Bright Cluster Manager 5.2

Administrator Manual

Revision: 6776

Date: Fri, 27 Nov 2015

Trademarks

Linux is a registered trademark of Linus Torvalds. Pathscale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of The Portland Group Compiler Technology, STMicroelectronics, Inc. SGE is a trademark of Sun Microsystems, Inc. FLEXlm is a registered trademark of Globetrotter Software, Inc. Maui Cluster Scheduler is a trademark of Adaptive Computing, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *Administrator Manual* for the Bright Cluster Manager 5.2 cluster environment.

0.1 Quickstart

For readers who want to get a cluster up and running as quickly as possible with Bright Cluster Manager, Appendix F is a quickstart installation guide.

0.2 About This Manual

The rest of this manual is aimed at helping system administrators install, understand, and manage a cluster running Bright Cluster Manager so as to get the best out of it.

The *Administrator Manual* covers administration topics which are specific to the Bright Cluster Manager environment. Readers should already be familiar with basic Linux system administration, which the manual does not generally cover. Aspects of system administration that require a more advanced understanding of Linux concepts for clusters are explained appropriately.

This manual is not intended for users interested only in interacting with the cluster to run compute jobs. The *User Manual* is intended to get such users up to speed with the user environment and workload management system.

Updated versions of the *Administrator Manual*, as well as the *User Manual*, are always available on the cluster at `/cm/shared/docs/cm`.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications.

The manuals also regularly incorporate customer feedback. Administrator and user input is is greatly valued at Bright Computing, so that any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

0.3 Getting Administrator-Level Support

If the Bright Cluster Manager software was obtained through a reseller or system integrator, then the first line of support is provided by the reseller or system integrator. The reseller or system integrator in turn contacts the Bright Computing support department if 2nd or 3rd level support is required.

If the Bright Cluster Manager software was purchased directly from Bright Computing, then `support@brightcomputing.com` can be contacted for all levels of support.

Section 11.2 has more details on working with support.

1

Introduction

1.1 What Is Bright Cluster Manager?

Bright Cluster Manager 5.2 is a cluster management application built on top of a major Linux distribution. It is available for:

- Scientific Linux 5 and 6 (`x86_64` only)
- Red Hat Enterprise Linux Server 5 and 6 (`x86_64` only)
- CentOS 5 and 6 (`x86_64` only)
- SUSE Enterprise Server 11 (`x86_64` only)

This chapter introduces some basic features of Bright Cluster Manager and describes a basic cluster in terms of its hardware.

1.2 Cluster Structure

In its most basic form, a cluster running Bright Cluster Manager contains:

- One machine designated as the *head node*
- Several machines designated as *compute nodes*
- One or more (possibly managed) *Ethernet switches*
- One or more *power distribution units* (Optional)

The head node is the most important machine within a cluster because it controls all other devices, such as compute nodes, switches and power distribution units. Furthermore, the head node is also the host that all users (including the administrator) log in to. The head node is the only machine that is connected directly to the external network and is usually the only machine in a cluster that is equipped with a monitor and keyboard. The head node provides several vital services to the rest of the cluster, such as central data storage, workload management, user management, DNS and DHCP service. The head node in a cluster is also frequently referred to as the *master node*.

A cluster typically contains a considerable number of non-head, or *(regular) nodes*, also referred to simply as nodes.

Most of these nodes are *compute nodes*. Compute nodes are the machines that will do the heavy work when a cluster is being used for large

computations. In addition to compute nodes, larger clusters may have other types of nodes as well (e.g. storage nodes and login nodes). Nodes can be easily installed through the (network bootable) node provisioning system that is included with Bright Cluster Manager. Every time a compute node is started, the software installed on its local hard drive is synchronized automatically against a software image which resides on the head node. This ensures that a node can always be brought back to a "known state". The node provisioning system greatly eases compute node administration and makes it trivial to replace an entire node in the event of hardware failure. Software changes need to be carried out only once (in the software image), and can easily be undone. In general, there will rarely be a need to log on to a compute node directly.

In most cases, a cluster has a private internal network, which is usually built from one or multiple managed Gigabit Ethernet switches. The internal network connects all nodes to the head node and to each other. Compute nodes use the internal network for booting, data storage and interprocess communication. In more advanced cluster setups, there may be several dedicated networks. Note that the external network (which could be a university campus network, company network or the Internet) is not normally directly connected to the internal network. Instead, only the head node is connected to the the external network.

Figure 1.1 illustrates a typical cluster network setup.

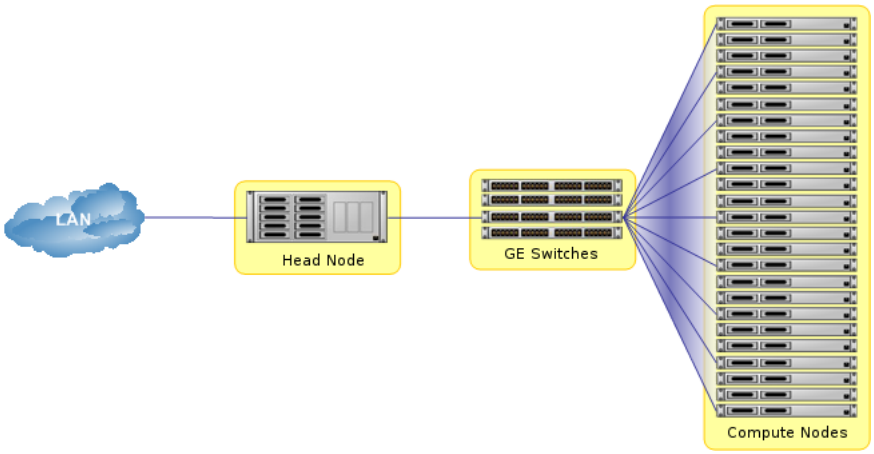

Figure 1.1: Cluster network

Most clusters are equipped with one or more power distribution units. These units supply power to all compute nodes and are also connected to the internal cluster network. The head node in a cluster can use the power control units to switch compute nodes on or off. From the head node, it is straightforward to power on/off a large number of compute nodes with a single command.

1.3 Bright Cluster Manager Administrator And User Environment

Bright Cluster Manager contains several tools and applications to facilitate the administration and monitoring of a cluster. In addition, Bright Cluster Manager aims to provide users with an optimal environment for

developing and running applications that require extensive computational resources.

1.4 Organization Of This Manual

The following chapters of this manual describe all aspects of Bright Cluster Manager from the perspective of a cluster administrator.

Chapter 2 gives step-by-step instructions to installing Bright Cluster Manager on the head node of a cluster. Readers with a cluster that was shipped with Bright Cluster Manager pre-installed may safely skip this chapter.

Chapter 3 introduces the main concepts and tools that play a central role in Bright Cluster Manager, laying down groundwork for the remaining chapters.

Chapter 4 explains how to configure and further set up the cluster after software installation of Bright Cluster Manager on the head node.

Chapter 5 describes how power management within the cluster works.

Chapter 6 explains node provisioning in detail.

Chapter 7 explains how accounts for users and groups are managed.

Chapter 8 explains how workload management is implemented and used.

Chapter 9 demonstrates a number of techniques and tricks for working with software images and keeping images up to date.

Chapter 10 explains how the monitoring features of Bright Cluster Manager can be used.

Chapter 11 summarizes several useful tips and tricks for day to day monitoring.

Chapter 12 describes a number of third party software packages that play a role in Bright Cluster Manager.

Chapter 13 gives details and setup instructions for high availability features provided by Bright Cluster Manager. These can be followed to build a cluster with redundant head nodes.

The appendices generally give supplementary details to the main text.

2

Installing Bright Cluster Manager

This chapter describes the installation of Bright Cluster Manager onto the head node of a cluster. Sections 2.1 and 2.2 list hardware requirements and supported hardware, while section 2.3 gives step-by-step instructions on installing Bright Cluster Manager from a DVD onto a head node.

2.1 Minimal Hardware Requirements

The following are minimal hardware requirements:

2.1.1 Head Node

- Intel Xeon or AMD Opteron CPU (64-bit)
- 2GB RAM
- 80GB diskspace
- 2 Gigabit Ethernet NICs
- DVD drive

2.1.2 Compute Nodes

- Intel Xeon or AMD Opteron CPU (64-bit)
- 1GB RAM (at least 4GB is recommended for diskless nodes)
- 1 Gigabit Ethernet NIC

2.2 Supported Hardware

The following hardware is supported:

2.2.1 Compute Nodes

- SuperMicro
- Cray
- Dell
- IBM

- Asus
- Hewlett Packard

Other brands are unsupported, but are also expected to work.

2.2.2 Ethernet Switches

- HP Procurve
- Nortel
- Cisco
- Dell
- SuperMicro
- Netgear

Other brands are unsupported, but are also expected to work.

2.2.3 Power Distribution Units

- APC (American Power Conversion) Switched Rack PDU

Other brands are unsupported, but are also expected to work.

2.2.4 Management Controllers

- IPMI 1.5/2.0
- HP iLO 1/2/3

2.2.5 InfiniBand

- Mellanox HCAs, and most other InfiniBand HCAs
- Mellanox InfiniBand switches, and most other InfiniBand switches

2.3 Head Node Installation

This section describes the steps in installing a Bright Cluster Manager head node. To start the install, the time in the BIOS of the head node is set to local time and the head node is set to boot from DVD. The head node is then booted from the Bright Cluster Manager DVD.

2.3.1 Welcome Screen

The welcome screen (figure 2.1) displays version and license information. Two installation modes are available, normal mode and express mode. Selecting the express mode installs the head node with the predefined configuration that the DVD was created with. The administrator password automatically set when express mode is selected is: `system`. Clicking on the `Continue` button brings up the Bright Cluster Manager software license screen, described next.

Figure 2.1: Installation welcome screen for Bright Cluster Manager

2.3.2 Software License

The "`Bright Computing Software License`" screen (figure 2.2) explains the applicable terms and conditions that apply to use of the Bright Cluster Manager software.

Accepting the terms and conditions, and clicking on the `Continue` button leads to the `Base Distribution EULA` (End User License Agreement) (figure 2.3).

Accepting the terms and conditions of the base distribution EULA, and clicking on the `Continue` button leads to two possibilities.

1. If express mode was selected earlier, then the installer skips ahead to the `Summary` screen (figure 2.24), where it shows an overview of the predefined installation parameters, and awaits user input to start the install.

2. Otherwise, if normal installation mode was selected earlier, then the "`Kernel Modules`" configuration screen is displayed, described next.

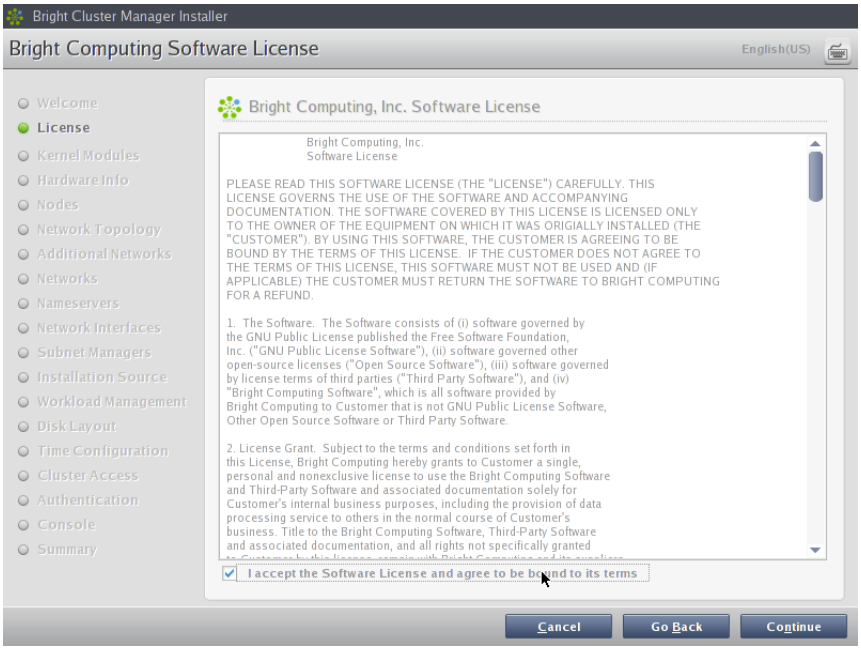

Figure 2.2: Bright Cluster Manager Software License

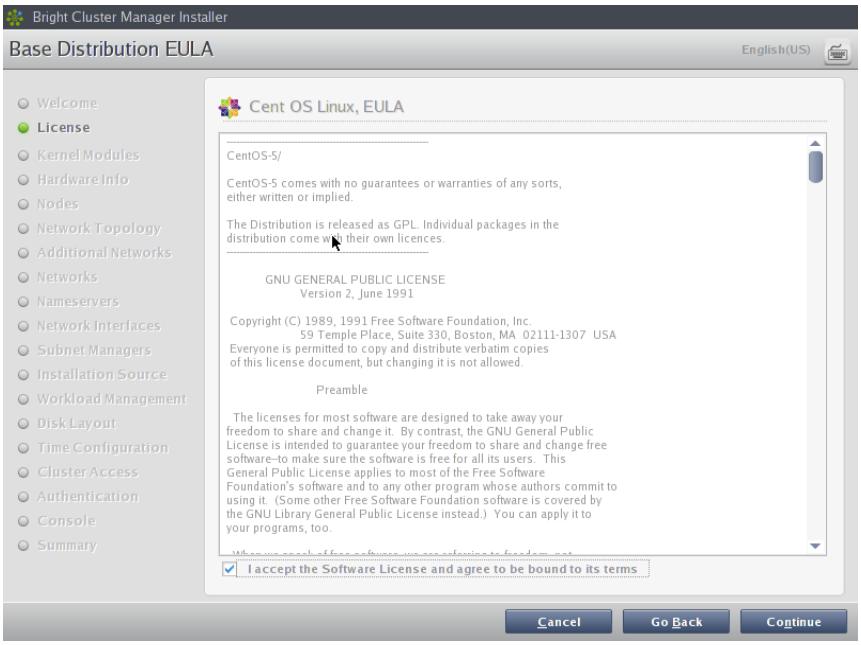

Figure 2.3: Base Distribution End User License Agreement

2.3.3 Kernel Modules Configuration

The `Kernel Modules` screen (figure 2.4) shows the kernel modules recommended for loading based on hardware auto-detection. Clicking the ⊕ button opens an input box for entering the module name and optional module parameters. Clicking the `Add` button in the input box adds the kernel module. The ⊖ button removes a selected module from the list, and the arrow buttons move a kernel module up or down in the list. Kernel module loading order decides the exact name assigned to a device

(e.g. `sda`, `sdb`, `eth0`, `eth1`).

After optionally adding or removing kernel modules, clicking `Continue` leads to the "`Hardware Information`" overview screen, described next.

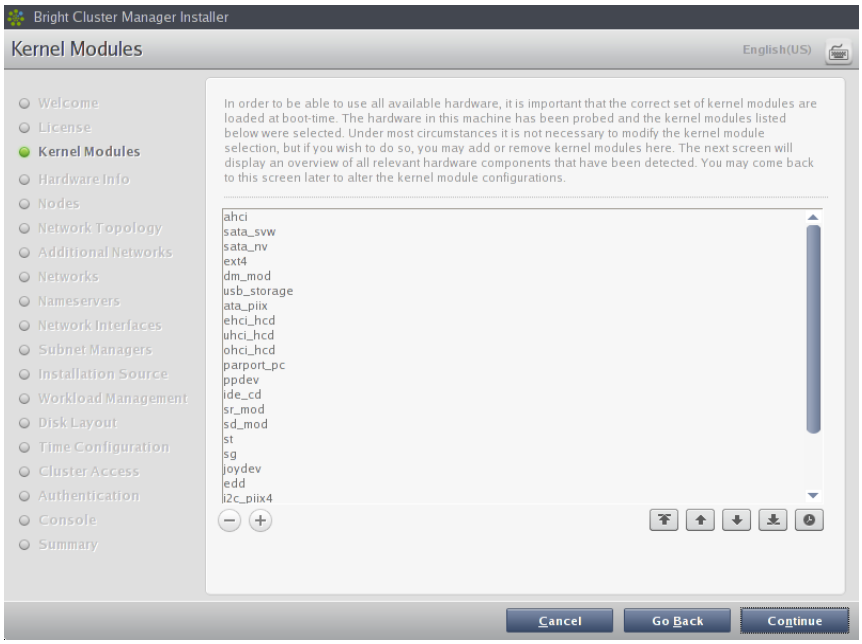

Figure 2.4: Kernel Modules Recommended For Loading After Probing

2.3.4 Hardware Overview

The "`Hardware Information`" screen (figure 2.5) provides an overview of detected hardware depending on the kernel modules that have been loaded. If any hardware is not detected at this stage, the "`Go Back`" button is used to go back to the "`Kernel Modules`" screen (figure 2.4) to add the appropriate modules, and then the "`Hardware Information`" screen is returned to, to see if the hardware has been detected. Clicking `Continue` in this screen leads to the `Nodes` configuration screen, described next.

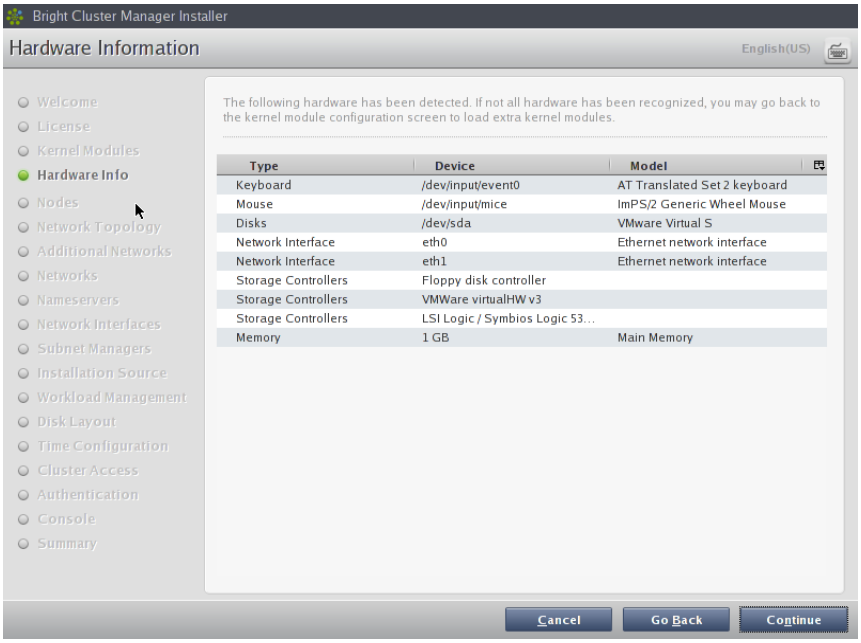

Figure 2.5: Hardware Overview Based On Loaded Kernel Modules

2.3.5 Nodes Configuration

The `Nodes` screen (figure 2.6) configures the number of racks, the number of nodes, the node basename, the number of digits for nodes, and the hardware manufacturer.

The maximum number of digits is 5, to keep the hostname reasonably readable.

The "`Node Hardware Manufacturer`" selection option initializes any monitoring parameters relevant for that manufacturer's hardware. If the manufacturer is not known, then `Other` is selected from the list.

Clicking `Continue` in this screen leads to the "`Network Topology`" selection screen, described next.

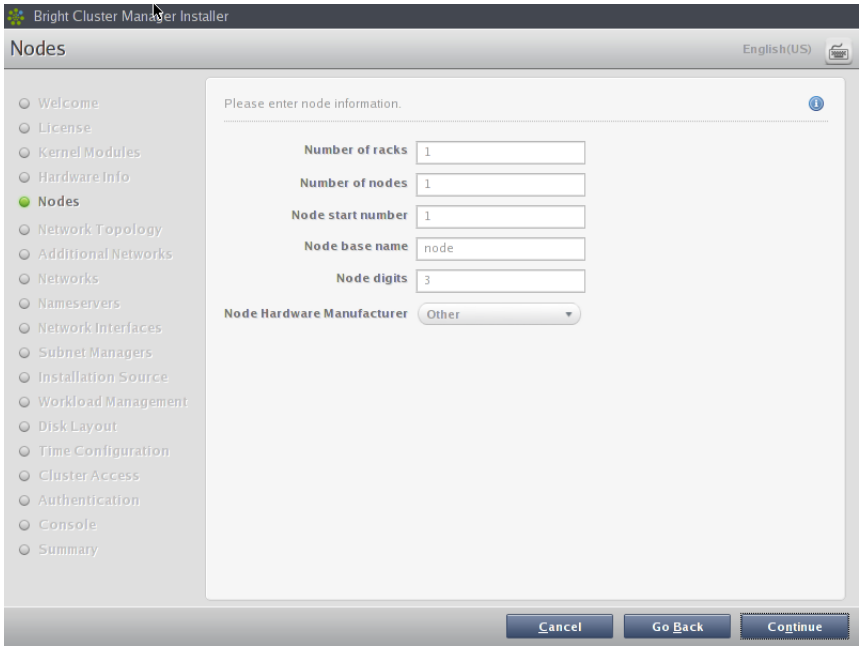

Figure 2.6: Nodes Configuration

2.3.6 Network Topology

The "`Network Topology`" screen allows selection of one of three different network topologies.

A *type 1* network (figure 2.7), with nodes connected on a private internal network. It is the default network setup.

A *type 2* network (figure 2.8), with nodes connected on a public network.

A *type 3* network (figure 2.9), with nodes connected on a routed public network.

Selecting the network topology helps decide the predefined networks on the `Networks` settings screen later (figure 2.11). Clicking `Continue` here leads to the "`Additional Network Configuration`" screen, described next.

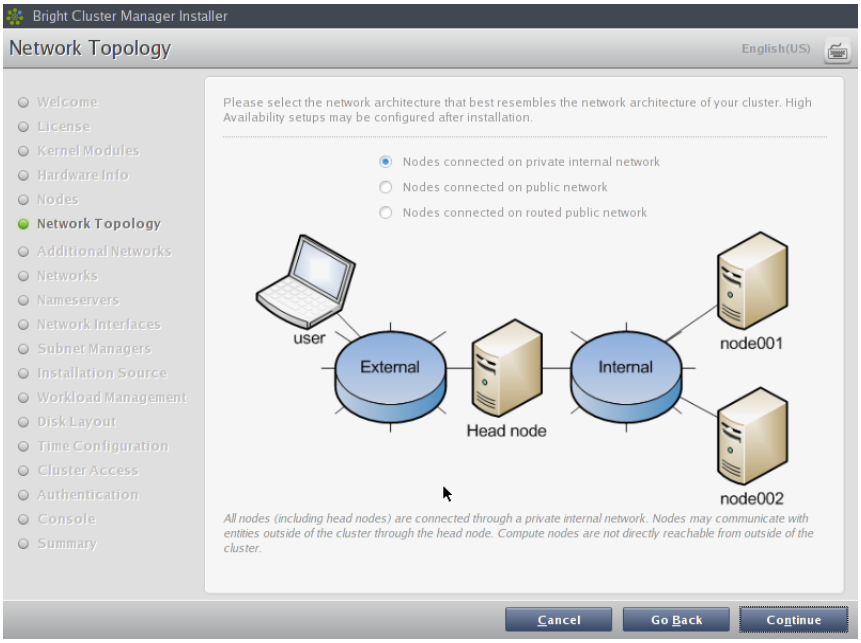

Figure 2.7: Networks Topology: nodes connected on a private internal network

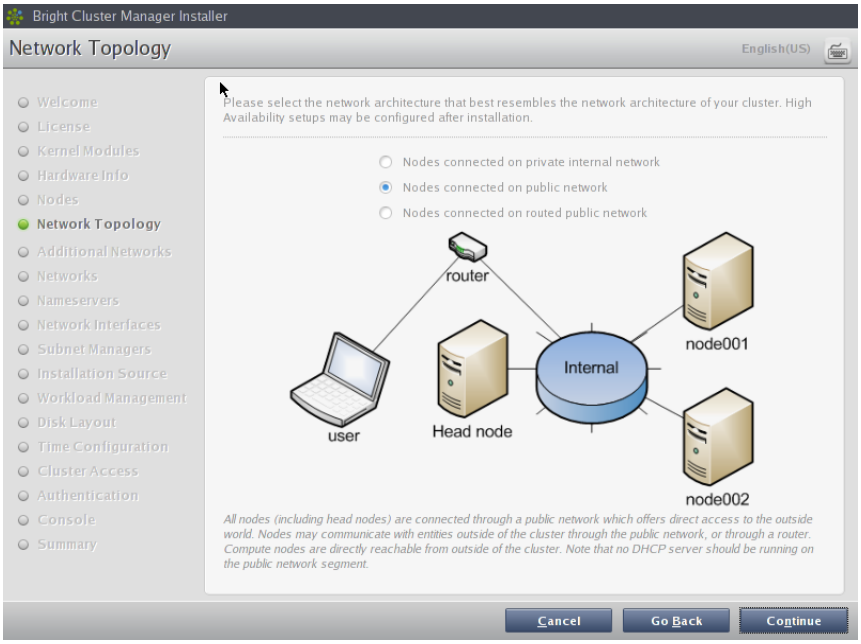

Figure 2.8: Networks Topology: nodes connected on a public network

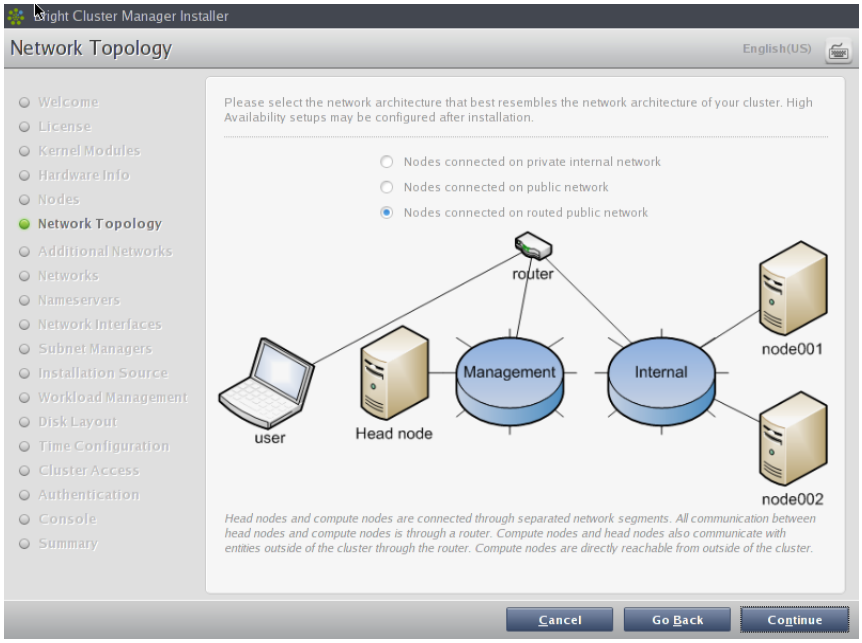

Figure 2.9: Network Topology: nodes connected on a routed public network

2.3.7 Additional Network Configuration

The "`Additional Network Configuration`" screen (figure 2.10) allows InfiniBand and IPMI/iLO networks to be configured. The IPMI password is set to a random value if IPMI is to be used. Retrieving and changing an IPMI password is covered in section 4.4.

Clicking `Continue` in figure 2.10 leads to the `Networks` configuration screen, described next.

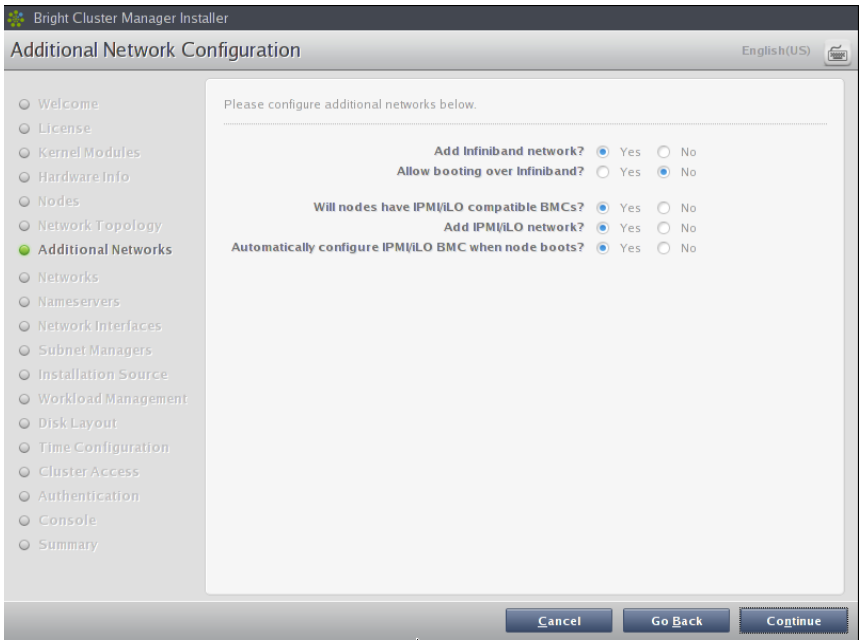

Figure 2.10: Additional Network Configuration

2.3.8 Networks Configuration

The `Networks` configuration screen (figure 2.11) displays the predefined list of networks, based on the selected network topology. IPMI and InfiniBand networks are defined based on selections made in the "`Additional Network Configuration`" screen earlier (figure 2.10).

The parameters of the network interfaces can be configured in this screen.

For a *type 1* setup, an external network and an internal network are always defined.

For a *type 2* setup only an internal network is defined and no external network is defined.

For a *type 3* setup, an internal network and a management network are defined.

Clicking `Continue` in this screen validates all network settings. Invalid settings for any of the defined networks cause an alert to be displayed, explaining the error. A correction is then needed to proceed further.

If all settings are valid, the installation proceeds on to the `Nameservers` screen, described in the next section.

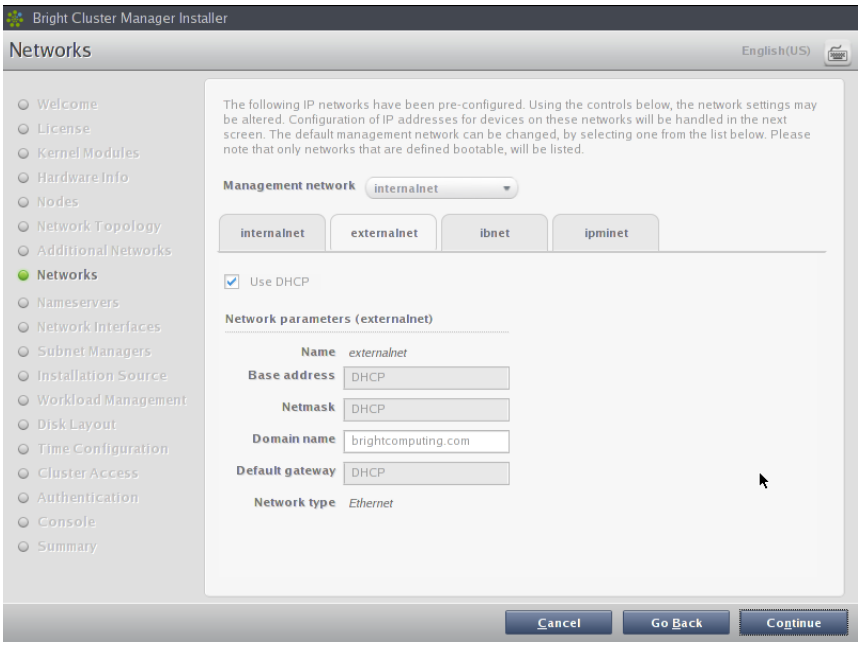

Figure 2.11: Networks Configuration

2.3.9 Nameservers And Search Domains

Search domains and external name servers can be added or removed using the `Nameservers` screen (figure 2.12). Using an external name server is recommended. Clicking on `Continue` leads to the "`Network Interfaces`" configuration screen, described next.

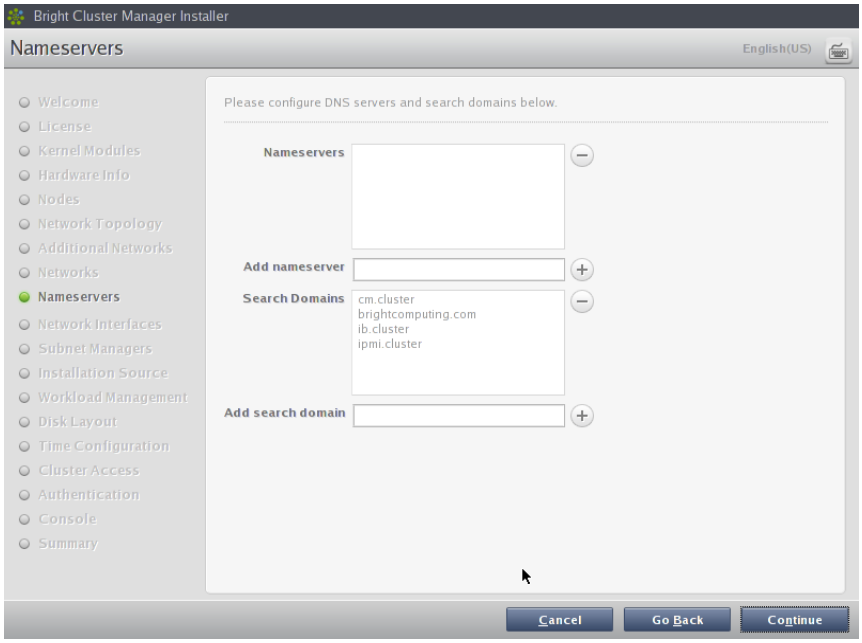

Figure 2.12: Nameservers and search domains

2.3.10 Network Interfaces Configuration

The "`Network Interfaces`" screens (figures 2.13 and 2.14) show the list of network interfaces that have been predefined for *type 1* and *type 3* setups respectively. Each screen has a network configuration section for the head node and for the regular nodes.

For node network interfaces, the IP offset can be modified. The offset is used to calculate the IP address assigned to the interface on the selected network. For example, a different offset might be desirable when no IPMI network has been defined, but nodes do have IPMI. In this case the `BOOTIF` and `ipmi0` interfaces have IP addresses assigned on the same network, but if a different offset is entered for the `ipmi0` interface, then the assigned IP address starts from the offset specified.

A different network can be selected for each interface using the drop-down box in the `Network` column. Selecting `Unassigned` disables a network interface.

If the corresponding network settings are changed (e.g., base address of the network) the IP address of the head node interface needs to be modified accordingly. If IP address settings are invalid, an alert is displayed, explaining the error.

Clicking `Continue` on a "`Network Interfaces`" screen validates IP address settings for all node interfaces, and if all setting are correct, and if InfiniBand networks have been defined, leads to the "`Subnet Managers`" screen (figure 2.15), described in the next section. If no InfiniBand networks are defined, or if InfiniBand networks have not been enabled on the networks settings screen, then clicking `Continue` instead leads to the `CD/DVD ROMs` selection screen (figure 2.16).

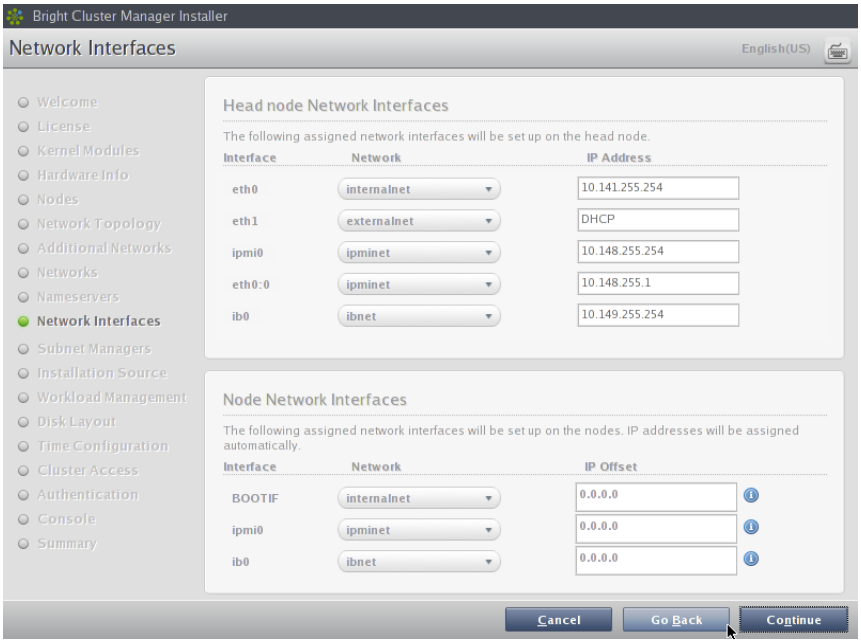

Figure 2.13: Network Interface Configuration: *type 1*

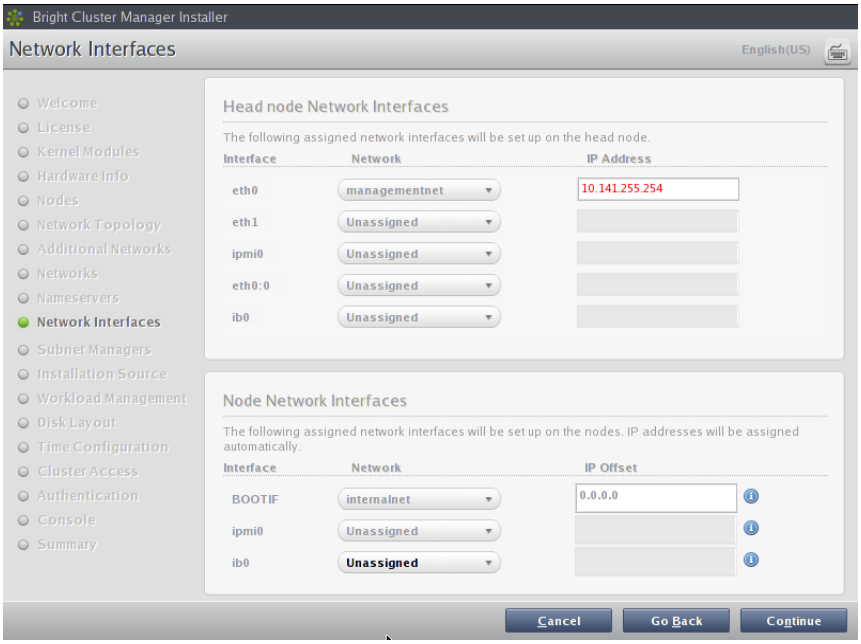

Figure 2.14: Network Interface Configuration: *type 3*

2.3.11 Select Subnet Managers

The "`Subnet Managers`" screen in figure 2.15 is only displayed if an InfiniBand network was defined, and lists all the nodes that can run the InfiniBand subnet manager. The nodes assigned the role of a subnet manager are ticked, and the `Continue` button is clicked to go on to the "`CD/DVD ROMs`" selection screen, described next.

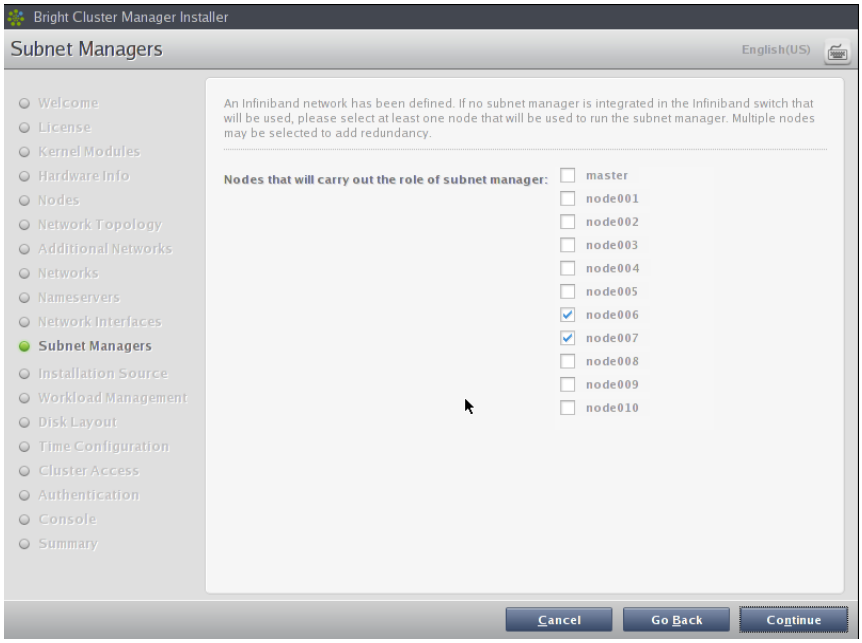

Figure 2.15: Subnet Manager Nodes

2.3.12 Select CD/DVD-ROM

The "`CD/DVD ROMs`" screen in figure 2.16 lists all detected CD/DVD-ROM devices. If multiple drives are found, then the drive with the Bright Cluster Manager DVD needs to be selected by the administrator. Clicking on `Continue` then brings up the "`Workload Management`" setup screen, described next.

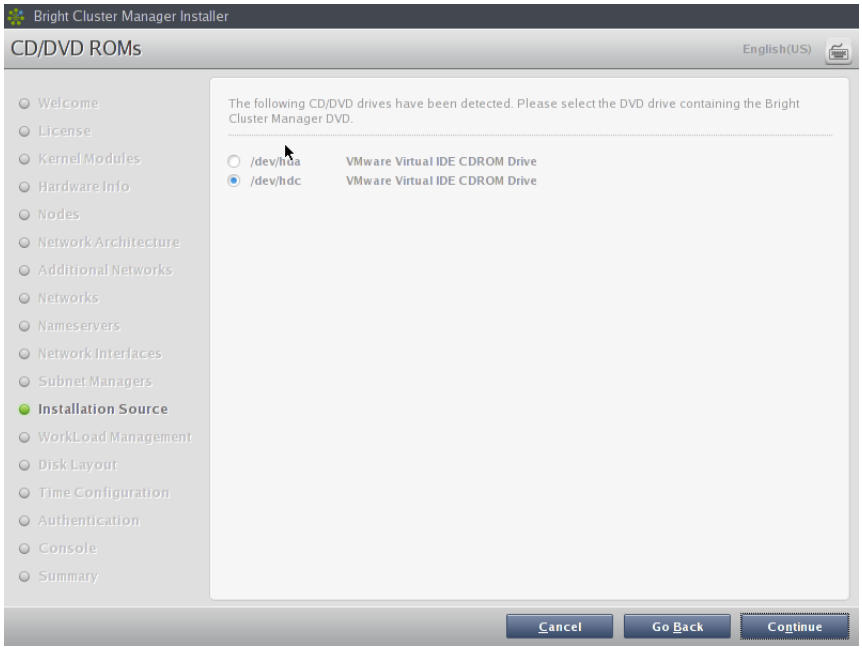

Figure 2.16: DVD Selection

2.3.13 Workload Management Configuration

The "`Workload Management`" configuration screen (figure 2.17) allows selection from a list of supported workload managers. A workload management system is highly recommended to run multiple compute jobs on a cluster.

To prevent a workload management system from being set up, select `None`. If a workload management system is selected, then the number of slots per node can be set, otherwise the slots setting is ignored. If no changes are made, then the number of slots defaults to the CPU count on the head node.

The head node can also be selected for use as a compute node, which can be a sensible choice on small clusters. The setting is ignored if no workload management system is selected.

Clicking `Continue` on this screen leads to the "`Disk Partitioning and Layouts`" screen, described next.

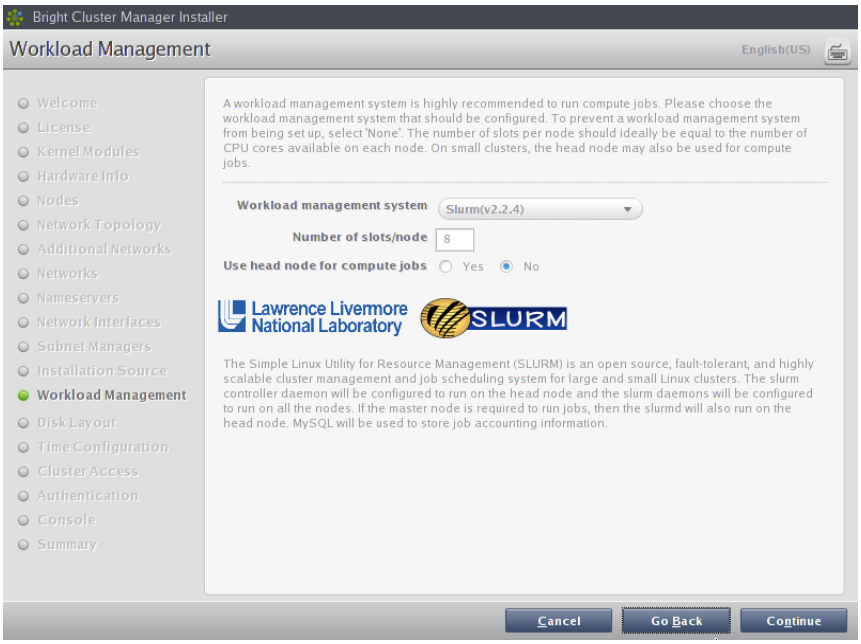

Figure 2.17: Workload Management Setup

2.3.14 Disk Partitioning And Layouts

The "`Disk Partitioning and Layouts`" configuration screen (figure 2.18):

- selects the drive that the cluster manager is installed onto on the head node.
- sets the disk partitioning layout for the head node and regular nodes with the two options: "`Head node disk layout`" and "`Node disk layout`".
  - A partitioning layout other than the default can be selected for installation from the drop-down boxes.
  - A text editor pops up when an option's edit button is clicked (figure 2.19). This can be used to view and change values. The `Save` and `Reset` buttons are enabled on editing, and save or

undo the text editor changes. Once saved, the changes cannot be reverted automatically in the text editor, but must be done manually.

– A partitioning layout is the only installation setting that cannot easily be changed after the completion (section 2.3.20) of installation. It should therefore be decided upon with care.

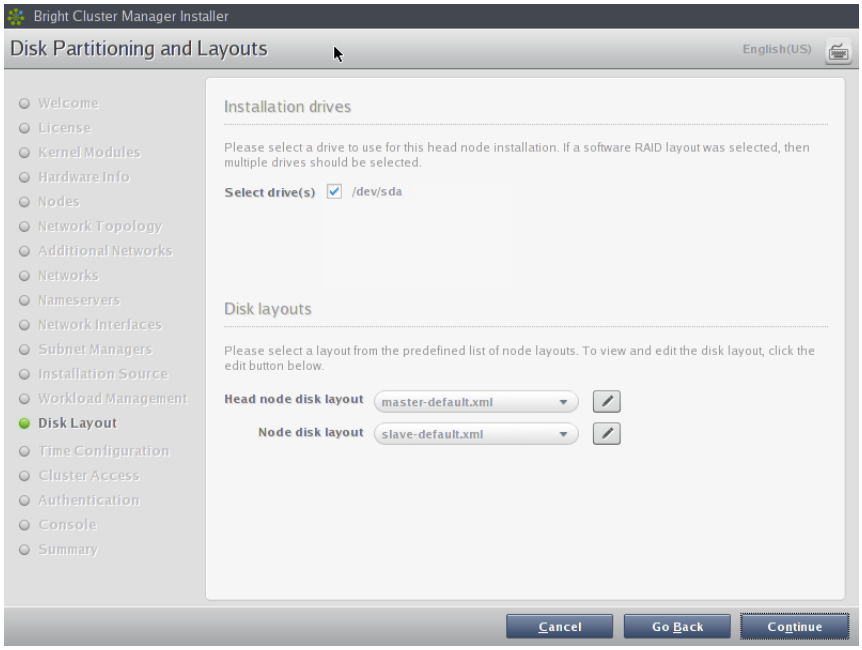

Figure 2.18: Disk Partitioning And Layouts

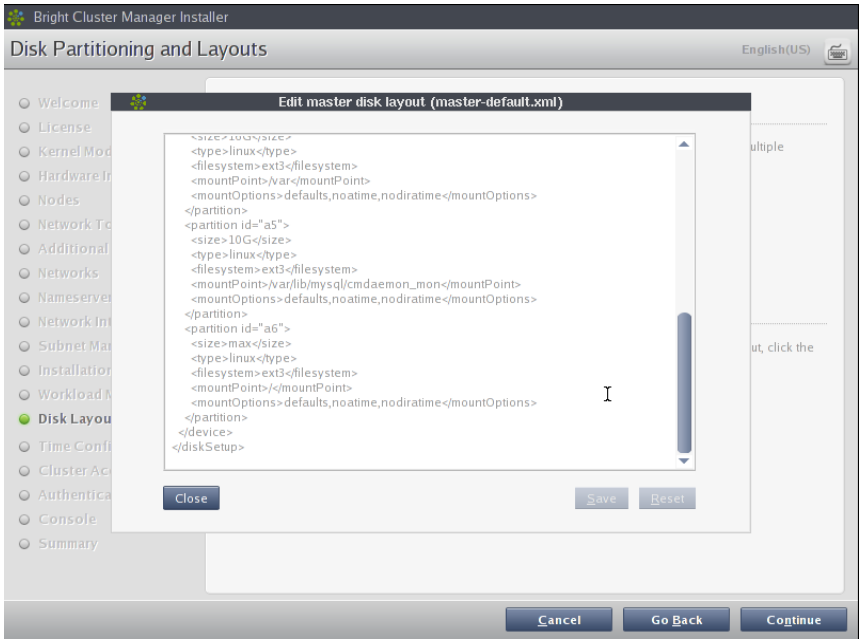

Figure 2.19: Edit Head Node Disk Partitioning

Clicking Continue on this screen leads to the "Time Configuration" screen, described next.

2.3.15 Time Configuration

The "Time Configuration" screen (figure 2.20) displays a predefined list of time servers. Timeservers can be removed by selecting a time server from the list and clicking the ⊖ button. Additional time servers can be added by entering the name of the time server and clicking the ⊕ button. A timezone can be selected from the drop-down box if the default is incorrect. Clicking Continue leads to the "Cluster Access" screen, described next.

