=per-object and --enable-refcount=lock-free
    - Thanks to Marcin Zalewski from Indiana University for the report
  - Fix MPI_Scatter failure with MPI_IN_PLACE
    - Thanks to Mellanox for the report
  - Fix MPI_Scatter failure with cyclic host files
  - Fix deadlocks in PSM interface for multi-threaded jobs
    - Thanks to Marcin Zalewski from Indiana University for the report
  - Fix MPI_Bcast failures in SCALAPACK
    - Thanks to Jerome Vienne from TACC for the report
  - Fix build errors with newer Ekopath compiler
  - Fix a bug with shmem collectives in PSM interface
  - Fix memory corruption when more entries specified in mv2.conf than the requested number of rails
    - Thanks to Akihiro Nomura from Tokyo Institute of Technology for the report
  - Fix memory corruption with CR configuration in Nemesis interface

MVAPICH2-1.9b (02/28/2013)

* Features and Enhancements (since 1.9a2):
  - Based on MPICH-3.0.2
    - Support for all MPI-3 features
  - Support for single copy intra-node communication using Linux supported CMA (Cross Memory Attach)
    - Provides flexibility for intra-node communication: shared memory, LiMIC2, and CMA
  - Checkpoint/Restart using LLNL's Scalable Checkpoint/Restart Library (SCR)
    - Support for application-level checkpointing
    - Support for hierarchical system-level checkpointing
  - Improved job startup time
    - Provided a new runtime variable MV2_HOMOGENEOUS_CLUSTER for optimized startup on homogeneous clusters
  - New version of LiMIC2 (v0.5.6)
    - Provides support for unlocked ioctl calls
  - Tuned Reduce, Allgather, Reduce_Scatter, Allgatherv collectives
  - Introduced option to export environment variables automatically with mpirun_rsh
  - Updated to HWLOC v1.6.1
  - Provided option to use CUDA library call instead of CUDA driver to check buffer pointer type
- Thanks to Christian Robert from Sandia for the suggestion
- Improved debug messages and error reporting

* Bug-Fixes (since 1.9a2):
- Fix page fault with memory access violation with LiMIC2 exposed by newer Linux kernels
  - Thanks to Karl Schulz from TACC for the report
- Fix a failure when lazy memory registration is disabled and CUDA is enabled
  - Thanks to Jens Glaser from University of Minnesota for the report
- Fix an issue with variable initialization related to DPM support
- Rename a few internal variables to avoid name conflicts with external applications
  - Thanks to Adam Moody from LLNL for the report
- Check for libattr during configuration when Checkpoint/Restart and Process Migration are requested
  - Thanks to John Gilmore from Vastech for the report
- Fix build issue with --disable-cxx
- Set intra-node eager threshold correctly when configured with LiMIC2
- Fix an issue with MV2_DEFAULT_PKEY in partitioned InfiniBand network
  - Thanks to Jesper Larsen from FC2O for the report
- Improve makefile rules to use automake macros
  - Thanks to Carmelo Ponti from CSCS for the report
- Fix configure error with automake conditionals
  - Thanks to Evren Yurtesen from Abo Akademi for the report
- Fix a few memory leaks and warnings
- Properly cleanup shared memory files (used by XRC) when applications fail

MVAPICH2-1.9a2 (11/08/2012)

* Features and Enhancements (since 1.9a):
- Based on MPICH2-1.5
- Initial support for MPI-3:
  (Available for all interfaces: OFA-IB-CH3, OFA-IWARP-CH3, OFA-RoCE-CH3, uDAPL-CH3, OFA-IB-Nemesis, PSM-CH3)
  - Nonblocking collective functions available as "MPIX_" functions (e.g., "MPIX_Ibcast")
  - Neighborhood collective routines available as "MPIX_" functions (e.g., "MPIX_Neighbor_allgatherv")
  - MPI_Comm_split_type function available as an "MPIX_" function
  - Support for MPIX_Type_create_hindexed_block
  - Nonblocking communicator duplication routine MPIX_Comm_idup (will only work for single-threaded programs)
  - MPIX_Comm_create_group support
  - Support for matched probe functionality (e.g., MPIX_Mprobe, MPIX_Improbe, MPIX_Mrecv, and MPIX_Imrecv), (Not Available for PSM)
  - Support for "Const" (disabled by default)
- Efficient vector, hindexed datatype processing on GPU buffers
- Tuned alltoall, Scatter and Allreduce collectives
- Support for Mellanox Connect-IB HCA
- Adaptive number of registration cache entries based on job size

(continues on next page)
- Revamped Build system:
  - Uses automake instead of simplemake,
  - Allows for parallel builds ("make -j8" and similar)

* Bug-Fixes (since 1.9a):
  - CPU frequency mismatch warning shown under debug
  - Fix issue with MPI_IN_PLACE buffers with CUDA
  - Fix ptmalloc initialization issue due to compiler optimization
    - Thanks to Kyle Sheumaker from ACT for the report
  - Adjustable MAX_NUM_PORTS at build time to support more than two ports
  - Fix issue with MPI_Allreduce with MPI_IN_PLACE send buffer
  - Fix memleak in MPI_Cancel with PSM interface
    - Thanks to Andrew Friedley from LLNL for the report

MVAPICH2-1.9a (09/07/2012)

* Features and Enhancements (since 1.8):
  - Support for InfiniBand hardware UD-multicast
  - UD-multicast-based designs for collectives
    (Bcast, Allreduce and Scatter)
  - Enhanced Bcast and Reduce collectives with pt-to-pt communication
  - LiMIC-based design for Gather collective
  - Improved performance for shared-memory-aware collectives
  - Improved intra-node communication performance with GPU buffers
    using pipelined design
  - Improved inter-node communication performance with GPU buffers
    with non-blocking CUDA copies
  - Improved small message communication performance with
    GPU buffers using CUDA IPC design
  - Improved automatic GPU device selection and CUDA context management
  - Optimal communication channel selection for different
    GPU communication modes (DD, DH and HD) in different
    configurations (intra-IOH and inter-IOH)
  - Removed libibumad dependency for building the library
  - Option for selecting non-default gid-index in a loss-less
    fabric setup in RoCE mode
  - Option to disable signal handler setup
  - Tuned thresholds for various architectures
  - Set DAPL-2.0 as the default version for the uDAPL interface
  - Updated to hwloc v1.5
  - Option to use IP address as a fallback if hostname cannot be resolved
  - Improved error reporting

* Bug-Fixes (since 1.8):
  - Fix issue in intra-node knomial bcast
  - Handle gethostbyname return values gracefully
  - Fix corner case issue in two-level gather code path
  - Fix bug in CUDA events/streams pool management
  - Fix ptmalloc initialization issue when MALLOC_CHECK_ is
    defined in the environment
    - Thanks to Mehmet Belgin from Georgia Institute of
Technology for the report
- Fix memory corruption and handle heterogeneous architectures in gather collective
- Fix issue in detecting the correct HCA type
- Fix issue in ring start-up to select correct HCA when MV2_IBA_HCA is specified
- Fix SEGFAULT in MPI_Finalize when IB loop-back is used
- Fix issue in ring start-up to select correct HCA when MV2_IBA_HCA is specified
- Thanks to M Xie for the report
- Fix hang in MPI_Finalize with Nemesis interface when ptmalloc initialization fails
  - Thanks to Carson Holt from OICR for the report
- Fix memory corruption in shared memory communication
  - Thanks to Craig Tierney from NOAA for the report and testing the patch
- Fix issue in IB ring start-up selection with mpiexec.hydra
- Fix issue in selecting CUDA run-time variables when running on single node in SMP only mode
- Fix few memory leaks and warnings

MVAPICH2-1.8 (04/30/2012)

* Features and Enhancements (since 1.8rc1):
  - Introduced a unified run time parameter MV2_USE_ONLY_UD to enable UD only mode
  - Enhanced designs for Alltoall and Allgather collective communication from GPU device buffers
  - Tuned collective communication from GPU device buffers
  - Tuned Gather collective
  - Introduced a run time parameter MV2_SHOW_CPU_BINDING to show current CPU bindings
  - Updated to hwloc v1.4.1
  - Remove dependency on LEX and YACC

* Bug-Fixes (since 1.8rc1):
  - Fix hang with multiple GPU configuration
    - Thanks to Jens Glaser from University of Minnesota for the report
  - Fix buffer alignment issues to improve intra-node performance
  - Fix a DPM multispawn behavior
  - Enhanced error reporting in DPM functionality
  - Quote environment variables in job startup to protect from shell
  - Fix hang when LIMIC is enabled
  - Fix hang in environments with heterogeneous HCAs
  - Fix issue when using multiple HCA ports in RDMA_CM mode
    - Thanks to Steve Wise from Open Grid Computing for the report
  - Fix hang during MPI_Finalize in Nemesis IB netmod
  - Fix for a start-up issue in Nemesis with heterogeneous architectures
  - Fix few memory leaks and warnings

MVAPICH2-1.8rc1 (03/22/2012)
* Features & Enhancements (since 1.8a2):
  - New design for intra-node communication from GPU Device buffers using CUDA IPC for better performance and correctness
    - Thanks to Joel Scherpelz from NVIDIA for his suggestions
  - Enabled shared memory communication for host transfers when CUDA is enabled
  - Optimized and tuned collectives for GPU device buffers
  - Enhanced pipelined inter-node device transfers
  - Enhanced shared memory design for GPU device transfers for large messages
  - Enhanced support for CPU binding with socket and numanode level granularity
  - Support suspend/resume functionality with mpirun_rsh
  - Exporting local rank, local size, global rank and global size through environment variables (both mpirun_rsh and hydra)
  - Update to hwloc v1.4
  - Checkpoint-Restart support in OFA-IB-Nemesis interface
  - Enabling run-through stabilization support to handle process failures in OFA-IB-Nemesis interface
  - Enhancing OFA-IB-Nemesis interface to handle IB errors gracefully
  - Performance tuning on various architecture clusters
  - Support for Mellanox IB FDR adapter

* Bug-Fixes (since 1.8a2):
  - Fix a hang issue on InfiniHost SDR/DDR cards
    - Thanks to Nirmal Seenu from Fermilab for the report
  - Fix an issue with runtime parameter MV2_USE_COALESCE usage
  - Fix an issue with LiMIC2 when CUDA is enabled
  - Fix an issue with intra-node communication using datatypes and GPU device buffers
  - Fix an issue with Dynamic Process Management when launching processes on multiple nodes
    - Thanks to Rutger Hofman from VU Amsterdam for the report
  - Fix build issue in hwloc source with mcmmodel=medium flags
    - Thanks to Nirmal Seenu from Fermilab for the report
  - Fix a build issue in hwloc with --disable-shared or --disable-static options
  - Use portable stdout and stderr redirection
    - Thanks to Dr. Axel Philipp from "MTU" Aero Engines for the patch
  - Fix a build issue with PGI 12.2
    - Thanks to Thomas Rothrock from U.S. Army SMDC for the patch
  - Fix an issue with send message queue in OFA-IB-Nemesis interface
  - Fix a process cleanup issue in Hydra when MPI_ABORT is called (upstream MPICH2 patch)
  - Fix an issue with non-contiguous datatypes in MPI_Gather
  - Fix a few memory leaks and warnings

MVAPICH2-1.8a2 (02/02/2012)

* Features and Enhancements (since 1.8a1p1):
  - Support for collective communication from GPU buffers
  - Non-contiguous datatype support in point-to-point and collective communication from GPU buffers
- Efficient GPU-GPU transfers within a node using CUDA IPC (for CUDA 4.1)
- Alternate synchronization mechanism using CUDA Events for pipelined device data transfers
- Exporting processes local rank in a node through environment variable
- Adjust shared-memory communication block size at runtime
- Enable XRC by default at configure time
- New shared memory design for enhanced intra-node small message performance
- Tuned inter-node and intra-node performance on different cluster architectures
- Update to hwloc v1.3.1
- Support for fallback to R3 rendezvous protocol if RGET fails
- SLURM integration with mpiexec.mpirun_rsh to use SLURM allocated hosts without specifying a hostfile
- Support added to automatically use PBS_NODEFILE in Torque and PBS environments
- Enable signal-triggered (SIGUSR2) migration

* Bug Fixes (since 1.8alpl):
  - Set process affinity independently of SMP enable/disable to control the affinity in loopback mode
  - Report error and exit if user requests MV2_USE_CUDA=1 in non-cuda configuration
  - Fix for data validation error with GPU buffers
  - Updated.WRAPPER_CPPFLAGS when using --with-cuda. Users should not have to explicitly specify CPPFLAGS or LDFLAGS to build applications
  - Fix for several compilation warnings
  - Report an error message if user requests MV2_USE_XRC=1 in non-XRC configuration
  - Remove debug prints in regular code path with MV2_USE_BLOCKING=1
    - Thanks to Vaibhav Dutt for the report
  - Handling shared memory collective buffers in a dynamic manner to eliminate static setting of maximum CPU core count
  - Fix for validation issue in MPICH2 strided_get_indexed.c
  - Fix a bug in packetized transfers on heterogeneous clusters
  - Fix for deadlock between psm_ep_connect and PMGR_COLLECTIVE calls on QLogic systems
    - Thanks to Adam T. Moody for the patch
  - Fix a bug in MPI_Allocate_mem when it is called with size 0
    - Thanks to Michele De Stefano for reporting this issue
  - Create vendor for Open64 compilers and add rpath for unknown compilers
    - Thanks to Martin Hilgemen from Dell Inc. for the initial patch
  - Fix issue due to overlapping buffers with sprintf
    - Thanks to Mark Debbage from QLogic for reporting this issue
  - Fallback to using GNU options for unknown f90 compilers
  - Fix hang in PMI_Barrier due to incorrect handling of the socket return values in mpirun_rsh
  - Unify the redundant FTB events used to initiate a migration
  - Fix memory leaks when mpirun_rsh reads hostfiles
  - Fix a bug where library attempts to use in-active rail in multi-rail scenario

MVAPICH2-1.8alpl (11/14/2011)
* Bug Fixes (since 1.8a1)
  - Fix for a data validation issue in GPU transfers
    - Thanks to Massimiliano Fatica, NVIDIA, for reporting this issue
  - Tuned CUDA block size to 256K for better performance
  - Enhanced error checking for CUDA library calls
  - Fix for mpirun_rsh issue while launching applications on Linux Kernels (3.x)

MVAPICH2-1.8a1 (11/09/2011)

* Features and Enhancements (since 1.7):
  - Support for MPI communication from NVIDIA GPU device memory
    - High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
    - High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
    - Communication with contiguous datatype
  - Reduced memory footprint of the library
  - Enhanced one-sided communication design with reduced memory requirement
  - Enhancements and tuned collectives (Bcast and Alltoallv)
  - Update to hwloc v1.3.0
  - Flexible HCA selection with Nemesis interface
    - Thanks to Grigori Inozemtsev, Queens University
  - Support iWARP interoperability between Intel NE020 and Chelsio T4 Adapters
  - RoCE enable environment variable name is changed from MV2_USE_RDMAOE to MV2_USE_RoCE

* Bug Fixes (since 1.7):
  - Fix for a bug in mpirun_rsh while doing process clean-up in abort and other error scenarios
  - Fixes for code compilation warnings
  - Fix for memory leaks in RDMA CM code path

MVAPICH2-1.7 (10/14/2011)

* Features and Enhancements (since 1.7rc2):
  - Support SHMEM collectives up to 64 cores/node
  - Update to hwloc v1.2.2
  - Enhancement and tuned collective (GatherV)

* Bug Fixes:
  - Fixes for code compilation warnings
  - Fix job clean-up issues with mpirun_rsh
  - Fix a hang with RDMA CM

MVAPICH2-1.7rc2 (09/19/2011)

* Features and Enhancements (since 1.7rc1):
  - Based on MPICH2-1.4.1p1
  - Integrated Hybrid (UD-RC/XRC) design to get best performance on large-scale systems with reduced/constant memory footprint
- Shared memory backed Windows for One-Sided Communication
- Support for truly passive locking for intra-node RMA in shared memory and LIMIC based windows
- Integrated with Portable Hardware Locality (hwloc v1.2.1)
- Integrated with latest OSU Micro-Benchmarks (3.4)
- Enhancements and tuned collectives (Allreduce and Allgatherv)
- MPI_THREAD_SINGLE provided by default and MPI_THREAD_MULTIPLE as an option
- Enabling Checkpoint/Restart support in pure SMP mode
- Optimization for QDR cards
- On-demand connection management support with IB CM (RoCE interface)
- Optimization to limit number of RDMA Fast Path connections for very large clusters (Nemesis interface)
- Multi-core-aware collective support (QLogic PSM interface)

* Bug Fixes:
- Fixes for code compilation warnings
- Compiler preference lists reordered to avoid mixing GCC and Intel compilers if both are found by configure
- Fix a bug in transferring very large messages (>2GB)
  - Thanks to Tibor Pausz from Univ. of Frankfurt for reporting it
- Fix a hang with One-Sided Put operation
- Fix a bug in ptmalloc integration
- Avoid double-free crash with mpispawn
- Avoid crash and print an error message in mpirun_rsh when the hostfile is empty
- Checking for error codes in PMI design
- Verify programs can link with LIMIC2 at runtime
- Fix for compilation issue when BLCR or FTB installed in non-system paths
- Fix an issue with RDMA-Migration
- Fix for memory leaks
- Fix an issue in supporting RoCE with second port on available on HCA
  - Thanks to Jeffrey Konz from HP for reporting it
- Fix for a hang with passive RMA tests (QLogic PSM interface)

MVAPICH2-1.7rc1 (07/20/2011)

* Features and Enhancements (since 1.7a2)
- Based on MPICH2-1.4
- CH3 shared memory channel for standalone hosts (including laptops) without any InfiniBand adapters
- HugePage support
- Improved on-demand InfiniBand connection setup
- Optimized Fence synchronization (with and without LIMIC2 support)
- Enhanced mpirun_rsh design to avoid race conditions and support for improved debug messages
- Optimized design for collectives (Bcast and Reduce)
- Improved performance for medium size messages for QLogic PSM
- Support for Ekopath Compiler

* Bug Fixes
- Fixes in Dynamic Process Management (DPM) support
- Fixes in Checkpoint/Restart and Migration support
- Fix Restart when using automatic checkpoint
  - Thanks to Alexandr for reporting this
- Compilation warnings fixes
- Handling very large one-sided transfers using RDMA
- Fixes for memory leaks
- Graceful handling of unknown HCAs
- Better handling of shmem file creation errors
- Fix for a hang in intra-node transfer
- Fix for a build error with --disable-weak-symbols
  - Thanks to Peter Willis for reporting this issue
- Fixes for one-sided communication with passive target synchronization
- Proper error reporting when a program is linked with both static and
  shared MVAPICH2 libraries

MVAPICH2-1.7a2 (06/03/2011)

* Features and Enhancements (Since 1.7a)
  - Improved intra-node shared memory communication performance
  - Tuned RDMA Fast Path Buffer size to get better performance
    with less memory footprint (CH3 and Nemesis)
  - Fast process migration using RDMA
  - Automatic inter-node communication parameter tuning
    based on platform and adapter detection (Nemesis)
  - Automatic intra-node communication parameter tuning
    based on platform
  - Efficient connection set-up for multi-core systems
  - Enhancements for collectives (barrier, gather and allgather)
  - Compact and shorthand way to specify blocks of processes on the same
    host with mpirun_rsh
  - Support for latest stable version of HWLOC v1.2
  - Improved debug message output in process management and fault tolerance
    functionality
  - Better handling of process signals and error management in mpispawn
  - Performance tuning for pt-to-pt and several collective operations

* Bug fixes
  - Fixes for memory leaks
  - Fixes in CR/migration
  - Better handling of memory allocation and registration failures
  - Fixes for compilation warnings
  - Fix a bug that disallows '=' from mpirun_rsh arguments
  - Handling of non-contiguous transfer in Nemesis interface
  - Bug fix in gather collective when ranks are in cyclic order
  - Fix for the ignore_locks bug in MPI-I0 with Lustre

MVAPICH2-1.7a (04/19/2011)

* Features and Enhancements
  - Based on MPICH2-1.3.2p1
  - Integrated with Portable Hardware Locality (hwloc v1.1.1)
- Supporting Large Data transfers (>2GB)
- Integrated with Enhanced LiMIC2 (v0.5.5) to support Intra-node
  large message (>2GB) transfers
- Optimized and tuned algorithm for AlltoAll
- Enhanced debugging config options to generate
  core files and back-traces
- Support for Chelsio's T4 Adapter

MVAPICH2-1.6 (03/09/2011)

* Features and Enhancements (since 1.6-RC3)
  - Improved configure help for MVAPICH2 features
  - Updated Hydra launcher with MPICH2-1.3.3 Hydra process manager
  - Building and installation of OSU micro benchmarks during default
    MVAPICH2 installation
  - Hydra is the default mpiexec process manager

* Bug fixes (since 1.6-RC3)
  - Fix hang issues in RMA
  - Fix memory leaks
  - Fix in RDMA_FP

MVAPICH2-1.6-RC3 (02/15/2011)

* Features and Enhancements
  - Support for 3D torus topology with appropriate SL settings
    - For both CH3 and Nemesis interfaces
  - Thanks to Jim Schutt, Marcus Epperson and John Nagle from
    Sandia for the initial patch
  - Quality of Service (QoS) support with multiple InfiniBand SL
    - For both CH3 and Nemesis interfaces
  - Configuration file support (similar to the one available in MVAPICH).
    Provides a convenient method for handling all runtime variables
    through a configuration file.
  - Improved job-startup performance on large-scale systems
  - Optimization in MPI_Finalize
  - Improved pt-to-pt communication performance for small and
    medium messages
  - Optimized and tuned algorithms for Gather and Scatter collective
    operations
  - Optimized thresholds for one-sided RMA operations
  - User-friendly configuration options to enable/disable various
    checkpoint/restart and migration features
  - Enabled ROMIO's auto detection scheme for filetypes
    on Lustre file system
  - Improved error checking for system and BLCR calls in
    checkpoint-restart and migration codepath
  - Enhanced OSU Micro-benchmarks suite (version 3.3)

Bug Fixes
  - Fix in aggregate ADIO alignment
  - Fix for an issue with LiMIC2 header
- XRC connection management
- Fixes in registration cache
- IB card detection with MV2_IBA_HCA runtime option in multi rail design
- Fix for a bug in multi-rail design while opening multiple HCAs
- Fixes for multiple memory leaks
- Fix for a bug in mpirun_rsh
- Checks before enabling aggregation and migration
- Fixing the build errors with --disable-cxx
- Thanks to Bright Yang for reporting this issue
- Fixing the build errors related to "pthread_spinlock_t" seen on RHEL systems

MVAPICH2-1.6-RC2 (12/22/2010)

* Features and Enhancements
  - Optimization and enhanced performance for clusters with nVIDIA GPU adapters (with and without GPUDirect technology)
  - Enhanced R3 rendezvous protocol
    - For both CH3 and Nemesis interfaces
  - Robust RDMA Fast Path setup to avoid memory allocation failures
    - For both CH3 and Nemesis interfaces
  - Multiple design enhancements for better performance of medium sized messages
  - Enhancements and optimizations for one sided Put and Get operations
  - Enhancements and tuning of Allgather for small and medium sized messages
  - Optimization of AllReduce
  - Enhancements to Multi-rail Design and features including striping of one-sided messages
  - Enhancements to mpirun_rsh job start-up scheme
  - Enhanced designs for automatic detection of various architectures and adapters

* Bug fixes
  - Fix a bug in Post-Wait/Start-Complete path for one-sided operations
  - Resolving a hang in mpirun_rsh termination when CR is enabled
  - Fixing issue in MPI_Allreduce and Reduce when called with MPI_IN_PLACE
    - Thanks to the initial patch by Alexander Alekhin
  - Fix for an issue in rail selection for small RMA messages
  - Fix for threading related errors with comm_dup
  - Fix for alignment issues in RDMA Fast Path
  - Fix for extra memcpy in header caching
  - Fix for an issue to use correct HCA when process to rail binding scheme used in combination with XRC.
  - Fix for an RMA issue when configured with enable-g=meminit
    - Thanks to James Dinan of Argonne for reporting this issue
  - Only set FC and F77 if gfortran is executable

(continues on next page)
**Features and Enhancements**

- Using LiMIC2 for efficient intra-node RMA transfer to avoid extra memory copies
- Upgraded to LiMIC2 version 0.5.4
- Removing the limitation on number of concurrent windows in RMA operations
- Support for InfiniBand Quality of Service (QoS) with multiple lanes
- Enhanced support for multi-threaded applications
- Fast Checkpoint-Restart support with aggregation scheme
- Job Pause-Migration-Restart Framework for Pro-active Fault-Tolerance
- Support for new standardized Fault Tolerant Backplane (FTB) Events for Checkpoint-Restart and Job Pause-Migration-Restart Framework
- Dynamic detection of multiple InfiniBand adapters and using these by default in multi-rail configurations (OLA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3 interfaces)
- Support for process-to-rail binding policy (bunch, scatter and user-defined) in multi-rail configurations (OFA-IB-CH3, OFA-iWARP-CH3 and OFA-RoCE-CH3 interfaces)
- Enhanced and optimized algorithms for MPI_Reduce and MPI_AllReduce operations for small and medium message sizes.
- XRC support with Hydra Process Manager
- Improved usability of process to CPU mapping with support of delimiters ('+', '-', ' -') in CPU listing
- Thanks to Gilles Civario for the initial patch
- Use of gfortran as the default F77 compiler
- Support of Shared-Memory-Nemesis interface on multi-core platforms requiring intra-node communication only (SMP-only systems, laptops, etc.)

**Bug fixes**

- Fix for memory leak in one-sided code with --enable-g=all --enable-error-messages=all
- Fix for memory leak in getting the context of intra-communicator
- Fix for shmat() return code check
- Fix for issues with inter-communicator collectives in Nemesis
- KNEM patch for osu_bibw issue with KNEM version 0.9.2
- Fix for osu_bibw error with Shared-memory-Nemesis interface
- Fix for Win_test error for one-sided RDMA
- Fix for a hang in collective when thread level is set to multiple
- Fix for intel test errors with rsend, bsend and ssnd operations in Nemesis
- Fix for memory free issue when it allocated by scandir
- Fix for a hang in Finalize
- Fix for issue with MPIU_Find_local_and_external when it is called from MPIDI_CH3I_comm_create
- Fix for handling CPPFLGS values with spaces
- Dynamic Process Management to work with XRC support
- Fix related to disabling CPU affinity when shared memory is turned off at run time
- Significantly reduce memory footprint on some systems by changing the stack size setting for multi-rail configurations
- Optimization to the number of RDMA Fast Path connections
- Performance improvements in Scatterv and Gatherv collectives for CH3 interface (Thanks to Dan Kokran and Max Suarez of NASA for identifying the issue)
- Tuning of Broadcast Collective
- Support for tuning of eager thresholds based on both adapter and platform type
- Environment variables for message sizes can now be expressed in short form K=Kilobytes and M=Megabytes (e.g. MV2_IBA_EAGER_THRESHOLD=12K)
- Ability to selectively use some or all HCAs using colon separated lists. e.g. MV2_IBA_HCA=mlx4_0:mlx4_1
- Improved Bunch/Scatter mapping for process binding with HWLOC and SMT support (Thanks to Dr. Bernd Kallies of ZIB for ideas and suggestions)
- Update to Hydra code from MPICH2-1.3b1
- Auto-detection of various iWARP adapters
- Specifying MV2_USE_IWARP=1 is no longer needed when using iWARP
- Changing automatic eager threshold selection and tuning for iWARP adapters based on number of nodes in the system instead of the number of processes
- PSM progress loop optimization for QLogic Adapters (Thanks to Dr. Avneesh Pant of QLogic for the patch)

* Bug fixes
- Fix memory leak in registration cache with --enable-g=all
- Fix memory leak in operations using datatype modules
- Fix for rdma_cross_connect issue for RDMA CM. The server is prevented from initiating a connection.
- Don’t fail during build if RDMA CM is unavailable
- Various mpirun_rsh bug fixes for CH3, Nemesis and uDAPL interfaces
- ROMIO panfs build fix
- Update panfs for not-so-new ADIO file function pointers
- Shared libraries can be generated with unknown compilers
- Explicitly link against DL library to prevent build error due to DSO link change in Fedora 13 (introduced with gcc-4.4.3-5.fc13)
- Fix regression that prevents the proper use of our internal HWLOC component
- Remove spurious debug flags when certain options are selected at build time
- Error code added for situation when received eager SMP message is larger than receive buffer
- Fix for Gather and GatherV back-to-back hang problem with LiMIC2
- Fix for packetized send in Nemesis
- Fix related to eager threshold in nemesis ib-netmod
- Fix initialization parameter for Nemesis based on adapter type
- Fix for uDAPL one sided operations (Thanks to Jakub Fedoruk from Intel for reporting this)
- Fix an issue with out-of-order message handling for iWARP
- Fixes for memory leak and Shared context Handling in PSM for QLogic Adapters (Thanks to Dr. Avneesh Pant of QLogic for the patch)
MVAPICH2-1.5 (07/09/10)

* Features and Enhancements (since 1.5-RC2)
  - SRQ turned on by default for Nemesis interface
  - Performance tuning - adjusted eager thresholds for
    variety of architectures, vbuf size based on adapter
    types and vbuf pool sizes
  - Tuning for Intel iWARP NE020 adapter, thanks to Harry
    Cropper of Intel
  - Introduction of a retry mechanism for RDMA_CM connection
    establishment

* Bug fixes (since 1.5-RC2)
  - Fix in build process with hwloc (for some Distros)
  - Fix for memory leak (Nemesis interface)

MVAPICH2-1.5-RC2 (06/21/10)

* Features and Enhancements (since 1.5-RC1)
  - Support for hwloc library (1.0.1) for defining CPU affinity
  - Deprecating the PLPA support for defining CPU affinity
  - Efficient CPU affinity policies (bunch and scatter) to
    specify CPU affinity per job for modern multi-core platforms
  - New flag in mpirun_rsh to execute tasks with different group IDs
  - Enhancement to the design of Win_complete for RMA operations
  - Flexibility to support variable number of RMA windows
  - Support for Intel iWARP NE020 adapter

* Bug fixes (since 1.5-RC1)
  - Compilation issue with the ROMIO adio-lustre driver, thanks
    to Adam Moody of LLNL for reporting the issue
  - Allowing checkpoint-restart for large-scale systems
  - Correcting a bug in clear_kvc function. Thanks to T J (Chris) Ward,
    IBM Research, for reporting and providing the resolving patch
  - Shared lock operations with RMA with scatter process distribution.
    Thanks to Pavan Balaji of Argonne for reporting this issue
  - Fix a bug during window creation in uDAPL
  - Compilation issue with --enablealloca, Thanks to E. Borisch,
    for reporting and providing the patch
  - Improved error message for ibv_poll_cq failures
  - Fix an issue that prevents mpirun_rsh to execute programs without
    specifying the path from directories in PATH
  - Fix an issue of mpirun_rsh with Dynamic Process Migration (DPM)
  - Fix for memory leaks (both CH3 and Nemesis interfaces)
  - Updatefiles correctly update LiMIC2
  - Several fixes to the registration cache
    (CH3, Nemesis and uDAPL interfaces)
  - Fix to multi-rail communication
  - Fix to Shared Memory communication Progress Engine
  - Fix to all-to-all collective for large number of processes
MVAPICH2-1.5-RC1 (05/04/10)

* Features and Enhancements
  - MPI 2.2 compliant
  - Based on MPICH2-1.2.1p1
  - OFA-IB-Nemesis interface design
    - OpenFabrics InfiniBand network module support for MPICH2 Nemesis modular design
    - Support for high-performance intra-node shared memory communication provided by the Nemesis design
    - Adaptive RDMA Fastpath with Polling Set for high-performance inter-node communication
    - Shared Receive Queue (SRQ) support with flow control, uses significantly less memory for MPI library
  - Header caching
  - Advanced AVL tree-based Resource-aware registration cache
  - Memory Hook Support provided by integration with ptmalloc2 library. This provides safe release of memory to the Operating System and is expected to benefit the memory usage of applications that heavily use malloc and free operations.
  - Support for TotalView debugger
  - Shared Library Support for existing binary MPI application programs to run ROMIO Support for MPI-I0
  - Support for additional features (such as hwloc, hierarchical collectives, one-sided, multithreading, etc.), as included in the MPICH2 1.2.1p1 Nemesis channel
    - Flexible process manager support
      - mpirun_rsh to work with any of the eight interfaces (CH3 and Nemesis channel-based) including OFA-IB-Nemesis, TCP/IP-CH3 and TCP/IP-Nemesis
      - Hydra process manager to work with any of the eight interfaces (CH3 and Nemesis channel-based) including OFA-IB-CH3, OFA-iWARP-CH3, OFA-RoCE-CH3 and TCP/IP-CH3
    - MPIEXEC_TIMEOUT is honored by mpirun_rsh

* Bug fixes since 1.4.1
  - Fix compilation error when configured with --enable-thread-funneled'
  - Fix MPE functionality, thanks to Anthony Chan for reporting and providing the resolving patch
  - Cleanup after a failure in the init phase is handled better by mpirun_rsh
  - Path determination is correctly handled by mpirun_rsh when DPM is used
  - Shared libraries are correctly built (again)
* Enhancements since mvapich2-1.4
  - MPMD launch capability to mpirun_rsh
  - Portable Hardware Locality (hwloc) support, patch suggested by Dr. Bernd Kallies <kallies@zib.de>
  - Multi-port support for iWARP
  - Enhanced iWARP design for scalability to higher process count
  - Ring based startup support for RDMAoE

* Bug fixes since mvapich2-1.4
  - Fixes for MPE and other profiling tools as suggested by Anthony Chan (chan@mcs.anl.gov)
  - Fixes for finalization issue with dynamic process management
  - Removed overrides to PSM_SHAREDCONTEXT, PSM_SHAREDCONTEXTS_MAX variables. Suggested by Ben Truscott <b.s.truscott@bristol.ac.uk>.
  - Fixing the error check for buffer aliasing in MPI_Reduce as suggested by Dr. Rajeev Thakur <thakur@mcs.anl.gov>
  - Fix Totalview integration for RHEL5
  - Update simplemake to handle build timestamp issues
  - Fixes for --enable-g={mem, meminit}
  - Improved logic to control the receive and send requests to handle the limitation of CQ Depth on iWARP
  - Fixing assertion failures with IMB-EXT tests
  - VBUF size for very small iWARP clusters bumped up to 33K
  - Replace internal mallocs with MPIU_Malloc uniformly for correct tracing with --enable-g=mem
  - Fixing multi-port for iWARP
  - Fix memory leaks
  - Shared-memory reduce fixes for MPI_Reduce invoked with MPI_IN_PLACE
  - Handling RDMA_CM_EVENT_TIMEWAIT_EXIT event
  - Fix for threaded-ctxdup mpich2 test
  - Detecting spawn errors, patch contributed by Dr. Bernd Kallies <kallies@zib.de>
  - IMB-EXT fixes reported by Yutaka from Cray Japan
  - Fix alltoall assertion error when limic is used

MVAPICH2-1.4

* Enhancements since mvapich2-1.4rc2
  - Efficient runtime CPU binding
  - Add an environment variable for controlling the use of multiple cq's for iWARP interface.
  - Add environmental variables to disable registration cache for All-to-All on large systems.
  - Performance tune for pt-to-pt Intra-node communication with LiMIC2
  - Performance tune for MPI_Broadcast

* Bug fixes since mvapich2-1.4rc2
  - Fix the reading error in lock_get_response by adding initialization to req->mrail.protocol
  - Fix mpirun_rsh scalability issue with hierarchical ssh scheme when launching greater than 8K processes.
  - Add mvapich_prefix to yacc functions. This can avoid some namespace
issues when linking with other libraries. Thanks to Manhui Wang <wangm9@cardiff.ac.uk> for contributing the patch.

MVAPICH2-1.4-rc2

* Enhancements since mvapich2-1.4rc1
- Added Feature: Check-point Restart with Fault-Tolerant Backplane Support (FTB_CR)
- Added Feature: Multiple CQ-based design for Chelsio iWARP
- Distribute LiMIC2-0.5.2 with MVAPICH2. Added flexibility for selecting and using a pre-existing installation of LiMIC2
- Increase the amount of command line that mpirun_rsh can handle (Thanks for the suggestion by Bill Barth @ TACC)

* Bug fixes since mvapich2-1.4rc1
- Fix for hang with packetized send using RDMA Fast path
- Fix for allowing to use user specified P_Key's (Thanks to Mike Heinz @ QLogic)
- Fix for allowing mpirun_rsh to accept parameters through the parameters file (Thanks to Mike Heinz @ QLogic)
- Modify the default value of shmem_bcast_leaders to 4K
- Fix for one-sided with XRC support
- Fix hang with XRC
- Fix to always enabling MVAPICH2.Sync_Checkpoint functionality
- Fix build error on RHEL 4 systems (Reported by Nathan Baca and Jonathan Atencio)
- Fix issue with PGI compilation for PSM interface
- Fix for one-sided accumulate function with user-defined contiguous datatypes
- Fix linear/hierarchical switching logic and reduce threshold for the enhanced mpirun_rsh framework.
- Clean up intra-node connection management code for iWARP
- Fix --enable-g=all issue with uDAPL interface
- Fix one sided operation with on demand CM.
- Fix VPATH build

MVAPICH2-1.4-rc1

* Bugs fixed since MVAPICH2-1.2p1
- Changed parameters for iWARP for increased scalability
- Fix error with derived datatypes and Put and Accumulate operations Request was being marked complete before data transfer had actually taken place when MV_RNDV_PROTOCOL=R3 was used
- Unregister stale memory registrations earlier to prevent malloc failures
- Fix for compilation issues with --enable-g=mem and --enable-g=all
- Change dapl_prepost_noop_extra value from 5 to 8 to prevent

(continues on next page)
credit flow issues.

- Re-enable RGET (RDMA Read) functionality

- Fix SRQ Finalize error
  Make sure that finalize does not hang when the srq_post_cond is being waited on.

- Fix a multi-rail one-sided error when multiple QPs are used

- PMI Lookup name failure with SLURM

- Port auto-detection failure when the 1st HCA did not have an active failure

- Change default small message scheduling for multirail for higher performance

- MPE support for shared memory collectives now available

MVAPICH2-1.2p1 (11/11/2008)

* Changes since MVAPICH2-1.2
  - Fix shared-memory communication issue for AMD Barcelona systems.

MVAPICH2-1.2 (11/06/2008)

* Bugs fixed since MVAPICH2-1.2-rc2
  - Ignore the last bit of the pkey and remove the pkey_ix option since the index can be different on different machines. Thanks for Pasha@Mellanox for the patch.

  - Fix data types for memory allocations. Thanks for Dr. Bill Barth from TACC for the patches.

  - Fix a bug when MV2_NUM_HCAS is larger than the number of active HCAs.

  - Allow builds on architectures for which tuning parameters do not exist.

* Changes related to the mpirun_rsh framework
  - Always build and install mpirun_rsh in addition to the process manager(s) selected through the --with-pm mechanism.

  - Cleaner job abort handling

  - Ability to detect the path to mpispawn if the Linux proc filesystem is available.

  - Added Totalview debugger support
- Stdin is only available to rank 0. Other ranks get /dev/null.

* Other miscellaneous changes

- Add sequence numbers for RPUT and RGET finish packets.
- Increase the number of allowed nodes for shared memory broadcast to 4K.
- Use /dev/shm on Linux as the default temporary file path for shared memory communication. Thanks for Doug Johnson@OSC for the patch.
- MV2_DEFAULT_MAX_WQE has been replaced with MV2_DEFAULT_MAX_SEND_WQE and MV2_DEFAULT_MAX_RECV_WQE for send and recv wqes, respectively.
- Fix compilation warnings.

MVAPICH2-1.2-RC2 (08/20/2008)

* Following bugs are fixed in RC2

- Properly handle the scenario in shared memory broadcast code when the datatypes of different processes taking part in broadcast are different.
- Fix a bug in Checkpoint-Restart code to determine whether a connection is a shared memory connection or a network connection.
- Support non-standard path for BLCR header files.
- Increase the maximum heap size to avoid race condition in realloc().
- Use int32_t for rank for larger jobs with 32k processes or more.
- Improve mvapich2-1.2 bandwidth to the same level of mvapich2-1.0.3.
- An error handling patch for uDAPL interface. Thanks for Nilesh Awate for the patch.
- Explicitly set some of the EP attributes when on demand connection is used in uDAPL interface.

MVAPICH2-1.2-RC1 (07/02/08)

* Following features are added for this new mvapich2-1.2 release:

- Based on MPICH2 1.0.7
- Scalable and robust daemon-less job startup
  -- Enhanced and robust mpirun_rsh framework (non-MPD-based) to provide scalable job launching on multi-thousand core clusters
-- Available for OpenFabrics (IB and iWARP) and uDAPL interfaces
   (including Solaris)

- Adding support for intra-node shared memory communication with Checkpoint-restart
  -- Allows best performance and scalability with fault-tolerance support

- Enhancement to software installation
  -- Change to full autoconf-based configuration
  -- Adding an application (mpiname) for querying the MVAPICH2 library
     version and configuration information

- Enhanced processor affinity using PLPA for multi-core architectures
- Allows user-defined flexible processor affinity
- Enhanced scalability for RDMA-based direct one-sided communication
  with less communication resource
- Shared memory optimized MPI_Bcast operations
- Optimized and tuned MPI_Alltoall

MVAPICH2-1.0.2 (02/20/08)

* Change the default MV2_DAPL_PROVIDER to OpenIB-cma

* Remove extraneous parameter is_blocking from the gen2 interface for
  MPIDI_CH3I_MRAILI_Get_next_vbuf

* Explicitly name unions in struct ibv_wr_descriptor and reference the
  members in the code properly.

* Change "inline" functions to "static inline" properly.