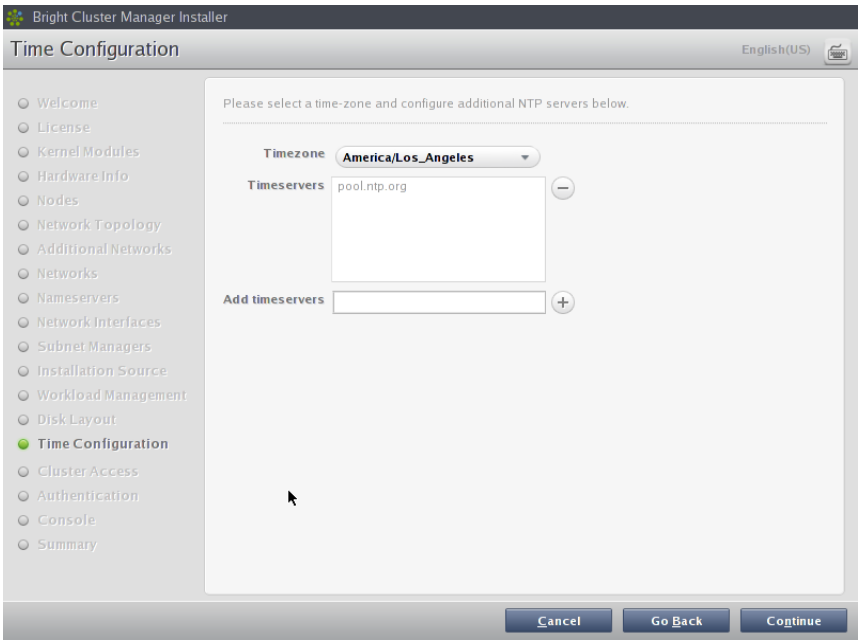

Figure 2.20: Time Configuration

2.3.16 Cluster Access

The "Cluster Access" screen (figure 2.21) sets the existence of a cluster management web portal service, and also sets network access to several services.

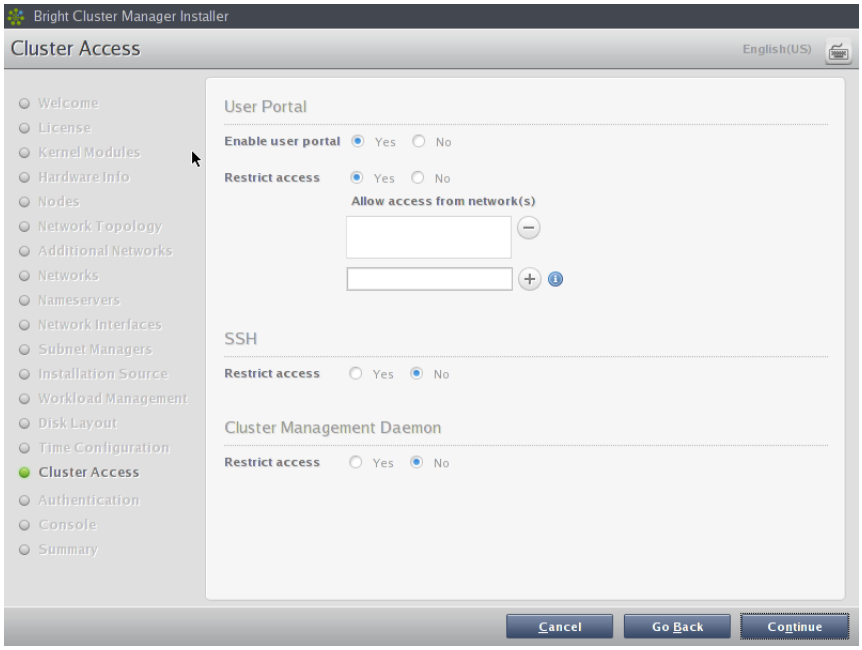

Figure 2.21: Cluster Access

These services are the web portal, ssh, and the cluster management daemon.

If restricting network access for a service is chosen, then an editable list of networks that may access the service is displayed. By default the list has no members. The screen will not move on to the next screen until the list contains at least one CIDR-format network IP address.

If the conditions for this screen are satisfied, then clicking `Continue` leads to the `Authentication` screen, described next.

2.3.17 Authentication

The `Authentication` screen (figure 2.22) requires the password to be set twice for the cluster administrator. The cluster name and the head node hostname can also be set in this screen. Clicking `Continue` validates the passwords that have been entered, and if successful, leads to the `Console` screen, described next.

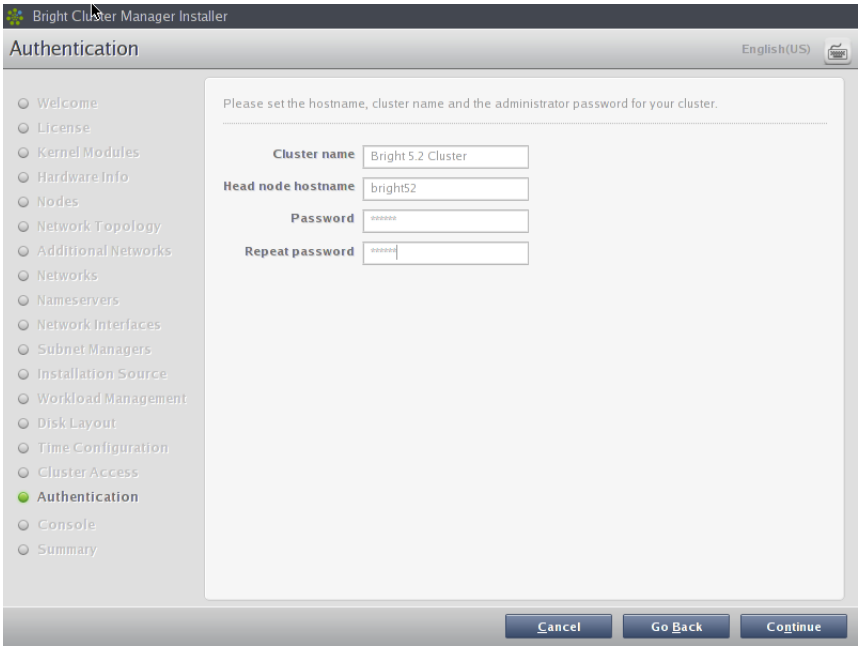

Figure 2.22: Authentication

2.3.18 Console

The `Console` screen (figure 2.23) allows selection of a graphical mode or a text console mode for when the head node or regular nodes boot. Clicking `Continue` leads to the `Summary` screen, described next.

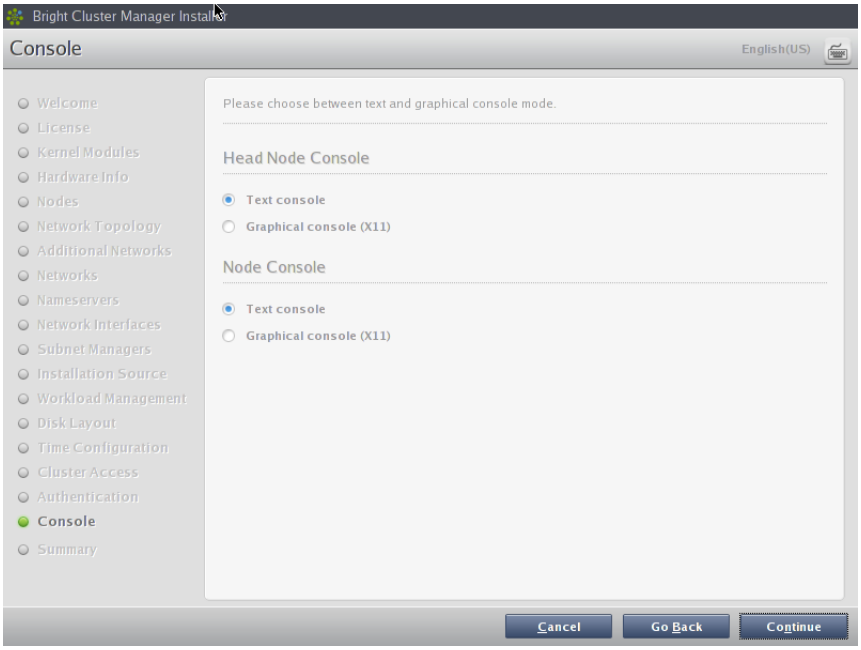

Figure 2.23: Console

2.3.19 Summary

The `Summary` screen (figure 2.24), summarizes some of the installation settings and parameters configured during the previous stages. If the express mode installation was chosen, then it summarizes the predefined

settings and parameters. Changes to the values on this screen are made by navigating to previous screens and correcting the values there.

When the summary screen displays the right values, clicking on the `Start` button leads to the "`Installation Progress`" screen, described next.

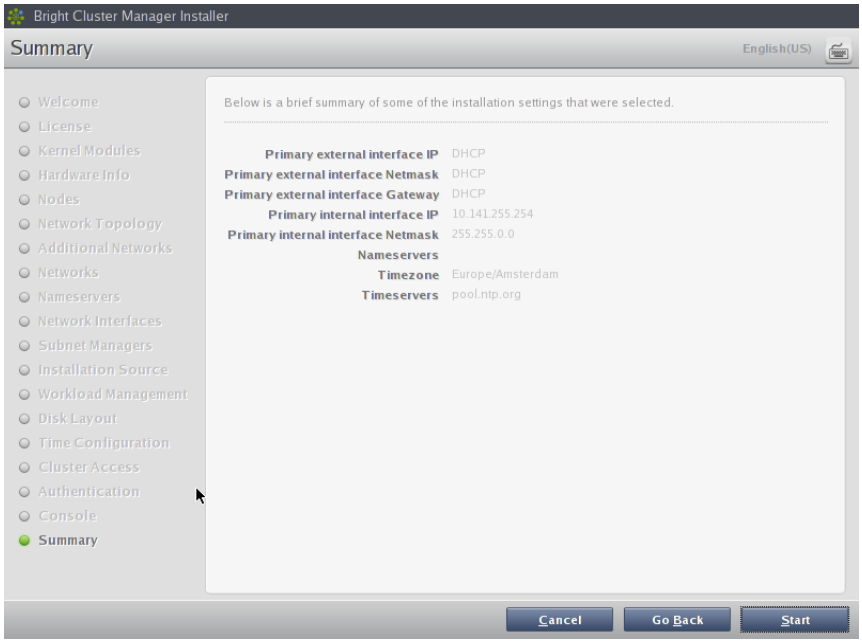

Figure 2.24: Summary of Installation Settings

2.3.20 Installation

The "`Installation Progress`" screen (figure 2.25) shows the progress of the installation. It is not possible to navigate back to previous screens once the installation has begun. When the installation is complete (figure 2.26), the installation log can be viewed in detail by clicking on "`Install Log`".

The `Reboot` button restarts the machine. The BIOS boot order may need changing or the DVD should be removed, in order to boot from the hard drive on which Bright Cluster Manager has been installed.

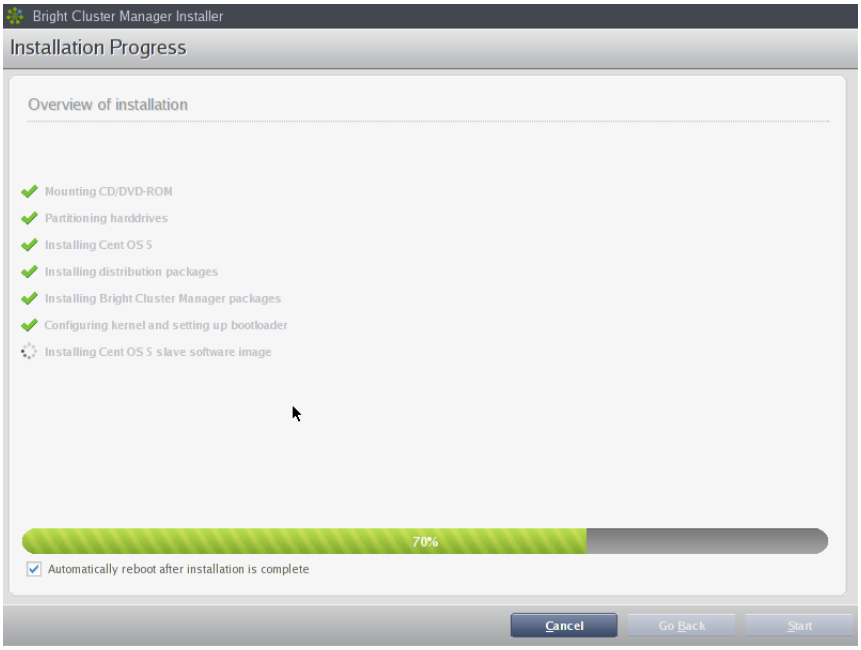

Figure 2.25: Installation Progress

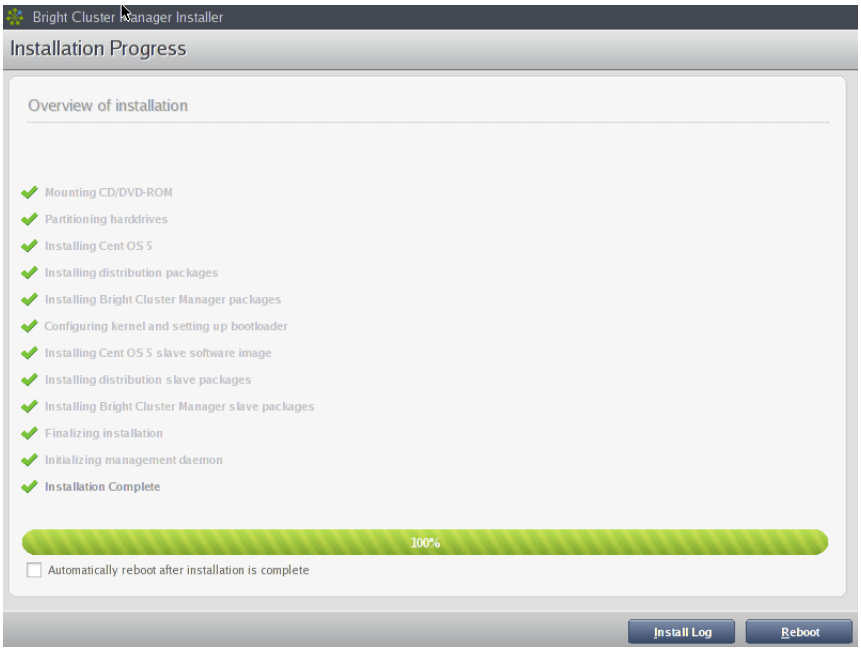

Figure 2.26: Installation Completed

After rebooting, the system starts and presents a login prompt. After logging in as `root` using the password that was set during the installation procedure, the system is ready to be configured. If express installation mode was chosen earlier as the install method, then the password is preset to `system`.

Next, in Chapter 3, some of the tools and concepts that play a central role in Bright Cluster Manager are introduced. Chapter 4 then explains how to configure and further set up the cluster.

3

Cluster Management With Bright Cluster Manager

This chapter introduces cluster management with Bright Cluster Manager. A cluster running Bright Cluster Manager exports a cluster management interface to the outside world, which can be used by any application designed to communicate with the cluster.

Section 3.1 introduces a number of concepts which are key to cluster management using Bright Cluster Manager.

Section 3.2 gives a short introduction on how the modules environment can be used by administrators. The modules environment provides facilities to control aspects of a users' interactive sessions and also the environment used by compute jobs.

Section 3.3 introduces how authentication to the cluster management infrastructure works and how it is used.

Section 3.4 and section 3.5 introduce the cluster management GUI (`cmgui`) and cluster management shell (`cmsh`) respectively. These are the primary applications that interact with the cluster through its management infrastructure.

Section 3.6 describes the basics of the cluster management daemon, CMDaemon, running on all nodes of the cluster.

3.1 Concepts

In this section some concepts central to cluster management with Bright Cluster Manager are introduced.

3.1.1 Devices

A *device* in the Bright Cluster Manager cluster management infrastructure represents physical hardware components of a cluster. A device can be any of the following types:

- Head Node
- Node
- Virtual SMP Node[1]

[1] a hardware component because it is fundamentally made up of several physical nodes, even though it is seen by the end user as virtual nodes

- GPU Unit
- Ethernet Switch
- InfiniBand Switch
- Myrinet Switch
- Power Distribution Unit
- Rack Sensor Kit
- Generic Device

A device can have a number of properties (e.g. rack position, hostname, switch port) which can be set in order to configure the device. Using the cluster management infrastructure, operations (e.g. power on) may be performed on a device. The property changes and operations that can be performed on a device depend on the type of device. For example, it is possible to mount a new filesystem to a node, but not to an Ethernet switch.

Every device that is present in the cluster management infrastructure has a device state associated with it. The table below describes the most important states for devices:

Device State	Description
`UP`	device is reachable
`DOWN`	device is not reachable
`CLOSED`	device has been taken offline by administrator

There are a number of other states which are described in detail in section 6.5.

`DOWN` and `CLOSED` states have an important difference. In the case of `DOWN`, the device was intended to be available, but instead is down. In the case of `CLOSED`, the device is intentionally unavailable.

3.1.2 Software Images

A *software image* is a blueprint for the contents of the local file-systems on a regular node. In practice, a software image is a directory on the head node containing a full Linux file-system. When an regular node boots, the node provisioning system (Chapter 6) sets up the node with a copy of the software image, which by default is called `default-image`.

Once the node is fully booted, it is possible to instruct the node to re-synchronize its local filesystems with the software image. This procedure can be used to distribute changes to the software image without rebooting nodes (section 6.6.2).

Software images can be changed using regular Linux tools and commands (such as `rpm` and `chroot`). More details on making changes to software images and doing image package management can be found in Chapter 9.

3.1.3 Node Categories

The collection of settings in Bright Cluster Manager that can apply to a node is called the configuration of the node. The administrator usually configures nodes using the `cmgui` (section 3.4) and `cmsh` (section 3.5) front end tools, and the configurations are managed internally with a database.

A *node category* is a group of regular nodes that share the same configuration. Node categories allow efficiency, allowing an administrator to:

- configure a large group of nodes at once. For example, to set up a group of nodes with a particular disk layout.
- operate on a large group of nodes at once. For example, to carry out a reboot on an entire category.

A node is in exactly one category at all times, which is `default` by default.

Nodes are typically divided into node categories based on the hardware specifications of a node or based on the task that a node is to perform. Whether or not a number of nodes should be placed in a separate category depends mainly on whether the configuration—for example: monitoring setup, disk layout, role assignment—for these nodes differs from the rest of the nodes.

A node inherits values from the category it is in. Each value is treated as the default property value for a node, and is overruled by specifying the node property value for the node.

One configuration property value of a node category is its software image (section 3.1.2). However, there is no requirement for a one-to-one correspondence between node categories and software images. Therefore multiple node categories may use the same software image, and multiple images may be used in the same node category.

By default, all nodes are placed in the `default` category. Alternative categories can be created and used at will, such as:

Example

Node Category	Description
`nodes-ib`	nodes with InfiniBand capabilities
`nodes-highmem`	nodes with extra memory
`login`	login nodes
`storage`	storage nodes

3.1.4 Node Groups

A *node group* consists of nodes that have been grouped together for convenience. The group can consist of any mix of all kinds of nodes, irrespective of whether they are head nodes or regular nodes, and irrespective of what (if any) category they are in. A node may be in 0 or more node groups at one time. I.e.: a node may belong to many node groups.

Node groups are used mainly for carrying out operations on an entire group of nodes at a time. Since the nodes inside a node group do not necessarily share the same configuration, configuration changes cannot be carried out using node groups.

Example

Node Group	Members
broken	node087, node783, node917
headnodes	mycluster-m1, mycluster-m2
rack5	node212..node254
top	node084, node126, node168, node210

3.1.5 Roles

A *role* is a task that can be performed by a node. By assigning a certain role to a node, an administrator activates the functionality that the role represents on this node. For example, a node can be turned into provisioning node, or a storage node by assigning the corresponding roles to the node.

Roles can be assigned to individual nodes or to node categories. When a role has been assigned to a node category, it is implicitly assigned to all nodes inside of the category.

Some roles allow parameters to be set that influence the behavior of the role. For example, the `SLURM Client Role` (which turns a node into a SLURM client) uses parameters to control how the node is configured within SLURM in terms of queues and the number of GPUs.

When a role has been assigned to a node category with a certain set of parameters, it is possible to override the parameters for a node inside the category. This can be done by assigning the role again to the individual node with a different set of parameters. Roles that have been assigned to nodes override roles that have been assigned to a node category.

Examples of role assignment are given in sections 6.2.2 and 6.2.3.

3.2 Modules Environment

The *modules environment* is a third-party software (section 12.1) that allows users to modify their shell environment using pre-defined *modules*. A module may, for example, configure the user's shell to run a certain version of an application.

Details of the modules environment from a user perspective are discussed in the *User Manual*. However some aspects of it are relevant for administrators and are therefore discussed here.

3.2.1 Adding And Removing Modules

Modules may be loaded and unloaded, and also be combined for greater flexibility.

Modules currently installed are listed with:

```
module list
```

The modules available for loading are listed with:

```
module avail
```

Loading and removing specific modules is done with `module load` and `module remove`, using this format:

```
module load <MODULENAME1> [<MODULENAME2> ...]
```

Loading the `shared` module (section 3.2.2), the `gcc` compiler, the `openmpi` parallel library, and the `gotoblas/opteron` 64-bit library, then allows an MPI application to be compiled with Opteron GOTOBLAS optimizations.

Example

```
module add shared
module add gcc
module add openmpi/gcc
module add gotoblas/opteron/64
mpicc -o myapp myapp.c
```

Specifying version numbers explicitly is typically only necessary when multiple versions of an application are installed and available. When there is no ambiguity, module names without a further path specification may be used.

3.2.2 Using Local And Shared Modules

Applications and their associated modules are divided into *local* and *shared* groups. Local applications are installed on the local file-system, whereas shared applications reside on a shared (i.e. imported) file-system.

It is recommended that the `shared` module be loaded by default for ordinary users. Loading it gives access to the modules belonging to shared applications, and allows the `module avail` command to show these extra modules.

Loading the `shared` module automatically for `root` is not recommended on a cluster where shared storage is not on the head node itself, because `root` logins could be obstructed if this storage is unavailable.

On clusters without external shared storage, `root` can safely load the `shared` module automatically at login. This can be done by running the following command as `root`:

```
module initadd shared
```

Other modules can also be set to load automatically by the user at login by using "`module initadd`" with the full path specification. With the `initadd` option, individual users can customize their own default modules environment.

Modules can be combined in *meta-modules*. By default, the `default-environment` meta-module exists, which allows the loading of several modules at once by a user. Cluster administrators are encouraged to customize the `default-environment` meta-module to set up a recommended environment for their users. The `default-environment` meta-module is empty by default.

3.2.3 Setting Up A Default Environment For All Users

The administrator can set up a default modules environment for all users.

To illustrate this, the following case may be considered, where all users currently have just the following modules:

Example

```
[fred@bright52 ~]$ module list
Currently Loaded Modulefiles:
  1) null        2) shared      3) gcc/4.4.6
```

Two ways in which the Torque and Maui modules can then be set up by the administrator as a default for all users are:

- Loading up the required modules in the standard user skeleton file for `.bashrc` at `/etc/skel/.bashrc` (some output omitted):

  Example

  ```
  [root@bright52 ~]# cat /etc/skel/.bashrc
  ...
  # User specific aliases and functions
  module load torque
  module load maui
  ```

- Alternatively, if `/cm/shared/modulefiles/default-environment` is specified as:

  Example

  ```
  [root@bright52 ~]# cat /cm/shared/modulefiles/default-environment
  #%Module1.0#####################################################
  ## default modulefile
  ##
  proc ModulesHelp { } {
          puts stderr "\tLoads default environment modules for thi\
  s cluster"
  }

  module-whatis   "adds default environment modules"

  # Add any modules here that should be added by when a user loads\
   the 'default-environment' module
  module add torque maui
  ```

  then the last lines of `/etc/skel/.bashrc` can be specified as this instead:

  Example

  ```
  [root@bright52 ~]# cat /etc/skel
  ...
  # User specific aliases and functions
  module load default-environment
  ```

More details on the modules environment from an administrator's perspective are given in section 12.1.

3.3 Authentication

3.3.1 Changing Administrative Passwords On The Cluster

How to set up or change regular user passwords is not discussed here, but in Chapter 7 on user management.

Amongst the administrative passwords associated with the cluster are:

1. **The root password of the head node:** This allows a root login to the head node.
2. **The root password of the software images:** This allows a root login to a regular node, and is stored in the image file.
3. **The root password of the node-installer:** This allows a root login to the node when the node-installer, a stripped-down operating system, is running. The node-installer stage prepares the node for the final operating system when the node is booting up. Section 6.4 discusses the node-installer in more detail.
4. **The root password of mysql:** This allows a root login to the mysql server.
5. **The administrator certificate password:** This decrypts the `/root/admin.pfx` file so that the administrator certificate can be presented to `cmdaemon` when administrator tasks require running. Section 3.3.2 discusses certificates in more detail.

To avoid having to remember the disparate ways in which to change these 5 passwords, the `cm-change-passwd` command runs a dialog prompting the administrator on which of them, if any, should be changed, as in the following example:

```
[root@bright52 ~]# cm-change-passwd
With this utility you can easily change the following passwords:
  * root password of head node
  * root password of slave images
  * root password of node-installer
  * root password of mysql
  * administrator certificate for use with cmgui (/root/admin.pfx)

Note: if this cluster has a high-availability setup with 2 head nodes,
      be sure to run this script on both head nodes.

Change password for root on head node? [y/N]: y
Changing password for root on head node.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

Change password for root in default-image [y/N]: y
Changing password for root in default-image.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
```

```
passwd: all authentication tokens updated successfully.

Change password for root in node-installer? [y/N]: y
Changing password for root in node-installer.
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

Change password for MYSQL root user? [y/N]: y
Changing password for MYSQL root user.
Old password:
New password:
Re-enter new password:

Change password for admin certificate file? [y/N]: y
Enter old password:
Enter new password:
Verify new password:
Password updated
```

3.3.2 Certificates

While a Bright Cluster Manager cluster accepts ordinary `ssh` based logins for cluster usage, the cluster management infrastructure requires public key authentication using `X509v3` certificates. Public key authentication using `X509v3` certificates means in practice that the person authenticating to the cluster management infrastructure must present their certificate (i.e the public key) and in addition must have access to the private key that corresponds to the certificate. There are two main file formats in which certificates and private keys are stored:

- **PEM**: In this, the certificate and private key are stored as plain text in two separate PEM-encoded files.
- **PFX** (also known as **PKCS12**): In this, the certificate and private key are stored in one encrypted file.

Although both formats are supported, the PFX format is preferred since it is more convenient (a single file instead of two files) and allows the private key data to be encrypted conveniently with a password.

By default, one administrator certificate is created to interact with the cluster management infrastructure. The certificate and corresponding private key can be found on a newly installed Bright Cluster Manager cluster in both PFX and PEM format in the following locations:

```
/root/.cm/cmgui/admin.pfx

/root/.cm/cmsh/admin.pem
/root/.cm/cmsh/admin.key
```

The administrator password provided during Bright Cluster Manager installation encrypts the `admin.pfx` file generated as part of the installation. The same password is also used as the initial root password of all nodes, as well as for the other passwords discussed in section 3.3.1.

The GUI utility cmgui (section 3.4) connects to the head node if the user types in the password to the admin.pfx file. If the root login password to head node is changed, typically by typing the unix passwd command in the root shell of the node, then the administrator PFX password, remains unchanged unless it, too, is changed explicitly.

The password of the PFX file can be changed with the passwdpfx utility. This is besides the cm-change-passwd utility discussed in section 3.3.1. The passwdpfx utility is part of cmd, a module that includes CMDaemon and associated utilities (section 3.6):

```
[root@mycluster ~]# module load cmd
[root@mycluster ~]# passwdpfx
Enter old password: ******
Enter new password: *******
Verify new password: *******
Password updated
[root@mycluster ~]#
```

If the admin.pfx password is forgotten, then a new admin.pfx certificate can be created using a CMDaemon option:

```
[root@mycluster ~]# service cmd stop
[root@mycluster ~]# cmd -c secretpa55word
[root@mycluster ~]# service cmd start
```

3.3.3 Profiles

Certificates that authenticate to the cluster management infrastructure contain a *profile*.

A profile determines which cluster management operations the certificate holder may perform. The administrator certificate is created with the admin profile, which is a built-in profile that allows all cluster management operations to be performed. In this sense it is similar to the root account on unix systems. Other certificates may be created with different profiles giving certificate owners access to a pre-defined subset of the cluster management functionality (section 7.5).

3.4 Cluster Management GUI

This section introduces the basics of the cluster management GUI (cmgui). This is the graphical interface to cluster management in Bright Cluster Manager. It may be run on the head node or on a login node of the cluster using X11-forwarding:

Example

```
user@desktop:~> ssh -X root@mycluster cmgui
```

However, typically it is installed and run on the administrator's desktop computer. This saves user-discernable lag time if the user is hundreds of kilometers away from the head node.

3.4.1 Installing Cluster Management GUI

To install cmgui on a desktop computer running Linux or Windows, the installation package must be downloaded first. These are available on any Bright Cluster Manager cluster in the directory:

```
/cm/shared/apps/cmgui/dist
```

Installation packages are available for Linux and for Windows XP/Vista, and a MacOS X version will be available in the future.

On a Windows desktop, `cmgui` is installed by running the installer and following the installation procedure. After the installation, `cmgui` is started through the Start menu or through the desktop shortcut.

For a Linux desktop, `cmgui` is installed by:

- untarring the `tar.bz2` file
- reading the accompanying README file to determine what packages are required for the build to complete successfully
- installing any required packages
- compiling the `cmgui` executable with `build.sh`

After building the `cmgui` executable, it can be run from the `cmgui` directory.

Example

```
user@desktop:~> tar -xjf cmgui-5.2-r2510-src.tar.bz2
user@desktop:~> cd cmgui-5.2-r2510
user@desktop:~/cmgui-5.2-r2510> ./build.sh
[...]
user@desktop:~/cmgui-5.2-r2510> cd cmgui
user@desktop:~/cmgui-5.2-r2510/cmgui> ./cmgui
```

If running `cmgui` reports unresolved symbols, then additional packages from the Linux distribution need to be installed, and recompilation done.

At least the following software libraries must be installed in order to run `cmgui`:

- OpenSSL library
- GTK library
- GLib library
- Boost library (at least the `thread` and `signals` components)

3.4.2 Connecting To A Cluster

As explained in section 3.3.2, a certificate and private key are required to connect to the cluster management infrastructure. Both are available when running `cmgui` on the cluster. However, before making the initial connection from a desktop computer running `cmgui`, a PFX file containing both the certificate and private key must be copied from the cluster and stored in a secure location on the local filesystem.

Example

```
user@desktop:~> mkdir ~/cmgui-keys
user@desktop:~> chmod 700 ~/cmgui-keys
user@desktop:~> scp root@mycluster:/root/.cm/cmgui/admin.pfx ~/cmgui-ke\
ys/mycluster-admin.pfx
```

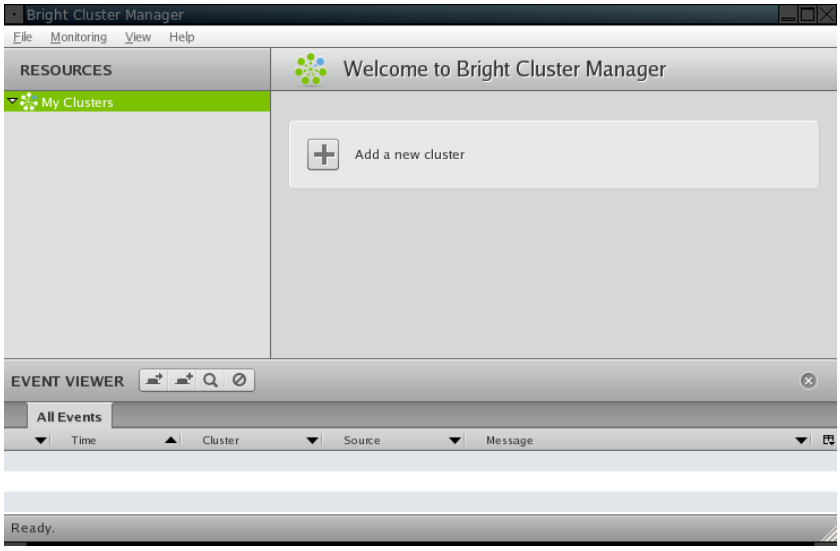

Figure 3.1: Cluster Management GUI welcome screen

When `cmgui` is started for the first time, the welcome screen (figure 3.1) is displayed. To configure `cmgui` for connections to a new Bright Cluster Manager cluster, the cluster is added to `cmgui` by clicking the ⊞ button in the welcome screen. Figure 3.2 shows the dialog window in which the connection parameters can be entered.

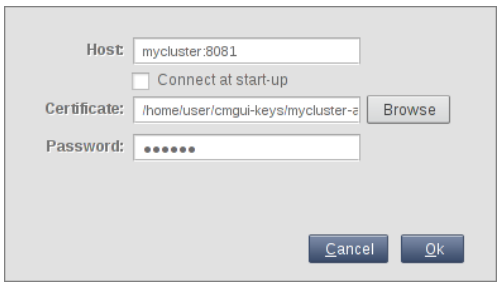

Figure 3.2: Edit Cluster dialog window

The host can be a name or an IP address. If the port on the host is not specified, then port 8081 is added automatically. The certificate location entry is where the administrator certificate `admin.pfx` file is located. The password is the password to the administrator certificate. Section 3.3 has details on the `admin.pfx` file, as well as on how to change the password used in the dialog with the `passwdpfx` or `cm-change-passwd` utilities.

After the cluster is added, the screen displays the connection parameters for the cluster (figure 3.3).

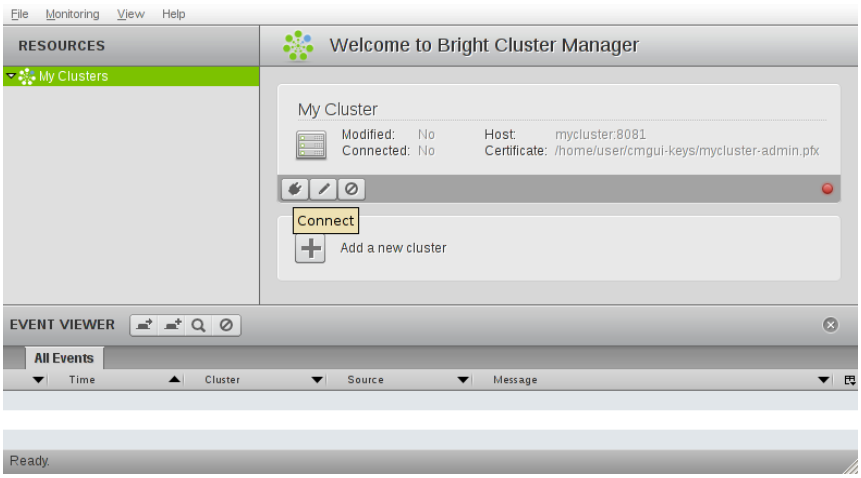

Figure 3.3: Connecting to a cluster

Clicking on the `Connect` button establishes a connection to the cluster, and `cmgui` by default then displays a tabbed pane overview screen of the cluster (figure 3.4):

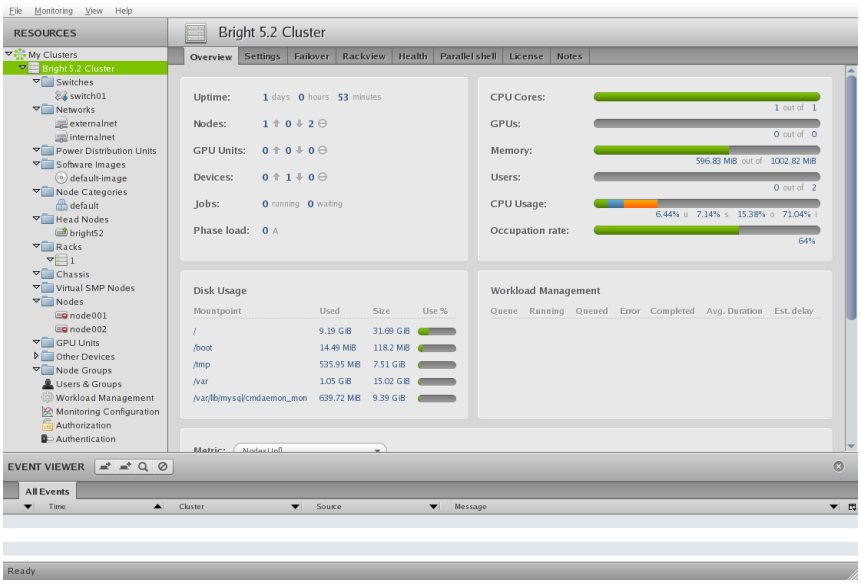

Figure 3.4: Cluster Overview

3.4.3 Navigating The Cluster Management GUI

Aspects of the cluster can be managed by administrators using `cmgui` (figure 3.4).

The resource tree, displayed on the left side of the window, consists of hardware resources such as nodes and switches as well as non-hardware resources such as `Users & Groups` and `Workload Management`. Selecting a resource opens an associated tabbed pane on the right side of the window that allows tab-related parameters to be viewed and managed.

The number of tabs displayed and their contents depend on the resource selected. The following standard tabs are available for most resources:

- `Overview`: provides an overview containing the most important status details for the resource.
- `Tasks`: accesses tasks that operate on the resource.
- `Settings`: allows configuration of properties of the resource.

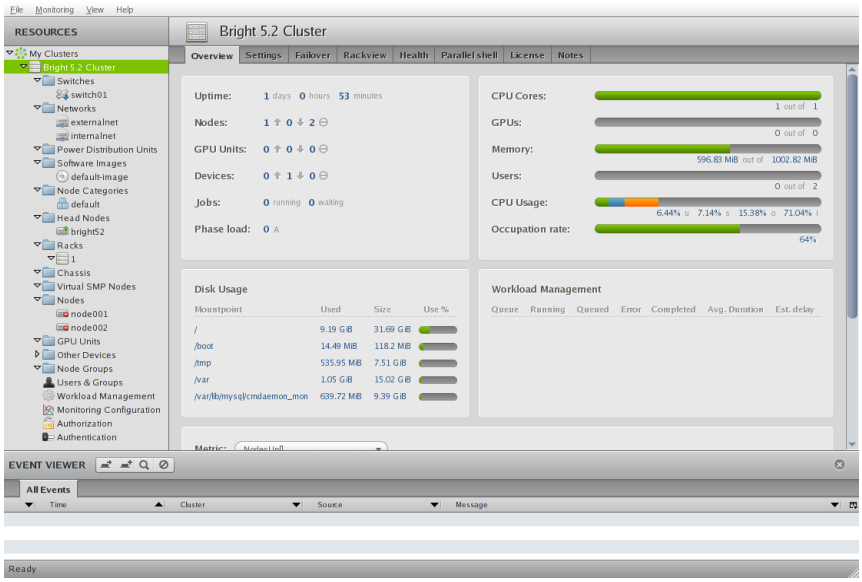

Figure 3.5: Node Settings

For example, the `Settings` tab of the `node001` resource (figure 3.5) displays properties, such as the hostname, that can be changed. The `Save` button on the bottom of the tab makes the changes active and permanent, while the `Revert` button undoes all unsaved changes.

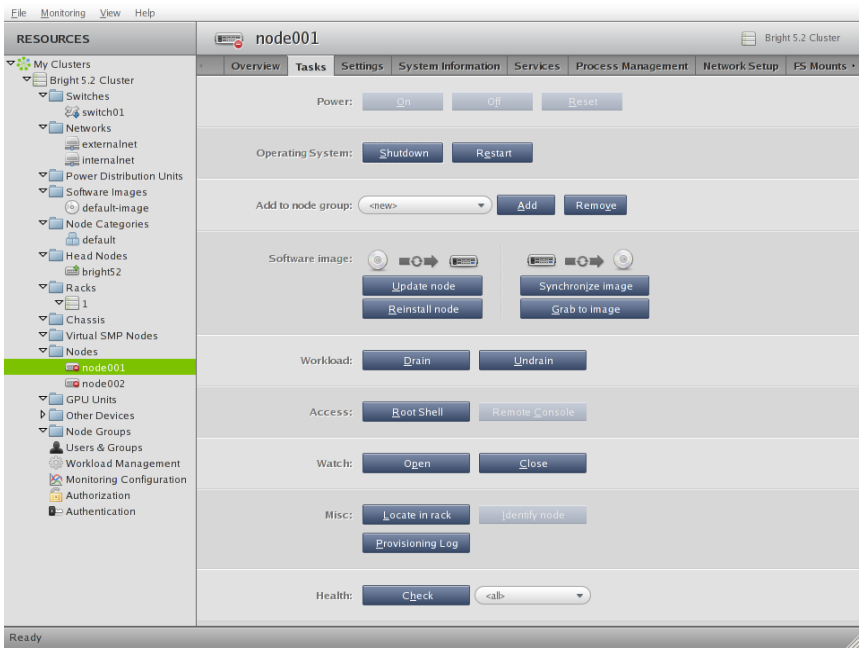

Figure 3.6: Node Tasks

Figure 3.6 shows the Tasks tab of the node001 resource. The tab displays operations that can be performed on the node001 resource. Details on setting these up, their use, and meaning are provided in the remaining chapters of this manual.

It is also possible to select a resource folder (rather than a resource item) in the tree. For example: Node Categories, Nodes, and Networks. Selecting a resource folder in the tree by default opens an Overview tab, which displays a list of resource items inside the folder. These are displayed in the resource tree and in the tabbed pane. Resource items in the tabbed pane can be selected, and operations carried out on them by clicking on the buttons at the bottom of the tabbed pane. For example, for Nodes, one or more nodes can be selected, and the Open, Add, Clone and Remove buttons can be clicked to operate on the selection (figure 3.7).

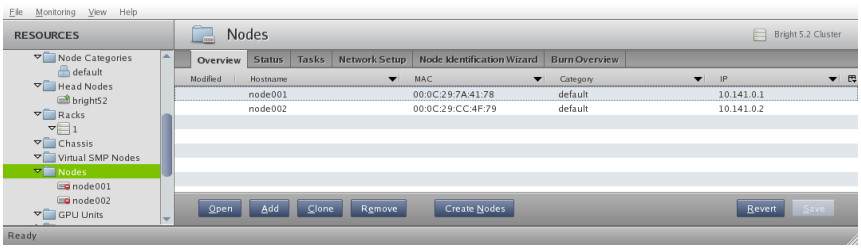

Figure 3.7: Nodes Overview

3.5 Cluster Management Shell

This section introduces the basics of the cluster management shell, cmsh. This is the command-line interface to cluster management in Bright Cluster Manager. Since cmsh and cmgui give access to the same cluster management functionality, an administrator need not become familiar with both interfaces. Administrators intending to manage a cluster with only cmgui may therefore safely skip this section.

Usually cmsh is invoked from an interactive session (e.g. through ssh) on the head node, but it can also be used to manage the cluster from outside.

3.5.1 Invoking cmsh

From the head node, cmsh can be invoked as follows:

```
[root@mycluster ~]# cmsh
[mycluster]%
```

Running cmsh without arguments starts an interactive cluster management session. To go back to the unix shell, a user enters quit:

```
[mycluster]% quit
[root@mycluster ~]#
```

The -c flag allows cmsh to be used in batch mode. Commands may be separated using semi-colons:

```
[root@mycluster ~]# cmsh -c "main showprofile; device status apc01"
admin
apc01 ..............  [   UP   ]
[root@mycluster ~]#
```

Alternatively, commands can be piped to cmsh:

```
[root@mycluster ~]# echo device status | cmsh
apc01 ............... [   UP   ]
mycluster ........... [   UP   ]
node001 ............. [   UP   ]
node002 ............. [   UP   ]
switch01 ............ [   UP   ]
[root@mycluster ~]#
```

In a similar way to unix shells, cmsh sources ~/.cm/cmsh/.cmshrc upon start-up in both batch and interactive mode. This is convenient for defining command aliases which may subsequently be used to abbreviate longer commands. For example, putting the following in .cmshrc allows the ds command to be used as an alias for device status:

Example

```
alias ds device status
```

The options usage information for cmsh is obtainable with cmsh -h (figure 3.8).

```
Usage: cmsh [options] ................ Connect to localhost using default port
       cmsh [options] <--certificate|-i certfile> <--key|-k keyfile> <host[:port]>
            Connect to a cluster using certificate and key in PEM format
       cmsh [options] <--certificate|-i certfile> [-password|-p password] <uri[:port]>
            Connect to a cluster using certificate in PFX format

       Valid options:
       --help|-h ..................... Display this help
       --noconnect|-u ................ Start unconnected
       --controlflag|-z .............. ETX in non-interactive mode
       --noredirect|-r ............... Do not follow redirects
       --norc|-n ..................... Do not load cmshrc file on start-up
       --command|-c <"c1; c2; ..."> .. Execute commands and exit
       --file|-f <filename> .......... Execute commands in file and exit
       --echo|-x ..................... Echo all commands
       --quit|-q ..................... Exit immediately after error
```

Figure 3.8: Usage information for cmsh

3.5.2 Levels, Modes, Help, And Commands Syntax In cmsh

The *top-level* of cmsh is the level that cmsh is in when entered without any options.

To avoid overloading a user with commands, cluster management functionality has been grouped and placed in separate cmsh *modes*. Modes and their levels are a hierarchy available below the top-level, and therefore to perform cluster management functions, a user switches and descends into the appropriate mode.

Figure 3.9 shows the top-level commands available in cmsh. These commands are displayed when help is typed in at the top-level of cmsh:

```
alias ..................... Set aliases
category .................. Enter category mode
cert ...................... Enter cert mode
color ..................... Manage console text color settings
connect ................... Connect to cluster
delimiter ................. Display/set delimiter
device .................... Enter device mode
disconnect ................ Disconnect from cluster
events .................... Manage events
exit ...................... Exit from current object or mode
export .................... Display list of aliases current list formats
help ...................... Display this help
history ................... Display command history
jobqueue .................. Enter jobqueue mode
jobs ...................... Enter jobs mode
list ...................... List state for all modes
main ...................... Enter main mode
modified .................. List modified objects
monitoring ................ Enter monitoring mode
network ................... Enter network mode
nodegroup ................. Enter nodegroup mode
partition ................. Enter partition mode
process ................... Enter process mode
profile ................... Enter profile mode
quit ...................... Quit shell
rack ...................... Enter rack mode
refresh ................... Refresh all modes
run ....................... Execute cmsh commands from specified file
session ................... Enter session mode
softwareimage ............. Enter softwareimage mode
unalias ................... Unset aliases
user ...................... Enter user mode
```

Figure 3.9: Top level commands in `cmsh`

All levels inside `cmsh` provide these top-level commands.
Passing a command as an argument to `help` gets details for it:

Example

```
[myheadnode]% help run
Name:
         run - Execute all commands in the given file(s)

Usage:
         run [OPTIONS] <filename> [<filename2> ...]

Options:
         -x, --echo
              Echo all commands

         -q, --quit
              Exit immediately after error
```

```
[myheadnode]%
```

In the general case, invoking `help` at any level without an argument provides the list of top-level commands, followed by commands that may be used at that level (list of top-level commands elided in example below):

Example

```
[myheadnode]% session
[myheadnode->session]% help
================================ Top ================================
...
============================== session ==============================
id ........................... Display current session id
killsession .................. Kill a session
list ......................... Provide overview of active sessions
[myheadnode->session]%
```

In the preceding example, `session` mode is entered, and help without any argument lists the possible commands at that level.

To enter a mode, a user enters the mode name at the `cmsh` prompt. The prompt changes to indicate that `cmsh` is in the requested mode, and commands for that mode can then be run. To leave a mode, and go back up a level, the `exit` command is used. At the top-level, `exit` has the same effect as the `quit` command, that is, the user leaves `cmsh` and returns to the unix shell.

Example

```
[bright52]% device
[bright52->device]% list
Type                 Hostname (key)   MAC                Category
-------------------- ---------------- ------------------ ------------
EthernetSwitch       switch01         00:00:00:00:00:00
MasterNode           bright52         00:0C:29:5D:55:46
PhysicalNode         node001          00:0C:29:7A:41:78  default
PhysicalNode         node002          00:0C:29:CC:4F:79  default
[bright52->device]% exit
[bright52]%
```

A command can also be executed in a mode without entering that mode. This is done by specifying the mode before the command. Most commands also accept arguments after the command. Multiple commands can be executed in one line by separating commands with semicolons.

A `cmsh` input line has the following syntax:

<mode> <cmd> <arg> ... <arg>; ... ; <mode> <cmd> <arg> ... <arg>

where *<mode>* and *<arg>* are optional. [2]

Example

[2] A more precise synopsis is:
`[<mode>] <cmd> [<arg> ... ] [; ... ; [<mode>] <cmd> [<arg> ... ]]`

```
[bright52->network]% device status master; list
bright52 ............ [   UP   ]
Name (key)       Netmask bits     Base address     Domain name
---------------- ---------------- ---------------- --------------------
externalnet      16               192.168.1.0      clustervision.com
internalnet      16               10.141.0.0       cm.cluster
[bright52->network]%
```

In the preceding example, while in `network` mode, the `status` command is executed in `device` mode and passed the argument `master`, making it display the status of the head node. The `list` command on the same line after the semi-colon still runs in `network` mode, as expected, and displays a list of networks.

3.5.3 Working With Objects

Modes in `cmsh` work with associated groupings of data called *objects*. For instance, `device` mode works with `device` objects, and `network` mode works with `network` objects. The commands used to deal with objects are the same in all modes, although generally not all of them function with an object:

Command	Description
`use`	Use the specified object. I.e.: Make the specified object the *current object*
`add`	Create the object and use it
`assign`	Assign a new object
`unassign`	Unassign an object
`clone`	Clone the object and use it
`remove`	Remove the object
`commit`	Commit local changes done to an object to the cluster management infrastructure
`refresh`	Undo local changes done to the object
`list`	List all objects at current level
`format`	Set formatting preferences for `list` output
`foreach`	Execute a set of commands on several objects
`show`	Display all properties of the object
`get`	Display specified property of the object
`set`	Set a specified property of the object
`clear`	Set empty value for a property of the object. If no property is specified, clear every value of the object
`append`	Append a value to a property of the object, for a multi-valued property
`removefrom`	Remove a value from a specific property of the object, for a multi-valued property

...continues

...continued

Command	Description
modified	List objects with uncommitted local changes
usedby	List objects that depend on the object
validate	Do a validation check on the properties of the object

Working with objects with these commands is demonstrated with several examples in this section.