* Increase the maximum number of buffer allocations for communication
  intensive applications

* Corrections for warnings from the Sun Studio 12 compiler.

* If malloc hook initialization fails, then turn off registration cache

* Add MV_R3_THRESHOLD and MV_R3_NOCACHE_THRESHOOLD which allows
  R3 to be used for smaller messages instead of registering the
  buffer and using a zero-copy protocol.

* Fixed an error in message coalescing.

* Setting application initiated checkpoint as default if CR is turned on.

(continues on next page)
MVAPICH2-1.0.1 (10/29/07)

* Enhance udapl initialization, set all ep_attr fields properly.
  Thanks for Kanoj Sarcar from NetXen for the patch.

* Fixing a bug that miscalculates the receive size in case of complex
datatype is used.
  Thanks for Patrice Martinez from Bull for reporting this problem.

* Minor patches for fixing (i) NBO for rdma-cm ports and (ii) rank
  variable usage in DEBUG_PRINT in rdma-cm.c
  Thanks to Steve Wise for reporting these.

MVAPICH2-1.0 (09/14/07)

* Following features and bug fixes are added in this new MVAPICH2-1.0 release:

- Message coalescing support to enable reduction of per Queue-pair
  send queues for reduction in memory requirement on large scale
  clusters. This design also increases the small message messaging
  rate significantly. Available for Open Fabrics Gen2-IB.

- Hot-Spot Avoidance Mechanism (HSAM) for alleviating
  network congestion in large scale clusters. Available for
  Open Fabrics Gen2-IB.

- RDMA CM based on-demand connection management for large scale
  clusters. Available for OpenFabrics Gen2-IB and Gen2-iWARP.

- uDAPL on-demand connection management for large scale clusters.
  Available for uDAPL interface (including Solaris IB implementation).

- RDMA Read support for increased overlap of computation and
  communication. Available for OpenFabrics Gen2-IB and Gen2-iWARP.

- Application-initiated system-level (synchronous) checkpointing in
  addition to the user-transparent checkpointing. User application can
  now request a whole program checkpoint synchronously with BLCR by
  calling special functions within the application. Available for
  OpenFabrics Gen2-IB.

- Network-Level fault tolerance with Automatic Path Migration (APM)
  for tolerating intermittent network failures over InfiniBand.
  Available for OpenFabrics Gen2-IB.

- Integrated multi-rail communication support for OpenFabrics
  Gen2-iWARP.

- Blocking mode of communication progress. Available for OpenFabrics
Gen2-IB.
- Based on MPICH2 1.0.5p4.

* Fix for hang while using IMB with -multi option.
  Thanks to Pasha (Mellanox) for reporting this.

* Fix for hang in memory allocations > 2^31 - 1.
  Thanks to Bryan Putnam (Purdue) for reporting this.

* Fix for RDMA_CM finalize rdma_destroy_id failure.
  Added Timeout env variable for RDMA_CM ARP.
  Thanks to Steve Wise for suggesting these.

* Fix for RDMA_CM invalid event in finalize. Thanks to Steve Wise and Sean Hefty.

* Fix for shmem memory collectives related memory leaks

* Updated src/mpi/romio/adio/ad_panfs/Makefile.in include path to find mpi.h.
  Contributed by David Gunter, Los Alamos National Laboratory.

* Fixed header caching error on handling datatype messages with small vector sizes.

* Change the finalization protocol for UD connection manager.

* Fix for the "command line too long" problem. Contributed by Xavier Bru
  <xavier.bru@bull.net> from Bull (http://www.bull.net/)

* Change the CKPT handling to invalidate all unused registration cache.

* Added ofed 1.2 interface change patch for iwarp/rdma_cm from Steve Wise.

* Fix for rdma_cm_get_event err in finalize. Reported by Steve Wise.

* Fix for when MV2_IBA_HCA is used. Contributed by Michael Schwind
  of Technical Univ. of Chemnitz (Germany).

MVAPICH2-0.9.8 (11/10/06)

* Following features are added in this new MVAPICH2-0.9.8 release:
  - BLCR based Checkpoint/Restart support
  - iWARP support: tested with Chelsio and Ammasso adapters and OpenFabrics/Gen2 stack
  - RDMA CM connection management support
  - Shared memory optimizations for collective communication operations
- uDAPL support for NetEffect 10GigE adapter.

**MVAPICH2-0.9.6 (10/22/06)**

* Following features and bug fixes are added in this new MVAPICH2-0.9.6 release:
  - Added on demand connection management.
  - Enhance shared memory communication support.
  - Added ptmalloc memory hook support.
  - Runtime selection for most configuration options.

**MVAPICH2-0.9.5 (08/30/06)**

* Following features and bug fixes are added in this new MVAPICH2-0.9.5 release:
  - Added multi-rail support for both point to point and direct one side operations.
  - Added adaptive RDMA fast path.
  - Added shared receive queue support.
  - Added TotalView debugger support

* Optimization of SMP startup information exchange for USE_MPD_RING to enhance performance for SLURM. Thanks to Don and team members from Bull and folks from LLNL for their feedbacks and comments.

* Added uDAPL build script functionality to set DAPL_DEFAULT_PROVIDER explicitly with default suggestions.

* Thanks to Harvey Richardson from Sun for suggesting this feature.

**MVAPICH2-0.9.3 (05/20/06)**

* Following features are added in this new MVAPICH2-0.9.3 release:
  - Multi-threading support
  - Integrated with MPICH2 1.0.3 stack
  - Advanced AVL tree-based Resource-aware registration cache
  - Tuning and Optimization of various collective algorithms
  - Processor affinity for intra-node shared memory communication
- Auto-detection of InfiniBand adapters for Gen2

**MVAPICH2-0.9.2 (01/15/06)**

* Following features are added in this new MVAPICH2-0.9.2 release:
  - InfiniBand support for OpenIB/Gen2
  - High-performance and optimized support for many MPI-2 functionalities (one-sided, collectives, datatype)
  - Support for other MPI-2 functionalities (as provided by MPICH2 1.0.2p1)
  - High-performance and optimized support for all MPI-1 functionalities

**MVAPICH2-0.9.0 (11/01/05)**

* Following features are added in this new MVAPICH2-0.9.0 release:
  - Optimized two-sided operations with RDMA support
  - Efficient memory registration/de-registration schemes for RDMA operations
  - Optimized intra-node shared memory support (bus-based and NUMA)
  - Shared library support
  - ROMIO support
  - Support for multiple compilers (gcc, icc, and pgi)

**MVAPICH2-0.6.5 (07/02/05)**

* Following features are added in this new MVAPICH2-0.6.5 release:
  - uDAPL support (tested for InfiniBand, Myrinet, and Ammasso GigE)

**MVAPICH2-0.6.0 (11/04/04)**

* Following features are added in this new MVAPICH2-0.6.0 release:
  - MPI-2 functionalities (one-sided, collectives, datatype)
  - All MPI-1 functionalities
  - Optimized one-sided operations (Get, Put, and Accumulate)
- Support for active and passive synchronization
- Optimized two-sided operations
- Scalable job start-up
- Optimized and tuned for the above platforms and different network interfaces (PCI-X and PCI-Express)
- Memory efficient scaling modes for medium and large clusters

5.14 SLURM Release Information

The following is reproduced essentially verbatim from files contained within the SLURM tarball downloaded from https://slurm.schedmd.com.

SLURM was produced at Lawrence Livermore National Laboratory in collaboration with various organizations.

Copyright (C) 2012-2013 Los Alamos National Security, LLC.
Copyright (C) 2011 Trinity Centre for High Performance Computing
Copyright (C) 2010-2015 SchedMD LLC
Copyright (C) 2009-2013 CEA/DAM/DIF
Copyright (C) 2009-2011 Centro Svizzero di Calcolo Scientifico (CSCS)
Copyright (C) 2008-2011 Lawrence Livermore National Security
Copyright (C) 2008 Vijay Ramasubramanian
Copyright (C) 2007-2008 Red Hat, Inc.
Copyright (C) 2007-2013 National University of Defense Technology, China
Copyright (C) 2007-2015 Bull
Copyright (C) 2005-2008 Hewlett-Packard Development Company, L.P.
Copyright (C) 2004-2009, Marcus Holland-Moritz
Copyright (C) 2002-2007 The Regents of the University of California
Copyright (C) 2002-2003 Linux NetworX
Copyright (C) 2002 University of Chicago
Copyright (C) 2001, Paul Marquess
Copyright (C) 2000 Markus Friedl
Copyright (C) 1999, Kenneth Albanowski
Copyright (C) 1998 Todd C. Miller <Todd.Miller@courtesan.com>
Copyright (C) 1996-2003 Maximum Entropy Data Consultants Ltd,
Copyright (C) 1995 Tatu Ylonen <ylo@cs.hut.fi>, Espoo, Finland

Many other organizations contributed code and/or documentation without including a copyright notice.

Written by:
Amjad Majid Ali (Colorado State University)
Par Andersson (National Supercomputer Centre, Sweden)
Don Albert (Bull)
Ernest Artiaga (Barcelona Supercomputer Center, Spain)
Danny Auble (LLNL, SchedMD LLC)
Susanne Balle (HP)
Anton Blanchard (Samba)
Janne Blomqvist (Aalto University, Finland)
David Bremer (LLNL)
Jon Bringhurst (LANL)
Bill Brophy (Bull)
Hongjia Cao (National University of Defense Technology, China)
Daniel Christians (HP)
Gilles Civario (Bull)
Chuck Clouston (Bull)
Joseph Donaghy (LLNL)
Chris Dunlap (LLNL)
Joey Ekstrom (LLNL/Brigham Young University)
Josh England (TGS Management Corporation)
Kent Engstrom (National Supercomputer Centre, Sweden)
Jim Garlick (LLNL)
Didier Gazen (Laboratoire d’Aerologie, France)
Raphael Geissert (Debian)
Yiannis Georgiou (Bull)
Andriy Grytsenko (Massive Solutions Limited, Ukraine)
Mark Grondona (LLNL)
Takao Hatazaki (HP, Japan)
Matthieu Hautreux (CEA, France)
Chris Holmes (HP)
David Hoppner
Nathan Huff (North Dakota State University)
David Jackson (Adaptive Computing)
Morris Jette (LLNL, SchedMD LLC)
Klaus Joas (University Karlsruhe, Germany)
Greg Johnson (LANL)
Jason King (LLNL)
Aaron Knister (Environmental Protection Agency)
Nancy Kritkausky (Bull)
Roman Kurakin (Institute of Natural Science and Ecology, Russia)
Eric Lin (Bull)
Don Lipari (LLNL)
Puenlap Lee (Bull)
Dennis Leepow
Bernard Li (Genome Sciences Centre, Canada)
Donald Lipari (LLNL)
Steven McDougall (SiCortex)
Donna Mecozzi (LLNL)
Bjorn-Helge Mevik (University of Oslo, Norway)
Chris Morrone (LLNL)
Pere Munt (Barcelona Supercomputer Center, Spain)
Michal Novotny (Masaryk University, Czech Republic)
Bryan O’Sullivan (Pathscale)
Gennaro Oliva (Institute of High Performance Computing and Networking, Italy)
Alejandro Lucero Palau (Barcelona Supercomputer Center, Spain)
Daniel Palermo (HP)
Dan Phung (LLNL/Columbia University)
This file is part of SLURM, a resource management program.

SLURM is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

SLURM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SLURM; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

OUR NOTICE AND TERMS OF AND CONDITIONS OF THE GNU GENERAL PUBLIC LICENSE

Our Preamble Notice

Auspices

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Disclaimer

This work was sponsored by an agency of the United States government. Neither the United States Government nor Lawrence Livermore National Security, LLC, nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial products, process, or services by trade names, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

The precise terms and conditions for copying, distribution and modification is provided in the file named "COPYING" in this directory.

This file describes changes in recent versions of Slurm. It primarily documents those changes that are of interest to users and administrators.

* Changes in Slurm 18.08.6-2

-- Remove deadlock situation when logging and --enable-debug is used.
-- Fix RPM packaging for accounting_storage/mysql.

* Changes in Slurm 18.08.6

-- Added parsing of -H flag with scancel.
-- Fix slurmsswd build on 32-bit systems.
-- acct_gather_filesystem/lustre - add support for Lustre 2.12 client.
-- Fix per-partition TRES factors/priority
-- Fix per-partition NICE priority
-- Fix partition access check validation for multi-partition job submissions.
-- Prevent segfault on empty response in 'scontrol show dwstat'.
-- node_features/knl_cray plugin - Preserve node's active features if it has already booted when slurmctld daemon is reconfigured.
-- Detect missing burst buffer script and reject job.
-- GRES: Properly reset the topo_gres_cnt_alloc counter on slurmctld restart to prevent underflow.
-- Avoid errors from packing accounting_storage_mysql.so when RPM is built with out mysql support.
-- Remove deprecated -t option from slurmctld --help.
-- acct_gather_filesystem/lustre - fix stats gathering.
-- Enforce documented default usage start and end times when querying jobs from the database.
-- Fix issues when querying running jobs from the database.
-- Deny sacct request where start time is later than the end time requested.
(continues on next page)
-- Fix sacct verbose about time and states queried.
-- burst_buffer/cray - allow 'scancel --hurry <jobid>' to tear down a burst buffer that is currently staging data out.
-- X11 forwarding - allow setup if the DISPLAY environment variable lacks a screen number. (Permit both "localhost:10.0" and "localhost:10".)
-- docs - change HTML title to include the page title or man page name.
-- X11 forwarding - fix an unnecessary error message when using the local_xauthority X11Parameters option.
-- Add use_raw_hostname to X11Parameters.
-- Fix small so it passes job arrays to seff correctly.
-- Don't check InactiveLimit for salloc --no-shell jobs.
-- Add SALLOCGRES and SBATCHGRES as input to salloc/sbatch.
-- Remove drain state when node doesn't reboot by ResumeTimeout.
-- Fix considering "resuming" nodes in scheduling.
-- Do not kill suspended jobs due to exceeding time limit.
-- Add NoAddrCache CommunicationParameter.
-- Don't ping powering up cloud nodes.
-- Add cloud_dns SlurmctldParameter.
-- Consider --sbindir configure option as the default path to find slurmstepd.
-- Fix node state printing of DRAINED
-- Fix spamming dbd of down/drained nodes in maintenance reservation.
-- Avoid buffer overflow in time_str2secs.
-- Calculate suspended time for suspended steps.
-- Add null check for step_ptr->step_node_bitmap in _pick_step_nodes.
-- Fix multi-cluster srun issue after 'scontrol reconfigure' was called.
-- Fix accessing response_cluster_rec outside of write locks.
-- Fix Lua user messages not showing up on rejected submissions.
-- Fix printing multi-line error messages on rejected submissions.

* Changes in Slurm 18.08.5-2
==================================
-- Fix Perl build for 32-bit systems.

* Changes in Slurm 18.08.5
==================================
-- Backfill - If a job has a time_limit guess the end time of a job better if OverTimeLimit is Unlimited.
-- Fix "sacctmgr show events event=cluster"
-- Fix sacctmgr show runawayjobs from sibling cluster
-- Avoid bit offset of -1 in call to bit_nclear().
-- Insure that "hbm" is a configured GresType on knl systems.
-- Fix NodeFeaturesPlugins=node_features/knl_generic to allow other gres other than knl.
-- cons_res: Prevent overflow on multiply.
-- Better debug for bad values in gres.conf.
-- Fix double accounting of energy at end of job.
-- Read gres.conf for cloud nodes on slurmctld.
-- Don't assume the first node of a job is the batch host when purging jobs from a node.
-- Better debugging when a job doesn't have a job_resrcs ptr.
-- Store ave watts in energy plugins.
-- Add XCC plugin for reading Lenovo Power.
-- Fix minor memory leak when scheduling rebootable nodes.
-- Fix debug2 prefix for sched log.
-- Fix printing correct SLURM_JOB_ACCOUNT_PACK_GROUP_* in env for a Het Job.
-- sbatch - search current working directory first for job script.
-- Make it so held jobs reset the AccrueTime and do not count against any
  AccrueTime limits.
-- Add SchedulerParameters option of bf_hetjob_prio=[min|avg|max] to alter the
  job sorting algorithm for scheduling heterogeneous jobs.
-- Fix initialization of assoc_mgr_locks and slurmctld_locks lock structures.
-- Fix segfault with job arrays using X11 forwarding.
-- Revert regression caused by e0ee1c7054 which caused negative values and
  values starting with a decimal to be invalid for PriorityWeightTRES and
  TRESBillingWeight.
-- Fix possibility to update a job's reservation to none.
-- Suppress connection errors to primary slurmd when backup dbd is active.
-- Suppress connection errors to primary db when backup db kicks in.
-- Add missing fields for sacct --completion when using jobcomp/filetxt.
-- Fix incorrect values set for UserCPU, SystemCPU, and TotalCPU sacct fields
  when JobAcctGatherType=jobacct_gather/cgroup.
-- Fixed srun from double printing invalid option msg twice.
-- Remove unused -b flag from getopt call in sbatch.
-- Disable reporting of node TRES in sreport.
-- Re-enabling features combined by OR within parenthesis for non-knl setups.
-- Prevent sending duplicate requests to reboot a node before ResumeTimeout.
-- Down nodes that don't reboot by ResumeTimeout.
-- Update seff to reflect API change from rss_max to tres_usage_in_max.
-- Add missing TRES constants from perl API.
-- Fix issue where sacct would return incorrect array tasks when querying
  specific tasks.
-- Add missing variables to slurmdb_stats_t in the perlapi.
-- Fix nodes not getting reboot RPC when job requires reboot of nodes.
-- Fix failing update the partition list of a job.
-- Use slurm.conf gres ids instead of gres.conf names to get a gres type name.
-- Add mitigation for a potential heap overflow on 32-bit systems in xmalloc.
CVE-2019-6438.

* Changes in Slurm 18.08.4
==========================
-- burst_buffer/cray - avoid launching a job that would be immediately
  cancelled due to a DataWarp failure.
-- Fix message sent to user to display preempted instead of time limit when a
  job is preempted.
-- Fix memory leak when a failure happens processing a nodes gres config.
-- Improve error message when failures happen processing a nodes gres config.
-- When building rpms ignore redundant standard rpaths and insecure relative
  rpaths, for RHEL based distros which use "check-rpaths" tool.
-- Don't skip jobs in scontrol hold.
-- Avoid locking the job_list when unneeded.
-- Allow --cpu-bind=verbose to be used with SLURM_HINT environment variable.
-- Make it so fixing runaway jobs will not alter the same job requeued
  when not runaway.
-- Avoid checking state when searching for runaway jobs.
-- Remove redundant check for end time of job when searching for runaway jobs.
-- Make sure that we properly check for runaway jobs where another job might have the same id (for example, if a job was requeued) by also checking the submit time.
-- Add scontrol update job ResetAccrueTime to clear a job's time previously accrued for priority.
-- cons_res: Delay exiting cr_job_test until after cores/cpus are calculated and distributed.
-- Fix bug where binary in cwd would trump binary in PATH with test_exec.
-- Fix check to test printf("%s\n", NULL); to not require -Wno-format-truncation CFLAG.
-- Fix JobAcctGatherParams=UsePss to report the correct usage.
-- Fix minor memory leak in pmix plugin.
-- Fix minor memory leak in slurmctld when reading configuration.
-- Handle return codes correctly from pthread_* functions.
-- Fix minor memory leak when a slurmd is unable to contact a slurmctld when trying to register.
-- Fix sreport sizesbyaccount report when using Flatview and accounts.
-- Fix incorrect shift when dealing with node weights and scheduling.
-- libslurm/perl - Fix segfault caused by incorrect hv_to_slurm_ctl_conf.
-- Add qos and assoc options to confirmation dialogs.
-- Handle updating identical license or partition information correctly.
-- Makes sure accounts and QOS' are all lower case to match documentation when read in from the slurm.conf file.
-- Don't consider partitions without enough nodes in reservation, main scheduler.
-- Set SLURM_NTASKS correctly if having to determine from other options.
-- Removed GCP scripts from contribs. Now located at: https://github.com/SchedMD/slurm-gcp.
-- Don't check existence of srun --prolog or --epilog executables when set to "none" and SLURM_TEST_EXEC is used.
-- Add "P" suffix support to job and step tres specifications.
-- When doing a reconfigure handle QOS' GrpJobsAccrue correctly.
-- Remove unneeded extra parentheses from sh5util.
-- Fix jobacct_gather/cgroup to work correctly when more than one task is started on a node.
-- If requesting --ntasks-per-node with no tasks set tasks correctly.
-- Accept modifiers for TRES originally added in 6f0342e0358.
-- Don't remove reservation on slurmd restart if nodes are removed from configuration.
-- Fix bad xfree in task/cgroup.
-- Fix removing counters if a job array isn't subject to limits and is canceled while pending.
-- Make sure SLURM_NTASKS_PER_NODE is set correctly when env is overwritten by the command line.
-- Clean up step on a failed node correctly.
-- mpi/pmix: Fixed the logging of collective state.
-- mpi/pmix: Make multi-slurmd work correctly when using ring communication.
-- mpi/pmix: Fix double invocation of the PMIx lib fence callback.
-- mpi/pmix: Remove unneeded libpmix callback drop in tree-based coll.
-- Fix race condition in route/topology when the slurmd is reconfigured.
-- In route/topology validate the slurmd doesn't try to initialize the
node system.

-- Fix issue when requesting invalid gres.
-- Validate job_ptr in backfill before restoring preempt state.
-- Fix issue when job's environment is minimal and only contains variables
  Slurm is going to replace internally.
-- When handling runaway jobs remove all usage before rollup to remove any
  time that wasn't existent instead of just updating lines that have time
  with a lesser time.
-- salloc - set SLURM_NTASKS_PER_CORE and SLURM_NTASKS_PER_SOCKET in the
  environment if the corresponding command line options are used.
-- slurmd - fix handling of the -f flag to specify alternate config file
  locations.
-- Fix scheduling logic to avoid using nodes that require a reboot for KNL
  node change when possible.
-- Fix scheduling logic bug. There should have been a test for _not_
  NODE_SET_REBOOT to continue.
-- Fix a scheduling logic bug with respect to XOR operation support when there
  are down nodes.
-- If there is a constraint construct of the form "[...&...]
  then an error is generated if more than one of those specifications
  contains KNL NUMA or MCDRAM modes.
-- Fix stepd segfault race if slurmctld hasn't registered with the launching
  slurmd yet delivering it's TRES list.
-- Add SchedulerParameters option of bf_ignore_newly_avail_nodes to avoid
  scheduling lower priority jobs on resources that become available during
  the backfill scheduling cycle when bf_continue is enabled.
-- Decrement message_connections in stepd code on error path correctly.
-- Decrease an error message to be debug.
-- Fix missing suffixes in squeue.
-- pam_slurm_adopt - send an error message to the user if no Slurm jobs
  can be located on the node.
-- Run SlurmctldPrimaryOffProg when the primary slurmctld process shuts down.
-- job_submit/lua: Add several slurmctld return codes.
-- job_submit/lua: Add user/group info to jobs.
-- Fix formatting issues when printing uint64_t.
-- Bump RLIMIT_NOFILE for daemons in systemd services.
-- Expand %x in job name in 'scontrol show job'.
-- salloc/sbatch/srun - print warning if mutually exclusive options of --mem
  and --mem-per-cpu are both set.

* Changes in Slurm 18.08.3
==================================
-- Fix regression in 18.08.1 that caused dbd messages to not be queued up
  when the dbd was down.
-- Fix regression in 18.08.1 that can cause a slurmctld crash when splitting
  job array elements.

* Changes in Slurm 18.08.2
==================================
-- Correctly initialize variable in env_array_user_default().
-- Remove race condition when signaling starting step.
-- Fix issue where 17.11 job's using GRES in didn't initialize new 18.08
structures after unpack.
-- Stop removing nodes once the minimum CPU or node count for the job is
reached in the cons_res plugin.
-- Process any changes to MinJobAge and SlurmdTimeout in the slurmctld when
it is reconfigured to determine changes in its background timers.
-- Use previous SlurmdTimeout in the slurmctld after a reconfig to
determine the time a node has been down.
-- Fix multi-cluster srun between clusters with different SelectType plugins.
-- Fix removing job licenses on reconfig/restart when configured license
counts are 0.
-- If a job requested multiple licenses and one license was removed then on
a reconfigure/restart all of the licenses -- including the valid ones
would be removed.
-- Fix issue where job's license string wasn't updated after a restart when
licenses were removed or added.
-- Add allow_zero_lic to SchedulerParameters.
-- Avoid scheduling tasks in excess of ArrayTaskThrottle when canceling tasks
of an array.
-- Fix jobs that request memory per node and task count that can't be
scheduled right away.
-- Avoid infinite loop with jobacct_gather/linux when pids wrap around
/proc/sys/kernel/pid_max.
-- Fix --parsable2 output for sacct and sstat commands to remove a stray
trailing delimiter.
-- When modifying a user's name in sacctmgr enforce PreserveCaseUser.
-- When adding a coordinator or user that was once deleted enforce
PreserveCaseUser.
-- Correctly handle scenarios where a partitions MaxMemPerCPU is less than
a jobs --mem-per-cpu and also -c is greater than 1.
-- Set AccrueTime correctly when MaxJobsAccrue is disabled and BeginTime has
not been established.
-- Correctly account for job arrays for new {Max/Grp}JobsAccrue limits.

* Changes in Slurm 18.08.1
==========================
-- Remove commented-out parts of man pages related to cons_tres work in 19.05,
as these were showing up on the web version due to a syntax error.
-- Prevent slurmd performance issues in main background loop if multiple
backup controllers are unavailable.
-- Add missing user read association lock in burst_buffer/cray during init().
-- Fix incorrect spacing for PartitionName lines in 'scontrol write config'.
-- Fix creation of step hwloc xml file for after cpuset cgroup has been
created.
-- Add userspace as a valid default governor.
-- Add timers to group_cache_lookup so if going slow advise
LaunchParameters=send_gids.
-- Fix SLURM_STEPGRES=none to work correctly.
-- Fix potential memory leak when a failure happens unpacking a ctld_multi_msg.
-- Fix potential double free when a failure happens when unpacking a
node_registration_status_msg.
-- Fix sacctmgr show runaways.
-- Removed non-POSIX append operator from configure script for non-bash
-- Fix incorrect spacing for PartitionName lines in 'scontrol write config'.
-- Fix sacct to not print huge reserve times when the job was never eligible.
-- burst_buffer/cray - Add missing locks around assoc_mgr when timing out a burst buffer.
-- burst_buffer/cray - Update burst buffers when an association or qos is removed from the system.
-- Remove documentation for deprecated Cray/ALPS systems. Please switch to Native Cray mode instead.
-- Completely copy features when copying the list in the slurmd.
-- PMIX - Fix issue with packing processes when using an arbitrary task distribution.
-- Fix hostlists to be able to handle nodenames with '-' in them surrounded by integers.
-- Fix correct job CPU count allocated.
-- Fix sacctmgr setting GrpJobs limit when setting GrpJobsAccrue limit.
-- Change the defaults to MemLimitEnforce=no and NoOverMemoryKill (See RELEASE_NOTES).
-- Prevent abort when using Cray node features plugin on non-knl.
-- Add ability to reboot down nodes with scontrol reboot_nodes.
-- Protect against sending to the slurmd if the connection has gone away.
-- Fix invalid read when not using backup slurmds.
-- Prevent acct coordinators from changing default acct on add user.
-- Don't allow scontrol top do modify job priorities when priority == 1.
-- slurmsmwd - change parsing code to handle systems with the svid or inst fields set in xtconsumer output.
-- Fix infinite loop in slurmd if GRES is specified without a count.
-- sacct: Print error when unknown arguments are found.
-- Fix checking missing return codes when unpacking structures.
-- Fix slurm.spec-legacy including slurmsmwd
-- More explicit error message when cgroup oom-kill events detected.
-- When updating an association and are unable to find parent association initialize old fairshare association pointer correctly.
-- Wrap slurm_cond_signal() calls with mutexes where needed.
-- Fix correct timeout with resends in slurm_send_only_node_msg.
-- Fix pam_slurm_adopt to honor action_adopt_failure.
-- Have the slurmd recreate the hwloc xml file for the full system on restart.
-- sdiag - correct the units for the gettimeofday() stat to microseconds.
-- Set SLURM_CLUSTER_NAME environment variable in MailProg to the ClusterName.
-- small - use SLURM_CLUSTER_NAME environment variable.
-- job_submit/lua - expose argc/argv options through lua interface.
-- slurmd - prevent false-positive warning about innodb settings having been set too low if they're actually set over 2GB.

* Changes in Slurm 18.08.0

-- Fix segfault on job arrays when starting controller without dbd up.
-- Fix pmi2 to build with gcc 8.0+.
-- Remove the development snapshot of select/cons_tres plugin.
-- Fix slurmd -C to not print benign error from xcpuinfo.
-- Fix potential double locks in the assoc_mgr.
-- Fix sacct truncate flag behavior Truncated pending jobs will always
return a start and end time set to the window end time, so elapsed
time is 0.
-- Fix extern step hanging forever when canceled right after creation.
-- sdiag - add slurmctld agent count.
-- Remove requirement to have cgroup_allowed_devices_file.conf in order to
  constrain devices. By default all devices are allowed and GRES, that are
  associated with a device file, that are not requested are restricted.
-- Fix proper alignment of clauses when determining if more nodes are needed
  for an allocation.
-- Fix race condition when canceling a federation job that just started
  running.
-- Prevent extra resources from being allocated when combining certain flags.
-- Fix problem in task/affinity plugin that can lead to slurmd fatal()'ing
  when using --hint=nomultithread.
-- Fix left over socket file when step is ending and using pmi2 with
  %n or %h in the spool dir.
-- Don't remove hwloc full system xml file when shutting down the slurmd.
-- Fix segfault that could happen with a het job when it was canceled while
  starting.
-- Fix scan-build false-positive warning about invalid memory access in the
  _ping_controller() function.
-- Add control_inx value to trigger_info_msg_t to permit future work in the
  trigger management code to distinguish which of multiple backup controllers
  has changed state.

* Changes in Slurm 18.08.0rc1
==================================
-- Add TimelimitRaw sacct output field to display timelimit numbers.
-- Fix job array preemption during backfill scheduling.
-- Fix scontrol -o show assoc output.
-- Add support for sacct --whole-hetjob=[yes|no] option.
-- Make salloc handle node requests the same as sbatch.
-- Add shutdown_on_reboot SlurmdParameter to control whether the Slurmd will
  shutdown itself down or not when a reboot request is received.
-- Add cancel_reboot scontrol option to cancel pending reboot of nodes.
-- Make Users case insensitive in the database based on
  Parameters=PreserveCaseUser in the slurmd.conf.
-- Improve scheduling when dealing with node_features that could have a
  boot delay.
-- Fix issue if a step launch fails we don't get a bunch of '(null)' strings
  in the step record for usage.
-- Changed the default AuthType for slurmd to auth/munge.
-- Make it so libpmi.so doesn't link to libslurm.so.$apiversion.
-- Added 'remote-fs.target' to After directive of slurmd.service file.
-- Fix filetxt plugin to handle it when you aren't running a jobacct_gather
  plugin.
-- Remove drain on node when reboot nextstate used.
-- Speed up pack of job's qos.
-- Fix race condition when trying to update reservation in the database.
-- For the PrologFlags slurm.conf option, make NoHold mutually exclusive with
  Contain and/or X11 options.
-- Revise the handling of SlurmctldSyslogLevel and SlurmdSyslogLevel options

(continues on next page)
in slurm.conf and DebugLevelSyslog in slurmdbd.conf.
-- Gate reading the cgroup.conf file.
-- Gate reading the acct_gather_* plugins.
-- Add sacctmgr options to prevent/manage job queue stuffing:
  - GrpJobsAccrue=<max_jobs>
    Maximum number of pending jobs in aggregate able to accrue age priority for this association and all associations which are children of this association. To clear a previously set value use the modify command with a new value of -1.
  - MaxJobsAccrue=<max_jobs>
    Maximum number of pending jobs able to accrue age priority at any given time for the given association. This is overridden if set directly on a user. Default is the cluster's limit. To clear a previously set value use the modify command with a new value of -1.
  - MinPrioThreshold
    Minimum priority required to reserve resources when scheduling.

* Changes in Slurm 18.08.0pre2

-- Remove support for "ChosLoc" configuration parameter.
-- Configuration parameters "ControlMachine", "ControlAddr", "BackupController" and "BackupAddr" replaced by an ordered list of "SlurmctldHost" records with the optional address appended to the name enclosed in parenthesis. For example: "SlurmctldHost=head(12.34.56.78)". An arbitrary number of backup servers can be configured.
-- When a pending job's state includes "UnavailableNodes" do not include the nodes in FUTURE state.
-- Remove --immediate option from sbatch.
-- Add infrastructure for per-job and per-step TRES parameters: tres-per-job, tres-per-node, tres-per-socket, tres-per-task, cpus-per-tres, mem-per-tres, tres-bind and tres-freq. These new parameters are not currently used, but have been added to the appropriate RPCs.
-- Add DefCpuPerGpu and DefMemPerGpu to global and per-partition configuration parameters. Shown in scontrol/sview as "JobDefaults=...". NOTE: These options are for future use and currently have no effect.
-- Fix for setting always the correct status on job update in mysql
-- Add ValidateMode configuration parameter to knl_cray.conf for static MCDRAM/NUMA configurations.
-- Fix security issue in accounting_storage/mysql plugin by always escaping strings within the slurmdbd. CVE-2018-7033.
-- Disable local PTY output processing when using 'srun --unbuffered'. This prevents the PTY subsystem from inserting extraneous \r characters into the output stream.
-- Change the column name for the %U (User ID) field in squeue to 'UID'.
-- CRAY - Add CheckGhalQuiesce to the CommunicationParameters.
-- When a process is core dumping, avoid terminating other processes in that task group. This fixes a problem with writing out incomplete OpenMP core files.
-- CPU frequency management enhancements: If scaling_available_frequencies file is not available, then derive values from scaling_min_freq and scaling_max_freq values. If cpuinfo_cur_freq file is not available then try to use scaling_cur_freq.
-- Add pending jobs count to sdiag output.
-- Fix update job function. There were some inconsistencies on the behavior that caused time limits to be modified when swapping QoS, bad permissions check for a coordinator and AllowQOS and DenyQOS were not enforced on job update.
-- Add configuration parameters SlurmctldPrimaryOnProg and SlurmctldPrimaryOffProg, which define programs to execute when a slurmctld daemon becomes the primary server or goes from primary to backup mode.
-- Add configuration parameters SlurmctldAddr for use with virtual IP to manage backup slurmctld daemons.
-- Explicitly shutdown the slurmd process when instructed to reboot.
-- Add ability to create/update partition with TRESBillingWeights through scontrol.
-- Calculate TRES billing values at submission so that billing limits can be enforced at submission with QoS DenyOnLimit.
-- Add node_features plugin function "node_features_p_reboot_weight()" to return the node weight to be used for a compute node that requires reboot for use (e.g. to change the NUMA mode of a KNL node).
-- Add NodeRebootWeight parameter to knl.conf configuration file.
-- Fix insecure handling of job requested gid field. CVE-2018-10995.
-- Fix srun to return highest signal of any task.
-- Completely remove "gres" field from step record. Use "tres_per_node", "tres_per_socket", etc.
-- Add "Links" parameter to gres.conf configuration file.
-- Force slurm_mktime() to set tm_isdst to -1 so anyone using the function doesn't forget to set it.
-- burst_buffer.conf - Add SetExecHost flag to enable burst buffer access from the login node for interactive jobs.
-- Append ", with requeued tasks" to job array "end" emails if any tasks in the array were requeued. This is a hint to use "sacct --duplicates" to see the whole picture of the array job.
-- Add ResumeFailProgram slurm.conf option to specify a program that is called when a node fails to respond by ResumeTimeout.
-- Add new job pending reason of "ReqNodeNotAvail, reserved for maintenance".
-- Remove AdminComment += syntax from 'scontrol update job'.
-- sched/backfill: Reset job time limit if needed for deadline scheduling.
-- For heterogeneous job component with required nodes, explicitly exclude those nodes from all other job components.
-- Add name of partition used to output of srun --test-only output (valuable for jobs submitted to multiple partitions).
-- If MailProg is not configured and "/bin/mail" (the default) does not exist, but "/usr/bin/mail" does exist then use "/usr/bin/mail" as a default value.
-- sdiag output now reports outgoing slurmctld message queue contents.
-- Fix issue in performance when reading slurm conf having nodes with features.
-- Make it so the slurmd's pid file gets created before initing the database.
-- Improve escaping special characters on user commands when specifying paths.
-- Fix directory names with special char '\' that are not handled correctly.
-- Add salloc/sbatch/srun option of --gres-flags=disable-binding to disable filtering of CPUs with respect to generic resource locality. This option is currently required to use more CPUs than are bound to a GRES (i.e. if a GPU is bound to the CPUs on one socket, but resources on more than one socket
are required to run the job). This option may permit a job to be allocated
resources sooner than otherwise possible, but may result in lower job
performance.

-- SlurmDBD - Print warning if MySQL/MariaDB internal tuning is not at least
half of the recommended values.

-- Move libpmi from src/api to contribs/pmi.

-- Add ability to specify a node reason when rebooting nodes with "scontrol
reboot".

-- Add nextstate option to "scontrol reboot" to dictate state of node after
reboot.

-- Consider "resuming" (nextstate-resume) nodes as available in backfill
future scheduling and don't replace "resuming" nodes in reservations.

-- Add the use of a xml file to help performance when using hwloc.

* Changes in Slurm 18.08.0pre1

-- Add new burst buffer state of "teardown-fail" to indicate the burst buffer
teardown operation is failing on specific buffers. This changes the numeric
value of the BB_STATE_COMPLETE type. Any Slurm version 17.02 or 17.11 tool
used to report burst buffer state information will report a state of "66"
rather than "complete" for burst buffers which have been deleted, but still
exist in the slurmctld daemon's tables (a very short-lived situation).

-- Multiple backup slurmctld daemons can be configured:
  * Specify "BackupController#=<hostname> and "BackupAddr#=<address>" to
    identify up to 9 backup servers.
  * Output format of "scontrol ping" and the daemon status at the end of
    "scontrol status" is modified to report up status of the primary and all
    backup servers.
  * "scontrol takeover [#]" command can now identify the SlurmctldHost
    index number. Default value is "1" (the first backup configured
    SlurmctldHost).

-- Enable jobs with zero node count for creation and/or deletion of persistent
burst buffers.
  * The partition default MinNodes configuration parameter is now 0
    (previously 1 node).
  * Zero size jobs disabled for job arrays and heterogeneous jobs, but
    supported for salloc, sbatch and srun commands.

-- Add "scontrol show dwstat" command to display Cray burst buffer
status.

-- Add "GetSysStatus" option to burst_buffer.conf file. For burst_buffer/cray
this would indicate the location of the "dwstat" command.

-- Add node and partition configuration options of "CpuBind" to control default
task binding. Modify the scontrol to report and modify these parameters.

-- Add "NumaCpuBind" option to knl.conf file to automatically change a node's
    CpuBind parameter based upon changes to a node's NUMA mode.

-- Add sbatch "--batch" option to identify features required on batch node.
    For example "sbatch --batch=haswell ...".

-- Add "BatchFeatures" field to output of "scontrol show job".

-- Add support for "--bb" option to sbatch command.

-- Add new SystemComment field to job data structure and database. Currently
    used for Burst Buffer error logs.

-- Expand reservation "flags" field from 32 to 64 bits.

-- Add job state flag of "SIGNALING" to avoid race condition with multiple
SIGSTOP/SIGCONT signals for the same job being active at the same time.
-- Properly handle srun --will-run option when there are jobs in COMPLETING state.
-- Properly report who is signaling a step.
-- Don't combine updated reservation records in sreport's reservation report.
-- node_features plugin - Add support for XOR & XAND of job constraints (node feature specifications).
-- Add support for parenthesis in a job's constraint specification to group like options together. For example
  --constraint="[(knl&sn4&flat)*4&haswell*1]" might be used to specify that four nodes with the features "knl", "sn4" and "flat" plus one node with the feature "haswell" are required.
-- Improvements to how srun searches for the executable when using cwd.
-- Now programs can be checked before execution if test_exec is set when using multi-prog option.
-- Report NodeFeatures plugin configuration with scontrol and sview commands.
-- Add acct_gather_profile/influxdb plugin.
-- Add new job state of SO/STAGE_OUT indicating that burst buffer stage-out operation is in progress.
-- Correct SLURM_NTASKS and SLURM_NPROCS environment variable for heterogeneous job step. Report values representing full allocation.
-- Expand advanced reservation feature specification to support parenthesis and counts of nodes with specified features. Nodes with the feature currently active will be preferred.
-- Defer job signaling until prolog is completed
-- Have the primary slurmctld wait until the backup has completely shutdown before taking control.
-- Fix issue where unpacking job state after TRES count changed could lead to invalid reads.
-- Heterogeneous job step allocations supported with
  * Open MPI (with Slurm's PMI2 and PMIx plugins) and
  * Intel MPI (with Slurm's PMI2 plugin)
-- Remove redundant function arguments from task plugins:
  * Remove "job_id" field from task_p_slurmd_batch_request() function.
  * Remove "job_id" field from task_p_slurmd_launch_request() function.
  * Remove "job_id" field from task_p_slurmd_reserve_resources() function.
-- Change function name from node_features_p_changable_features() to node_features_p_changeable_feature in node_features plugin.
-- Add Slurm configuration file check logic using "slurmctld -t" command.