Working With Objects: `use`
Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% status
node001 ............. [   UP   ]
[mycluster->device[node001]]% exit
[mycluster->device]%
```

In the preceding example, `use node001` issued from within `device` mode makes `node001` the *current object*. The prompt changes accordingly. The `status` command, without an argument, then returns status information just for `node001`, because making an object the current object makes all subsequent commands apply only to that object. Finally, the `exit` command unsets the current object.

Working With Objects: `add`, `commit`
Example

```
[mycluster->device]% add physicalnode node100 10.141.0.100
[mycluster->device*[node100*]]% category add test-category
[mycluster->category*[test-category*]]% device; use node100
[mycluster->device*[node100*]]% set category test-category
[mycluster->device*[node100*]]% commit
[mycluster->device[node100]]% exit
[mycluster->device]%
```

In the preceding example, within `device` mode, a new object `node100` is added, of type `physicalnode`, and with IP address `10.141.0.100`. The `category` test-category is then added, and the test-category object level within `category` mode is automatically dropped into when the command is executed. This is usually convenient, but not in this example, where it is assumed a property still needs to be set at the device node object level. To return to device mode again, at the level it was left, a multiple command "`device; use node100`" is executed. The multiple command can actually be run here with the "`;`" character removed, because the `use` command implies the existence of "`;`" character. The category property of the the `node100` object is set to the newly created category `test-category` and the object is then committed to store it permanently. Until the newly added object has been committed, it remains a local change that is lost when `cmsh` is exited.

Asterisk tags in the prompt are a useful reminder of a modified state, with each asterisk indicating a tagged object that has an unsaved, modified property.

In most modes the add command takes only one argument, namely the name of the object that is to be created. However, in device mode an extra object-type, in this case physicalnode, is also required as argument, and an optional extra IP argument may also be specified. The response to "help add" while in device mode gives details:

```
[myheadnode->device]% help add
Name:
      add - Create a new device of the given type with specified hostname

Usage:
      add <type> <hostname>
      add <type> <hostname> <ip>

Arguments:
      type
          physicalnode, virtualsmpnode, masternode, ethernetswitch,
          ibswitch, myrinetswitch, powerdistributionunit, genericdevice,
          racksensor, chassis, gpuunit
```

Working With Objects: clone, modified, remove

Continuing on with the node object node100 that was created in the previous example, it can be cloned to node101 as follows:

Example

```
[mycluster->device]% clone node100 node101
Warning: The Ethernet switch settings were not cloned, and have to be
set manually
[mycluster->device*[node101*]]% exit
[mycluster->device*]% modified
State  Type                     Name
------ ------------------------ -----------------------------------
+      Cloned                   node101
[mycluster->device*]% commit
[mycluster->device]%
[mycluster->device]% remove node100
[mycluster->device*]% commit
[mycluster->device]%
```

The modified command is used to check what objects have uncommitted changes, and the new object node101 that is seen to be modified, is saved with a commit. The device node100 is then removed by using the remove command. A commit executes the removal.

The modified command corresponds roughly to the functionality of the List of Changes menu option under the View menu of cmgui's main menu bar.

The "+" entry in the State column in the output of the modified command in the preceding example indicates the object is a newly added one, but not yet committed. Similarly, a "-" entry indicates an object that is to be removed on committing, while a blank entry indicates that the object has been modified without an addition or removal involved.

Cloning an object is a convenient method of duplicating a fully configured object. When duplicating a device object, cmsh will attempt to automatically assign a new IP address using a number of heuristics. In the preceding example, node101 is assigned IP address 10.141.0.101.

Working With Objects: `get`, `set`, `refresh`, `revision`

The `get` command is used to retrieve a specified property from an object, and `set` is used to set it:

Example

```
[mycluster->device]% use node101
[mycluster->device[node101]]% get category
test-category
[mycluster->device[node101]]% set category default
[mycluster->device*[node101*]]% get category
default
[mycluster->device*[node101*]]% modified
State  Type                     Name
------ ----------------------- ------------------------------
       Device                   node101
[mycluster->device*[node101*]]% refresh
[mycluster->device[node101]]% modified
No modified objects of type device
[mycluster->device[node101]]% get category
test-category
[mycluster->device[node101]]%
```

Here, the `category` property of the `node101` object is retrieved by using the `get` command. The property is then changed using the `set` command. Using `get` confirms that the value of the property has changed, and the `modified` command reconfirms that `node101` has local uncommitted changes. The `refresh` command undoes the changes made, and the `modified` command confirms that no local changes exist. Finally the `get` command reconfirms that no local changes exist.

A string can be set as a revision label for any object:

Example

```
[mycluster->device[node101]]% set revision "changed on 10th May"
[mycluster->device*[node101*]]% get revision
[mycluster->device*[node101*]]% changed on 10th May 2011
```

This can be useful when using shell scripts with an input text to label and track revisions when sending commands to `cmsh`. How to send commands from the shell to `cmsh` is introduced in section 3.5.1.

Some properties are booleans. For these, the values "`yes`", "`1`", "`on`" and "`true`" are equivalent to each other, as are their opposites "`no`", "`0`", "`off`" and "`false`". These values are case-insensitive.

Working With Objects: `clear`

Example

```
[mycluster->device]% set node101 mac 00:11:22:33:44:55
[mycluster->device*]% get node101 mac
00:11:22:33:44:55
[mycluster->device*]% clear node101 mac
[mycluster->device*]% get node101 mac
00:00:00:00:00:00
[mycluster->device*]%
```

The get and set commands are used to view and set the MAC address of node101 without running the use command to make node101 the *current object*. The clear command then unsets the value of the property. The result of clear depends on the type of the property it acts on. In the case of string properties, the empty string is assigned, whereas for MAC addresses the special value 00:00:00:00:00:00 is assigned.

Working With Objects: list, format

The list command is used to list all device objects. The -f flag takes a format string as argument. The string specifies what properties are printed for each object, and how many characters are used to display each property in the output line. In following example a list of objects is requested, displaying the hostname, ethernetswitch and ip properties for each object.

Example

```
[bright52->device]% list -f hostname:14,ethernetswitch:15,ip
hostname (key) ethernetswitch  ip
-------------- --------------- --------------------
apc01                          10.142.254.1
bright52       switch01:46     10.142.255.254
node001        switch01:47     10.142.0.1
node002        switch01:45     10.142.0.2
switch01                       10.142.253.1
[bright52->device]%
```

Running the list command with no argument uses the current format string for the mode.

Running the format command without arguments displays the current format string, and also displays all available properties including a description of each property. For example (output truncated):

Example

```
[bright52->device]% format
Current list printing format:
-----------------------------
hostname:[10-14]

Valid fields:
-------------
Type                    : The type of the device
activation              : Date on which node was defined
banks                   : Number of banks
burnconfig              : Burning configuration
burning                 : Indicates that the node is in burn mode
category                : Category to which this node belongs
...
```

To change the current format string, a new format string can be passed as an argument to format.

The format print specification uses the delimiter ":" to separate the parameter and the value for the width of the parameter column. For example, a width of 10 can be set with:

Example

```
[bright52->device]% format hostname:10
[bright52->device]% list
hostname (
----------
apc01
bright60
node001
node002
switch01
```

A range of widths can be set, from a minimum to a maximum, using square brackets. A single minimum width possible is chosen from the range that fits all the characters of the column. If the number of characters in the column exceeds the maximum, then the maximum value is chosen. For example:

Example

```
[bright52->device]% format hostname:[10-14]
[bright52->device]% list
hostname (key)
--------------
apc01
bright60
node001
node002
switch01
```

The parameters to be viewed can be chosen from a list of valid fields by running the format command without any options, as shown earlier.

Multiple parameters can be specified as a comma-separated list (with a colon-delimited width specification for each parameter). For example:

Example

```
[bright52->device]% format hostname:[10-14], ethernetswitch:14, ip:20
[bright52->device]% list
hostname (key) ethernetswitch  ip
-------------- -------------- --------------------
apc01                         10.142.254.1
bright52       switch01:46    10.142.255.254
node001        switch01:47    10.142.0.1
node002        switch01:45    10.142.0.2
switch01                      10.142.253.1
```

The default parameter settings can be restored with the `-r|-reset` option:

Example

```
[bright52->device]% format -r
[bright52->device]% format | head -3
Current list printing format:
-----------------------------
type:22, hostname:[16-32], mac:18, category:16, ip:15, network:14, powe\
rdistributionunits:16
[bright52->device]%
```

Working With Objects: `append`, `removefrom`

When dealing with a property of an object that can take more than one value at a time—a list of values—the `append` and `removefrom` commands can be used to respectively append to and remove elements from the list. However, the `set` command may also be used to assign a new list at once. In the following example values are appended and removed from the `powerdistributionunits` properties of device `node001`. The `powerdistributionunits` properties represents the list of ports on power distribution units that a particular device is connected to. This information is relevant when power operations are performed on a node. Chapter 5 has more information on power settings and operations.

Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% get powerdistributionunits
apc01:1
[mycluster->device[node001]]% append powerdistributionunits apc01:5
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:1 apc01:5
[mycluster->device*[node001*]]% append powerdistributionunits apc01:6
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:1 apc01:5 apc01:6
[mycluster->device*[node001*]]% removefrom powerdistributionunits apc01:5
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:1 apc01:6
[mycluster->device*[node001*]]% set powerdistributionunits apc01:1 apc01:02
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:1 apc01:2
[mycluster->device*[node001*]]%
```

Working With Objects: `usedby`

Removing a specific object is only possible if other objects do not have references to it. To help the administrator discover a list of objects that depend on ("use") the specified object, the `usedby` command may be used. In the following example, objects depending on device `apc01` are requested. The `usedby` property of `powerdistributionunits` indicates that device objects `node001` and `node002` contain references to ("use") the object `apc01`. In addition, the `apc01` device is itself displayed as being in the `up` state, indicating a dependency of `apc01` on itself. If the device is to be removed, then the 2 references to it first need to be removed, and the device also first has to be brought to the `closed` state by using the `close` command.

Example

```
[mycluster->device]% usedby apc01
Device used by the following:
Type             Name       Parameter
---------------- ---------- --------------------
Device           apc01      Device is up
Device           node001    powerDistributionUnits
Device           node002    powerDistributionUnits
[mycluster->device]%
```

Working With Objects: `validate`

Whenever committing changes to an object, the cluster management infrastructure checks the object to be committed for consistency. If one or more consistency requirements are not met, then `cmsh` reports the violations that must be resolved before the changes are committed. The `validate` command allows an object to be checked for consistency without committing local changes.

Example

```
[mycluster->device]% use node001
[mycluster->device[node001]]% clear category
[mycluster->device*[node001*]]% commit
Code  Field                   Message
----- ----------------------- --------------------------
1     category                The category should be set
[mycluster->device*[node001*]]% set category default
[mycluster->device*[node001*]]% validate
All good
[mycluster->device*[node001*]]% commit
[mycluster->device[node001]]%
```

3.5.4 Accessing Cluster Settings

The management infrastructure of Bright Cluster Manager is designed to allow cluster partitioning in the future. A cluster partition can be viewed as a virtual cluster inside a real cluster. The cluster partition behaves as a separate cluster while making use of the resources of the real cluster in which it is contained. Although cluster partitioning is not yet possible in the current version of Bright Cluster Manager, its design implications do decide how some global cluster properties are accessed through `cmsh`.

In `cmsh` there is a `partition` mode which will, in a future version, allow an administrator to create and configure cluster partitions. Currently, there is only one fixed partition, called `base`. The `base` partition represents the physical cluster as a whole and cannot be removed. A number of properties global to the cluster exist inside the `base` partition. These properties are referenced and explained in remaining parts of this manual.

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% partition use base
[myheadnode->partition[base]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Administrator e-mail
Burn configs                   <2 in submode>
Cluster name                   My Cluster
Default burn configuration     default
Default category               default
Default software image         default-image
External network               externalnet
Failover                       not defined
IPMI Password                  *********
IPMI User ID                   2
```

```
IPMI User name                  ADMIN
Management network              internalnet
Masternode                      myheadnode
Name                            base
Name servers                    192.168.101.1
Node basename                   node
Node digits                     3
Notes                           <0 bytes>
Revision
Search domains                  clustervision.com
Time servers                    pool.ntp.org
Time zone                       America/Los_Angeles
```

3.5.5 Advanced cmsh Features

This section describes some advanced features of cmsh and may be skipped on first reading.

Command Line Editing

Command line editing and history features from the readline library are available. See http://tiswww.case.edu/php/chet/readline/rluserman.html for a full list of key-bindings.

The most useful features provided by readline are tab-completion of commands and arguments, and command history using the arrow keys.

Mixing cmsh And Unix Shell Commands

Occasionally it can be useful to be able to execute unix commands while performing cluster management. For this reason, cmsh allows users to execute unix commands by prefixing the command with a "!" character:

Example

```
[mycluster]% !hostname -f
mycluster.cm.cluster
[mycluster]%
```

Executing the ! command by itself will start an interactive login sub-shell. By exiting the sub-shell, the user will return to the cmsh prompt.

Besides simply executing commands from within cmsh, the output of unix shell commands can also be used within cmsh. This is done by using the "backtick syntax" available in most unix shells.

Example

```
[mycluster]% device use `hostname`
[mycluster->device[mycluster]]% status
mycluster ................. [   UP   ]
[mycluster->device[mycluster]]%
```

Output Redirection

Similar to unix shells, cmsh also supports output redirection to the shell through common operators such as >, >> and |.

Example

```
[mycluster]% device list > devices
[mycluster]% device status >> devices
[mycluster]% device list | grep node001
Type           Hostname (key)      MAC (key)           Category
-------------- --------------      ------------------- ----------
PhysicalNode   node001             00:E0:81:2E:F7:96   default
```

Looping Over Objects With `foreach`

It is frequently convenient to be able to execute a `cmsh` command on several objects at once. The `foreach` command is available in a number of `cmsh` modes for this purpose. A `foreach` command takes a list of space-separated object names (the keys of the object) and a list of commands that must be enclosed by parentheses, i.e.: "(" and ")". The `foreach` command will then iterate through the objects, executing the list of commands on the iterated object each iteration.

The `foreach` syntax is:

`foreach` *<obj>* ··· *<obj>* (*<cmd>* ; ··· ; *<cmd>*)

Example

```
[mycluster->device]% foreach node001 node002 (get hostname; status)
node001
node001 ..... [  UP  ]
node002
node002 ..... [  UP  ]
[mycluster->device]%
```

With the `foreach` command it is possible to perform `set` commands on groups of objects simultaneously, or to perform an operation on a group of objects.

For extra convenience, `device` mode in `cmsh` supports a number of additional flags (`-n`, `-g` and `-c`) which can be used for selecting devices. Instead of passing a list of objects to `foreach` directly, the flags may be used to select the nodes to loop over. The `-g` and `-c` flags take a node group and category argument respectively. The `-n` flag takes a node-list argument. Node-lists may be specified using the following syntax:

<node> ,..., *<node>* , *<node>* .. *<node>*

Example

```
[demo->device]% foreach -c default (status)
node001 ..... [  DOWN  ]
node002 ..... [  DOWN  ]
[demo->device]% foreach -g rack8 (status)
...
[demo->device]% foreach -n node001,node008..node016,node032..node080 (status)
...
[demo->device]%
```

The -o (`--clone`) option allows the cloning (section 3.5.3) of objects in a loop. In the following example, from `device` mode, `node001` is used as the base object from which other nodes from `node022` up to `node024` are cloned:

Example

```
[bright52->device]% foreach --clone node001 -n node022..node024 ()
[bright52->device*]% list | grep node
Type          Hostname (key) MAC                Category  Ip
------------  -------------- -----------------  --------  -----------
PhysicalNode  node001        00:0C:29:EF:40:24  default   10.141.0.1
PhysicalNode  node022        00:00:00:00:00:00  default   10.141.0.22
PhysicalNode  node023        00:00:00:00:00:00  default   10.141.0.23
PhysicalNode  node024        00:00:00:00:00:00  default   10.141.0.24
[bright52->device*]% commit
```

To avoid possible confusion: the cloned objects are merely objects (placeholder schematics and settings, with some different values for some of the settings, such as IP addresses, decided by heuristics). So it is explicitly not the software disk image of `node001` that is duplicated by object cloning to the other nodes.

Finally, the wildcard character `*` with `foreach` implies all the objects that the `list` command lists for that mode. It is used without flags:

Example

```
[myheadnode->device]% foreach * (get ip; status)
10.141.253.1
switch01 ............ [  DOWN  ]
10.141.255.254
myheadnode .......... [   UP   ]
10.141.0.1
node001 ............. [ CLOSED ]
10.141.0.2
node002 ............. [ CLOSED ]
[myheadnode->device]%
```

3.6 Cluster Management Daemon

The *cluster management daemon* or *CMDaemon* is a server process that runs on all nodes of the cluster (including the head node). The cluster management daemons work together to make the cluster manageable. When applications such as `cmsh` and `cmgui` communicate with the cluster management infrastructure, they are actually interacting with the cluster management daemon running on the head node. Cluster management applications never communicate directly with cluster management daemons running on non-head nodes.

The CMDaemon application starts running on any node automatically when it boots, and the application continues running until the node shuts down. Should CMDaemon be stopped manually for whatever reason, its cluster management functionality becomes unavailable, making it hard for administrators to manage the cluster. However, even with the daemon stopped, the cluster remains fully usable for running computational jobs using a workload manager.

The only route of communication with the cluster management daemon is through TCP port `8081`. The cluster management daemon accepts only SSL connections, thereby ensuring all communications are encrypted. Authentication is also handled in the SSL layer using client-side `X509v3` certificates (section 3.3).

On the head node, the cluster management daemon uses a MySQL database server to store all of its internal data. Monitoring data is also stored in a MySQL database.

3.6.1 Controlling The Cluster Management Daemon

It may be useful to shut down or restart the cluster management daemon. For instance, a restart may be necessary to activate changes when the cluster management daemon configuration file is modified. The cluster management daemon operation can be controlled through the following init script arguments to `/etc/init.d/cmd`:

Init Script Operation	Description
`stop`	stop the cluster management daemon
`start`	start the cluster management daemon
`restart`	restart the cluster management daemon
`status`	report whether cluster management daemon is running
`full-status`	report detailed statistics about cluster management daemon
`upgrade`	update database schema after version upgrade (*expert only*)
`debugon`	enable debug logging (*expert only*)
`debugoff`	disable debug logging (*expert only*)

Example

To restart the cluster management daemon on the head node of a cluster:

```
[root@mycluster ~]# /etc/init.d/cmd restart
Waiting for CMDaemon (2916) to terminate...
                                                           [  OK  ]
Waiting for CMDaemon to start...                           [  OK  ]
[root@mycluster ~]#
```

3.6.2 Configuring The Cluster Management Daemon

Some cluster configuration changes can be done by modifying the cluster management daemon configuration file. For the head node, this is located at:

```
/cm/local/apps/cmd/etc/cmd.conf
```

For regular nodes, it is located inside of the software image that the node uses.

Appendix C describes all recognized configuration file directives and how they can be used. Normally there is no need to modify the default settings.

After modifying the configuration file, the cluster management daemon must be restarted to activate the changes.

3.6.3 Configuration File Generation

As part of its tasks, the cluster management daemon writes out a number of system configuration files. Some configuration files are written out in

their entirety, whereas other configuration files only contain sections that have been inserted by the cluster management daemon. Appendix A lists all system configuration files that are generated.

A file that has been generated by the cluster management daemon contains a header:

```
# This file was automatically generated by cmd. Do not edit manually!
```

Sections of files that have been generated by the cluster management daemon will read as follows:

```
# This section of this file was automatically generated by cmd. Do not edit manuall
# BEGIN AUTOGENERATED SECTION -- DO NOT REMOVE
...
# END AUTOGENERATED SECTION   -- DO NOT REMOVE
```

When generated files or sections of files are modified manually, the changes are automatically overwritten the next time the content is accessed, an event is generated, and the manually modified configuration file is backed up to:

```
/var/spool/cmd/saved-config-files
```

Sometimes, overriding the automatically generated configuration file contents may be necessary. The `FrozenFile` configuration file directive in `cmd.conf` allows this.

Example

```
FrozenFile = { "/etc/dhcpd.conf", "/etc/postfix/main.cf" }
```

3.6.4 Configuration File Conflicts Between The Standard Distribution And Bright

While Bright Cluster Manager changes as little as possible of the standard distributions that it manages, there can sometimes be unavoidable issues. In particular, sometimes a standard distribution utility or service generates configuration files that conflict with what the Bright Cluster Manager configuration files listed in Appendix A carry out.

For example, the Red Hat security configuration tool `system-config-securitylevel` can conflict with what `shorewall` (section 12.2) does, while the Red Hat Authentication Configuration Tool `authconfig` can conflict with what the LDAP and PAM configuration settings of Bright Cluster Manager achieve.

In such cases the configuration generated by Bright Cluster Manager must be given precedence, and the generation of configuration files from the standard distribution tools should be avoided. Sometimes using fully or partial frozen configuration files (section 3.6.3) allow a workaround. Otherwise, the functionality of the Bright Cluster Manager version usually allows the required configuration function to be implemented.

4

Configuring The Cluster

After Bright Cluster Manager software has been installed on the head node, the cluster must be configured. This chapter goes through a number of basic cluster configuration aspects that are important to get all the hardware up and running.

Section 4.1 explains how a Bright Cluster Manager license is viewed, verified, requested, and installed.

Section 4.2 details how the internal and external network parameters of the cluster can be changed.

Section 4.3 covers how InfiniBand is set up.

Section 4.4 describes how IPMI is set up.

Section 4.5 describes how switches are set up.

Section 4.6 explains how disk layouts are configured, as well as how diskless nodes are set up.

Section 4.7 describes how NFS volumes are exported from an NFS server and mounted to nodes using the integrated approach of Bright Cluster Manager.

Section 4.8 describes how services can be run from Bright Cluster Manager.

Section 4.9 describes how a rack can be configured and managed with Bright Cluster Manager.

More elaborate aspects of cluster configuration such as power management, user management, package management, and workload management are covered in later chapters.

4.1 Installing A License

Any Bright Cluster Manager installation requires a *license file* to be present on the head node. The license file specifies the conditions under which a particular Bright Cluster Manager installation has been licensed.

Example

- the "`Licensee`" details, which include the name of the organization, is an attribute of the license file that specifies the condition that only the specified organization may use the software
- the "`Licensed nodes`" attribute specifies the maximum number of nodes that the cluster manager may manage. Head nodes are regarded as nodes too for this attribute.

- the "Expiration date" of the license is an attribute that sets when the license expires. It is sometimes set to a date in the near future so that the cluster owner effectively has a trial period. A new license with a longer period can be requested (section 4.1.3) after the owner decides to continue using the cluster with Bright Cluster Manager

A license file can only be used on the machine for which it has been generated and cannot be changed once it has been issued. This means that to change licensing conditions, a new license file must be issued.

The license file is sometimes referred to as the *cluster certificate*, because it is the X509v3 certificate of the head node, and is used throughout cluster operations. Section 3.3 has more information on certificate based authentication.

4.1.1 Displaying License Attributes

Before starting the configuration of a cluster, it is important to verify that the attributes included in the license file have been assigned the correct values. The license file is installed in the following location:

```
/cm/local/apps/cmd/etc/cert.pem
```

and the associated private key file is in:

```
/cm/local/apps/cmd/etc/cert.key
```

To verify that the attributes of the license have been assigned the correct values, the License tab of the GUI can be used to display license details (figure 4.1):

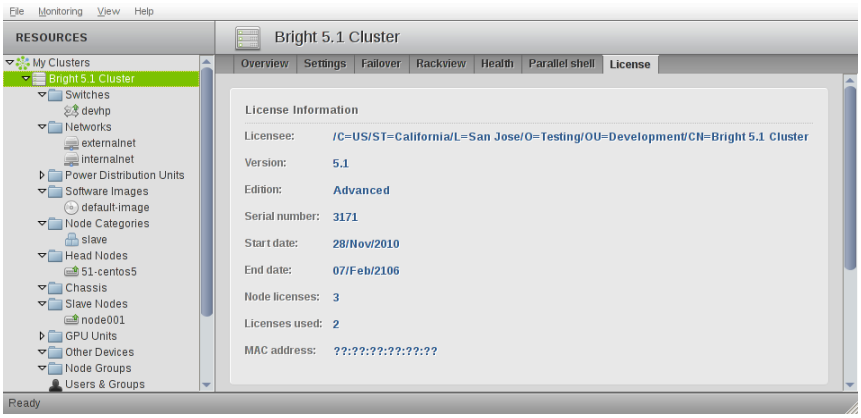

Figure 4.1: License Information

Alternatively the licenseinfo in cmsh main mode may be used:

Example

```
[root@52-centos5 ~]# cmsh
[52-centos5]% main licenseinfo
License Information
------------------- ------------------------------------------------
Licensee            /C=US/ST=California/L=San Jose/O=BC/OU=Development/C\
                    N=Bright 5.2 Cluster
Serial Number       5657
Start Time          Tue Apr  5 00:00:00 2011
```

```
End Time            Tue Mar 30 23:59:59 2038
Version             5.2
Edition             Advanced
Licensed Nodes      3
Node Count          3
MAC Address         ??:??:??:??:??:??
```

The license in the example above allows just 3 nodes to be used. It is not tied to a specific MAC address, so it can be used anywhere. For convenience, the `Node Count` field in the output of `licenseinfo` shows the current number of nodes used.

4.1.2 Verifying A License—The `verify-license` Utility

The `verify-license` utility is used to check licenses independent of whether the cluster management daemon is running.

When an invalid license is used, the cluster management daemon cannot start. The license problem is logged in the cluster management daemon logfile:

Example

```
[root@bright52 ~]# /etc/init.d/cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
[root@bright52 ~]# tail -1 /var/log/cmdaemon
Dec 30 15:57:02 bright52 CMDaemon: Fatal: License has expired
```

but further information cannot be obtained with, for example, `cmgui` and `cmsh` , because these clients themselves obtain their information from the cluster management daemon.

In such a case, the `verify-license` utility is meant for troubleshooting license issues, using the following options:

The `info` option of `verify-license` prints license details:

Example

```
[root@bright52 ~]# verify-license
Usage: verify-license <path to certificate> <path to keyfile> <verify|info>
[root@bright52 ~]# cd /cm/local/apps/cmd/etc/
[root@bright52 etc]# verify-license cert.pem cert.key info
========= Certificate Information ========
Version:              5.2
Edition:              Advanced
Common name:          Bright 5.2 Cluster
Organization:         Bright Computing
Organizational unit: Development
Locality:             San Jose
State:                California
Country:              US
Serial:               5657
Starting date:         5 Apr 2011
Expiration date:      30 Mar 2038
MAC address:          ??:??:??:??:??:??
Licensed nodes:       3
==========================================
[root@bright52 etc]#
```

The `verify` option of `verify-license` checks the validity of the license:

- If the license is valid then no output is produced, and the utility exits with exit-code `0`.
- If the license is invalid then output is produced indicating what is wrong. Messages such as these are then displayed:
  - If the license is old:

```
[root@bright52 etc]# verify-license cert.pem cert.key verify
License has expired
License verification failed.
```

  - If the certificate is not from Bright Computing:

```
[root@bright52 etc]# verify-license cert.pem cert.key verify
Invalid license: This certificate was not signed by Bright
Computing
License verification failed.
```

4.1.3 Requesting And Installing A License Using A Product Key

Verifying License Attributes

It is important to verify that the license attributes are correct before proceeding with cluster configuration. In particular, the license date should be checked to make sure that the license has not expired.

If the attributes of the license are correct, the remaining parts of this section (4.1.3) may safely be skipped.

Requesting A License

If the license has expired, or if the license attributes are otherwise not correct, a new license file must be requested.

The request for a new license file is made using a *product key* with the `request-license` command.

The product key: The product key entitles the user to request a license, and is a sequence of digits similar to the following:

```
000354-515786-112224-207441-186713
```

A product key is obtained from any Bright Cluster Manager reseller, and is activated by the user when obtaining the license. A product key can obtain a license only once. Upon product key activation, the license obtained permits the cluster to work with particular settings for, amongst others, the period of use and the number of nodes.

The `request-license` command: The `request-license` command requests a license, and works most conveniently with a cluster that is able to access the internet. The request can also be made regardless of cluster connectivity to outside networks.

There are four options to use the product key to get the license:

1. If the cluster has access to the WWW port, the product key is activated immediately on successfully completing the dialog started by the `request-license` command.

- If the cluster uses a web-proxy, then the environment variable `http_proxy` must be set before `request-license` is run. From a bash prompt this is set with:

  ```
  export http_proxy=<proxy>
  ```

  where `<proxy>` is the hostname or IP address of the proxy.

2. If the cluster does not have access to the WWW port, the administrator may activate the product key by pointing an off-cluster web-browser to:

   ```
   http://support.brightcomputing.com/licensing
   ```

   The CSR (Certificate Sign Request) data generated by running the `request-license` command on the cluster is entered in the web form at that URL, and a (signed) license will be returned. This license is in the form of a plain text certificate.

   As the web form response explains, it is to be saved to the head node as a file, and saving it directly is possible from most browsers. Cutting and pasting it into an editor and saving it on the head node as a file will do the job too, since it is plain text.

   The license certificate is then installed by running the command

   ```
   install-license <filename>
   ```

   on the head node, where `<filename>` is the name of the signed license file that was saved.

3. If no web access is available to the administrator, then the point at which the `request-license` command prompts "`Submit certificate request to http://support.brightcomputing.com/licensing/ ?`" should be answered negatively. CSR data generated is then conveniently displayed on the screen as well as saved, and it may be sent by email to `ca@brightcomputing.com`. A certificate will be emailed back from the Bright Cluster Manager License Desk. This certificate can then be handled further as described in option 2.

4. If no internet access is available at all to the administrator, the CSR data may be faxed or sent by postal mail to any Bright Cluster Manager reseller. A certificate will be faxed or sent back in response. This certificate can then be handled further as described in option 2.

Example

```
[root@bright52 ~]# request-license
Product Key (XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX):
000354-515786-112224-207440-186713

Country Name (2 letter code): US
State or Province Name (full name): California
Locality Name (e.g. city): San Jose
```

```
Organization Name (e.g. company): Bright Computing, Inc.
Organizational Unit Name (e.g. department): Development
Cluster Name: My Cluster
Private key data saved to /cm/local/apps/cmd/etc/cert.key.new

MAC Address of primary head node (bright52) for eth0 [00:0C:29:87:B8:B3]:
Will this cluster use a high-availability setup with 2 head nodes? [y/N] n

Certificate request data saved to /cm/local/apps/cmd/etc/cert.csr.new
Submit certificate request to http://support.brightcomputing.com/licensing/ ?
[Y/n] y

Contacting http://support.brightcomputing.com/licensing/...
License granted.
License data was saved to /cm/local/apps/cmd/etc/cert.pem.new
Install license ? [Y/n] n
Use "install-license /cm/local/apps/cmd/etc/cert.pem.new" to install the
license.
```

Installing A License

Referring to the preceding example:

After the certificate request is sent to Bright Computing and approved, the license is granted.

If the prompt "`Install license ?`" is answered with a "`Y`" (the default), the `install-license` script is run.

If the prompt is answered with an "`n`" then the `install-license` script must be run separately in order to complete installation of the license.

The `install-license` script takes the temporary location of the new license file generated by `request-license` as its argument, and installs related files on the head node. Running it completes the license installation on the head node.

Example

Assuming the new certificate is saved as `cert.pem.new`:

```
[root@bright52 ~]# install-license /cm/local/apps/cmd/etc/cert.pem.new
========= Certificate Information ========
Version:             5.2
Edition:             Advanced
Common name:         My Cluster
Organization:        Bright Computing, Inc.
Organizational unit: Development
Locality:            San Jose
State:               California
Country:             US
Serial:              5310
Starting date:       22 Mar 2011
Expiration date:     16 Mar 2038
MAC address:         00:0C:29:87:B8:B3
Licensed nodes:      2048
==========================================

Is the license information correct ? [Y/n] y
```

```
In order to authenticate to the cluster using the Cluster Management GUI
(cmgui), one must hold a valid certificate and a corresponding key. The
certificate and key are stored together in a password-protected PFX (a.k.a.
PKCS#12) file.

Please provide a password that will be used to password-protect the PFX file
holding the administrator certificate (/root/.cm/cmgui/admin.pfx).

Password:
Verify password:

Installed new license

Waiting for CMDaemon to stop: OK

Installing admin certificates

Waiting for CMDaemon to start: OK

New license was installed. In order to allow nodes to obtain a new
node certificate, all nodes must be rebooted.

Please issue the following command to reboot all nodes:
        pexec reboot
```

Rebooting Nodes After An Install

The first time a product key is used: After using a product key with the command `request-license` during a cluster installation, and then running `install-license`, a reboot is required of all nodes in order for them to pick up and install their new certificates. The `install-license` script has at this point already renewed the administrator certificates for use with `cmsh` and `cmgui` on the head node. The parallel execution command `pexec reboot` suggested towards the end of the `install-license` script output is what can be used to reboot all other nodes. Since such a command is best done by an administrator manually, `pexec reboot` is not scripted.

The subsequent times that a product key is used: On running the command `request-license` for the cluster, the administrator is prompted on whether to re-use the existing keys and settings from the existing license. If the existing keys are kept, a `pexec reboot` is not required. This is because these keys are X509v3 certificates issued from the head node. Any user or node certificates generated using the same certificate are therefore still valid and so regenerating them for nodes via a reboot is not required, allowing users to continue working uninterrupted.

After the license is installed, verifying the license attribute values is a good idea. This can be done using the `licenseinfo` command in `cmsh`, or the `License` tab in `cmgui`'s cluster resource tabbed pane (section 4.1.1).

4.2 Network Settings

A simplified quickstart guide to setting up the external head node network configuration on a vendor-prepared cluster is given in Appendix J. This section (4.2) covers network configuration more thoroughly.

After the cluster is set up with the correct license, the next configuration step is to define the networks that are present (sections 4.2.1 and 4.2.2).

During the Bright Cluster Manager installation at least two default network objects were created:

`internalnet`: the primary internal cluster network, or management network. This is used for booting non-head nodes and for all cluster management communication. In the absence of other internal networks, `internalnet` it is also used for storage and communication between compute jobs. Changing the configuration of this network is described on page 70 under the subheading "Changing Internal Network Parameters For The Cluster".

`externalnet`: the network connecting the cluster to the outside world (typically a corporate or campus network). Changing the configuration of this network is described on page 67 under the subheading "Changing External Network Parameters For The Cluster".

For a Type 1 cluster (section 2.3.6) this is illustrated conceptually by figure 4.2.

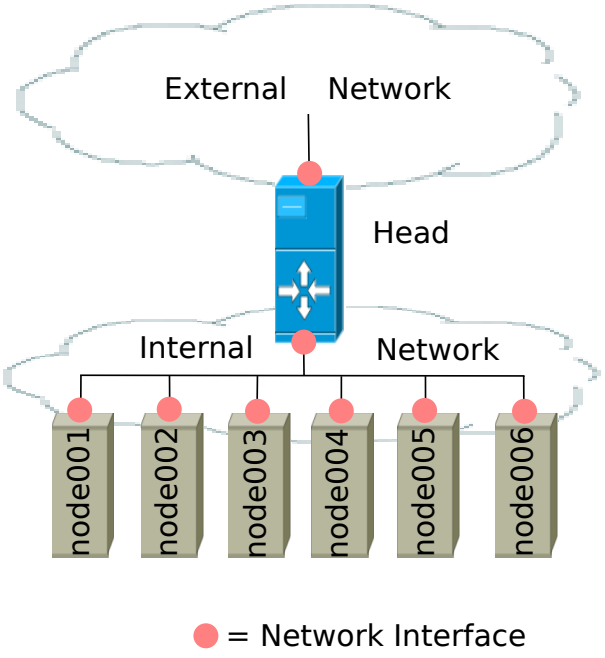

Figure 4.2: Network Settings Concepts

The configuration of network settings is completed when, after having configured the general network settings, specific IP addresses are then also assigned to the interfaces of devices connected to the networks. Changing the configuration of the head node external interface is described on page 68 under the subheading "The IP address of the cluster". Changing the configuration of the internal network interfaces is described

on page 72 under the subheading "The IP addresses and other interface values of the internal network".

4.2.1 Configuring Networks

The `network` mode in `cmsh` gives access to all network-related operations using the standard object commands. Section 3.5.3 introduces `cmsh` modes and working with objects.

In `cmgui`, a network can be configured by selecting the `Networks` item in the resource tree (figure 4.3).

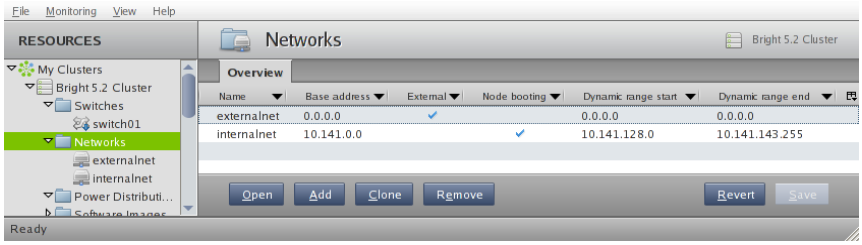

Figure 4.3: Networks

In the context of the OSI Reference Model, each network object represents a layer 3 (i.e. Network Layer) IP network, and several layer 3 networks can be layered on a single layer 2 network (e.g. routes on an Ethernet segment).

Selecting a network such as `internalnet` or `externalnet` in the resource tree displays its tabbed pane. By default, the tab displayed is the `Overview` tab. This gives a convenient overview of the IP addresses of items assigned in the selected network, grouped as nodes, switches, Power Distribution Units, GPU units, and other devices (figure 4.4).

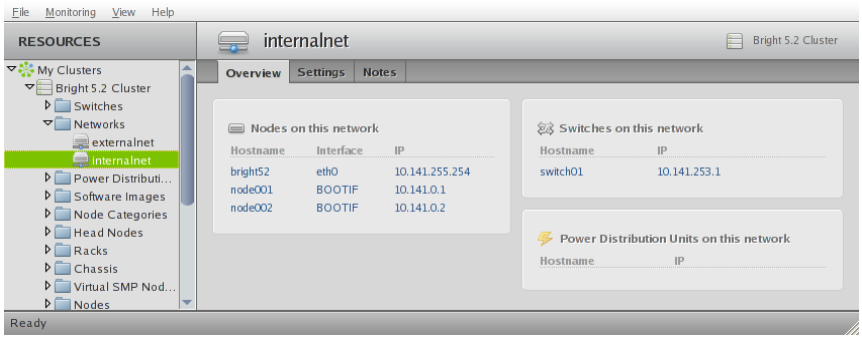

Figure 4.4: Network Overview

Selecting the `Settings` tab allows a number of network properties to be changed (figure 4.5).

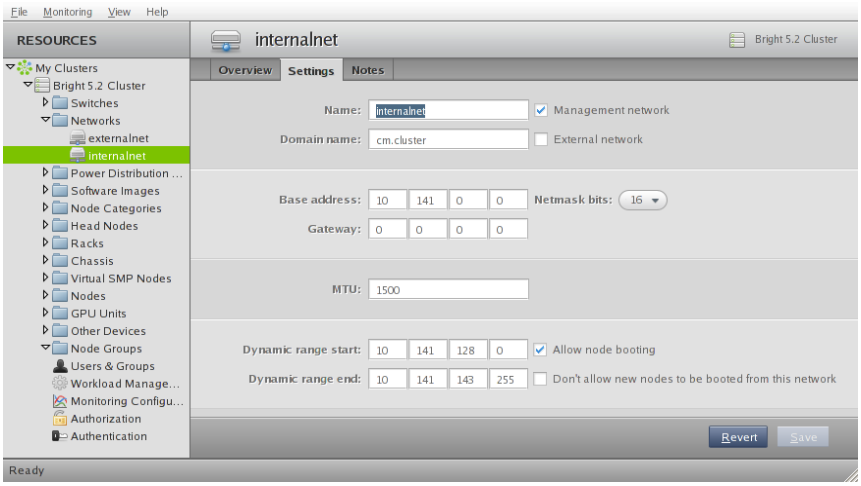

Figure 4.5: Network Settings

The properties of figure 4.5 are introduced in table 4.2.1.

Table 4.2.1: Network Configuration Settings

Property	Description
`Name`	Name of this network.
`Domain name`	DNS domain associated with the network.
`Management network`	Switch to treat the network as having nodes managed by the head node.
`External network`	Switch to treat the network as an external network.
`Base address`	Base address of the network (also known as the *network address*)
`Gateway`	Default route IP address
`Netmask bits`	Prefix-length, or number of bits in netmask. The part after the "/" in CIDR notation.
`MTU`	Maximum Transmission Unit. The maximum size of an IP packet transmitted without fragmenting.
`Dynamic range start/end`	Start/end IP addresses of the DHCP range temporarily used by nodes during PXE boot.
`Allow node booting`	Nodes set to boot from this network (useful in the case of nodes on multiple networks).
`Don't allow new nodes to boot from this network`	New nodes are not offered a PXE DHCP IP address from this network.

In basic networking concepts, a network is a range of IP addresses. The first address in the range is the *base address*. The length of the range, i.e. the *subnet*, is determined by the *netmask*, which uses CIDR notation. CIDR notation is the so-called / ("slash") representation, in which, for example, a CIDR notation of 192.168.0.1/28 implies an IP address of 192.168.0.1 with a traditional netmask of 255.255.255.240 applied to the 192.168.0.0 network. The netmask 255.255.255.240 implies that bits 28–32

of the 32-bit dotted-quad number 255.255.255.255 are unmasked, thereby implying a 4-bit-sized host range of 16 (i.e. 2^4) addresses.

The `sipcalc` utility installed on the head node is a useful tool for calculating or checking such IP subnet values (`man sipcalc` or `sipcalc -h` for help on this utility):

Example

```
user@brightcluster:~$ sipcalc 192.168.0.1/28
-[ipv4 : 192.168.0.1/28] - 0

[CIDR]
Host address            - 192.168.0.1
Host address (decimal)  - 3232235521
Host address (hex)      - C0A80001
Network address         - 192.168.0.0
Network mask            - 255.255.255.240
Network mask (bits)     - 28
Network mask (hex)      - FFFFFFF0
Broadcast address       - 192.168.0.15
Cisco wildcard          - 0.0.0.15
Addresses in network    - 16
Network range           - 192.168.0.0 - 192.168.0.15
Usable range            - 192.168.0.1 - 192.168.0.14
```

Every network has an associated DNS domain which can be used to access a device through a particular network. For `internalnet`, the default DNS domain is set to `cm.cluster`, which means that the hostname `node001.cm.cluster` can be used to access device `node001` through the primary internal network. If a dedicated storage network has been added with DNS domain `storage.cluster`, then `node001.storage.cluster` can be used to reach `node001` through the storage network. Internal DNS zones are generated automatically based on the network definitions and the defined nodes on these networks. For networks marked as external, no DNS zones are generated.

4.2.2 Adding Networks

The `Add` button in the networks overview tab of figure 4.3 can be used to add a new network. After the new network has been added, the `Settings` tab (figure 4.5) can be used to further configure the newly added network.

After a network has been added, it can be used in the configuration of network interfaces for devices.

The default assignment of networks (internalnet to `Management network` and externalnet to `External network`) can be changed in the `Settings` tab of the cluster object (figure 4.6).

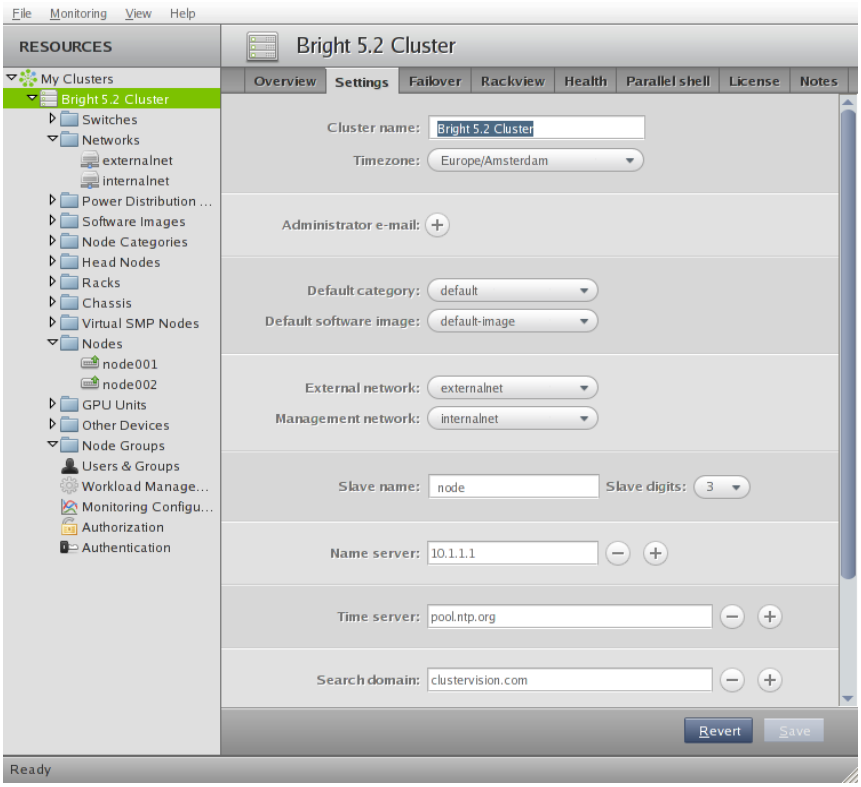

Figure 4.6: Cluster Settings

4.2.3 Changing Network Parameters

After both internal and external networks are defined, it may be necessary to change network-related parameters from their original or default installation settings.

Changing Head Node Hostname

Normally the name of a cluster is used as the hostname of the head node. To reach the `head node` from inside the cluster, the alias `master` may be used at all times. Setting the hostname of the head node itself to `master` is not recommended.

To change the hostname of the head node, the device object corresponding to the head node must be modified.

- In `cmgui`, the device listed under `Head Nodes` in the resource tree is selected and its `Settings` tab selected from the tabbed pane (figure 4.7). The hostname is changed by modifying the `Hostname` property and clicking on `Save`. When setting a hostname, a domain is not included.

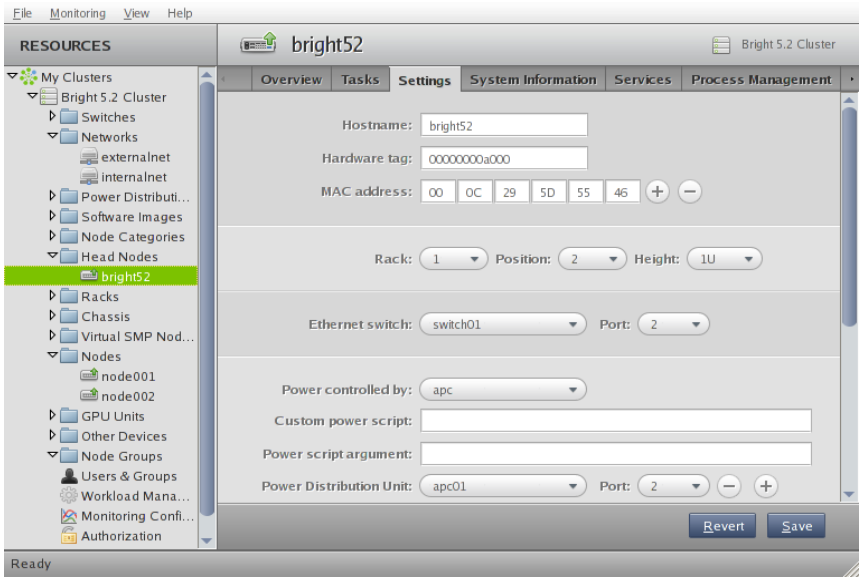

Figure 4.7: Head Node Settings

- The hostname of the head node can also be changed via `cmsh`:

  Example