* Changes in Slurm 17.11.10
===========================
-- Move priority_sort_part_tier from slurmctld to libslurm to make it possible to run the regression tests 24.* without changing that code since it links directly to the priority plugin where that function isn't defined.
-- Fix issue where job time limits can increase to max walltime when updating a job with scontrol.
-- Fix invalid protocol_version manipulation on big endian platforms causing srun and sattach to fail.
-- Fix for QOS, Reservation and Alias env variables in srun.
-- mpi/pmi2 - Backport 6a702158b49c4 from 18.08 to avoid dangerous detached thread.
-- When allowing heterogeneous steps make sure we copy all the options to
   avoid copying strings that may be overwritten.
-- Print correctly when sh5util finds and empty file.
-- Fix sh5util to not seg fault on exit.
-- Fix sh5util to check correctly for H5free_memory.
-- Adjust OOM monitoring function in task/cgroup to prevent problems in
   regression suite from leaked file descriptors.
-- Fix issue with gres when defined with a type and no count
   (i.e. gres=gpu/tesla) it would get a count of 0.
-- Allow sstat to talk to slurmd's that are new in protocol version.
-- Permit database names over 33 characters in accounting_storage/mysql.
-- Fix negative values when profiling.
-- Fix srn segfault caused by invalid memory reads on the env.
-- Fix segfault on job arrays when starting controller without dbd up.
-- Fix pm2 to build with gcc 8.0+.
-- Fix proper alignment of clauses when determining if more nodes are needed
   for an allocation.
-- Fix race condition when canceling a federation job that just started
   running.
-- Prevent extra resources from being allocated when combining certain flags.
-- Fix problem in task/affinity plugin that can lead to slurmd fatal()'ing
   when using --hint=nomultithread.
-- Fix left over socket file when step is ending and using pm2 with
   %n or %h in the spool dir.
-- Fix incorrect spacing for PartitionName lines in 'scontrol write config'.
-- Fix sacct to not print huge reserve times when the job was never eligible.
-- burst_buffer/cray - Add missing locks around assoc_mgr when timing out a
   burst buffer.
-- burst_buffer/cray - Update burst buffers when an association or qos
   is removed from the system.
-- If failed over to a backup controller, ensure the agent thread is launched
   to handle deferred tasks.
-- Fix correct job CPU count allocated.
-- Protect against sending to the slurmd if the connection has gone away.
-- Fix checking missing return codes when unpacking structures.
-- Fix slurmspec-legacy including slurmsmwd
-- More explicit error message when cgroup oom-kill events detected.
-- When updating an association and are unable to find parent association
   initialize old fairshare association pointer correctly.
-- Wrap slurm_cond_signal() calls with mutexes where needed.
-- Fix correct timeout with resends in slurm_send_only_node_msg.
-- Fix pam_slurm_adopt to honor action_adopt_failure.
-- job_submit/lua - expose argc/argv options through lua interface.

* Changes in Slurm 17.11.9-2
================================
-- Fix printing of node state "drain + reboot" (and other node state flags).
-- Fix invalid read (segfault) when sorting multi-partition jobs.
-- Move several new error() messages to debug() to keep them out of users'
   srn output.

* Changes in Slurm 17.11.9
-- Fix segfault in slurmctld when a job's node bitmap is NULL during a scheduling cycle. Primarily caused by EnforcePartLimits=ALL.
-- Remove erroneous unlock in acct_gather_energy/ipmi.
-- Enable support for hwloc version 2.0.1.
-- Fix 'srun -q (--qos) option handling.
-- Fix socket communication issue that can lead to lost task completion messages, which will cause a permanently stuck srun process.
-- Handle creation of TMPDIR if environment variable is set or changed in a task prolog script.
-- Avoid node layout fragmentation if running with a fixed CPU count but without Sockets and CoresPerSocket defined.
-- burst_buffer/cray - Fix datawarp swap default pool overriding jobdw.
-- Fix incorrect job priority assignment for multi-partition job with different PriorityTier settings on the partitions.
-- Fix sinfo to print correct node state.

* Changes in Slurm 17.11.8

-- Fix incomplete RESPONSE_ [RESOURCE|JOB_PACK]_ALLOCATION building path.
-- Do not allocate nodes that were marked down due to the node not responding by ResumeTimeout.
-- task/cray plugin - search for "mems" cgroup information in the file "cpuset.mems" then fall back to the file "mems".
-- Fix ipmi profile debug uninitialized variable.
-- Improve detection of Lua package on older RHEL distributions.
-- PMIx: fixed the direct connect inline msg sending.
-- MYSQL: Fix issue not handling all fields when loading an archive dump.
-- Allow a job_submit plugin to change the admin_comment field during job_submit_plugin_modify().
-- job_submit/lua - fix access into reservation table.
-- MySQL - Prevent deadlock caused by archive logic locking reads.
-- Don't enforce MaxQueryTimeRange when requesting specific jobs.
-- Modify --test-only logic to properly support jobs submitted to more than one partition.
-- Prevent slurmctld from abort when attempting to set non-existing qos as def_qos.id.
-- Add new job dependency type of "afterburstbuffer". The pending job will be delayed until the first job completes execution and it's burst buffer stage-out is completed.
-- Reorder proctrack/task plugin load in the slurmstepd to match that of slurmd and avoid race condition calling task before proctrack can introduce.
-- Prevent reboot of a busy KNL node when requesting inactive features.
-- Revert to previous behavior when requesting memory per cpu/node introduced in 17.11.7.
-- Fix to reinitialize previously adjusted job members to their original value when validating the job memory in multi-partition requests.
-- Fix _step_signal() from always returning SLURM_SUCCESS.
-- Combine active and available node feature change logs on one line rather than one line per node for performance reasons.
-- Prevent occasionally leaking freezer cgroups.
-- Fix potential segfault when closing the mpi/mpi2 plugin.
-- Fix issues with --exclusive=[user|mcs] to work correctly with preemption or when job requests a specific list of hosts.
-- Make code compile with hdf5 1.10.2+
-- mpi/pmix: Fixed the collectives canceling.
-- SlurmDBD: improve error message handling on archive load failure.
-- Fix incorrect locking when deleting reservations.
-- Fix incorrect locking when setting up the power save module.
-- Fix setting format output length for squeue when showing array jobs.
-- Add xstrstr function.
-- Fix printing out of --hint options in sbatch, salloc --help.
-- Prevent possible divide by zero in _validate_time_limit().
-- Add Delegate=yes to the slurmd.service file to prevent systemd from interfering with the jobs' cgroup hierarchies.
-- Change the backlog argument to the listen() syscall within srun to 4096 to match elsewhere in the code, and avoid communication problems at scale.

* Changes in Slurm 17.11.7
==========================
-- Fix for possible slurmd daemon abort with NULL pointer.
-- Fix different issues when requesting memory per cpu/node.
-- PMIx - override default paths at configure time if --with-pmix is used.
-- Have sprio display jobs before eligible time when PriorityFlags=ACCRUE_ALWAYS is set.
-- Make sure locks are always in place when calling _post_qos_list().
-- Notify srun and cctd when unkillable stepd exits.
-- Fix slurmstepd deadlock in stepd cleanup caused by race condition in the jobacct_gather fini() interfaces introduced in 17.11.6.
-- Fix slurmstepd deadlock in PMIx startup.
-- task/cgroup - fix invalid free() if the hwloc library does not return a string as expected.
-- Fix insecure handling of job requested gid field. CVE-2018-10995.

* Changes in Slurm 17.11.6
==========================
-- CRAY - Add slurmsmwd to the contribs/cray dir.
-- sview - fix crash when closing any search dialog.
-- Fix initialization of variable in stepd when using native x11.
-- Fix reading slurm_io_init_msg to handle partial messages.
-- Fix scontrol create res segfault when wrong user/account parameters given.
-- Fix documentation for sacct on parameter -X (--allocations)
-- Change TRES Weights debug messages to debug3.
-- FreeBSD - assorted fixes to restore build.
-- Fix for not tracking environment variables from unrelated different jobs.
-- PMIX - Added the direct connect authentication.
   When upgrading this may cause issues with jobs using pmix starting on mixed slurmstepd versions where some are less than 17.11.6.
-- Prevent the backup slurmd daemon from losing the active/available node features list on takeover.
-- Add documentation for fix IDLE*+POWER due to capmc stuck in Cray systems.
-- Fix missing mutex unlock when prolog is failing on a node, leading to a hung slurmd.
-- Fix locking around Cray CCM prolog/epilog.
-- Add missing fed_mgr read locks.
-- Fix issue incorrectly setting a job time_start to 0 while requeuing.
-- smail - remove stray '-s' from mail subject line.
-- srun - prevent segfault if ClusterName setting is unset but SLURM_WORKING_CLUSTER environment variable is defined.
-- In configurator.html web pages change default configuration from task/none to task/affinity plugin and from select/linear plugin to select/cons_res plus CR_Core.
-- Allow jobs to run beyond a FLEX reservation end time.
-- Fix problem with wrongly set as Reservation job state_reason.
-- Prevent bit_ffs() from returnig value out of bitmap range.
-- Improve performance of 'squeue -u' when PrivateData=jobs is enabled.
-- Make UnavailableNodes value in job reason be correct for each job.
-- Fix 'squeue -o %s' on Cray systems.
-- Fix incorrect error thrown when cancelling part of a job array.
-- Fix error code and scheduling problem for --exclusive=[user|mcs].
-- Fix build when l24 is in a non-standard location.
-- Be able to force power_down of cloud node even if in power_save state.
-- Allow cloud nodes to be recognized in Slurm when booted out of band.
-- Fixes race condition in _pack_job_gres() when is called multiple times.
-- Increase duration of "sleep" command used to keep extern step alive.
-- Remove unsafe usage of pthread_cancel in slurmstepd that can lead to deadlock in glibc.
-- Fix total TRES Billing on partitions.
-- Don't tear down a BB if a node fails and --no-kill or resize of a job happens.
-- Remove unsafe usage of pthread_cancel in pmix plugin that can lead to deadlock in glibc.
-- Fix fatal in controller when loading completed trigger
-- Ignore reservation overlap at submission time.
-- GRES type model and QOS limits documentation added
-- slurmd - fix ABRT on SIGINT after reconfigure with MemSpecLimit set.
-- PMIx - move two error messages on retry to debug level, and only display the error after the retry count has been exceeded.
-- Increase number of tries when sending responses to srun.
-- Fix checkpointing requeued/completing jobs in a bad state which caused a segfault on restart.
-- Fix srun on ppc64 platforms.
-- Prevent slurmd from starting steps if the Prolog returns an error when using PrologFlags=alloc.
-- priority/multifactor - prevent segfault running sprio if a partition has just been deleted and PriorityFlags=CALCULATE_RUNNING is turned on.
-- job_submit/lua - add ESLURM_INVALID_TIME_LIMIT return code value.
-- job_submit/lua - print an error if the script calls log.user in job_modify() instead of returning it to the next submitted job erroneously.
-- select/linear - handle job resize correctly.
-- select/cons_res - improve handling of --cores-per-socket requests.

* Changes in Slurm 17.11.5

-- Fix cloud nodes getting stuck in DOWN+POWER_UP+NO_RESPOND state after not responding by ResumeTimeout.

(continues on next page)
-- Add job's array_task_cnt and user_name along with partitions
  [max|def]_mem_per_[cpu|node], max_cpus_per_node, and max_share with the
  SHARED_FORCE definition to the job_submit/lua plugin.
-- srun - fix for SLURM_JOB_NUM_NODES env variable assignment.
-- sacctmgr - fix runaway jobs identification.
-- Fix for setting always the correct status on job update in mysql.
-- Fix issue if running with an association manager cache (slurmdbd was down
  when slurmctld was started) you could loose QOS usage information.
-- CRAY - Fix spec file to work correctly.
-- Set scontrol exit code to 1 if attempting to update a node state to DRAIN
  or DOWN without specifying a reason.
-- Fix race condition when running with an association manager cache
  (slurmdbd was down when slurmctld was started).
-- Print out missing SLURM_PERSIST_INIT slurmdbd message type.
-- Fix two build errors related to use of the O_CLOEXEC flag with older glibc.
-- Add Google Cloud Platform integration scripts into contribs directory.
-- Fix minor potential memory leak in backfill plugin.
-- Add missing node flags (maint/power/etc) to node states.
-- Fix issue where job time limits may end up at 1 minute when using the
  NoReserve flag on their QOS.
-- Fix security issue in accounting_storage/mysql plugin by always escaping
  strings within the slurmdbd. CVE-2018-7033.
-- Soften messages about best_fit topology to debug2 to avoid alarm.
-- Fix issue in sreport reservation utilization report to handle more
  allocated time than 100% (Flex reservations).
-- When a job is requesting a Flex reservation prefer the reservation's nodes
  over any other nodes.
* Changes in Slurm 17.11.4
==========================
-- Add fatal_abort() function to be able to get core dumps if we hit an
  "impossible" edge case.
-- Link slurmd against all libraries that slurmdstepd links to.
-- Fix limits enforce order when they're set at partition and other levels.
-- Add slurm_load_single_node() function to the Perl API.
-- slurm.spec - change dependency for --with lua to use pkgconfig.
-- Fix small memory leaks in node_features plugins on reconfigure.
-- slurmdbd - only permit requests to update resources from operators or
  administrators.
-- Fix handling of partial writes in io_init_msg_write_to_fd() which can
  lead to job step launch failure under higher cluster loads.
-- MYSQL - Fix to handle quotes in a given work_dir of a job.
-- sbcast - fix a race condition that leads to "Unspecified error".
-- Log that support for the ChosLoc configuration parameter will end in Slurm
  version 18.08.
-- Fix backfill performance issue where bf_min_prio_reserve was not respected.
-- Fix MaxQueryTimeRange checks.
-- Print MaxQueryTimeRange in "sacctmgr show config".
-- Correctly check return codes when creating a step to check if needing to
  wait to retry or not.
-- Fix issue where a job could be denied by Reason=MaxMemPerLimit when not
  requesting any tasks.
-- In perl tools, fix for regexp that caused extra incorrectly shown results.
-- Add some extra locks in fed_mgr to be extra safe.
-- Minor memory leak fixes in the fed_mgr on slurmctld shutdown.
-- Make sreport job reports also report duplicate jobs correctly.
-- Fix issues restoring certain Partition configuration elements, especially when ReconfigFlags=KeepPartInfo is enabled.
-- Don’t add TRES whose value is NO_VAL64 when building string line.
-- Fix removing array jobs from hash in slurmctld.
-- Print out missing user messages from jobsubmit plugin when srun/salloc are waiting for an allocation.
-- Handle --clusters=all as case insensitive.
-- Only check requested clusters in federation when using --test-only submission option.
-- In the federation, make it so you can cancel stranded sibling jobs.
-- Silence an error from PSS memory stat collection process.
-- Requeue jobs allocated to nodes requested to DRAIN or FAIL if nodes are POWER_SAVE or POWER_UP, preventing jobs to start on NHC-failed nodes.
-- Make MAINT and OVERLAP reservation flags order agnostic on overlap test.
-- Preserve node features when slurmctld daemons reconfigured including active and available KNL features.
-- Prevent creation of multiple io_timeout threads within srun, which can lead to fatal() messages when those unexpected and additional mutexes are destroyed when srun shuts down.
-- burst_buffer/cray - Prevent use of "#DW create_persistent" and "#DW destroy_persistent" directives available in Cray CLE6.0UP06. This will be supported in Slurm version 18.08. Use "#BB" directives until then.
-- Fix task/cgroup affinity to behave correctly.
-- FreeBSD - fix build on systems built with WITHOUT_KERBEROS.
-- Fix to restore pn_min_memory calculated result to correctly enforce MaxMemPerCPU setting on a partition when the job uses --mem.
-- slurmdbd - prevent infinite loop if a QOS is set to preempt itself.
-- Fix issue with log rotation for slurmstepd processes.

* Changes in Slurm 17.11.3-2
==========================
-- Revert node_features changes in 17.11.3 that lead to various segfaults on slurmctld startup.

* Changes in Slurm 17.11.3
==========================
-- Send SIG_UME correctly to a step.
-- Sort sreport's reservation report by cluster, time_start, resv_name instead of cluster, resv_name, time_start.
-- Avoid setting node in COMPLETING state indefinitely if the job initiating the node reboot is cancelled while the reboot in in progress.
-- Scheduling fix for changing node features without any NodeFeatures plugins.
-- Improve logic when summarizing job arrays mail notifications.
-- Add scontrol -F/--future option to display nodes in FUTURE state.
-- Fix REASONABLE_BUF_SIZE to actually be 3/4 of MAX_BUF_SIZE.
-- When a job array is preemting make it so tasks in the array don't wait to preempt other possible jobs.
-- Change free_buffer to FREE_NULL_BUFFER to prevent possible double free

(continues on next page)
in slurmctld.

-- node_feature/knl_cray - Fix memory leaks that occur when slurmctld
reconfigured.

-- node_feature/knl_cray - Fix memory leak that can occur during normal
operation.

-- Fix srun environment variables for --prolog script.

-- Fix job array dependency with "aftercorr" option and some task arrays in
the first job fail. This fix lets all task array elements that can run
proceed rather than stopping all subsequent task array elements.

-- Fix potential deadlock in the slurmctld when using list_for_each.

-- Fix for possible memory corruption in srun when running heterogeneous job
steps.

-- Fix job array dependency with "aftercorr" option and some task arrays in
the first job fail. This fix lets all task array elements that can run
proceed rather than stopping all subsequent task array elements.

-- Fix output file containing "%t" (task ID) for heterogeneous job step to
be based upon global task ID rather than task ID for that component of the
heterogeneous job step.

-- MYSQL - Fix potential abort when attempting to make an account a parent of
itself.

-- Fix potentially uninitialized variable in slurmctld.

-- MYSQL - Fix issue for multi-dimensional machines when using sacct to
find jobs that ran on specific nodes.

-- Reject --acctg-freq at submit if invalid.

-- Added info string on sh5util when deleting an empty file.

-- Correct dragonfly topology support when job allocation specifies desired
switch count.

-- Fix minor memory leak on an sbcast error path.

-- Fix issues when starting the backup slurmd.

-- Revert uid check when requesting a jobid from a pid.

-- task/cgroup - add support to detect OOM_KILL cgroup events.

-- Fix whole node allocation cpu counts when --hint=nomultihtread.

-- Allow execution of task prolog/epilog when uid has access
rights by a secondary group id.

-- Validate command existence on the srun *[pro|epi]log options
if LaunchParameter test_exec is set.

-- Fix potential memory leak if clean starting and the TRES didn't change
from when last started.

-- Fix for association MaxWall enforcement when none is given at submission.

-- Add a job's allocated licenses to the [Pro|Epi]logSlurmctld.

-- burst_buffer/cray: Attempts by job to create persistent burst buffer when
one already exists owned by a different user will be logged and the job
held.

-- CRAY - Remove race in the core_spec where we add the slurmstepd to the
job container where if the step was canceled would also cancel the stepd
erroneously.

-- Make sure the slurmstepd blocks signals like SIGTERM correctly.

-- SPANK - When slurm_spank_init_post_opt() fails return error correctly.

-- When revoking a sibling job in the federation we want to send a start
message before purging the job record to get the uid of the revoked job.

-- Make JobAcctGatherParams options case-insensitive. Previously, UsePss
was the only correct capitalization; UsePSS or usepss were silently
ignored.
-- Prevent pthread_atfork handlers from being added unnecessarily after
'scontrol reconfigure', which can eventually lead to a crash if too
many handlers have been registered.
-- Better debug messages when MaxSubmitJobs is hit.
-- Docs - update squeue man page to describe all possible job states.
-- Prevent orphaned step_extern steps when a job is cancelled while the
prolog is still running.

* Changes in Slurm 17.11.2
====================================
-- jobcomp/elasticsearch - append Content-Type to the HTTP header.
-- MYSQL - Fix potential abort of slurmd when job has no TRES.
-- Add advanced reservation flag of "REPLACE_DOWN" to replace DOWN or DRAINED
nodes.
-- slurum.spec-legacy - add missing libslurmfull.so to slurm.files.
-- Fix squeue job ID filtering for pending job array records.
-- Fix potential deadlock in _run_prog() in power save code.
-- MYSQL - Add dynamic_offset in the database to force range for auto
increment ids for the tres_table.
-- MYSQL - Fix fallout from MySQL auto increment bug, see RELEASE_NOTES,
only affects current 17.11 users tracking licenses or GRES in the database.
-- Refactor logging logic to avoid possible memory corruption on non-x86
architectures.
-- Fix memory leak when getting jobs from the slurmd.
-- Fix incorrect logic behind MemorySwappiness, and only set the value when
specified in the configuration.

* Changes in Slurm 17.11.1-2
====================================
-- MYSQL - Make index for pack_job_id

* Changes in Slurm 17.11.1
====================================
-- Fix --with-shared-libslurm option to work correctly.
-- Make it so only daemons log errors on configuration option duplicates.
-- Fix for ConstrainDevices=yes to work correctly.
-- Fix to purge old jobs using burst buffer if slurmd daemon restarted
after the job's burst buffer work was already completed.
-- Make logging prefix for slurmdstep to happen as soon as possible.
-- mpi/pmix: Fix the job registration for the PMIx v2.1.
-- Fix uid check for signaling a step with anything but SIGKILL.
-- Return ESLURM_TRANSITION_STATE_NO_UPDATE instead of EAGAIN when trying to
signal a step that is still running a prolog.
-- Update Cray slurm_playbook.yaml with latest recommended version.
-- Only say a prolog is done running after the extern step is launched.
-- Wait to start a batch step until the prolog and extern step are
fully ran/launched. Only matters if running with
PrologFlags=[contain|alloc].
-- Truncate a range for SlurmctlPort to FD_SETSIZE elements and throw an
error, otherwise network traffic may be lost due to poll() not detecting
traffic.

(continues on next page)
-- Fix for srun --pack-group option that can reuse/corrupt memory.
-- Fix handling ultra long hostlists in a hostfile.
-- X11: fix xauth regex to handle '-' in hostnames again.
-- Fix potential node reboot timeout problem for "scontrol reboot" command.
-- Add ability for squeue to sort jobs by submit time.
-- CRAY - Switch to standard pid files on Cray systems.
-- Update jobcomp records on duplicate inserts.
-- If unrecognized configuration file option found then print an appropriate
  fatal error message rather than relying upon random errno value.
-- Initialize job_desc_msg_t's instead of just memseting them.
-- Fix divide by zero when job requests no tasks and more memory than
MaxMemPer{CPU|NODE}.
-- Avoid changing Slurm internal errno on syslog() failures.
-- BB - Only launch dependent jobs after the burst buffer is staged-out
  completely instead of right after the parent job finishes.
-- node_features/knl_generic - If plugin can not fully load then do not spawn
  a background pthread (which will fail with invalid memory reference).
-- Don't set the next jobid to give out to the highest jobid in the system on
  controller startup. Just use the checkpointed next use jobid.
-- Docs - add Slurm/PMIx and OpenMPI build notes to the mpi_guide page.
-- Add lustre_no_flush option to LaunchParameters for Native Cray systems.
-- Fix rpmbuild issue with rpm 4.13+ / Fedora 25+.
-- sacct - fix the display for the NNodes field when using the --units option.
-- Prevent possible double-xfree on a buffer in stepd_completion.
-- Fix for record job state on successful allocation but failed reply message.
  (Which is the default behavior if PrologFlags=send_gids is not enabled.)
  This prevents job launch problems for sites using UsePAM=1.
-- Handle syncing federated jobs that ran on non-origin clusters and were
  cancelled while the origin cluster was down.
-- Fix accessing variable outside of lock.
-- slurm.spec: move libpmi to a separate package to solve a conflict with the
  version provided by PMIx. This will require a separate change to PMIx as
  well.
-- X11 forwarding: change xauth handling to use hostname/unix:display format,
  rather than localhost:display.
-- mpi/pmix - Fix warning if not compiling with debug.

* Changes in Slurm 17.11.0

==========================
-- Fix documentation for MaxQueryTimeRange option in slurmd.conf.
-- Avoid srun abort trying to run on heterogeneous job component that has
  ended.
-- Add SLURM_PACK_JOB_ID,SLURM_PACK_JOB_OFFSET to PrologSlurmctld and
  EpilogSlurmctld environment.
-- Treat ":;" in #SBATCH arguments as fatal error. The "#SBATCH packjob" syntax
  must be used instead.
-- job_submit/lua plugin: expose pack_job fields to get.
-- Prevent scheduling deadlock with multiple components of heterogeneous job
  in different partitions (i.e. one heterogeneous job component is higher
  priority in one partition and another component is lower priority in a
  different partition).
-- Fix for heterogeneous job starvation bug.
-- Fix some slurmctld memory leaks.
-- Add SLURM_PACK_JOB_NODELIST to PrologSlurmctld and EpilogSlurmctld
environment.
-- If PrologSlurmctld fails for pack job leader then requeue or kill all
components of the job.
-- Fix for multiple --pack-group srun arguments given out of order.
-- Update slurm.conf(5) man page with updated example logrotate script.
-- Add SchedulerParameters=whole_pack configuration parameter. If set, then
hold, release and cancel operations on any component of a heterogeneous job
will be applied to all components
-- Handle FQDNs in xauth cookies for x11 display forwarding properly.
-- For heterogeneous job steps, the srun --open-mode option default value will
set to "append".
-- Pack job scheduling list not being cleared between runs of the backfill
scheduler resulted in various anomalies.
-- Fix that backward compat for pmix version < 1.1.5.
-- Fix use-after-free that can lead to slurmstepd segfaulting when setting
ulimit values.
-- Add heterogeneous job start data to sdiag output.
-- X11 forwarding - handle systems with X11UseLocalhost=no set in sshd_config.
-- Fix potential missing issue with missin symbols in gres plugins.
-- Ignore querying clusters in federation that are down from status commands.
-- Base federated jobs off of origin job and not the local cluster in API.
-- Remove erroneous double '-' on path for libslurmfull.
-- Remove version from libslurmfull and move it to $LIBDIR/slurm since the ABI
could change from one version to the other.
-- Fix unused wall time for reservations.
-- Convert old reservation records to insert unused wall into the rows.
-- slurm.spec: further restructuring and improvements.
-- Allow nodes state to be updated between FAIL and DRAIN.
-- x11 forwarding: handle build with alternate location for libssh2.

* Changes in Slurm 17.11.0rc3
==============================
-- Fix extern step to wait until launched before allowing job to start.
-- Add missing locks around figuring out TRES when clean starting the
slurmctld.
-- Cray modulefile: avoid removing /usr/bin from path on module unload.
-- Make reoccurring reservations show up in the database.
-- Adjust related resources (cpus, tasks, gres, mem, etc.) when updating
NumNodes with scontrol.
-- Don't initialize MPI plugins for batch or extern steps.
-- slurm.spec - do not install a slurm.conf file under /etc/ld.so.conf.d.
-- X11 forwarding - fix keepalive message generation code.
-- If heterogeneous job step is unable to acquire MPI reserved ports then
avoid referencing NULL pointer. Retry assigning ports ONLY for
non-heterogeneous job steps.
-- If any acct_gather_*_init fails fatal instead of error and keep going.
-- launch/slurm plugin - Avoid using global variable for heterogeneous job
steps, which could corrupt memory.

(continues from next page)
* Changes in Slurm 17.11.0rc2
=======================================
-- Prevent slurmctld abort with NodeFeatures=knl_cray and non-KNL nodes lacking
any configured features.
-- The --cpu_bind and --mem_bind options have been renamed to --cpu-bind
and --mem-bind for consistency with the rest of Slurm's options. Both
old and new syntaxes are supported for now.
-- Add slurmd_connection_commit to the slurmd api to commit when needed.
-- Add the federation api's to the slurmd.h file.
-- Add job functions to the db_api.
-- Fix sacct to always use the db_api instead of sometimes calling functions
directly.
-- Fix sacctmgr to always use the db_api instead of sometimes calling functions
directly.
-- Fix sreport to always use the db_api instead of sometimes calling functions
directly.
-- Make global uid to the db_api to minimize calls to getuid().
-- Add support for HWLOC version 2.0.
-- Added more validation logic for updates to node features.
-- Added node_features_p_node_update_valid() function to node_features plugin.
-- If a job is held due to bad constraints and a node's features change then
test the job again to see if it can run with the new features.
-- Added node_features_p_changeable_feature() function to node_features plugin.
-- Avoid rebooting a node if a job's requested feature is not under the control
of the node_features plugin and is not currently active.
-- node_features/knl_generic plugin: Do not clear a node's non-KNL features
specified in slurm.conf.
-- Added SchedulerParameters configuration option "disable_hetero_steps" to
disable job steps that span multiple components of a heterogeneous job.
Disabled by default except with mpi/none plugin. This limitation to be
removed in Slurm version 18.08.

* Changes in Slurm 17.11.0rc1
=======================================
-- Added the following jobcomp/script environment variables: CLUSTER,
   DEPENDENCY, DERIVED_EC, EXITCODE, GROUPNAME, QOS, RESERVATION, USERNAME.
The format of LIMIT (job time limit) has been modified to D-HH:MM:SS.
-- Fix QOS usage factor applying to individual TRES run minute usage.
-- Print numbers using exponential format if required to fit in allocated
   field width. The sacctmgr and sshare commands are impacted.
-- Make it so a backup DBD doesn't attempt to create database tables and
   relies on the primary to do so.
-- By default have Slurm dynamically link to libslurm.so instead of static
   linking. If static linking is desired configure with
   --without-shared-libslurm.
-- Change --workdir in sbatch to be --chdir as in all other commands (salloc,
   srun).
-- Add WorkDir to the job record in the database.
-- Make the UsageFactor of a QOS work when a qos has the nodecay flag.
-- Add MaxQueryTimeRange option to slurmd.conf to limit accounting query
   ranges when fetching job records.
-- Add LaunchParameters=batch_step_set_cpu_freq to allow the setting of the cpu
(continues on next page)
frequency on the batch step.
- CRAY - Fix statically linked applications to CRAY's PMI.
- Fix - Raise an error back to the user when trying to update currently unsupported core-based reservations.
- Do not print TmpDisk space as part of 'slurmd -C' line.
- Fix to test MaxMemPerCPU/Node partition limits when scheduling, previously only checked on submit.

- Work for heterogeneous job support (complete solution in v17.11):
  * Set SLURM_PROCID environment variable to reflect global task rank (needed by MPI).
  * Set SLURM_NTASKS environment variable to reflect global task count (needed by MPI).
  * In srun, if only some steps are allocated and one step allocation fails, then delete all allocated steps.
  * Get SPANK plugins working with heterogeneous jobs. The spank_init_post_opt() function is executed once per job component.
  * Modify sbcast command and srun's --bcast option to support heterogeneous jobs.
  * Set more environment variables for MPI: SLURM_GTIDS and SLURM_NODEID.
  * Prevent a heterogeneous job allocation from including the same nodes in multiple components (required by MPI jobs spanning components).
  * Modify step create logic so that call components of a heterogeneous job launched by a single srun command have the same step ID value.

- Modify output of "--mpi=list" to avoid duplicates for version numbers in mpi/pmix plugin names.
- Allow nodes to be rebooted while in a maintenance reservation.
- Show nodes as down even when nodes are in a maintenance reservation.
- Harden the slurmctld HA stack to mitigate certain split-brain issues.

- Work for heterogeneous job support (complete solution in v17.11):
  * Add burst buffer support.
  * Remove srun's --mpi-combine option (always combined).
  * Add SchedulerParameters configuration option "enable_hetero_steps" to enable job steps that span multiple components of a heterogeneous job. Disabled by default as most MPI implementations and Slurm configurations are not currently supported. Limitation to be removed in Slurm version 18.08.
  * Synchronize application launch across multiple components with debugger.
  * Modify slurm_kill_job_step() to cancel all components of a heterogeneous job step (used by MPI).
  * Set SLURM_JOB_NUM_NODES environment variable as needed by MVAPICH.
  * Base time limit upon the time that the latest job component is available (after all nodes in all components booted and ready for use).

- Add cluster name to smail tool email header.
- Speedup arbitrary distribution algorithm.
- Modify "srun --mpi=list" output to match valid option input by removing the "mpi/" prefix on each line of output.
- Automatically set the reservation's partition for the job if not the cluster default.
- mpi/pmi2 plugin - vestigial pointer could be referenced at shutdown with invalid memory reference resulting.
- Fix to _is_gres_cnt_zero() return false for improper input string
- Cleanup all pthread_create calls and replace with new slurm_thread_create
macro.
-- Removed obsolete MPI plugins. Remaining options are openmpi, pmi2, pmix.
-- Removed obsolete checkpoint/poe plugin.
-- Process spank environment variable options before processing spank command
   line options. Spank plugins should be able to handle option callbacks being
   called multiple times.
-- Add support for specialized cores with task/affinity plugin (previously
   only supported with task/cgroup plugin).
-- Add "TaskPluginParam=SlurmdOffSpec" option that will prevent the Slurm
   compute node daemons (slurmd and slurmstepd) from executing on specialized
   cores.
-- CRAY - Make native mode default, use --disable-native-cray to use ALPS
   instead of native Slurm.
-- Add ability to prevent suspension of some count of nodes in a specified
   range using the SuspendExcNodes configuration parameter.
-- Add SLURM_WCKEY to PrologSlurmctld and EpilogSlurmctld environment.
-- Return user response string in response to successful job allocation request
   not only on failure. Set in LUA using function 'slurm.user_msg("STRING")'.
-- Add 'scontrol write batch_script <jobid>' command to retrieve the batch
   script for a given job.
-- Remove option to display the batch script as part of 'scontrol show job'.
-- On native Cray system the configured RebootProgram is executed on the
   head node by the slurmctld daemon rather than by the slurmd daemons on the
   compute nodes. The "capmc_resume" program from "contribs/cray" can be used.
-- Modify "scontrol top" command to accept a comma separated list of job IDs
   as an argument rather than a single job ID.
-- Add MemorySwappiness value to cgroup.conf.
-- Add new "billing" TRES which allows jobs to be limited based on the job's
   billable TRES calculated by the job's partition's TRESBillingWeights.
-- sbatch - force line-buffered output so 'sbatch -W' returns the jobid
   over a piped output immediately.
-- Regular user use of "scontrol top" command is now disabled. Use the
   configuration parameter "SchedulerParameters=enable_user_top" to enable
   that functionality. The configuration parameter
   "SchedulerParameters=disable_user_top" will be silently ignored.
-- Add -TALL to sreport.
-- Removed unused SlurmdPlugstack option and associated framework.
-- Correct logic for line continuation in srun --multi-prog file.
-- Add DBD Agent queue size to sdiag output.
-- Add running job count to sdiag output.
-- Print unix timestamps next to ASCII timestamps in sdiag output.
-- In a job allocation spanning KNL and non-KNL nodes and requiring a reboot,
   do not attempt to set default NUMA or MCDRAM modes on non-KNL nodes.
-- Change default to let pending jobs run outside of reservation after
   reservation is gone to put jobs in held state. Added NO_HOLD_JOBS_AFTER_END
   reservation flag to use old default.
-- When creating a reservation, validate the CoreCnt specification matches
   the number of nodes listed.
-- When creating a reservation, correct logic to ignoring job allocations on
   request.
-- Deprecate BLCR plugin, and do not build by default.
-- Change sreport report titles from "Use" to "Usage"
* Changes in Slurm 17.11.0pre2

-- Initial work for heterogeneous job support (complete solution in v17.11):
* Modified salloc, sbatch and srun commands to parse command line, job
  script and environment variables to recognize requests for heterogeneous
  jobs. Same commands also modified to set environment variables describing
  each component of the heterogeneous job.
* Modified job allocate, batch job submit and job "will-run" requests to
  pass a list of job specifications and get a list of responses.
* Modify slurmctld daemon to process a heterogeneous job request and create
  multiple job records as needed.
* Added new fields to job record: pack_job_id, pack_job_offset and
  pack_job_set (set of job IDs). Added to slurmctld state save/restore
  logic and job information reported.
* Display new job fields in "scontrol show job" output.
* Modify squeue command to display heterogeneous job records using "#++" format. The squeue --job=# output lists all components of a heterogeneous
  job.
* Modify scancel logic to cancel all components of a heterogeneous job with
  a single request/RPC.
* Configuration parameter DebugFlags value of "HeteroJobs" added.
* Job requeue and suspend/resume modified to operate on all components of
  a heterogeneous job with a single request/RPC.
* New web page added to describe heterogeneous jobs.
* Descriptions of new API added to man pages.
* Modified email notifications to only operate on the first job component.
* Purge heterogeneous job records at the same time and not by individual
  components.
* Modified logic for heterogeneous jobs submitted to multiple clusters
  ("--clusters=...") so the job will be routed to the cluster that is
  expected to start all components earliest.
* Modified srun to create multiple job steps for heterogeneous job allocations.
* Modified launch plugin to accept a pointer to job step options structure
  rather than work from a single/common data structure.
-- Improve backfill scheduling algorithm with respect to starting jobs as soon
  as possible while avoiding advanced reservations.
-- Add URG as an option to 'scancel --signal'.
-- Check if the buffer returned from slurm_persist_msg_pack() isn't NULL.
-- Modify all daemons to re-open log files on receipt of SIGUSR2 signal. This
  is much than using SIGHUP to re-read the configuration file and rebuild
  various tables.
-- Add PrivateData=events configuration parameter
-- Work for heterogeneous job support (complete solution in v17.11):
  * Add pointer to job option structure to job_step_create_allocation() 
    function used by srun.
  * Parallelize task launch for heterogeneous job allocations (initial work).
  * Make packjobid, packjoboffset, and packjobidset fields available in squeue
    output.
  * Modify smap command to display heterogeneous job records using "#++" format.

(continues on next page)
* Add srun --pack-group and --mpi-combine options to control job step launch behaviour (not fully implemented).
* Add pack job component ID to srun --label output (e.g. "P0 1:" for job component 0 and task 1).
* jobcomp/elasticsearch: Add pack_job_id and pack_job_offset fields.
* svie: Modified to display pack job information.
* Major re-write of task state container logic to support for list of containers rather than one container per srun command.
* Add some regression tests.
* Add srun pack job environment variables when performing job allocation.

-- Set Reason=dependency over Reason=JobArrayTaskLimit for pending jobs.
-- Add slurm.conf configuration parameters SlurmctldSyslogDebug and SlurmdSyslogDebug to control which messages from the slurmctld and slurmd daemons get written to syslog.
-- Add slurmd.db.conf configuration parameter DebugLevelSyslog to control which messages from the slurmd daemon get written to syslog.
-- Fix handling of GroupUpdateForce option.
-- Work for heterogeneous job support (complete solution in v17.11):
  * Add support to sched/backfill for concurrent allocation of all pack job components including support of --time-min option.
  * Defer initiation of a heterogeneous job until a components can be started at the same time, taking into consideration association and QOS limits for the job as a whole.
  * Perform limit check on heterogeneous job as a whole at submit time to reject jobs that will never be able to run.
  * Add pack_job_id and pack_job_offset to accounting database.
  * Modified sacct to accept pack job ID specification using "##" notation.
  * Modified sstat to accept pack job ID specification using "##" notation.
-- Clear a job's "wait reason" value of BeginTime after that time has passed. Previously a reason of "BeginTime" could be reported long after the job's requested begin time had passed.
-- Split group_info in slurm_ctl_conf_t into group_force and group_time.
-- Work for heterogeneous job support (complete solution in v17.11):
  * Fix I/O race condition on step termination for srun launching multiple pack job groups.
  * If prolog is running when attempting to signal a step, then return EAGAIN and retry rather than simply returning SLURM_ERROR and aborting.
  * Modify launch/slurm plugin to signal all components of a pack job rather than just the one (modify to use a list of step context records).
  * Add logic to support srun --mpi-combine option.
  * Set up debugger data structures.
  * Disable cancellation of individual component while the job is pending.
  * Modify scontrol job hold/release and update to operate with heterogeneous job id specification (e.g. "scontrol hold 123+4").
  * If srun lacks application specification for some component, the next one specified will be used for earlier components.

* Changes in Slurm 17.11.0pre1

============================================
-- Interpet all format options in output/error file to log prolog errors. Prior logic only supported "%j" (job ID) option.
-- Add the configure option --with-shared-libslurm which will link to

(continues on next page)
libslurm.so instead of libslurm.o thus reducing the footprint of all the binaries.

-- In switch plugin, added plugin_id symbol to plugins and wrapped
  switch_jobinfo_t with dynamic_plugin_data_t in interface calls in
  order to pass switch information between clusters with different switch
types.

-- Switch naming of acct_gather_infiniband to acct_gather_interconnect

-- Make it so you can "stack" the interconnect plugins.

-- Add a last_sched_eval timestamp to record when a job was last evaluated
  by the main scheduler or backfill.

-- Add scancel "--hurry" option to avoid staging out any burst buffer data.

-- Simplify the sched plugin interface.

-- Add new advanced reservation flags of "weekday" (repeat on each weekday;
  Monday through Friday) and "weekend" (repeat on each weekend day; Saturday
  and Sunday).

-- Add new advanced reservation flag of "flex", which permits jobs requesting
  the reservation to begin prior to the reservation’s start time and use
  resources inside or outside of the reservation. A typical use case is to
  prevent jobs not explicitly requesting the reservation from using those
  reserved resources rather than forcing jobs requesting the reservation to
  use those resources in the time frame reserved.

-- Add NoDecay flag to QOS.

-- Node "OS" field expanded from "sysname" to "sysname release version" (e.g.
  change from "Linux" to
  "Linux 4.8.0-28-generic #28-Ubuntu SMP Sat Feb 8 09:15:00 UTC 2017").

-- jobcomp/elasticsearch - Add "job_name" and "wc_key" fields to stored
  information.

-- jobcomp/filetxt - Add ArrayJobId, ArrayTaskId, ReservationName, Gres,
  Account, QOS, WcKey, Cluster, SubmitTime, EligibleTime, DerivedExitCode and
  ExitCode.

-- scontrol modified to report core IDs for reservation containing individual
  cores.

-- MYSQL - Get rid of table join during rollup which speeds up the process
  dramatically on large job/step tables.

-- Add ability to define features on clusters for directing federated jobs to
  different clusters.

-- Add new RPC to process multiple federation RPCs in a single communication.