  ```
  [root@bright52 ~]# cmsh
  [bright52]% device use master
  [bright52->device[bright52]]% set hostname foobar
  [foobar->device*[foobar*]]% commit
  [foobar->device[foobar]]% quit
  [root@bright52 ~]# sleep 30; hostname -f
  foobar.cm.cluster
  [root@bright52 ~]#
  ```

  The prompt string shows the new hostname when a new shell is started.

Changing External Network Parameters For The Cluster

The external network parameters of the cluster: When a cluster interacts with an external network, such as a company or a university network, its connection behavior is determined by the settings of two objects: firstly, the external network settings of the `Networks` resource, and secondly, by the cluster network settings.

1. **The external network object** contains the network settings for all objects configured to connect to the external network, for example, a head node. Network settings are configured in the `Settings` tab of the `Networks` resource of `cmgui`. Figure 4.5 shows a settings tab for when the `internalnet` item has been selected, but in the current case the `externalnet` item must be selected instead. The following parameters can then be configured:

   - the IP network parameters of the cluster (but not the IP address of the cluster):

  - `Base address`: the network address of the external network (the "IP address of the external network"). This is not to be confused with the IP address of the cluster, which is described shortly after this.
  - `Netmask bits`: the netmask size, or prefix-length, of the external network, in bits.
  - `Gateway`: the default route for the external network.
  - `Dynamic range start` and `Dynamic range end`: the start and end respectively of the DHCP range, if using DHCP to set the IP address of the cluster on the external network.

- the `Domain name`: the network domain (LAN domain, i.e. what domain machines on the external network use as their domain),
- network name (what the external network itself is called), by default this is `externalnet` on a newly installed Type 1 cluster,
- the `External network` checkbox: this is checked for a Type 1 cluster,
- and MTU size (the maximum value for a TCP/IP packet before it fragments on the external network—the default value is 1500).

2. **The cluster object** contains other network settings used to connect to the outside. These are configured in the `Settings` tab of the cluster object resource in `cmgui` (figure 4.6):

   - the external name servers used by the cluster to resolve external host names,
   - the DNS search domain (what the cluster uses as its domain),
   - and NTP time servers (used to synchronize the time on the cluster with standard time) and time zone settings.

Changing the networking parameters of a cluster (apart from the IP address of the cluster) therefore requires making changes in the settings of the two preceding objects.

The IP address of the cluster: The cluster object itself does not contain an IP address value. This is because it is the cluster network topology type that determines whether a direct interface exists from the cluster to the outside world. Thus, the IP address of the cluster in the Type 1, Type 2, and Type 3 configurations (section 2.3.6) is defined by the cluster interface that faces the outside world. For Type 1, this is the interface of the head node to the external network (figure 4.2). For Type 2 and Type 3 interfaces the cluster IP address is effectively that of an upstream router, and thus not a part of Bright Cluster Manager configuration. Thus, logically, the IP address of the cluster is not a part of the cluster object or external network object configuration.

For a Type 1 cluster, the head node IP address can be set in Bright Cluster Manager, separately from the cluster object settings. This is then the IP address of the cluster according to the outside world.

Setting the network parameters of the cluster and the head node IP address: These values can be set using `cmgui` or `cmsh`:

With `cmgui`: The associated cluster network settings tabs are accessed as shown in figure 4.8 for the external network object, and as shown in figure 4.6 for the cluster object.

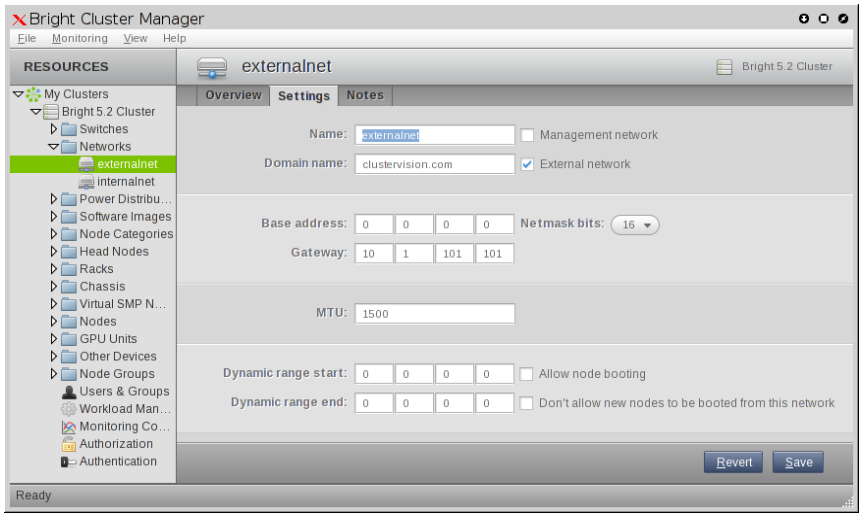

Figure 4.8: Network Settings For External Network

Setting the static IP address of the head node can be done by selecting the head node from the `Head Nodes` resources tab, then selecting the `Network Setup` tabbed pane, then selecting the interface for the address. Clicking on the `Edit` button opens up an editor that allows the IP address to be changed (figure 4.9).

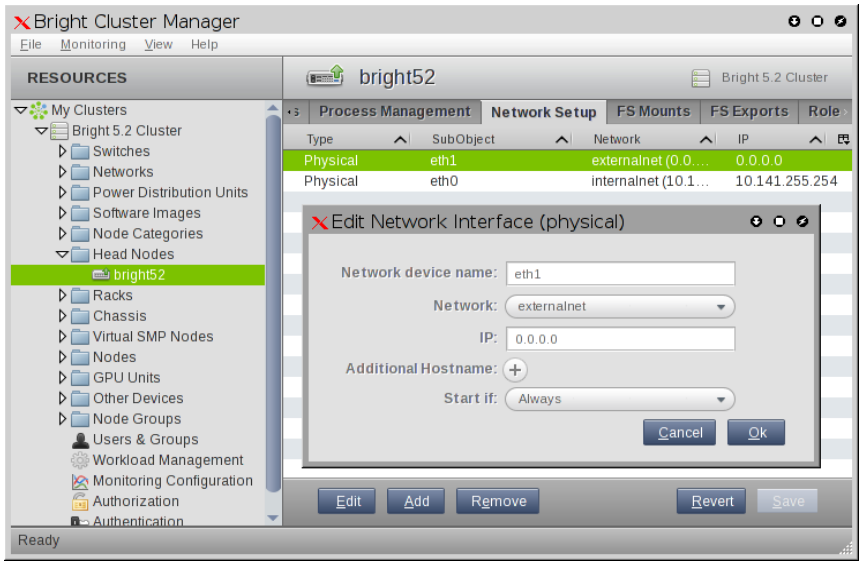

Figure 4.9: Setting The IP Address On A Head Node In `cmgui`

With `cmsh`: The preceding `cmgui` configuration can also be done in `cmsh`, using the `network`, `partition` and `device` modes, as in the

following example:

Example

```
[bright52]% network use externalnet
[bright52->network[externalnet]]% set baseaddress 192.168.1.0
[bright52->network*[externalnet*]]% set netmaskbits 24
[bright52->network*[externalnet*]]% set gateway 192.168.1.1
[bright52->network*[externalnet*]]% commit
[bright52->network[externalnet]]% partition use base
[bright52->partition[base]]% set nameservers 192.168.1.1
[bright52->partition*[base*]]% set searchdomains x.com y.com
[bright52->partition*[base*]]% append timeservers ntp.x.com
[bright52->partition*[base*]]% commit
[bright52->partition[base]]% device use master
[bright52->device[bright52]]% interfaces
[bright52->device[bright52]->interfaces]% use eth1
[bright52->device[bright52]->interfaces[eth1]]% set ip 192.168.1.176
[bright52->device[bright52]->interfaces*[eth1*]]% commit
[bright52->device[bright52]->interfaces[eth1]]%
```

After changing the external network configurations, a reboot of the head node is necessary to activate the changes.

Using DHCP to supply network values for the external interface: Connecting the cluster via DHCP on the external network is not generally recommended for production clusters. This is because DHCP-related issues can complicate networking troubleshooting compared with using static assignments.

For a Type 1 network, to make the cluster and head node use some of the DHCP-supplied external network values, the base address of `externalnet` and the external interface IP address of the head node must both be set to `0.0.0.0`. This can be done in `cmgui`, as shown in figures 4.8 and 4.9; or in `cmsh`, as shown in the preceding example in the `network` and `device` modes. The gateway address, the name server(s), and the external IP address of the head node are then obtained via a DHCP lease. Time server configuration for `externalnet` is not picked up from the DHCP server, having been set during installation (figure 2.20). The time servers can be changed using `cmgui` as in figure 4.6, or using `cmsh` in `partition` mode as in the preceding example. The time zone can be changed similarly.

It is usually sensible to reboot after implementing these changes in order to test the changes are working as expected.

Changing Internal Network Parameters For The Cluster

When a cluster interacts with the internal network that the compute nodes and other devices are on, its connection behavior with the devices on that network is determined by settings in:

1. the internal network of the `Networks` resource (page 71)
2. the cluster network for the internal network (page 72)
3. the individual device network interface (page 72)

4. the node categories network-related items for the device (page 73), in the case of the device being a regular node.

In more detail:

1. The internal network object: has the network settings for all devices connecting to the internal network, for example, a login node, a head node via its `internalnet` interface, or a managed switch on the internal network. For individual nodes, these settings are configured in the `Settings` tab of the `Networks` resource of `cmgui` (figure 4.5) for the following parameters:

- the IP network parameters of the internal network (but not the internal IP address):
  - "`Base address`": the internal network address of the cluster (the "IP address of the internal network"). This is not to be confused with the IP address of the internal network interface of the head node. The default value is `10.141.0.0`.
  - "`Netmask bits`": the netmask size, or prefix-length, of the internal network, in bits. The default value is `16`.
  - `Gateway`:the default gateway route for the internal network. If unset, or 0.0.0.0 (the default), then its value is decided by the DHCP server on the head node, and nodes are assigned a default gateway route of `10.141.255.254`, which corresponds to using the head node as a default gateway route for the interface of the regular node. The effect of this parameter is overriden by any default gateway route value set by the value of `Default gateway` in the node category.
  - "`Dynamic range start`" and "`Dynamic range end`": These are the DHCP ranges for nodes. DHCP is unset by default. When set, the internal nodes can use the dynamic IP address values assigned to the node by the node-installer. These values range by default from `10.141.128.0` to `10.141.143.255`.
  - `Allow node booting`: This allows nodes to boot from the the provisioning system controlled by CMDaemon. The parameter is normally set for the management network (that is the network over which CMDaemon communicates to manage nodes) but booting can instead be carried out over a separate physical non-management network. Booting over InfiniBand is one of the possibilities (section 6.1.3). Only if the `Allow node booting` option is ticked does ticking the "`Don't allow new nodes to be booted from this network`" checkbox have any effect, and stop new nodes from booting. New nodes are those nodes which are detected but the cluster cannot identify based on CMDaemon records. Details on booting, provisioning, and how a node is detected as new are described further in chapter 6.
- the "`domain name`" of the network. This is the LAN domain, which is the domain machines on this network use as their domain. By default, set to `cm.cluster`.

- the network name, or what the internal network is called. By default, set to `internalnet`.
- The `MTU` size, or the maximum value for a TCP/IP packet before it fragments on this network. By default, set to 1500.

2. The cluster object: has other network settings that the internal network in the cluster uses. These particulars are configured in the `Settings` tab of the cluster object resource in `cmgui` (figure 4.6):

- the "`Management network`". This is the name of the network over which CMDaemon manage the nodes. By default, set to `internalnet` for Type 1 and Type 2 networks, and `managementnet` in Type 3 networks. This can also be set at category level.
- the "`Node name`" can be set to decide the prefix part of the node name. By default, set to `node`.
- the "`Node digits`" can be set to decide the possible size of numbers used for suffix part of the node name. By default, set to 3.
- the "`Default category`". This sets the category the nodes are in by default. By default, it is set to `default`.
- the "`Default software image`". This sets the image the nodes use by default, By default, it is set to `default-image`.

3. The internal IP addresses and other internal interface values: The internal nodes can be configured in the "`Network Setup`" tab of the `Nodes` object resource in `cmgui`, which displays options in a similar way to figure 4.9. Network configuration can be done in a similar way to that described for external network configuration.

The items that can be set are:

- the `Network device name`: By default, this is set to `BOOTIF` for a node that boots from the same interface as the one from which it is provisioned.
- the `Network`: By default, this is set to a value of `internalnet`.
- the `IP` address: By default, this is automatically assigned a static value, in the range `10.141.0.1` to `10.141.255.255`, with the first node being given the first address. Using a static address instead of a DHCP-served dynamic address is recommended for busy clusters for reliability, although setting `0.0.0.0` allows dynamic DHCP addressing to be used.

  When using dynamic DHCP addressing, a start and end range in the internal network object is used, as explained earlier on page 71 in the section describing the internal network object.

  The static address can be changed manually, in case there is an IP address or node ID conflict due to an earlier inappropriate manual intervention.

- `Additional Hostname`: In the case of nodes this is in addition to the default node name set during provisioning. The node name set during provisioning takes a default form of `node<3 digit number>`, as explained earlier on page 72 in the section describing the cluster object settings.

4. Node category network values: are settings for the internal network that can be configured for node categories using the `Settings` tab in `cmgui` for that category of node. If there are node settings that can be configured in `cmgui` or `cmsh`, then the node settings override the corresponding category settings for those particular nodes.

The category properties involved in internal networking that can be set include:

- `Default gateway`: The default gateway route for nodes in the node category. If unset, or 0.0.0.0 (the default), then the node default gateway route is decided by the internal network object `Gateway` value. If the default gateway is set as a node category value, then nodes use the node category value as their default gateway route instead.
- `Management network`: The management network is the network used by CMDaemon to manage devices. The default setting is a property of the node object. It can be set as a category property.
- `Name server`: The default setting for the name server on all nodes is set by the node-installer to refer to the head node, and is not configurable using `cmgui` or `cmsh`. The name server can however be set as a category property to override the default value.
- `Time server`: The default setting for the time server on all nodes is set by the node-installer to refer to the head node, and is not configurable using `cmgui` or `cmsh`. The time server can however be set as a category property to override the default value.
- `Search domain`. The default setting for the search domain on all nodes is set by the node-installer to refer to the head node, and is not configurable using `cmgui` or `cmsh`. The search domain can however be set as a category property to override the default value.

After changing network configurations, a reboot of the node is necessary to activate the changes.

4.3 Configuring InfiniBand Interfaces

On clusters with an InfiniBand interconnect, the InfiniBand Host Channel Adapter (HCA) in each node must be configured before it can be used. This section describes how to set up the InfiniBand service on the nodes for regular use. Setting up InfiniBand for booting and provisioning purposes is described in Chapter 6.

4.3.1 Installing Software Packages

On a standard Bright Cluster Manager cluster, the OFED (OpenFabrics Enterprise Distribution) packages that are part of the Linux base distribution are used. These packages provide RDMA fabric technologies such

as InfiniBand and iWarp (RDMA over ethernet). By default, all relevant OFED packages are installed on the head node and software images. It is possible to replace the distribution OFED with an OFED provided by the Bright Cluster Manager repository or another custom version. The replacement can be for the entire cluster, or only for certain software images. Administrators may choose to switch to a different OFED version if the HCAs used are not supported by the distribution OFED version, or to increase performance by using an OFED version that has been optimized for a particular HCA. Installing the Bright Cluster Manager OFED packages is covered in section 12.6.

If the InfiniBand network is enabled during installation, then the `rdma` script runs during the `init` stage of booting up for the enabled nodes. For SLES and Linux distributions based on versions prior to Red Hat 6, the `openibd` script is used instead of the `rdma` script.

The `rdma` or `openibd` script takes care of loading the relevant InfiniBand HCA kernel modules. When adding an InfiniBand network after installation, it may be necessary to use `chkconfig` manually to configure the `rdma` or `openibd` script to be run at boot-time on the head node and inside the software images.

4.3.2 Subnet Managers

Every InfiniBand subnet requires at least one subnet manager to be running. The subnet manager takes care of routing, addressing and initialization on the InfiniBand fabric. Some InfiniBand switches include subnet managers. However, on large InfiniBand networks or in the absence of a switch-hosted subnet manager, a subnet manager needs to be started on at least one node inside of the cluster. When multiple subnet managers are started on the same InfiniBand subnet, one instance will become the active subnet manager whereas the other instances will remain in passive mode. It is recommended to run 2 subnet managers on all InfiniBand subnets to provide redundancy in case of failure.

On a Linux machine that is not running Bright Cluster Manager, an administrator sets a subnet manager service[1] to start at boot-time with a command such as: "`chkconfig opensm on`". However, for clusters managed by Bright Cluster Manager, a subnet manager is best set up using CMDaemon. There are two ways of setting CMDaemon to start up the subnet manager on a node at boot time:

1. by assigning a role.

   In `cmsh` this can be done with:

   ```
   [root@bright52 ~]# cmsh -c "device roles <node>; assign subnetmanager; \
   commit"
   ```

   where `<node>` is the name of a node on which it will run, for example: `bright52`, `node001`, `node002`...

   In `cmgui` the subnet manager role is assigned by selecting a head node or regular node from the resources tree, and assigning it the "`Subnet Manager Role`" from the "`Roles`" tab.

[1] usually `opensm`, but `opensmd` in SLES

2. by setting the service[1] up.

   In cmsh this is done with:

   Example

```
[root@bright52 ~]# cmsh
[bright52]% device services node001
[bright52->device[node001]->services]% add opensm
[bright52->device[node001]->services*[opensm*]]% set autostart yes
[bright52->device[node001]->services*[opensm*]]% set monitored yes
[bright52->device[node001]->services*[opensm*]]% commit
[bright52->device[node001]->services[opensm]]%
```

   In cmgui the subnet manager service is configured by selecting a head node or regular node from the resources tree, and adding the service to it.

When the head node in a cluster is equipped with an InfiniBand HCA, it is a good candidate to run as a subnet manager for smaller clusters.

On large clusters a dedicated node is recommended to run the subnet manager.

4.3.3 InfiniBand Network Settings

Although not strictly necessary, it is recommended that InfiniBand interfaces are assigned an IP address (i.e. IP over IB). First, a network object in the cluster management infrastructure should be created. The procedure for adding a network is described in section 4.2.2. The following settings are recommended as defaults:

Property	Value
Name	ibnet
Domain name	ib.cluster
External network	false
Base address	10.149.0.0
Netmask bits	16
Broadcast address	10.149.255.255

Once the network has been created all nodes must be assigned an InfiniBand interface on this network. The easiest method of doing this is to create the interface for one node device and then to clone that device several times.

For large clusters, a labor-saving way to do this is using the addinterface command (section 4.4.1) as follows:

```
[root@bright52 ~]# echo "device
addinterface -n node001..node150 physical ib0 ibnet 10.149.0.1
commit" | cmsh -x
```

When the head node is also equipped with an InfiniBand HCA, it is important that a corresponding interface is added and configured in the cluster management infrastructure.

Example

Assigning an IP address on the InfiniBand network to the head node:

```
[bright52->device[brigh52]->interfaces]% add physical ib0
[bright52->device[bright52]->interfaces*[ib0*]]% set network ibnet
[bright52->device[bright52]->interfaces*[ib0*]]% set ip 10.149.255.254
[bright52->device[bright52]->interfaces*[ib0*]]% commit
```

As with any change to the network setup, the head node needs to be restarted to make the above change active.

4.3.4 Verifying Connectivity

After all nodes have been restarted, the easiest way to verify connectivity is to use the `ping` utility

Example

Pinging node015 while logged in to node014 through the InfiniBand interconnect:

```
[root@node014 ~]# ping node015.ib.cluster
PING node015.ib.cluster (10.149.0.15) 56(84) bytes of data.
64 bytes from node015.ib.cluster (10.149.0.15): icmp_seq=1 ttl=64
time=0.086 ms
...
```

If the ping utility reports that ping replies are being received, the InfiniBand is operational. The `ping` utility is not intended to benchmark high speed interconnects. For this reason it is usually a good idea to perform more elaborate testing to verify that bandwidth and latency are within the expected range.

The quickest way to stress-test the InfiniBand interconnect is to use the Intel MPI Benchmark (IMB), which is installed by default in `/cm/shared/apps/imb/current`. The `setup.sh` script in this directory can be used to create a template in a user's home directory to start a run.

Example

Running the Intel MPI Benchmark using `openmpi` to evaluate performance of the InfiniBand interconnect between `node001` and `node002`:

```
[root@bright52 ~]# su - cmsupport
[cmsupport@bright52 ~]$ cd /cm/shared/apps/imb/current/
[cmsupport@bright52 current]$ ./setup.sh
[cmsupport@bright52 current]$ cd ~/BenchMarks/imb/3.2.2
[cmsupport@bright52 3.2.2]$ module load openmpi/gcc
[cmsupport@bright52 3.2.2]$ module initadd openmpi/gcc
[cmsupport@bright52 3.2.2]$ make -f make_mpi2
[cmsupport@bright52 3.2.2]$ mpirun -np 2 -machinefile ../nodes IMB-MPI1 PingPong
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         0.78         0.00
```

```
     1         1000         1.08         0.88
     2         1000         1.07         1.78
     4         1000         1.08         3.53
     8         1000         1.08         7.06
    16         1000         1.16        13.16
    32         1000         1.17        26.15
    64         1000         1.17        52.12
   128         1000         1.20       101.39
   256         1000         1.37       177.62
   512         1000         1.69       288.67
  1024         1000         2.30       425.34
  2048         1000         3.46       564.73
  4096         1000         7.37       530.30
  8192         1000        11.21       697.20
 16384         1000        21.63       722.24
 32768         1000        42.19       740.72
 65536          640        70.09       891.69
131072          320       125.46       996.35
262144          160       238.04      1050.25
524288           80       500.76       998.48
1048576          40      1065.28       938.72
2097152          20      2033.13       983.71
4194304          10      3887.00      1029.07

# All processes entering MPI_Finalize
```

To run on nodes other than `node001` and `node002`, the `../nodes` file must be modified to contain different hostnames. To perform a more extensive run, the `PingPong` argument should be omitted.

4.4 Configuring IPMI Interfaces

Bright Cluster Manager also takes care of the initialization and configuration of the baseboard management controller (BMC) that may be present on devices. The IPMI or iLO interface that is exposed by a BMC is treated in the cluster management infrastructure as a special type of network interface belonging to a device. In the most common setup a dedicated network (i.e. IP subnet) is created for IPMI communication. The `10.148.0.0/16` network is used by default for IPMI interfaces by Bright Cluster Manager.

4.4.1 IPMI And Network Settings

The first step in setting up IPMI is to add the IPMI network as a network object in the cluster management infrastructure. The procedure for adding a network is described in section 4.2.2. The following settings are recommended as defaults:

Property	Value
Name	ipminet
Domain name	ipmi.cluster
External network	false
Base address	10.148.0.0
Netmask bits	16
Broadcast address	10.148.255.255

Once the network has been created all nodes must be assigned an IPMI interface on this network. The easiest method of doing this is to create the interface for one node device and then to clone that device several times.

For larger clusters this can be laborious, and a simple `bash` loop can be used to do the job instead:

```
[bright52 ~]# for ((i=1; i<=150; i++)) do
echo "
device interfaces node$(printf '%03d' $i)
add ipmi ipmi0
set network ipminet
set ip 10.148.0.$i
commit"; done | cmsh -x     # -x usefully echoes what is piped into cmsh
```

The preceding loop can conveniently be replaced with the `addinterface` command, run from within the `device` mode of `cmsh`:

```
[bright52 ~]# echo "
device
addinterface -n node001..node150 ipmi ipmi0 ipminet 10.148.0.1
commit" | cmsh -x
```

The `help` text for `addinterface` gives more details on how to use it.

In order to be able to communicate with the IPMI interfaces, the head node also needs an interface on the IPMI network. Depending on how the IPMI interfaces are physically connected to the head node, the head node has to be assigned an IP address on the IPMI network one way or another. There are two possibilities for how the IPMI interface are physically connected:

- When the IPMI interfaces are connected to the primary internal network, the head node should be assigned an alias interface configured with an IP address on the IPMI network.
- When the IPMI interfaces are connected to a dedicated physical network, the head node must also be physically connected to this network. A physical interface must be added and configured with an IP address on the IPMI network.

Example

Assigning an IP address on the IPMI network to the head node using an alias interface:

```
[bright52->device[bright52]->interfaces]% add alias eth0:0
[bright52->device[bright52]->interfaces*[eth0:0*]]% set network ipminet
[bright52->device[bright52]->interfaces*[eth0:0*]]% set ip 10.148.255.254
[bright52->device[bright52]->interfaces*[eth0:0*]]% commit
[bright52->device[bright52]->interfaces[eth0:0]]%
Mon Dec 6 05:45:05 bright52: Reboot required: Interfaces have been modified
[bright52->device[bright52]->interfaces[eth0:0]]% quit
[root@bright52 ~]# /etc/init.d/network restart
```

As with any change to the network setup, the head node needs to be restarted to make the above change active, although in this particular case restarting the `network` service would suffice.

4.4.2 IPMI Authentication

The node-installer described in Chapter 6 is responsible for the initialization and configuration of the IPMI interface of a device. In addition to a number of network-related settings, the node-installer also configures IPMI authentication credentials. By default IPMI interfaces are configured with username `ADMIN` and a random password that is generated during the installation of the head node. The values can be read with the "`get`" command of `cmsh` in `partition` mode.

Changing the IPMI authentication credentials is currently only possible through `cmsh`.

For example, the current values of the IPMI username and password for the entire cluster can be obtained and changed as follows:

Example

```
[bright52]% partition use base
[bright52->partition[base]]% get ipmiusername
ADMIN
[bright52->partition[base]]% get ipmipassword
Za4ohni1ohMa2zew
[bright52->partition[base]]% set ipmiusername ipmiadmin
[bright52->partition*[base*]]% set ipmipassword
enter new password: ******
retype new password: ******
[bright52->partition*[base*]]% commit
[bright52->partition[base]]%
```

It is possible to change the authentication credentials cluster-wide or by category. Category settings override cluster-wide settings. The relevant properties are:

Property	Description
`IPMI User ID`	User type. Normally set to 2 for administrator access.
`IPMI User Name`	User name
`IPMI Password`	Password for specified user name

...continues

...continued

Property	Description

The cluster management infrastructure stores the configured IPMI username and password not just to configure the IPMI interface from the node-installer. The information is also used to authenticate to the IPMI interface once it has been brought up, in order to perform IPMI management operations (e.g. power cycling nodes and collecting hardware metrics).

4.5 Configuring Switches And PDUs

4.5.1 Configuring With The Manufacturer's Configuration Interface

Network switches and PDUs that will be used as part of the cluster should be configured with the PDU/switch configuration interface described in the PDU/switch documentation supplied by the manufacturer. Typically the interface is accessed by connecting via a web browser or telnet to an IP address preset by the manufacturer.

The IP settings of the PDU/switch must be configured to match the settings of the device in the cluster management software.

- In `cmgui` this is done by selecting the `Switches` resource, selecting the particular switch from within that resource, then selecting the associated `Settings` tab (figure 4.10). The IP address can then be set and saved.
- In `cmsh` this can be done in `device` mode, with a `set` command:

  Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright5522->device]% set switch01 ip 10.141.253.2
[bright52->device*]% commit
```

4.5.2 Configuring SNMP

Moreover, in order to allow the cluster management software to communicate with the switch or PDU, SNMP must be enabled and the SNMP community strings should be configured correctly. By default, the SNMP community strings for switches and PDUs are set to `public` and `private` for respectively read and write access. If different SNMP community strings have been configured in the switch or PDU, the `readstring` and `writestring` properties of the corresponding switch device should be changed.

Example

```
[bright52]% device use switch01
[bright52->device[switch01]]% get readstring
public
[bright52->device[switch01]]% get writestring
private
[bright52->device[switch01]]% set readstring public2
[bright52->device*[switch01*]]% set writestring private2
[bright52->device*[switch01*]]% commit
```

4.5.3 Uplink Ports

Uplink ports are switch ports that are connected to other switches. CMDaemon must be told about any switch ports that are uplink ports, or the traffic passing through an uplink port will lead to mistakes in what CMDaemon knows about port and MAC correspondence. Uplink ports are thus ports that CMDaemon is told to ignore.

To inform CMDaemon about what ports are uplink ports, `cmgui` or `cmsh` are used:

- In `cmgui`, the switch is selected from the `Switches` folder, and the `Settings` tabbed pane is opened (figure 4.10). The port number corresponding to uplink port number is filled in the blank field beside the "`Uplink:`" label. More uplinks can be appended by clicking on the ⊕ widget. The state is saved with the `Save` button.

Figure 4.10: Notifying CMDaemon About Uplinks With `cmgui`

- In `cmsh`, the switch is accessed from the `device` mode. The uplink port numbers can be appended one-by-one with the `append` command, or `set` in one go by using space-separated numbers.

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->device]% set switch01 uplinks 15 16
[bright52->device*]% set switch02 uplinks 01
[bright52->device*]% commit
successfully committed 3 Devices
```

4.5.4 The `showport` MAC Address-Port Matching Tool

The `showport` command can be used in troubleshooting network topology issues, as well as checking and setting up new nodes (section 6.4.2).

Basic Use Of `showport`

In the `device` mode of `cmsh` is the `showport` command, which works out which ports on which switch are associated with a specified MAC address.

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->device]% showport 00:30:48:30:73:92
[bright52->device]% switch01:12
```

When running showport, CMDaemon on the head node queries all switches until a match is found.

If a switch is also specified using the "-s" option, then the query is carried out for that switch first. Thus the preceding example can also be specified as:

```
[bright52->device]% showport -s switch01 00:30:48:30:73:92
[bright52->device]% switch01:12
```

If there is no port number returned for the specified switch, then the scan continues on other switches.

Mapping All Port Connections In The Cluster With showport

A list indicating the port connections and switches for all connected devices that are up can be generated using this script:

Example

```
#!/bin/bash
for nodename in $(cmsh -c "device; foreach * (get hostname)")
do
  macad=$(cmsh -c "device use $nodename; get mac")
  echo -n "$macad $nodename "
  cmsh -c "device showport $macad"
done
```

The script may take a while to finish its run. It gives an output like:

Example

```
00:00:00:00:00:00 switch01: No ethernet switch found connected to this m\
ac address
00:30:48:30:73:92 bright52: switch01:12
00:26:6C:F2:AD:54 node001: switch01:1
00:00:00:00:00:00 node002: No ethernet switch found connected to this ma\
c address
```

4.6 Disk Layouts: Disked, Semi-Diskless, And Diskless Node Configuration

Configuring the disk layout for head and regular nodes is done as part of the initial setup (section 2.3.14). The disk layout can also be re-configured for regular nodes once the cluster is running.

4.6.1 Disk Layouts

A disk layout is specified using an XML schema (appendix D.1). Possible disk layouts include the following:

- Default layout (appendix D.2)
- RAID setup (appendix D.3)
- LVM setup (appendix D.5)
- Diskless setup (appendix D.6)
- Semi-diskless setup (appendix D.7)

4.6.2 Disk Layout Assertions

Disk layouts can be set to *assert*

- that particular hardware be used, using XML element tags such as `vendor` or `requiredSize` (appendix D.8)
- custom assertions using an XML `assert` element tag to run scripts placed in CDATA sections (appendix D.9)

4.6.3 Changing Disk Layouts

A disk layout applied to a category of nodes is inherited by default by the nodes in that category. A disk layout that is then applied to an individual node within that category overrides the category setting. This is an example of the standard behavior for categories, as mentioned in section 3.1.3.

By default, the cluster is configured with a standard layout specified in section D.2. The layouts can be accessed from `cmgui` or `cmsh`, as is illustrated by the example in section 4.6.4, which covers changing a node from disked to diskless mode:

4.6.4 Changing A Disk Layout From Disked To Diskless

The XML schema for a node configured for diskless operation is shown in appendix D.6. This can often be deployed as is, or it can be modified during deployment using `cmgui` or `cmsh` as follows:

Changing A Disk Layout Using `cmgui`

To change a disk layout with `cmgui`, the current disk layout is accessed by selecting a node category or a specific node from the resource tree. The "`Disk Setup`" tab is then selected. Clicking on the `Load` button shows several possible configurations that can be loaded up, and if desired, edited to suit the situation (figure 4.11). To switch from the existing disk

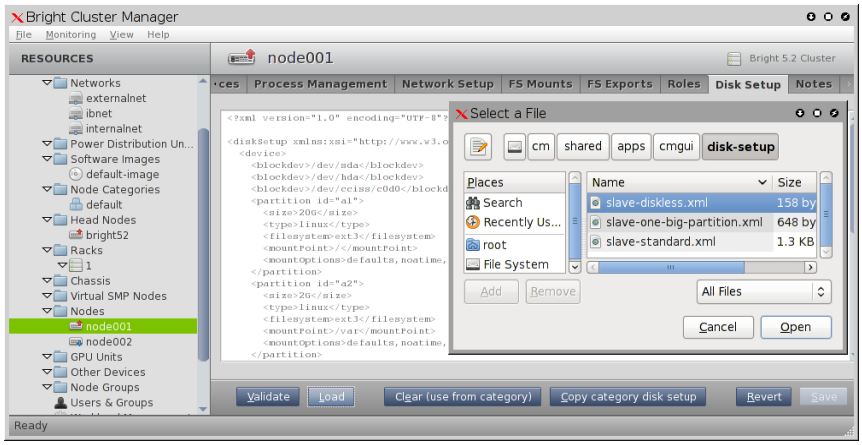

Figure 4.11: Changing A Disked Node To A Diskless Node With `cmgui`

layout to a diskless one, the diskless XML configuration is loaded and saved to the node or node category.

Changing A Disk Layout Using `cmsh`

To edit an existing disk layout from within `cmsh`, the existing XML configuration is accessed by editing the `disksetup` property in `device` mode

for a particular node, or by editing the disksetup property in category mode for a particular category. Editing is done using the set command, which opens up a text editor:

Example

```
[root@bright52 ~]# cmsh
[bright52]% device use node001
[bright52->device[node001]]% set disksetup
```

After editing and saving the XML configuration, the change is then committed to CMDaemon with the commit command.

Instead of editing an existing disk layout, another XML configuration can also be assigned. A diskless configuration may be chosen and set as follows:

Example

```
[bright52->device[node001]]% set disksetup /cm/shared/apps/cmgui/disk-s\
etup/slave-diskless.xml
```

In this example, after committing the change and rebooting the node, the node then functions without using its own disk.

4.7 Configuring NFS Volume Exports And Mounts

NFS allows unix NFS clients shared access to a file system on an NFS server. The accessed file system is called an NFS volume by remote machines. The NFS server exports the filesystem to selected hosts or networks, and the clients can then mount the exported volume locally.

An unformatted filesystem cannot be used. The drive must be partitioned beforehand with fdisk or similar partitioning tools, and its filesystem formatted with mkfs or similar before it can be exported.

In Bright Cluster Manager, the head node is typically used to export an NFS volume to the regular nodes, and the regular nodes then mount the volume locally.

If auto-mounting is used, then the configuration files for exporting should be set up on the NFS server, and the mount configurations set up on the software images. The service "autofs" or the equivalent can be set up using cmgui via the "Services" tab (section 4.8) on the head node.

The rest of this section describes the configuration of NFS for static mounts, using cmgui or cmsh.

4.7.1 Exporting A Filesystem Using cmgui And cmsh

Exporting A Filesystem Using cmgui

As an example, if an NFS volume exists at "bright52:/modeldata" it can be exported using cmgui as follows:

The head node is selected from under the "Head Nodes" resource, and the "FS Exports" tab is then selected. This shows the list of exports (figure 4.12).

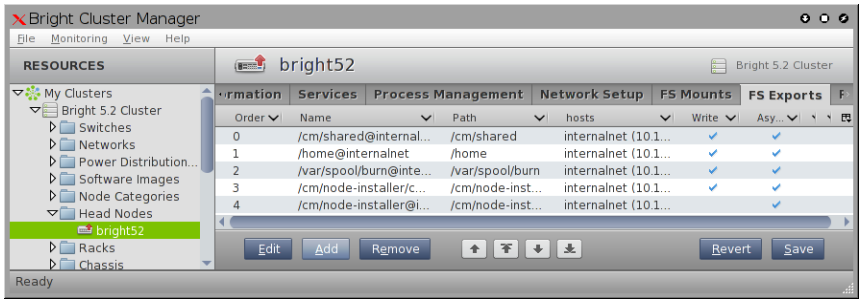

Figure 4.12: NFS Exports From A Head Node Viewed Using `cmgui`

Using the `Add` button, a new entry (figure 4.13) can be configured with values:

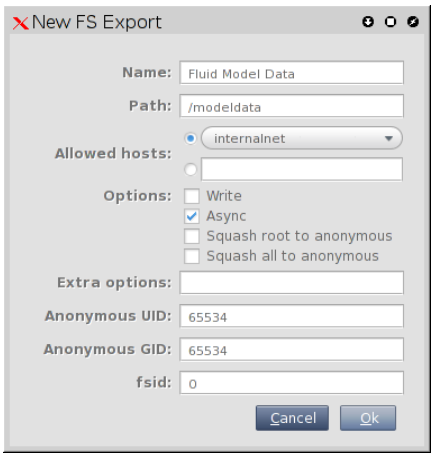

Figure 4.13: Setting Up An NFS Export Using `cmgui`

For this example, the value for "`Name`" is set arbitrarily to "`Fluid Model Data`", the value for `Path` is set to `/modeldata`, and the value for "`Allowed hosts`" is set from the selection menu to allowing access to `internalnet` (this is by default `10.141.0.0/16` in CIDR notation).

By leaving the `Write` option disabled, read-only access is kept. Saving this means the NFS server now provides NFS access to this file system for `internalnet`.

The network can be set to other values using CIDR notation, and also to particular hosts such as just node001 and node002, by specifying a value of "`node001 node002`" instead. Other settings and options are also possible and are given in detail in the man pages for `exports(5)`.

Exporting A Filesystem Using `cmsh`

The equivalent to the preceding `cmgui` NFS export procedure can be done in `cmsh` by using the `fsexports` submode on the head node (some output elided):

Example

```
[root@bright52 ~]# cmsh
[bright52]% device use bright52
[bright52->device[bright52]]% fsexports
[...->fsexports]% add "Fluid Model Data"
[...->fsexports*[Fluid Model Data*]]% set path /modeldata
```

```
[...[Fluid Model Data*]]% set hosts 10.141.0.0/16
[...[Fluid Model Data*]]% commit
[...->fsexports[Fluid Model Data]]% list | grep Fluid
Name (key)           Path            Hosts            Write
-------------------  -------------   --------------   ------
Fluid Model Data     /modeldata      10.141.0.0/16    no
```

4.7.2 Mounting A Filesystem Using `cmgui` And `cmsh`

Continuing on with the export example from the preceding section, the administrator decides to mount the remote filesystem over the default category of nodes. Nodes can also mount the remote filesystem individually, but that is usually not a common requirement in a cluster. The administrator also decides not to re-use the exported name from the head node. That is, the remote mount name `modeldata` is not used locally, even though NFS allows this and many administrators prefer to do this. Instead, a local mount name of `/modeldatagpu` is used, perhaps because it avoids confusion about which filesystem is local to a person who is logged in, and perhaps to emphasize the volume is being mounted by nodes with GPUs.

Mounting A Filesystem Using `cmgui`

Thus, in `cmgui`, values for the remote mount point (`bright52:/modeldata`), the file system type (`nfs`), and the local mount point (`/modeldatagpu`) can be set in category mode, while the remaining options stay at their default values (figure 4.14).

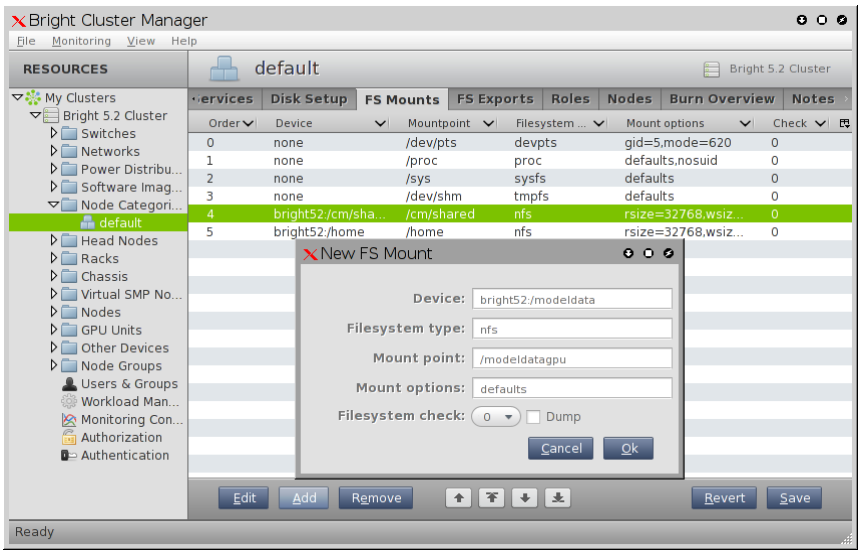

Figure 4.14: Setting Up NFS Mounts On A Node Category Using `cmgui`

Clicking on the `Ok` button saves the values, creating the local mount point, and the volume can now be accessed by nodes in that category.

Mounting A Filesystem Using `cmsh`

The equivalent to the preceding `cmgui` NFS mount procedure can be done in `cmsh` by using the `fsmounts` submode, for example on the `default` category. The `add` method under the `fsmounts` submode sets the mountpoint path, in this case `/modeldatagpu` (some output elided):

Example

```
[root@bright52 ~]# cmsh
[bright52]% category use default
[bright52->category[default]]% fsmounts
[bright52->category[default]->fsmounts]% add /modeldatagpu
[bright52->...*[/modeldatagpu*]]% set device bright52:/modeldata
[bright52->...*[/modeldatagpu*]]% set filesystem nfs
[bright52->category*[default*]->fsmounts*[/modeldatagpu*]]% commit
[bright52->category[default]->fsmounts[/modeldatagpu]]%
Device                  Mountpoint (key)     Filesystem
----------------------  ------------------   ----------
...
bright52:/modeldata     /modeldatagpu        nfs
[bright52->category[default]->fsmounts[/modeldatagpu]]% show
Parameter             Value
--------------------  ---------------------
Device                bright52:/modeldata
Dump                  no
Filesystem            nfs
Filesystem Check      0
Mount options         defaults
Mountpoint            /modeldatagpu
```

General Considerations On Mounting A Filesystem

There may be a requirement to segregate the access of nodes. For example, in the case of the preceding, because some nodes have no associated GPUs.

Besides the "`Allowed hosts`" options of NFS exports mentioned earlier in section 4.7.1, Bright Cluster Manager offers two more methods to fine tune node access to mount points:

- Nodes can be placed in another category that does not have the mount point.
- Nodes can have the mount point set, not by category, but per device within the `Nodes` resource. For this, the administrator must ensure that nodes that should have access have the mount point explicitly set.

Other considerations on mounting are that:

- When adding a mount point object:
  - The settings take effect right away by default on the nodes or node categories.
  - If `noauto` is set as a mount option, then the option only takes effect on explicitly mounting the filesystem.
  - If "`AutomaticMountAll=0`" is set as a CMDaemon option, then `/etc/fstab` is written, but the `mount -a` command is not run by CMDaemon. However, since `mount -a` is run by the distribution init script system during boot, a reboot of the node will implement a mount change.

- While a mount point object may have been removed, umount does not take place until reboot, to prevent mount changes outside of the cluster manager.
- When manipulating mount points, the administrator should be aware which mount points are inherited by category, and which are set for the individual node.
  - In `cmgui`, viewing category-inherited mount points for an individual node requires checking the checkbox for "`Display category inherited mounts`" in the `FS Mounts` tabbed view for that node (figure 4.15).

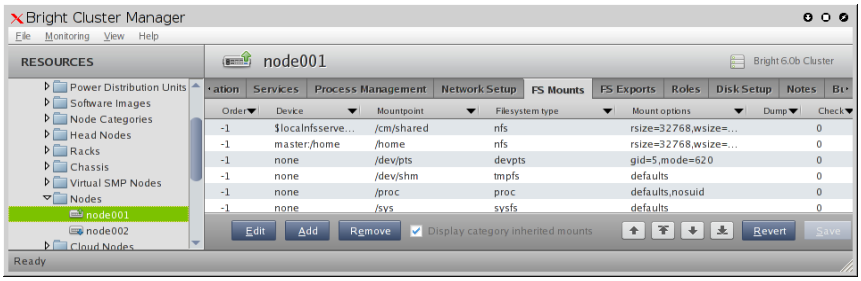

Figure 4.15: Display Of Category Inherited Mounts In `cmgui`

  - In `cmsh`, the category a mount belongs to is displayed in brackets. This is displayed from within the `fsmounts` submode of the `device` mode for a specified node:

Example

```
[root@bright52 ~]# cmsh -c "device; fsmounts node001; list"

Device                     Mountpoint (key)     Filesystem
-------------------------- -------------------- ----------
[default] none             /dev/pts             devpts
[default] none             /proc                proc
[default] none             /sys                 sysfs
[default] none             /dev/shm             tmpfs
[default] $localnfsserv+ /cm/shared             nfs
[default] bright52:/home /home                  nfs
bright52:/cm/shared/exa+ /home/examples         nfs
[root@bright52 ~]#
```

To remove a mount point defined at category level for a node, it must be removed from within the category, and not from the specific node.

Mount Order Considerations

Care is sometimes needed in deciding the order in which mounts are carried out.

- For example, if both `/usr/share/doc` and a replacement directory subtree `/usr/share/doc/compat-gcc-34-3.4.6java` are to be used, then the stacking order should be that `/usr/share/doc` is mounted first. This order ensures that the replacement directory subtree overlays the first mount. If, instead, the replacement directory were the first mount, then it would be overlaid, inaccessible, and inactive.

- There may also be dependencies between the subtrees to consider, some of which may prevent the start up of applications and services until they are resolved. In some cases, resolution may be quite involved.

The order in which such mounts are mounted can be modified with the `up` and `down` commands within the `fsmounts` submode of `cmsh`, or by using the arrow buttons at the bottom of the `FS Mounts` tabbed pane.

4.7.3 Mounting A Filesystem Subtree For A Diskless Node Over NFS

NFS Vs `tmpfs` For Diskless Nodes

For diskless nodes (appendix D.6), the software image (section 3.1.2) is typically installed from a provisioning node by the node-installer during the provisioning stage, and held as a filesystem in RAM on the diskless node with the `tmpfs` filesystem type.

It can be worthwhile to replace subtrees under the diskless node filesystem held in RAM with subtrees provided over NFS. This can be particularly worthwhile for less frequently accessed parts of the diskless node filesystem. This is because, although providing the files over NFS is much slower than accessing it from RAM, it has the benefit of freeing up RAM for tasks and jobs that run on diskless nodes, thereby increasing the cluster capacity.

An alternative "semi-diskless" way to free up RAM is to use a local disk on the node itself for supplying the subtrees. This is outlined in appendix D.7.

Moving A Filesystem Subtree Out Of `tmpfs` To NFS

To carry out subtree provisioning over NFS, the subtrees are exported and mounted using the methods outlined in the previous examples in sections 4.7.1 and 4.7.2. For the diskless case, the exported filesystem subtree is thus a particular path under `/cm/images/<image>`[2] on the provisioning node, and the subtree is mounted accordingly under `/` on the diskless node.

These paths cannot be mounted in Bright Cluster Manager 5.2:

`/bin`, `/dev`, `/etc`, `/proc`, `/sbin`, `/sys`, `/tmp`, and `/var`.

When `cmgui` or `cmsh` are used to manage the NFS export and mount of the subtree filesystem, then `tmpfs` on the diskless node is automatically reduced in size due to automatically excluding the subtree from `tmpfs` during provisioning.

An example might be to export `/cm/images/default-image` from the head node, and mount the directory available under it, `usr/share/doc` at a mount point `/usr/share/doc` on the diskless node. In `cmsh`, such an export can be done by creating a software image object `defaultimage` with the following indicated properties (some output elided):

Example

```
[root@bright52 ~]# cmsh
```

[2] by default <*image*> is `default-image` on a newly-installed cluster

```
[bright52]% device use bright52; fsexports
[bright52->device[bright52]->fsexports]% add defaultimage
[br...defaultimage*]]% set path /cm/images/default-image
[br...defaultimage*]]% set hosts 10.141.0.0/16
[br...defaultimage*]]% commit
[br...defaultimage]]% list | grep defaultimage
Name (key)        Path                       Hosts          Write
----------------  -------------------------  -------------  -----
defaultimage      /cm/images/default-image   10.141.0.0/16  no
```

As the output to `list` shows, the NFS export should be kept read-only, which is the default. Appropriate parts of the export can then be mounted by a node or node category. The mount is defined by setting the mount point, the `nfs` filesystem property, and the export device. For example, for a node category (some output elided):

```
[br...defaultimage]]% category use default
[bright52->category[default]]% fsmounts
[bright52->category[default]->fsmounts]% add /usr/share/doc
[bright52->...*[/usr/share/doc*]]% set device bright52:/cm/images/defau\
lt-image/user/share/doc
[bright52->...*[/usr/share/doc*]]% set filesystem nfs
[bright52->category*[default*]->fsmounts*[/usr/share/doc*]]% commit
[bright52->category[default]->fsmounts[/usr/share/doc]]% list
Device                           Mountpoint (key)     Filesystem
-------------------------------  -------------------  ----------
...                              ...                  ...
bright52:/cm/images/usr/share/doc /usr/share/doc      nfs
[bright52->category[default]->fsmounts[/usr/share/doc]]% show
Parameter          Value
-----------------  ------------------------------------------------
Device             bright52:/cm/images/default-image/usr/share/doc
Dump               no
Filesystem         nfs
Filesystem Check   0
Mount options      defaults
Mountpoint         /usr/share/doc
```

Other mount points can be also be added according to the judgment of the system administrator. Some consideration of mount order may be needed, as discussed on page 88 under the subheading "Mount Order Considerations".

An Example Of Several NFS Subtree Mounts

The following mounts save about 500MB from `tmpfs` on a diskless node with CentOS 6, as can be worked out from the following subtree sizes:

```
[root@bright52 ~]# cd /cm/images/default-image/
[root@bright52 default-image]# du -sh usr/share/locale usr/java usr/sha\
re/doc usr/src
262M    usr/share/locale
78M     usr/java
107M    usr/share/doc
45M     usr/src
```

The filesystem mounts can then be created using the techniques in this section. After doing that, the result is then something like (some lines omitted):

```
[root@bright52 default-image]# cmsh
[bright52]% category use default; fsmounts
[bright52->category[default]->fsmounts]% list -f device:53,mountpoint:17
device                                                mountpoint (key)
----------------------------------------------------  -----------------
...                                                   ...
master:/cm/shared                                     /cm/shared
master:/home                                          /home
bright52:/cm/images/default-image/usr/share/locale    /usr/share/locale
bright52:/cm/images/default-image/usr/java            /usr/java
bright52:/cm/images/default-image/usr/share/doc       /usr/share/doc
bright52:/cm/images/default-image/usr/src             /usr/src
[bright52->category[default]->fsmounts]%
```

Diskless nodes that have NFS subtree configuration carried out on them can be rebooted to start them up with the new configuration.

4.8 Managing And Configuring Services

4.8.1 Why Use The Cluster Manager For Services?

Unix services can be managed from the command line using the standard distribution tools, `chkconfig` and `/etc/init.d/<service name>`, where <*service name*> indicates a service such as `mysql`, `nfs`, `postfix` and so on.

The services can also be managed using Bright Cluster Manager's `cmgui` and `cmsh` tools. An additional convenience that comes with the cluster manager tools is that some functions useful for cluster management are very easily configured, whether on the head node, a regular node, or for a node category. These functions are:

- `monitoring`: monitoring a service so that information is displayed and logged about whether it is running or not
- `autostart`: restarting a service on failure
- `runif`: (only honored for head nodes) whether the service should run with a state of:
  - `active`: run on the active node only
  - `passive`: run on the passive only
  - `always`: run both on the active and passive
  - `preferpassive`: preferentially run on the passive if it is available

The details of a service configuration remain part of the configuration methods of the service software itself. Bright Cluster Manager configuration handles general services only at the generic service level to which all unix services conform.

4.8.2 Managing And Configuring Services—An Example

After having installed the CUPS software, with, for example, "`yum install cups`", the CUPS service can be started up in several ways:

From The Regular Shell, Outside Of CMDaemon

Standard unix commands from the bash prompt work, as shown by this session:

```
[root@bright52 ~]# chkconfig cups on
[root@bright52 ~]# /etc/init.d/cups start
```

Using cmsh

The following session illustrates adding the CUPS service from within device mode and the services submode. The device in this case is a regular node, node001, but a head node can also be chosen. Monitoring and autostarting are also set in the session:

```
[bright52]% device services node001
[bright52->device[node001]->services]% add cups
[bright52->device*[node001*]->services*[cups*]]% show
Parameter                      Value
------------------------------ ----------------------------------
Autostart                      no
Belongs to role                no
Monitored                      no
Revision
Run if                         ALWAYS
Service                        cups
[bright52->device*[node001*]->services*[cups*]]% set monitored on
[bright52->device*[node001*]->services*[cups*]]% set autostart on
[bright52->device*[node001*]->services*[cups*]]% commit
[bright52->device[node001]->services[cups]]%
Tue Jul 26 16:42:17 2011 [notice] node001: Service cups was started
[bright52->device[node001]->services[cups]]%
```

Within cmsh, the start, stop, restart, and reload options to the "/etc/init.d/*<service name>*" command can be used to manage the service at the services submode level. For example, continuing with the preceding session, stopping the CUPS service can be done by running the cups service command with the stop option as follows:

```
[bright52->device[node001]->services[cups]]% stop
Wed Jul 27 14:42:41 2011 [notice] node001: Service cups was stopped
Successfully stopped service cups on: node001
[bright52->device[node001]->services[cups]]%
```

The reset option is not a service option in the regular shell, but is used by CMDaemon (cmsh and cmgui) to clear a Failed state of a service as seen by the monitoring system.

The monitoring system sets the state of a service to Failed if 10 restarts of the service in a row fail. CMDaemon then no longer attempts to restart the service until the reset option is executed.

The CUPS service can also be added for a node category from category mode:

```
[root@bright52 ~]# cmsh
[bright52]% category services default
[bright52->category[default]->services]% add cups
```

As before, after adding the service, the monitoring and autostart functions can be set for the service. Also as before, the options to the "`init.d/<service name>`" command and also the `reset` option can be used to manage the service at the `services` submode level. The settings apply to the entire node category:

Example

```
[bright52->category*[default*]->services*[cups*]]% show
[bright52->category*[default*]->services*[cups*]]% set autostart yes
[bright52->category*[default*]->services*[cups*]]% set monitored yes
[bright52->category*[default*]->services*[cups*]]% commit
[bright52->category[default]->services[cups]]%
Tue Aug 23 12:23:17 2011 [notice] node001: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node002: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node003: Service cups was started
Tue Aug 23 12:23:17 2011 [notice] node004: Service cups was started
[bright52->category[default]->services[cups]]% status
cups            [   UP   ]
```

Using `cmgui`

Using `cmgui`, a service can be managed from a `Services` tab. The tab is accessible from a "`Head Nodes`" item, from a "`Node Categories`" item, or from a `Nodes` item.

Figure 4.16 shows the `Services` tab accessed from the `default` software image, which is an item within the "`Node Categories`" folder.

The "`init.d/<service name>`" command options `start`, `stop`, `reload`, and so on, are displayed as buttons in the `Services` tab.

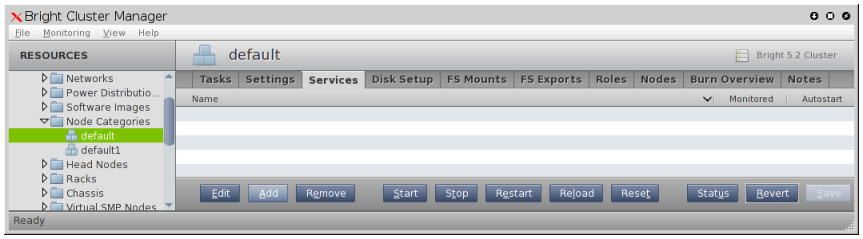

Figure 4.16: A `Services` Tab In `cmgui`

The existence of the service itself can be managed using the `Edit`, `Add`, and `Remove` buttons. The change can be saved or reverted with the `Save` and `Revert` buttons.

Figure 4.17 shows CUPS being set up from an `Add` dialog in the `Services` tab, accessed by clicking on the `Add` button in the `Services` tab of figure 4.16.

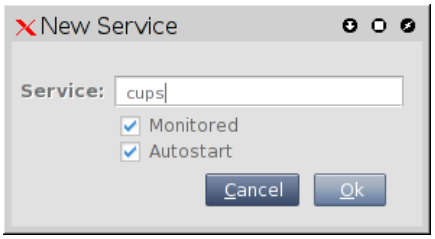

Figure 4.17: Setting Up A Service Using `cmgui`

For a service in the Services tab, clicking on the Status button in figure 4.16 displays a grid of the state of services on a running node as either Up or Down (figure 4.18). If the node is not running, it shows blank service entries.

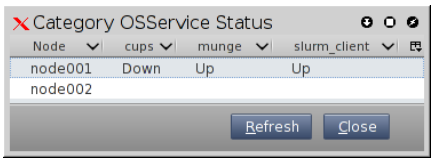

Figure 4.18: Displaying The Status Of Services Across Nodes Using cmgui

4.9 Managing And Configuring A Rack

4.9.1 Racks

A cluster may have local nodes grouped physically into racks. A rack is 42 units in height by default, and nodes normally take up one unit.

Racks Overview

Racks overview in cmgui: Selecting the Racks resource in cmgui opens up the Overview tabbed pane (figure 4.19). Racks can then be added, removed, or edited from that pane.

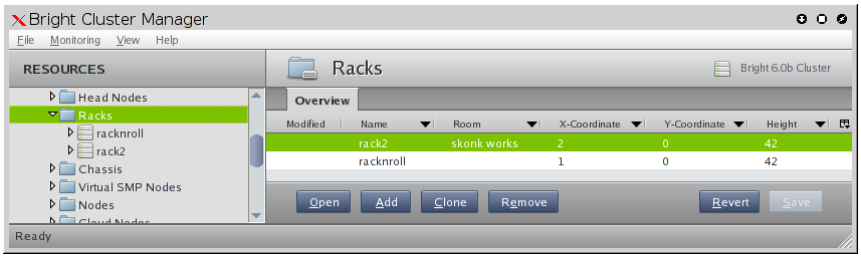

Figure 4.19: Racks Resource Overview Using cmgui

Within the Overview tabbed pane:

- a new rack item can be added with the Add button.
- an existing rack item can be edited by selecting it and clicking on the Open button, or by double clicking on the item itself in the tabbed pane.

An existing rack item can also be edited by simply selecting its name in the resource tree pane. This brings up its Settings tabbed pane (figure 4.20) by default.

Racks overview in cmsh: The rack mode in cmsh allows racks defined in the cluster manager to be listed:

```
[bright52->rack]% list
Name (key)      Room           x-Coordinate   y-Coordinate   Height
--------------  -------------  -------------  -------------  ------
racknroll                      1              0              42
rack2           skonk works    2              0              42
```

Rack Configuration Settings

Rack configuration settings in `cmgui`: The `Settings` tabbed pane for editing a rack item selected from the `Racks` resource is shown in figure 4.20.

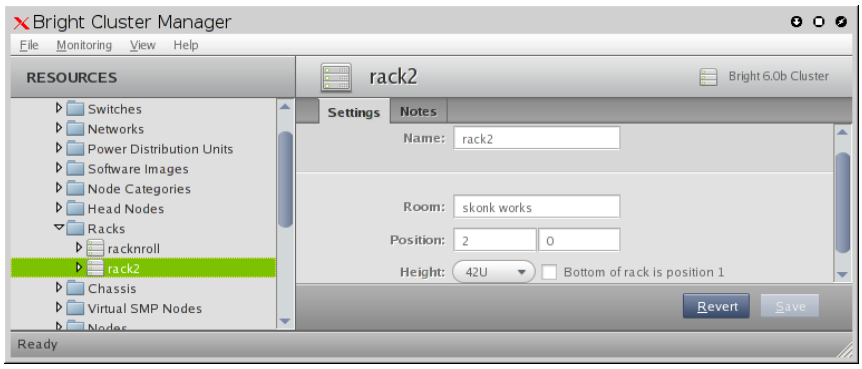

Figure 4.20: Rack Configuration Settings Using `cmgui`

The rack item configuration settings are:

- `Name`: A unique name for the rack item. Names such as `rack001`, `rack002` are a sensible choice
- `Room`: A unique name for the room the rack is in.
- `Position`: The x- and y-coordinates of the rack in a room. In the rack view visualization (section 4.9.2), the racks are displayed in one row, with nodes ordered so that the lowest x-value is the leftmost node for nodes with the same y-value. These coordinates are meant to be a hint for the administrator about the positioning of the racks in the room, and as such are optional, and can be arbitrary. The `Notes` tabbed pane can be used as a supplement or as an alternative for hints.
- `Height`: by default this is the standard rack size of 42U.
- `Bottom of rack is position 1`: Normally, a rack uses the number 1 to mark the top and 42 to mark the bottom position for the places that a device can be positioned in a rack. However, some manufacturers use 1 to mark the bottom instead. Ticking the checkbox displays the numbering layout accordingly for all racks in `Rackview` (section 4.9.2), if the checkboxed rack is the first rack seen in `Rackview`.

Rack configuration settings in `cmsh`: In `cmsh`, tab-completion suggestions for the `set` command in `rack` mode display the racks available for configuration. On selecting a particular rack (for example, `rack2` as in the following example), tab-completion suggestions then display the configuration settings available for that rack:

Example

```
[bright52->rack]% set rack
rack1  rack2  rack3
```

```
[bright52->rack]% set rack2
height        inverted      name          notes         room
x-coordinate  y-coordinate
```

The configuration settings for a particular rack obviously match with the parameters associated with and discussed in figure 4.20. The only slightly unobvious match is the boolean parameter `inverted` in `cmsh`, which simply corresponds directly to "`Bottom of rack is position 1`" in `cmgui`.

Setting the values can be done as in this example:

Example

```
[bright52->rack]% use rack2
[bright52->rack[rack2]]% set room "skonk works"
[bright52->rack*[rack2*]]% set x-coordinate 2
[bright52->rack*[rack2*]]% set y-coordinate 0
[bright52->rack*[rack2*]]% set inverted no
[bright52->rack*[rack2*]]% commit
[bright52->rack[rack2]]%
```

4.9.2 Rack View

The `Rackview` tab is available within `cmgui` after selecting the cluster resource item (figure 4.21).

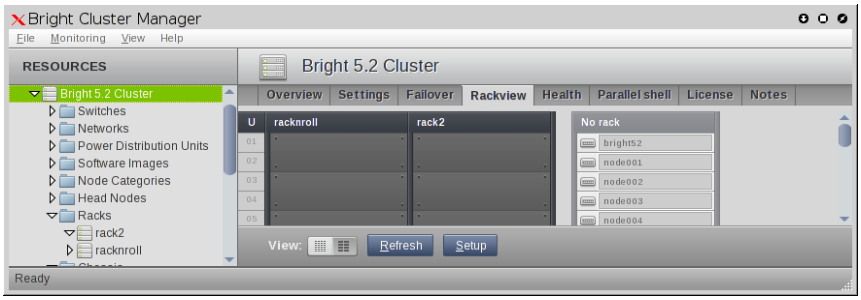

Figure 4.21: Rack View Using `cmgui`

Using Rack View

The `Rackview` pane of `cmgui` is a visualization of the layout of the racks, and also a visualization of the layout of devices such as nodes, switches, and chassis within a rack.

A device can be visually located in a rack clicking on the "`Locate in rack`" button in `Tasks` tabbed pane of that device. Clicking on the button opens up the `Rackview` pane with the device highlighted in red.

Some of the `Racks` configuration settings of figure 4.20 can also be visualized within rack view.

An additional visualization that can be set up from within the `Rackview` pane itself, is that a metric value can be displayed as a calibrated color. Set up details are given soon (page 97).

In the `Rackview` pane, the color associated with a metric value is displayed upon a box representing a device in the rack such as a switch or a node. This can be useful in getting an idea of the spread of values across a cluster at a glance. In some cases patterns related to the physical positioning of devices in the rack and of the physical positioning of the racks can become obvious.

For example, if ventilation is not working well in one part of the room, then metrics for the rack devices in that part may show a higher temperature compared to devices that are beter ventilated. Mapped out as colors in the `Rackview` pane, the discrepancy would be noticed at a glance.

Rack View Configuration

In rack view:

- The rack name is displayed at the top of the rack.
- Items such as nodes can be assigned to particular positions within racks (section 4.9.3).
- The remaining items that are not assigned to a rack are kept in a space labeled "'`No rack`".
- Tooltip text above a node assigned to a rack shows useful information about that particular node and any rack-view-associated cluster metrics from the "`Rack Setup`" window (figure 4.22).
- The `View` buttons at the bottom allow two kinds of rack views to be displayed:
  - "`Detailed view`" shows the node names assigned to the rack, along with colored boxes associated with each of up to 4 (the first 4 if there are more than 4) metrics assigned to the cluster. The color of the boxes correspond to the retrieved metric value.
  - "`Simple view`" does not show node names, it displays the color associated with the value of only the first metric (if it exists), and also has a more "zoomed out" visualization of the racks in the cluster.
- The `Refresh` button does a manual refresh of the rack view.
- The `Setup` button opens up the "`Rack Setup`" window (figure 4.22).

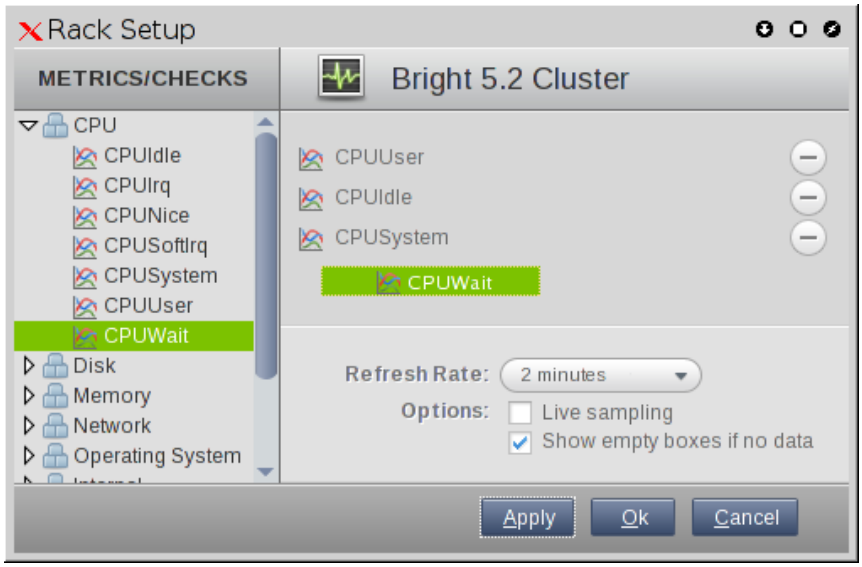

Figure 4.22: Rack Setup, Showing Drag-And-Drop Of `CPUWait` Metric

The "Rack Setup" window allows the racks to be configured with metrics. Rack view then displays the metrics in "Detailed View" and "Simple View" according to an automatically calibrated color scale.

In the "Rack Setup" window

- The metrics can be configured by drag-and-drop.
- The Apply button applies a change.
- The Ok button applies any changes, and closes the "Rack Setup" screen.
- The "Refresh Rate" selection sets the time interval between metric measurement retrieval.
- A checkmark in the "Live sampling" checkbox instructs that the selected metrics be measured and displayed at the refresh interval. Scripts that take longer than about a second to run can be used, but are best avoided because the display is not very "live" in that case.

4.9.3 Assigning Devices To A Rack

Devices such as nodes, switches, and chassis, can be assigned to racks.

By default, no devices such as nodes, switches, and chassis are assigned to a rack. All devices are thus originally shown as part of the "No rack" grouping in the Rackview tabbed pane.

Devices can be assigned to a particular rack and to a particular position within the rack as follows:

Assigning Devices To A Rack Using cmgui

Using cmgui, the assignments of a device such as a node to a rack can be done from the Settings tabbed pane of the device, within the Rack section (figure 4.23):

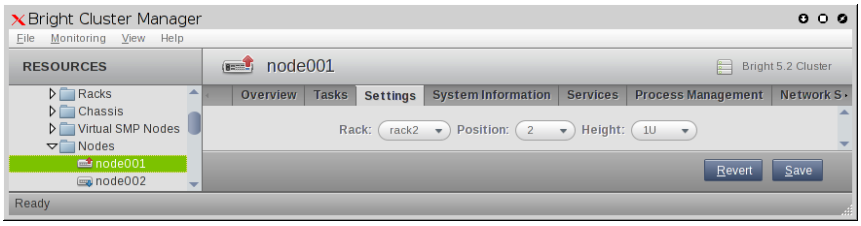

Figure 4.23: Rack Assignment Using cmgui

Assigning Devices To A Rack Using cmsh

Using cmsh, the assignment can be done to a rack as follows:

```
[bright52->device]% foreach -n node001..node003 (set deviceheight 1; se\
t deviceposition 2; set rack rack2)
[bright52->device*]% commit
Successfully committed 3 Devices
[bright52->device]%
```

The Convention Of The Top Of The Device Being Its Position

Since rack manufacturers usually number their racks from top to bottom, the position of a device in a rack (using the parameter Position in cmgui,

and the parameter `deviceposition` in `cmsh`) is always taken to be where the top of the device is located. This is the convention followed even for the less usual case where the rack numbering is from bottom to top.

Most devices are 1U in height, so that the top of the device is at the same position as the bottom of the device, and no confusion is possible. The administrator should however be aware that for any device that is greater than 1U in height, for example, a blade enclosure chassis 4.9.4, the convention means that it is the position of the top of the device that is noted as being the position of the device rather than the bottom.

4.9.4 Assigning Devices To A Chassis

A Chassis As A Physical Part Of A Cluster

In a cluster, several local nodes may be grouped together physically into a chassis. This is common for clusters using blade systems. Clusters made up of blade systems use less space, less hardware, and less electrical power than non-blade clusters with the same computating power. In blade systems, the blades are the nodes, and the chassis is the blade enclosure.

A blade enclosure chassis is typically 6 to 10U in size, and the node density is typically 2 blades per unit with current (2012) technology.

Chassis Configuration And Node Assignment

Chassis configuration and node assignment with `cmgui`: General chassis configuration in `cmgui` is done from the `Chassis` resource. Within the `Chassis` resource is the chassis `Members` tabbed pane, which allows assignment of nodes to a chassis, and which is described in detail soon (page 101).

If the `Chassis` resource is selected from the resource tree, then a new chassis item can be added, or an existing chassis item can be opened.

The chassis item can then be configured from the available tabs, some of which are:

- The `Tasks` tab (figure 4.24).

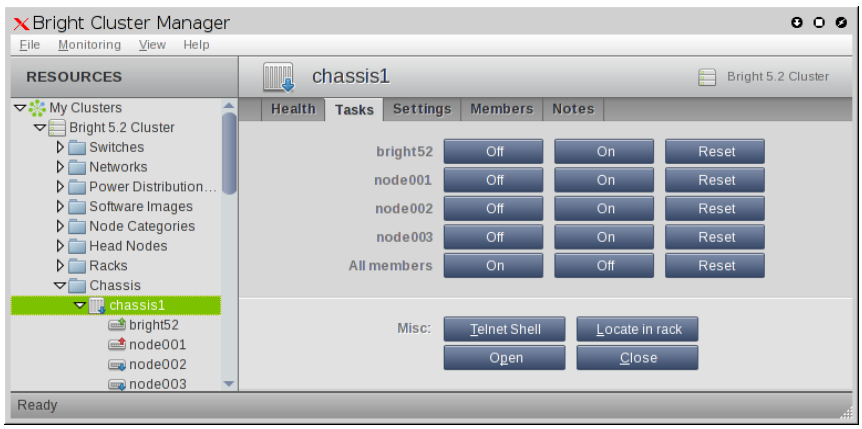

Figure 4.24: Chassis Tasks Tab

The `Tasks` tab for a chassis item allows:

- Individual member nodes of the chassis to be powered on and off, or reset.

– All the member nodes of the chassis to be powered on and off, or reset.

Miscellaneous tasks in the `Tasks` tab are:

– The ability to start a telnet session, if possible, with the chassis operating system.
– The ability to locate the chassis in the visual representation of rack view.
– The ability to open or close the chassis state. States are described in section 6.5.

- The `Settings` tab (figure 4.25).

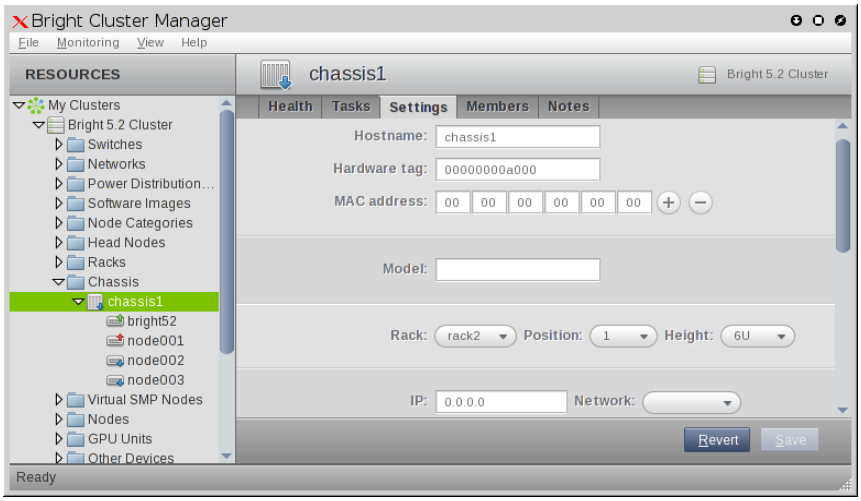

Figure 4.25: Chassis Settings Tab

Amongst other options, the `Settings` tab for a chassis item allows selection of:

– A rack from the set of defined racks.
– A position within the rack (typically between 1 and 42 units).
– The height of the chassis (typically between 6 and 10 units). Because the typical numbering of a rack is from position 1 at the top to position 42 at the bottom, a chassis of height 6 at position 1 will take up the space from postion 1 to 6. Effectively the base of the chassis is then at position 6, and the space from 1 to 6 can only be used by hardware that is to go inside the chassis.

The settings become active when the `Save` button is clicked.

- The `Members` tab (figure 4.26).

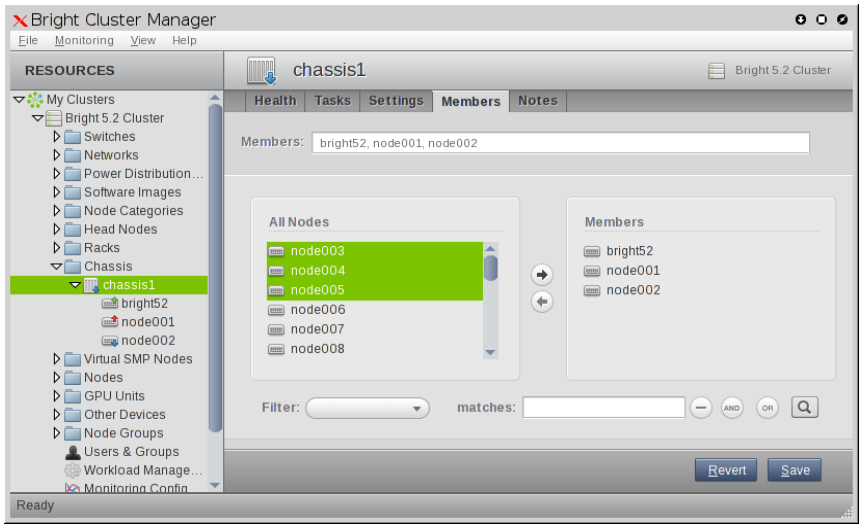

Figure 4.26: Chassis Members Tab

The `Members` tab for a chassis item allows the following actions:

- Nodes can be selected, then added to or removed from the chassis.
- Multiple nodes can be selected at a time for addition or removal.
- The selection can be done with standard mouse and keyboard operations, or using the filter tool.
- The filter tool allows selection based on some useful node parameters such as the hostname, the MAC address, the category. Several rules can made to operate together.

Chassis configuration and node assignment with `cmsh`: The `chassis` mode in `cmsh` allows configuration related to a particular chassis. Tab-completion suggestions for a selected chassis with the `set` command show possible parameters that may be set:

Example

```
[bright52->device[chassis1]]% set
custompingscript            ip              powercontrol
custompingscriptargument    mac             powerdistributionunits
custompowerscript           members         rack
custompowerscriptargument   model           slots
deviceheight                network         tag
deviceposition              notes           userdefined1
ethernetswitch              partition       userdefined2
hostname                    password        username
```

Whether the suggested parameters are actually supported depends on the chassis hardware. For example, if the chassis has no network interface of its own, then the `ip` and `mac` address settings may be set, but cannot function.

The "positioning" parameters of the chassis within the rack can be set as follows with `cmsh`:

Example

```
[bright52->device[chassis1]]% set rack rack2
[bright52->device*[chassis1*]]% set deviceposition 1; set deviceheight 6
[bright52->device*[chassis1*]]% commit
```

The members of the chassis can be set as follows with `cmsh`:

Example

```
[bright52->device[chassis1]]% set members bright52 node001..node005
[bright52->device*[chassis1*]]% commit
```

An additional feature that can be configured in `cmsh` is that any slots in the chassis can be assigned an arbitrary value. A chassis may physically have slots labeled with one number per node, in which case the default slot assignment of setting `1` to the first slot, `2` to the second node, and so on, is probably a good start. If the numbering is not right, or if there is no numbering, then the administrator can set their own arbitrary value. For example:

Example

```
[bright52->device[chassis1]]% set slots bright60 "leftmost top"
[bright52->device*[chassis1*]]% set slots node001 "leftmost bottom"
[bright52->device*[chassis1*]]% set slots node002 "red cable tie top"
[bright52->device*[chassis1*]]% set slots node003 "red cable tie bottom"
```

The values set are saved with the `commit` command.

4.9.5 An Example Of Assigning A Device To A Rack, And Of Assigning A Device To A Chassis

The following illustrative case explains how to set up a blade server in a cluster. This breaks down naturally into 2 steps: firstly configuring the chassis (the blade server device) for the rack, and secondly configuring the nodes (the blade devices) for the chassis.

1. **Assigning a chassis to a position within a rack:** A position within a rack of the cluster is first chosen for the chassis.

   To assign a chassis to a rack in the cluster manager, the rack must first exist according to the cluster manager. If the rack is not yet known to the cluster manager, it can be configured as described in section 4.9.

   Assuming racks called `rack1`, `rack2`, and `rack3` exist, it is decided to place the chassis in `rack2`.

   If the blade server is 10 units in height, with 20 nodes, then 10 units of room from the rack must be available. If positions 5 to 14 are free within `rack2`, then suitable positional settings are:

   - `rack2` for the rack value,
   - `5` for the position value,
   - `10` for the height value.

   These values can then be set as in the example of figure 4.25, and come into effect after saving them.

2. **Assigning nodes to the chassis:** The nodes of the chassis can then be made members of the chassis, as is done in the example of figure 4.26, and saving the members settings.

   The nodes of the chassis must be assigned rack, position within the rack, and height values that are the same values as those of the chassis. These values must be explicitly assigned per node, as is done in the example of figure 4.23.

   Writing out these values for 20 nodes is quicker with some `cmsh` scripting within `device` mode:

```
[bright52->device]% foreach -n node001..node020 (set rack rack2; s\
et deviceposition 5; set deviceheight 10)
[bright52->device*]% commit
```

   After the nodes rack, position within the rack, and height values match those held by the chassis, they display correctly within the chassis space within rack view.

5

Power Management

Being able to control power inside a cluster through software is important for remote cluster administration and creates opportunities for power savings. It allows cluster burn tests to be carried out (appendix N), and can also be useful to be able to measure power usage over time. This chapter describes the Bright Cluster Manager power management features.

In section 5.1 the configuration of the methods used for power operations is described.

Section 5.2 then describes the way the power operations commands themselves are used to allow the administrator turn power on or off, reset the power, and retrieve the power status. It explains how these operations can be applied to devices in various ways.

Section 5.3 briefly covers monitoring power.

The integration of power saving with workload management systems is covered in the chapter on Workload Management (section 8.9).

5.1 Configuring Power Parameters

Several methods exist to control power to devices:

- Power Distribution Unit (PDU) based power control
- IPMI-based power control (for node devices only)
- Custom power control
- HP iLO-based power control (for node devices only)

5.1.1 PDU-Based Power Control

For PDU-based power control, the power supply of a device is plugged into a port on a PDU. The device can be a node, but also anything else with a power supply, such as a switch. The device can then be turned on or off by changing the state of the PDU port.

To use PDU-based power control, the PDU itself must be a device in the cluster and be reachable over the network. The `Settings` tab of each device object plugged into the PDU is then used to configure the PDU ports that control the device. Figure 5.1 shows the `Settings` tab for a head node.

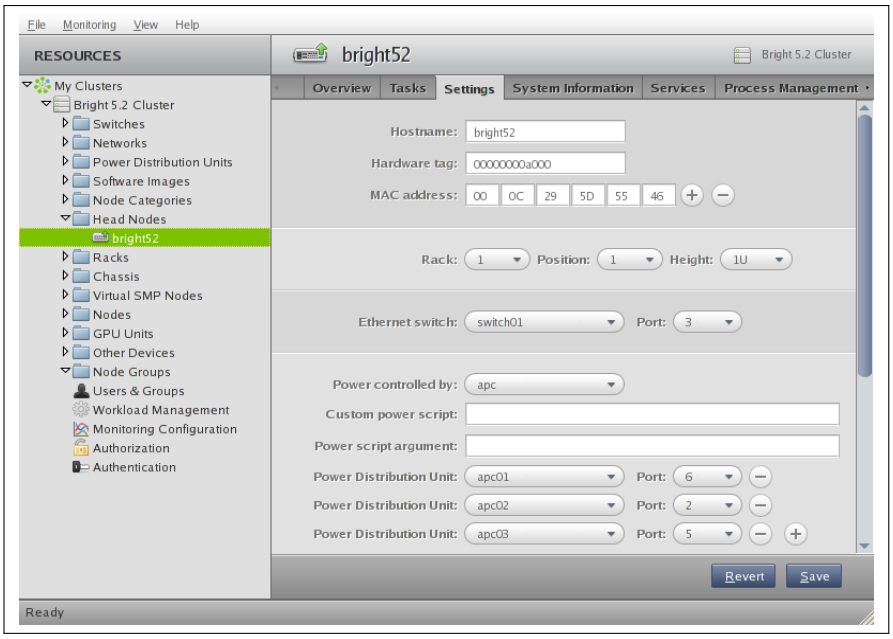

Figure 5.1: Head Node Settings

Each device plugged into the PDU can have PDU ports added and removed with the ⊕ and ⊖ buttons in their `Settings` tab. For the APC brand of PDUs, the "`Power controlled by`" property in the `Settings` tab should be set to `apc`, or the list of PDU ports is ignored by default. Overriding the default is described in section 5.1.3.

Since nodes may have multiple power feeds, there may be multiple PDU ports defined for a single device. The cluster management infrastructure takes care of operating all ports of a device in the correct order when a power operation is done on the device.

It is also possible for multiple devices to share the same PDU port. This is the case for example when *twin nodes* are used (i.e. two nodes sharing a single power supply). In this case, all power operations on one device apply to all nodes sharing the same PDU port.

If the PDUs defined for a node are not manageable, then the node's baseboard management controllers (that is, IPMI/iLO and similar) are assumed to be inoperative and are therefore assigned an unknown state. This means that dumb PDUs, which cannot be managed remotely, are best not assigned to nodes in Bright Cluster Manager. Administrators wishing to use Bright Cluster Manager to record that a dumb PDU is assigned to a node can deal with it as follows:

- in `cmgui` the `Notes` tab or the "`User defined 1`"/"`User defined 2`" options in the `Settings` tab for that node can be used.
- in `cmsh` the equivalent is accessible when using the node from device mode, and running "`set notes`", "`set userdefined1`", or "`set userdefined2`".

For PDUs that are manageable:

- In `cmgui`, the `Overview` tab of a PDU (figure 5.2) provides an overview of the state of PDU ports and devices that have been associated with each port.

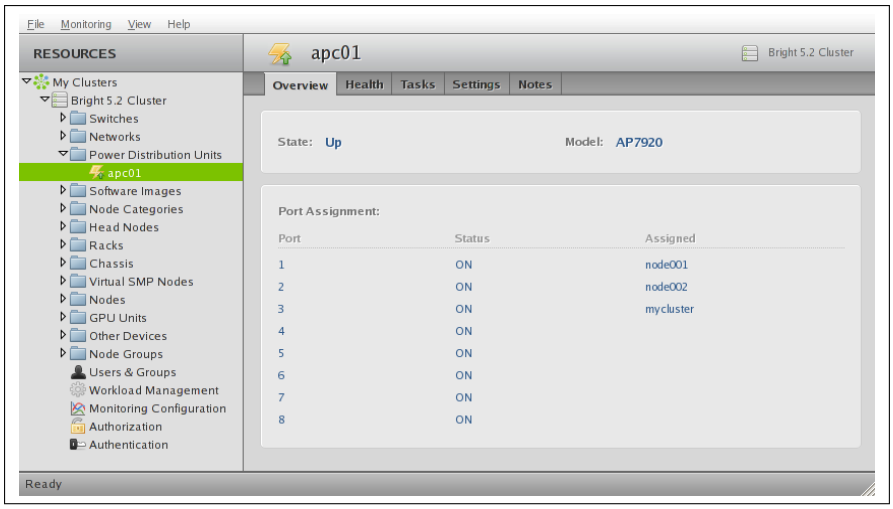

Figure 5.2: PDU Overview

The power status of a node can be seen by selecting the node from the `Nodes` resource, then selecting the `Overview` tabbed pane. The first tile of the `Overview` tab displays the power state and assignment of any PDUs for the node.

- In `cmsh`, power-related options can be accessed from `device` mode, after selecting a device:

Example

```
[bright52]% device use node001
[bright52->device[node001]]% show | grep -i power
Custom power script argument
Ipmi/iLO power reset delay             0
Power control                          apc
PowerDistributionUnits                 apc01:6 apc01:7
```

The power status of a node can be accessed with:

Example

```
[bright52->device[node001]]% power status
```

If the node is up and has one or more PDUs assigned to it, then the power status is one of `ON`, `OFF`, `RESET`, `FAILED`, or `UNKNOWN`:

Power Status	Description
`ON`	Power is on
`OFF`	Power is off
`RESET`	Shows during the short time the power is off during a power reset. The reset is a hard power off for PDUs, but can be a soft or hard reset for other power control devices.
`FAILED`	Power status script communication failure.
`UNKNOWN`	Power status script timeout

5.1.2 IPMI-Based Power Control

IPMI-based power control relies on the baseboard management controller (BMC) inside a node. It is therefore only available for node devices. Blades inside a blade chassis typically use IPMI for power management. For details on setting up networking and authentication for IPMI interfaces, see section 4.4.

To carry out IPMI-based power control operations, the "`Power controlled by`" property in figure 5.1 must be set to the IPMI interface through which power operations should be relayed. Normally this IPMI interface is `ipmi0`. Any list of configured APC PDU ports displayed in the GUI is ignored by default when the "`Power controlled by`" property is not `apc`.

Example

Configuring power parameters settings for a node using `cmsh`:

```
[mycluster]% device use node001
[mycluster->device[node001]]% set powerdistributionunits apc01:6 apc01:7 apc01:8
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:6 apc01:7 apc01:8
[mycluster->device*[node001*]]% removefrom powerdistributionunits apc01:7
[mycluster->device*[node001*]]% get powerdistributionunits
apc01:6 apc01:8
[mycluster->device*[node001*]]% set powercontrol apc
[mycluster->device*[node001*]]% get powercontrol
apc
[mycluster->device*[node001*]]% commit
```

5.1.3 Combining PDU- and IPMI-Based Power Control

By default when nodes are configured for IPMI Based Power Control, any configured PDU ports are ignored. However, it is sometimes useful to change this behavior.

For example, in the CMDaemon configuration file directives in `/cm/local/apps/cmd/etc/cmd.conf` (introduced in section 3.6.2 and listed in Appendix C), the default value of `PowerOffPDUOutlet` is `false`. It can be set to `true` on the head node, and CMDaemon restarted to activate it.

With `PowerOffPDUOutlet` set to `true` it means that CMDaemon, after receiving an IPMI-based power off instruction for a node, and after powering off that node, also subsequently powers off the PDU port. Powering off the PDU port shuts down the BMC, which saves some additional power—typically a few watts per node. When multiple nodes share the same PDU port, the PDU port only powers off when all nodes served by that particular PDU port are powered off.

When a node has to be started up again the power is restored to the node. It is important that the node BIOS is configured to automatically power on the node when power is restored.

5.1.4 Custom Power Control

For a device which cannot be controlled through any of the standard existing power control options, it is possible to set a custom power management script. This is then invoked by the cluster management daemon on the head node whenever a power operation for the device is done.

Power operations are described further in section 5.2.

Using `custompowerscript`

To set a custom power management script for a device, the `powercontrol` attribute is set to `custom` using either `cmgui` or `cmsh`, and the value of `custompowerscript` is specified. The value for `custompowerscript` is the full path to an executable custom power management script on the head node(s) of a cluster.

A custom power script is invoked with the following mandatory arguments:

```
myscript <operation> <device>
```

where `<device>` is the name of the device on which the power operation is done, and `<operation>` is one of the following:

```
ON
OFF
RESET
STATUS
```

On success a custom power script exits with exit code 0. On failure, the script exits with a non-zero exit-code.

Using `custompowerscriptargument`

The mandatory argument values for `<operation>` and `<device>` are passed to a custom script for processing. For example, in `bash` the positional variables `$1` and `$2` are typically used for a custom power script. A custom power script can also be passed a further argument value by setting the value of `custompowerscriptargument` for the node via `cmsh` or `cmgui`. This further argument value would then be passed to the positional variable `$3` in `bash`.

An example custom power script is located at `/cm/local/examples/cmd/custompower`. In it, setting `$3` to a positive integer delays the script via a `sleep` command by `$3` seconds.

An example that is conceivably more useful than a "`sleep $3`" command is to have a "`wakeonlan $3`" command instead. If the `custompowerscriptargument` value is set to the MAC address of the node, that means the MAC value is passed on to `$3`. Using this technique, the power operation ON can then carry out a Wake On LAN operation on the node from the head node.

Setting the `custompowerscriptargument` can be done like this for all nodes:

```
#!/bin/bash
for nodename in $(cmsh -c "device; foreach * (get hostname)")
do
  macad=‘cmsh -c "device use $nodename; get mac"‘
  cmsh -c "device use $nodename; set customscriptargument $macad; commit"
done
```

The preceding material usefully illustrates how `custompowerscriptargument` can be used to pass on arbitrary parameters for execution to a custom script.

However, the goal of the task can be achieved in a simpler and quicker way using the environment variables available in the cluster management daemon environment. This is explained next.

Using Environment Variables With `custompowerscript`

Simplification of the steps needed for custom scripts in CMDaemon is often possible because there are values in the CMDaemon environment already available to the script. A line such as:

```
env > /tmp/env
```

added to the start of a custom script dumps the names and values of the environment variables to `/tmp/env` for viewing.

One of the names is `$CMD_MAC`, and it holds the MAC address string of the node being considered.

So, it is not necessary to retrieve a MAC value for `custompowerscriptargument` with a bash script as shown in the previous section, and then pass the argument via `$3` such as done in the command "`wakeonlan $3`". Instead, `custompowerscript` can simply call "`wakeonlan $CMD_MAC`" directly in the script when run as a power operation command from within CMDaemon.

5.1.5 Hewlett Packard iLO-Based Power Control

If "`Hewlett Packard`" is chosen as the node manufacturer during installation, and the nodes have an iLO management interface, then Hewlett-Packard's iLO management package, `hponcfg`, is installed by default on the nodes and head nodes.

The `hponcfg` rpm package is normally obtained and upgraded for specific HP hardware from the HP website. Using an example of `hponcfg-3.1.1-0.noarch.rpm` as the package downloaded from the HP website, and to be installed, the installation can then be done on the head node, the software image, and in the node-installer as follows:

```
rpm -iv hponcfg-3.1.1-0.noarch.rpm
rpm --root /cm/images/default-image -iv hponcfg-3.1.1-0.noarch.rpm
rpm --root /cm/node-installer -iv hponcfg-3.1.1-0.noarch.rpm
```

To use iLO over all nodes, the following steps are done:

1. The iLO interfaces of all nodes are set up like the IPMI interfaces outlined in section 5.1.2. Bright Cluster Manager treats HP iLO interfaces just like regular IPMI interfaces.
2. The `ilo_power.pl` custom power script must be configured on all nodes. This can be done with a `cmsh` script. For example, for all nodes in the `default` category:

   Example

   ```
   [mycluster]% device foreach -c default (set custompowerscript /cm/loc\
   al/apps/cmd/scripts/powerscripts/ilo_power.pl)
   [mycluster]% device foreach -c default (set powercontrol custom)
   [mycluster]% device commit
   ```

5.2 Power Operations

Power operations may be done on devices from either `cmgui` or `cmsh`. There are four main power operations:

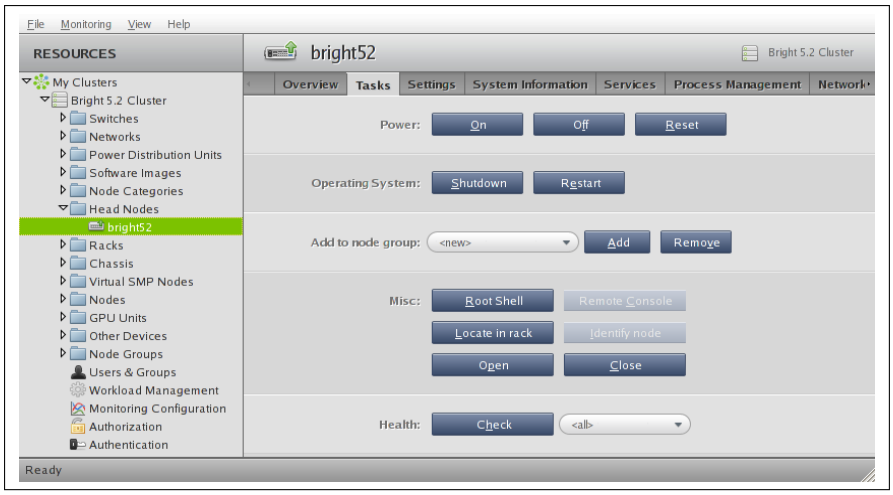

Figure 5.3: Head Node Tasks

- Power On: power on a device
- Power Off: power off a device
- Power Reset: power off a device and power it on again after a brief delay
- Power Status: check power status of a device

5.2.1 Power Operations With `cmgui`

In `cmgui`, buttons for executing On/Off/Reset operations are located under the `Tasks` tab of a device. Figure 5.3 shows the `Tasks` tab for a head node.

The `Overview` tab of a device can be used to check its power status information. In the display in figure 5.4, for a head node, the green LEDs indicate that all three PDU ports are turned on. Red LEDs would indicate power ports that have been turned off, while gray LEDs would indicate an unknown power status for the device.

Performing power operations on multiple devices at once is possible through the `Tasks` tabs of node categories and node groups.

It is also possible to do power operations on ad hoc groups through the `Nodes` folder in the resource tree: The members of the ad hoc group can be selected using the `Overview` tab, and then operated on by a task chosen from the `Tasks` tab.

When doing a power operation on multiple devices, CMDaemon ensures a 1 second delay occurs by default between successive devices, to avoid power surges on the infrastructure. The delay period may be altered using `cmsh`'s "`-d|--delay`" flag.

The `Overview` tab of a PDU object (figure 5.5), allows power operations on PDU ports by the administrator directly. All ports on a particular PDU can have their power state changed, or a specific PDU port can have its state changed.

5.2.2 Power Operations Through `cmsh`

All power operations in `cmsh` are done using the `power` command in `device` mode. Some examples of usage are now given:

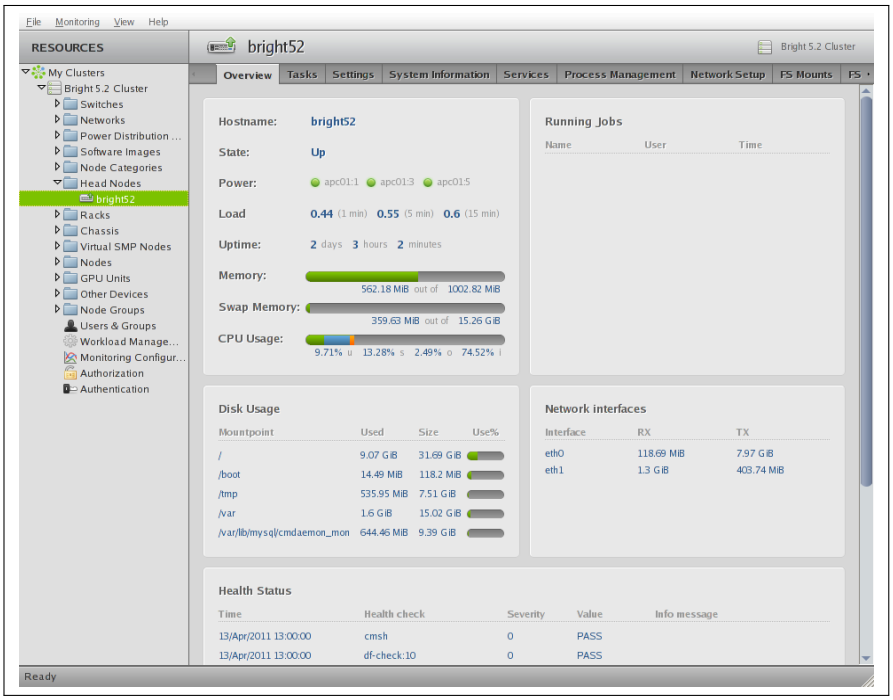

Figure 5.4: Head Node Overview

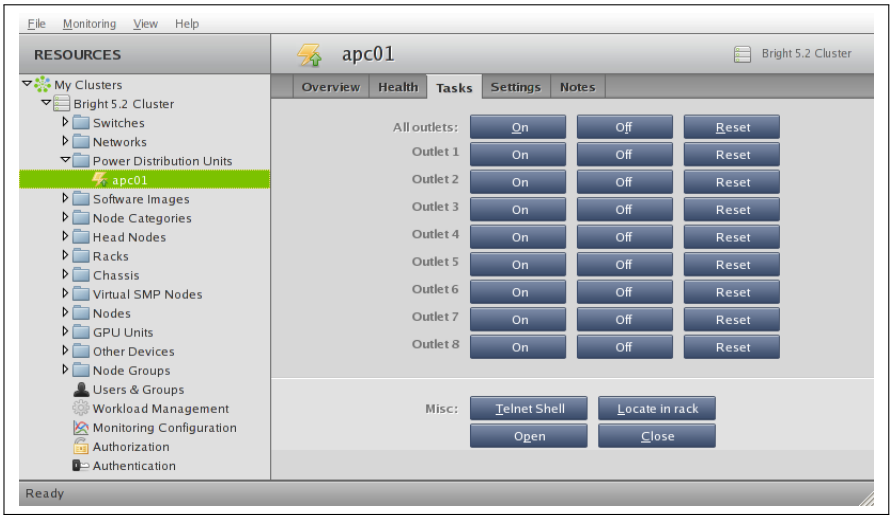

Figure 5.5: PDU Tasks

- Powering on node001, and nodes from node018 to node033 (output truncated):

  Example

```
[mycluster]% device power -n node001,node018..node033 on
apc01:1 ............. [   ON    ] node001
apc02:8 ............. [   ON    ] node018
apc02:9 ............. [   ON    ] node019
...
```

- Powering off all nodes in the default category with a 100ms delay between nodes (some output elided):

  Example

```
[mycluster]% device power -c default -d 0.1 off
apc01:1 ............. [   OFF   ] node001
apc01:2 ............. [   OFF   ] node002
...
apc23:8 ............. [   OFF   ] node953
```

- Retrieving power status information for a group of nodes:

  Example

```
[mycluster]% device power -g mygroup status
apc01:3 ............. [   ON    ] node003
apc01:4 ............. [   OFF   ] node004
```

Figure 5.6 shows usage information for the power command.

5.3 Monitoring Power

Monitoring power consumption is important since electrical power is an important component of the total cost of ownership for a cluster. The monitoring system of Bright Cluster Manager collects power-related data from PDUs in the following metrics:

- PDUBankLoad: Phase load (in amperes) for one (specified) bank in a PDU
- PDULoad: Total phase load (in amperes) for one PDU

Chapter 10 on cluster monitoring has more on metrics and how they can be visualized.

```
Name:
   power - Manipulate or retrieve power state of devices

Usage:

   power [OPTIONS] status
   power [OPTIONS] on
   power [OPTIONS] off
   power [OPTIONS] reset

Options:
   -n, --nodes node(list)
     List of nodes, e.g. node001..node015,node20..node028,node030 or ^/some/file/containing/hostnames

   -g, --group group(list)
     Include all nodes that belong to the node group, e.g. testnodes or test01,test03

   -c, --category category(list)
     Include all nodes that belong to the category, e.g. default or default,gpu

   -r, --rack rack(list)
     Include all nodes that are located in the given rack, e.g rack01 or rack01..rack04

   -h, --chassis chassis(list)
     Include all nodes that are located in the given chassis, e.g chassis01 or chassis03..chassis05

   -p, --powerdistributionunitport <pdu:port>(list)
     perform power operation directly on power distribution units. Use port '*' for all ports

   -b, --background
     Run in background, output will come as events

   -d, --delay <seconds>
     Wait <seconds> between executing two sequential power commands. This option is ignored for the status
     command

   -s, --state <states>
     Only run power command on nodes in specified states, e.g. UP, "CLOSED|DOWN", "INST.*"

   -f, --force
     Force power command on devices which have been closed

Examples:
   power status          Display power status for all devices or current device
   power on -n node001   Power on node001
```

Figure 5.6: Help Text For `power` Command In `device` Mode Of `cmsh`

6

Node Provisioning

This chapter covers *node provisioning*. Node provisioning is the process of how nodes obtain an image. Typically, this happens during their stages of progress from power-up to becoming active in a cluster, but node provisioning can also take place when updating a running node.

Section 6.1 describes the stages leading up to the loading of the kernel onto the node.

Section 6.2 covers configuration and behavior of the provisioning nodes that supply the software images.

Section 6.3 describes the configuration and loading of the kernel, the ramdisk, and kernel modules.

Section 6.4 elaborates on how the node-installer identifies and places the software image on the node in a 13-step process.

Section 6.5 explains node states during normal boot, as well node states that indicate boot problems.

Section 6.6 describes how running nodes can be updated, and modifications that can be done to the update process.

Section 6.7 explains how to add new nodes to a cluster so that node provisioning will work for these new nodes too. The `cmsh` and `cmsh` front ends for creating new node objects and properties in CMDaemon are described.

Section 6.8 describes troubleshooting the node provisioning process.

6.1 Before The Kernel Loads

Immediately after powering up a node, and before it is able to load up the Linux kernel, a node starts its boot process in several possible ways:

6.1.1 PXE Booting

By default, nodes boot from the network when using Bright Cluster Manager. This is called a *network boot*, or sometimes a *PXE boot*. It is recommended as a BIOS setting for nodes.