-- Modify slurm_load_jobs() function to load job information from all clusters
  in a federation.

-- Add squeue --local and --sibling options to modify filtering of jobs on
  federated clusters.

-- Add SchedulerParameters option of bf_max_job_user_part to specify the
  maximum number of jobs per user for any single partition. This differs from
  bf_max_job_user in that a separate counter is applied to each partition
  rather than having a single counter per user applied to all partitions.

-- Modify backfill logic so that bf_max_job_user, bf_max_job_part and
  bf_max_job_user_part options can all be used independently of each other.

-- Add sprio -p/--partition option to filter jobs by partition name.

-- Add partition name to job priority factor response message.

-- Add sprio --local and --sibling options for use in federation of clusters.

-- Add sprio "%c" format to print cluster name in federation mode.

-- Modify sinfo logic to provided unified view of all nodes and partitions

(continues on next page)
in a federation, add --local option to only report local state information
even in a cluster, print cluster name with "%V" format option, and
optionally sort by cluster name.
-- If a task in a parallel job fails and it was launched with the
--kill-on-bad-exit option then terminate the remaining tasks using the
SIGCONT, SIGTERM and SIGKILL signals rather than just sending SIGKILL.
-- Include submit_time when doing the sort for job scheduling.
-- Modify sacct to report all jobs in federation by default. Also add --local
option.
-- Modify sacct to accept "--cluster all" option (in addition to the old
"--cluster -1", which is still accepted).
-- Modify sreport to report all jobs in federation by default. Also add --local
option.
-- sched/backfill: Improve assoc_limit_stop configuration parameter support.
-- KNL features: Always keep active and available features in the same order:
first site-specific features, next MCDRAM modes, last NUMA modes.
-- Changed default ProctrackType to cgroup.
-- Add "cluster_name" field to node_info_t and partition_info_t data structure.
It is filled in only when the cluster is part of a federation and
SHOW_FEDERATION flag used.
-- Functions slurm_load_node() slurm_load_partitions() modified to show all
nodes/partitions in a federation when the SHOW_FEDERATION flag is used.
-- Add federated views to sview.
-- Add --federation option to sacct, scontrol, sinfo, sprio, squeue, sreport to
show a federated view. Will show local view by default.
-- Add FederationParameters=fed_display slurm.conf option to configure status
commands to display a federated view by default if the cluster is a member
of a federation.
-- Log the down nodes whenever slurmctld restarts.
-- Report that "CPUs" plus "Boards" in node configuration invalid only if the
CPUs value is not equal to the total thread count.
-- Extend the output of the seff utility to also include the job's wall-clock
time.
-- Add bf_max_time to SchedulerParameters.
-- Add bf_max_job_assoc to SchedulerParameters.
-- Add new SchedulerParameters option bf_window_linear to control the rate at
which the backfill test window expands. This can be used on a system with
a modest number of running jobs (hundreds of jobs) to help prevent expected
start times of pending jobs to get pushed forward in time. On systems with
large numbers of running jobs, performance of the backfill scheduler will
suffer and fewer jobs will be evaluated.
-- Improve scheduling logic with respect to license use and node reboots.
-- CRAY - Alter algorithm to come up with the SLURM_ID_HASH.
-- Implement federated scheduling and federated status outputs.
-- The '-q' option to srun has changed from being the short form of
'--quit-on-interrupt' to '--qos'.
-- Change sched_min_interval default from 0 to 2 microseconds.

* Changes in Slurm 17.02.11
===========================
-- Fix insecure handling of user_name and gid fields. CVE-2018-10995.
* Changes in Slurm 17.02.10

-- Fix updating of requested TRES memory.
-- Cray modulefile: avoid removing /usr/bin from path on module unload.
-- Fix issue when resetting the partition pointers on nodes.
-- Show reason field in 'sinfo -R' when nodes is marked as failed.
-- Fix potential of slurmdstepd segfaulting when the extern step fails to start.
-- Allow nodes state to be updated between FAIL and DRAIN.
-- Avoid registering a job'd credential multiple times.
-- Fix sbatch --wait to stop waiting after job is gone from memory.
-- Fix memory leak of MailDomain configuration string when slurmctld daemon is reconfigured.
-- Fix to properly remove extern steps from the starting_steps list.
-- Fix Slurm to work correctly with HDF5 1.10+.
-- Add support in salloc/srun --bb option for "access_mode" in addition to "access" for consistency with DW options.
-- Fix potential deadlock in _run_prog() in power save code.
-- MYSQL - Add dynamic_offset in the database to force range for auto increment ids for the tres_table.
-- Avoid setting node in COMPLETING state indefinitely if the job initiating the node reboot is cancelled while the reboot in in progress.
-- node_feature/knl_cray - Fix memory leaks that occur when slurmctld reconfigured.
-- node_feature/knl_cray - Fix memory leak that can occur during normal operation.
-- Fix job array dependency with "aftercorr" option and some task arrays in the first job fail. This fix lets all task array elements that can run proceed rather than stopping all subsequent task array elements.
-- Fix whole node allocation cpu counts when --hint=nomultihtread.
-- NRT - Fix issue when running on a HFI (p775) system with multiple protocols.
-- Fix uninitialized variables when unpacking slurmdb_archive_cond_t.
-- Fix security issue in accounting_storage/mysql plugin by always escaping strings within the slurmdbd. CVE-2018-7033.

* Changes in Slurm 17.02.9

-- When resuming powered down nodes, mark DOWN nodes right after ResumeTimeout has been reached (previous logic would wait about one minute longer).
-- Fix sreport not showing full column name for TRES Count.
-- Fix slurmdb_reservations_get() giving wrong usage data when job's spanned reservation that was modified.
-- Fix sreport reservation utilization report showing bad data.
-- Show all TRES' on a reservation in sreport reservation utilization report by default.
-- Fix sacctmgr show reservation handling "end" parameter.
-- Work around issue with sysmacros.h and gcc7 / glibc 2.25.
-- Fix layouts code to only allow setting a boolean.
-- Fix sbatch --wait to keep waiting even if a message timeout occurs.
-- CRAY - If configured with NodeFeatures=knl_cray and there are non-KNL nodes which include no features the slurmctld will abort without this patch when attempting strtok_r(NULL).
-- Fix regression in 17.02.7 which would run the spank_task_privileged as

(continues on next page)
part of the slurmstepd instead of it's child process.
-- Fix security issue in Prolog and Epilog by always prepending SPANK_ to
all user-set environment variables. CVE-2017-15566.

* Changes in Slurm 17.02.8
==========================
-- Add 'slurmdbd:' to the accounting plugin to notify message is from dbd
instead of local.
-- mpi/mvapich - Buffer being only partially cleared. No failures observed.
-- Fix for job --switch option on dragonfly network.
-- In salloc with --uid option, drop supplementary groups before changing UID.
-- jobcomp/elasticsearch - strip any trailing slashes from JobCompLoc.
-- jobcomp/elasticsearch - fix memory leak when transferring generated buffer.
-- Prevent slurmstepd ABRT when parsing gres.conf CPUs.
-- Fix sbatch --signal to signal all MPI ranks in a step instead of just those
on node 0.
-- Check multiple partition limits when scheduling a job that were previously
only checked on submit.
-- Cray: Avoid running application/step Node Health Check on the external
job step.
-- Optimization enhancements for partition based job preemption.
-- Address some build warnings from GCC 7.1, and one possible memory leak if
/proc is inaccessible.
-- If creating/altering a core based reservation with scontrol/sview on a
remote cluster correctly determine the select type.
-- Fix autoconf test for libcurl when clang is used.
-- Fix default location for cgroup_allowed_devices_file.conf to use correct
default path.
-- Document NewName option to sacctmgr.
-- Reject a second PMI2_Init call within a single step to prevent slurmstepd
from hanging.
-- Handle old 32bit values stored in the database for requested memory
correctly in sacct.
-- Fix memory leaks in the task/cgroup plugin when constraining devices.
-- Make extremely verbose info messages debug2 messages in the task/cgroup
plugin when constraining devices.
-- Fix issue that would deny the stepd access to /dev/null where GRES has a
'type' but no file defined.
-- Fix issue where the slurmstepd would fatal on job launch if you have no
gres listed in your slurm.conf but some in gres.conf.
-- Fix validating time spec to correctly validate various time formats.
-- Make scontrol work correctly with job update timelimit [+|-]=.
-- Reduce the visibility of a number of warnings in _part_access_check.
-- Prevent segfault in sacctmgr if no association name is specified for
an update command.
-- burst_buffer/cray plugin modified to work with changes in Cray UP05
software release.
-- Fix job reasons for jobs that are violating assoc MaxTRESPerNode limits.
-- Fix segfault when unpacking a 16.05 slurm_cred in a 17.02 daemon.
-- Fix setting TRES limits with case insensitive TRES names.
-- Add alias for xstrncmp() -- slurm_xstrncmp().
-- Fix sorting of case insensitive strings when using xstrcasestrcmp().
-- Gracefully handle race condition when reading /proc as process exits.
-- Avoid error on Cray duplicate setup of core specialization.
-- Skip over undefined (hidden in Slurm) nodes in pbsnodes.
-- Add empty hashes in perl api's slurm_load_node() for hidden nodes.
-- CRAY - Add rpath logic to work for the alpscomm libs.
-- Fixes for administrator extended TimeLimit (job reason & time limit reset).
-- Fix gres selection on systems running select/linear.
-- sview: Added window decorator for maximize,minimize,close buttons for all systems.
-- squeue: interpret negative length format specifiers as a request to delimit values with spaces.
-- Fix the torque pbsnodes wrapper script to parse a gres field with a type set correctly.

* Changes in Slurm 17.02.7

---

-- Fix deadlock if requesting to create more than 10000 reservations.
-- Fix potential memory leak when creating partition name.
-- Execute the HealthCheckProgram once when the slurmd daemon starts rather than executing repeatedly until an exit code of 0 is returned.
-- Set job/step start and end times to 0 when using --truncate and start > end.
-- Make srun --pty option ignore EINTR allowing windows to resize.
-- When resuming node only send one message to the slurmd.
-- Modify srun --pty option to use configured SrunPortRange range.
-- Fix issue with whole gres not being printed out with Slurm tools.
-- Fix issue with multiple jobs from an array are prevented from starting.
-- Fix for possible slurmd abort with use of salloc/sbatch/srun --gres-flags=enforce-binding option.
-- Fix race condition when using jobacct_gather/cgroup where the memory of the step wasn't always gathered correctly.
-- Better debug when slurmd queue is filling up in the slurmd.
-- Fixed truncation on scontrol show config output.
-- Serialize updates from from the dbd to the slurmd.
-- Fix memory leak in slurmd when agent queue to the DBD has filled up.
-- CRAY - Throttle step creation if trying to create too many steps at once.
-- If failing after switch_g_job_init happened make sure switch_g_job_fini is called.
-- Fix minor memory leak if launch fails in the slurmstepd.
-- Fix issue where UnkillableStepProgram if step was in an ending state.
-- Fix bug when tracking multiple simultaneous spawned ping cycles.
-- jobcomp/elasticsearch plugin now saves state of pending requests on slurmd daemon shutdown so then can be recovered on restart.
-- Fix issue when an alternate munge key when communicating on a persistent connection.
-- Document inconsistent behavior of GroupUpdateForce option.
-- Fix bug in selection of GRES bound to specific CPUs where the GRES count is 2 or more. Previous logic could allocate CPUs not available to the job.
-- Increase buffer to handle long /proc/<pid>/stat output so that Slurm can read correct RSS value and take action on jobs using more memory than requested.
-- Fix srun job jobs that can run immediately to run in the highest priority partition when multiple partitions are listed. scontrol show jobs can
potentially show the partition list in priority order.
-- Fix starting controller if StateSaveLocation path didn't exist.
-- Fix inherited association 'max' TRES limits combining multiple limits in
  the tree.
-- Sort TRES id's on limits when getting them from the database.
-- Fix issue with pmi[2|x] when TreeWidth=1.
-- Correct buffer size used in determining specialized cores to avoid possible
  truncation of core specification and not reserving the specified cores.
-- Close race condition on Slurm structures when setting DebugFlags.
-- Make it so the cray/switch plugin grabs new DebugFlags on a reconfigure.
-- Fix incorrect lock levels when creating or updating a reservation.
-- Fix overlapping reservation resize.
-- Add logic to help support Dell KNL systems where syscfg is different than
  the normal Intel syscfg.
-- CRAY - Fix BB to handle type= correctly, regression in 17.02.6.

* Changes in Slurm 17.02.6
==========================
-- Fix configurator.easy.html to output the SelectTypeParameters line.
-- If a job requests a specific memory requirement then gets something else
  from the slurmd make sure the step allocation is made aware of it.
-- Fix missing initialization in slurmd.
-- Fix potential degradation when running HTC (> 100 jobs a sec) like
  workflows through the slurmd.
-- Fix race condition which could leave a stepd hung on shutdown.
-- CRAY - Add configuration for ATP to the ansible play script.
-- Fix potential to corrupt DBD message.
-- burst_buffer logic modified to support sizes in both SI and EIC size units
  (e.g. M/MiB for powers of 1024, MB for powers of 1000).

* Changes in Slurm 17.02.5
==========================
-- Prevent segfault if a job was blocked from running by a QOS that is then
  deleted.
-- Improve selection of jobs to preempt when there are multiple partitions
  with jobs subject to preemption.
-- Only set kmem limit when ConstrainKmemSpace=yes is set in cgroup.conf.
-- Fix bug in task/affinity that could result in slurmd fatal error.
-- Increase number of jobs that are tracked in the slurmd as finishing at one
  time.
-- Note when a job finishes in the slurmd to avoid a race when launching a
  batch job takes longer than it takes to finish.
-- Improve slurmd startup on large systems (> 10000 nodes)
-- Add LaunchParameters option of cray_net_exclusive to control whether all
  jobs on the cluster have exclusive access to their assigned nodes.
-- Make sure srun inside an allocation gets --ntasks-per-[core|socket]
  set correctly.
-- Only make the extern step at job creation.
-- Fix for job step task layout with --cpus-per-task option.
-- Fix --ntasks-per-core option/environment variable parsing to set
  the requested value, instead of always setting one (srun).
-- Correct error message when ClusterName in configuration files does not match

(continues on next page)
the name in the slurmctld daemon's state save file.
-- Better checking when a job is finishing to avoid underflow on job's
submitted to a QOS/association.
-- Handle partition QOS submit limits correctly when a job is submitted to
more than 1 partition or when the partition is changed with scontrol.
-- Performance boost for when Slurm is dealing with credentials.
-- Fix race condition which could leave a stepd hung on shutdown.
-- Add lua support for opensuse.

* Changes in Slurm 17.02.4
================================
-- Do not attempt to schedule jobs after changing the power cap if there are
already many active threads.
-- Job expansion example in FAQ enhanced to demonstrate operation in
heterogeneous environments.
-- Prevent scontrol crash when operating on array and no-array jobs at once.
-- knl_cray plugin: Log incomplete capmc output for a node.
-- knl_cray plugin: Change capmc parsing of mcdram_pct from string to number.
-- Remove log files from test20.12.
-- When rebooting a node and using the PrologFlags=alloc make sure the
prolog is ran after the reboot.
-- node_features/knl_generic - If a node is rebooted for a pending job, but
fails to enter the desired NUMA and/or MCDRAM mode then drain the node and
requeue the job.
-- node_features/knl_generic disable mode change unless RebootProgram
configured.
-- Add new burst_buffer function bb_g_job_revoke_alloc() to be executed
if there was a failure after the initial resource allocation. Does not
release previously allocated resources.
-- Test if the node_bitmap on a job is NULL when testing if the job's nodes
are ready. This will be NULL is a job was revoked while beginning.
-- Fix incorrect lock levels when testing when job will run or updating a job.
-- Add missing locks to job_submit/pbs plugin when updating a jobs
dependencies.
-- Add support for lua5.3
-- Add min_memory_per_node|cpu to the job_submit/lua plugin to deal with lua
not being able to deal with pn_min_memory being a uint64_t. Scripts are
urged to change to these new variables avoid issue. If not set the
variables will be 'nil'.
-- Calculate priority correctly when 'nice' is given.
-- Fix minor typos in the documentation.
-- node_features/knl_cray: Preserve non-KNL active features if slurmctld
reconfigured while node boot in progress.
-- node_features/knl_generic: Do not repeatedly log errors when trying to read
KNL modes if not KNL system.
-- Add missing QOS read lock to backfill scheduler.
-- When doing a dlopen on liblua only attempt the version compiled against.
-- Fix null-dereference in sreport cluster utilization when configured with
memory-leak-debug.
-- Fix Partition info in 'scontrol show node'. Previously duplicate partition
names, or Partitions the node did not belong to could be displayed.
-- Fix it so the backup slurmdbd will take control correctly.

(continues on next page)
-- Fix unsafe use of MAX() macro, which could result in problems cleaning up accounting plugins in slurmd, or repeat job cancellation attempts in scancel.
-- Fix 'scontrol update reservation duration=unlimited' to set the duration to 365-days (as is done elsewhere), rather than 49710 days.
-- Check if variable given to scontrol show job is a valid jobid.
-- Fix WithSubAccounts option to not include WithDeleted unless requested.
-- Prevent a job tested on multiple partitions from being marked WHOLE_NODE_USER.
-- Prevent a race between completing jobs on a user-exclusive node from leaving the node owned.
-- When scheduling take the nodes in completing jobs out of the mix to reduce fragmentation. SchedulerParameters=reduce_completing_frag
-- For jobs submitted to multiple partitions, report the job's earliest start time for any partition.
-- Backfill partitions that use QOS Grp limits to "float" better.
-- node_features/knl_cray: don't clear configured GRES from non-KNL node.
-- sacctmgr - prevent segfault in command when a request is denied due to an insufficient privileges.
-- Add warning about libcurl-devel not being installed during configure.
-- Streamline job purge by handling file deletion on a separate thread.
-- Always set RLIMIT_CORE to the maximum permitted for slurmd, to ensure core files are created even on non-developer builds.
-- Fix --ntasks-per-core option/environment variable parsing to set the requested value, instead of always setting one.
-- If trying to cancel a step that hasn't started yet for some reason return a good return code.
-- Fix issue with sacctmgr show where user="

* Changes in Slurm 17.02.3
====================================
-- Increase --cpu_bind and --mem_bind field length limits.
-- Fix segfault when using AdminComment field with job arrays.
-- Clear Dependency field when all dependencies are satisfied.
-- Add --array-unique to squeue which will display one unique pending job array element per line.
-- Reset backfill timers correctly without skipping over them in certain circumstances.
-- When running the "scontrol top" command, make sure that all of the user's jobs have a priority that is lower than the selected job. Previous logic would permit other jobs with equal priority (no jobs with higher priority).
-- Fix perl api so we always get an allocation when calling Slurm::new().
-- Fix issue with cleaning up cpuset and devices cgroups when multiple steps end at the same time.
-- Document that PriorityFlags option of DEPTH_OBLIVIOUS precludes the use of FAIR_TREE.
-- Fix issue if an invalid message came in a Slurm daemon/command may abort.
-- Make it impossible to use CR_CPU* along with CR_ONE_TASK_PER_CORE. The options are mutually exclusive.
-- ALPS - Fix scheduling when ALPS doesn't agree with Slurm on what nodes are free.
-- When removing a partition make sure it isn't part of a reservation.
-- Fix seg fault if loading attempting to load non-existent burstbuffer plugin.
-- Fix to backfill scheduling with respect to QOS and association limits. Jobs
submitted to multiple partitions are most likley to be effected.
-- sched/backfill: Improve assoc_limit_stop configuration parameter support.
-- CRAY - Add ansible play and README.
-- sched/backfill: Fix bug related to advanced reservations and the need to
reboot nodes to change KNL mode.
-- Preempt plugins - fix check for 'preempt_youngest_first' option.
-- Preempt plugins - fix incorrect casts in preempt_youngest_first mode.
-- Preempt/job_prio - fix incorrect casts in sort function.
-- Fix to make task/affinity work with ldoms where there are more than 64
cpus on the node.
-- When using node_features/knl_generic make it so the slurmd doesn't segfault
when shutting down.
-- Fix potential double-xfree() when using job arrays that can lead to
slurmctld crashing.
-- Fix priority/multifactor priorities on a slurmctld restart if not using
accounting_storage/[mysql|slurmdbd].
-- Fix NULL dereference reported by CLANG.
-- Update proctrack documentation to strongly encourage use of
proctrack/cgroup.
-- Fix potential memory leak if job fails to begin after nodes have been
selected for a job.
-- Handle a job that made it out of the select plugin without a job_resrcs
pointer.
-- Fix potential race condition when persistent connections are being closed at
shutdown.
-- Fix incorrect locks levels when submitting a batch job or updating a job
in general.
-- CRAY - Move delay waiting for job cleanup to after we check once.
-- MYSQL - Fix memory leak when loading archived jobs into the database.
-- Fix potential race condition when starting the priority/multifactor plugin's
decay thread.
-- Sanity check to make sure we have started a job in acct_policy.c before we
clear it as started.
-- Allow reboot program to use arguments.
-- Message Aggr - Remove race condition on slurmd shutdown with respects to
destroying a mutex.
-- Fix updating job priority on multiple partitions to be correct.
-- Don't remove admin comment when updating a job.
-- Return error when bad separator is given for scontrol update job licenses.

* Changes in Slurm 17.02.2
----------------------------------------
-- Update hyperlink to LBNL Node Health Check program.
-- burst_buffer/cray - Add support for line continuation.
-- If a job is cancelled by the user while it's allocated nodes are being
reconfigured (i.e. the capmc_resume program is rebooting nodes for the job)
and the node reconfiguration fails (i.e. the reboot fails), then don't
requeue the job but leave it in a cancelled state.
-- capmc_resume (Cray resume node script) - Do not disable changing a node's
active features if SyscfgPath is configured in the knl.conf file.
-- Improve the srun documentation for the --resv-ports option.
-- burst_buffer/cray - Fix parsing for discontinuous allocated nodes. A job allocation of "20,22" must be expressed as "20\n22".
-- Fix rare segfault when shutting down slurmctld and still sending data to the database.
-- Fix gres output of a job if it is updated while pending to be displayed correctly with Slurm tools.
-- Fix pam_slurm_adopt.
-- Fix missing unlock when job_list doesn't exist when starting priority/multifactor.
-- Fix segfault if slurmctld is shutting down and the slurmdbd plugin was in the middle of setting db_indexes.
-- Add ESLURM_JOB_SETTING_DB_INX to errno to note when a job can't be updated because the dbd is setting a db_index.
-- Fix possible double insertion into database when a job is updated at the moment the dbd is assigning a db_index.
-- Fix memory error when updating a job's licenses.
-- Fix seff to work correctly with non-standard perl installs.
-- Export missing slurmdbd_defs_[init|fini] needed for libslurmdb.so to work.
-- Fix sacct from returning way more than requested when querying against a job array task id.
-- Fix double read lock of tress when updating gres or licenses on a job.
-- Make sure locks are always in place when calling assoc_mgr_make_tres_str_from_array.
-- Prevent slurmctld SEGV when creating reservation with duplicated name.
-- Consider QOS flags Partition[Min|Max]Nodes when doing backfill.
-- Fix slurmdbd_defs.c to not have half symbols go to libslurm.so and the other half go to libslurmdb.so.
-- Fix 'scontrol show jobs' to remove an errant newline when 'Switches' is printed.
-- Better code for handling memory required by a task on a heterogeneous system.
-- Fix regression in 17.02.0 with respects to GrpTresMins on a QOS or Association.
-- Cleanup to make make dist work.
-- Schedule interactive jobs quicker.
-- Perl API - correct value of MEM_PER_CPU constant to correctly handle memory values.
-- Fix 'flags' variable to be 32 bit from the old 16 bit value in the perl api.
-- Export sched_nodes for a job in the perl api.
-- Improve error output when updating a reservation that has already started.
-- Fix --ntasks-per-node issue with srun so DenyOnLimit would work correctly.
-- node_features/knl_cray plugin - Fix memory leak.
-- Fix wrong cpu_per_task count issue on heterogeneous system when dealing with steps.
-- Fix double free issue when removing usage from an association with sacctmgr.
-- Fix issue with SPANK plugins attempting to set null values as environment variables, which leads to the command segfaulting on newer glibc versions.
-- Fix race condition on slurmctld startup when plugins have not gone through init() ahead of the rpc_manager processing incoming messages.
-- job_submit/lua - expose admin_comment field.
-- Allow AdminComment field to be set by the job_submit plugin.
-- Allow AdminComment field to be changed by any Administrator.
-- Fix key words in jobcomp select.
-- MYSQL - Streamline job flush sql when doing a clean start on the slurmctld.
-- Fix potential infinite loop when talking to the DBD when shutting down the slurmctld.
-- Fix MCS filter.
-- Make it so pmix can be included in the plugin rpm without having to specify --with-pmix.
-- MYSQL - Fix initial load when not using he DBD.
-- Fix scontrol top to not make jobs priority 0 (held).
-- Downgrade info message about exceeding partition time limit to a debug2.

* Changes in Slurm 17.02.1-2
=================================
-- Replace clock_gettime with time(NULL) for very old systems without the call.

* Changes in Slurm 17.02.1
=================================
-- Modify pam module to work when configured NodeName and NodeHostname differ.
-- Update to sbatch/srun man pages to explain the "filename pattern" clearer
-- Add %x to sbatch/srun filename pattern to represent the job name.
-- job_submit/lua - Add job "bitflags" field.
-- Update slurm.spec file to note obsolete RPMs.
-- Fix deadlock scenario when dumping configuration in the slurmctld.
-- Remove unneeded job lock when running assoc_mgr cache. This lock could cause potential deadlock when/if TRES changed in the database and the slurmctld wasn't made aware of the change. This would be very rare.
-- Fix missing locks in gres logic to avoid potential memory race.
-- If gres is NULL on a job don't try to process it when returning detailed information about a job to scontrol.
-- Fix print of consumed energy in sstat when no energy is being collected.
-- Print formatted tres string when creating/updating a reservation.
-- Fix issues with QOS flags Partition[Min|Max]Nodes to work correctly.
-- Prevent manipulation of the cpu frequency and governor for batch or external steps. This addresses an issue where the batch step would inadvertently set the cpu frequency maximum to the minimum value supported on the node.
-- Convert a slurmctld power management data structure from array to list in order to eliminate the possibility of zombie child suspend/resume processes.
-- Burst_buffer/cray - Prevent slurmctld daemon abort if "paths" operation fails. Now job will be held. Update job update time when held.
-- Fix issues with QOS flags Partition[Min|Max]Nodes to work correctly.
-- Refactor slurmctld agent logic to eliminate some pthreads.
-- Added "SyscfgTimeout" parameter to knl.conf configuration file.
-- Fix for CPU binding for job steps run under a batch job.

* Changes in Slurm 17.02.0
=================================
-- job_submit/lua - Make "immediate" parameter available.
-- Fix srun I/O race condition to eliminate a error message that might be generated if the application exits with outstanding stdin.
-- Fix regression when purging/archiving jobs/events.
-- Add new job state JOB_OOM indicating Out Of Memory condition as detected
by task/cgroup plugin.
-- If QOS has been added to the system go refigure out Deny/AllowQOS on
partitions.
-- Deny job with duplicate GRES requested.
-- Fix loading super old assoc_mgr usage without segfaulting.
-- CRAY systems: Restore TaskPlugins order of task/cray before task/cgroup.
-- Task/cray: Treat missing "mems" cgroup with "debug" messages rather than
"error" messages. The file may be missing at step termination due to a
change in how cgroups are released at job/step end.
-- Fix for job constraint specification with counts, --ntasks-per-node value,
and no node count.
-- Fix ordering of step task allocation to fill in a socket before going into
another one.
-- Fix configure to not require C++
-- job_submit/lua - Remove access to slurmctld internal reservation fields of
job_pend_cnt and job_run_cnt.
-- Prevent job_time_limit enforcement from blocking other internal operations
if a large number of jobs need to be cancelled.
-- Add 'preempt_youngest_order' option to preempt/partition_prio plugin.
-- Fix controller being able to talk to a pre-released DBD.
-- Added ability to override the invoking uid for "scontrol update job"
by specifying "--uid=<uid>|-u <uid>".
-- Changed file broadcast "offset" from 32 to 64 bits in order to support files
over 2 GB.
-- slurm.spec - do not install init scripts alongside systemd service files.

* Changes in Slurm 17.02.0rc1
==============================================
-- Add port info to 'sinfo' and 'scontrol show node'.
-- Fix errant definition of USE_64BIT_BITSTR which can lead to core dumps.
-- Move BatchScript to end of each job's information when using
"scontrol -dd show job" to make it more readable.
-- Add SchedulerParameters configuration parameter of "default_gbytes", which
treats numeric only (no suffix) value for memory and tmp disk space as being
in units of Gigabytes. Mostly for compatibility with LSF.
-- Fix race condition in srun/sattach logic which would prevent srun from
terminating.
-- Bitstring operations are now 64bit instead of 32bit.
-- Replace hwweight() function in bitstring with faster version.
-- scancel would treat a non-numeric argument as the name of jobs to be
cancelled (a non-documented feature). Cancelling jobs by name now require
the "--jobname=" command line argument.
-- scancel modified to note that no jobs satisfy the filter options when the
--verbose option is used along with one or more job filters (e.g. "--qos=").
-- Change _pack_cred to use pack_bit_str_hex instead of pack_bit_fmt for
better scalability and performance.
-- Add BootTime configuration parameter to knl.conf file to optimize resource
allocations with respect to required node reboots.
-- Add node_features_p_boot_time() to node_features plugin to optimize
scheduling with respect to node reboots.

(continues on next page)
-- Avoid allocating resources to a job in the event that its run time plus boot
time (if needed) extend into an advanced reservation.
-- Burst_buffer/cray - Avoid stage-out operation if job never started.
-- node_features/knl_cray - Add capability to detected Uncorrectable Memory
   Errors (UME) and if detected then log the event in all job and step stderr
   with a message of the form:
   error: *** STEP 1.2 ON tux1 UNCORRECTABLE MEMORY ERROR AT 2016-12-14T09:09:37 ***
   Similar logic added to node_features/knl_generic in version 17.02.0pre4.
-- If job is allocated nodes which are powered down, then reset job start time
   when the nodes are ready and do not charge the job for power up time.
-- Add the ability to purge transactions from the database.
-- Add support for requeue'ing of federated jobs (BETA).
-- Add support for interactive federated jobs (BETA).
-- Add the ability to purge rolled up usage from the database.
-- Properly set SLURM_JOB_GPUS environment variable for Prolog.

* Changes in Slurm 17.02.0pre4

----------------------------------------
-- Add support for per-partition OverTimeLimit configuration.
-- Add --mem_bind option of "sort" to run zonesort on KNL nodes at step start.
-- Add LaunchParameters=mem_sort option to configure running of zonesort
   by default at step startup.
-- Add "FreeSpace" information for each pool to the "scontrol show burstbuffer"
   output. Required changes to the burst_buffer_info_t data structure.
-- Add new node state flag of NODE_STATE_REBOOT for node reboots triggered by
   "scontrol reboot" commands. Previous logic re-used NODE_STATE_MAINT flag,
   which could lead to inconsistencies. Add "ASAP" option to "scontrol reboot"
   command that will drain a node in order to reboot it as soon as possible,
   then return it to service.
-- Allow unit conversion routine to convert 1024M to 1G.
-- switch/cray plugin - change legacy spool directory location.
-- Add new PriorityFlags option of INCR_ONLY, which prevents a job's priority
   from being decremented.
-- Make it so we don't purge job start messages until after we purge step
   messages. Hopefully this will reduce the number of messages lost when
   filling up memory when the database/DBD is down.
-- Added SchedulingParameters option of "bf_job_part_count_reserve". Jobs below
   the specified threshold will not have resources reserved for them.
-- If GRES are configured with file IDs, then "scontrol -d show node" will
   not only identify the count of currently allocated GRES, but their specific
   index numbers (e.g. "GresUsed=gpu:alpha:2(IDX:0,2),gpu:beta:0(IDX:N/A)").
   Ditto for job information with "scontrol -d show job".
-- Add new mcs/account plugin.
-- Add "GresEnforceBind=Yes" to "scontrol show job" output if so configured.
-- Add support for SALLOC_CONSTRAINT, SBATCH_CONSTRAINT and SLURM_CONSTRAINT
   environment variables to set default constraints for salloc, sbatch and
   srun commands respectively.
-- Provide limited support for the MemSpecLimit configuration parameter without
   the task/cgroup plugin.
-- node_features/knl_generic - Add capability to detected Uncorrectable Memory
   Errors (UME) and if detected then log the event in all job and step stderr
   with a message of the form:
error: *** STEP 1.2 ON tux1 UNCORRECTABLE MEMORY ERROR AT 2016-12-14T09:09:37 ***

-- Add SLURM_JOB_GID to TaskProlog environment.
-- burst_buffer/cray - Remove leading zeros from node ID lists passed to
dw_wlm_cli program.
-- Add "Partitions" field to "scontrol show node" output.
-- Remove sched/wiki and sched/wiki2 plugins and associated code.
-- Remove SchedulerRootFilter option and slurm_get_root_filter() API call.
-- Add SchedulerParameters option of spec_cores_first to select specialized
cores from the lowest rather than highest number cores and sockets.
-- Add PrologFlags option of Serial to disable concurrent launch of
Prolog and Epilog scripts.
-- Fix security issue caused by insecure file path handling triggered by the
failure of a Prolog script. To exploit this a user needs to anticipate or
cause the Prolog to fail for their job. CVE-2016-10030.

* Changes in Slurm 17.02.0pre3

---

-- Add srun host & PID to job step data structures.
-- Avoid creating duplicate pending step records for the same srun command.
-- Rewrite srun's logic for pending steps for better efficiency (fewer RPCs).
-- Added new SchedulerParameters options step_retry_count and step_retry_time
to control scheduling behaviour of job steps waiting for resources.
-- Optimize resource allocation logic for --spread-job job option.
-- Modify cpu_bind and mem_bind map and mask options to accept a repetition
count to better support large task count. For example:
"mask_mem:0x0f*2,0xf0*2" is equivalent to "mask_mem:0x0f,0x0f,0xf0,0xf0".
-- Add support for --mem_bind=prefer option to prefer, but not restrict memory
use to the identified NUMA node.
-- Add mechanism to constrain kernel memory allocation using cgroups. New
cgroup.conf parameters added: ConstrainKmemSpace, MaxKmemPercent, and
MinKmemSpace.
-- Correct invokation of man2html, which previously could cause FreeBSD builds
to hang.
-- MYSQL - Unconditionally remove 'ignore' clause from 'alter ignore'.
-- Modify service files to not start Slurm daemons until after Munge has been
started.
NOTE: If you are not using Munge, but are using the "service" scripts to
start Slurm daemons, then you will need to remove this check from the
/etc/slurm*service scripts.
-- Do not process SALLOC_HINT, SBATCH_HINT or SLURM_HINT environment variables
if any of the following salloc, sbatch or srun command line options are
specified: -B, --cpu_bind, --hint, --ntasks-per-core, or --threads-per-core.
-- burst_buffer/cray: Accept new jobs on backup slurmctld daemon without access
to dw_wlm_cli command. No burst buffer actions will take place.
-- Do not include SLURM_JOB_DERIVED_EC, SLURM_JOB_EXIT_CODE, or
SLURM_JOB_EXIT_CODE in PrologSlurmctld environment (not available yet).
-- Cray - set task plugin to fatol() if task/cgroup is not loaded after
task/cray in the TaskPlugin settings.
-- Remove separate slurm_blcr package. If Slurm is built with BLCR support,
the files will now be part of the main Slurm packages.
-- Replace sjstat, seff and sjobexit RPM packages with a single "contribs"
package.
-- Remove long since defunct slurmdb-direct scripts.
-- Add SbcastParameters configuration option to control default file
  destination directory and compression algorithm.
-- Add new SchedulerParameter (max_array_tasks) to limit the maximum number of
  tasks in a job array independently from the maximum task ID (MaxArraySize).
-- Fix issue where number of nodes is not properly allocated when sbatch and
  salloc are requested with -n tasks < hosts from -w hostlist or from -N.
-- Add infrastructure for submitting federated jobs.

* Changes in Slurm 17.02.0pre2

-- Add new RPC (REQUEST_EVENT_LOG) so that slurmdb and slurmstepd can log events
  through the slurmctld daemon.
-- Remove sbatch --bb option. That option was never supported.
-- Automatically clean up task/cgroup cpuset and devices cgroups after steps
  are completed.
-- Add federation read/write locks.
-- Limit job purge run time to 1 second at a time.
-- The database index for jobs is now 64 bits. If you happen to be close to
  4 billion jobs in your database you will want to update your slurmctld at
  the same time as your slurmdbd to prevent roll over of this variable as
  it is 32 bit previous versions of Slurm.
-- Optionally lock slurmstepd in memory for performance reasons and to avoid
  possible SIGBUS if the daemon is paged out at the time of a Slurm upgrade
  (changing plugins). Controlled via new LaunchParameters options of
  slurmstepd_memlock and slurmstepd_memlock_all.
-- Add event trigger on burst buffer errors (see strigger man page,
  --burst_buffer option).
-- Add job AdminComment field which can only be set by a Slurm administrator.
-- Add salloc, sbatch and srun option of --delay-boot=<time>, which will
  temporarily delay booting nodes into the desired state for a job in the
  hope of using nodes already in the proper state which will be available at
  a later time.
-- Add job burst_buffer_state and delay_boot fields to scontrol and squeue
  output. Also add ability to modify delay_boot from scontrol.
-- Fix for node's available TRES array getting filled in with configured GRES
  model types.
-- Log if job --bb option contains any unrecognized content.
-- Display configured and allocated TRES for nodes in scontrol show nodes.
-- Change all memory values (in MB) to uint64_t to accommodate > 2TB per node.
-- Add MailDomain configuration parameter to qualify email addresses.
-- Refactor the persistent connections within the federation code to use
  the same logic that was found in the slurmdb. Now both functionalities
  share the same code.
-- Remove BlueGene/L and BlueGene/P support.
-- Add "flag" field to launch_tasks_request_msg. Remove the following fields
  (moved into flags): multi_prog, task_flags, user_managed_io, pty,
  buffered_stdio, and labelio.
-- Add protocol version to slurmd startup communications for slurmstepd to
  permit changes in the protocol.

* Changes in Slurm 17.02.0pre1

(continues on next page)
-- burst_buffer/cray - Add support for round-up the size of a buffer request if the DataWarp configuration "equalize_fragments" is used.
-- Remove AIX support.
-- Rename "in" to "input" in slurm_step_io_fds data structure defined in slurm.h. This is needed to avoid breaking Python with by using one of its keywords in a Slurm data structure.
-- Remove eligible_time from jobcomp/elasticsearch.
-- Enable the deletion of a QOS, even if no clusters have been added to the database.
-- SlurmDBD - change all timestamps to bigint from int to solve Y2038 problem.
-- Add salloc/sbatch/srun --spread-job option to distribute tasks over as many nodes as possible. This also treats the --ntasks-per-node option as a maximum value.
-- Add ConstrainKmemSpace to cgroup.conf, defaulting to yes, to allow group Kmem enforcement to be enabled while still using ConstrainKmemSpace.
-- Add sbatch --bbf option to specify a burst buffer input file.
-- Added support for burst buffer support for job array. Add new SchedulerParameters configuration parameter of bb_array_stage_cnt=# to indicate how many pending tasks of a job array should be made available for burst buffer resource allocation.
-- Add start_delay field to jobcomp/elasticsearch.
-- Fixed small memory leak when a job fails to load from state save.
-- Fix invalid read when attempting to delete clusters from database.
-- Fix small memory leak when deleting clusters from database.
-- Added infrastructure for setting up federations in database and establishing connections between federation clusters.
-- Added start_delay field to jobcomp/elasticsearch.
-- In order to support federated jobs, the MaxJobID configuration parameter default value has been reduced from 2,147,418,112 to 67,043,328 and its maximum value is now 67,108,863. Upon upgrading, any pre-existing jobs that have a job ID above the new range will continue to run and new jobs will get job IDs in the new range.
-- Added infrastructure for setting up federations in database and establishing connections between federation clusters.
-- Restructure job accounting query to use 'id_job in (1, 2, ..)' format instead of logically equivalent 'id_job = 1 || id_job = 2 || ..'.
-- Added start_delay field to jobcomp/elasticsearch.
-- Change all timestamps to bigint from int to solve Y2038 problem.
--- Fix memory error when updating a job's licenses.
--- Fix double read lock of tres when updating gres or licenses on a job.
--- Fix regression in 16.05.10 with respects to GrpTresMins on a QOS or
  Association.
--- ALPS - Fix scheduling when ALPS doesn't agree with Slurm on what nodes
  are free.
--- Fix seg fault if loading attempting to load non-existent burstbuffer plugin.
--- Fix to backfill scheduling with respect to QOS and association limits. Jobs
  submitted to multiple partitions are most likely to be effected.
--- Avoid erroneous errno set by the mariadb 10.2 api.
--- Fix security issue in Prolog and Epilog by always prepending SPANK_ to
  all user-set environment variables. CVE-2017-15566.

* Changes in Slurm 16.05.10-2
=================================
--- Replace clock_gettime with time(NULL) for very old systems without the call.