6.1.2 gPXE Booting From A Disk Drive

Also by default, on disked nodes, gPXE software is placed on the drive during node installation. If the boot instructions from the BIOS for PXE booting fail, and if the BIOS instructions are that a boot attempt should then be made from the hard drive, it means that a PXE network boot

attempt is done again, as instructed by the bootable hard drive. This can be a useful fallback option that works around certain BIOS features or problems.

6.1.3 gPXE Booting Using InfiniBand

On clusters that have InfiniBand hardware, it is normally used for data transfer as a service after the nodes have fully booted up (section 4.3). InfiniBand can also be used for PXE booting (described here) and used for node provisioning (section 6.3.3). However these uses are not necessary, even if InfiniBand is used for data transfer as a service later on, because booting and provisioning is available over Ethernet by default. This section (about boot over InfiniBand) may therefore safely be skipped when first configuring a cluster.

Booting over InfiniBand via PXE is enabled by carrying out these 3 steps:

1. Making the Bright Cluster Manager aware that nodes are to be booted over InfiniBand. This can be done during the initial installation on the head node by marking the option "`Allow booting over InfiniBand`" (figure 2.10). Alternatively, if the cluster is already installed, then node booting (section 4.2.3, page 71) can be set from `cmsh` or `cmgui` as follows:

   (a) From `cmsh`'s `network` mode: If the InfiniBand network name is `ibnet`, then a `cmsh` command that will set it is:
   ```
   cmsh -c "network; set ibnet nodebooting yes; commit"
   ```

   (b) From `cmgui`'s "`Settings`" tab from the "`Networks`" resource (figure 4.5): The network item selected must be the InfiniBand network, "`ibnet`" by default, and the "`Allow node booting`" option is then set and saved.

   If the InfiniBand network does not yet exist, then it must be created (section 4.2.2). The recommended default values used are described in section 4.3.3.

   The administrator should also be aware that the interface from which a node boots, (conveniently labeled `BOOTIF`), must not be an interface that is already configured for that node in CMDaemon. For example, if `BOOTIF` is the device `ib0`, then `ib0` must not already be configured in CMDaemon . Either `BOOTIF` or the `ib0` configuration should be changed so that node installation can succeed.

2. Flashing gPXE onto the InfiniBand HCAs. (The ROM image is obtained from the HCA vendor).

3. Configuring the BIOS of the nodes to boot from the InfiniBand HCA.

Administrators who enable gPXE booting almost always wish to provision over InfiniBand too. Configuring provisioning over InfiniBand is described in section 6.3.3.

6.1.4 Booting From The Drive

Besides network boot, a node can also be configured to start booting and get to the stage of loading up its kernel entirely from its drive (section (6.4.4), just like a normal standalone machine.

6.2 Provisioning Nodes

The action of node provisioning is done by special nodes that do the provisioning, called the *provisioning nodes*. In a default Bright installation the head node is already set up as a provisioning node, and this is recommended for simple clusters. More complex clusters can have several *provisioning nodes*, thereby distributing network traffic loads when many nodes are booting.

Creating provisioning nodes is done by assigning the *provisioning role* to a node or category of nodes.

6.2.1 Provisioning Nodes: Configuration Settings

The provisioning role has several parameters that can be set:

Property	Description
`allImages`	When set to "`yes`", the provisioning node provides all available images regardless of any other parameters set. By default it is set to "`no`".
`images`	A list of software images provided by the provisioning node. These are used only if `allImages` is "`no`".
`maxProvisioningNodes`	The maximum number of nodes that can be provisioned in parallel by the provisioning node. The optimum number depends on the infrastructure. The default value is 10, which is safe for typical cluster setups. Setting it lower may sometimes be needed to prevent network and disk overload.
`nodegroups`	A list of node groups. If set, the provisioning node only provisions members of the listed groups. By default, this value is unset and the provisioning node supplies any node. Typically, this is used to set up a convenient hierarchy of provisioning, for example based on grouping by rack and by groups of racks.

A provisioning node keeps a copy of all the images it provisions on its local drive, in the same directory as where the head node keeps such images. The local drive of a provisioning node must therefore have enough space available for these images, which may require changes in its disk layout.

6.2.2 Provisioning Nodes: Role Setup With cmsh

In the following cmsh example the administrator creates a new category called misc. The default category default already exists in a newly installed cluster.

The administrator then assigns the role called provisioning from the list of assignable roles to nodes in the misc category.

As an aside from the topic of provisioning, from an organizational perspective, other assignable roles include monitoring, storage, and failover. Tab-completion prompting after the assign command has been typed in lists all the possible roles.

The nodes in the misc category assigned the provisioning role then have default-image set as the image that they provision to other nodes, and have 20 set as the maximum number of other nodes to be provisioned simultaneously (some text is elided in the following example):

Example

```
[bright52]% category add misc
[bright52->category*[misc*]]% roles
[bright52->category*[misc*]->roles]% assign provisioning
[brig...*]->roles*[provisioning*]]% set allimages false
[brig...*]->roles*[provisioning*]]% set images default-image
[brig...*]->roles*[provisioning*]]% set maxprovisioningnodes 20
[brig...*]->roles*[provisioning*]]% show
Parameter                Value
------------------------ ------------------------------------------------
Name                     provisioning
Type                     ProvisioningRole
allImages                no
images                   default-image
maxProvisioningNodes     20
nodegroups
[bright52->category*[misc*]->roles*[provisioning*]]% commit
[bright52->category[misc]->roles[provisioning]]%
```

Assigning a provisioning role can also be done for an individual node instead, if using a category is deemed overkill:

Example

```
[bright52]% device use node001
[bright52->device[node001]]% roles
[bright52->device[node001]->roles]% assign provisioning
[bright52->device*[node001*]->roles*[provisioning*]]%
...
```

After carrying out a role change, the updateprovisioners command described in section 6.2.4 should be run manually so that the images are propagated to the provisioners and so that CMDaemon is able to stay up-to-date on which nodes do provisioning. Running it manually makes sense in order to avoid rerunning the command several times as typically several role changes are made for several nodes when configuring the provisioning of a cluster. The command in any case runs automatically after some time (section 6.2.4).

6.2.3 Provisioning Nodes: Role Setup With cmgui

The provisioning configuration outlined in cmsh mode in section 6.2.2 can be done via cmgui too, as follows:

The provisioning category is added by clicking on the Add button in the Overview tabbed pane in the Node Categories resource (figure 6.1).

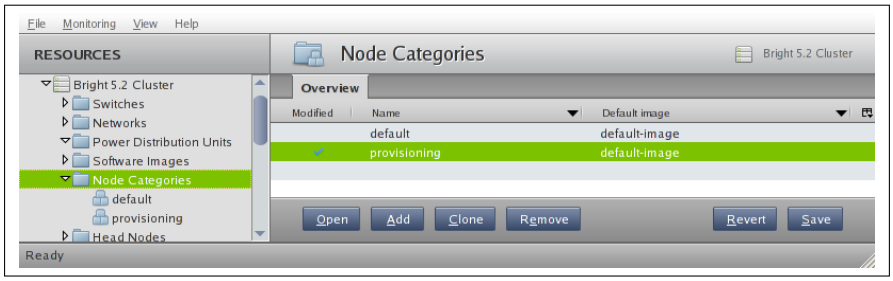

Figure 6.1: cmgui: Adding A provisioning Category

Clicking on the provisioning category in the resource tree on the left hand side (or alternatively, double-clicking on provisioning category in the Overview tabbed pane of the Node Categories right hand side pane) then opens up the provisioning category (figure 6.2).

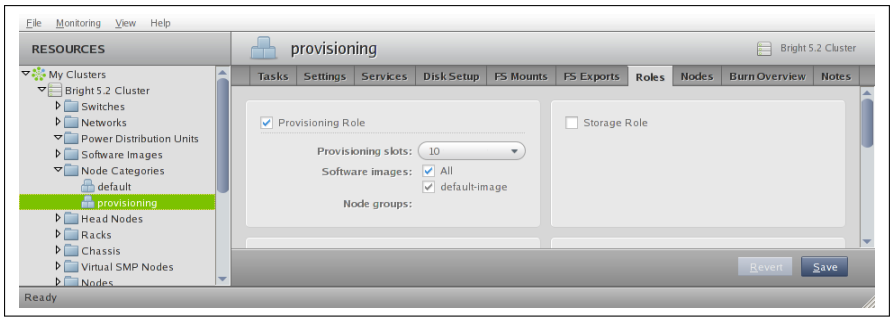

Figure 6.2: cmgui: Configuring A provisioning Role

Selecting the Roles tab in this category displays roles that are part of the provisioning category. Ticking the checkbox of a role assigns the role to the category, and displays the settings that can be configured for this role. The Provisioning slots setting (maxProvisioningNodes in cmsh) decides how many images can be supplied simultaneously from the provisioning node, while the Software images settings (related to the images and allimages attributes of cmsh) decides what images the provisioning node supplies.

The Software image in the Roles tab should not be confused with the Software image selection possibility within the Settings tab, which is the image the provisioning node requests for itself.

6.2.4 Provisioning Nodes: Housekeeping

The head node does housekeeping tasks for the entire provisioning system. Provisioning is done on request for all non-head nodes on a first-come, first-serve basis. Since provisioning nodes themselves, too, need to be provisioned, it means that to cold boot an entire cluster up quickest, the head node should be booted and be up first, followed by provisioning nodes, and finally by all other non-head nodes. Following this start-up se-

quence ensures that all provisioning services are available when the other non-head nodes are started up.

Some aspects of provisioning housekeeping are discussed next:

Provisioning Node Selection

When a node requests provisioning, the head node allocates the task to a provisioning node. If there are several provisioning nodes that can provide the image required, then the task is allocated to the provisioning node with the lowest number of already-started provisioning tasks.

Limiting Provisioning Tasks With `MaxNumberOfProvisioningThreads`

Besides limiting how much simultaneous provisioning per provisioning node is allowed with `maxProvisioningNodes` (section 6.2.1), the head node also limits how many simultaneous provisioning tasks are allowed to run on the entire cluster. This is set using the `MaxNumberOfProvisioningThreads` directive in the head node's CMDaemon configuration file, `/etc/cmd.conf`, as described in Appendix C.

A provisioning request is deferred if the head node is not able to immediately allocate a provisioning node for the task. Whenever an ongoing provisioning task has finished, the head node tries to re-allocate deferred requests.

Provisioning Role Change Notification With `updateprovisioners`

Whenever `updateprovisioners` is invoked, the provisioning system waits for all running provisioning tasks to end before updating all images located on any provisioning nodes by using the images on the head node. It also re-initializes its internal state with the updated provisioning role properties, i.e. keeps track of what nodes are provisioning nodes.

The `updateprovisioners` command can be accessed from the `softwareimage` mode in `cmsh`. It can also be accessed from `cmgui` (figure 6.3):

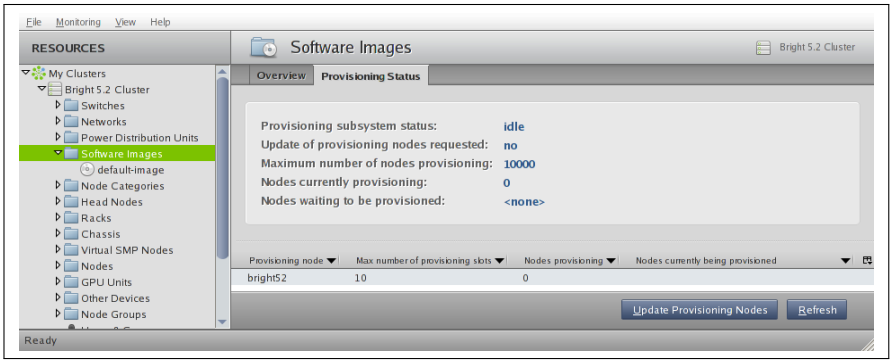

Figure 6.3: `cmgui`: A Button To Update Provisioning Nodes

In examples in section 6.2.2, changes were made to provisioning role attributes for an individual node as well as for a category of nodes.

The `updateprovisioners` command should be run after changing provisioning role settings, to update software images on the provisioners from the software image on the head node to the role settings changes, and to update provisioning role changes.

The `updateprovisioners` command also runs automatically in two

other cases where CMDaemon is involved: during software image changes and during a provision request. If on the other hand, the software image is changed outside of the CMDaemon frontends (`cmgui` and `cmsh`), for example by an administrator adding a file by copying it into place from the bash prompt, then running `updateprovisioners` should be run manually.

In any case, if it is not run during one of the above times, there is also a scheduled time for it to run to ensure that it runs at least once every 24 hours.

The `updateprovisioners` command is in all cases subject to safeguards that prevent it running too often in a short period. Appendix C has details on how the directives `ProvisioningNodeAutoUpdateTimer` and `ProvisioningNodeAutoUpdate` in `cmd.conf` control aspects of how `updateprovisioners` functions.

Example

```
[bright52]% softwareimage updateprovisioners
Provisioning nodes will be updated in the background.

Sun Dec 12 13:45:09 2010 bright52: Starting update of software image(s)\
 provisioning node(s). (user initiated).
[bright52]% softwareimage updateprovisioners [bright52]%
Sun Dec 12 13:45:41 2010 bright52: Updating image default-image on prov\
isioning node node001.
[bright52]%
Sun Dec 12 13:46:00 2010 bright52: Updating image default-image on prov\
isioning node node001 completed.
Sun Dec 12 13:46:00 2010 bright52: Provisioning node node001 was updated
Sun Dec 12 13:46:00 2010 bright52: Finished updating software image(s) \
on provisioning node(s).
```

6.3 The Kernel Image, Ramdisk And Kernel Modules

A *software image* is a complete Linux file system that is to be installed on a non-head node. Chapter 9 describes images and their management in detail.

The head node holds the head copy of the software images. Whenever files in the head copy are changed using CMDaemon, the changes automatically propagate to all provisioning nodes via the `updateprovisioners` command (section 6.2.4).

6.3.1 Booting To A "Good State" Software Image

When nodes boot from the network in simple clusters, the head node supplies them with a *known good state* during node start up. The known good state is maintained by the administrator and is defined using a software image that is kept in a directory of the filesystem on the head node. Supplementary filesystems such as `/home` are served via NFS from the head node by default.

For a diskless node the known good state is copied over from the head node, after which the node becomes available to cluster users.

For a disked node, by default, the hard disk contents on specified local directories of the node are checked against the known good state on

the head node. Content that differs on the node is changed to that of the known good state. After the changes are done, the node becomes available to cluster users.

Each software image contains a Linux kernel and a ramdisk. These are the first parts of the image that are loaded onto a node during early boot. The kernel is loaded first. The ramdisk is loaded next, and contains driver modules for the node's network card and local storage. The rest of the image is loaded after that, during the node-installer stage (section 6.4).

6.3.2 Selecting Kernel Driver Modules To Load Onto Nodes

Kernel Driver Modules With `cmsh`

In `cmsh`, the modules that are to go on the ramdisk can be placed using the `kernelmodules` submode of the `softwareimage` mode. The order in which they are listed is the attempted load order.

Whenever a change is made via the `kernelmodules` submode to the kernel module selection of a software image, CMDaemon automatically runs the `createramdisk` command. The `createramdisk` command regenerates the ramdisk inside the initrd image and sends the updated image to all provisioning nodes, to the image directory, set by default to `/cm/images/default-image/boot/`. The original initrd image is saved as a file with suffix "`.orig`" in that directory. An attempt is made to generate the image for all software images that CMDaemon is aware of, regardless of category assignment, unless the image is protected from modification by CMDaemon with a `FrozenFile` directive (Appendix C).

The `createramdisk` command can also be run from `cmsh` at any time manually by the administrator when in `softwareimage` mode, which is useful if a kernel or modules build is done without using CMDaemon.

Kernel Driver Modules With `cmgui`

In `cmgui` the selection of kernel modules is done from by selecting the `Software Images` resource, and then choosing the "`Kernel Config`" tabbed pane (figure 6.4).

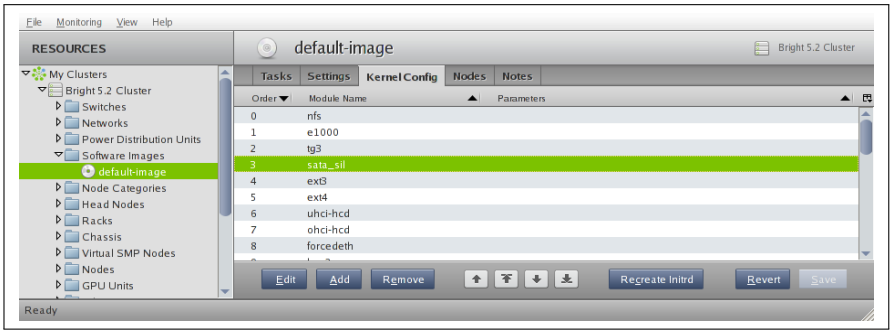

Figure 6.4: `cmgui`: Selecting Kernel Modules For Software Images

The order of module loading can be rearranged by selecting a module and clicking on the arrow keys. Clicking on the "`Recreate Initrd`" button runs the `createramdisk` command.

Implementation Of Kernel Driver Via Ramdisk Or Kernel Parameter

An example of regenerating the ramdisk is seen in section 6.8.5.

Sometimes, testing or setting a kernel driver as a kernel parameter

may be more convenient. How to do that is covered in section 9.3.4.

6.3.3 InfiniBand Provisioning

On clusters that have InfiniBand hardware, it is normally used for data transfer as a service after the nodes have fully booted up (section 4.3). It can also be used for PXE booting (section 6.1.3) and for node provisioning (described here), but these are not normally a requirement. This section (about InfiniBand node provisioning) may therefore safely be skipped in almost all cases.

During node startup on a setup for which InfiniBand networking has been enabled, the `init` process runs the `rdma` script. For SLES and distributions based on versions prior to Red Hat 6, the `openib` script is used instead of the `rdma` script. The script loads up InfiniBand modules into the kernel. When the cluster is finally fully up and running, the use of InfiniBand is thus available for all processes that request it. Enabling InfiniBand is normally set by configuring the InfiniBand network when installing the head node, during the `Additional Network Configuration` screen (figure 2.10).

Provisioning nodes over InfiniBand is not implemented by default, because the `init` process, which handles initialization scripts and daemons, takes place only after the node-provisioning stage launches. InfiniBand modules are therefore not available for use during provisioning, which is why, for default kernels, provisioning in Bright Cluster Manager is done via Ethernet.

Provisioning at the faster InfiniBand speeds rather than Ethernet speeds is however a requirement for some clusters. To get the cluster to provision using InfiniBand requires both of the following two configuration changes to be carried out:

1. configuring InfiniBand drivers for the ramdisk image that the nodes first boot into, so that provisioning via InfiniBand is possible during this pre-`init` stage

2. defining the provisioning interface of nodes that are to be provisioned with InfiniBand. It is assumed that InfiniBand networking is already configured, as described in section 4.3.

   The administrator should be aware that the interface from which a node boots, (conveniently labeled `BOOTIF`), must not be an interface that is already configured for that node in CMDaemon. For example, if `BOOTIF` is the device `ib0`, then `ib0` must not already be configured in CMDaemon. Either `BOOTIF` or the `ib0` configuration should be changed so that node installation can succeed.

How these two changes are carried out is described next:

InfiniBand Provisioning: Ramdisk Image Configuration

An easy way to see what modules must be added to the ramdisk for a particular HCA can be found by running `rdma` (or `openibd`), and seeing what modules do load up on a fully booted system.

One way to do this is to run the following three lines as root:

```
modlist(){ cut -f1 -d" " /proc/modules; }
IB=/etc/init.d/rdma
diff <($IB stop; modlist) <($IB start; modlist)
```

The first line sets up a function `modlist` that lists the modules in use by the system at any instant. The list is obtained by using the `cut` operation to extract only the first column of `/proc/modules`.

For the second line, the InfiniBand `init` script is set to using `rdma`. The `rdma` setting should be replaced by `openibd` when using SLES, or distributions based on versions of Red Hat prior to version 6.

In the third line, the `diff` command then finds the difference between `modlist` output when starting up or stopping InfiniBand, using a bash redirection technique called process substitution.

For `rdma`, the output may display something like:

Example

```
1c1,13
< Unloading OpenIB kernel modules:                          [  OK  ]
---
> Loading OpenIB kernel modules:                            [  OK  ]
> ib_ipoib
> rdma_ucm
> ib_ucm
> ib_uverbs
> ib_umad
> rdma_cm
> ib_cm
> iw_cm
> ib_addr
> ib_sa
> ib_mad
> ib_core
```

As suggested by the output, the modules `ib_ipoib`, `rdma_ucm` and so on are the modules loaded when `rdma` starts, and are therefore the modules that are needed for this particular HCA. Other HCAs may cause different modules to be loaded.

The modules should then be part of the initrd image in order to allow InfiniBand to be used during the node provisioning stage.

The initrd image for the nodes is created by adding the required InfiniBand kernel modules to it. How to load kernel modules into a ramdisk is covered more generally in section 6.3.2. A typical Mellanox HCA may have it created as follows (some text elided in the following example):

Example

```
[root@bright52 ~]# cmsh
[bright52]% softwareimage use default-image
[bright52->softwareimage[default-image]]% kernelmodules
[b...image[default-image]->kernelmodules]% add mlx4_ib
[b...image*[default-image*]->kernelmodules*[mlx4_ib*]]% add ib_ipoib
[b...image*[default-image*]->kernelmodules*[ib_ipoib*]]% add ib_umad
[b...image*[default-image*]->kernelmodules*[ib_umad*]]% commit
[bright52->softwareimage[default-image]->kernelmodules[ib_umad]]%
Tue May 24 03:45:35 2011 bright52: Initial ramdisk for image default-im\
age was regenerated successfully.
```

InfiniBand Provisioning: Network Configuration

It is assumed that the networking configuration for the final system for InfiniBand is configured following the general guidelines of section 4.3. If it is not, that should be checked first to see if all is well with the InfiniBand network.

The provisioning aspect is set by defining the provisioning interface. An example of how it may be set up for 150 nodes with a working InfiniBand interface `ib0` in `cmsh` is:

Example

```
[root@bright52~]# cmsh
%[bright52]% device
[bright52->device]% foreach -n node001..node150 (set provisioninginter\
face ib0)
[bright52->device*]% commit
```

6.4 Node-Installer

After the kernel has started up, and the ramdisk kernel modules are in place on the node, the node launches the node-installer.

The node-installer interacts with CMDaemon on the head node and takes care of the rest of the boot process. Once the node-installer has completed its tasks, the local drive of the node has a complete Linux system. The node-installer ends by calling `/sbin/init` from the local drive and the boot process then proceeds as a normal Linux boot.

The steps the node-installer goes through for each node are:

1. requesting a node certificate (section 6.4.1)
2. deciding or selecting node configuration (section 6.4.2)
3. starting up all network interfaces (section 6.4.3)
4. determining install-mode type and execution mode (section 6.4.4)
5. running `initialize` scripts (section 6.4.5)
6. checking partitions, mounting filesystems (section 6.4.6)
7. synchronizing the local drive with the correct software image (section 6.4.7)
8. writing network configuration files to the local drive (section 6.4.8)
9. creating an `/etc/fstab` file on the local drive (section 6.4.9)
10. installing GRUB bootloader if configured (section 6.4.10)
11. running `finalize` scripts (section 6.4.11)
12. unloading specific drivers no longer needed (section 6.4.12)
13. switching the root device to the local drive and calling `/sbin/init` (section 6.4.13)

These 13 node-installer steps and related matters are described in detail in the corresponding sections 6.4.1–6.4.13.

6.4.1 Requesting A Node Certificate

Each node communicates with the CMDaemon on the head node using a certificate. If no certificate is found, it automatically requests one from CMDaemon running on the head node (figure 6.5).

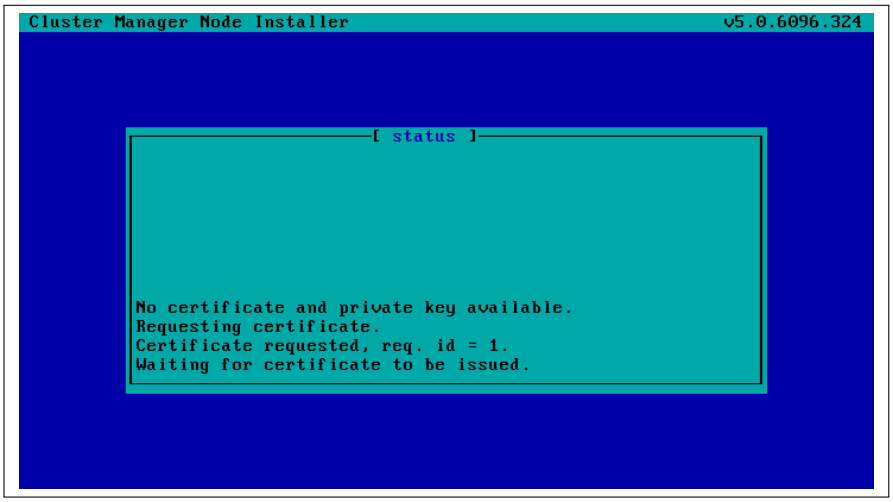

Figure 6.5: Certificate Request

Null Cipher Certificates

By default, a null cipher is used on internal networks such as `internalnet`, to keep communications speedy. Using encryption on even these networks is sometimes a requirement in very unusual situations. In that case, setting the advanced configuration flag `AllowNullCipherNetwork=0` in `cmd.conf` (appendix C) forces encryption on after CMDaemon is restarted. By default, its value is `1`.

Certificate Auto-signing

By default, *certificate auto-signing* means the cluster management daemon automatically issues a certificate to any node that requests a certificate.

For untrusted networks it may be wiser to approve certificate requests manually to prevent new nodes being added automatically without getting noticed. Disabling certificate auto-signing can then be done by issuing the `autosign off` command from `cert` mode in `cmsh`.

Section 3.3 has more information on certificate management in general.

Example

Disabling certificate auto-sign mode:

```
[bright52]% cert autosign
on
[bright52]% cert autosign off
off
[bright52]% cert autosign
off
[bright52]%
```

Certificate Storage And Removal Implications

After receiving a valid certificate, the node-installer stores it in `/cm/node-installer/certificates/<node mac address>/` on the head node. This directory is NFS exported to the nodes, but can only be accessed by the `root` user. The node-installer does not request a new certificate if it finds a certificate in this directory, valid or invalid.

If an invalid certificate is received, the screen displays a communication error. Removing the node's corresponding certificate directory allows the node-installer to request a new certificate and proceed further.

6.4.2 Deciding Or Selecting Node Configuration

Once communication with the head node CMDaemon is established, the node-installer tries to identify the node it is running on so that it can select a configuration from CMDaemon's record for it, if any such record exists. It correlates any node configuration the node is expected to have according to network hardware detected. If there are issues during this correlation process then the administrator is prompted to select a node configuration until all nodes finally have a configuration.

Possible Node Configuration Scenarios

The correlations process and corresponding scenarios are now covered in more detail:

It starts with the node-installer sending a query to CMDaemon to check if the MAC address used for net booting the node is already associated with a node in the records of CMDaemon. In particular, it checks the MAC address for a match against the existing *node configuration* properties, and decides whether the node is *known* or *new*.

- the node is **known** if the query matches a node configuration. It means that node has been booted before.
- the node is **new** if no configuration is found.

In both cases the node-installer then asks CMDaemon to find out if the node is connected to an Ethernet switch, and if so, to which port. Setting up Ethernet switches for port detection is covered in section 4.5.

If a port is detected for the node, the node-installer queries CMDaemon for a node configuration associated with the detected Ethernet switch port. If a port is not detected for the node, then either the hardware involved with port detection needs checking, or a node configuration must be selected manually.

There are thus several scenarios:

1. The node is new, and an Ethernet switch port is detected. A previous configuration associated with the port is found. The node-installer suggests to the administrator that the new node use this configuration, and displays the configuration along with a confirmation dialog (figure 6.6). This suggestion can be interrupted, and other node configurations can be selected manually instead through a sub-dialog (figure 6.7). By default (in the main dialog), the original suggestion is accepted after a timeout.

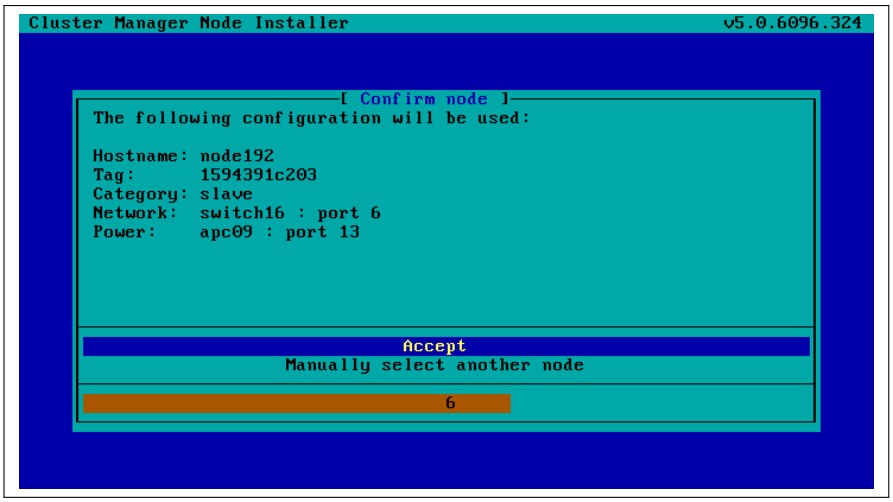

Figure 6.6: Scenarios: Configuration Found, Confirm Node Configuration

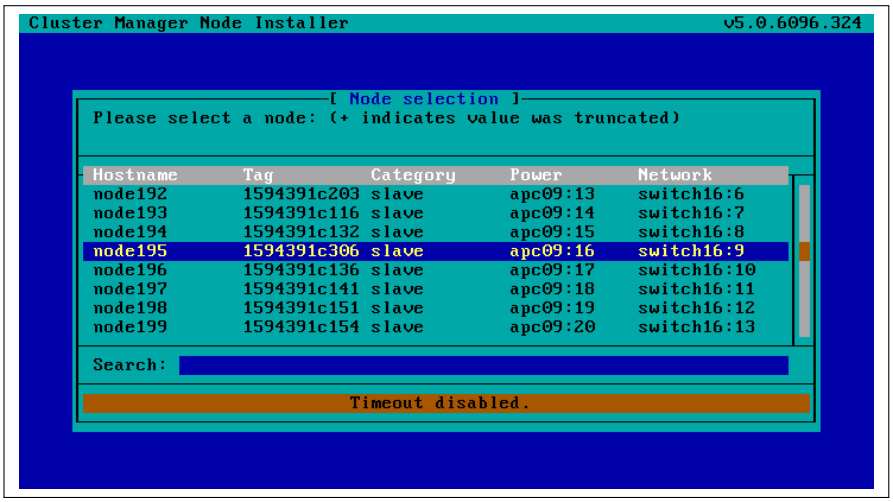

Figure 6.7: Scenarios: Node Selection Sub-Dialog

2. The node is new, and an Ethernet switch port is detected. A previous configuration associated with the port is not found. The node-installer then displays a dialog that allows the administrator to either retry Ethernet switch port detection (figure 6.8) or to drop into a sub-dialog to manually select a node configuration (figure 6.7). By default, port detection is retried after a timeout.

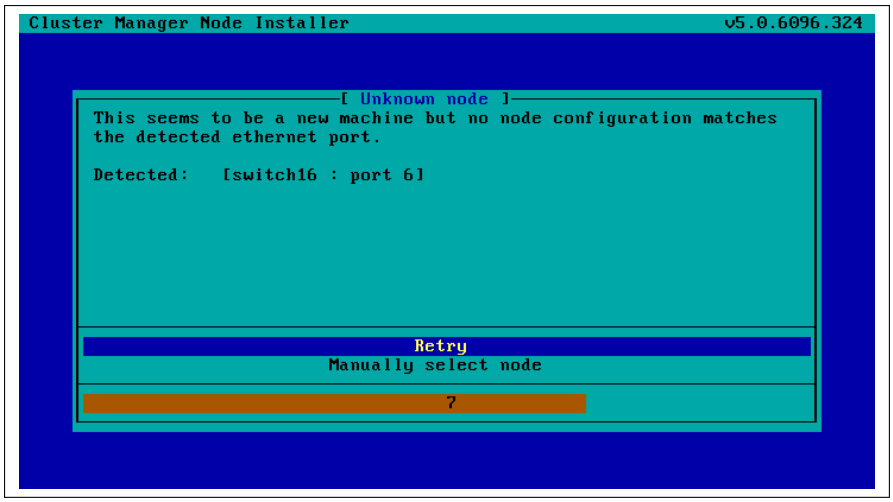

Figure 6.8: Scenarios: Unknown Node, Ethernet Port Detected

3. The node is new, and an Ethernet switch port is not detected. The node-installer then displays a dialog that allows the user to either retry Ethernet switch port detection (figure 6.9) or to drop into a sub-dialog to manually select a node configuration (figure 6.7). By default, port detection is retried after a timeout.

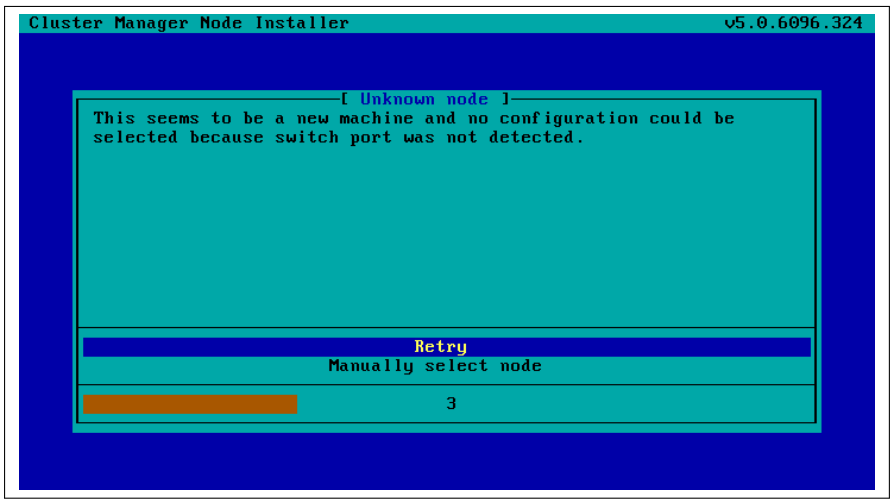

Figure 6.9: Scenarios: Unknown Node, No Ethernet Port Detected

4. The node is known, and an Ethernet switch port is detected. The configuration associated with the port is the same as the configuration associated with the node's MAC address. The node-installer then displays the configuration as a suggestion along with a confirmation dialog (figure 6.6). The suggestion can be interrupted, and other node configurations can be selected manually instead through a sub-dialog (figure 6.7). By default (in the main dialog), the original suggestion is accepted after a timeout.

5. The node is known, and an Ethernet switch port is detected. However, the configuration associated with the port is not the same as the configuration associated with the node's MAC address. This is

called a *port mismatch*. This type of port mismatch situation occurs typically during a mistaken *node swap*, when two nodes are taken out of the cluster and returned, but their positions are swapped by mistake (or equivalently, they are returned to the correct place in the cluster, but the switch ports they connect to are swapped by mistake). To prevent configuration mistakes, the node-installer displays a port mismatch dialog (figure 6.10) allowing the user to retry, accept a node configuration that is associated with the detected Ethernet port, or to manually select another node configuration via a sub-dialog (figure 6.7). By default (in the main port mismatch dialog), port detection is retried after a timeout.

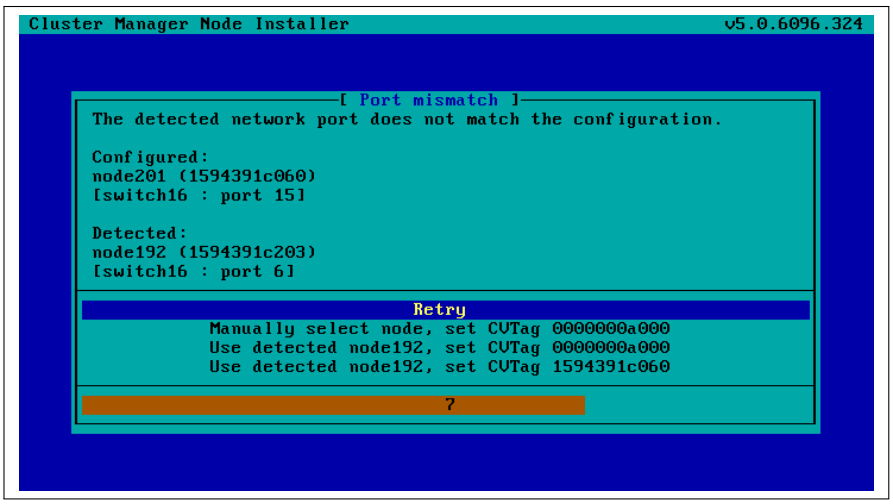

Figure 6.10: Scenarios: Port Mismatch Dialog

6. The node is known, and an Ethernet switch port is not detected. However, the configuration associated with the node's MAC address does have an Ethernet port associated with it. This is also considered a port mismatch. To prevent configuration mistakes, the node-installer displays a port mismatch dialog similar to figure 6.10, allowing the user to retry or to drop into a sub-dialog and manually select a node configuration. By default (in the port mismatch dialog), port detection is retried after a timeout.

7. The node is known, and an Ethernet switch port is detected. However, the configuration associated with the node's MAC address has no Ethernet switch port associated with it. This is not considered a port mismatch but an unset switch port configuration, and it typically occurs if switch port configuration has not been carried out, whether by mistake or deliberately. The node-installer displays the configuration as a suggestion along with a confirmation dialog (figure 6.11). The suggestion can be interrupted, and other node configurations can be selected manually instead using a sub-dialog. By default (in the main dialog) the configuration is accepted after a timeout.

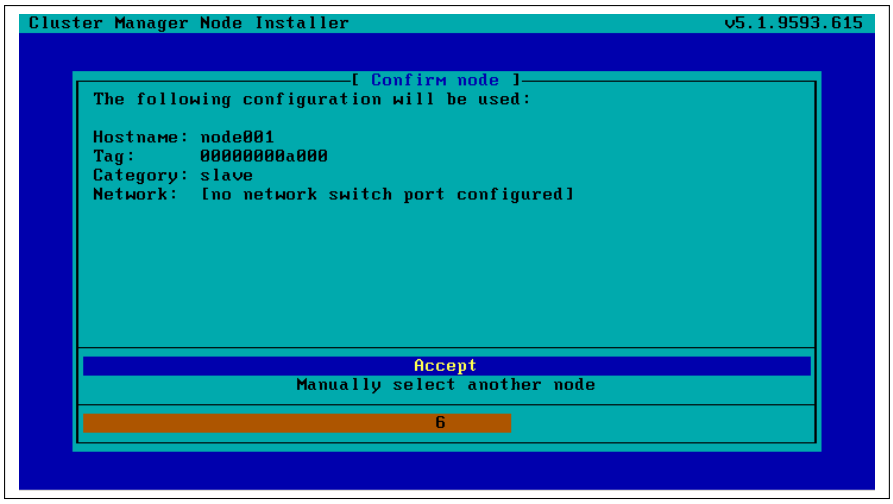

Figure 6.11: Scenarios: Port Unset Dialog

A truth table summarizing the scenarios is helpful:

Scenario	Node known?	Switch port detected?	Switch port configuration found?	Switch port configuration conflicts with node configuration?
1	No	Yes	Yes	No
2	No	Yes	No	No
3	No	No	No	No
4	Yes	Yes	Yes	No
5	Yes	Yes	Yes	Yes (configurations differ)
6	Yes	No	Yes	Yes (port expected by MAC configuration not found)
7	Yes	Yes	No	No (port not expected by MAC configuration)

In these scenarios, whenever the user manually selects a node configuration in the prompt dialog, an attempt to detect an Ethernet switch port is repeated. If a port mismatch still occurs, it is handled by the system as if the user has not made a selection.

Summary Of Behavior During Hardware Changes

The logic of the scenarios means that an unpreconfigured node always boots to a dialog loop requiring manual intervention during a first install (scenarios 2 and 3). For subsequent boots the behavior is:

- If the node MAC hardware has changed (scenarios 1, 2, 3):
  - if the node is new and the detected port has a configuration,

the node automatically boots to that configuration (scenario 1).

- else manual intervention is needed (scenarios 2, 3)

- If the node MAC hardware has not changed (scenarios 4, 5, 6, 7):
  - if there is no port mismatch, the node automatically boots to its last configuration (scenarios 4, 7).
  - else manual intervention is needed (scenarios 5, 6).

The `newnodes` Command

`newnodes` basic use: New nodes that have not been configured yet can be detected using the `newnodes` command from within the `device` mode of `cmsh`. A new node is detected when it reaches the node installer stage after booting, and contacts the head node.

Example

```
[bright52->device]% newnodes
The following nodes (in order of appearance) are waiting to be assigned:
MAC               First appeared                Detected on switch port
----------------  ----------------------------  -----------------------
00:0C:29:01:0F:F8  Mon, 14 Feb 2011 10:16:00 CET  [no port detected]
```

At this point the node installer is seen by the administrator to be looping, waiting for input on what node name is to be assigned to the new node.

The nodes can be uniquely identified by their MAC address or switch port address.

The port and switch to which a particular MAC address is connected can be discovered by using the `showport` command (section 4.5.4). After confirming that they are appropriate, the `ethernetswitch` property for the specified device can be `set` to the port and switch values.

Example

```
[bright52->device]% showport 00:0C:29:01:0F:F8
switch01:8
[bright52->device]% set node003 ethernetswitch switch01:8
[bright52->device*]% commit
```

When the node name (`node003` in the preceding example) is assigned, the node installer stops looping and goes ahead with the installation to the node.

The preceding basic use of `newnodes` is useful for small numbers of nodes. For larger number of nodes, the advanced options of `newnodes` may help carry out node-to-MAC assignment with less effort.

`newnodes` advanced use—options: The list of MAC addresses discovered by a `newnodes` command can be assigned in various ways to nodes specified by the administrator. Node objects should be created in advance to allow the assignment to take place. The easiest way to set up node objects is to use the `--clone` option of the `foreach` command (section 3.5.5).

The advanced options of `newnodes` are particularly useful for quickly assigning node names to specific physical nodes. All that is needed is to

power the nodes up in the right order. For nodes with the same hardware, the node that is powered up first reaches the stage where it tries to connect with the node installer first. So its MAC address is detected first, and arrives on the list generated by `newnodes` first. If some time after the first node is powered up, the second node is powered up, then its MAC address becomes the second MAC address on the list, and so on for the third, fourth, and further nodes.

When assigning node names to a physical node, on a cluster that has no such assignment already, the first node that arrived on the list gets assigned the name `node001`, the second node that arrived on the list gets assigned the name `node002` and so on.

The advanced options are shown in `device` mode by running the `help newnodes` command. The options can be introduced as being of three kinds: straightforward, grouping, and miscellaneous:

- The straightforward options:

  `-n|--nodes`

  `-w|--write`

  `-s|--save`

  Usually the most straightforward way to assign the nodes is to use the `-n` option, which accepts a list of nodes, together with a `-w` or `-s` option. The `-w` (`--write`) option sets the order of nodes to the corresponding order of listed MAC addresses, and is the same as setting an object in `cmsh`. The `-s` (`--save`) option is the same as setting and committing an object in `cmsh`, so `-s` implies a `-w` option is run at the same time.

  So, for example, if 8 new nodes are discovered by the node installer on a cluster with no nodes so far, then:

  Example

  ```
  [bright52->device]% newnodes -w -n node001..node008
  ```

  assigns (but does not commit) the sequence node001 to node008 the new MAC address according to the sequence of MAC addresses displaying on the list.

- The grouping options:

  `-g|--group`

  `-c|--category`

  `-h|--chassis`

  `-r|--rack`

  options: The "`help newnodes`" command in `device` mode shows assignment options other than `-n` for a node range are possible. For example, the assignments can also be made for a group (`-g`), per category (`-c`), per chassis (`-h`), and per rack (`-r`).

- The miscellaneous options:

`-f|--force`

`-o|--offset`

By default, the `newnodes` command fails when it attempts to set a node name that is already taken. The `-f` (`--force`) option forces the new MAC address to be associated with the old node name. When used with an assignment grouping, (node range, group, category, chassis, or rack) the entire grouping loses their assigned node-to-MAC assignment and get new assignments. The `-f` option should therefore be used with care.

The `-o` (`--offset`) option takes a number <*number*> and skips <*number*> nodes in the list of detected unknown nodes, before setting or saving values from the assignment grouping.

Examples of how to use the advanced options follow.

`newnodes` **advanced use—range assignment behavior example:** For example, supposing there is a cluster with nodes assigned all the way up to node022. That is, CMDaemon knows what node is assigned to what MAC address. For the discussion that follows, the three nodes node020, node021, node022 can be imagined as being physically in a rack of their own. This is simply to help to visualize a layout in the discussion and tables that follow and has no other significance. An additional 3 new, that is unassigned, nodes are placed in the rack, and allowed to boot and get to the node installer stage.

The `newnodes` command discovers the new MAC addresses of the new nodes when they reach their node-installer stage, as before (the switch port column is omitted in the following text for convenience):

Example

```
[bright52->device]% newnodes
MAC                First appeared
-----------------  ----------------------------
00:0C:29:EF:40:2A  Tue, 01 Nov 2011 11:42:31 CET
00:0C:29:95:D3:5B  Tue, 01 Nov 2011 11:46:25 CET
00:0C:29:65:9A:3C  Tue, 01 Nov 2011 11:47:13 CET
```

The assignment of MAC to node address could be carried out as follows:

Example

```
[bright52->device]% newnodes -s -n node023..node025
MAC                 First appeared                Hostname
------------------- ----------------------------  --------
00:0C:29:EF:40:2A   Tue, 01 Nov 2011 11:42:31 CET  node023
00:0C:29:95:D3:5B   Tue, 01 Nov 2011 11:46:25 CET  node024
00:0C:29:65:9A:3C   Tue, 01 Nov 2011 11:47:13 CET  node025
```

Once this is done, the node-installer is able to stop looping, and to go ahead and install the new nodes with an image.

The physical layout in the rack may then look as indicated by this:

before	after	MAC
node020	node020	
node021	node021	
node022	node022	
	node023	...A
	node024	...B
	node025	...C

Here, `node023` is the node with the MAC address ending in `A`.

If instead of the previous `newnodes` command, an offset of 1 is used to skip assigning the first new node:

Example

```
[bright52->device]% newnodes -s -o 1 node024..node025
```

then the rack layout looks like:

before	after	MAC
node020	node020	
node021	node021	
node022	node022	
	unassigned	...A
	node024	...B
	node025	...C

Here, *unassigned* is where `node023` of the previous example is physically located, that is, the node with the MAC address `...A`. The lack of assignment means there is actually no association of the name `node023` with that MAC address, due to the `newnodes` command having skipped over it with the `-o` option.

If instead the assignment is done with:

Example

```
[bright52->device]% newnodes -s 1 node024..node026
```

then the `node023` name is unassigned, and the name `node024` is assigned instead to the node with the MAC address `...A`, so that the rack layout looks like:

before	after	MAC
node020	node020	
node021	node021	
node022	node022	
	node024	...A
	node025	...B
	node026	...C

`newnodes` **advanced use—assignment grouping example:** Node range assignments are one way of using `newnodes`. However assignments can also be made to a category, a rack, or a chassis. For example, with `cmgui`, assigning node names to a rack can be done from the `Nodes` resource and selecting the `Settings` tab. Within the tab, the `Rack` values can be set appropriately, and saved for each node (figure 6.12).

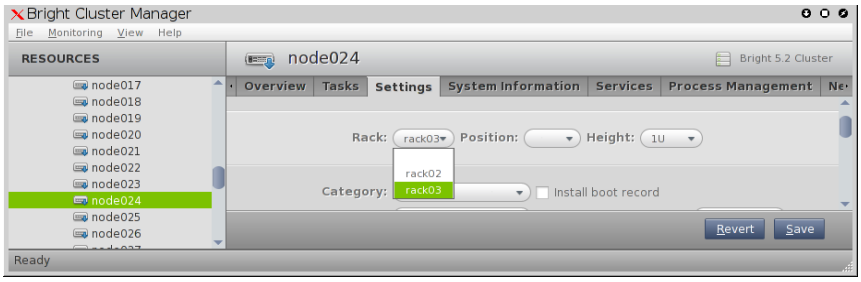

Figure 6.12: Assigning A Node To A Rack

In `cmsh`, the assignment of multiple node names to a rack can conveniently be done with a `foreach` loop from within `device` mode:

Example

```
[bright52->device]% foreach -n node020..node029 (set rack rack02)
[bright52->device*]% commit
[bright52->device]% foreach -n node030..node039 (set rack rack03)
[bright52->device*]% commit
```

The assignment of node names with the physical node in the rack can then be arranged as follows: If the nodes are identical hardware, and are powered up in numerical sequence, from `node020` to `node039`, with a few seconds in between, then the list that the basic `newnodes` command (without options) displays is arranged in the same numerical sequence. Assigning the list in the rack order can then be done by running:

Example

```
[bright52->device]% newnodes -s -r rack02..rack03
```

If it turns out that the boot order was done very randomly and incorrectly for all of `rack02`, and that the assignment for `rack02` needs to be done again, then a simple way to deal with it is to clear out all of the `rack02` current MAC associations, and redo them according to the correct boot order:

Example

```
[bright52->device]% foreach -r rack02 ( clear mac ) ; commit
  [...removes MAC association with nodes from CMDaemon...]

  [...now reboot nodes in rack02 in sequence...]

[bright52->device]% newnodes

  [...shows sequence as the nodes come up..]
```

```
[bright52->device]% newnodes -s -r rack02

  [...assigns sequence in boot order...]
```

`newnodes` **advanced use—assignment forcing example:** The `--force` option can be used in the following case: Supposing that `node022` fails, and a new node hardware comes in to replace it. The new node has a new MAC address. So, as explained by scenario 3 (section 6.4.2), if there is no switch port assignment in operation for the nodes, then the node installer loops around, waiting for intervention.[1]

This situation can be dealt with from the command line by:

- accepting the node configuration at the head node console, via a sub-dialog
- accepting the node configuration via `cmsh`, without needing to be at the head node console:

```
[bright52->device]% newnodes -s -f -n node022
```

Node Identification Wizard

The *node identification wizard* can be accessed from the tabbed pane under the `Nodes` resource (figure 6.13). It can also be accessed by double-clicking, in the event viewer, on the event message "`Nodes waiting to be identified. Double click event to assign`".