* Changes in Slurm 16.05.10
============================
--- Record job state as PREEMPTED instead of TIMEOUT when GraceTime is reached.
--- task/cgroup - print warnings to stderr when --cpu_bind=verbose is enabled
  and the requested processor affinity cannot be set.
--- power/cray - Disable power cap get and set operations on DOWN nodes.
--- Jobs preempted with PreemptMode=REQUEUE were incorrectly recorded as
  REQUEUED in the accounting.
--- PMIX - Use volatile specifier to avoid flag caching and lock the flag to
  make sure it is protected.
--- PMIX/PMI2 - Make it possible to use %n or %h in a spool dir.
--- burst_buffer/cray - Support default pool which is not the first pool
  reported by DataWarp and log in Slurm when pools that are added or removed
  from DataWarp.
--- Insure job does not start running before PrologSlurmctld is complete and
  node is booted (all nodes for interactive job, at least first node for batch
  job without burst buffers).
--- Fix minor memory leak in the slurmctld when removing a QOS.
--- burst_buffer/cray - Do not execute "pre_run" operation until after all nodes
  are booted and ready for use.
--- scontrol - return an error when attempting to use the +=/-- syntax to
  update a field where this is not appropriate.
--- Fix task/affinity to work correctly with --ntasks-per-socket.
--- Honor --ntasks-per-node and --ntasks option when used with job constraints
  that contain node counts.
--- Prevent deadlocked slurmstepd processes due to unsafe use of regcomp with
  older glibc versions.
--- Fix squeue when SLURM_BITSTR_LEN=0 is set in the user environment.
--- Fix comments in acct_policy.c to reflect actual variables instead of
  old ones.
--- Fix correct variables when validating GrpTresMins on a QOS.
--- Better debug output when a job is being held because of a GrpTRES[Run]Min
  limits.
--- Fix correct state reason when job can't run 'safely' because of an
  association GrpWall limit.

(continues on next page)
-- Squeue always loads new data if user_id option specified
-- Fix for possible job ID parsing failure and abort.
-- If node boot in progress when slurmctld daemon is restarted, then allow
  sufficient time for reboot to complete and not prematurely DOWN the node as
  "Not responding".
-- For job resize, correct logic to build "resize" script with new values.
  Previously the scripts were based upon the original job size.
-- Fix squeue to not limit the size of partition, burst_buffer, exec_host, or
  reason to 32 chars.
-- Fix potential packing error when packing a NULL slurmdb_clus_res_rec_t.
-- Fix potential packing errors when packing a NULL slurmdb_reservation_cond_t.
-- Burst_buffer/cray - Prevent slurmctld daemon abort if "paths" operation
  fails. Now job will be held. Update job update time when held.
-- Fix issues with QOS flags Partition[Min|Max]Nodes to work correctly.
-- Increase number of ResumePrograms that can be managed without leaving
  zombie/orphan processes from 10 to 100.
-- Refactor slurmctld agent logic to eliminate some pthreads.

* Changes in Slurm 16.05.9
==========================
-- Fix parsing of SBCAST_COMPRESS environment variable in sbcast.
-- Change some debug messages to errors in task/cgroup plugin.
-- backfill scheduler: Stop trying to determine expected start time for a job
  after 2 seconds of wall time. This can happen if there are many running jobs
  and a pending job can not be started soon.
-- Improve performance of cr_sort_part_rows() in cons_res plugin.
-- CRAY - Fix deadlock issue when updating accounting in the slurmctld and
  scheduling a Datawarp job.
-- Correct the job state accounting information for jobs requeued due to burst
  buffer errors.
-- burst_buffer/cray - Avoid "pre_run" operation if not using buffer (i.e.
  just creating or deleting a persistent burst buffer).
-- Fix slurm.spec file support for BlueGene builds.
-- Fix missing TRES read lock in acct_policy_jobRunnable_pre_select() code.
-- Fix debug2 message printing value using wrong array index in
  _qos_jobRunnable_post_select() code.
-- Prevent job timeout on node power up.
-- MYSQL - Fix minor memory leak when querying steps and the sql fails.
  Make it so sacctmgr accepts column headers like MaxTRESPU and not MaxTRESP.
-- Only look at SLURM_STEP_KILLED_MSG_NODE_ID on startup, to avoid race
  condition later when looking at a steps env.
-- Make backfill scheduler behave like regular scheduler in respect to
  'assoc_limit_stop'.
-- Allow a lower version client command to talk to a higher version controller
  using the multi-cluster options (e.g. squeue -M<cluster>).
-- slurmctld/agent race condition fix: Prevent job launch while PrologSlurmctld
  daemon is running or node boot in progress.
-- MYSQL - Fix a few other minor memory leaks when uncommon failures occur.
-- burst_buffer/cray - Fix race condition that could cause multiple batch job
  launch requests resulting in drained nodes.
-- Correct logic to purge old reservations.
-- Fix DBD cache restore from previous versions.
-- Fix to logic for getting expected start time of existing job ID with
  explicit begin time that is in the past.
-- Clear job's reason of "BeginTime" in a more timely fashion and/or prevents
  them from being stuck in a PENDING state.
-- Make sure acct policy limits imposed on a job are correct after requeue.

* Changes in Slurm 16.05.8
==========================
-- Remove StoragePass from being printed out in the slurmdbd log at debug2
  level.
-- Defer PATH search for task program until launch in slurmstepd.
-- Modify regression test1.89 to avoid leaving vestigial job. Also reduce
  logging to reduce likelyhood of Expect buffer overflow.
-- Do not PATH search for mult-prog launches if LaunchParameters=test_exec is
  enabled.
-- Fix for possible infinite loop in select/cons_res plugin when trying to
  satisfy a job's ntasks_per_core or socket specification.
-- If job is held for bad constraints make it so once updated the job doesn't
  go into JobAdminHeld.
-- sched/backfill - Fix logic to reserve resources for jobs that require a
  node reboot (i.e. to change KNL mode) in order to start.
-- When unpacking a node or front_end record from state and the protocol
  version is lower than the min version, set it to the min.
-- Remove redundant lookup for part_ptr when updating a reservation's nodes.
-- Fix memory and file descriptor leaks in slurmd daemon's sbcast logic.
-- Do not allocate specialized cores to jobs using the --exclusive option.
-- Cancel interactive job if Prolog failure with "PrologFlags=contain" or
  "PrologFlags=alloc" configured. Send new error prolog failure message to
  the salloc or srun command as needed.
-- Prevent possible out-of-bounds read in slurmstepd on an invalid #! line.
-- Fix check for PluginDir within slurmctld to work with multiple directories.
-- Cancel interactive jobs automatically on communication error to launching
  srun/salloc process.
-- Fix security issue caused by insecure file path handling triggered by the
  failure of a Prolog script. To exploit this a user needs to anticipate or
  cause the Prolog to fail for their job. CVE-2016-10030.

* Changes in Slurm 16.05.7
==========================
-- Fix issue in the priority/multifactor plugin where on a slurmctld restart,
  where more time is accounted for than should be allowed.
-- cray/busrt_buffer - If total_space in a pool decreases, reset used_space
  rather than trying to account for buffer allocations in progress.
-- cray/busrt_buffer - Fix for double counting of used_space at slurmctld
  startup.
-- Fix regression in 16.05.6 where if you request multiple cpus per task (-c2)
  and request --ntasks-per-core=1 and only 1 task on the node
  the slurmd would abort on an infinite loop fatal.
-- cray/busrt_buffer - Internally track both allocated and unusable space.
  The reported UsedSpace in a pool is now the allocated space (previously was
  unusable space). Base available space on whichever value leaves least free
  space.
-- cray/burst_buffer - Preserve job ID and don't translate to job array ID.
-- cray/burst_buffer - Update "instance" parsing to match updated dw_wlm_cli output.
-- sched/backfill - Insure we don't try to start a job that was already started and requeued by the main scheduling logic.
-- job_submit/lua - add access to the job features field in job_record.
-- select/linear plugin modified to better support heterogeneous clusters when topology/none is also configured.
-- Permit cancellation of jobs in configuring state.
-- acct_gather_energy/rapl - prevent segfault in slurmd from race to gather data at slurmd startup.
-- Integrate node_feature/knl_generic with "hbm" GRES information.
-- Fix output routines to prevent rounding the TRES values for memory or BB.
-- switch/cray plugin - fix use after free error.
-- docs - elaborate on how to clear TRES limits in sacctmgr.
-- knl_cray plugin - Avoid abort from backup slurmdctld at start time.
-- cgroup plugins - fix two minor memory leaks.
-- If a node is booting for some job, don't allocate additional jobs to the node until the boot completes.
-- testsuite - fix job id output in test17.39.
-- Modify backfill algorithm to improve performance with large numbers of running jobs. Group running jobs that end in a "similar" time frame using a time window that grows exponentially rather than linearly. After one second of wall time, simulate the termination of all remaining running jobs in order to respond in a reasonable time frame.
-- Fix slurm_job_cpus_allocated_str_on_node_id() API call.
-- sched/backfill plugin: Make malloc match data type (defined as uint32_t and allocated as int).
-- srun - prevent segfault when terminating job step before step has launched.
-- sacctmgr - prevent segfault when trying to reset usage for an invalid account name.
-- Make the openssl crypto plugin compile with openssl >= 1.1.
-- Fix SuspendExcNodes and SuspendExcParts on slurmdctld reconfiguration.
-- sbcast - prevent segfault in slurmd due to race condition between file transfers from separate jobs using zlib compression
-- cray/burst_buffer - Increase time to synchronize operations between threads from 5 to 60 seconds ("setup" operation time observed over 17 seconds).
-- node_features/knl_cray - Fix possible race condition when changing node state that could result in old KNL mode as an active features.
-- Make sure if a job can't run because of resources we also check accounting limits after the node selection to make sure it doesn't violate those limits and if it does change the reason for waiting so we don't reserve resources on jobs violating accounting limits.
-- NRT - Make it so a system running against IBM's PE will work with PE version 1.3.
-- NRT - Make it so protocols pgas and test are allowed to be used.
-- NRT - Make it so you can have more than 1 protocol listed in MP_MSG_API.
-- cray/burst_buffer - If slurmdctld daemon restarts with pending job and burst buffer having unknown file stage-in status, teardown the buffer, defer the job, and start stage-in over again.
-- On state restore in the slurmdctld don't overwrite the mem_spec_limit given from the slurmd.conf when using FastSchedule=0.
-- Recognize a KNL's proper NUMA count (rather than setting it to the value in slurm.conf) when using FastSchedule=0.
-- Fix parsing in regression test1.92 for some prompts.
-- sbcast - use slurmd's gid cache rather than a separate lookup.
-- slurmd - return error if setgroups() call fails in _drop_privileges().
-- Remove error messages about gres counts changing when a job is resized on a slurmdctld restart or reconfig, as they aren't really error messages.
-- Fix possible memory corruption if a job is using GRES and changing size.
-- jobcomp/elasticsearch - fix printf format for a value on 32-bit builds.
-- task/cgroup - Change error message if CPU binding can not take place to better identify the root cause of the problem.
-- Fix issue where task/cgroup would not always honor --cpu_bind=threads.
-- Fix race condition in with getgrouplist() in slurmd that can lead to user accounts being granted access to incorrect group memberships during job launch.

* Changes in Slurm 16.05.6
==========================
-- Docs - the correct default value for GroupUpdateForce is 0.
-- mpi/pmix - improve point to point communication performance.
-- SlurmDB - include pending jobs in search during 'sacctmgr show runawayjobs'.
-- Add client side out-of-range checks to --nice flag.
-- Fix support for sbatch "-W" option, previously needed to use "--wait".
-- node_features/knl_cray plugin and capmc_suspend/resume programs modified to sleep and retry capmc operations if the Cray State Manager is down. Added CapmcRetries configuration parameter to knl_cray.conf.
-- node_features/knl_cray plugin: Remove any KNL MCDRAM or NUMA features from node's configuration if capmc does NOT report the node as being KNL.
-- node_features/knl_cray plugin: drain any node not reported by "capmc node_status" on startup or reconfig.
-- node_features/knl_cray plugin: Substantially streamline and speed up logic to load current node state on reconfigure failure or unexpected node boot.
-- node_features/knl_cray plugin: Add separate thread to interact with capmc in response to unexpected node reboots.
-- node_features plugin - Add "mode" argument to node_features_p_node_xlate() function to fix some bugs updating a node's features using the node update RPC.
-- node_features/knl_cray plugin: If the reconfiguration of nodes for an interactive job fails, kill the job (it can't be requeued like a batch job).
-- Testsuite - Added srun/salloc/sbatch tests with --use-min-nodes option.
-- Fix typo when an error occurs when discovering pmix version on configure.
-- Fix configuring pmix support when you have your lib dir symlinked to lib64.
-- Fix waiting reason if a job is waiting for a specific limit instead of always just AccountingPolicy.
-- Correct SchedulerParameters=bf_busy_nodes logic with respect to the job's minimum node count. Previous logic would not decrement counter in some locations and reject valid job request for not reaching minimum node count.
-- Fix FreeBSD-11 build by using llabs() function in place of abs().
-- Cray: The slurmd can manipulate the socket/core/thread values reported based upon the configuration. The logic failed to consider select/cray with SelectTypeParameters=other_cons_res as equivalent to select/cons_res.
-- If a node's socket or core count are changed at registration time (e.g. a
KNL node's NUMA mode is changed), change it's board count to match.
-- Prevent possible divide by zero in select/cons_res if a node's board count
is higher than it's socket count.
-- Allow an advanced reservation to contain a license count of zero.
-- Preserve non-KNL node features when updating the KNL node features for a
multi-node job in which the non-KNL node features vary by node.
-- task/affinity plugin: Honor a job's --ntasks-per-socket and
--ntasks-per-core options in task binding.
-- slurmd - do not print ClusterName when using 'slurmd -C'.
-- Correct a bitmap test function (used only by the select/bluegene plugin).
-- Do not propagate SLURM_UMASK environment variable to batch script.
-- Added node_features/knl_generic plugin for KNL support on non-Cray systems.
-- Cray: Prevent abort in backfill scheduling logic for requeued job that has
been cancelled while NHC is running.
-- Improve reported estimates of start and end times for pending jobs.
-- pbsnodes: Show OS value as "unknown" for down nodes.
-- BlueGene - correctly scale node counts when enforcing MaxNodes limit take 2.
-- Fix "sbatch --hold" to set JobHeldUser correctly instead of JobHeldAdmin.
-- Cray - print warning that task/cgroup is required, and must be after
task/cray in the TaskPlugin settings.
-- Document that node Weight takes precedence over load with LLN scheduling.
-- Fix issue where gang scheduling could happen even with OverSubscribe=NO.
-- Expose JOB_SHARED_* values to job_submit/lua plugin.
-- Fix issue where number of nodes is not properly allocated when srun is
requested with --tasks < hosts from -w hostlist.
-- Update srun documentation for -N, -w and -m arbitrary.
-- Fix bug that was clearing MAINT mode on nodes scheduled for reboot (bug
introduced in version 16.05.5 to address bug in overlapping reservations).
-- Add logging of node reboot requests.
-- Docs - remove recommendation for ReleaseAgent setting in cgroup.conf.
-- Make sure a job cleans up completely if it has a node fail. Mostly an
issue with gang scheduling.

* Changes in Slurm 16.05.5
=================================
-- Fix accounting for jobs requeued after the previous job was finished.
-- slurmd modified to pre-load all relevant plugins at startup to avoid
the possibility of modified plugins later resulting in inconsistent API
or data structures and a failure of slurmd.
-- Export functions from parse_time.c in libslurm.so.
-- Export unit convert functions from slurm_protocol_api.c in libslurm.so.
-- Fix scancel to allow multiple steps from a job to be cancelled at once.
-- Update and expand upgrade guide (in Quick Start Administrator web page).
-- burst_buffer/cray: Requeue, but do not hold a job which fails the pre_run
operation.
-- Insure reported expected job start time is not in the past for pending jobs.
-- Add support for PMIx v2.
-- mpi/pmix: support for passing TMPDIR path through info key
-- Cray: update slurmconfgen_smw.py script to correctly identify service nodes
versus compute nodes.
-- FreeBSD - fix build issue in knl_cray plugin.

(continues on next page)
-- Corrections to gres.conf parsing logic.
-- Make partition State independent of EnforcePartLimits value.
-- Fix multipart srun submission with EnforcePartLimits=NO and job violating
  the partition limits.
-- Fix problem updating job state_reason.
-- pmix - Provide HWLOC topology in the job-data if Slurm was configured
  with hwloc.
-- Cray - Fix issue restoring jobs when blade count increases due to hardware
  reconfiguration.
-- burst_buffer/cray - Hold job after 3 failed pre-run operations.
-- sched/backfill - Check that a user's QOS is allowed to use a partition
  before trying to schedule resources on that partition for the job.
-- sacctmgr - Fix displaying nodenames when printing out events or
  reservations.
-- Fix mpiexec wrapper to accept task count with more than one digit.
-- Add mpiexec man page to the script.
-- Add sallct_wait_nodes option to the SchedulerParameters parameter in the
  slurm.conf file controlling when the sallct command returns in relation to
  when nodes are ready for use (i.e. booted).
-- Handle case when slurmctld daemon restart while compute node reboot in
  progress. Return node to service rather than setting DOWN.
-- Preserve node "RESERVATION" state when one of multiple overlapping
  reservations ends.
-- Restructure srun command locking for task_exit processing logic for improved
  parallelism.
-- Modify srun task completion handling to only build the task/node string for
  logging purposes if it is needed. Modified for performance purposes.
-- Docs - update sallct/sbact/srun man pages to mention corresponding
  environment variables for --mem/--mem-per-cpu and allowed suffixes.
-- Silence srun warning when overriding the job ntasks-per-node count
  with a lower task count for the step.
-- Docs - assorted spelling fixes.
-- node_features/knl_cray: Fix bug where MCDRAM state could be taken from
  capmc rather than cnselect.
-- node_features/knl_cray: If a node is rebooted outside of Slurm's direction,
  update it's active features with current MCDRAM and NUMA mode information.
-- Restore ability to manually power down nodes, broken in 15.08.12.
-- Don't log error for job end_time being zero if node health check is still
  running.
-- When powering up a node to change it's state (e.g. KNL NUMA or MCDRAM mode)
  then pass to the ResumeProgram the job ID assigned to the nodes in the
  SLURM_JOB_ID environment variable.
-- Allow a node's PowerUp state flag to be cleared using update_node RPC.
-- capmc_suspend/resume - If a request modify NUMA or MCDRAM state on a set of
  nodes or reboot a set of nodes fails then just requeue the job and abort the
  entire operation rather than trying to operate on individual nodes.
-- node_features/knl_cray plugin: Increase default CapmcTimeout parameter from
  10 to 60 seconds.
-- Fix squeue filter by job license when a job has requested more than 1
  license of a certain type.
-- Fix bug in PMIX_Ring in the pmi2 plugin so that it supports singleton mode.
  It also updates the testpmixring.c test program so it can be used to check

(continues on next page)
singleton runs.
-- Automically clean up task/cgroup cpuset and devices cgroups after steps are
completed.
-- Testsuite - Fix test1.83 to handle gaps in node names properly.
-- BlueGene - correctly scale node counts when enforcing MaxNodes limit.
-- Make sure no attempt is made to schedule a requeded job until all steps are
completed (Node Health Check completes for all steps on a Cray).
-- KNL: Correct task affinity logic for some NUMA modes.
-- Add salloc/sbatch/srun --priority option of "TOP" to set job priority to
the highest possible value. This option is only available to Slurm operators
and administrators.
-- Add salloc/sbatch/srun option --use-min-nodes to prefer smaller node counts
when a range of node counts is specified (e.g. "-N 2-4").
-- Validate salloc/sbatch --wait-all-nodes argument.
-- Add "sbatch_wait_nodes" to SchedulerParameters to control default sbatch
behaviour with respect to waiting for all allocated nodes to be ready for
use. Job can override the configuration option using the --wait-all-nodes=#
option.
-- Prevent partition group access updates from resetting last_part_update when
no changes have been made. Prevents backfill scheduler from restarting
mid-cycle unnecessarily.
-- Cray - add NHC_ABSOLUTELY_NO to never run NHC, even on certain edge cases
that it would otherwise be run on with NHC_NO.
-- Ignore GRES/QOS updates that maintain the same value as before.
-- mpi/pmix - prepare temp directory for application.
-- Fix display for the nice and priority values in sprio/scontrol/squeue.

* Changes in Slurm 16.05.4
==========================
-- Fix potential deadlock if running with message aggregation.
-- Streamline when schedule() is called when running with message aggregation
on batch script completes.
-- Fix incorrect casting when [un]packing derived_ec on slurmdb_job_rec_t.
-- Document that persistent burst buffers can not be created or destroyed using
the salloc or srun --bb options.
-- Add support for setting the SLURM_JOB_ACCOUNT, SLURM_JOB_QOS and
SLURM_JOB_RESERVAION environment variables are set for the salloc command.
Document the same environment variables for the salloc, sbatch and srun
commands in their man pages.
-- Fix issue where sacctmgr load cluster.cfg wouldn't load associations
that had a partition in them.
-- Don't return the extern step from sstat by default.
-- In sstat print 'extern' instead of 4294967295 for the extern step.
-- Make advanced reservations work properly with core specialization.
-- Fix race condition in the account_gather plugin that could result in job
stuck in COMPLETING state.
-- Regression test fixes if SelectTypePlugin not managing memory and no node
memory size set (defaults to 1 MB per node).
-- Add missing partition write locks to _slurm_rpc_dump_nodes/node_single to
prevent a race condition leading to inconsistent sinfo results.
-- Fix task:CPU binding logic for some processors. This bug was introduced
in version 16.05.1 to address KNL bunding problem.
-- Fix two minor memory leaks in slurmctld.
-- Improve partition-specific limit logging from slurmctld daemon.
-- Fix incorrect access check when using MaxNodes setting on the partition.
-- Fix issue with sacctmgr when specifying a list of clusters to query.
-- Fix issue when calculating future StartTime for a job.
-- Make EnforcePartLimit support logic work with any ordering of partitions
   in job submit request.
-- Prevent restoration of wrong CPU governor and frequency when using
   multiple task plugins.
-- Prevent slurmd abort if hwloc library fails to populate the "children"
   arrays (observed with hwloc version "dev-333-g85ea6e4").
-- burst_buffer/cray: Add "--groupid" to DataWarp "setup" command.
-- Fix lustre profiling putting it in the Filesystem dataset instead of the
   Network dataset.
-- Fix profiling documentation and code to match be consistent with
   Filesystem instead of Lustre.
-- Correct the way watts is calculated in the rapl plugin when using a poll
   frequency other than AcctGatherNodeFreq.
-- Don't about step launch if job reaches expected end time while node is
   configuring/booting (NOTE: The job end time will be adjusted after node
   becomes ready for use).
-- Fix several print routines to respect a custom output delimiter when
   printing NO_VAL or INFINITE.
-- Correct documented configurations where --ntasks-per-core and
   --ntasks-per-socket are supported.
-- task/affinity plugin buffer allocated too small, can corrupt memory.

* Changes in Slurm 16.05.3
==========================
-- Make it so the extern step uses a reverse tree when cleaning up.
-- If extern step doesn't get added into the proctrack plugin make sure the
   sleep is killed.
-- Fix areas the slurmctld can segfault if an extern step is in the system
   cleaning up on a restart.
-- Prevent possible incorrect counting of GRES of a given type if a node has
   the multiple "types" of a given GRES "name", which could over-subscribe
   GRES of a given type.
-- Add web links to Slurm Diamond Collectors (from Harvard University) and
   collectd (from EDF).
-- Add job_submit plugin for the "reboot" field.
-- Make some more Slurm constants (INFINITE, NO_VAL64, etc.) available to
   job_submit/lua plugins.
-- Send in a -1 for a taskid into spank_task_post_fork for the extern_step.
-- MYSQL - Sightly better logic if a job completion comes in with an end time
   of 0.
-- task/cgroup plugin is configured with ConstrainRAMSpace=yes, then set soft
   memory limit to allocated memory limit (previously no soft limit was set).
-- Document limitations in burst buffer use by the salloc command (possible
   access problems from a login node).
-- Fix proctrack plugin to only add the pid of a process once
   (regression in 16.05.2).
-- Fix for sstat to print correct info when requesting jobid.batch as part of

(continues on next page)
a comma-separated list.

-- CRAY - Fix issue if pid has already been added to another job container.
-- CRAY - Fix add of extern step to AELD.
-- burstbufer/cray: avoid batch submit error condition if waiting for stagein.
-- CRAY - Fix for reporting steps lingering after they are already finished.
-- Testsuite - fix test1.29 / 17.15 for limits with values above 32-bits.
-- CRAY - Simplify when a NHC is called on a step that has unkillable processes.
-- CRAY - If trying to kill a step and you have NHC_NO_STEPS set run NHC anyway to attempt to log the backtraces of the potential unkillable processes.
-- Fix gang scheduling and license release logic if single node job killed on bad node.
-- Make scontrol show steps show the extern step correctly.
-- Do not scheduled powered down nodes in FAILED state.
-- Do not start slurmctld power_save thread until partition information is read in order to prevent race condition that can result invalid pointer when trying to resolve configured SuspendExcParts.
-- Add SLURM_PENDING_STEP id so it won't be confused with SLURM_EXTERN_CONT.
-- Fix for core selection with job --gres-flags=enforce-binding option.
    Previous logic would in some cases allocate a job zero cores, resulting in slurmctld abort.
-- Minimize preempted jobs for configurations with multiple jobs per node.
-- Improve partition AllowGroups caching. Update the table of UIDs permitted to use a partition based upon it's AllowGroups configuration parameter as new valid UIDs are found rather than looking up that user's group information for every job they submit. If the user is now allowed to use the partition, then do not check that user's group access again for 5 seconds.
-- Add routing queue information to Slurm FAQ web page.
-- Do not select_g_step_finish() a SLURM_PENDING_STEP step, as nothing has been allocated for the step yet.
-- Fixed race condition in PMIX Fence logic.
-- Prevent slurmctld abort if job is killed or requeued while waiting for reboot of its allocated compute nodes.
-- Treat invalid user ID in AllowUserBoot option of knl.conf file as error rather than fatal (log and do not exit).
-- qsub - When doing the default output files for an array in qsub style make them using the master job ID instead of the normal job ID.
-- Create the extern step while creating the job instead of waiting until the end of the job to do it.
-- Always report a 0 exit code for the extern step instead of being canceled or failed based on the signal that would always be killing it.
-- Fix to allow users to update QOS of pending jobs.
-- CRAY - Fix minor memory leak in switch plugin.
-- CRAY - Change slurmconfiggen_smw.py to skip over disabled nodes.
-- Fix eligible_time for elasticsearch as well as add queue_wait (difference between start of job and when it was eligible).

* Changes in Slurm 16.05.2

-------------------
-- CRAY - Fix issue where the proctrack plugin could hang if the container id wasn't able to be made.

(continues on next page)
-- Move test for job wait reason value of BurstBufferResources and BurstBufferStageIn later in the scheduling logic.
-- Document which srun options apply to only job, only step, or job and step allocations.
-- Use more compatible function to get thread name (>= 2.6.11).
-- Fix order of job then step id when noting cleaning flag being set.
-- Make it so the extern step sends a message with accounting information back to the slurmctld.
-- Make it so the extern step calls the select_g_step_start|finish functions.
-- Don't print error when extern step is canceled because job is ending.
-- Handle a few error codes when dealing with the extern step to make sure we have the pids added to the system correctly.
-- Add support for job dependencies with job array expressions. Previous logic required listing each task of job array individually.
-- Make sure tres_cnt is set before creating a slurmdb_assoc_usage_t.
-- Prevent backfill scheduler from starting a second "singleton" job if another one started during a backfill sleep.
-- Fix for invalid array pointer when creating advanced reservation when job allocations span heterogeneous nodes (differing core or socket counts).
-- Fix hostlist_ranged_string_xmalloc_dims to correctly not put brackets on hostlists when brackets == 0.
-- Make sure we don't get brackets when making a range of reserved ports for a step.
-- Change fatal to an error if port ranges aren't correct when reading state for steps.

* Changes in Slurm 16.05.1
==========================
-- Fix __cplusplus macro in spank.h to allow compilation with C++.
-- Fix compile issue with older glibc < 2.12
-- Fix for starting batch step with mpi/pmix plugin.
-- Fix for "scontrol -dd show job" with respect to displaying the specific CPUs allocated to a job on each node. Prior logic would only display the CPU information for the first node in the job allocation.
-- Print correct return code on failure to update active node features through sview.
-- Allow QOS timelimit to override partition timelimit when EnforcePartLimits is set to all/any.
-- Make it so qsub will do a "basename" on a wrapped command for the output and error files.
-- Fix issue where slurmd could core when running the ipmi energy plugin.
-- Documentation - clean up typos.
-- Add logic so that slurmstepd can be launched under valgrind.
-- Increase buffer size to read /proc/*/stat files.
-- Fix for tracking job resource allocation when slurmctld is reconfigured while Cray Node Health Check (NHC) is running. Previous logic would fail to record the job's allocation then perform release operation upon NHC completion, resulting in underflow error messages.
-- Make "scontrol show daemons" work with long node names.
-- CRAY - Collect energy using a uint64_t instead of uint32_t.
-- Fix incorrect if statements when determining if the user has a default account or wckey.
-- Prevent job stuck in configuring state if slurmctld daemon restarted while PrologSlurmctld is running. Also re-issue burst_buffer/pre-load operation as needed.
-- Correct task affinity support for FreeBSD.
-- Fix for task affinity on KNL in SNC2/Flat mode.
-- Recalculate a job's memory allocation after node reboot if job requests all of a node's memory and FastSchedule=0 is configured. Intel KNL memory size can change on reboot with various MCDRAM modes.
-- Fix small memory leak when printing HealthCheckNodeState.
-- Eliminate memory leaks when AuthInfo is configured.
-- Improve sdiag output description in man page.
-- Cray/capmc_resume script modify a node's features (as needed) when the reinit (reboot) command is issued rather than wait for the nodes to change to the "on" state.
-- Correctly print ranges when using step values in job arrays.
-- Allow from file names / paths over 256 characters when launching steps, as well as spaces in the executable name.
-- job_submit.license.lua example modified to send message back to user.
-- Document job --mem=0 option means all memory on a node.
-- Set SLURM_JOB_QOS environment variable to QOS name instead of description.
-- knl_cray.conf file option of CnselectPath added.
-- node_features/knl_cray plugin modified to get current node NUMA and MCDRAM modes using cnselect command rather than capmc command.
-- liblua - add SLES12 paths to runtime search list.
-- Fix qsub default output and error files for task arrays.
-- Fix qsub to set job_name correctly when wrapping a script (-b y)
-- Cray - set EnforcePartLimits=any in slurm.conf template.

* Changes in Slurm 16.05.0
================================
-- Update seff to fix warnings with ncpus, and list slurm-perlapi dependency in spec file.
-- Fix testsuite to consistent use /usr/bin/env {bash,expect} construct.
-- Cray - Ensure that step completion messages get to the database.
-- Fix step cpus_per_task calculation for heterogeneous job allocation.
-- Fix --with-json= configure option to use specified path.
-- Add back thread_id to "thread_id" LogTimeFormat to distinguish between multiple threads with the same name. Now displays thread name and id.
-- Change how Slurm determines the NUMA count of a node. Ignore KNL NUMA that only include memory.
-- Cray - Fix node list parsing in capmc_suspend/resume programs.
-- Fix sbatch #BSUB parsing for -W and -M options.
-- Fix GRES task layout bug that could cause slurmctld to abort.
-- Fix to --gres-flags=enforce-binding logic when multiple sockets needed.
6.1 Under Revision

The Programmer’s Guide is currently under revision.
7.1 Preface

Welcome to the Scyld ClusterWare Reference Guide. This document describes Scyld ClusterWare commands that can be invoked by the ordinary user and the maintenance utilities that are intended for the cluster administrator. It also provides in-depth information on the configuration file /etc/beowulf/config and the cluster node file system table /etc/beowulf/fstab. The document also includes an extensive library/function reference for the beostat Beowulf Status library and the BProc Beowulf Process Control library.

The Scyld Beowulf functionality is implemented through several packages, notably the following:

- The BProc package, which implements the Scyld BProc unified process space functionality.
- The libbeostat package, which monitors the state of the compute nodes and gathers performance metrics, making these metrics available through a library API.

This Reference Guide is written with the assumption that the reader has a background in a Linux or Unix operating environment. Therefore, this document does not cover basic Linux system use, administration, or application development.

7.2 Scyld ClusterWare Commands

This section of the Reference Guide describes the Scyld ClusterWare commands that are intended to be invoked by the ordinary user. Most of the commands are found in the directory /opt/scyld/bin.

7.2.1 beoboot

7.2.1.1 Name

beoboot – Generate Scyld ClusterWare boot images
7.2.1.2 Synopsis


7.2.1.3 Description

beoboot is a script that builds images to boot for Scyld compute nodes.

The final boot image is provided by one or more master nodes designated as “boot masters”. This final boot image has the run-time kernel and initial information needed to contact an operational master.

7.2.1.4 Options

- **-h** Display a help message and exit.
- **-v** Display version information and exit.
- **-2** Create a phase 2 image. This image contains the final kernel to run.
- **-i** Create stand-alone images (kernel and ramdisk). These images will be appropriate for use with other boot mechanisms. The kernel and ramdisk image will be stored in: outfile and outfile.initrd
- **-n** Create a netboot image. If no output_file argument is specified, then the image file will be named /var/beowulf/boot.img.
- **-o output_file** Set output filename to output_file.
- **-L, --libdir dir** Find beoboot files in dir instead of /usr/lib/beoboot/.
- **-k, --kernel kernimg** Use kernimg as the kernel image instead of the image given in the configuration file (final boot image only).
  - If this is not specified on the command line, the default is taken out of /etc/beowulf/config. If it is not specified there, /boot/vmlinuz is used.
- **-c, --cmdline cmdline** Use the command line cmdline instead of the default “kernelcommandline” line found in the etc/beowulf/config config file.
- **-m, --modules dir** Look for modules matching the kernel image in dir instead of /lib/modules/ <kernelversion>, which is the default.

Caution

When you are making a final boot image, you must be running the kernel you are putting in the image, whether this kernel is specified on the command line or in /etc/beowulf/config.

7.2.1.5 Examples

Creating a final boot image:

```
Building phase 2 file system image in /tmp/beoboot.6684...
ram disk image size (uncompressed): 1888K
compressing...done
ram disk image size (compressed): 864K
Kernel image is:  "/tmp/beoboot.6684".
```
Initial ramdisk is: "/tmp/beoboot.6684.initrd".
Netboot image is in: /var/beowulf/boot.img

### 7.2.2 beoconfig

#### 7.2.2.1 Name

**beoconfig** – View or manipulate a Scyld ClusterWare configuration files.

#### 7.2.2.2 Synopsis

```
```

#### 7.2.2.3 Description

**beoconfig** manipulates a Scyld ClusterWare configuration file to insert, replace, or delete “string” entries. A “string” consists of an initial keyword, plus zero or more parameters, plus an optional comment. This utility is commonly used in script files to retrieve parameters from the config file.

#### 7.2.2.4 Options

The following options are available to the **beoconfig** program.

- **-a, --all search-string**  
  Return all entries with specified `search-string` keyword.

- **-c, --config file**  
  Read configuration file. Default is `/etc/beowulf/config`.

- **-d, --delete string**  
  Delete the specified `string` from the config file.

- **-D, --deleteall string**  
  Delete all instances of specified `string` from the config file.

- **-i, --insert string**  
  Append the specified `string` to the config file if it does not already exist. When inserting a “node” entry, you must also specify a `--node nodes` argument.

- **-h, --help**  
  Show a usage message.

- **-i, --insert string**  
  Append the specified `string` to the config file if it does not already exist. When inserting a “node” entry, you must also specify a `--node nodes` argument.

- **-l, --syslog**  
  Log error messages to the syslog (`/var/log/messages`).

- **-n, --node nodes**  
  Perform action on specified `nodes` only. Nodes can be specified individually, as ranges, or as a comma-separated list of individual nodes and/or ranges.

- **-r, --replace <string1 string2>**  
  Replace `string1` with `string2`.

- **-u, --usage**  
  Show a usage summary.

- **-V, --version**  
  Show this version number.

- **-w, --withcomments**  
  Show entries including comments.
7.2.2.5 Examples

[user@cluster user]$ export CLUSTERDEV="beocfg interface"
[user@cluster user]$ $CLUSTERDEV
eth1

View MAC addresses for all nodes in /etc/beowulf/config:

[user@cluster user]$ beocfg -a "node"
00:50:45:01:03:68 00:50:45:01:03:69
00:50:45:5C:29:F6 00:50:45:5C:29:F7
00:50:45:BB:A6:EA 00:50:45:BB:A6:EB
off
off
00:50:45:CD:BE:61

Demonstrate --withcomments

[root@cluster ~]# beocfg --insert "newkeyword param1 # with comment"
[root@cluster ~]# beocfg newkeyword
param1
[root@cluster ~]# beocfg -w newkeyword
param1 # with comment

# Now remove the unnecessary "newkeyword" entry:
[root@cluster ~]# beocfg -d newkeyword

Enable only Nodes 1, 2 and 8, to use IPMI

[root@cluster ~]# beocfg --node 1 --insert "ipmi enabled"
[root@cluster ~]# beocfg --node 8 --insert "ipmi enabled"
[root@cluster ~]# beocfg --node 2 --insert "ipmi enabled"
[root@cluster ~]# beocfg --node 1 "ipmi"
enabled
[root@cluster ~]# beocfg --node 3 "ipmi"
disabled

Looking in /etc/beowulf/config, one will see that the 'ipmi' keyword has a global value of 'disabled'. However, there is another 'ipmi' keyword entry, and this one has an embedded node-set.

[root@cluster ~]# cat /etc/beowulf/config | grep ipmi
ipmi disabled
ipmi 1-2,8 enabled

Replace functionality: Suppose the config file has a 'nodewake' line to invoke an IPMI version 2.0 script for nodes 1 through 10, but node 5 has been replaced with a machine that supports only IPMI version 1.5. An admin must now replace node 5's nodewake script.

[root@cluster ~]# cat /etc/beowulf/config | grep nodewake
#nodewake /usr/lib/beoboot/bin/node_wake_ipmi
nodewake 1-10 /nfs/support/scripts/nodeup_ipmi2.0
[root@cluster ~]# beocfg --node 5 --replace "nodewake *" \
"nodewake /nfs/support/scripts/nodeup_ipmiv1.5"
[root@cluster ~]# cat /etc/beowulf/config | grep nodewake
#nodewake /usr/lib/beoboot/bin/node_wake_ipmi

(continues on next page)
Insert node-holes for node's 0 through 9, while adding a MAC address for node 10

```
[root@cluster ~]# beoconfig -a node
[root@cluster ~]# beoconfig -i "node 00:11:22:33:44:55 " --node 10
[root@cluster ~]# beoconfig -a node
off ##node number 000
off ##node number 001
off ##node number 002
off ##node number 003
off ##node number 004
off ##node number 005
off ##node number 006
off ##node number 007
off ##node number 008
off ##node number 009
node 00:11:22:33:44:55
```

Get the MAC address for node 10

```
[root@cluster ~]# beoconfig -n 10 node
00:11:22:33:44:55
```

### 7.2.3 beomap

#### 7.2.3.1 Name

**beomap** – Show a job map from the beomap scheduler.

#### 7.2.3.2 Synopsis

```
```

#### 7.2.3.3 Description

This program retrieves a job map from the currently installed beomap scheduler. This is the same job map that would be used by an integrated application (such as **beorun** or **mpprun**) started with the same scheduling parameters at that instant in time.

The **beomap** command may be used to generate a job map for applications that do not have their own scheduler interface, in scripts, or to examine the current scheduling state of the system.

You can influence the job map either by setting environment variables or by entering command line options. Note that command-line options take precedence over the environment variable settings.
### 7.2.3.4 Options

The following general command line options are available to `beomap`. Also see the next section, which describes the job map parameters.

- `-h, --help` Print the command usage message and exit. If `-h` is in the option list, all other options will be ignored.

- `-V, --version` Print the command version number and exit. Any other options will be parsed and handled.

You can influence the `beomap` job map either by entering command line options or by setting environment variables. Following are the available command line options, together with their equivalent environment variables. Note that the command line options take precedence over the environment variables.

All the `beomap` job map parameters listed below can also be used directly with `beorun` and `mpprun`.

- `--all-cpus` Create a process map consisting of all “up” nodes, with each node number repeated to represent the number of CPUs on that node. This parameter is not allowed in conjunction with the `--map` parameter.

- `--all-nodes` Create a process map consisting of all “up” nodes, with one CPU mapped on each of the “up” nodes. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `ALL_NODES`.

- `--all-local` Create a process map consisting entirely of master node entries. This option eliminates everything except node `-1` from the pool of candidate node numbers, thus forcing the map to use node `-1` (the master node) for everything.

  The equivalent environment variable is `ALL_LOCAL`.

- `--no-local` Exclude the master in the process map. This option is essentially a syntactic shortcut for including `-1` in the `--exclude nodelist` option. For MPI jobs, this option puts the “rank 0” job on a compute node instead of on the master node. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `NO_LOCAL`.

- `--exclude nodelist` Build a process map that excludes listed nodes. The `nodelist` consists of a colon-delimited list. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `EXCLUDE=nodelist`.

- `--map nodelist` Explicitly specify a process map consisting of a colon-delimited list of nodes. Each node in `nodelist` indicates where one process will be assigned. The number of entries in the job map implies the number of ranks in the job.

  Listing a node more than once in the list will assign multiple processes to that node. Typically, this is done to assign one process to each processor (or core) on a node, but this can also be used to “oversubscribe”, i.e., to assign more processes to a node than it has processors (or cores).

  The equivalent environment variable is `BEOWULF_JOB_MAP=nodelist`.

- `--np num-processes` Specify the number of processes to run. The `beomap` command attempt to place one process per processor (or core), but will “oversubscribe” and assign multiple processes per processor (or core) if there are not enough individual processors or cores available. This parameter is not allowed in conjunction with the `--map` parameter.
The equivalent environment variable is `NP=num-processes`.

The environment variables have an order of priority. The `BEOWULF_JOB_MAP` variable acts as a “master override” for the other environment variables. If `BEOWULF_JOB_MAP` is not set, then the following priorities apply:

Three of the environment variables determine how many ranks to schedule in the map: (1) `ALL_CPUS`, (2) `ALL_NODES`, and (3) `NP`. If none of these are set explicitly by the user, then `NP=1` is the default.