The node identification wizard is roughly the `cmgui` equivalent to the `newnodes` command of `cmsh`. Like `newnodes`, the wizard lists the MAC address and switch port of any unassigned node that the head node detects. Also, like `newnodes`, it can help assign a node name to the node, assuming the node object exists. After assignment is done, a prompt to save the new status appears.

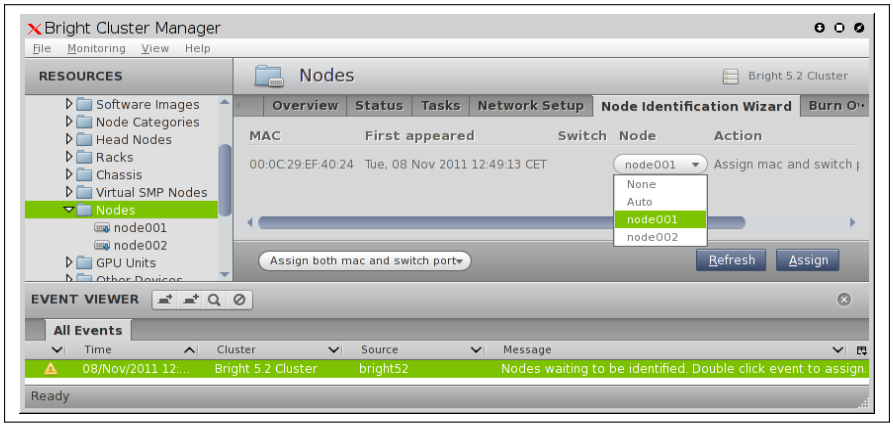

Figure 6.13: Node Identification Wizard

The most useful way of using the wizard is for node assignment in large clusters.

[1] with switch port assignment in place, scenario 1 means the new node simply boots up by default and becomes the new `node022` without further intervention

To do this, it is assumed that the node objects have already been created for the new nodes. The creation of the node objects means that the node names exist, and so assignment to the node names is able to take place. An easy way to create nodes, set their provisioning interface, and set their IP addresses is described in the section on the *node creation wizard* (section 6.7.2). Node objects can also be created by running `cmsh`'s `foreach` loop command on a node with a `-clone` option (section 3.5.5).

The nodes are also assumed to be set for net booting, typically set from a BIOS setting.

The physical nodes are then powered up in an arranged order. Because they are unknown new nodes, the node-installer keeps looping after a timeout. The head node in the meantime detects the new MAC addresses and switch ports in the sequence in which they first have come up and lists them in that order.

By default, all these newly detected nodes are set to `auto`, which means their numbering goes up sequentially from whatever number is assigned to the preceding node in the list. Thus, if there are 10 new unassigned nodes that are brought into the cluster, and the first node in the list is assigned to the first available number, say `node327`; then clicking on assign automatically assigns the remaining nodes to the next available numbers, say `node328-node337`.

After the assignment, the node-installer looping process on the new nodes notices that the nodes are now known. The node-installer then breaks out of the loop, and installation goes ahead without any intervention needed at the node console.

6.4.3 Starting Up All Network Interfaces

At the end of section 6.4.2, the node-installer knows which node it is running on, and has decided what its node configuration is.

It now gets on with setting up the IP addresses on the interfaces required for the node-installer, while taking care of matters that come up on the way:

Avoiding Duplicate IP Addresses

The node-installer brings up all the network interfaces configured for the node. Before starting each interface, the node-installer first checks if the IP address that is about to be used is not already in use by another device. If it is, then a warning and retry dialog is displayed until the IP address conflict is resolved.

Using `BOOTIF` To Specify The Boot Interface

`BOOTIF` is a special name for one of the possible interfaces. The node-installer automatically translates `BOOTIF` into the name of the device, such as eth0 or eth1, used for network booting. This is useful for a machine with multiple network interfaces where it can be unclear whether to specify, for example, eth0 or eth1 for the interface that was used for booting. Using the name `BOOTIF` instead means that the underlying device, eth0 or eth1 in this example, does not need to be specified in the first place.

Halting On Missing Kernel Modules For The Interface

For some interface types like VLAN and channel bonding, the node-installer halts if the required kernel modules are not loaded or are loaded

with the wrong module options. In this case the kernel modules configuration for the relevant software image should be reviewed. Recreating the ramdisk and rebooting the node to get the interfaces up again may be necessary, as described in section 6.8.5.

Initializing IPMI Interfaces

IPMI interfaces, if present and set up in the node's configuration, are also initialized with correct IP address, netmask and user/password settings.

Restarting The Network Interfaces

At the end of this step (i.e. section 6.4.3) the network interfaces are up. When the node-installer has completed the remainder of its 13 steps (sections 6.4.4–6.4.13), control is handed over to the local `init` process running on the local drive. During this handover, the node-installer brings down all network devices. These are then brought back up again by `init` by the distribution's standard networking `init` scripts, which run from the local drive and expect networking devices to be down to begin with.

6.4.4 Determining Install-mode Type And Execution Mode

Stored *install-mode* values decide whether synchronization is to be applied fully to the local drive of the node, only for some parts of its filesystem, not at all, or even whether to drop into a maintenance mode instead.

Related to install-mode values are execution mode values that determine whether to apply the install-mode values to the next boot, to new nodes only, to individual nodes or to a category of nodes.

Related to execution mode values is the confirmation requirement toggle value in case of a full installation is to take place.

These values are merely determined at this stage; nothing is executed yet.

Install-mode Values

The install-mode can have one of four values: `AUTO`, `FULL`, `MAIN` and `NOSYNC`.

- If the install-mode is set to `FULL`, the node-installer re-partitions, creates new file systems and synchronizes a full image onto the local drive. This process wipes out all pre-boot drive content.
- If the install-mode is set to `AUTO`, the node-installer checks the partition table and file systems of the local drive against the node's stored configuration. If these do not match because, for example, the node is new, or if they are corrupted, then the node-installer recreates the partitions and file systems by carrying out a `FULL` install. If however the drive partitions and file systems are healthy, the node-installer only does an incremental software image synchronization. Synchronization tends to be quick because the software image and the local drive usually do not differ much.

  Synchronization also removes any extra local files that do not exist on the image, for the files and directories considered. Section 6.4.7 gives details on how it is decided what files and directories are considered.
- If the install-mode is set to `MAIN`, the node-installer halts in maintenance mode, allowing manual investigation of specific problems.

The local drive is untouched.

- If the install-mode is set to NOSYNC, and the partition and filesystem check matches the stored configuration, then the node-installer skips synchronizing the image to the node, so that contents on the local drive persist from the previous boot. If however the partition or filesystem does not match the stored configuration, a FULL image sync is triggered.

Install-mode's Execution Modes

Execution of an install-mode setting is possible in several ways, both permanently or just temporarily for the next boot. Execution can be set to apply to categories or individual nodes. The node-installer looks for install-mode execution settings in this order:

1. The "New node installmode" property of the node's category. This decides the install mode for a node that is detected to be new.

   It can be set using cmgui (figure 6.14):

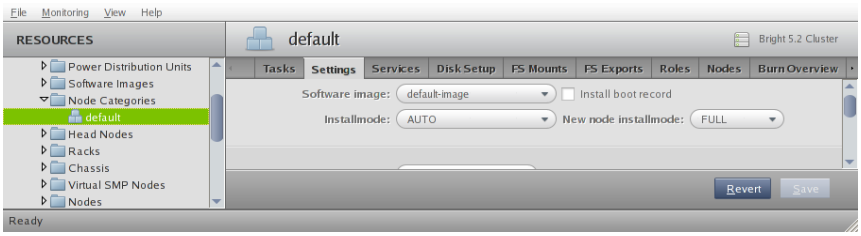

Figure 6.14: cmgui Install-mode Settings Under Node Category

   or using cmsh with a one-liner like:

```
cmsh -c "category use default; set newnodeinstallmode FULL; commit"
```

   By default, the "New node installmode" property is set to FULL.

2. The Install-mode setting as set by choosing a PXE menu option on the console of the node before it loads the kernel and ramdisk (figure 6.15). This only affects the current boot. By default the PXE menu install mode option is set to AUTO.

Figure 6.15: PXE Menu With Install-mode Set To AUTO

3. The "Next boot install-mode" property of the node configuration. This can be set using cmgui (figure 6.16):

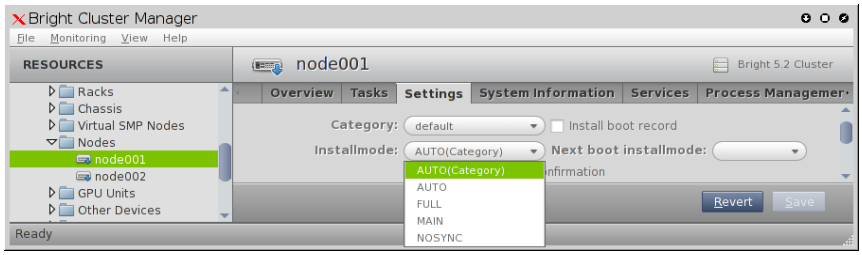

Figure 6.16: cmgui Install-mode Settings For The Node

It can also be set using cmsh with a one-liner like:

```
cmsh -c "device use node001; set nextinstallmode FULL; commit"
```

The property is cleared when the node starts up again, after the node-installer finishes its installation tasks. So it is empty unless specifically set by the administrator during the current uptime for the node.

4. The install-mode property can be set in the node configuration using cmgui (figure 6.16), or using cmsh with a one-liner like:

```
cmsh -c "device use node001; set installmode FULL; commit"
```

By default, the install-mode property is auto-linked to the property set for install-mode for that category of node. Since the property for that node's category defaults to AUTO, the property for the install-mode of the node configuration defaults to "AUTO (Category)".

5. The `install-mode` property of the node's category. This can be set using `cmgui` (figure 6.14), or using `cmsh` with a one-liner like:

   ```
   cmsh -c "category use default; set installmode FULL; commit"
   ```

   As already mentioned in a previous point, the install-mode is set by default to `AUTO`.

6. A dialog on the console of the node (figure 6.17) gives the user a last opportunity to overrule the install-mode value as determined by the node-installer. By default, it is set to `AUTO`:

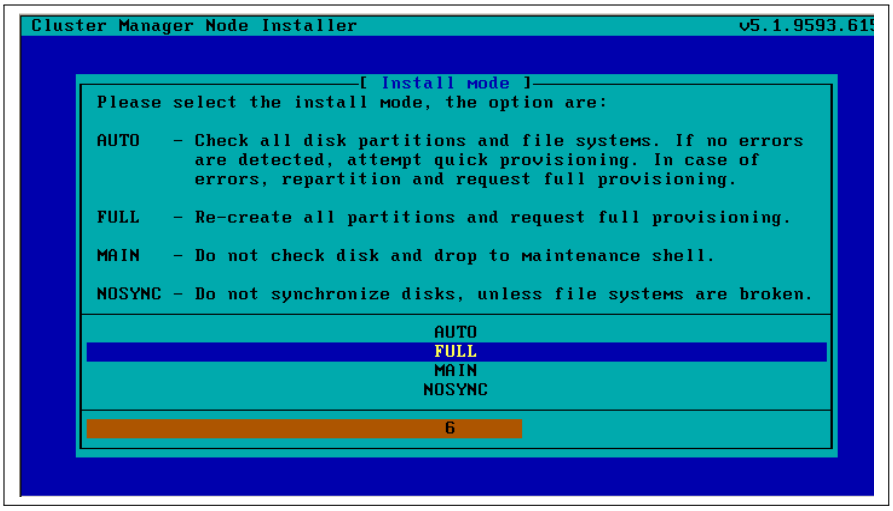

Figure 6.17: Install-mode Setting Option During Node-Installer Run

Require Full Install Confirmation Toggle

Related to execution mode values is the "`Require full install confirmation`" value. This must be set in order to ask for a confirmation in case a FULL installation is about to take place. If set, the node-installer waits for a confirmation before going ahead with the FULL install.

The property can be set in the node configuration with `cmgui` (figure 6.18):

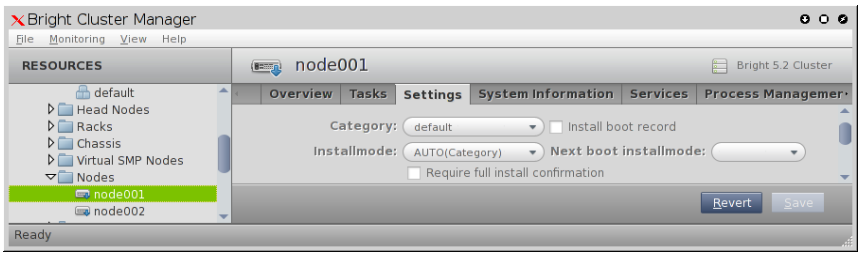

Figure 6.18: `cmgui` Require Full Install Confirmation Checkbox

Alternatively, it can be set using a `cmsh` one-liner like:

```
cmsh -c "device use node001; set requirefullinstallconfirmation yes;\
 commit"
```

The reason behind having such a setting available is that a FULL installation can be triggered by a change in disk/partition, or a change in the MAC address. If that happens, then:

- considering a drive, say, /dev/sda that fails, this means that any drive /dev/sdb would then normally become /dev/sda upon reboot. In that case an unwanted FULL install would not only be triggered by an install-mode settings of FULL, but also by the install-mode settings of AUTO or NOSYNC. Having the new, "accidental" /dev/sda have a FULL install is unlikely to be the intention, since it would probably contain useful data that the node installer earlier left untouched.

- considering a node with a new MAC address, but with local storage containing useful data from earlier. In this case, too, an unwanted FULL install would not only be triggered by an install-mode setting of FULL, but also by the install-mode settings AUTO or NOSYNC.

Thus, in these two cases, having a confirmation required before proceeding with overwriting local storage contents is a good idea.

By default, no confirmation is required when a FULL installation is about to take place.

6.4.5 Running Initialize Scripts

An *initialize script* is used when custom commands need to be executed before checking partitions and mounting devices. For example, to initialize some unsupported hardware, or to do a RAID configuration lookup for a particular node. In such cases the custom commands are added to an initialize script.

An initialize script can be added to both a node's category and the node configuration. The node-installer first runs an initialize script, if it exists, from the node's category, and then an initialize script, if it exists, from the node's configuration.

The node-installer sets several environment variables which can be used by the initialize script. Appendix E contains an example script documenting these variables.

Related to the initialize script are:

- The finalize script (section 6.4.11). This may run after node provisioning is done, but just before the init process on the node run.

- The imageupdate_initialize and imageupdate_finalize scripts, which may run when the imageupdate command runs (section 6.6.2).

6.4.6 Checking Partitions, Mounting File Systems

In section 6.4.4 the node-installer determines the install-mode value, along with when to apply it to a node. The install-mode value defaults mostly to AUTO. If AUTO applies to the current node, it means the node-installer then checks the partitions of the local drive and its file systems and recreates them in case of errors. Partitions are checked by comparing the partition layout of the local drive(s) against the drive layout as configured in the node's category configuration and the node configuration.

After the node-installer checks the drive(s) and, if required, recreates the layout, it mounts all file systems to allow the drive contents to be synchronized with the contents of the software image.

If install-mode values of FULL or MAIN apply to the current node instead, then no partition checking or filesystem checking is done by the node-installer.

If the install-mode value of NOSYNC applies, then if the partition and filesystem checks both show no errors, the node starts up without getting an image synced to it from the provisioning node. If the partition or the filesystem check show errors, then the node partition is rewritten, and a known good image is synced across.

The node-installer is capable of creating advanced drive layouts, including software RAID and LVM setups. Drive layout examples and relevant documentation are in appendix D.

6.4.7 Synchronizing The Local Drive With The Software Image

After having mounted the local filesystems, these can be synchronized with the contents of the software image associated with the node (through its category). Synchronization is skipped if NOSYNC is set, and takes place if install-mode values of FULL or AUTO are set. Synchronization is delegated by the node-installer to the CMDaemon provisioning system. The node-installer just sends a provisioning request to CMDaemon on the head node.

For an install-mode of FULL, or for an install-mode of AUTO where the local filesystem is detected as being corrupted, full provisioning is done. For an install-mode of AUTO where the local filesystem is healthy and agrees with that of the software image, sync provisioning is done.

On receiving the provisioning request, CMDaemon assigns the provisioning task to one of the provisioning nodes. The node-installer is notified when image synchronization starts, and also when the image synchronization task ends—whether it is completed successfully or not.

Exclude Lists: excludelistsyncinstall And excludelistfullinstall
What files are synchronized is decided by an *exclude list*. An exclude list is a property of the node category, and is a list of directories and files that are excluded from consideration during synchronization. The excluded list that is used is decided by the type of synchronization chosen: full or sync:

- A full type of synchronization rewrites the partition table of the node, then copies the filesystem from a software image to the node, using a list to specify files and directories to exclude from consideration when copying over the file system. The list of exclusions used is specified by the excludelistfullinstall property.

  The intention of full synchronization is to allow a complete working filesystem to be copied over from a known good software image to the node. By default the excludelistfullinstall list contains /proc/, /sys/, and lost+found/, which have no content in Bright Cluster Manager's default software image. The list can be modified to suit the requirements of a cluster, but it is recommended to have the list adhere to the principle of allowing a complete working node filesystem to be copied over from a known good software image.

- A sync type of synchronization uses the excludelistsyncinstall property to specify what files and directories to exclude from con-

sideration when copying parts of the filesystem from a known good software image to the node. The `excludelistsyncinstall` property is in the form of a list of exclusions, or more accurately in the form of two sub-lists.

The contents of the sub-lists specify the parts of the filesystem that should be retained or not copied over from the software image during `sync` synchronization when the node is booting. The intention behind this is to have the node boot up quickly, updating only the files from the image to the node that need updating due to the reboot of the node, and otherwise keeping files that are already on the node hard disk unchanged. The contents of the sub-lists are thus items such as the node log files, or items such as the `/proc` and `/sys` pseudo-filesystems which are generated during node boot.

The administrator should be aware that nothing on a node hard drive can be regarded as persistent because a FULL sync takes place if any error is noticed during a partition or filesystem check.

Anything already on the node that matches the content of these sub-lists is not overwritten by image content during an `excludelistsyncinstall` sync. However, image content that is not on the node is copied over to the node only for items matching the first sub-list. The remaining files and directories on the node, that is, the ones that are not in the sub-lists, lose their original contents, and are copied over from the software image.

A `cmsh` one-liner to get an exclude list for a category is:

```
cmsh -c "category use default; get excludelistfullinstall"
```

Similarly, to set the list:

```
cmsh -c "category use default; set excludelistfullinstall; commit"
```

where a text-editor opens up to allow changes to be made to the list. Figure 6.19 illustrates how the setting can be modified via `cmgui`.

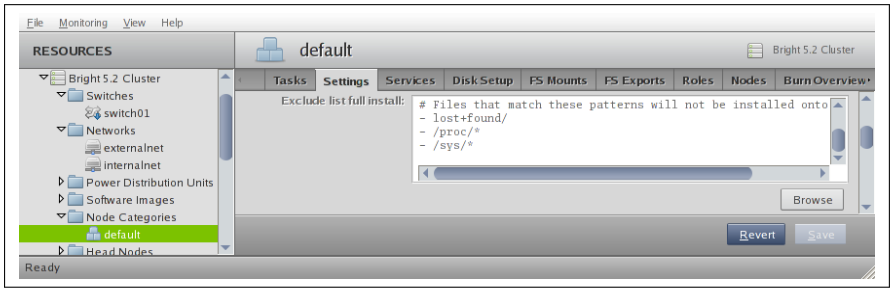

Figure 6.19: Setting up exclude lists with `cmgui` for provisioning

Image synchronization is done using `rsync`, and the syntax of the items in the exclude lists conforms to the "`INCLUDE/EXCLUDE PATTERN RULES`" section of the `rsync(1)` man page, which includes patterns such as "`**`", "`?`", and "`[[:alpha:]]`".

The `excludelistfullinstall` and `excludelistsyncinstall` properties decide how a node synchronizes to an image during boot. For a node that is already fully up, the related `excludelistupdate` property decides

how a running node synchronizes to an image without a reboot event, and is discussed in section 6.6.

Interface Used To Receive Image Data: `provisioninginterface`
For nodes with multiple interfaces, one interface may be faster than the others. If so, it can be convenient to receive the image data via the fastest interface. Setting the value of `provisioninginterface`, which is a property of the node configuration, allows this. By default it is set to `BOOTIF`.

Transport Protocol Used For Image Data: `provisioningtransport`
The provisioning system can send the image data encrypted or unencrypted. The `provisioningtransport` property of the node configuration can have these values:

- `rsyncdaemon`, which sends the data unencrypted
- `rsyncssh`, which sends the data encrypted

Because encryption severely increases the load on the provisioning node, using `rsyncssh` is only suggested if the users on the network cannot be trusted. By default, `provisioningtransport` is set to `rsyncdaemon`.

Tracking The Status Of Image Data Provisioning: `provisioningstatus`
The `provisioningstatus` command within the `softwareimage` mode of `cmsh` displays an updated state of the provisioning system. As a one-liner, it can be run as:

```
bright52:~ # cmsh -c "softwareimage provisioningstatus"
Provisioning subsystem status:          idle, accepting requests
Update of provisioning nodes requested: no
Maximum number of nodes provisioning:   10000
Nodes currently provisioning:           0
Nodes waiting to be provisioned:        <none>
Provisioning node bright52:
  Max number of provisioning nodes:     10
  Nodes provisioning:                   0
  Nodes currently being provisioned:    <none>
```

The `cmgui` equivalent is accessed from the "`Provisioning Status`" tabbed pane in the "`Software Images`" resource (figure 6.3).

Tracking The Provisioning Log Changes: `synclog`
For a closer look into the image file changes carried out during provisioning requests, the `synclog` command from `device` mode can be used (lines elided in the following output):

Example

```
[bright52->device]% synclog node001
Tue, 11 Jan 2011 13:27:17 CET - Starting rsync daemon based provisionin\
g. Mode is SYNC.

sending incremental file list
./
...
deleting var/lib/ntp/etc/localtime
var/lib/ntp/var/run/ntp/
```

```
...
sent 2258383 bytes  received 6989 bytes  156232.55 bytes/sec
total size is 1797091769  speedup is 793.29

Tue, 11 Jan 2011 13:27:31 CET - Rsync completed.
```

In cmgui, the equivalent output to cmsh's synclog is displayed by selecting a specific device or a specific category from the resource tree. Then, within the tasks tabbed pane that opens up, the "Provisioning Log" button at the bottom right is clicked (figure 6.20):

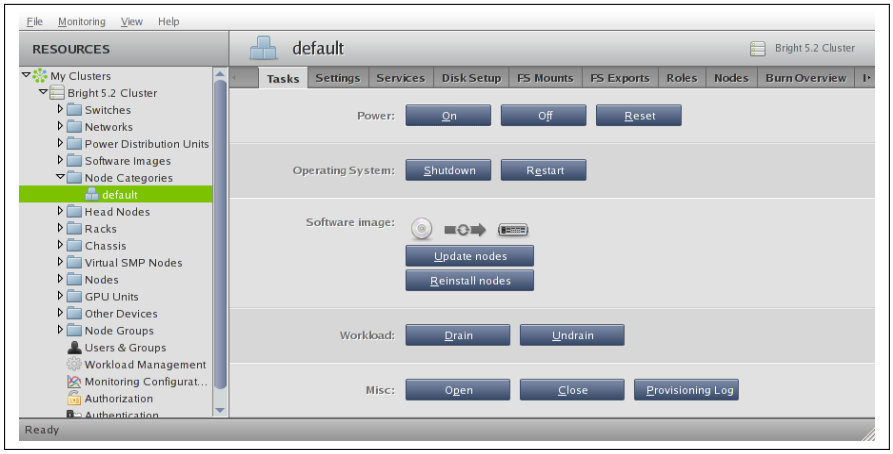

Figure 6.20: cmgui: Provisioning Log Button For A Device Resource

6.4.8 Writing Network Configuration Files

In the previous section, the local drive of the node is synchronized according to install-mode settings with the software image from the provisioning node. The node-installer now sets up configuration files for each configured network interface. These are files like:

`/etc/sysconfig/network-scripts/ifcfg-eth0`

for Red Hat, Scientific Linux, and CentOS, while SUSE would use:

`/etc/sysconfig/network/ifcfg-eth0`

These files are placed on the local drive.

When the node-installer finishes its remaining tasks (sections 6.4.9–6.4.13) it brings down all network devices and hands over control to the local `/sbin/init` process. Eventually a local `init` script uses the network configuration files to bring the interfaces back up.

6.4.9 Creating A Local `/etc/fstab` File

The `/etc/fstab` file on the local drive contains local partitions on which filesystems are mounted as the `init` process runs. The actual drive layout is configured in the category configuration or the node configuration, so the node-installer is able to generate and place a valid local `/etc/fstab` file. In addition to all the mount points defined in the drive layout, several extra mount points can be added. These extra mount points, such as NFS imports, /proc, /sys and /dev/shm, can be defined and managed in the node's category and in the specific configuration of the node configuration, using cmgui or cmsh (section 4.7.2).

6.4.10 Installing GRUB Bootloader

By default, a node-installer boots from the software image on the head node via the network.

Optionally, the node-installer installs a boot record on the local drive if the `installbootrecord` property of the node configuration or node category is set to `on`, so that the next boot can be from the local drive.

For a hard drive boot to work:

1. hard drive booting must be set to have a higher priority than network booting in the BIOS of the node. Otherwise regular PXE booting occurs, despite whatever value `installbootrecord` has.

2. the GRUB bootloader with a boot record must be installed in the MBR of the local drive, overwriting the default gPXE boot record.

   To set the GRUB bootloader in `cmgui`, the "Install boot record" checkbox must be ticked and saved in the node configuration or in the node category.

   The `cmsh` equivalents are commands like:

   ```
   cmsh -c "device use node001; set installbootrecord yes; commit"
   ```

   or

   ```
   cmsh -c "category use default; set installbootrecord yes; commit"
   ```

Arranging for the two items in the preceding list ensures that the next boot is from GRUB on the hard drive.

Simply unsetting "Install boot record" and rebooting the node does not restore its gPXE boot record and hence its ability to gPXE boot. To restore the gPXE boot record, the node can be booted from the default image copy on the head node via a network boot again. Typically this is done by manual intervention during node boot to select network booting from the BIOS of the node.

As suggested by the Bright Cluster Manager gPXE boot prompt, setting network booting to work from the BIOS (regular "PXE" booting) is preferred to gPXE booting from the disk.

6.4.11 Running Finalize Scripts

A *finalize script* is similar to an `initialize` script (section 6.4.5), only it runs a few stages later in the node-provisioning process.

It is used when custom commands need to be executed after the preceding mounting, provisioning, and housekeeping steps, but before handing over control to the node's local `init` process. For example, custom commands may be needed to:

- initialize some unsupported hardware before `init` takes over
- supply a configuration file for the software image that cannot simply be added to the software image and used by `init` because it needs node-specific settings
- load a slightly altered standard software image on particular nodes, typically with the change depending on automatically detecting the

hardware of the node it is being loaded onto. While this could also be done by creating a full new software image and loading it on to the nodes according to the hardware, it usually turns out to be better for simplicity's sake (future maintainability) to minimize the number of software images for the cluster

The custom commands used to implement such changes are then added to the `finalize` script.

A `finalize` script can be added to both a node's category and the node configuration. The node-installer first runs a `finalize` script, if it exists, from the node's category, and then a `finalize` script, if it exists, from the node's configuration.

The node-installer sets several environment variables which can be used by the `finalize` script. Appendix E contains an example script which documents these variables.

Similar to the `finalize` script are:

- The `initialize` script (section 6.4.5). This may run several stages before the `finalize` script.
- The `imageupdate_initialize` and `imageupdate_finalize` scripts, which may run when the `imageupdate` command runs (section 6.6.2).

6.4.12 Unloading Specific Drivers

Many kernel drivers are only required during the installation of the node. After installation they are not needed and can degrade node performance.

The IPMI drivers are an egregious example of this. The IPMI drivers are required to have the node-installer configure the IP address of any IPMI cards. Once the node is configured, these drivers are no longer needed, but they continue to consume significant CPU cycles and power if they stay loaded.

To solve this, the node-installer can be configured to unload a specified set of drivers just before it hands over control to the local init process. This is done by editing the `removeModulesBeforeInit` setting in the node-installer configuration file `/cm/node-installer/scripts/node-installer.conf`. By default, the IPMI drivers are placed in the `removeModulesBeforeInit` setting.

6.4.13 Switching To The Local `init` Process

At this point the node-installer is done. The node's local drive now contains a complete Linux installation and is ready to be started. The node-installer hands over control to the local `/sbin/init` process, which continues the boot process and starts all runlevel services. From here on the boot process continues as if the machine was started from the drive just like any other regular Linux machine.

6.5 Node States

During the boot process, several state change messages are sent to the head node CMDaemon or detected by polling from the head node CMDaemon. The most important node states for a cluster after boot up are introduced in section 3.1.1. These states are described again, along with some less common ones to give a more complete picture of node states.

6.5.1 Node States Indicating Regular Start Up

During a successful boot process the node goes through the following states:

- `INSTALLING`. This state is normally entered as soon as the node-installer has determined on which node the node-installer is running. Within this state, information messages display indicating what is being done while the node is in the `INSTALLING` state. Possible messages in the status displays for the node within `cmgui` and `cmsh` are normally, in sequence:

  1. `node-installer started`
  2. `checking disks`
  3. `recreating partitions and file systems`
  4. `mounting disks`
  5. One of these following two messages:
     (a) `waiting for FULL provisioning to start`
     (b) `waiting for SYNC provisioning to start`
  6. `provisioning started, waiting for completion`
  7. `provisioning complete`

  Between steps 1 and 2 in the preceding, these optional messages can also show up:

  - If `burn` mode is entered or left:

    ```
    running burn-in tests
    burn-in test completed successfully
    ```

  - If `maintenance` mode is entered:

    ```
    entered maintenance mode
    ```

- `INSTALLER_CALLINGINIT`. This state is entered as soon as the node-installer has handed over control to the local `init` process. The associated message normally seen with it in `cmsh` or `cmgui` is:

  - `switching to local root`

- `UP`. This state is entered as soon as the CMDaemon of the node connects to the head node CMDaemon.

6.5.2 Node States That May Indicate Problems

Other node states are often associated with problems in the boot process:

- `DOWN`. This state is registered as soon as the CMDaemon of the node is no longer detected by CMDaemon on the head node. The additional text `pingable` is used alongside the state messages to indicate whether a node responds to `cmdaemon`'s ICMP pings, which is useful for troubleshooting node and network behavior.

- `CLOSED`. This state is set for the node by the administrator.

  It can be set from the `device` mode of `cmsh` using the `close` command. The `help` text for the command gives details on how it can be applied to categories, groups and so on. The "`-m`" option sets a message for the closed node or nodes.

Example

```
root@b52 ~]# cmsh
[b52]% device
[b52->device]% close -m "fan dead" -n node001,node009,node020
Mon May  2 16:32:01 2011 [notice] b52: node001 ....[ CLOSED ] (fan dead)
Mon May  2 16:32:01 2011 [notice] b52: node009 ....[ CLOSED ] (fan dead)
Mon May  2 16:32:01 2011 [notice] b52: node020 ....[ CLOSED ] (fan dead)
```

The state can also be set from `cmgui`. This is done via the `Tasks` tab of the node item in the `Nodes` resource, or from the category item in the `Node Categories` resource (figure 6.21).

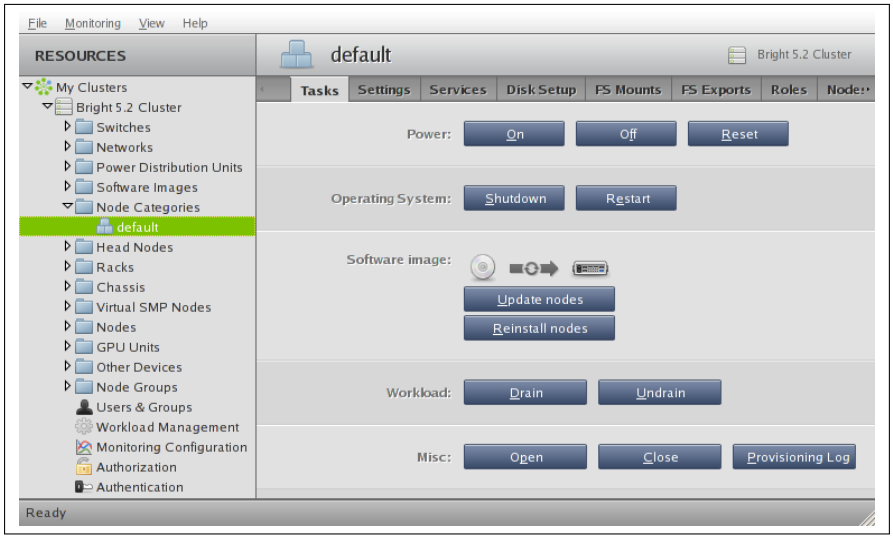

Figure 6.21: Acting On A CLOSED State From `cmgui`

When the `CLOSED` state is set, CMDaemon does not register pings to and from the node that tell it the node state, nor does it monitor the node.

Only a few CMDaemon commands attempt to act on a closed node. For example, in the `device` mode of `cmsh`:

- `open`
- `drain` and `undrain`
- For nodes that have power control[2]:
  - `power -f on`
  - `power -f off`
  - `power -f reset`

The `cmgui` equivalents in figure 6.21 are:

- The `Open` button from the `Watch` row
- The `Drain` and `Undrain` buttons from the `Workload` row
- The `On`, `Off`, and `Reset` buttons from the PDU `Power` row

[2]power control mechanisms such as IPMI, PDUs, HP iLO and custom power scripts are described in Chapter 5

Since the CLOSED state is not monitored by the cluster management system, powering down a CLOSED node, for example, does not change its monitored state to DOWN. Nor does a reset on a CLOSED node eventually bring its state to UP. The only way out of a CLOSED state is for the administrator to tell the node to open via the aforementioned cmsh or cmgui "open" options. Whether the node listens or not does not matter, the head node records it as being in an OPENING state for a short time, during which period the next state (UP, DOWN, etc) is agreed upon by the head node and the node.

The CLOSED state is sometimes set to take a node that is unhealthy out of the cluster management system. The node can then still be up, and even continue running workload jobs, since workload managers run independent of CMDaemon. So, if the workload manager is still running, the jobs themselves are still handled by the workload manager, even if CMDaemon is no longer aware of their status until the node is re-opened. For this reason, draining a node is often done before closing a node, although it is not obligatory.

- OPENING. This transitional state is entered as soon as the CMDaemon of the node rescinds the CLOSED state with an "open" command from cmsh or cmgui. The state usually lasts no more than about 5 seconds, and never more than 30 seconds in the default configuration settings of Bright Cluster Manager. The help text for the open command of cmsh gives details on its options.

- INSTALLER_FAILED. This state is entered from the INSTALLING state when the node-installer has detected an unrecoverable problem during the boot process. For instance, it cannot find the local drive, or a network interface cannot be started. This state can also be entered from the INSTALLER_CALLINGINIT state when the node takes too long to enter the UP state. This could indicate that handing over control to the local init process failed, or the local init process was not able to start the CMDaemon on the node. Lastly, this state can be entered when the previous state was INSTALLER_REBOOTING and the reboot takes too long.

- INSTALLER_UNREACHABLE. This state is entered from the INSTALLING state when the head node CMDaemon can no longer ping the node. It could indicate the node has crashed while running the node-installer.

- INSTALLER_REBOOTING. In some cases the node-installer has to reboot the node to load the correct kernel. Before rebooting it sets this state. If the subsequent reboot takes too long, the head node CMDaemon sets the state to INSTALLER_FAILED.

6.6 Updating Running Nodes

Updating running nodes is the process of provisioning a software image to a node without rebooting the node. A configuration setting called excludelistupdate (section 6.6.1), is used by the imageupdate command (section 6.6.2) when carrying out an update to a running node.

The converse of the imageupdate command is the grabimage command, which grabs a node state and creates a software image from it. The grabimage command is discussed in section 9.5.2.

6.6.1 Updating Running Nodes: Configuration With excludelistupdate

Changes made to the contents of the head node's software image for nodes become part of the provisioning system according to its housekeeping system (section 6.2.4). The image is then installed from the provisioning system onto a regular node when it (the regular node) reboots via a provisioning request (section 6.4.7).

However, updating a running node with the latest changes from the software image is also possible without rebooting it to re-install the image. Such an update can be requested using cmsh or cmgui, and is queued and delegated to a provisioning node just like an ordinary provisioning request.

Like the provisioning requests done at the time of install it uses an exclude list, with the same structure and rsync patterns syntax to those detailed in section 6.4.7. This exclude list is however defined in the excludelistupdate property of the node's category. To distinguish the intention behind the exclude lists, the administrator should note that it is the excludelist*update* that is being discussed here, in contrast with the excludelistsync*install*/excludelistfull*install* from section 6.4.7.

So, the excludelistupdate property settings here concern an *update* to a running system, while the other two from section 6.4.7 are about an *install* during node start-up.

Similar to the sync case of section 6.4.7, which uses excludelistsync, the update case in this section uses excludelistupdate.

The running node update type of synchronization uses the excludelistupdate property to specify what files and directories to exclude from consideration when copying parts of the filesystem from a known good software image to the node. The excludelistupdateinstall property is in the form of a list of exclusions, or more accurately in the form of two sub-lists.

The contents of the sub-lists specify the parts of the filesystem that should be retained on the node during update synchronization. The intention behind this is to have the node synchronize quickly, updating only the files from the image to the node that need updating due to the changes on the software image, and otherwise keeping files that are already on the node hard disk unchanged. The contents of the sub-lists are thus the files and directories that are expected to be present in a running node.

Anything already on the node that matches the content of these sub-lists is not overwritten by image content during an excludelistupdate update. However, image content that is not on the node is copied over to the node only for items matching the first sub-list. The remaining files and directories on the node, that is, the ones that are not in the sub-lists, lose their original contents, and are copied over from the software image.

A sample cmsh one-liner which opens up a text editor in a category to set the exclude list for updates for is:

```
cmsh -c "category use default; set excludelistupdate; commit"
```

The exclude list for updates can be edited in cmgui as indicated in figure 6.22.

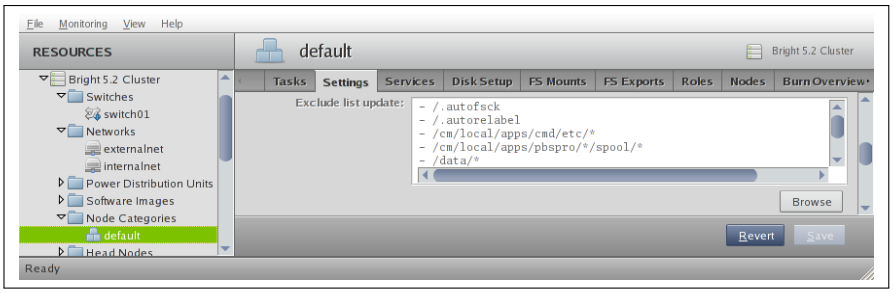

Figure 6.22: Setting up exclude lists with cmgui for node updates

In addition to the paths excluded using the excludelistupdate property, the provisioning system automatically adds any NFS, Lustre, FUSE, PanFS, FhGFS, GlusterFS, and GPFS imported file systems on the node. If this were not done, all data on these filesystems would be wiped since they are not part of the software image.

6.6.2 Updating Running Nodes

Updating Running Nodes: With cmsh Using imageupdate

Using a defined excludelistupdate property (section 6.6.1), the imageupdate command of cmsh is used to start an update on a running node:

Example

```
[bright52->device]% imageupdate -n node001
Performing dry run (use synclog command to review result, then pass -w \
to perform real update)...
Tue Jan 11 12:13:33 2011 bright52: Provisioning started on node node001
[bright52->device]% imageupdate -n node001: image update in progress ...
[bright52->device]%
Tue Jan 11 12:13:44 2011 bright52: Provisioning completed on node node0\
01
```

By default the imageupdate command performs a dry run, which means no data on the node is actually written. Before passing the "-w" switch, it is recommended to analyze the rsync output using the synclog command (section 6.4.7).

If the user is now satisfied with the changes that are to be made, the imageupdate command is invoked again with the "-w" switch to implement them:

Example

```
[bright52->device]% imageupdate -n node001 -w
Provisioning started on node node001
node001: image update in progress ...
[bright52->device]% Provisioning completed on node node001
```

Updating Running Nodes: With `cmgui` Using The "`Update node`" Button
In `cmgui` an image update can be carried out by selecting the specific node or specific category from the resource tree. Then, within the tasks tabbed pane that opens up, the "`Update node`" button is clicked (figure 6.23). This opens up a dialog which has a dry-run checkbox marked by default.

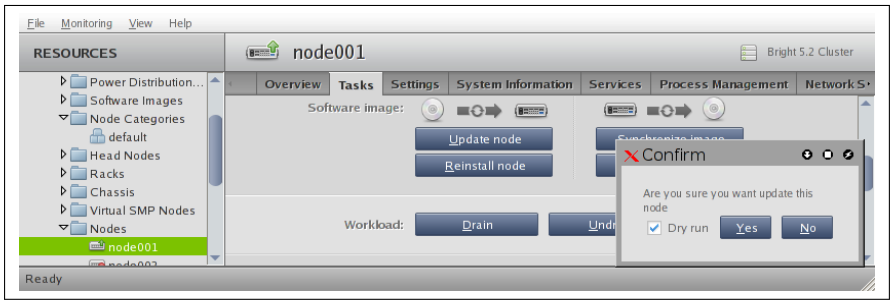

Figure 6.23: Updating A Running Node With `cmgui`

The dry-run can be reviewed by clicking on the "`Provisioning Log`" button further down the same tabbed pane. The update can then be done again with the dry-run check mark off to actually implement the update.

Updating Running Nodes: Considerations
Updating an image via `cmsh` or `cmgui` automatically updates the provisioners first via the `updateprovisioners` command (section 6.2.4) if the provisioners have not been updated in the last 5 minutes. So, if there has been an update within the last 5 minutes, then provisioners do not get an updated image when doing the updates. Running the `updateprovisioners` command just before running the `imageupdate` command therefore usually makes sense.

Also, when updating services, the services on the nodes may not restart since the `init` process may not notice the replacement.

For these reasons, especially for more extensive changes, it can be safer for the administrator to simply reboot the nodes instead of using `imageupdate` to provision the images to the nodes. A reboot ensures that a node will have the latest image and that services should start up as intended.

Updating Running Nodes: Pre- And Post-update Scripts
Two further scripts associated with the `imageupdate` command and its `cmgui` equivalent may run as part the execution of the update. These are located at `/cm/images/default-image/cm/local/apps/cmd/scripts/` in the default software image:

- The *imageupdate_initialize script* runs before the software image starts updating. If the `imageupdate_initialize` script exits with non-zero, then the image does not update
- The *imageupdate_finalize script* runs after an `imageupdate` command is run on that node, and right after the software image has updated.

These differ from the `initialize` (section 6.4.5) and `finalize` (section 6.4.11) scripts because they run on nodes that are fully up rather than

on nodes that are booting, so they are able to access a fully running system, and because they run on all nodes using that image rather than also being configurable for individual nodes within that image.

6.7 Adding New Nodes

6.7.1 Adding New Nodes With cmsh And cmgui Add Functions

Node objects can be added from within the device mode of cmsh by running the add command:

Example

```
[bright52->device]% add physicalnode node002
[bright52->device*[node002*]% commit
```

The cmgui equivalent of this is to go within the Nodes resource, and after the Overview tabbed pane for the Nodes resource comes up, to click on the Add button (figure 6.24)

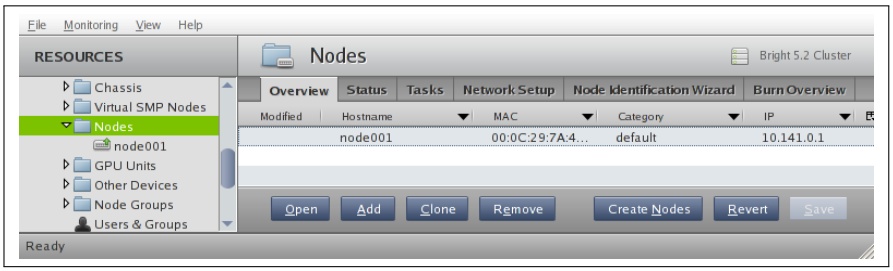

Figure 6.24: Buttons To Add, Remove And Set Up Nodes

When adding the node objects in cmsh and cmgui, some values (IP addresses for example) may need to be filled in before the object validates.

Adding new node objects as "placeholders" can also be done from cmsh or cmgui. By placeholders, here it is meant that an incomplete node object is set. For example, sometimes it is useful to create a node object with the MAC address setting unfilled because it is still unknown. Why this can be useful is covered shortly.

6.7.2 Adding New Nodes With The Node Creation Wizard

Besides adding nodes using the add command of cmsh or the Add button of cmgui as in the previous section, there is also a cmgui wizard that guides the administrator through the process—the *node creation wizard*. This is useful when adding many nodes at a time. It is available from the Nodes resource, by selecting the Overview tabbed pane and then the "Create Nodes" button (figure 6.24).

This wizard should not be confused with the closely related node *identification* wizard described earlier in section 6.4.2, which identifies unassigned MAC addresses and switch ports, and helps assign them node names.

The node *creation* wizard instead creates an object for nodes, assigns them node names, but it leaves the MAC address field for these nodes unfilled, keeping the node object as a "placeholder"(figure 6.25).

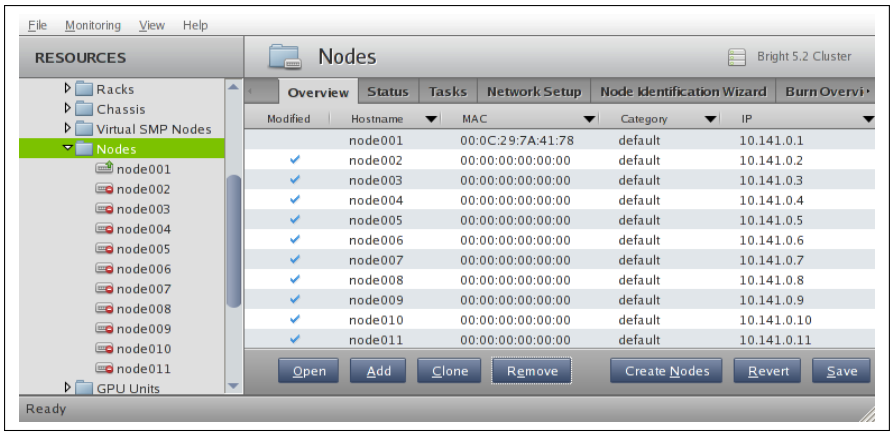

Figure 6.25: Node Creation Wizard: 10 Placeholders Created

The MAC addresses can be assigned to a node via the node identification wizard. However, leaving nodes in a "placeholder" state, where the MAC address entry is left unfilled, means that any new node with an unassigned MAC address that is started up is offered a choice out of the created node names by the provisioning system at its console. This happens when the node-installer reaches the node configuration stage during node boot as described in section 6.4.2. This is sometimes preferable to associating the node name with a MAC address remotely.

The node creation wizard can set IP addresses for the nodes. At one point in the dialog a value for `IP-offset` can also be set (figure 6.26).

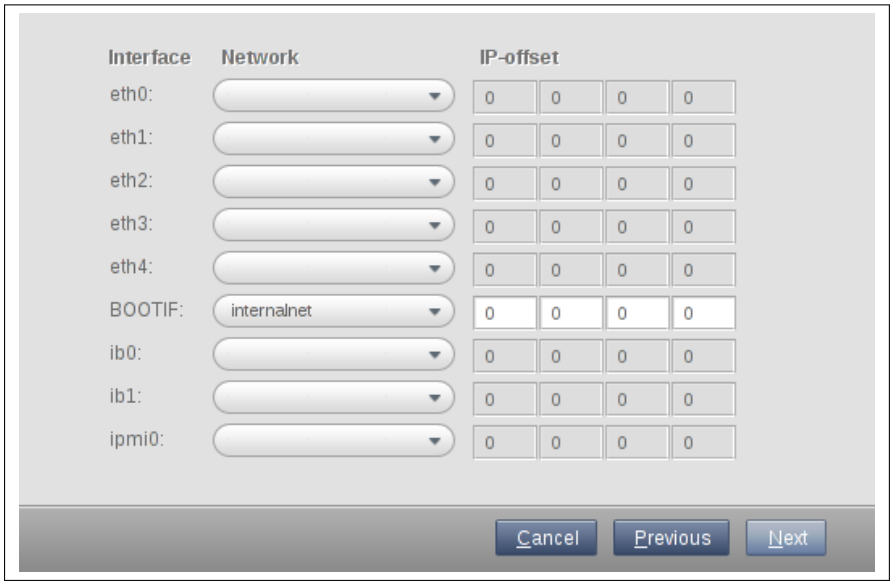

Figure 6.26: Node Creation Wizard: Setting Interfaces

The default setting for `IP-offset` is `0.0.0.0`, and means the default IP address is suggested for assignment to each node in the range. The default IP address is based on the node name, with node001 having the value 10.141.0.1, and so on. An offset of x implies that the xth IP address after the default is suggested for assignment to each node in the range. Some care must be taken when setting IP addresses using the wizard,

since no duplicate IP address checking is done.

Example

A node001 has its default IP address 10.141.0.1. The node005 is then added.

- If `IP-offset`=0.0.0.0, then 10.141.0.5 is suggested for assignment to node005, because, by default, the node name is parsed and its default IP address suggested.
- If `IP-offset`=0.0.0.2, then 10.141.0.7 is suggested for assignment to node005, because it is 2 IP addresses after the default.

The `cmsh` equivalent of the node creation wizard is the `foreach` loop with the `-clone` option acting on a node (section 3.5.5).

6.8 Troubleshooting The Node Boot Process

During the node boot process there are several common issues that can lead to an unsuccessful boot. This section describes some of these issues and their solutions. It also provides general hints on how to analyze boot problems.

6.8.1 Node Fails To PXE Boot

Possible reasons to consider if a node is not even starting to PXE boot in the first place:

- The boot sequence may be set wrongly in the BIOS. The boot interface should normally be set to be the first boot item in the BIOS.
- There may a a bad cable connection. This can be due to moving the machine, or heat creep, or another physical connection problem. Firmly inserting the cable into its slot may help. Replacing the cable or interface as appropriate may be required.
- The cable may be connected to the wrong interface. By default, eth0 is assigned the internal network interface, and eth1 the external network interface. However:
  - The two interfaces can be confused when physically viewing them and a connection to the wrong interface can therefore be made.
  - It is also possible that the administrator has changed the default assignment.