Three of the environment variables determine what node numbers are candidates for being mapped: (1) `ALL_LOCAL`, (2) `NO_LOCAL`, and (3) `EXCLUDE`.

Note: it is improper to use `NO_LOCAL` and `ALL_LOCAL` together. If both are used, then `ALL_LOCAL` takes precedence.

### 7.2.3.5 Examples

Find the set of machines available for use:

```plaintext
[user@cluster ~]$ beomap --all-cpus
```

Create a process map to run 20 processes on a cluster with 10 idle dual-processor compute nodes:

```plaintext
[user@cluster user]$ beomap --np 20
```

Note: Since `--no-local` was not specified, then the master node (listed as “-1”) is included in the map, and node 9 is listed only once.

Select an available machine to start up an application, while handling application termination or machine failure; note that the following works only for the sh family of shells (bash):

```plaintext
[user@cluster user]$ while ::; do export NODE='beomap --no-local -np 1'; \
   bpsh $NODE application-to-run; done
```

Provide an explicit map to run 5 processes on node 0:

```plaintext
[user@cluster user]$ beomap --np 5 --map 0:0:0:0:0
```

### 7.2.3.6 Special Notes

The underlying `beomap` system calls pluggable schedulers, which may use arbitrary scheduling inputs. The command line options replace and delete environment variables used by the Scyld-provided default schedulers/mappers, but other schedulers are free to ignore these advisory settings. Specifically, the `beomap` command does not confirm that the parameters, such as `--no-local`, are true in the resulting job map.
7.2.4 beonpc

7.2.4.1 Name

beonpc – Show the count of all user processes started by this master running on the specified compute node.

7.2.4.2 Synopsis


7.2.4.3 Description

The beonpc program prints the count of running processes on the specified cluster node. The count includes only the processes started by the current machine and running on the specified node.

beonpc prints “-1” for nodes that are not controlled by this master or are otherwise inaccessible.

beonpc is typically used to make and observe scheduling and job mapping decisions.

7.2.4.4 Options

The following options are available to the beonpc program.

- **-h, --help, -u, --usage** Print the command usage message on stdout and exit. When one of these options is recognized in the option list, all following options will be ignored.

- **-V** Print the command version number on stdout and exit. Any following options will be ignored.

- **-p, --pids** Show the process IDs in a process list.

- **--sum** Emit only a total cluster process count.

**node** Optionally, show for the specific node number, or all (the default) for all nodes, or list for nodes with a nonzero count.

7.2.4.5 Example

Find the number of jobs this master is running on cluster compute node 23:

```
[user@cluster user] $ beonpc 23
3
```

7.2.5 beorun

7.2.5.1 Name

beorun – Run a job on a Scyld cluster using dynamically selected nodes.
7.2.5.2 Synopsis


7.2.5.3 Description

The `beorun` program runs the specified program on a dynamically selected set of cluster nodes. It generates a job map from the currently installed `beomap` scheduler, and starts the program on each node specified in the map. The scheduling parameters from the command line and environment are the same as for `beomap`, and the resulting job map is identical to the job map that `beomap` would generate at that instant in time for that program name.

The `beorun` command may be used to start applications that are not cluster-aware or do not have their own scheduler interface.

7.2.5.4 Options

The following general command line options are available to `beorun`. Also see the next section, which describes the `beomap` job map parameters.

- `-h, --help, -u, --usage` Print the command usage message on `stdout` and exit. When one of these options is recognized in the option list, all following options will be ignored.
- `-V` Print the command version number on `stdout` and exit. Any following options will be ignored.

You can influence the `beorun` job map either by entering command line options or by setting environment variables. Following are the available command line options, together with their equivalent environment variables. Note that the command line options take precedence over the environment variables.

All the `beorun` job map parameters listed below can also be used directly with `beomap` and `mpprun`.

- `--all-cpus` Create a process map consisting of all “up” nodes, with each node number repeated to represent the number of CPUs on that node. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `ALL_CPUS`.

- `--all-nodes` Create a process map consisting of all “up” nodes, with one CPU mapped on each of the “up” nodes. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `ALL_NODES`.

- `--all-local` Create a process map consisting entirely of master node entries. This option eliminates everything except node -1 from the pool of candidate node numbers, thus forcing the map to use node -1 (the master node) for everything.

  The equivalent environment variable is `ALL_LOCAL`.

- `--no-local` Exclude the master in the process map. This option is essentially a syntactic shortcut for including -1 in the `--exclude nodelist` option. For MPI jobs, this option puts the “rank 0” job on a compute node instead of on the master node. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `NO_LOCAL`.

- `--exclude nodelist` Build a process map that excludes listed nodes. The `nodelist` consists of a colon-delimited list. This parameter is not allowed in conjunction with the `--map` parameter.

  The equivalent environment variable is `EXCLUDE=nodelist`. 

376 Chapter 7. Reference Guide
--map nodelist
Explicitly specify a process map consisting of a colon-delimited list of nodes. Each node in nodelist indicates where one process will be assigned. The number of entries in the job map implies the number of ranks in the job.

Listing a node more than once in the list will assign multiple processes to that node. Typically, this is done to assign one process to each processor (or core) on a node, but this can also be used to “oversubscribe”, i.e., to assign more processes to a node than it has processors (or cores).

The equivalent environment variable is BEOWULF_JOB_MAP=nodelist.

--np num-processes
Specify the number of processes to run. The beorun command attempts to place one process per processor (or core), but will “oversubscribe” and assign multiple processes per processor (or core) if there are not enough individual processors or cores available. This parameter is not allowed in conjunction with the --map parameter.

The equivalent environment variable is NP=num-processes.

The environment variables have an order of priority. The BEOWULF_JOB_MAP variable acts as a “master override” for the other environment variables. If BEOWULF_JOB_MAP is not set, then the following priorities apply:

Three of the environment variables determine how many ranks to schedule in the map: (1) ALL_CPUS, (2) ALL_NODES, and (3) NP. If none of these are set explicitly by the user, then NP=1 is the default.

Three of the environment variables determine what node numbers are candidates for being mapped: (1) ALL_LOCAL, (2) NO_LOCAL, and (3) EXCLUDE.

Note: it is improper to use NO_LOCAL and ALL_LOCAL together. If both are used, then ALL_LOCAL takes precedence.

Unrecognized options and invalid option formats are reported on stderr and the command exits with exit status 1 (invalid option) or 2 (no command specified or invalid command).

NOTE: beorun does not pass information from stdin to applications. In cases where an application must read data from stdin, it is suggested that bpsh be used instead. Please see the bpsh man page for usage information; command line options for bpsh are similar to those for beorun, but not exactly the same.

### 7.2.5.5 Examples

Run uptime on any two available cluster compute nodes:

```
11:05am up 2 days, 11:16, 0 users, load average: 0.05, 0.24, 0.65
11:05am up 2 days, 11:16, 0 users, load average: 0.01, 0.07, 0.37
```

### 7.2.6 beosi

#### 7.2.6.1 Name

beosi – Collects or extracts cluster configuration information.
7.2.6.2 Synopsis


7.2.6.3 Description

The primary function of the beosi utility is to collect configuration and state information from the master node and/or compute nodes on a Scyld ClusterWare cluster, organize the information into ASCII files within a new directory in the current working directory, and finally tar that directory and uuencode the gzipped tarball into a compressed, portable archive file that can be saved locally or transmitted (e.g., by ftp or email). beosi can also be used to uudecode a previously assembled archive to reform the tarball for later extraction as desired.

beosi should be executed with root access, since much of the interesting information can only be accessed as root. The -m and -n options are typically used together to produce a directory named conf-<date>, where the YY-MM-DD date indicates the current year, month, and day. The -m option creates ASCII files in the subdirectory conf-<date>/master/, and the -n option creates ASCII files in per-node subdirectories, e.g., conf-/Node0/ and conf-/Node1/. The beosi default end product is an archive file named conf-<date>.encoded.

beosi can be used to capture configuration information for later retrieval and comparison. For example, if the current configuration is working, you can execute beosi and store the archive file for safekeeping. If a subsequent configuration change causes your cluster to stop working, then you can create another archive, extract both the new archive and the previous archive, and examine the differences between the two configurations (e.g. using diff) to determine which change caused the problem.

7.2.6.4 Options

The following options are available to the beosi utility.

- **-m** Collect information about the master node, typically used together with the -n option. By default, produces a uuencoded gzipped tar file in the current directory called conf-<date>.encoded

- **-n** Collect information about individual (compute) nodes, typically used together with the -m option. By default, produces a uuencoded gzipped tar file in the current directory called conf-<date>.encoded

- **-I** The -I option overrides the default action of converting the conf-<date> directory of captured information into a uuencoded gzipped tar file. Thus, the directory of information is retained in its fully “exploded” form, conf-<date>, which allows the cluster administrator to view all the information that would otherwise be bundled into the uuencoded gzipped tar file. This is especially useful to discern why beosi might be producing an unexpectedly large uuencoded file.

- **-d file** Decodes information from an archive created previously by beosi. The result is a gzipped tar file with the same root filename.

- **-h** Display a summary of beosi command arguments.

- **-v** Display program version information and exit.
7.2.6.5 Examples

Suppose the current date is May 22, 2018. To inspect the configuration information on the master node, first run:

```bash
[root@cluster ~]# beosi -m
[root@cluster ~]# ls
conf-18-05-22.encoded
```

Then extract the information:

```bash
[root@cluster ~]# beosi -d conf-18-05-22.encoded
[root@cluster ~]# tar -zxvf conf-18-05-22.tar.gz
```

Alternatively, avoid producing a uuencoded tar file and thus retain the fully explorable directory of information:

```bash
[root@cluster ~]# beosi -m -I
[root@cluster ~]# ls
conf-18-05-22
```

Use a prior configuration to identify individual files that differ:

```bash
[root@cluster ~]# diff -r --brief conf-17-12-30 conf-18-05-22
Files conf-17-12-30/master/ifconfig and conf-18-05-22/master/ifconfig differ
Files conf-17-12-30/master/lsmod and conf-18-05-22/master/lsmod differ
Files conf-17-12-30/master/network and conf-18-05-22/master/network differ
Files conf-17-12-30/master/proc_buddyinfo and conf-18-05-22/master/proc_buddyinfo differ
...
```

Use a prior configuration to compare individual files:

```bash
[root@cluster ~]# diff -u conf-17-12-30/master/lsmod conf-18-05-22/master/lsmod
--- conf-17-12-30/master/lsmod 2009-12-14 09:19:45.000000000 -0800
@@ -1,11 +1,12 @@
 Module Size Used by
+iptable_filter 7745 0
 bproc 181208 2
 task_packer 24708 1 bproc
 filecache 28220 2 bproc,task_packer
 ipt_MASQUERADE 9025 1
 iptable_nat 34149 2 ipt_MASQUERADE
 ip_conntrack 57369 2 ipt_MASQUERADE,iptable_nat
-ip_tables 25537 2 ipt_MASQUERADE,iptable_nat
+ip_tables 25537 3 iptable_filter,ipt_MASQUERADE,iptable_nat
 nfsd 274657 17
 exportfs 10945 1 nfsd
 lockd 82833 2 nfsd
```

7.2. Scyld ClusterWare Commands
To gather complete information about the cluster (i.e., master and all compute nodes):

```bash
[root@cluster ~]# beosi -m -n
[root@cluster ~]# ls
    conf-18-05-22.encoded
```

### 7.2.7 beostatus

#### 7.2.7.1 Name

**beostatus** – Display status information about the cluster.

#### 7.2.7.2 Synopsis

```
```

#### 7.2.7.3 Description

**beostatus** is a utility that displays status information for the master node and all compute nodes in the cluster. The default display is a graphical user interface (GUI) known as the “Classic” mode, which is a tabular format, one row per node, showing per-node specific state and resource usage information. An optional non-GUI “Curses” display mode is also available that shows the same per-node information as the “Classic” mode: the assigned number for each node, the node state, CPU usage, memory usage, swap space usage, root filesystem usage, and network bandwidth usage.

Alternate GUI display modes can be selected by **beostatus** command line option or by using a pulldown menu within each of the GUI displays.

Various filtering options are available in the “Curses” display mode that limit the displayed information to nodes that are being currently utilized by a specified user. Note: filtering functionality requires that TORQUE be installed on the cluster.

**beostatus** can also be used to access cluster state information for a remote node. This requires the presence of **beoweb** functionality on the remote node.

See the Administrator’s Guide for additional information about **beostatus**.

#### 7.2.7.4 Options

The following options are available to the **beostatus** utility:

- **--classic**
  
  Display output in GUI “Classic” mode. This is the default display mode.

- **-c, --curses**
  
  Display output in non-GUI “Curses” mode. It displays the same information as the GUI “Classic” mode. It is appropriate for simple terminal windows and when X is unavailable.

- **-C, --combined-spider**
  
  Display output in GUI “Combined Spider” mode.

- **-d, --dots**
  
  Display output in GUI “Dots” mode. Each node is represented by a colored box. The user selects the status element that the box represents (e.g., node state or CPU
utilization), and different colors indicate different status values for that element (e.g., node state “up” vs. “down”, or gradations of CPU loading).

-H, --html Display output in HTML format.
-l, --levometer Display output in GUI “Levometer” mode.
-p, --pie Display output in GUI “Piechart” mode.
-P port, --port=port When retrieving remote beostatus information, override the default port number 5000 with another port value.
-r host, --remote=host Retrieve beostatus information from a remote host.
-s, --stripchart Display output in GUI “Stripchart” mode.
-u seconds, --update=seconds Override the default update rate of 5 seconds. Units are integer seconds.
-U name, --user=name When retrieving remote beostatus information, authenticate as user name.
--disable-ssl When retrieving remote beostatus information, don’t use SSL encryption.
-v, --version Show version information.
-h, --help Show usage information and exit. If -h is one of the first two options, all other options will be ignored. If -h is not one of the first two options, it will be ignored.

### 7.2.7.5 FILTERING OPTIONS

Various filtering options are available when in “Curses” mode. Each is enabled by a single lowercase letter keystroke, and is disabled by a matching uppercase letter keystroke:

**f, F** Limit the display to only those nodes that are running TORQUE jobs for a specific user. If the current user is root, then beostatus prompts for a username; otherwise, the username defaults to the current user.

**j, J** Limit the display to only those nodes that are running TORQUE jobs for a specific user, and yet those jobs have no processes actually executing. If the current user is root, then beostatus prompts for a username; otherwise, the username defaults to the current user.

**p, P** Limit the display to only those nodes that are running processes (irrespective of TORQUE) for a specific user. If the current user is root, then beostatus prompts for a username; otherwise, the username defaults to the current user.

**z, Z** Limit the display to only those nodes that are running TORQUE jobs for any user, and yet those jobs have no processes actually executing.

**q, Q** Terminate the beostatus utility.

### 7.2.7.6 Examples

Print cluster status. Use “q” to exit continuously updating display:

```bash
[user@cluster user] $ beostatus \-c
```

<table>
<thead>
<tr>
<th>Node</th>
<th>State</th>
<th>CPU 0</th>
<th>CPU 1</th>
<th>CPU 2</th>
<th>CPU 3</th>
<th>Memory</th>
<th>Swap</th>
<th>Disk</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>up</td>
<td>0.2%</td>
<td>11.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>12.4%</td>
<td>0.0%</td>
<td>38.2%</td>
<td>36 kbps</td>
</tr>
<tr>
<td>0</td>
<td>up</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>9.6%</td>
<td>0.0%</td>
<td>2.2%</td>
<td>23 kbps</td>
</tr>
<tr>
<td>1</td>
<td>up</td>
<td>90.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>22.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.2%</td>
<td>13 kbps</td>
</tr>
<tr>
<td>2</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.2. Scyld ClusterWare Commands
7.2.8 beostat

7.2.8.1 Name

beostat – Display raw data from the Beostat system.

7.2.8.2 Synopsis


7.2.8.3 Description

The beostat command is a low-level utility that displays data being managed by the beostat package. This data is collected on each compute node and sent by the sendstats daemon to the recvstats daemon on the master node. Other commands, such as beostatus, present a more user-friendly higher-level picture of the cluster.

The default beostat command shows all available information on all “up” nodes. The ––verbose option shows all nodes, including “down” nodes. Various other options constrain the output to show more specific information. You may present multiple options to see multiple data classifications.

7.2.8.4 Options

The following options are available to the beostat command.

- -v, –verbose
  Verbose, display information on all nodes, even if those nodes are down. Normally, beostat displays only “up” nodes.

- -N node-num, –node=node-num
  Show only the data for node node-num rather than for all nodes.

- -c, –cpuinfo
  Display the CPU model information.

- -m, –meminfo
  Display memory statistics.

- -l, –loadavg
  Display load average information.

- -n, –net
  Display network interface information.

- -s, –stat
  Display CPU statistics information.

- -f, –file
  Display root filesystem information.

- -h, –help
  Display brief help and exit.

- -C, –cpupercent
  Convenience option to display CPU info.

- -D, –diskusage
  Convenience option to display disk usage of root filesystem.

- -R, –netrate
  Convenience option to display network rate of all interfaces.

- -I, –idle=threshold
  Convenience option to display number of CPUs more idle than threshold.

- -b, –brief
  Display convenience values (cpupercent, diskusage, netrate, or idle) with no extra text. This eliminates the need to parse the output to obtain the specific values. Only valid when used with one of the convenience options (-C, -D, -R, or -I).

--version
  Display program version information and exit.
7.2.9 bpcp

7.2.9.1 Name

bpcp – Copies files and/or directories between cluster machines.

7.2.9.2 Synopsis

bpcp [-h ] [-v ] [-a ] [-p ] [-r ] [host1: {file1 }] [host2: {file2 }]

7.2.9.3 Description

The bpcp utility is part of the BProc package and is installed by default. It is similar to the Linux rcp command. bpcp will copy files and/or directories between machines. Each file or directory is either a remote file name of the form rhost:path or a local file name. You must have read permission on the source and write permission on the destination. bpcp also handles node-to-node copies, where neither the source nor destination files are on the current node.

7.2.9.4 Options

The following options are available to the bpcp program.

- **-h**
  Print the bpcp usage message and exit. If -h is the first option, all other options will be ignored. If -h is not the first option, the other options will be parsed up to the -h option, but no action will be taken.

- **-v**
  Print the bpcp version number and exit. If -v is the first option, all other options will be ignored. If -v is not the first option, the other options will be parsed up to the -v option, but no action will be taken.

- **-a**
  Copy the local file(s) to every up node. This option does not allow for a host1 option specifying a source node, nor for a host2 option specifying a target node.

- **-p**
  Preserve the attributes of the source files, ignoring the umask. By default, bpcp will modify the time, permission bits, user and group information when the file is copied. This parameter will cause time and permission bits to be unchanged, but the user and group will change to reflect the new user.

- **-r**
  Descend source directory tree recursively and copy files and tree to destination. In this case, the destination must be a directory.

[host1:file1] The name of the file to be copied (and optionally the name of the host it resides on if other than the local host). file1 can be the directory name when used with the -r option.

[host2:file2] The name of the file and/or host where the specified file should be copied. file2 can be a directory name.
7.2.9.5 Examples

Copy file1 from the master node to compute node 1 as file2 in /home/user. Like cp, the directory will not be created if it does not exist.

```
[user@cluster user] $ bpcp /home/user/file1 1:/home/user/file2
```

Copy file1 from the master node to every up compute node as file2 in /home/user. Like cp, the directory will not be created if it does not exist.

```
[user@cluster user] $ bpcp -a /home/user/file1 /home/user/file2
```

Copy all files and sub-directories from compute node 2 in /home/user to compute node 1 in /home/user. The directory tree will be created if it does not exist.

```
[user@cluster user] $ bpcp -r 2:/home/user/ 1:/home/user/
```

Using node 1 as an intermediary, copy file1.txt on node 0 to file1.txt on node 2.

```
[user@cluster user] $ bpsh 1 bpcp 0:/tmp/file1.txt 2:/tmp/
```

Copy /tmp/file.txt from the master node to the /tmp directory on every node in the cluster that is “up”.

```
[user@cluster user] $ bpsh -a bpcp master:/tmp/file1.txt /tmp
```

Note: bpcp will give an “rfork: Invalid argument” message when the node is unreachable.

7.2.10 bpcp

7.2.10.1 Name

bpcp – Set the time on a compute node

7.2.10.2 Synopsis

```
bpcp node
```

7.2.10.3 Description

This program replicates the master node’s time-of-day to a compute node. The program has one required argument: the number of the compute node.

This program is usually run from the node_up script, so that the time gets set on the compute nodes at node boot. After the node is up, BProc’s bpmaster daemon on the master node works with the bpslave daemon on each compute node to synchronize the time-of-day across the cluster.
7.2.10.4 Options

The following options are available to the `bpdate` program.

**node**  The number of the node to set the time on.

7.2.10.5 Examples

Set the time on node 1:

```
[user@cluster user] $ bpdate 1
```

7.2.11 bpsh

7.2.11.1 Name

bpsh, bprsh – Run a command on the indicated node(s).

7.2.11.2 Synopsis

```
```

7.2.11.3 Description

This utility is part of the BProc package and is installed by default on Scyld ClusterWare systems. It is the basic mechanism for running programs on nodes, and it is patterned after the `rsh` and `ssh` commands.

The `targetnodes` can range from -1 (the master) to one less than the number of accessible nodes. `bpsh` will also accept a delimited list of nodes; use `-a` for all nodes that are “up” and `-A` for all nodes that are communicating (e.g., states “up”, “error” and “unavailable”).

`bpsh` forwards `stdin`, `stdout` and `stderr` for the remote processes, unless directed otherwise by `-n` or `-N` arguments. `stdin` will be duplicated for every process on each remote node selected. For a single remote process, the exit status of `bpsh` will be the exit status of that process. Non-normal exit status will also be captured and displayed. For multiple processes, `bpsh` exits with the highest exit status.

`bpsh` throttles the maximum number of command executions that are outstanding at any point in time as `bpsh` services the entire `targetnodes` list. The default fanout is 64, which can be overridden by the environment variable `BPSH_FANOUT=<number>`, which is currently capped at 128. The fanout when using the `-s` (serialize) option is fixed at 1.

The `bprsh` utility is a variant of `bpsh`. See the EXAMPLES.
7.2.11.4 Options

The following options are available to the `bpsh` program.

- **-h**  
  Print the command usage message and exit. If `-h` is the first option, all other options will be ignored. If `-h` is not the first option, the other options will be parsed up to the `-h` option, but no action will be taken.

- **-v**  
  Print the command version number and exit. If `-v` is the first option, all other options will be ignored. If `-v` is not the first option, the other options will be parsed up to the `-v` option, but no action will be taken.

- **-a**  
  Specifies that the command will be run on all nodes in the “up” state.

- **-A num**  
  Specifies that the command will be run on all nodes in either the “up”, “error”, and “unavailable” states. Note that non-root users may get “BProc move failed” errors, since they are only allowed to run on “up” nodes, regardless of other node permissions.

- **-L state**  
  Line buffer output from nodes.

- **-p**  
  Prefix the node number on each output line from the node that sent it.

- **-s**  
  List sequentially all the output from each node.

- **-d**  
  Print a divider line between the sequential output from each node.

- **-b num**  
  Set the IO line buffer size to the number of bytes. The default is 4096.

- **-n**  
  Get stdin from /dev/null. On any read from stdin, /dev/null will return EOF. This is useful for any program that you background or daemonize. Like `rsh`, `bpsh` will not exit immediately if stdin is left open and the program has not completed. `bpsh` assumes the program may want input from stdin.

- **-N**  
  No IO forwarding.

- **-I file, --stdin file**  
  Redirect standard input from the specified file on the remote node.

- **-O file, --stdout file**  
  Redirect standard output to the specified file on the remote node.

- **-E file, --stderr file**  
  Redirect standard error to the specified file on the remote node.

**targetnodes** The node(s) on which to run the command.

**command** The command/program to run.

**command-args** The arguments for command.

7.2.11.5 Examples

Run the `ls` command on nodes -1, 0 and 2, and prefix the node number to each line. Note, due to the way `getopt` works, the master (node -1) cannot be first in the node list:

```
[user@cluster user]$ bpsh 0,-1,2 -p ls /tmp
-1: f1.txt
-1: foo.txt
 0: f3.txt
 2: newfoo.txt
 2: oops.txt
```

Run the `uptime` command on nodes in the “up” state:
Run the same command with a **fanout** value override:

```
[user@cluster user] $ BPSPH_FANOUT=128 bpsh -a -d uptime
```

Run a single instance of the **uptime** command on a node chosen by the scheduler, displaying the node number before the output.

```
[user@cluster user] $ bpsh -p 0-3 uptime
0 2:42pm up 2 days, 23:51, 0 users
1 2:42pm up 3 days, 5:38, 0 users
3 2:42pm up 3 days, 5:38, 0 users
```

Node 2 is down

Run a complex command that consists of multiple commands that displays all the “up” nodes that have been up for less than 24 hours. Note: the **bpsh** utility expects command to be a single command, so to execute multiple commands you must use **bash -c** with the desired command in quotes:

```
[user@cluster user] $ bpsh -sap bash -c "uptime | grep -v days"
```

Alternatively, **bprsh** accepts the more complex command as-is, just as **rsh** would do:

```
[user@cluster user] $ bprsh -sap "uptime | grep -v days"
```

### 7.2.12 **bpstat**

#### 7.2.12.1 Name

**bpstat** – Show cluster node status and cluster process mapping.

#### 7.2.12.2 Synopsis

7.2.12.3 Description

This utility displays the BProc status of cluster nodes, and processes running on those nodes. Node information includes the node’s IP address, state, user ownership, group ownership, and running node user processes.

7.2.12.4 Options

The following options are available to the bpstat program.

- `-a, --address` Prints the IP address of the indicated node.
- `-A hostname` Prints the node number that corresponds to the specified hostname or IP address.
- `-c, --compact` Print compacted listing of nodes (default).
- `-h, --help` Print the command usage message and exit. If `-h` is the first option, all other options will be ignored. If `-h` is not the first option, the other options will be parsed up to the `-h` option, and those options will be processed.
- `-l, --long` Print long list of node status. This includes IP address, status, mode, user and group information.
- `-M` Prints the status of the master node, in addition to the specified compute node(s), for the default case where no specific nodes are specified.
- `-n, --number` Prints the node numbers that are being used and/or are available for the nodes in the cluster.
- `-N, --sort-number` Prints the node list sorted by node number.
- `-O, --keep-order` Prints the nodes in the order returned by the system (no sorting is done).
- `-p` Prints a list of processes (by PID) that are currently running on the specified node.
- `-P nodes` Postprocesses the output from the ps command, prepending the node number that BProc-controlled processes are running on. This is typically used as ps aux | bpstat -P. Processes not controlled by the BProc system will not have a number appended. If the optional `[nodes]` is supplied, then the ps output is filtered to show only the specified node(s). Node(s) can be identified by names, numbers, or a list of numbers.
- `-R, --sort-reverse` Prints the node list in reverse sorted order.
- `-s, --status` Prints the state for the indicated node. The BProc states are “down”, “boot”, “error”, “unavailable”, “up”, “reboot”, “halt”, and “pwoff”.
- `-S, --sort-status` Prints the node list sorted by node status.
- `-t, --total` Prints the total number of compute nodes configured for the cluster. The number is calculated from the cluster configuration in the `/etc/beowulf/config` file. Note that this is the potential maximum size of the cluster, not the current number of available nodes or the count of machines assigned node numbers.
- `-U, --update` Continuously update the status; otherwise, print status once and exit.
- `-V, --version` Print the command version number and exit. If `-V` is the first option, all other options will be ignored. If `-V` is not the first option, the other options will be parsed up to the `-V` option, and those options will be processed.

[nodes | allstate] Optionally, specify the nodes for which information is to be displayed. Nodes can be specified individually, as ranges, or in a comma-separated list of individual nodes and/or ranges. Alternatively, allstate
specifies all nodes that are in a particular state, e.g., allup, alldown, allboot, allerror. Note: allup does not include the master node, even if -M is present.

### 7.2.12.5 Examples

Print the number of available nodes:

```
[user@cluster user] $ bpstat --total allup
9
```

Generate a list of all usable nodes:

```
[user@cluster user] $ bpstat --number allup
 0 1 2 4 5 10 16 17 20
[user@cluster user] $ bpstat --number allup | awk '{ print "."$1 }'
 0 .1 .2 .4 .5 .10 .16 .17 .20
```

Print status for all nodes, including the master node:

```
[user@cluster user] $ bpstat

<table>
<thead>
<tr>
<th>Node(s)</th>
<th>Status</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,11,22-31</td>
<td>down</td>
<td>----------</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>3,6-7,9,12-15,18-19,21</td>
<td>error</td>
<td>---x------</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>-1,0-2,4-5,10,16-17,20</td>
<td>up</td>
<td>---x-x-x-x</td>
<td>root</td>
<td>root</td>
</tr>
</tbody>
</table>
```

Print the PIDs and associated node number of currently running processes:

```
[user@cluster user] $ bpstat -p

<table>
<thead>
<tr>
<th>PID</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>7503</td>
<td>0</td>
</tr>
<tr>
<td>8262</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Print status for specific nodes:

```
[user@cluster user] $ bpstat 0-2,3,8

<table>
<thead>
<tr>
<th>Node(s)</th>
<th>Status</th>
<th>Mode</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>down</td>
<td>------</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>3</td>
<td>error</td>
<td>---x--</td>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>0-2</td>
<td>up</td>
<td>---x--x x</td>
<td>root</td>
<td>root</td>
</tr>
</tbody>
</table>
```

Augment `ps aux` for node numbers:

```
[user@cluster user] $ ps aux | bpstat -P

<table>
<thead>
<tr>
<th>NODE USER</th>
<th>PID %CPU %MEM VSZ RSS TTY</th>
<th>STAT START</th>
<th>TIME COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>1 0.0 0.0 4756 552 ?</td>
<td>S 10:58</td>
<td>0:02 init [5]</td>
</tr>
<tr>
<td>root</td>
<td>2 0.0 0.0 0 0 ?</td>
<td>S 10:58</td>
<td>0:00 [migration/0]</td>
</tr>
</tbody>
</table>
```

Filter `ps aux` for nodes n1 and n2:

```
[user@cluster user] $ ps aux | bpstat -P n1,n2

<table>
<thead>
<tr>
<th>NODE USER</th>
<th>PID %CPU %MEM VSZ RSS TTY</th>
<th>STAT START</th>
<th>TIME COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 root</td>
<td>1328 0.0 0.0 6864 692 ?</td>
<td>Ss 12:45</td>
<td>0:00 [portmap]</td>
</tr>
<tr>
<td>2 root</td>
<td>32397 0.0 0.0 6864 692 ?</td>
<td>Ss 12:45</td>
<td>0:00 [portmap]</td>
</tr>
</tbody>
</table>
```

### 7.2. Scyld ClusterWare Commands
7.2.13 mpprun

7.2.13.1 Name

mpprun – Run a series of commands on a Scyld cluster using a dynamically generated job map.

7.2.13.2 Synopsis


7.2.13.3 Description

The mpprun program sequentially runs the specified program on a dynamically selected set of cluster nodes. It generates a job map from the currently installed beomap scheduler, and runs the program on each node specified in the map. The scheduling parameters from the command line and environment are the same as for beomap, and the resulting job map is identical to the job map that beomap would if generate at that instant in time for that program name.

mpprun is similar to the beorun program, but beorun starts the job simultaneously on the cluster nodes, whereas mpprun starts the job sequentially.

7.2.13.4 Options

The following general command line options are available to mpprun. Also see the next section, which describes the beomap job map parameters.

- h, --help, -u, --usage Print the command usage message on stdout and exit. When one of these options is recognized in the option list, all following options will be ignored.

-V Print the command version number on stdout and exit. Any following options will be ignored.

-p, --prefix Prefix each line of output the node number.

You can influence the mpprun job map either by entering command line options or by setting environment variables. Following are the available command line options, together with their equivalent environment variables. Note that the command line options take precedence over the environment variables.

All the mpprun job map parameters listed below can also be used directly with beorun and beomap.

--all-cpus Create a process map consisting of all “up” nodes, with each node number repeated to represent the number of CPUs on that node. This parameter is not allowed in conjunction with the --map parameter.

--all-nodes Create a process map consisting of all “up” nodes, with one CPU mapped on each of the “up” nodes. This parameter is not allowed in conjunction with the --map parameter.

The equivalent environment variable is ALL_NODES.

--all-local Create a process map consisting entirely of master node entries. This option eliminates everything except node -1 from the pool of candidate node numbers, thus forcing the map to use node -1 (the master node) for everything.

The equivalent environment variable is ALL_LOCAL.
**--no-local**

Exclude the master in the process map. This option is essentially a syntactic shortcut for including `--exclude nodelist` option. For MPI jobs, this option puts the “rank 0” job on a compute node instead of on the master node. This parameter is not allowed in conjunction with the `--map` parameter.

The equivalent environment variable is `NO_LOCAL`.

**--exclude nodelist**

Build a process map that excludes listed nodes. The `nodelist` consists of a colon-delimited list. This parameter is not allowed in conjunction with the `--map` parameter.

The equivalent environment variable is `EXCLUDE=nodelist`.

**--map nodelist**

Explicitly specify a process map consisting of a colon-delimited list of nodes. Each node in `nodelist` indicates where one process will be assigned. The number of entries in the job map implies the number of ranks in the job.

Listing a node more than once in the list will assign multiple processes to that node. Typically, this is done to assign one process to each processor (or core) on a node, but this can also be used to “oversubscribe”, i.e., to assign more processes to a node than it has processors (or cores).

The equivalent environment variable is `BEOWULF_JOB_MAP=nodelist`.

**--np num-processes**

Specify the number of processes to run. The `mpprun` command attempts to place one process per processor (or core), but will “oversubscribe” and assign multiple processes per processor (or core) if there are not enough individual processors or cores available. This parameter is not allowed in conjunction with the `--map` parameter.

The equivalent environment variable is `NP=num-processes`.

The environment variables have an order of priority. The `BEOWULF_JOB_MAP` variable acts as a “master override” for the other environment variables. If `BEOWULF_JOB_MAP` is not set, then the following priorities apply:

Three of the environment variables determine how many ranks to schedule in the map: (1) `ALL_CPUS`, (2) `ALL_NODES`, and (3) `NP`. If none of these are set explicitly by the user, then `NP=1` is the default.

Three of the environment variables determine what node numbers are candidates for being mapped: (1) `ALL_LOCAL`, (2) `NO_LOCAL`, and (3) `EXCLUDE`.

Note: it is improper to use `NO_LOCAL` and `ALL_LOCAL` together. If both are used, then `ALL_LOCAL` takes precedence.

Unrecognized options and invalid option formats are reported on `stderr` and the command exits with exit status 1 (invalid option) or 2 (no command specified or invalid command).

### 7.2.13.5 Examples

Run `uptime` on any two available cluster compute nodes.

```
[user@cluster user]$ mpprun --np 2 --no-local uptime
11:05am up 2 days, 11:16, 0 users, load average: 0.05, 0.24, 0.65
11:05am up 2 days, 11:16, 0 users, load average: 0.01, 0.07, 0.37
```
7.3 Scyld ClusterWare Maintenance Commands

This section of the Reference Guide describes the Scyld ClusterWare maintenance utilities. These commands can be used by the cluster administrator, and are not intended for use by the ordinary user.

7.3.1 beofdisk

7.3.1.1 Name

beofdisk – Query and modify hard drive partitions on compute nodes.

7.3.1.2 Synopsis


7.3.1.3 Description

This script allows you to partition the hard drives on compute nodes.

When you query, it will create files in /etc/beowulf/fdisk/, one for each device/drive geometry it finds. These files can then be modified by hand, or with the defaults options, then written back to the hard drives.

7.3.1.4 Options

- **-h, --help** Display a help message and exit.
- **-v, --version** Display version information and exit.
- **-q, --query** Queries the hard drives and writes their current partition tables into files in /etc/beowulf/fdisk/. If no -n num node is specified, then all nodes are queried.
- **-w, --write** Matches the files in /etc/beowulf/fdisk/ with the hard drives and changes the partition tables on the compute nodes to match what is in the files. If no -n num node is specified, then all nodes are written.
  
  WARNING: This option is potentially dangerous. It modifies partition tables, and incorrect partition tables can cause problems.

- **-d, --default** This will cause beofdisk to go through the files in /etc/beowulf/fdisk/ and set them all to contain default partitioning schemes that include a beoboot partition, a swap partition, and the rest as /.

- **-M, --MBR** Write a simple Master Boot Record to the hard drive that directs the BIOS to “boot next device” after each failure to boot. Typically, this ultimately results in a PXE boot. If no -n num node is specified, then all nodes are written with this new MBR.

- **-n num, --node num** By default, the apply the specified operation to all nodes. Optionally, apply the operation only to node num.
7.3.1.5 Examples

Creating default partition schemes:

```
[root@cluster ~] # beofdisk -d
Creating a default partition table for hda:2495:255:63
Creating a default partition table for hda:1222:255:63
```

Writing the defaults to node 0's hard drive:

```
[root@cluster ~] # beofdisk -w -n 0
Disk /dev/hda: 2495 cylinders, 255 heads, 63 sectors/track
Old situation:
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>#cyls</th>
<th>#blocks</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/hda1</td>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>1-</td>
<td>8001</td>
<td>89</td>
<td>Unknown</td>
</tr>
<tr>
<td>/dev/hda2</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>257040</td>
<td>82</td>
<td>Linux swap</td>
<td></td>
</tr>
<tr>
<td>/dev/hda3</td>
<td>33</td>
<td>2494</td>
<td>2462</td>
<td>19776015</td>
<td>83</td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>/dev/hda4</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Empty</td>
</tr>
</tbody>
</table>

New situation:
Units = sectors of 512 bytes, counting from 0

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>#sectors</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/hda1</td>
<td>*</td>
<td>63</td>
<td>16064</td>
<td>16002</td>
<td>89</td>
<td>Unknown</td>
</tr>
<tr>
<td>/dev/hda2</td>
<td>16065</td>
<td>546209</td>
<td>530145</td>
<td>82</td>
<td>Linux swap</td>
<td></td>
</tr>
<tr>
<td>/dev/hda3</td>
<td>546210</td>
<td>40082174</td>
<td>39535965</td>
<td>83</td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>/dev/hda4</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Empty</td>
</tr>
</tbody>
</table>
```

Successfully wrote the new partition table

Re-reading the partition table ...

If you created or changed a DOS partition, /dev/foo7, say, then use dd(1) to zero the first 512 bytes: dd if=/dev/zero of=/dev/foo7 bs=512 count=1 (See fdisk(8)).

Node partition tables have been modified.
You must reboot each affected node for changes to take effect.

Query the disks on the compute nodes to determine how they are partitioned:

```
[root@cluster ~] # beofdisk -q
```

The following creates a partition file in /etc/beowulf/fdisk, with a name similar to sda:512:128:32 and containing lines similar to the following:

```
[root@cluster ~] # cat sda:512:128:32
/dev/sda1 : start= 32, size= 8160, id=89, bootatable
/dev/sda2 : start= 8192, size= 1048576, Id=82
/dev/sda3 : start= 1056768, size= 1040384, Id=83
/dev/sda4 : start= 0, size= 0, Id=0
```
7.3.2 beorsync

7.3.2.1 Name

beorsync – Sync files between two servers in an HA configuration.

7.3.2.2 Synopsis

beorsync syncfiles

7.3.2.3 Description

beorsync is a perl script used to synchronize individual files and the contents of entire directories between master nodes in a High Availability master node failover environment.

The script has one required argument: syncfiles, which is the name of a file containing a list of files and directories to be synchronized.

beorsync expects to execute on the passive master node of a passive-active pair. Both the source and target nodes must be running heartbeat. The script pulls only those files that have changed on the active master node.

Diagnostic messages are logged to /var/log/beorsync.log.

7.3.2.4 Errors

If beorsync is invoked on the active master node, then the script exits with an error message.

7.3.2.5 Examples

A typical syncfiles contains the following list of files and directories to be synchronized:

<table>
<thead>
<tr>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/hosts</td>
</tr>
<tr>
<td>/etc/resolv.conf</td>
</tr>
<tr>
<td>/etc/ntp.conf</td>
</tr>
<tr>
<td>/root/bin/</td>
</tr>
<tr>
<td>/var/spool/cron/</td>
</tr>
<tr>
<td>/etc/beowulf/</td>
</tr>
<tr>
<td>/etc/passwd</td>
</tr>
<tr>
<td>/etc/shadow</td>
</tr>
<tr>
<td>/etc/group</td>
</tr>
<tr>
<td>/etc/nsswitch.conf</td>
</tr>
<tr>
<td>/etc/exports</td>
</tr>
<tr>
<td>/etc/services</td>
</tr>
<tr>
<td>/etc/ha.d/haresources</td>
</tr>
<tr>
<td>/var/spool/torque/mom_priv/config</td>
</tr>
<tr>
<td>/var/spool/torque/server_priv/jobs/</td>
</tr>
<tr>
<td>/var/spool/torque/server_priv/serverdb</td>
</tr>
</tbody>
</table>

Commonly, a cron job should be set up that periodically executes the beorsync script. For example, the following cron entry executes the script every 5 minutes, syncing all of the files and directories listed in the syncfiles file named /etc/beowulf/beorsyncfiles:
7.3.3 beoserv

7.3.3.1 Name

beoserv – The daemon that serves IP addresses and boot files to compute nodes

7.3.3.2 Synopsis


7.3.3.3 Description

The beoserv daemon is started by the ClusterWare service and responds to DHCP, PXEboot, TFTP, and TCP get-file requests from compute node clients. The daemon reports cluster events to /var/log/messages.

7.3.3.4 Options

- **-h, --help**  Display a help message and exit.
- **-V, --version**  Display version information and exit.
- **-v**  Increase verbosity level. Each additional `v` increases verbosity.
- **-f file**  Read configuration from file instead of /etc/beowulf/config.
- **-n file**  Write unrecognized nodes to file instead of /var/beowulf/unknown_addresses.

7.3.3.5 Examples

Start daemon using file myconfig.

```
[root@cluster ~] # /usr/sbin/beoserv -f /etc/beowulf/myconfig
```

7.3.4 bpctl

7.3.4.1 Name

bpctl – Control the operational state and ownership of compute nodes.
### 7.3.4.2 Synopsis

```
```

### 7.3.4.3 Description

This utility is part of the BProc package and is installed by default. It allows the root user to modify the state of the compute nodes. Compute nodes may be in one of eight states: **down**, **boot**, **up**, **error**, **unavailable**, **reboot**, **halt**, **poweroff**. The states are described as follows:

- **down**  No communication with compute node, and prior node state is unknown.
- **boot**  Node has initialized communication and started but not completed the node_up script. This state is not commandable. It is status information only.
- **up**  Node is communicating and has completed the node_up script without errors.
- **error**  Node is communicating and encountered an error while running the node_up script.
- **unavailable**  Node is communicating and the cluster administrator has marked the node as unavailable to non-root users.
- **reboot**  Node will do a software reboot. Node status will show reboot through start of machine shutdown until node_up script has begun.
- **halt**  Node has been commanded to halt. This command causes the node CPUs to execute the halt machine instruction. Once halted the node must be reset by external means to resume normal operations.
- **poweroff, pwroff**  Node will power off. This command is valid for nodes that meet the ATX specification. This command requires BIOS support. Non-ATX machines may reboot on this command.

Normally the node will transition from **down** to **boot** to **up**, and will remain **up** until commanded otherwise. **up** is the operational state for user programs. User BProc commands will be rejected if the node is not **up**.

BProc supports a simplified user and group compute node access scheme. Before any action is taken on a node, BProc checks if the user or group match. If either is matched the user action is processed. Note, normal file permissions are still in effect on each node. BProc permissions simply allow users to execute a program on a node. Root bypasses the check and always has access.

User and group changes made with `bpctl` remain in effect until the affected node(s) are restarted. After a restart, the user and group information is read from the `/etc/beowulf/config` file. For persistent changes, you must edit the config file. Changes to the config file take effect when you issue a ClusterWare service `reload` or you reboot the nodes via a `restart`. With `reload`, running jobs will not be affected unless they start a new process and are denied node access based on the permission changes.

Whenever the ClusterWare daemons are restarted, all nodes are initialized to the **down** state and node history is lost. When this occurs, previously communicating nodes will reboot and attempt to re-establish communication after the “ping timeout”, which by default is 30 seconds.
7.3.4.4 Options

The following options are available to `bpctl`:

- **-h**
  Print the command usage message and exit. If -h is the first option, all other options will be ignored. If -h is not the first option, the other options will be parsed up to the -h option, but no action will be taken.

- **-v**
  Print the command version number and exit. If -v is the first option, all other options will be ignored. If -v is not the first option, the other options will be parsed up to the -v option, but no action will be taken.

- **-f**
  Fast mode. Whenever possible, do not wait for acknowledgment from compute nodes.

- **-M**
  Specifies that the remaining options apply to the master node.

- **-S num**
  Specifies that the remaining options apply to the specified compute node. The num can range from 0 to the total number of nodes minus one.

- **-s state**
  Set the node to the indicated state. Valid state values are `down`, `up`, `error`, `unavailable`, `reboot`, `halt`, or `pwroff`. Setting state to `down` causes the node to reboot due to a communications timeout after the “ping timeout” interval, which by default is 30 seconds.

- **-m mode**
  Set the permission bits for the indicated node. Only the Execute mode bits are recognized, i.e., a logical or’ing of octal values 001, 010, and/or 100.

- **-u user**
  Set the user id for the indicated node. Will reject invalid users. Numbers or strings may be used. A numeric user id will be converted to a name if the name is known.

- **-g group**
  Set the group id for the indicated node. Will reject invalid groups. Numbers or strings may be used. A numeric group id will be converted to a name if the name is known.

- **-H, --halt**
  Halt the indicated node.

- **-P, --poweroff, --pwroff**
  Power off the indicated node.

- **-R, --reboot**
  Reboot the indicated node.

- **-O, --orphan**
  Direct the indicated node to become an immediate orphan.

- **-C r2c-state, --completion r2c-state**
  Turn run-to-completion mode on or off for the nodes specified by -S num. Acceptable r2c-state values are `on` (an “orphaned” node stays up indefinitely, until manually rebooted), `off` (an “orphaned” node reboots immediately), or a positive number of seconds of “effectively idle” time that an orphaned node will wait until rebooting.

- **-I idle-threshold, --idle idle-threshold**
  Override the default cpu usage percentage threshold that an “orphaned” compute node uses to determine whether or not the node is “effectively idle”.

  When a compute node becomes an “orphan” and the r2c-state specifies that the node reboot after the specified number of “effectively idle” seconds, BProc periodically determines how much cpu usage has occurred during the preceding interval (which is nominally 10 seconds). If the cpu usage is above the idle-threshold percentage, then the time-until-reboot is reset back to r2c-state seconds. The idle-threshold value must be a positive numeric value, and it may be an integer or a floating-point number. A too-low value means BProc will mistakenly interpret trivial amounts of cpu usage (e.g., executed by daemons that wake up and check for work) as being significant, and thus the node may never reboot. A too-high value

---

7.3. Scyld ClusterWare Maintenance Commands
means BProc will mistakenly interpret significant cpu usage as being insignificant, and thus the node may reboot while a low-usage process is doing important work.

### 7.3.4.5 Examples

This command will cause all nodes to reboot:

```
[root@cluster ~] # bpctl -S all -s reboot
```

This command returns an error, because boot is not commandable:

```
[root@cluster ~] # bpctl -S 4 -s boot
Non-commandable node state: boot
```

The following sets nodes 3 and 4 ownership to user “foo”, which must be a valid user:

```
[root@cluster ~] # bpctl -S 3-4 -u foo
```

The following sets permission on the master node to allow only user root to execute on the master node, e.g., to disallow a non-root user to execute on a compute node and **bpsh** a command to execute on the master:

```
[root@cluster ~] # bpctl -M -m 100
```

And this resets permission on the master node to allow any user to execute on the master node:

```
[root@cluster ~] # bpctl -M -m 111
```

This command resets the run-to-completion timeout to five minutes, and sets the “effectively idle” cpu usage percent to 1.5%:

```
[root@cluster ~] # bpctl -C 300 -I 1.5
```

### 7.3.4.6 Return Values

Upon successful completion, **bpctl** returns 0. On failure, an error message is printed to **stderr** and **bpctl** returns 1.

### 7.3.5 bplib

#### 7.3.5.1 Name

**bplib** – manages the VMAdump in-kernel library list and individual file list of cached files.
7.3.5.2 Synopsis


7.3.5.3 Description

This utility is part of the BProc package and is installed by default. It is used to modify entries of the in-kernel cache list.

7.3.5.4 Options

The following options are available to the `bplib` program.

- `-h` Print the command usage message and exits success.
- `-v` Print the command version number and exits success.
- `-c` Clears ALL cached entries in the in-kernel cache list.
- `-a libs` Adds the specified file or directory to the in-kernel cache list.
- `-d lib` Deletes the specified file or directory from the in-kernel cache list.
- `-l` Lists all entries known by the in-kernel cache list.

7.3.6 bpmaster

7.3.6.1 Name

`bpmaster` – Daemon for cluster control and communication.

7.3.6.2 Synopsis


7.3.6.3 Description

This daemon is part of the BProc package and is installed by default. It is the controller and message/IO manager for all the compute nodes and must be running for the cluster to function.

`bpmaster` is started by the ClusterWare initialization script, along with other BProc daemons, and forks a copy of itself for IO forwarding. With normal cluster operation there should be 2 PIDs for `bpmaster`. Type `ps -x | grep bpmaster` to check.

The `bpmaster` daemon may be restarted at any time by using a ClusterWare service `restart`, but note that this will cause all nodes to reboot. During normal operations, use a service `reload` to enable `/etc/beowulf/config` configuration changes. The daemon reports cluster events to `/var/log/messages`.

7.3. Scyld ClusterWare Maintenance Commands
7.3.6.4 Options

The following options are available to the `bpmaster` program.

- **-h**
  Print the command usage message and exit. If `-h` is the first option, all other options will be ignored. If `-h` is not the first option, the other options will be parsed up to the `-h` option, but no action will be taken.

- **-V**
  Print the command version number and exit. If `-v` is the first option, all other options will be ignored. If `-v` is not the first option, the other options will be parsed up to the `-v` option, but no action will be taken.

- **-d**
  Start the program in debug (verbose) mode. `bpmaster` will not daemonize, and all information and error messages will go to `stdout`. This information is useful when the daemon exits abnormally during operation as the information is not mixed in with the normal `/var/log/messages`.

- **-v**
  Increase verbosity level. This may be specified multiple times.

- **-i**
  Ignore interface version mismatch. This can be dangerous.

- **-c file**
  Specifies a different configuration file is to be used. The default is set to `/etc/beowulf/config`. This option is for debug and development. This option is not recommended for normal use.

- **-m file**
  Log master and node BProc messages to the indicated file. This information is intended for BProc debugging, and should not be enabled unless requested by a Scyld support engineer. This file grows in size rapidly depending of the number of nodes, approximately 2 megabytes/minute with six nodes.

7.3.6.5 Examples

Don’t start as daemon.

```
[root@cluster ~] # bpmaster -d
```

Start the daemon using the startup script.

```
[root@cluster ~] # service beowulf start
 Configuring network interface (eth1): [ OK ]
 Loading modules: [ OK ]
 Setting up libraries: [ OK ]
 Starting bpmaster:
 Starting beoserv:
 Starting recvstats:
 Starting sendstats:
```
7.3.7 bpslave

7.3.7.1 Name

bpslave – This program is the BProc distributed process space slave daemon that executes on each compute node.

7.3.7.2 Synopsis


7.3.7.3 Description

The bpslave daemon is part of the BProc package, and is installed by default. It is the controller and message and I/O manager run on each compute node, and must be running for the node to function.

bpslave is started by the Scyld compute node init process, which sets parameters based on what is passed in through the kernel command line option in the /etc/beowulf/config file. All parameters of the bpslave daemon are not accessible via the “kernelcommandline” keyword in /etc/beowulf/config.

The bpslave daemon is not intended to be run from the command line, nor otherwise executed started, except implicitly by the compute node init process.

7.3.7.4 Options

The following options are available to the bpslave program. These options are mainly intended for using BProc in a standard linux environment where the master and compute nodes both have full system installs.

- **-h** Show this message and exit. If -h is the first option, all other options will be ignored. If -h is not the first option, the other options will be parsed up to the -h option, but no action will be taken.

- **-V** Print version information and exit.

- **-l logfacility** Log to this log facility (default=daemon).

- **-r** Automatic reconnect on error or lost connection.

- **-i** Ignore BProc version mismatches (dangerous).

- **-d** Do not daemonize self.

- **-s addr** Connect from source address addr.

- **-c dir** Set library cache to dir.

- **-p port** Set library cache file request port to port. The default is port 932, which can be overridden by a config file directive server beofs2.

**Debugging options:**

- **-m file** Enable message trace to file.

- **-v** Increase verbose level (implies -d).

**Masterhostname [[port]]** The host name and (optionally) the port number of the bpmaster daemon. The default is port 933, which can be overridden by a config file directive server bproc.
7.3.8 node_down

7.3.8.1 Name

node_down – Bring a compute node down cleanly.

7.3.8.2 Synopsis

/usr/lib/beoboot/bin/node_down node [state]

7.3.8.3 Description

This script can be used to bring a node down (“reboot”, “halt”, “pwroff”) in such a way that the local filesystems on the compute node remain in a constant state.

node_down works by first changing the node’s state to “unavailable”, then remounting all of the filesystems read-only, followed by using bpcctl to perform the actual state change you requested.

7.3.8.4 Options

node  The node number of the node to bring down.
state  The state to put the node in after remounting all the filesystems. The state defaults to “reboot” if unspecified.

7.3.8.5 Examples

Cleanly bringing down node 3:

```
[root@cluster ~] # /usr/lib/beoboot/bin/node_down 3
Remounting / readonly...
Remounting /proc readonly...
Remounting /home readonly...
Remounting /dev/pts readonly...
Syncing disks on node 3.
Shutting down node 3 (reboot)
```

7.3.9 recvstats

7.3.9.1 Name

recvstats – master node daemon that receives status messages from compute nodes
7.3.9.2 Synopsis

recvstats [-p port] [-N initial-max-num-nodes] [-f]

7.3.9.3 Description

The recvstats daemon is part of the beostat package. It executes on the master node, receives periodic per-node status data sent by each cluster node’s sendstats daemon, and makes the data available to various commands and services on the master node.

The recvstats daemon parses the received data to ensure basic validity. The exact content and format of the sendstats messages is version specific, though it typically includes a unique node number identifying the sender, plus the dynamic values of the following proc file system files: /proc/cpuinfo, /proc/meminfo, /proc/loadavg, /proc/net/dev, and /proc/stat.

The recvstats daemon stores the incoming data in shared memory in the /dev/shm filesystem, which should be readable by everyone. If that filesystem doesn’t exist or the file permissions are not set correctly, then recvstats and the consumers of that data will not function correctly. Note: most consumers of this recvstats data access it using various commands (e.g., beostat(1), beostatus(1), ganglia, Scyld IMF) or use the libbeostat abstracted library interface.

The recvstats daemon is started by the ClusterWare service, and the sendstats daemon is started for BProc nodes by the node initialization script /etc/beowulf/init.d/13sendstats.

7.3.9.4 Options

The following options are available to the recvstats daemon.

- **-p port** Override the default listen port of 5545. Only use this option if there is a conflict with the default port, and use the same non-default port when executing sendstats on each cluster node.

- **-N initial-max-num-nodes** Start recvstats with an explicit non-default guess about how many cluster nodes will be sending data. If a node number above this initial maximum sends data, then recvstats dynamically expands the shared memory structure to accommodate it.

7.3.9.5 Examples

Start the recvstats daemon on the master node:

```
[root@cluster ~] # /usr/bin/recvstats -p 5545 -N `beoconfig nodes`
```

7.3.10 sendstats

7.3.10.1 Name

sendstats – compute node daemon that sends node status to the master node
7.3.10.2 Synopsis

sendstats [-h] [nodenumber] [IPaddress[:port] ...]

7.3.10.3 Description

The sendstats daemon is part of the beostat package. Typically, the daemon executes on each node in the cluster and periodically transmits status data to a recvstats daemon that executes on the master node. In a cluster with multiple master nodes, sendstats typically sends status data to every master node.

The optional nodenumber is unnecessary for normal uses of sendstats. The recvstats daemon is normally able to discern the sender’s node number from the sender’s IP address. If nodenumber is specified, then it must be seen by the receiving recvstats as being unique to one and only one sending node in the cluster.

The exact content and format of the sendstats messages is version specific, though it typically includes a unique identifying nodenumber plus the dynamic values of the following proc file system files: /proc/cpuinfo, /proc/meminfo, /proc/loadavg, /proc/net/dev, and /proc/stat.

The port number is optional, defaulting to port 5545. In the event of a collision with another preexisting service, which would typically be defined in /etc/services, you must override the default. Choose a new value that is not currently employed on the system, then add a server beostats `directive to `/etc/beowulf/config`.

The recvstats daemon is started by the ClusterWare service, and the sendstats daemon is started for BProc nodes by the node initialization script `/etc/beowulf/init.d/13sendstats`.

The sendstats daemon may be used on machines outside of the BProc cluster management domain. In any case, the port number must match the port on which recvstats listens.

7.3.10.4 Examples

Start the daemon on ClusterWare node n0, sending stats to the master at 10.20.30.1 using the default port:

```
[root@cluster ~] # bpsh 0 /usr/sbin/sendstats 10.20.30.1
```

Start the daemon on ClusterWare node n0, sending stats to the master at 10.20.30.1 and a second master at 10.20.30.2, using a non-default port:

```
[root@c ~] # bpsh 0 /usr/sbin/sendstats 10.20.30.1:939 10.20.30.2:939
```

Start the daemon on a non-ClusterWare node n1, using the default port:

```
[root@c ~] # ssh n1 /usr/sbin/sendstats 10.20.30.1:5545
```

7.4 Scyld ClusterWare Special Files

This section of the Reference Guide describes `/etc/beowulf/config` and `/etc/beowulf/fstab`, the configuration files that are used by the Scyld ClusterWare system.
7.4.1 beowulf-config

7.4.1.1 Name

/etc/beowulf/config – Scyld ClusterWare Configuration file

7.4.1.2 Description

The Beowulf config file /etc/beowulf/config defines the structure of a Scyld ClusterWare cluster and provides a central location for many of the operational parameters. The file contains the settings for beoboot, node initialization, BProc communication parameters, and other aspects of cluster operation.

The syntax of the ClusterWare configuration files is standardized and is intended for human editing with embedded comments. Tools are provided for reading and writing from common programming and scripting languages, with writing retaining comments and formatting.

Tip
Care must be taken when editing or otherwise modifying /etc/beowulf/config, e.g., avoid editing while new compute nodes are coming online and ClusterWare itself is adding or modifying ‘node’ lines. Also note that incorrect editing may leave the cluster unuseable.

7.4.1.3 Config File Format

The config file is a line-oriented sequence of configuration entries. Each configuration entry starts with a keyword followed by parameters. A line is terminated by a newline or ‘#’. The latter character starts a comment.

The keyword and following parameters have the same syntax rules: they may be preceded by whitespace and continue to the next whitespace or the end of the line.

Keywords and following parameters may include whitespace by quoting between a matching pair of ““ (double quote) or ‘‘’ (single quote) characters. A ‘\’ (backslash) removes the special meaning of the following quote character.

Note that comments and newlines take precedence over any other processing, thus a ‘#’ may not be used in a keyword or embedded in a parameter, and a backslash followed by a newline does not join lines.

Each configuration option is contained on a single line, with a keyword and optional parameters. Blank lines are ignored. Comments begin with an unquoted ‘#’ and continue to the end of the line.

7.4.1.4 Keywords

bootmodule modulename The bootmodule keyword specifies that the kernel binary module modulename be included in the compute nodes’ initrd image. These are typically network drivers needed to fully initialize a booting node. At node startup, the beoclient daemon on a compute node scans the node’s /proc/bus/pci/devices list and automatically executes a modprobe for every modulename driver named by a PCI device so discovered. However, note that if the PCI scan does not find a need for a particular driver, then no automatic modprobe occurs. Add an additional modprobe keyword to forcibly load the modulename.

firmware firmfile The firmware keyword specifies that the firmfile file, which typically resides on the master node in /lib/firmware/firmfile, be included in the compute nodes’ initrd image, if known to be needed by a particular bootmodule modulename. Adding one or more firmware keywords significantly increases the size of the initrd image. See the Administrator’s Guide for details.

fsck fsck-policy The fsck keyword specifies the file system checking policy to be used at node boot time. The valid policies are “never”, “safe” or “full”.

never The file system on the compute nodes will not be checked on boot.
safe  The file system on the compute nodes will go through a safe check every time the compute node boots.

full  The file system on the compute nodes will go through a full check every time the compute node boots. The full check might possibly remove files from the filesystem if they cannot be repaired.

host MACaddress IPaddress [hostname(s)] The host keyword assigns an IP addresses to a specific client device identified by its MAC address, if and when that client makes a DHCP request to the master. The IP addresses must be in dotted notation (e.g., 192.168.1.100), and it must be within the range of one of the hostrange IP address ranges. These host clients are not Scyld nodes, which are identified by node keywords and are assigned IP addresses from the iprange range. Rather, typically they are devices like smart Ethernet switches that connect to the cluster private network and issue a DHCP request to obtain an IP address. Up to six optional hostname names may be assigned to a client, and these names are recognized by the Beo NSS service.

hostrange [name] IPaddress-lwb IPaddress-upb The hostrange is used in conjunction with the host keyword. It declares a range of IP addresses that may later be used for host clients doing DHCP requests. An optional name may be associated with this range. Multiple hostrange keywords may be present.

ignore MACaddress The ignore keyword specifies a MAC address (e.g., 00:11:22:AA:BB:CC) that beoserv should ignore DHCP and PXE requests from. Multiple ignore keywords are allowed.

initrdimage [noderange] imagename The initrdimage keyword specifies the full path to the initrd image that should be used when creating the final boot images for the compute nodes. If noderange is specified, then this imagename applies only to the specified range of nodes; otherwise, imagename applies to all nodes.

insmod module-name [options] The insmod keyword specifies a kernel module to be loaded (usually a network driver). Options for the module may be specified as well.

interface interfacename The interface keyword specifies the name of the interface that connects the master node to the compute nodes. This is used by the cluster services and management tools such as the bpmaster daemon and the beoserv daemon. Common values are “eth0” or “eth1”. If present, entries after the interface name specify the IP address and netmask that the interface should be configured to.

iprange [nodenumber] IPaddress1 IPaddress2 The iprange keyword specifies the range of IP addresses to be assigned to nodes. If the optional nodenumber is given, the first address in the range will be assigned to that node, the second address to the next node, etc. If no node number is given, the address assignment will begin with the node following the node that was last assigned. If no nodes have been assigned, the assignment will begin with node 0.

kernelcommandline [noderange] options The kernelcommandline keyword specifies any options you wish to have passed to the kernel on the compute nodes. These are the same options that are normally passed with “append=” in lilo, or on the lilo prompt while the machine is booting (e.g., “kernelcommandline apm=power-off”). If noderange is specified, then these options apply only to the specified range of nodes; otherwise, options apply to all nodes.

kernelimage [noderange] imagename The kernelimage keyword specifies the full path to the kernel that should be used when creating the final boot images for the compute nodes. If noderange is specified, then this imagename applies only to the specified range of nodes; otherwise, imagename applies to all nodes.

libraries librarypath1 [, librarypath2, ...] The libraries keyword specifies a list of libraries that should be cached on the compute nodes when an application on the node references the library. The library path can be a directory or file. If a file name is specified, then that specific file may be cached, if needed. If a directory name is specified, then every file in that directory may be cached. If the directory name ends with “/”, then subdirectories under the specified directory may be cached.

logfacility facility The logfacility keyword specifies the log facility that the BProc master daemon should use. Some example log facility names are “daemon”, “syslog”, and “local0” (see the syslog documentation for more information). The default log facility is “daemon”.

masterdelay SECS The masterdelay keyword specifies the timeout value in seconds for a non-primary master node to delay sending a response to an incoming dhcp request. The default value is 15 seconds.
**masterorder nodes IPaddress_primary IPaddress_secondary**  The `masterorder` keyword specifies the cluster IP addresses of the primary master node and the secondary master node(s) for a given set of nodes. This is used by the `beoserv` daemon for Master-Failover (cold reparenting). A compute node’s PXE request broadcasts across the cluster network. The primary master node is given `masterpxedelay` seconds to respond, after which the first secondary master node will respond. If multiple secondary master nodes are specified, then each waits in turn for `masterpxedelay` seconds for a preferred master to respond. Similarly, the compute node’s subsequent DHCP broadcast gets serviced in the same order, with each secondary master waiting `masterdelay` seconds for a preferred master to respond.

Example:

```plaintext
masterorder 0,5,10-20 10.1.0.1 10.2.0.1
masterorder 1-4,21-30 10.2.0.1 10.1.0.1
```

If master 10.1.0.1 is down or fails to respond to PXE/DHCP requests to compute node 10, then master 10.2.0.1 becomes the primary parent for compute node 10.

**masterpxedelay SECS**  The `masterpxedelay` keyword specifies the timeout value in seconds for a non-primary master node to delay sending a response to an incoming PXE request. The default value is 5 seconds.

**mcastbcast interface**  The `mcastbcast` keyword directs the `beoserv` daemon to use broadcast instead of multicast when transmitting files over the interface. This is useful when network equipment has trouble with heavy multicast traffic.

**mcastthrottle interface rate**  The `mcastthrottle` keyword controls the rate at which data is transmitted over the specified interface. The rate is given in megabits per second. This is useful when the compute node interfaces cannot keep up with the master interface when sending large files.

**mkfs mkfs-policy**  The `mkfs` keyword specifies the policy to use when building a Linux file system on the compute nodes. The valid policies are “never”, “if_needed”, or “always”.

- **never**  The filesystem on the compute nodes will never be recreated on boot.
- **if_needed**  The filesystem on the compute nodes will only be recreated if the filesystem check fails.
- **always**  The filesystem on the compute nodes will be recreated on every boot. `fsck` will be assumed to be set to “never” when this is set.

**modarg options**  The `modarg` keyword specifies options to be used for modules that are loaded during the boot process without options. This is useful for specifying options to modules that get loaded during the PCI scan.

**moddep module-list**  The `moddep` keyword is used to specify module dependencies. The first module listed is dependent on the remaining modules in the space separated list. The first module will be loaded after all other listed modules. Module dependency information is normally automatically generated by the `beoboot` script.

**modprobe modulename [options]**  The `modprobe` keyword specifies the name of the kernel module to be loaded with dependency checking, along with any specified module options. Note that the modulename must also be named by a `bootmodule` keyword.

**node [nodenumber] MACaddress**  The `node` keyword is used to assign MAC addresses to node numbers. There should be one of these lines for each node in your cluster. Note the following:

- If a value is not provided for the nodenumber argument, the first node entry is node 0, the second is node 1, the third is node 2, etc.
- The value “off” can be used for the MACaddress argument to leave a place holder for that node number.
- To skip a node number, use the value “node” or “node off” for the MACaddress argument.
- To skip a node number and make sure it will never be automatically filled in by something later in the future, use the value “node reserved” for the MACaddress argument.
nodeaccesses \[ -M | -S nodenumber | all \] arglist  The nodeaccess keyword overrides the default access permissions for the master node (-M), for all compute nodes (all), or for a specific compute node (nodenumber). The remaining arglist is passed directly to the bpctl command for parsing and execution. See the Administrator’s Guide for details about node access permissions.

Example:

\begin{verbatim}
nodeaccess -M -m 0110
nodeaccess -S 5 -g physics
nodeaccess -S 6 -g physics
\end{verbatim}

nodeassign nodeassign-method  The nodeassign keyword specifies the node assignment strategy used when the beoserv daemon receives a new, unknown MAC address from a computer that is not currently entered in the node database. The total number of entries in the node database is limited to the number specified with the nodes keyword (see above).

The valid node assignment methods are “append”, “insert”, “manual”, or “locked”. Note the following:

- “Append” and “insert” are the only two choices that allow new nodes to be automatically given node numbers and welcomed into the cluster.
- Any failures of automatic node assignment through “append” or “insert” (such as when the node table is full) will cause the node assignment to be treated as “manual”.

append  This is the default setting. The system will append new MAC addresses to the end of the node list in the /etc/beowulf/config file. This is done by seeking out the highest already-assigned node number and attempting to go one number beyond it. If the highest node number in the cluster has already been assigned, the “append” method will fail and the “manual” method will take precedence.

insert  The system will insert new MAC addresses into the node list in the /etc/beowulf/config file, starting with the lowest vacant node number. If no spaces are available, the “append” method will be used instead. Typically, a user would choose “insert” when replacing a single node if they want the new node entry to appear in the same place as the old node entry. If the node table is full, the “insert” method will fail and the “manual” method will take precedence.

manual  The system will enter new MAC addresses in the /var/beowulf/unknown_addresses file, and require the user to manually assign the new nodes. The node entries will appear in the “Unknown” list in the BeoSetup GUI, which simplifies the node assignment process. An alternative to using the BeoSetup GUI is to manually edit the /etc/beowulf/config file and copy in the new MAC addresses from the /var/beowulf/unknown_addresses file.

locked  The system will ignore DHCP requests from any MAC addresses not already listed in the /etc/beowulf/config file. This prevents nodes from getting added to the cluster accidentally. This is particularly useful in a cluster with multiple masters, because it enables the Cluster Administrator to control which master responds to a new node request. When you are troubleshooting issues related to the cluster not “seeing” new nodes, one of the first things to check is whether nodeassign is set to “locked”.

See the Administrator’s Guide for additional information on configuring nodes with the BeoSetup GUI and on manual node configuration.

nodename name-format [IPv4 Offset or Base] [netgroup]  The nodename keyword defines the primary hostname, as well as additional hostname-aliases for compute nodes. It can also be used to define hostnames and hostname-aliases for non-compute node entities with a per compute node relationship (e.g., to define a hostname and IP address for the IPMI management interface on each compute node). The presence of the (optional) IPv4 parameter determines if the entry is for compute nodes or for non-compute node entities. If no ‘nodename’ keyword is defined for compute nodes, then compute nodes’ primary hostname is of the ‘dot-number’ format (e.g., node 10’s primary hostname is ‘.10’).

name-format  Define a hostname or hostname-alias. The first instance of the nodename keyword with no IPv4 parameter defines the primary hostname format for compute nodes. While the user may define the primary
hostname, the FIRST hostname alias shall always be of the ‘dot-number’ format. This allows compute nodes to always resolve their address from the ‘dot-number’ notation. Additional nodename entries without an IPv4 parameter define additional hostname aliases.

The name-format string must contain a conversion specification for node number substitution. The conversion specification is introduced by a percent sign (the ‘%’ symbol). An optional following digit in the range 1..5 specifies a zero-padded minimum field width. The specification is completed with an ‘N’. An unspecified or zero field width allows numeric interpretation to match compute node host names. For example, “n%N” will match “n23”, “n+23”, and “n0000023”. By contrast, “n%3N” will only match “n001” or “n023”, but not “n1” or “n23”.

**IPv4 Offset or Base** The presence of the optional IPv4 argument defines if the entry is for “compute nodes” (i.e. the entry will resolve to the ‘dot-number’ name) or if the entry is for non-cluster entities that are loosely associated with the compute node. If the argument has a leading zero, then the parameter specifies an IPv4 Offset. If the argument does not lead with a zero, then the argument specifies a ‘base’ from which IP addresses are computed, by adding the ‘node-number’ associated with the non-compute node entity.

**Netgroup** The netgroup parameter specifies a netgroup that contains all the entries generated by the nodename entry.

**nodes numnodes** The nodes keyword specifies the total possible number of nodes in the cluster. This should normally be set to match the iprange. However, if multiple ipranges are specified, then this value should represent the total number of nodes in all the iprange entries.

**pingtimeout SECS** The bpmaster daemon that executes on the master node sends periodic “ping” messages to the bpslave daemon that executes on each compute node, and each bpslave dutifully responds. This interaction serves as mutual bpmasterbpslave assurance that the other daemon and the network link is still alive and well. If bpslave does not see this “ping” message for SECS seconds, then the bpslave goes into “orphan mode”. If run-to-completion is enabled (see the Administrator’s Guide for details), then the node attempts to remain alive and functioning, despite its apparent inability to communicate with the master node. If run-to-completion is not enabled (which is the default), then the node reboots immediately. If bpmaster does not see a ping reply for SECS seconds, then it syslogs this event and breaks its side of the network connection to the compute node.

The default pingtimeout value is 32 seconds. In rare cases, a particular workload may trigger such a “ping timeout” and its associated spontaneous reboot, and using a pingtimeout keyword to increase the timeout value may stop the spontaneous rebooting.

**pci vendorid devicid drivename** The pci keyword specifies what driver should be used in support of the specified PCI device. A device is identified by a unique vendor ID and device ID pair. The vendor and device ID’s can be either in decimal or hexadecimal with the “0x” notation. You should have one of these lines for each PCI ID (a vendor ID combined with a device ID) for each device on your compute nodes that is not already recognized. Any module dependencies or arguments should be specified with moddep and modarg.

**prestage pathname** The prestage keyword names a specific file that each compute node pulls from the master at node boot time. Multiple instances of prestage can be used. If the pathname is a file in one of the libraries directories, then the pathname gets pulled into the compute node’s library cache. Otherwise, the file (and its directory hierarchy) is copied from the master to the compute nodes.

**server transport-protocol port** The server keyword specifies the port numbers that ClusterWare uses for specified transport protocols. Each transport protocol uses a unique default port number. In the event that a default port value conflicts with a port number used by another service (typically, specified in /etc/services), a server keyword must specify an override value. The allowable transport-protocol keywords are “beofs2” (default port 932), “bproc” (default port 933), “beonss” (default port 3045), and “beostats” (default port 5545). (The keyword “tcp” is deprecated - use “beofs2” instead.)
7.4.1.5 Examples

<table>
<thead>
<tr>
<th>Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>iprange</td>
<td>192.168.1.0 192.168.1.50</td>
</tr>
<tr>
<td>nodename</td>
<td>ipmi-n%N 0.0.1.0</td>
</tr>
</tbody>
</table>

In the above example, the hostname “ipmi-n0” has an address of 192.168.2.50. That is, the compute node’s address (192.168.1.50 for compute node 0) plus the IPv4 Offset of 0.0.1.0. The hostname “ipmi-n12” has an address of 192.168.2.12, which is compute node 12’s address plus the IPv4 Offset of 0.0.1.0.

<table>
<thead>
<tr>
<th>Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodename</td>
<td>ib0-n%N 0.1.0.0 infiniband</td>
</tr>
</tbody>
</table>

In the above example, define a hostname for the infiniband interface for each compute node. Using the iprange values in the previous example, the infiniband interface for compute node 0 has a primary hostname of “ib-n0” and resolves to the address 192.169.1.0: node 0’s basic iprange IP address, plus the increment 0.1.0.0. The infiniband interface for compute node 10 has a primary hostname of “ib-n10” and resolves to the address 192.169.1.10. Each of the “ib0-n%N” hostnames belong to the “infiniband” netgroup.

<table>
<thead>
<tr>
<th>Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodename</td>
<td>computenode%N</td>
</tr>
<tr>
<td></td>
<td>cnode%3N</td>
</tr>
</tbody>
</table>

In the above example, the primary hostname for compute node 0 is “computenode0”, and the primary hostname for compute node 12 is “computenode12”. The second nodename entry defines additional hostname aliases. The FIRST hostname alias will always be the ‘dot-number’ notation, so compute node 12’s first hostname alias is “.12”, and the second hostname alias will be “cnode012”. The ‘%’ followed by a three specifies a three-digit field width format for the entry.

The following is an example of a complete Beowulf Configuration File

```
# Beowulf Configuration file

# Network interface used for Beowulf
# Only first argument to interface is important
interface eth1 192.168.1.1 255.255.255.1

# These two should probably agree for most users
iprange 192.168.1.100 192.168.1.107
nodes 8

# Default location of boot images
bootfile /var/beowulf/boot.img
kernelimage /boot/vmlinuz-2.4.17-0.18.12.beo
kernelcommandline apm=power-off

# Default libraries
libraries /lib /usr/lib

# Default file system policies.
fsck full
mkfs if_needed

# beoserv settings
server beofs2 932

# Default Modules
```
# Non-kernel integrated drivers

```
bootmodule 3c959x 8139too dmfe eepro100 epic100 hp100 natsemi
bootmodule ne2k-pci pcnet32 sis900 starfire sundance tlan
bootmodule tulip via-rhine winbond-840 yellowfin
```

# Node assignment method

```
nodeassign append
```

# PCI Gigabit Ethernet.
# * AceNIC and SysKonnect firmwares are very large.
# * Some of these are distributed separate from the kernel

```
bootmodule dl2k hamachi e1000 ns83820 # acenic sk98lin
```

```
node 00:50:8B:D3:25:4D
node 00:50:8B:D3:07:8B
ignore 00:50:8B:D3:31:FB
node 00:50:8B:D3:62:A0
node 00:50:8B:D3:00:66
node 00:50:8B:D3:30:42
node 00:50:8B:D3:98:EA
```

## 7.4.2 beowulf-fstab

### 7.4.2.1 Name

/etc/beowulf/fstab – ClusterWare compute node filesystem control table

### 7.4.2.2 Description

The /etc/beowulf/fstab file on the master node contains a list of filesystems to be mounted on compute nodes at boot time. Its purpose, format, and contents are similar to the traditional /etc/fstab, plus a few additional cluster-specific features.

The ClusterWare fstab system is designed to keep all configuration information on a master node. The /etc/beowulf/fstab file is the default for all compute nodes. Any optional node-specific /etc/beowulf/fstab.N file overrides this default file for node number N.

The root filesystem on each compute node is a tmpfs filesystem that is automatically sized for the available RAM. In earlier versions of Scyld ClusterWare, this root filesystem was explicitly declared in fstab, but this is no longer done.

The compute node’s root filesystem is used to dynamically cache binaries and libraries from the master, to provide space for /tmp and /var/tmp, to provide mountpoints for NFS mounts, etc. Although ClusterWare does not require a harddrive on a Scyld compute node, some clusters employ harddrive(s) for node-local persistent storage, for “scratch” storage to avoid having /tmp and /var/tmp consume tmpfs RAM, or for swap space to expand the available virtual memory space and thus reduce Out-of-Memory conditions.

The ClusterWare fstab interacts with the mkfs and fsck directives in the /etc/beowulf/config file (see man beowulf-config) to control automatic creation or boot-time checking (and potentially repairing) of compute node filesystems on node-local harddrives.
A directive `mkfs always` specifies to rebuild at boot time every harddrive partition specified in `/etc/beowulf/fstab`, and thus should be used with great care so as to not automatically rebuild a partition and thus destroy data that is expected to survive across compute node reboots. Normally the default directive `mkfs never` is used.

A directive `fsck full` specifies to check and potentially repair at boot time every harddrive partition. Alternatively, `fsck safe` specifies to perform an fsck, but to not attempt any repairs; after boot, the cluster administrator may manually perform repairs as needed. A directive `fsck never` is the default, which specifies that no checking be done at boot time.

### 7.4.2.3 Syntax

The syntax and layout is identical to the master node’s `/etc/fstab` file. The file contents are processed line by line. All blank lines and lines that begin with a “#” are ignored. All other lines should have six fields, separated by tabs or spaces.

The first field is the device to mount. For filesystems on local harddrives, this should point to a `/dev` entry, such as `/dev/hda2`. If mounting an NFS filesystem, the device should be specified as hostname:directory, where hostname is the IP address of the NFS server, and directory is the path on the NFS server you want to mount. If the NFS server is the master node, you can use “$MASTER” as the hostname. Currently, hostname cannot be an actual alphanumeric hostname because `/etc/beowulf/fstab` is evaluated at boot time before the compute node’s name service is initialized. For some special filesystems, such as `proc` and `devpts`, the hostname can be set to “none”.

The second field is the mount point. For a swap partition, this should be “swap”, but for all other filesystems, this must be a path that begins with “/”. Any paths that you specify as mount points will be automatically created by the `node_up` script before it tries to mount the filesystem. Ensure that you do not specify the same mount point on more than one line, because this can cause problems. You can have multiple lines that use “swap” as the mount point, but that is the only exception to the rule.

The third field is the filesystem type. This should be “swap” for swap partitions, or a standard Linux filesystem type (e.g., “ext2”, “ext3”, “xfs”), or “nfs” for an NFS file system, or particular pseudo filesystem types (e.g., “proc” for the `proc` filesystem, “devpts” for the `devpts` filesystem). Any filesystem that can normally be used by Linux can also be specified here, but you must also take steps to create the harddrive filesystems on the compute nodes before attempting to mount them.

The fourth field lists the mount options for the filesystem. All options should be comma-separated with no spaces. If you do not know of any specific options to use, then you should use the “defaults” keyword.

In addition to the mount options normally supported by Linux, one additional option is supported by ClusterWare: “nonfatal”. Normally, any mount failure results in an immediate abort of the node boot, and the node state transitions from “boot” to “error”. Adding “nonfatal” to the options overrides this behavior and allows the node boot to continue, potentially to a node “up” state. However, because filesystem mounts have in fact failed, the node may not actually have full functionality. When using the “nonfatal” option, the cluster administrator is encouraged to view the ClusterWare boot log files found in directory `/var/log/beowulf/` to discover potential mount failures and other warnings or soft error conditions. The “nonfatal” option is useful for harddrive filesystems when not all compute nodes share the same number and partitioning of drives, or when NFS mounts might fail because an NFS server is temporarily unavailable or the specified filesystem is not currently exported.

The fifth and sixth fields are left there for compatibility with the standard `fstab` format. These fields are not used at the moment, but are required to be there. We recommend they both be set to “0”.
7.4.2.4 Examples

```bash
# This file is the fstab for nodes.
# One difference is that we allow for shell variable expansions...
#
# Variables that will get substituted:
# MASTER = IP address of the master node. (good for doing NFS mounts)

#/dev/hda2 swap swap defaults,nonfatal 0 0
#/dev/hda3 / ext2 defaults,nonfatal 0 0

# These should always be added
none /proc proc defaults 0 0
none /dev/pts devpts gid=5,mode=620 0 0

# NFS (for example and default friendliness)
$MASTER:/home /home nfs no锁定,nonfatal 0 0
```

7.4.2.5 Files

/etc/beowulf/fstab, /etc/beowulf/fstab, /etc/beowulf/config

7.4.3 beowulf-nsswitch.conf

7.4.3.1 Name

/etc/beowulf/conf.d/nsswitch.conf – NSS config file for compute nodes

7.4.3.2 Description

The Linux Name Service Switch (NSS) is configured by the /etc/nsswitch.conf file, which describes what sources the Name Service uses to resolve queries for each database category. For example, simple nsswitch.conf entries are:

```
password: files
group: files
hosts: files dns
```

A query for a user password uses the passwd database, which informs the Name Service to search the file /etc/passwd. A query for a group name uses the group database, searching /etc/group. A query for a host name uses the hosts database, first searching /etc/hosts, and then if that fails to find the name, then using /lib64/libnss_dns.so to query the DNS server.

The Scyld ClusterWare beonss package enhances the Name Service to provide consistent naming across the cluster. Installing the beonss package modifies the master's /etc/nsswitch.conf, adding “beo” and “bproc” sources to various database categories. For example, for the hosts database, the “beo” source uses functionality in /lib64/libnss_beo.so, interpreting the /etc/beowulf/config file’s nodename and iprange values to translate the node name “n32” into that node’s IP address, and the “bproc” source uses /lib64/libnss_bproc.so to translate the node name “.32” into the same IP address result.
/etc/beowulf/conf.d/nsswitch.conf is copied to each booting compute node and installed there as /etc/nsswitch.conf. A file with a numeric suffix, e.g., /etc/beowulf/conf.d/nsswitch.conf.32, specifies an alternative node-specific file, in this case copied to node n32 at boot time.

The /etc/nsswitch.conf on compute nodes understands the same “bproc” source, plus an additional “kickback” source for various database categories, using functionality in /lib64/libnss_kickback.so to further query the master node for name resolution, assuming that the /etc/beowulf/init.d/03kickbackproxyd is enabled on the master node.

For example, suppose the compute node's nsswitch.conf includes:

```
passwd: files kickback
group: files kickback
hosts: files bproc kickback
```

The addition of a new user and group on the master node amends the master's /etc/passwd and /etc/group files, although that does not affect those files on the already-booted compute nodes. When a compute node client subsequently asks for that new user name's password, the compute node's Name Service fails when searching the passwd "files" source, then the "kickback" functionality queries the master node for the name, which successfully replies with the new user's password information.

### 7.4.3.3 See Also

nss(5), nsswitch.conf(5), services(5), beowulf-config(5), getent(1), getpwent(3), gethostbyname(3), getservent(3), getnetent(3)

### 7.5 Scyld ClusterWare Beostat Libraries

This part of the Reference Guide describes the functions included in the Scyld ClusterWare C libraries for Beostat, the Beowulf Status library. The functions in this library can be used for retrieving performance information about the nodes on the cluster, such as CPU and memory utilization.

#### 7.5.1 beostat_count_idle_cpus

##### 7.5.1.1 Name

beostat_count_idle_cpus -- count number of idle CPUS in cluster

##### 7.5.1.2 Synopsis

```
#include <sys/beostat.h>
```
7.5.1.3 Arguments

threshold The value of CPU usage below which the CPU will be considered idle.

7.5.1.4 Description

beostat_count_idle_cpus executes on the master node and counts the number of CPUs in the entire cluster that are available to the current user/group and have CPU usage below a given threshold. Note that an easy way to count the total number of CPUs available to a user independent of the usage is to use an arbitrarily large threshold value.

7.5.1.5 Examples

```c
int max, fif;
max = beostat_count_idle_cpus (9999.0);
fif = beostat_count_idle_cpus (0.5);
printf("%d of the %d CPUs available are busy.\n", (max-fif), max);
```

7.5.1.6 Return Value

Returns the number of CPUs that are both available for the caller and have usage below the threshold. If an error occurs, it will return -1.

7.5.1.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.2 beostat_count_idle_cpus_on_node

7.5.2.1 Name

beostat_count_idle_cpus_on_node – count number of idle CPUS on a given node

7.5.2.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_count_idle_cpus_on_node (int node, float cpu_idle_threshold);
```
7.5.2.3 Arguments

node  The node of interest

CPU_idle_threshold  The value of CPU usage below which the CPU will be considered idle.

7.5.2.4 Description

beostat_count_idle_cpus_on_node executes on the master node and counts the number of CPUs on a given node that have CPU usage below a given threshold.

7.5.2.5 Examples

```c
int cnt;
cnt = beostat_count_idle_cpus_on_node (3, 0.5);
printf ("Node 3 has %d CPUs below 50% usage.\n", cnt);
```

7.5.2.6 Return Value

Returns the number of CPUs on the give node that have usage below the threshold. If an error occurs, it will return -1.

7.5.2.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.3 beostat_get_avail_nodes_by_id

7.5.3.1 Name

beostat_get_avail_nodes_by_id – get a list of available nodes for a given identity

7.5.3.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_avail_nodes_by_id (int **node_list, uid_t uid, gid_t *gid_list, int gid_
˓→size);
```
7.5.3.3 Arguments

**node_list** A handle that will have memory allocated and filled with the array of nodes. This memory must be freed by the caller.

**uid** The user identifier number

**gid_list** A pointer to a list of group identifier numbers

**gid_size** The number of elements in the previous arguments array

7.5.3.4 Description

`beostat_get_avail_nodes_by_id` executes on the master node and returns a list of nodes that are available to the given user identifier number who also is a member of the group identifier numbers listed. Memory allocated by the function for `node_list` must be freed by the caller.

7.5.3.5 Examples

```c
int cnt, *node_list, gid_size, i;
uid_t uid;
gid_t *gid_list;
uid = getuid();
gid_size = getgroups (0, gid_list);
gid_list = malloc (sizeof (gid_t) * gid_size);
getgroups (gid_size, gid_list);
cnt = beostat_get_avail_nodes_by_id (&node_list, uid, gid_list, gid_size);
printf ("You may run jobs on nodes: ");
for (i = 0; i < cnt; i++)
```

7.5.3.6 Return Value

Returns the number of nodes in `node_list`. If an error occurs, it will return -1.

7.5.3.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.4 `beostat_get_cpu_count`

7.5.4.1 Name

`beostat_get_cpu_count` – return the number of CPUs on the specified node
7.5.4.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_cpu_count (int node, size_t *ncpus);
```

7.5.4.3 Arguments

**node**  
The node to query.

**ncpus**  
A pointer to a `size_t`, which upon successful completion will contain the number of CPUs on the node specified by the `node` parameter.

7.5.4.4 Description

`beostat_get_cpu_count` executes on the master node and returns the number of CPUs on a specified compute node. A CPU count of zero means that the node’s `sendstats` daemon is not executing.

7.5.4.5 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.4.6 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the `sendstats` daemon on the remote node, the `recvstats` daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.5 `beostat_get_cpu_percent`

7.5.5.1 Name

`beostat_get_cpu_percent` – get the CPU usage on a node

7.5.5.2 Synopsis

```c
#include <sys/beostat.h>
float beostat_get_cpu_percent (int node, int cpu);
```

7.5.5.3 Arguments

**node**  
The node to query

**cpu**  
The CPU index on the particular node
7.5.5.4 Description

beostat_get_cpu_percent executes on the master node and returns the current CPU usage as a floating-point value between 0.0 and 1.0.

7.5.5.5 Examples

```c
printf ("CPU 0 on node 3 is %f percent busy.\n", beostat_get_cpu_percent (3, 0));
```

7.5.5.6 Return Value

Return a float between 0.0 and 1.0. If an error occurs, it will return -1.0.

7.5.5.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.6 beostat_get_cpuinfo_x86

7.5.6.1 Name

beostat_get_cpuinfo_x86 – get the time of the last update for node

7.5.6.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_cpuinfo_x86 (int node, struct cpuinfo_x86*cpuinfo);
```

7.5.6.3 Arguments

- **node**  The node to query
- **cpuinfo**  A pointer to a struct beostat_cpuinfo_x86, which is defined as follows (names in comments are entries from cpuinfo_x86 in asm/processor.h):

```
struct beostat_cpuinfo_x86
{
    int processor;  /* [which cpu (SMP)] */
    char vendor_id[16];  /* x86_vendor_id */
    int family;  /* x86 */
    int model;  /* x86 model */
    char name[64];  /* x86 model ID */
    int stepping;  /* x86 mask */
    float MHz;  /* derived from bogomips */
    int cache_size_KB;  /* x86_cache_size */
    boolean fdiv_bug;  /* same */
};
```

(continues on next page)
7.5.6.4 Description

beostat_get_cpuinfo_x86 executes on the master node and returns a structure describing information about the CPU on the host node. The information in this structure parallels the output seen in /proc/cpuinfo. Note that since this information is architecture specific, this function has “x86” in its name.

7.5.6.5 Examples

```c
struct beostat_cpuinfo_x86 cpuinfo;
beostat_get_cpuinfo_x86 (3, &cpuinfo);
printf ("Node 3 has a %f MHz processor\n", cpuinfo.MHz);
```

7.5.6.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.6.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.7 beostat_get_disk_usage

7.5.7.1 Name

beostat_get_disk_usage – get the disk usage on root partition of a node
7.5.7.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_disk_usage (int node, int *max, int *curr);
```

7.5.7.3 Arguments

- **node**: The node to query
- **max**: A pointer to an int. Upon successful completion, will contain the capacity of the root partition of the node’s disk in megabytes.
- **curr**: A pointer to an int. Upon successful completion will contain the current usage of the root partition of the node’s disk in megabytes.

7.5.7.4 Description

`beostat_get_disk_usage` executes on the master node and returns the current disk usage, as well as the total capacity of the disk in megabytes.

7.5.7.5 Examples

```c
int max, curr;
beostat_get_disk_usage (3, &max, &curr);
printf ("CPU 0 on node 3's disk is %f percent full.\n",
        (double) curr / (double) max );
```

7.5.7.6 Return Value

Returns 0 upon successful completion. If an error occurs, it will return -1.

7.5.7.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.8 beostat_get_last_multicast

7.5.8.1 Name

`beostat_get_last_multicast` – get file system statistics for the root file system on a node
7.5.8.2 Synopsis

```
#include <sys/beostat.h>
time_t beostat_get_last_multicast (void);
```

7.5.8.3 Description

`beostat_get_last_multicast` executes on the master node and returns the time of the last multicast request sent to the nodes. It is usually reserved for internal use.

7.5.8.4 Return Value

Returns the time in seconds since Epoch (00:00:00 UTC, January 1, 1970) of the last multicast request. If an error occurs, it will return -1.

7.5.8.5 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.9 beostat_get_loadavg

7.5.9.1 Name

`beostat_get_loadavg` – get load average on a node

7.5.9.2 Synopsis

```
#include <sys/beostat.h>
int beostat_get_loadavg (int node, struct beostat_loadavg *loadavg);
```

7.5.9.3 Arguments

- **node** The node to query
- **loadavg** A pointer to a `struct beostat_loadavg`, which is defined as follows:

```
struct beostat_loadavg
{
    float load[3];
    int num_active_procs;
    int total_procs;
    int last_pid;
};
```
7.5.9.4 Description

`beostat_get_loadavg` executes on the master node and returns the load average information of a node in the cluster. The three values returned are averages over increasing time durations.

7.5.9.5 Examples

```c
struct beostat_loadavg loadavg;
beostat_get_loadavg (3, &loadavg);
printf ("The load process ID on node 3 was %d.\n", loadavg.last_pid);
```

7.5.9.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.9.7 Errors

This function relies on the Beostat subsystem, which consists of the `proc` filesystem on the remote node, the `sendstats` daemon on the remote node, the `recvstats` daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.10 beostat_get_meminfo

7.5.10.1 Name

`beostat_get_meminfo` – get information about the memory usage on a node

7.5.10.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_meminfo (int node, struct beostat_meminfo *meminfo);
```

7.5.10.3 Arguments

node The node to query

meminfo A pointer to a `struct beostat_meminfo`, which is defined as follows:

```c
struct beostat_meminfo
{
    struct beostat_memusage mem;
    struct beostat_memusage swap;
    unsigned long long shared;
    unsigned long long buffers;
    unsigned long long cached;
};
```

where `struct beostat_memusage` is defined as follows:
struct beostat_memusage
{
    unsigned long long used;
    unsigned long long free;
};

7.5.10.4 Description

beostat_get_meminfo executes on the master node and returns the memory usage of a node in the cluster. All values are in bytes.

Warning: Since Linux aggressively caches the hard disk into memory it will often appear to always be about 90% used. Some have suggested that the values of buffers and cached added together should be subtracted from the reported memory usage. However, these values may not be mutually exclusive.

7.5.10.5 Examples

meminfo_t meminfo;
beostat_get_meminfo (3, &meminfo);
printf ("The node 3 has %s bytes free\n", meminfo.mem.free);

7.5.10.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.10.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.11 beostat_get_MHz

7.5.11.1 Name

beostat_get_MHz – get the speed of the processor on a node

7.5.11.2 Synopsis

#include <sys/beostat.h>
int beostat_get_MHz (int node, float *MHz);
7.5.11.3 Arguments

node  The node to query

MHz  A pointer to a float, which will contain the speed of processor on the node in megahertz upon successful completion.

7.5.11.4 Description

beostat_get_MHz executes on the master node and returns the speed of CPU(s) on a given node in units of megahertz. On multi-CPU (SMP) machines it is assumed that all CPUs are the same speed. This is currently a hardware requirement on all known SMP machines.

7.5.11.5 Examples

```c
float speed;
beostat_get_MHz (3, &speed);
printf ("The node 3 has a %f MHz processor\n", speed);
```

7.5.11.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.11.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.12 beostat_get_name

7.5.12.1 Name

beostat_get_name – get the name of node

7.5.12.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_name (int node, char **name);
```
7.5.12.3 Arguments

node  The node to query
name  A handle to a char, which will be allocated with an appropriate amount of memory and then set to the name of a node. The caller must free the allocated memory when it is done with the memory.

7.5.12.4 Description

beostat_get_name executes on the master node and returns the name of a given node.

7.5.12.5 Examples

```c
char *name;
beostat_get_name (3, &name);
printf ("The name for node 3 is %s\n", name);
free (name);
```

7.5.12.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.12.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.13 beostat_get_net_dev

7.5.13.1 Name

beostat_get_net_dev – get the network interface statistics on a node

7.5.13.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_net_dev (int node, struct beostat_net_dev *devs, int size);
```
### 7.5.13.3 Arguments

**node** The node to query

**devs** A pointer to a array of structures of the type `struct beostat_net_dev`, which is defined as follows:

```c
struct beostat_net_dev {
    char name[16];
    struct beostat_net_stat recv;
    unsigned long frame;
    unsigned long multicast;
    struct beostat_net_stat trans;
    unsigned long colls;
    unsigned long carrier;
};
```

where `beostat_net_stat` is defined as follows:

```c
struct beostat_net_stat {
    unsigned long bytes;
    unsigned long packets;
    unsigned long errs;
    unsigned long drop;
    unsigned long fifo;
    unsigned long compressed;
};
```

**size** The number of `beostat_net_dev` structures allocated by the caller.

### 7.5.13.4 Description

`beostat_get_net_dev` executes on the master node and returns the network interface statistics of a node. The caller must allocate the memory for the array of structures, and a maximum of `MAX_NET_DEV` or `size` entries will be filled (whichever is smaller). Unused space in the structure(s) are filled with zeros.

### 7.5.13.5 Examples

```c
int i;
struct beostat_net_dev net_dev[MAX_NET_DEV];
beostat_get_net_dev (3, net_dev, MAX_NET_DEV);
for (i = 0; i < MAX_NET_DEV; i++)
```
7.5.13.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.13.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.14 beostat_get_net_rate

7.5.14.1 Name

beostat_get_net_rate – get the cumulative network interface on a node

7.5.14.2 Synopsis

```
#include <sys/beostat.h>
unsigned long beostat_get_net_rate (int node);
```

7.5.14.3 Arguments

node The node to query

7.5.14.4 Description

beostat_get_net_rate executes on the master node and returns the current network usage rate in bytes per second across all interfaces on that node.

7.5.14.5 Examples

```
printf ("Node 3 is currently transferring %d bytes / second.\n", beostat_get_net_rate(3));
```

This function can give erroneous results for its transfer counts during the moment of rollover of each interface.

7.5.14.6 Return Value

Returns an unsigned long, which represents the network transfer rate. If an error occurs, it will return -1.
7.5.14.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.15 beostat_get_stat_cpu

7.5.15.1 Name

beostat_get_stat_cpu – get the statistics of CPU utilization

7.5.15.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_stat_cpu (int node, int cpu, struct beostat_stat_cpu *stat_cpu);
```

7.5.15.3 Arguments

- **node** The node to query
- **cpu** The CPU index on the particular node
- **stat_cpu** A pointer to a struct beostat_stat_cpu, which will be filled upon successful completion. struct beostat_stat_cpu is defined as follows:

```c
struct beostat_stat_cpu
{
    long user;
    long system;
    long nice;
    long idle;
};
```

The members of this structure have the following meanings:

- **user** The number of CPU ticks spend processing normal priority (0) user level instructions.
- **nice** The number of CPU ticks spend processing nice priority (>0) user level instructions.
- **system** The number of CPU ticks spend processing system (kernel) level instructions.
- **idle** The number of CPU ticks spend idle.
7.5.15.4 Description

`beostat_get_stat_cpu` executes on the master node and returns the cpu ticks counts on a given node/CPU. These ticks just keep incrementing over time until they overflow and wrap back around. To get actual CPU usage over some time period, you must either take the derivative of these values or use the `beostat` convenience function `beostat_get_cpu_percent`.

7.5.15.5 Examples

```c
struct beostat_stat_cpu stat_cpu;
beostat_get_stat_cpu (3, 0, &stat_cpu);
printf ("There have been %ld idle ticks on cpu 0 for node 3 is %s\n", stat_cpu.idle);
free (name);
```

7.5.15.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.15.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.16 beostat_get_statfs_p

7.5.16.1 Name

`beostat_get_statfs_p` – get file system statistics for the root file system on a node

7.5.16.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_statfs_p (int node, struct statfs *statfs);
```

7.5.16.3 Arguments

- **node** The node to query
- **statfs** A pointer to a `statfs` structure that will be filled upon successful completion. See the man page for statfs2 for a description of the fields.
7.5.16.4 Description

`beostat_get_statfs_p` executes on the master node and returns the filesystem statistics for the root filesystem on a given node.

*Warning:* Since Linux aggressively caches the hard disk into memory it will often appear to always be about 90% used. Some have suggested that the values of `buffers` and `cached` added together should be subtracted from the reported memory usage. However, these values may not be mutually exclusive.

7.5.16.5 Examples

```c
statfs_p_t statfs_p;
beostat_get_statfs_p (3, &statfs_p);
printf ("The node 3 has %s bytes free\n", statfs_p.mem.free);
```

7.5.16.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.16.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the `sendstats` daemon on the remote node, the `recvstats` daemon on the master node, and two shared memory files in the directory `/var/shm`. If any part of the system breaks down, this function could fail.

7.5.17 `beostat_get_time`

7.5.17.1 Name

`beostat_get_time` – get the time of the last update for node

7.5.17.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_get_time (int node, struct node_time *node_time);
```

7.5.17.3 Arguments

**node** The node to query

**node_time** A pointer to a struct `node_time`, which is defined as follows:

```c
struct node_time {
    time_t time;
};
```
7.5.17.4 Description

beostat_get_time executes on the master node and returns the time of the last update to the Beostat system by a given node. The Beostat functionality works by the having the sendstats daemon on each compute node periodically send node status information to the master node’s recvstats daemon. This function provides the time of the last update from a given node. It is useful when timely information is required and old information should be disregarded. The time is measured in seconds since the standard UNIX Epoch (00:00:00 UTC, January 1, 1970). Use functions like ctime() to convert to a human readable string.

7.5.17.5 Examples

```c
#include <sys/beostat.h>
time_t time;
beostat_get_time (3, &time);
printf ("The time of the last update for node 3 is %s\n", ctime(&time));
```

7.5.17.6 Return Value

Return 0 on success. If an error occurs, it will return -1.

7.5.17.7 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.5.18 beostat_is_node_available

7.5.18.1 Name

beostat_is_node_available – determine if a given user/group can run on a given node

7.5.18.2 Synopsis

```c
#include <sys/beostat.h>
int beostat_is_node_available (int node, uid_t uid, gid_t *gid_list, int gid_size);
```

7.5.18.3 Arguments

- **node** The node of interest
- **uid** The user identifier number
- **gid_list** A pointer to a list of group identifier numbers
- **gid_size** The number of elements in the previous arguments array
7.5.18.4 Description

beostat_is_node_available executes on the master node and determines if the given user with specified UID and belonging to the groups in gid_list has permission to run on a given node.

See the manual page for beostat_get_avail_nodes_by_id for a example of a similar function.

7.5.18.5 Return Value

Returns 1 if the node can be used, 0 if not, and -1 if an error occurs.

7.5.18.6 Errors

This function relies on the Beostat subsystem, which consists of the proc filesystem on the remote node, the sendstats daemon on the remote node, the recvstats daemon on the master node, and two shared memory files in the directory /var/shm. If any part of the system breaks down, this function could fail.

7.6 Scyld ClusterWare BProc Libraries

This part of the Reference Guide describes the functions included in the Scyld ClusterWare C libraries for BProc, the Beowulf Process Control library. The functions in this library are used to control jobs running on the cluster.

7.6.1 bproc_access

7.6.1.1 Name

bproc_access – Check if the current user may use a cluster node.

7.6.1.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_access (int node, int mode);
int _bproc_access (struct bproc_node_info_t *nodeinfo, int mode);
```

7.6.1.3 Arguments

- **node**  The node number to check.
- **mode**  The mode bits to check against.
- **nodeinfo**  A filled-in bproc_node_info structure to check against.
7.6.1.4 Description

The current user’s ability to execute processes on the specified cluster node is checked. The mode parameter specifies the mode bits to check.

See the Administrator’s Guide for details of the semantics of node ownership and how the settings interact with schedulers.

7.6.1.5 Return Value

If a process may be started on the node, 0 is returned.

If the node is not available or there is an error, -1 is returned and errno is set.

7.6.1.6 Errors

ENOSYS The BProc system is not available.
EIO The BProc system is loaded but is not configured or active.
EACCES This user does not have permission to start jobs on the node.
ENOMEM Insufficient kernel memory was available.

7.6.2 bproc_chown

7.6.2.1 Name

bproc_chown – Change the ownership for a cluster node.

7.6.2.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_chown (int node, int user);
int bproc_chgrp (int node, int group);
int bproc_chown (int node, int user, int group);
```

7.6.2.3 Arguments

- **node**  The node to change ownership of
- **user**  The numeric user ID to assign to the node
- **group**  The numeric group ID to assign to the node
7.6.2.4 Description

The owner of the cluster node is changed. The user specifies the desired user ID (UID). The second form, available only with the BProc v2 compatibility library, sets the group owner to group.

Previous BProc versions used the values BPROC_USER_ANY and BPROC_GROUP_ANY. The same effect can be achieved by setting world-execute permission using bproc_chmod3.

See the Administrator’s Guide for details of the semantics of node ownership and how the settings interact with schedulers.

7.6.2.5 Return Value

Returns 0 on success.

Returns -1 on error, and sets errno.

7.6.2.6 Errors

ENOSYS The BProc system is not available.

EIO The BProc system is loaded but not configured or active.

EPERM This process does not have permission to change node ownership.

ENOMEM Insufficient kernel memory was available to change ownership.

7.6.3 bproc_curnode

7.6.3.1 Name

bproc_curnode – Get the current node number

7.6.3.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_curnode (void);
```

7.6.3.3 Return Value

Returns the node number of the machine on which this process is currently running. The value BPROC_NODE_MASTER, -1, indicates that the process is running on the master.
7.6.3.4 Bugs

This function will return -1 if there is an error in processing, or if you are on the master node. If there is the possibility of ambiguity, the errno variable should be initialized to 0 before the call and checked for errors after the call.

7.6.3.5 Errors

ENOSYS  The BProc system is not available.
EIO   The BProc system is loaded, but is not configured or active.
ENOMEM Insufficient kernel memory was available to return a value.

7.6.4 bproc_detach

7.6.4.1 Name

bproc_detach – Remove the current process from the BProc process space.

7.6.4.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_detach(long code);
```

7.6.4.3 Description

bproc_detach removes the current process from the global BProc process space. After bproc_detach succeeds, the process continues to execute on its node, but is no longer visible from other nodes. From the viewpoint of other processes in the BProc system, the effect is as if the process had executed exit(code). See exit2 for a description of the effects seen by the parent process.

7.6.4.4 Return Value

On success, bproc_detach returns zero.
On error, bproc_detach returns -1 and sets errno appropriately.

7.6.4.5 Errors

EPERM  The caller does not have root permissions.
ENOSYS The BProc system is not loaded in the current kernel.

In addition, any errors listed in fork(2) may occur.
7.6.4.6 Notes

In the current release, `bproc_detach` may change the PID and PPID of the current process, but is not guaranteed to do so. The PPID may change at some point after `bproc_detach` has returned, but such behavior is not guaranteed.

In future releases, `bproc_detach` may be implemented to remove the current process from the BProc system, or it may be implemented to make a copy of the current process outside of the BProc system; other options are also possible. The user should make no assumptions beyond those documented in this man page.

Some implementations of GNU libc cache the value of `getpid` in userspace; thus, `getpid` may return inaccurate values if called both before and after `bproc_detach`.

7.6.5 `bproc_execmove`

7.6.5.1 Name

`bproc_execmove` – Exec a local binary on a remote node

7.6.5.2 Synopsis

```c
#include <sys/bproc.h>
int _bproc_execmove_io (int node, int port, const char * cmd, \char * const argv[], char * const envp[]);
int bproc_execmove (int node, const char * cmd, \char * const argv[], char * const envp[]);
```

7.6.5.3 Arguments

- **node**  The destination node for the child process.
- **port**  The IP port BProc should connect back to for I/O forwarding.
- **cmd**  The program to execute
- **argv**  The argument list
- **envp**  The environment

7.6.5.4 Description

This function allows execution of local binaries on remote nodes. BProc will load the binary image on the current node and then move it to a remote node, prior to executing the binary image.

**NOTE:** This migration mechanism will move the binary image but not any dynamically loaded libraries that the application might need. Therefore any libraries that the application uses must be present on the remote system. Function does not return on success. On failure, it returns -1 and sets `errno` appropriately.

`port` is the TCP port BProc should connect back to handle I/O forwarding. A `port` value of 0 means it assumes I/O forwarding is being done on the existing socket for stdout and stderr only. Any other value and it will try to connect back to that port and open three connections, one for stdout, one for stderr, and one for stdin.

If you use `bproc_execmove`, `port` has a default value of 0.
7.6.5.5 Return Value

Does not return on success.
Returns on error, and sets errno.

7.6.5.6 Errors

EPERM The filesystem where cmd resides is mounted nosuid and the program is suid or sgid
ENOMEM Out of memory
EBUSY No Master
EFAULT cmd, envp, or argv points to memory that is not accessible the by the program.
EACCES The program does not have execute permission on cmd
E2BIG Argument list is too big
ENOEXEC cmd is not in a recognized executable format or is for the wrong architecture
ENAMETOOLONG cmd is too long
ENOENT cmd does not exist.
ENOTDIR Part of the path to cmd is not a directory.
ELOOP Too many symbolic links were encountered when resolving cmd.
ETXTBSY cmd is open for writing by another program.
EIO An I/O error occurred.
ENFILE The limit on open files has been reached.
EINVAL An ELF executable had more than one PT_INTERP segment.

7.6.6 bproc_getnodebyname

7.6.6.1 Name

bproc_getnodebyname – Get a node number from a node name.

7.6.6.2 Synopsis

```
#include <sys/bproc.h>
int bproc_getnodebyname (const char * name);
```
7.6.6.3 Arguments

name  A machine name

7.6.6.4 Description

This function returns the node number associated with the string name. Valid strings include “master”, “self”, a string representation of a decimal number, and the string representation of a decimal number prepended with a “.”

Note that this function duplicates some of the functionality of the BeoNSS system, but with a limited set of names. Note also that this function does not use BProc kernel information: it returns a value even when the cluster system is not active, and it may return a node number that is outside the valid range of nodes.

7.6.6.5 Return Value

Returns the node number represented by name.
May return BPROC_NODE_SELF if the string is “self”.
Returns BPROC_NODE_NONE if a valid string was not passed.

7.6.6.6 Errors

No errors

7.6.7 bproc_masteraddr

7.6.7.1 Name

bproc_masteraddr – Get the private cluster network IP address for the master node.

7.6.7.2 Synopsis

```
#include <sys/bproc.h>
int bproc_masteraddr (struct sockaddr * addr, int * size);
```

7.6.7.3 Arguments

addr  pointer to a struct sockaddr
size  The size of addr
7.6.7.4 Description

Save the master node’s IP address in the struct sockaddr pointed to by addr. size should be initialized to indicate the amount of space pointed to by addr. On return it contains the actual size of the addr returned (in bytes).

7.6.7.5 Return Value

Returns 0 on success.

Returns -1 on error, and sets errno.

7.6.7.6 Errors

EFAULT addr or size points to memory that is not accessible by the program.

EIO There was an I/O error.

ENOMEM Out of memory error.

7.6.8 bproc_move

7.6.8.1 Name

bproc_move – Move the running process to another node

7.6.8.2 Synopsis

```
#include <sys/bproc.h>
int _bproc_move_io (int node, int flags, int port);
int _bproc_move (int node, int flags);
int bproc_move (int node);
```

7.6.8.3 Arguments

node The node to move to

flags Flags for VMAdump.

port The IP port BProc should connect back to for I/O forwarding.

7.6.8.4 Description

This call will move the current process to the remote node number given by node. It returns 0 on success, -1 on failure. errno is set on failure.

node is the node to move to.

flags can be one of the following: BPROC_DUMP_LIBS, BPROC_DUMP_EXEC, BPROC_DUMP_OTHER or any combination of them binary OR’d together. A binary OR of all three, BPROC_DUMP_ALL, is also provided as a shortcut. These flags tell VMAdump how much of the running process to dump and send to the compute node.
port is the port BProc should connect back to, to handle I/O forwarding. A port value of 0 means it assumes I/O forwarding is being done on the existing socket for stdout and stderr only. Any other value and it will try to connect back to that port and open three connections, one for stdout, one for stderr, and one for stdin.

If you use _bproc_move or bproc_move, port has a default value of 0. If you use bproc_move, flags takes a default value that is BPROC_DUMP_EXEC|BPROC_DUMP_OTHER if you’re trying to move to an up node or the master, otherwise it is BPROC_DUMP_EXEC|BPROC_DUMP_LIBS|BPROC_DUMP_OTHER

7.6.8.5 Return Value

Returns 0 on success.
Returns -1 on error, and sets errno.

7.6.8.6 Errors

EBUSY No master?
ENOMEM Out of memory.
EIO An I/O error occurred.

7.6.9 bproc_nodeaddr

7.6.9.1 Name

bproc_nodeaddr – Get the IP address for a node.

7.6.9.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_nodeaddr (int node, struct sockaddr * addr, int * size);
```

7.6.9.3 Arguments

- **node** The node number
- **addr** pointer to a struct sockaddr
- **size** The size of addr

7.6.9.4 Description

Save the node’s IP address in the struct sockaddr pointed to by addr. The size element should be initialized to indicate the amount of space pointed to by addr. On return it contains the actual size of the addr returned (in bytes).
7.6.9.5 Return Value

Returns 0 on success.
Returns -1 on error, and sets errno.

7.6.9.6 Errors

EFAULT addr or size points to memory that is not accessible by the program.
EIO There was an I/O error.
ENOMEM Out of memory error.

7.6.10 bproc_nodeinfo

7.6.10.1 Name

bproc_nodeinfo – Get general status information for a node

7.6.10.2 Synopsis

```
#include <sys/bproc.h>
int bproc_nodeinfo (int node, struct bproc_node_info_t * info);
```

7.6.10.3 Arguments

node The node you want information on.
info Pointer to a struct bproc_node_info_t.

7.6.10.4 Description

This function will get information about the node and fill that information into the struct bproc_node_info_t.

```
struct bproc_node_info_t {
    int node; /* Same as bproc_currnodule */
    int status; /* Same as bproc_nodestatus */
    int mode; /* The node's access permissions */
    uid_t user; /* The uid and gid of the user */
    gid_t group; /* to which the node is assigned */
    uint32_t addr; /* The node's 32-bit struct sockaddr */
}
```

See the Administrator’s Guide for more information on the user and group.
7.6.10.5 Return Value

Returns 0 on success.
Returns -1 on error, and sets errno.

7.6.10.6 Errors

EFAULT info points to memory that is inaccessible by the program.
EIO I/O Error
ENOMEM Out of Memory

7.6.11 bproc_nodenumber

7.6.11.1 Name

bproc_nodenumber – Get the node number based on the given IP address.

7.6.11.2 Synopsis

```
#include <sys/bproc.h>
int bproc_nodenumber (struct sockaddr * addr, int size);
```

7.6.11.3 Arguments

addr pointer to a struct sockaddr, that has the IP filled in
size The size of addr

7.6.11.4 Description

Retrieves the IP address from the sockaddr structure and provides the number of the node with that address. There is a direct one-to-one mapping of node number to IP address as given in the /etc/beowulf/config file. Node numbering starts at 0 with the first IP address in the range and increments by 1 up to the last IP address in the range.

7.6.11.5 Return Value

Returns the node number associated with the IP address.
Returns BPROC_NODE_NONE if no valid node was found.
7.6.11.6 Errors

EFAULT   addr points to memory that is not accessible by the program.
EIO       There was an I/O error.
ENOMEM    Out of memory error.

7.6.12 bproc_nodestatus

7.6.12.1 Name

bproc_nodestatus – Returns the status of the given node.

7.6.12.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_nodestatus (int node);
```

7.6.12.3 Arguments

node   The node number.

7.6.12.4 Description

This node argument should list one of the compute nodes, not the master. The master is considered to be always up.

7.6.12.5 Return Value

On error, it will return -1 and set errno appropriately.

The possible states are:

- **bproc_node_down**   The node is not connected to the master daemon. It may be off or crashed or not far enough along in its boot process to connect to the master daemon.
- **bproc_node_unavailable**   The node is running but is currently unavailable to users. Nodes are only in this state if set that way explicitly by the administrator.
- **bproc_node_error**   There is a problem with the node. Nodes are assigned this state if booting is unsuccessful.
- **bproc_node_up**   The node is up and ready to accept processes. This is the only state in which non-root users can send jobs to the node.
- **bproc_node_reboot**   The node was told to reboot and has not come back up yet.
- **bproc_node_halt**   The node was told to halt and is still down.
- **bproc_node_pwroff**   The node was told to power off and is still down.
- **bproc_node_boot**   The node is in the process of coming up (running the node_up script).
7.6.12.6 Errors

There was an I/O error.

ENOMEM Out of memory error.

7.6.13 bproc_numnodes

7.6.13.1 Name

bproc_numnodes – Get the count of cluster nodes.

7.6.13.2 Synopsis

#include <sys/bproc.h>

int bproc_numnodes (void);

7.6.13.3 Description

This function returns the number of nodes the cluster is configured to support. Note that this is the potential size of the cluster, not the current number of available nodes or the count of machines assigned node numbers.

7.6.13.4 Return Value

Returns the number of compute nodes the system is configured to support. If the BProc system is not loaded, returns 0 and sets errno to ENOSYS.

Returns -1 on error, and sets errno.

7.6.13.5 Errors

EIO The BProc system is loaded but not configured or active.

ENOMEM Insufficient kernel memory was available.

7.6.14 bproc_pidnode

7.6.14.1 Name

bproc_pidnode – Get the node a PID is running on.
7.6.14.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_pidnode (int pid);
```

7.6.14.3 Arguments

- **pid** The process id

7.6.14.4 Description

Retrieves the node number associated with the given Process ID from the BProc process space. Note that only user processes ghosted on the master are available in this way. Node kernel and internal BProc PIDs are not accessible.

7.6.14.5 Return Value

Return the node number that the PID is running on.

Returns **BPROC_NODE_NONE** if the PID is running on the master node or isn’t a valid PID.

7.6.14.6 Bugs

This function returns **BPROC_NODE_NONE** if there was an error accessing the BProc status file, or if **pid** was not found in the status file. There is currently no way to tell if **BPROC_NODE_NONE** resulted from an error or from **pid** not being masqueraded by BProc.

7.6.14.7 Errors

- **EACCES** The program does not have access to read the BProc status file.
- **ENOENT** The BProc status file does not exist.
- **ENOMEM** Insufficient kernel memory.
- **EMFILE** The limit of files that can be opened by the process has been reached.
- **ENFILE** The limit of files that can be opened by the system has been reached.
- **EAGAIN** The BProc status file has been locked.

7.6.15 bproc_rexec

7.6.15.1 Name

bproc_rexec – exec a program on a remote node
7.6.15.2 Synopsis

```c
#include <sys/bproc.h>
int _bproc_rexec_io (int node, int port, const char * cmd, 
    char * const argv[], char * const envp[]);
int bproc_rexec (int node, const char * cmd, 
    char * const argv[], char * const envp[]);
```

7.6.15.3 Arguments

- **node** The node the child should be on.
- **port** The port to BProc should connect back to for I/O forwarding.
- **cmd** The program to execute
- **argv** The argument list
- **envp** The environment

7.6.15.4 Description

This call has semantics similar to `execve`. It replaces the current process with a new one. The new process is created on `node` and the local process becomes the ghost representing it. All arguments are interpreted on the remote machine. The binary and all libraries it needs must be present on the remote machine. Currently, if remote process creation is successful but exec fails, the process will just exit with status 1. If remote process creation fails, the function will return -1 and `errno` is set appropriately.

`port` is the TCP port `BProc` should connect back to to handle I/O forwarding. A `port` value of 0 means it assumes I/O forwarding is being done on the existing socket for `stdout` and `stderr` only. Any other value and it will try to connect back to that port and open three connections, one for `stdout`, one for `stderr`, and one for `stdin`.

If you use `bproc_execmove`, `port` has a default value of 0.

7.6.15.5 Return Value

Does not return on success.

Returns -1 on error, and sets `errno`.

7.6.15.6 Errors

- **EPERM** The filesystem where `cmd` resides is mounted nosuid and the program is suid or sgid
- **ENOMEM** Out of memory
- **EBUSY** No Master
- **EFAULT** `cmd`, `envp`, or `argv` points to memory that is not accessible by the program.
- **EACCES** The program does not have execute permission on `cmd`
- **E2BIG** Argument list is too big
- **ENOEXEC** `cmd` is not in a recognized executable format or is for the wrong architecture
- **ENAMETOOLONG** `cmd` is too long
ENOENT  cmd does not exist.
ENOTDIR  Part of the path to cmd is not a directory.
ELOOP    Too many symbolic links were encountered when resolving cmd.
ETXTBSY  cmd is open for writing by another program.
EIO      An I/O error occurred.
ENFILE   The limit on open files has been reached.
EINVAL   An ELF executable had more than one PT_INTERP segment.

7.6.16  bproc_rfork

7.6.16.1 Name
bproc_rfork – fork, with the child ending up on a remote node.

7.6.16.2 Synopsis

```c
#include <sys/bproc.h>
int _bproc_rfork_io (int node, int flags, int port);
int _bproc_rfork (int node, int flags);
int bproc_rfork (int node);
```

7.6.16.3 Arguments

node  The node the child should be on.
flags Flags for VMAdump.
port  The port BProc should connect back to for I/O forwarding.

7.6.16.4 Description

The semantics of this function are designed to mimic fork, except that the child process created will end up on the node given by the node argument. The process forks a child and that child performs a bproc_move to move itself to the remote node. Combining these two operations in a system call prevents zombies and SIGCHLDs in the case that the fork is successful but the move is not.

On success, this function returns the process ID of the new child process to the parent and 0 to the child. On failure it returns -1, and errno is set appropriately.

node is the node the child should be on.
flags can be one of the following: BPROC_DUMP_LIBS, BPROC_DUMP_EXEC, BPROC_DUMP_OTHER or any combination of them binary OR’d together. If you wish to use all of them, you can also use BPROC_DUMP_ALL as a shortcut. These flags tell VMAdump how much of the running process to dump and send to the compute node.
port is the port BProc should connect back to for I/O forwarding. A port value of zero means it assumes I/O forwarding is being done on the existing socket for stdout and stderr only. Any other value and it will try to connect back to that port and open three connections, one for stdout, one for stderr, and one for stdin.
If you use _bproc_rfork or bproc_rfork, port has a default value of 0. If you use bproc_rfork, flags takes a default value that is BPROC_DUMP_EXEC|BPROC_DUMP_OTHER, if you are trying to move to an up node or the master, otherwise it is BPROC_DUMP_EXEC|BPROC_DUMP_LIBS|BPROC_DUMP_OTHER.

7.6.16.5 Return Value

For the parent process, this will return the PID of the child process.
For the child process, this will return 0.
If there is an error, -1 will be returned to the parent process and there will be no child process.

7.6.16.6 Errors

EBUSY No Master

7.6.17 bproc_setnodestatus

7.6.17.1 Name

bproc_setnodestatus – Change the status of a node

7.6.17.2 Synopsis

```c
#include <sys/bproc.h>
int bproc_setnodestatus (int node, int status);
```

7.6.17.3 Arguments

node The node to change the status of
status The new status for the node

7.6.17.4 Description

This call sets the status of a node. Note that it is not possible to change the status of a node that is marked as “down”, “pwroff”, or “halt”.

bproc_node_down The node is not connected to the master daemon. It may be off or crashed or not far enough along in its boot process to connect to the master daemon.

bproc_node_unavailable The node is running but is currently unavailable to users. Nodes are only in this state if set that way explicitly by the administrator.

bproc_node_error There is a problem with the node. Nodes are assigned this state if booting is unsuccessful.

bproc_node_up The node is up and ready to accept processes. This is the only state in which non-root users can send jobs to the node.

bproc_node_reboot Setting a node to this state will tell it to reboot.

bproc_node_halt Setting a node to this state will tell it to halt.

bproc_node_pwroff Setting a node to this state will tell it to power off.
bproc_node_boot  The node is in the process of coming up (running the node_up script). A node should only be put in this state by the BProc master daemon.

7.6.17.5  Return Value

Returns 0 on success.
Returns -1 on error, and sets errno.

7.6.17.6  Errors

EPERM  You do not have root access
ENOMEM  Out of memory
EIO  I/O error
We welcome any reports on errors or difficulties that you may find. We also would like your suggestions on improving this document. Please direct all comments and problems to support@penguincomputing.com.

When writing your email, please be as specific as possible, especially with errors in the text. Please include the chapter and section information. Also, please mention in which version of the manual you found the error. This version is Scyld ClusterWare Release v7.9.16-7916g0000.