  The connections should be checked to eliminate these possibilities.
- DHCP may not be running. A check should be done to confirm that DHCP is running on the internal network interface (usually eth0):

```
[root@testbox ~]# ps aux | grep dhcp
root 4368 0.0 0.0 27680 3484 ? Ss Apr07 0:01 /usr/sbin/dhcpd eth0
```

- A rogue DHCP server may be running. If there are all sorts of other machines on the network the nodes are on, then it is possible that there is a rogue DHCP server active on it, and interfering with PXE booting. Stray machines should be eliminated.
- Sometimes a manufacturer releases hardware with buggy drivers that have a variety of problems. For instance: Ethernet frames may be detected at the interface (for example, by `ethtool`), but TCP/IP packets may not be detected (for example, by `wireshark`). In that case, the manufacturer should be contacted to upgrade their driver.
- The interface may have a hardware failure. In that case, the interface should be replaced.

6.8.2 Node-installer Logging

If the node manages to get beyond the PXE stage to the node-installer stage, then the first place to look for hints on node boot failure is usually the node-installer log file. The node-installer sends logging output to a local `syslog` daemon. This forwards all log data to the IP address from which the node received its DHCP lease, which is typically the IP address of the head node or failover node. In a default Bright Cluster Manager setup, the `local5` facility is used and all messages are then logged to `/var/log/node-installer` of the head node.

Optionally, extra log information can be written by enabling debug logging, which sets the syslog importance level at `LOG_DEBUG`. To enable debug logging, the debug field is changed in `/cm/node-installer/scripts/node-installer.conf`.

From the console of the booting node the log file is also accessible by pressing `Alt+F7` on the keyboard.

A booting node console can be accessed remotely if Serial Over LAN (SOL) is enabled.

6.8.3 Serial Over LAN Console Access

SOL Considerations For Node Console Access

The serial port of a node is often used to send and receive data. This requires that the serial port is enabled. This is usually configured in the node BIOS.

If the serial port of a node is configured in the node kernel to redirect a console, then it allows serial console access. That is, the console can be viewed using a terminal software such as minicom (in Linux) or Hyperterminal (in Windows) on another machine to communicate with the node via the serial port.

Serial Over LAN (SOL) is a feature of IPMI 2.0, and iLO. It is enabled by configuring the baseboard management controller BIOS. When enabled, it redirects data that is going to the serial port to the LAN, so that SOL clients can process it.

SOL thus allows SOL clients on the LAN to access the Linux serial console if

- the serial port is enabled in the node BIOS
- the serial console is enabled in the node kernel
- SOL is enabled in the Baseboard Management Controller BIOS

A feature of SOL console clients is that the administrator is not presented with any text prompt from the node that is being accessed. This is useful in some cases, and can be a problem in others.

An example of the issue is the case where the administrator has already logged into the console and typed in a command in the console shell, but has no intention of pressing the <*ENTER*> key until some other tasks are first carried out. If the connection breaks at this point, then the command typed in is held in the console shell command buffer, but is not displayed when a serial connection is re-established to the console—the previously entered text is invisible to the client making the connection. A subsequent <*ENTER*> would then attempt to execute the command. This is why an <*ENTER*> is not sent as the last key sequence during automated SOL access, and it is left to the administrator to enter the appropriate key strokes.

To avoid commands in the console shell buffer inadvertantly being run when taking over the console, the administrator can start the session with a <*CTRL*>`-u` to clear out text in the shell before pressing <*ENTER*>.

SOL Console Configuration And Access With `cmgui`

In `cmgui`, within the "`Software Images`" resource, if the "`Console over SOL`" checkbox (figure 6.27) is checked for the software image that the node is using, then the kernel option to make the Linux serial console accessible is used, on rebooting the node that is to be accessed.

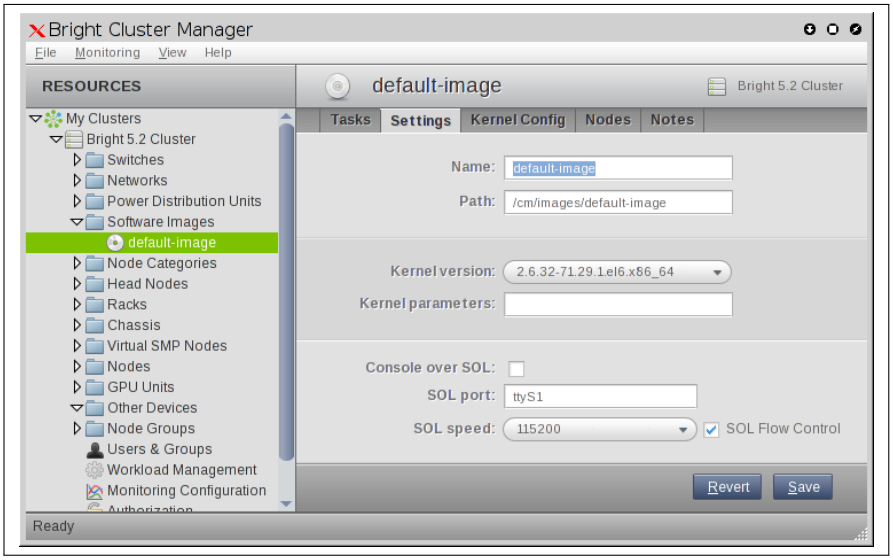

Figure 6.27: Configuring The SOL Console For A Node With `cmgui`

This means that if the serial port and SOL are enabled for the node hardware, then after the node reboots the Linux serial console is accessible over the LAN via an SOL client. Some of the settings for SOL can be set from the same screen.

With SOL correctly configured, an SOL client to access the Linux serial console can be launched using the "`Remote Console`" button for the node (figure 6.28).

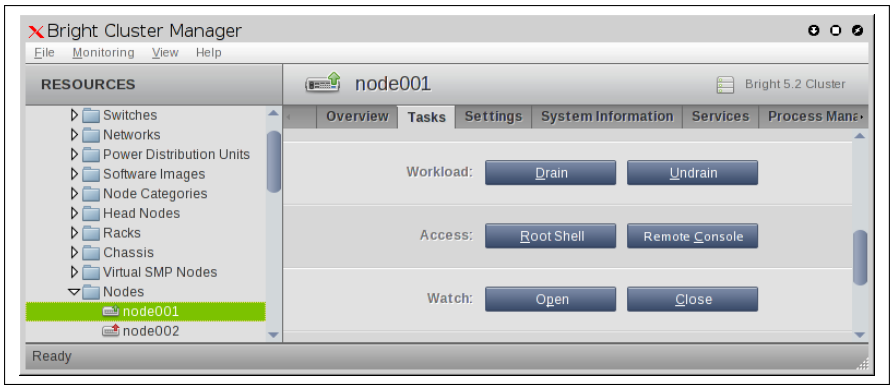

Figure 6.28: Accessing The SOL Console For A Node With `cmgui`

By default the expected SOL client is configured for `IPMI`. To configure the "`Remote Console`" button for `iLO` instead, the variable `type` in the script at `/cm/local/apps/cmd/scripts/ipmisol` must be set to `ilo` instead of `ipmi`.

SOL Console Configuration And Access With `cmsh`

In `cmsh`, the serial console kernel option for a software image can be enabled within the `softwareimage` mode of `cmsh`. For the default image of `default-image`, this can be done as follows:

Example

```
[root@bright52 ~]# cmsh
[bright52]% softwareimage use default-image

[bright52->softwareimage[default-image]]% set enablesol yes
[bright52->softwareimage*[default-image*]]% commit
```

The SOL settings for a particular image can be seen with the `show` command:

```
[bright52->softwareimage[default-image]]% show | grep SOL
Parameter                      Value
------------------------------ --------------
Enable SOL                     yes
SOL Flow Control               yes
SOL Port                       ttyS1
SOL Speed                      115200
```

Values can be adjusted if needed with the `set` command.

To access a node that has an IPMI Baseboard Management Controller via an SOL client, the node can be specified from within the `device` mode of `cmsh`, and the `ipmisol` command run on the head node:

```
[root@bright52 ~]# cmsh
[bright52]% device use node001
[bright52->device[node001]]% ipmisol
```

A node that has an iLO Baseboard Management Controller instead can provide remote Linux serial console access via iLO. The node can be accessed from the head node, using the command line or `cmsh` as follows:

1. From the command line, an ssh login to the Baseboard Management Controller IP address can be done, and the iLO command remcons is then run to get to the serial console of the remote node.

2. From cmsh, the remote node can be accessed from iLO using an ilosol command, just like the remote node is accessed from IPMI using the ipmisol command previously. This can be done with the following one-time file changes:

   - the alias ilosol is set to ipmisol in the cmsh default settings file at /root/.cm/cmsh/.cmshrc by adding the line:

     ```
     alias ilosol ipmisol
     ```

   - the variable type in the script at /cm/local/apps/cmd/scripts/ipmisol is set to ilo instead of ipmi

The Network, Username, And Password For The Baseboard Management Controller

The default network base address on which the IP address of the Baseboard Management Controller is to be found is 10.148.0.0. The username and password required to access the Baseboard Management Controller can be obtained from partition mode, by getting the values of ipmiusername and ipmipassword from the base object.

Example

```
[root@bright52 ~]# cmsh -c "partition use base;\
 get ipmiusername; get ipmipassword"
ADMIN
araoos98Y6gotdfadlipoiq9
[root@bright52 ~]# ssh -l ADMIN 10.148.0.1
ADMIN@10.148.0.1's password:

User:ADMIN logged-in to demoilo.internal.com(10.148.0.1)
iLO  1.94 at 12:11:27 Mar 19 2009
Server Name: host is unnamed
Server Power: On

</>hpiLO-> remcons

Starting remote console
Press 'ESC (' to return to the CLI Session

Red Hat Enterprise Linux Server release 6.2 (Santiago)
Kernel 2.6.32-220.2.1.el6.x86_64 on an x86_64

node001 login:
```

6.8.4 Provisioning Logging

The provisioning system sends log information to the CMDaemon log file. By default this is in /var/log/cmdaemon.

The image synchronization log file can be retrieved with the synclog command running from device mode in cmsh (section 6.4.7). Hints on provisioning problems are often found by looking at the tail end of the log.

6.8.5 Ramdisk Cannot Start Network

The ramdisk must activate the node's network interface in order to fetch the node-installer. To activate the network device, the correct kernel module needs to be loaded. If this does not happen, booting fails, and the console of the node displays something similar to figure 6.29.

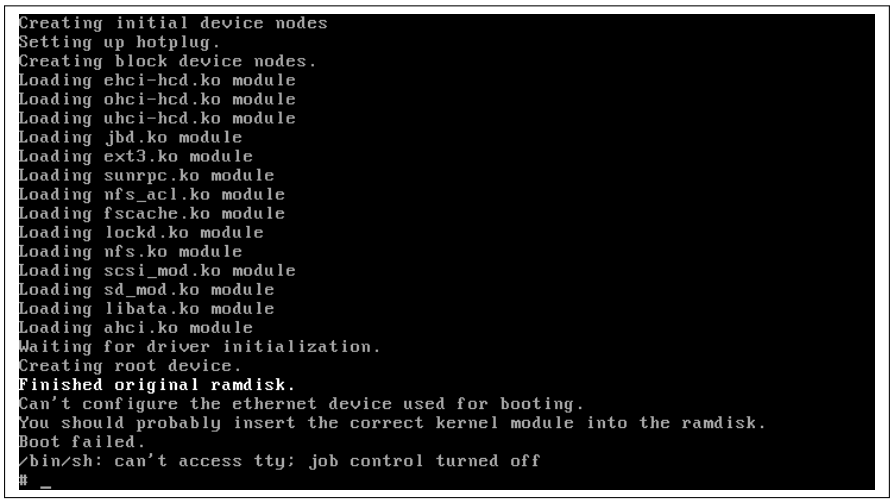

Figure 6.29: No Network Interface

To solve this issue the correct kernel module should be added to the software image's kernel module configuration (section 6.3.2). For example, to add the e1000 module to the default image using `cmsh`:

Example

```
[mc]% softwareimage use default-image
[mc->softwareimage[default-image]]% kernelmodules
[mc->softwareimage[default-image]->kernelmodules]% add e1000
[mc->softwareimage[default-image]->kernelmodules[e1000]]% commit
Initial ramdisk for image default-image was regenerated successfully
[mc->softwareimage[default-image]->kernelmodules[e1000]]%
```

After committing the change it typically takes about a minute before ramdisk creation is completed.

6.8.6 Node-Installer Cannot Create Disk Layout

When the node-installer is not able to create a drive layout it displays a message similar to figure 6.30. The node-installer log file (section 6.8.2) contains something like:

```
Mar 24 13:55:31 10.141.0.1 node-installer: Installmode is: AUTO
Mar 24 13:55:31 10.141.0.1 node-installer: Fetching disks setup.
Mar 24 13:55:31 10.141.0.1 node-installer: Checking partitions and
filesystems.
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/sda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/hda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Can not find device(s) (/dev\
/sda /dev/hda).
```

```
Mar 24 13:55:32 10.141.0.1 node-installer: Partitions and/or filesystems
are missing/corrupt. (Exit code 4, signal 0)
Mar 24 13:55:32 10.141.0.1 node-installer: Creating new disk layout.
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/sda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Detecting device '/dev/hda':
not found
Mar 24 13:55:32 10.141.0.1 node-installer: Can not find device(s) (/dev\
/sda /dev/hda).
Mar 24 13:55:32 10.141.0.1 node-installer: Failed to create disk layout.
(Exit code 4, signal 0)
Mar 24 13:55:32 10.141.0.1 node-installer: There was a fatal problem. T\
his node can not be installed until the problem is corrected.
```

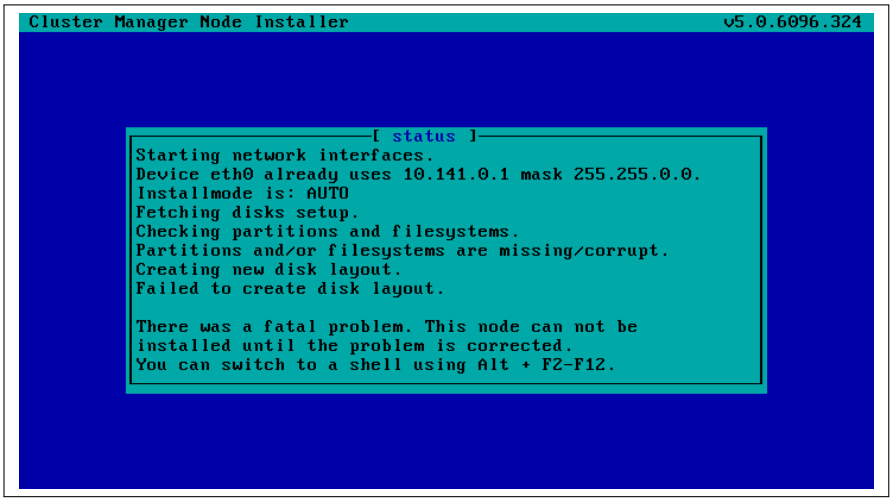

Figure 6.30: No Disk

It is likely that this issue is caused by the correct storage driver not being loaded. To solve this issue the correct kernel module should be added to the software image's kernel module configuration (section 6.3.2).

Experienced system administrators work out what drivers may be missing by checking the results of hardware probes. For example, the output of `lspci` provides a list of hardware detected in the PCI slots, giving the chipset name of the storage controller hardware in this case:

Example

```
[root@bright52 ~]# lspci | grep SCSI
00:10.0 Serial Attached SCSI controller: LSI Logic / Symbios Logic SAS2\
008 PCI-Express Fusion-MPT SAS-2 [Falcon] (rev 03)
```

The next step is to Google with likely search strings based on that output.

The Linux Kernel Driver DataBase (LKDDb) is a hardware database built from kernel sources that lists driver availability for Linux. It is available at `http://cateee.net/lkddb/`. Using the Google search engine's "`site`" operator to restrict results to the `cateee.net` web site only, a likely string to try might be:

Example

```
SAS2008 site:cateee.net
```

The search result indicates that the mpt2sas kernel module needs to be added to the node kernels. A look in the modules directory of the software image shows if it is available:

Example

```
find /cm/images/default-image/lib/modules/ -name "*mpt2sas*"
```

If it is is not available, the driver module must then be obtained. If it is a source file, it will need to be compiled. By default, nodes run on standard distribution kernels, so that only standard procedures need to be followed to compile modules.

If the module is available, it can be added to the default image using cmsh in softwareimage mode:

Example

```
[bright52]% softwareimage use default-image
[bright52->softwareimage[default-image]]% kernelmodules
[bright52->softwareimage[default-image]->kernelmodules]% add mpt2sas
[bright52->softwareimage[default-image]->kernelmodules*[mpt2sas*]]% com\
mit
[bright52->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:54:52 2011 [notice] bright52: Initial ramdisk for image de\
fault-image is being generated
[bright52->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:55:43 2011 [notice] bright52: Initial ramdisk for image de\
fault-image was regenerated successfully.
[bright52->softwareimage[default-image]->kernelmodules[mpt2sas]]%
```

After committing the change it can take some time before ramdisk creation is completed—typically about a minute, as the example shows. On rebooting the node, it should now continue past the disk layout stage.

6.8.7 Node-Installer Cannot Start IPMI Interface

In some cases the node-installer is not able to configure a node's IPMI interface, and displays an error message similar to figure 6.31. Usually the issue can be solved by adding the correct IPMI kernel modules to the software image's kernel module configuration. However, in some cases the node-installer is still not able to configure the IPMI interface. If this is the case the IPMI card probably does not support one of the commands the node-installer uses to set specific settings. To solve this issue, setting up IPMI interfaces can be disabled globally by setting the setupIpmi field in the node-installer configuration file /cm/node-installer/scripts/node-installer.conf to false. Doing this disables configuration of all IPMI interfaces by the node-installer. A custom finalize script (appendix E) can then be used to run the required commands instead.

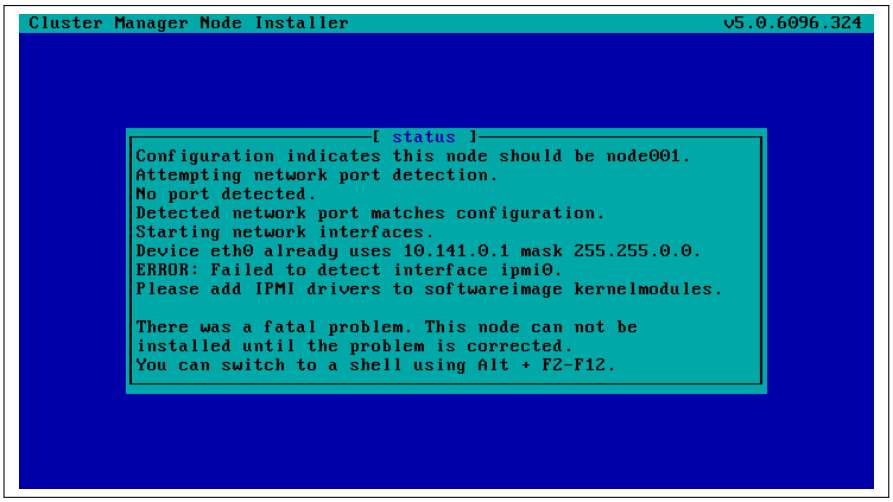

Figure 6.31: No IPMI Interface

7

User Management

Unix users and groups for the cluster are presented to the administrator in a single system paradigm. That is, if the administrator manages them with the Bright Cluster Manager, then the changes are automatically shared across the cluster via the LDAP service.

This chapter describes how to add, remove and edit users and groups using the Bright Cluster Manager.

7.1 Managing Users And Groups With `cmgui`

Selecting "`Users & Groups`" from the `Resources` tree within `cmgui` (figure 7.1) by default lists the LDAP object entries for regular users. These entries are clickable and can be managed further.

There is already one user on a newly installed Bright Cluster Manager: `cmsupport`. This user has no password set by default, and is used to run various diagnostics utilities. The user `cmsupport` should not be removed, nor should the default contents of its home directory be removed.

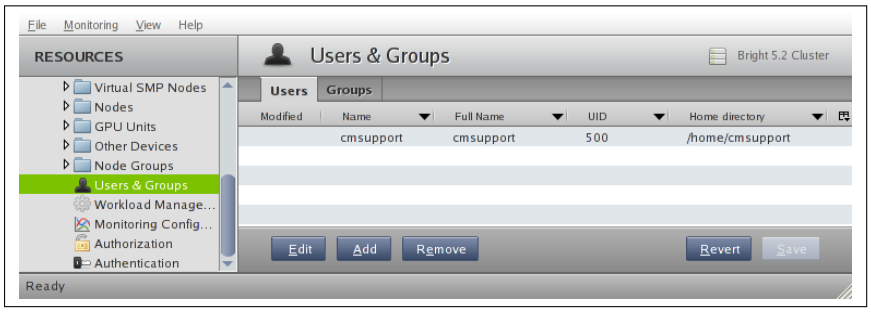

Figure 7.1: `cmgui` User Management

The following five buttons are available to manipulate the entries in the `Users & Groups` resource pane:

1. `Add`: allows users to be added via a dialog. These additions can be committed via the `Save` button.
2. `Save`: saves the as-yet-uncommitted `Add` or `Edit` operations. When saving an addition:
   - User and group ID numbers are automatically assigned from UID and GID 1000 onwards. Normally Red Hat and simi-

lar distributions assign from 500 onwards, while SUSE assigns from 1000 onwards.

- A home directory is created and a login shell is set. Users with unset passwords cannot log in.

3. `Edit`: allows users to be modified via a dialog (figure 7.2).

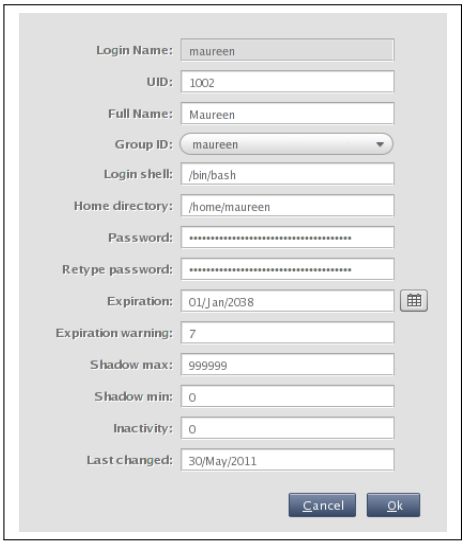

Figure 7.2: `cmgui` User Management: Edit Dialog

The less obvious items in the dialog are explained next:

- `Expiration warning`: The number of days, before the password expires, that the user is warned of the expiry
- `Shadow max`: The maximum number of days the password is valid
- `Shadow min`: The minimum number of days required between password changes. A value of zero means the user may change their password at any time
- `Inactivity`: The number of days of inactivity allowed for the user before the account is blocked. A value of zero means the user is never blocked

4. `Revert`: discards unsaved edits that have been made via the `Edit` button. The reversion goes back to the last save.

5. `Remove`: removes selected rows of users. By default, along with their home directories.

Group management in `cmgui` is started by selecting the `Groups` tab in the `Users & Groups` pane. Clickable LDAP object entries for regular groups then show up, similar to the user entries already covered. Management of these entries is done with the same button functions as for user management.

7.2 Managing Users And Groups With cmsh

This section goes through a session to cover the cmsh functions that correspond to the user management functions of cmgui in the previous section. These functions are run from within cmsh's user mode:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% user
[mycluster->user]%
```

7.2.1 Adding A User

(This corresponds roughly to the functionality of the Add button operation in section 7.1.) In user mode, the process of adding a user maureen to the LDAP directory is started with the add command:

Example

```
[mycluster->user]% add user maureen
[mycluster->user*[maureen*]]%
```

The cmsh helpfully drops into the context of the user just added, and the prompt shows the user name to reflect this. Going into user context would otherwise be done manually by typing use user maureen at the user mode level.

Asterisks in the prompt are a helpful reminder of a modified state, with each asterisk indicating that there is an unsaved, modified property at that asterisk's level.

The modified command displays a list of modified objects, and corresponds roughly to the functionality of the List of Changes menu option under the View menu of the main menu bar.

Running show at this point reveals a user name entry, but empty fields for the other properties of user maureen. So the account in preparation, while it is modified, is clearly not yet ready for use:

Example

```
[mycluster->user*[maureen*]]% show
Parameter                      Value
------------------------------ ----------------------------------------
Common name
Expiration date                2038/1/1
Group ID
Home directory
Inactive                       0
Last change                    1970/1/1
Login shell
Password                       < not set >
Revision
Shadow max                     999999
Shadow min                     0
Shadow warning                 7
User ID
User name                      maureen
```

7.2.2 Saving The Modified State

This corresponds roughly to the functionality of the Save button operation in section 7.1.

In section 7.2.1 above, user maureen was added. maureen now exists as a proposed modification, but has not yet been committed to the LDAP database.

Running the commit command now at the maureen prompt stores the modified state at the user maureen level:

Example

```
[mycluster->user*[maureen*]]% commit
[mycluster->user[maureen]]% show
Parameter                      Value
------------------------------ ----------------------------------------
Common name                    maureen
Expiration date                2038/1/1
Group ID                       1002
Home directory                 /home/maureen
Inactive                       0
Last change                    2011/5/30
Login shell                    /bin/bash
Password                       *********
Revision
Shadow max                     999999
Shadow min                     0
Shadow warning                 7
User ID                        1002
User name                      maureen
```

If, however, commit were to be run at the user mode level without dropping down to the username level, then instead of just that modified user, all modified users and groups would be committed.

When the commit is done, all the empty fields for the user are automatically filled in with defaults based the underlying Linux distribution used. Also, as a security precaution, if an empty field (that is, a "not set") password entry is committed, then a login into the account is not allowed. So, while the account exists at this stage, it still cannot be logged into until the password is set. Logging in requires first editing a property of user maureen, namely the empty password field. Editing passwords and other properties is covered in section 7.2.3.

The default file and directory permissions for the home directory of the user are defined by the umask settings in /etc/login.defs, as would be expected if the administrator were to use the standard useradd command.

7.2.3 Editing Properties Of Users And Groups

This corresponds roughly to the functionality of the Edit button operation in section 7.1.

In the preceding section 7.2.2, a user account maureen was made, which had as one of its properties an unset password. Account logins with an unset password are refused, and so the password needs to be set if the account is to function.

Editing Users With set And clear

The tool used to set user and group properties is the set command. Typing set and then either using tab to see the possible completions, or following it up with the enter key, suggests several parameters that can be set, one of which is password:

Example

```
[mycluster->user[maureen]]% set
Name:
          set - Set specific user or group property

Usage:
          set <parameter>
          set user <name> <parameter>
          set group <name> <parameter>

Arguments:
          name
               Name of the user or group

Parameters:
          commonname.......... Full user name
          expirationdate...... Indicates the date on which the user logi\
                               n will be disabled
          groupid............. Base group of this user
          homedirectory....... Home directory
          inactive............ Indicates the number of days of inactivit\
                               y allowed for the user
          loginshell.......... Login shell
          password............ Password
          shadowmax........... Indicates the maximum number of days for \
                               which the user password remains valid.
          shadowmin........... Indicates the minimum number of days requ\
                               ired between password changes
          shadowwarning....... The number of days of advance warning giv\
                               en to the user before the user password e\
                               xpires
          userid.............. User id number
          username............ User name

[mycluster->user[maureen]]%
```

Continuing the session from the end of section 7.2.2, the password can be set at the user context prompt like this:

Example

```
[mycluster->user[maureen]]% set password seteca5tr0n0my
[mycluster->user*[maureen*]]% commit
[mycluster->user[maureen]]%
```

At this point, the account maureen is finally ready for use.

The converse of the set command is the clear command, which clears properties:

Example

```
[mycluster->user[maureen]]% clear password; commit
```

Editing Groups With `append` And `removefrom`

While the above commands `set` and `clear` also work with groups, there are two other commands available which suit the special nature of groups. These supplementary commands are `append` and `removefrom`. They are used to add extra users to, and remove extra users from a group.

For example, it may be useful to have a `printer` group so that several users can share access to a printer. For the sake of this example (continuing our session from where it was left off above), `tim` and `fred` are now added to the LDAP directory, along with a `printer group`:

Example

```
[mycluster->user[maureen]]% add user tim; add user fred
[mycluster->user*[fred*]]% add group printer
[mycluster->user*[printer*]]% commit
[mycluster->user*[printer]]%
```

Note the context switch that happened here in the `cmsh` user mode environment: the context of user `maureen` was eventually replaced by the context of group `printer`. As a result, the group `printer` is committed, but the users `tim` and `fred` are not yet committed, which is indicated by the asterisk at the user mode level.

Continuing onwards, to add users to a group the `append` command is used. A list of users `maureen`, `tim` and `fred` can be added to the `printer` group like this:

Example

```
[mycluster->user[printer]]% append groupmembers maureen tim fred; commit
[mycluster->user*[printer]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Group ID                       1003
Group members                  maureen tim fred
Group name                     printer
```

To remove users from a group, the `removefrom` command is used. A list of specific users, for example, `tim` and `fred`, can be removed from a group like this:

```
[mycluster->user*[printer]]% removefrom groupmembers tim fred; commit
[mycluster->user*[printer]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Group ID                       1003
Group members                  maureen
Group name                     printer
```

The `clear` command can also be used to clear members—but also clears all of the extras from the group:

Example

```
[mycluster->user[printer]]% clear groupmembers
[mycluster->user*[printer*]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
```

```
Group ID                       1003
Group members
Group name                     printer
```

The commit command is intentionally left out at this point in the session in order to illustrate how reversion is used in the next section.

7.2.4 Reverting To The Unmodified State

This corresponds roughly to the functionality of the Revert button operation in section 7.1.

This section (7.2.4) continues on from the state of the session at the end of section 7.2.3. There, the state of group printers was changed so that the extra added members were removed. This state (the state with no group members showing) was however not yet committed.

The refresh command reverts an uncommitted object back to the last committed state.

This happens at the level of the object it is using. For example, the object that is being handled here is the properties of the group printer. Running revert at a higher level prompt (say, at user mode level) would revert everything at that level and below. So, in order to affect only the properties of the group printer, the refresh command is used at the group printer level prompt. It then reverts the properties of group printer back to their last committed state (and does not affect other objects):

Example

```
[mycluster->user*[printer*]]% refresh
[mycluster->user*[printer]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Group ID                       1003
Group members                  maureen
Group name                     printer
```

Here, the user maureen reappears because she was stored in the last save. Also, because only the group printer object has been committed, the asterisk indicates the existence of other uncommitted, modified objects.

7.2.5 Removing A User

Removing a user using cmsh corresponds roughly to the functionality of the Remove button operation in section 7.1.

The remove command removes a user or group. The useful "-r" flag added to the end of the username removes the user's home directory too. For example, within user mode, the command "remove user maureen -r; commit" removes user maureen, along with her home directory. Or, continuing the session at the end of section 7.2.4 from where it was left off:

Example

```
[mycluster->user*[printer]]% use user maureen
[mycluster->user*[maureen]]% remove -r;  commit
[mycluster->user*]% !ls -d /home/* | grep maureen   #no maureen left behind
[mycluster->user*]%
```

7.3 Using An External LDAP Server

When using an external LDAP server to serve the user database, a Bright cluster can be configured in different ways to authenticate against it.

For smaller clusters, a configuration where LDAP clients on all nodes point directly to the external server is recommended. An easy way to set this up is as follows:

- On the head node:
  - the URIs in `/etc/ldap.conf`, and in the image file `/cm/images/default-image/etc/ldap.conf` are set to point to the external LDAP server.[1]
  - the `updateprovisioners` command (section 6.2.4) is run to update any other provisioners.
- Then, to update configurations on the regular nodes:
  - They can simply be rebooted to pick up the updated configuration.
  - Alternatively, to avoid a reboot, the `imageupdate` command (section 6.6.2) can be run to pick up the new image from a provisioner.
- If using the user portal with user authentication (section 10.9), the changes described in section 10.9.1 for external LDAP server configuration should be carried out
- In the CMDaemon configuration file `cmd.conf` (Appendix C):
  - If another LDAP tool is to be used to manage external LDAP user management instead of `cmgui` or `cmsh`, then altering `cmd.conf` is not required.
  - If, however, system users and groups are to be managed via `cmgui` or `cmsh`, then CMDaemon, too, must refer to the external LDAP server instead of the default LDAP server on the head node. To set that up:
    - The `LDAPHost`, `LDAPUser`, `LDAPPass`, and `LDAPSearchDN` directives in `cmd.conf` are changed to refer to the external LDAP server.
    - CMDaemon is restarted to enable the new configurations.

For larger clusters the preceding solution can cause issues due to traffic, latency, security and connectivity fault tolerance. If such occur, a better solution is to replicate the external LDAP server onto the head node, hence keeping all cluster authentication local, and making the presence of the external LDAP server unnecessary except for updates. This optimization is described in the next section.

[1] In distributions that are derived from RHEL 6 or higher, `/etc/ldap.conf` file does not exist. The files in which the changes then need to be made are `/etc/nslcd.conf` and `/etc/pam_ldap.conf`.

7.3.1 External LDAP Server Replication

This section explains how to set up replication for an external LDAP server to an LDAP server that is local to the cluster, if improved LDAP services are needed. Section 7.3.2 then explains how this can then be made to work with a high availability setup.

Typically, the Bright LDAP server is configured as a replica (consumer) to the external LDAP server (provider), with the consumer refreshing its local database at set timed intervals. How the configuration is done varies according to the LDAP server used. The description in this section assumes the provider and consumer both use OpenLDAP.

External LDAP Server Replication: Configuring The Provider

It is advisable to back up any configuration files before editing them.

The provider is assumed to be an external LDAP server, and not necessarily part of the Bright cluster. The LDAP TCP ports `389` and `689` may therefore need to be made accessible between the consumer and the provider by changing firewall settings.

If a provider LDAP server is already configured then the following synchronization directives must be in the `slapd.conf` file to allow replication:

```
index entryCSN eq
index entryUUID eq
overlay syncprov
syncprov-checkpoint <ops> <minutes>
syncprov-sessionlog <size>
```

The openldap documentation (`http://www.openldap.org/doc/`) has more on the meanings of these directives. If the values for `<ops>`, `<minutes>`, and `<size>` are not already set, typical values are:

```
syncprov-checkpoint 1000 60
```

and:

```
syncprov-sessionlog 100
```

To allow the consumer to read the provider database, the consumer's access rights need to be configured. In particular, the `userPassword` attribute must be accessible. LDAP servers are often configured to prevent unauthorized users reading the `userPassword` attribute.

Read access to all attributes is available to users with replication privileges. So one way to allow the consumer to read the provider database is to bind it to replication requests.

Sometimes a user for replication requests already exists on the provider, or the root account is used for consumer access. If not, a user for replication access must be configured.

A replication user, `syncuser` with password `secret` can be added to the provider LDAP with adequate rights using the following `syncuser.ldif` file:

```
dn: cn=syncuser,<suffix>
objectClass: person
cn: syncuser
sn: syncuser
userPassword: secret
```

Here, <suffix> is the suffix set in slapd.conf, which is originally something like dc=example,dc=com. The syncuser is added using:

```
ldapadd -x -D "cn=root,<suffix>" -W -f syncuser.ldif
```

This prompts for the root password configured in slapd.conf.

To verify syncuser is in the LDAP database the output of ldapsearch can be checked:

```
ldapsearch -x "(sn=syncuser)"
```

To allow access to the userPassword attribute for syncuser the following lines in slapd.conf are changed, from:

```
access to attrs=userPassword
  by self write
  by anonymous auth
  by * none
```

to:

```
access to attrs=userPassword
  by self write
  by dn="cn=syncuser,<suffix>" read
  by anonymous auth
  by * none
```

Provider configuration is now complete and the server can be restarted using /etc/init.d/ldap restart.

External LDAP Server Replication: Configuring The Consumer(s)

The consumer is an LDAP server on a Bright head node. It is configured to replicate with the provider by adding the following lines to /cm/local/apps/openldap/etc/slapd.conf:

```
syncrepl rid=2
  provider=ldap://external.ldap.server
  type=refreshOnly
  interval=01:00:00:00
  searchbase=<suffix>
  scope=sub
  schemachecking=off
  binddn=cn=syncuser,<suffix>
  bindmethod=simple
  credentials=secret
```

Here:

- The rid=2 value is chosen to avoid conflict with the rid=1 setting used during high availability configuration (section 7.3.2).
- The provider argument points to the external LDAP server.
- The interval argument (format DD:HH:MM:SS) specifies the time interval before the consumer refreshes the database from the external LDAP. Here, the database is updated once a day.
- The credentials argument specifies the password chosen for the syncuser on the external LDAP server.

More on the syncrepl directive can be found in the openldap documentation (http://www.openldap.org/doc/).

The configuration files must also be edited so that:

- The <suffix> and rootdn settings in slapd.conf both use the correct<suffix> value, as used by the provider.
- The <base> value in the /etc/ldap.conf uses the correct <suffix> value as used by the provider. This is set on all Bright cluster nodes. If the ldap.conf file does not exist, then the footnote on page 174 applies.

Finally, before replication takes place, the consumer database is cleared. This can be done by removing all files, except for the DB_CONFIG file, from under the configured database directory, which by default is at /var/lib/ldap/.

The consumer is restarted using service ldap restart. This replicates the provider's LDAP database, and continues to do so at the specified intervals.

7.3.2 High Availability

No External LDAP Server Case

If the LDAP server is not external—that is, if the Bright Cluster Manager is set to its high availability configuration, with its LDAP servers running internally, on its own head nodes—then by default LDAP services are provided from both the active and the passive node. The high-availability setting ensures that CMDaemon takes care of any changes needed in the slapd.conf file when a head node changes state from passive to active or vice versa, and also ensures that the active head node propagates its LDAP database changes to the passive node via a syncprov/syncrepl configuration in slapd.conf.

External LDAP Server With No Replication Locally Case

In the case of an external LDAP server being used, but with no local replication involved, no special high-availability configuration is required. The LDAP client configuration in /etc/ldap.conf simply remains the same for both active and passive head nodes, pointing to the external LDAP server. The file /cm/images/default-image/etc/ldap.conf in each image directory also points to the same external LDAP server. If the ldap.conf files referred to here in the head and software images do not exist, then the footnote on page 174 applies.

External LDAP Server With Replication Locally Case

In the case of an external LDAP server being used, with the external LDAP provider being replicated to the high-availability cluster, it is generally more efficient for the passive node to have its LDAP database propagated and updated only from the active node to the passive node, and not updated from the external LDAP server.

The configuration should therefore be:

- an active head node that updates its consumer LDAP database from the external provider LDAP server

- a passive head node that updates its LDAP database from the active head node's LDAP database

Although the final configuration is the same, the sequence in which LDAP replication configuration and high availability configuration are done has implications on what configuration files need to be adjusted.

1. For LDAP replication configuration done after high availability configuration, adjusting the new suffix in `/cm/local/apps/openldap/etc/slapd.conf` and in `/etc/ldap.conf` on the passive node to the local cluster suffix suffices as a configuration. If the `ldap.conf` file does not exist, then the footnote on page 174 applies.

2. For high availability configuration done after LDAP replication configuration, the initial LDAP configurations and database are propagated to the passive node. To set replication to the passive node from the active node, and not to the passive node from an external server, the provider option in the `syncrepl` directive on the passive node must be changed to point to the active node, and the suffix in `/cm/local/apps/openldap/etc/slapd.conf` on the passive node must be set identical to the head node.

The high availability replication event occurs once only for configuration and database files in Bright Cluster Manager's high availability system. Configuration changes made on the passive node after the event are therefore persistent.

7.4 Using Kerberos Authentication

The default Bright Cluster Manager 5.2 setup uses LDAP for storing user information and for authentication. This section describes how LDAP can be configured to use a Kerberos V5 authentication back end, assuming a Kerberos server has already been set up.

The resulting combination setup then retains user information such as the login shell, home directory and UID in the LDAP database, while the password and validity period information are managed by the Kerberos database.

7.4.1 Matching Realms

Both LDAP and Kerberos manage different realms such as `example.com` or `cm.cluster`. For LDAP to authenticate against Kerberos there must be a matching realm between them. Changing the LDAP realms to match the Kerberos realm is done as follows:

1. The Kerberos realm can be accessed in `/etc/krb5.conf` on the Kerberos server. Its value is noted.

2. In `/cm/local/apps/openldap/etc/slapd.conf`, these lines should be updated to match the Kerberos realm by replacing `dc=cm,dc=cluster`:

```
suffix          "dc=cm,dc=cluster"
rootdn          "cn=root,dc=cm,dc=cluster"
```

The LDAP server is then restarted with the command:

```
service ldap restart
```

3. The `ldap.conf` file on all nodes should also be modified to match the new realm, by modifying `dc` attributes in the following line:

```
base dc=cm,dc=cluster
```

This modification can be implemented by changing:

(a) `/etc/ldap.conf` on the head node

(b) `/cm/images/<image>/etc/ldap.conf` on the head node, where `<image>` indicates the image used for the non-head nodes. Running the `imageupdate` command (section 6.6.2) then implements the changes to the non-head nodes.

If the `ldap.conf` file does not exist on any of the nodes, then the footnote on page 174 applies.

7.4.2 Configuring The LDAP Server As A Kerberos Client

Assuming the Kerberos server is a different server from the LDAP server, then the LDAP server on the head node should be configured as a Kerberos client. The changes are implemented as follows:

Configuring The LDAP Server As A Kerberos Client: LDAP Server Changes

The `/etc/krb5.conf` file is copied from the Kerberos server onto the Bright head node.

On the Kerberos server the `kadmin` shell is entered, and the LDAP server is created as a principal:

Example

```
addprinc -randkey host/master.cm.cluster
```

Here, `master.cm.cluster` should match the fully qualified domain name of the LDAP server. The `kadmin` shell is exited using the `exit` command.

On the LDAP server, the `kadmin` shell is entered, and the principal added to the keytab:

Example

```
ktadd host/master.cm.cluster
```

As with the `addprinc` command, `master.cm.cluster` should correspond to the LDAP server's fully qualified domain name.

Configuring The LDAP Server As A Kerberos Client: Node Changes

The procedure in the previous section is repeated for all nodes in the cluster.

The easiest way is to modify the image under `/cm/images`. The file `/etc/krb5.conf` is copied to `/cm/images/<image>/etc/krb5.conf`.

For each node the following command is issued on the Kerberos server using the `kadmin` shell:

Example

```
addprinc -randkey host/<nodenumber>.cm.cluster
```

where <*nodenumber*>.cm.cluster represents the node hostname, with <*nodenumber*> typically taking values of `node001`, `node002` and so on.

On the Bright head server, after chrooting to the <*image*> directory with:

```
chroot /cm/images/<image>/
```

the `kadmin` shell is entered. For each regular node in the image, the following keytab command is run:

```
ktadd host/<nodenumber>.cm.cluster
```

7.4.3 Configuring PAM

The system-auth service is configured in `/etc/pam.d/system-auth` with the following rules added:

```
auth sufficient pam_krb5.so use_first_pass

account [default=bad success=ok user_unknown=ignore] pam_krb5.so

password sufficient pam_krb5.so use_authtok

session optional pam_krb5.so
```

Similar entries exist for LDAP authentication, which if left in there allows users to either authenticate against LDAP or against Kerberos. LDAP authentication can be disabled by removing the lines including `pam_ldap.so`, thereby allowing users to only authenticate with Kerberos.

7.5 Tokens And Profiles

Tokens are used to assign capabilities to users, who are grouped according to their assigned capabilities. A *profile* is the name given to each such group. A profile thus consists of a set of tokens. The profile is stored as part of the authentication certificate generated to run authentication operations to the cluster manager for the certificate owner. Authentication is introduced earlier in section 3.3.

The certificate can be generated within `cmsh` by using the `createcertificate` operation from within `cert` mode. Alternatively, it can be generated within `cmgui` by using the `Add` dialog of the `Certificates` tabbed pane within the `Authentication` resource.

Every cluster management operation requires the user's profile to have the relevant tokens for the operation.

Profiles are handled with the `profiles` mode of `cmsh`, or from the `Authorization` resource of `cmgui`. The following default profiles are available:

Profile name	Default Tasks Allowed
Admin	all tasks
CMHealth	health-related prejob tasks
Node	node-related
Power	device power
Readonly	view-only

Custom profiles can be created to include a custom collection of capabilities in cmsh and cmgui. Cloning of profiles is also possible from cmsh.

7.5.1 Creating A New Certificate For cmsh Users

Creating a new certificate in cmsh is done from cert mode using the createcertificate command, which has the following help text:

```
[bright52->cert]% help createcertificate
Name:
          createcertificate - Create a new certificate

Usage:
          createcertificate <key-length> <common-name> <organization> <o\
rganizational-unit> <locality> <state> <country> <profile> <sys-login> <\
days> <key-file> <cert-file>

Arguments:
          key-file
                Path to key file that will be generated

          cert-file
                Path to pem file that will be generated
```

Accordingly, as an example, a certificate file with a read-only profile set to expire in 30 days, to be run with the privileges of user peter, can be created with:

Example

```
createcertificate 1024 democert a b c d ef readonly peter 30 /home/peter\
/peterfile.key /home/peter/peterfile.pem

Thu Apr 14 15:10:53 2011 [notice] bright52: New certificate request wit\
h ID: 1
[bright52->cert]% createcertificate 1024 democert a b c d ef readonly pe\
ter 30 /home/peter/peterfile.key /home/peter/peterfile.pem
Certificate key written to file: /home/peter/peterfile.key
Certificate pem written to file: /home/peter/peterfile.pem
```

Users given this certificate can then carry out cmdaemon tasks that have a read-only profile and as user peter.

7.5.2 Creating A New Certificate For cmgui Users

In a similar way to how cmsh creates a certificate and key files in the preceding section, cmgui users can create a certificate and a .pfx file. This is done via the Authentication resource of cmgui, using the Certificates tab (figure 7.3):

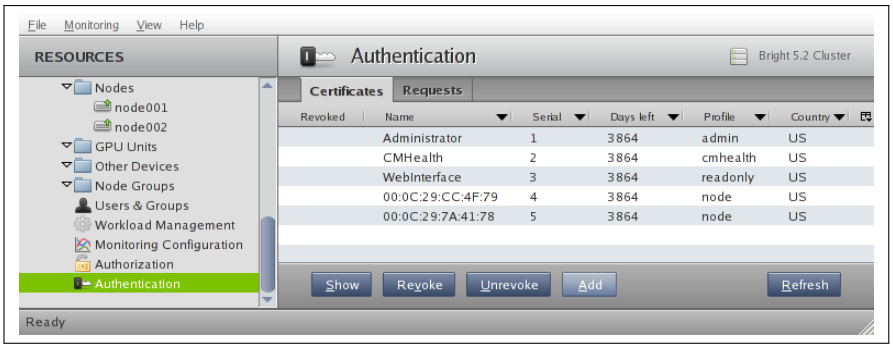

Figure 7.3: `cmgui` Certificates Tab

After clicking on the `Add` button of the `Certificates` tab, a dialog comes up in which the certificate is set up, and a profile selected (figure 7.4):

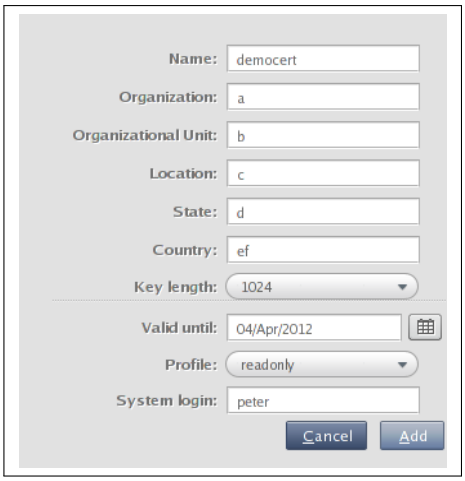

Figure 7.4: `cmgui` Add Certificate And Profile Dialog

Clicking on the `Add` button in figure 7.4 saves the certificate, and generates a `.pfx`. Another dialog then opens up to prompt the user for the path to where the key is to be saved. A password to protect the key with is also asked for (figure 7.5).

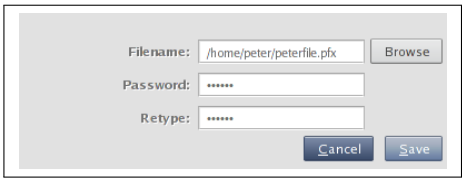

Figure 7.5: `cmgui` Password-protect Key And Save

Users that use this certificate for their `cmgui` clients are then restricted to the set of tasks allowed by their profile, and carry out the tasks with the privileges of the specified system login name (`peter` in figure 7.4).

8

Workload Management

For clusters that have many users and a significant load, a workload management system allows a more efficient use of resources to be enforced for all users than if there were no such system in place. This is because without resource management, there is a tendency for each individual user to over-exploit common resources.

When a workload manager is used, the user submits a batch (i.e. non-interactive) job to it. The workload manager assigns resources to the job, and checks the current availability as well as checking its estimates of the future availability of the cluster resources that the job is asking for. The workload manager then schedules and executes the job based on the assignment criteria that the administrator has set for the workload management system. After the job has finished executing, the job output is delivered back to the user.

Among the hardware resources that can be used for a job are GPUs. Installing CUDA software to enable the use of GPUs is described in section 12.5.

The details of job submission from a user's perspective are covered in the *User Manual*.

Sections 8.1–8.5 cover the installation procedure to get a workload manager up and running.

Sections 8.6–8.7 describe how `cmgui` and `cmsh` are used to view and handle jobs, queues and node drainage.

Section 8.8 shows examples of workload manager assignments handled by Bright Cluster Manager.

Section 8.9 ends the chapter by describing the power saving features of workload managers.

8.1 Workload Managers Choices

Some workload manager packages are installed by default, others require registration from the distributor before installation.

During cluster installation, a workload manager can be chosen (figure 2.17) for setting up. The choices are:

- None
- SLURM v2.2.7 (default)

- Grid Engine 6.2u5p2 (SGE, Open Grid Scheduler fork). An open source development of Sun Grid Engine
- Torque v2.5.12 and its built-in scheduler
- Torque v2.5.12 and the Maui scheduler
- Torque v2.5.12 and the Moab scheduler
- PBS Pro v11.02

These workload managers can also be chosen and set up later using the `wlm-setup` tool (section 8.3).

Besides the preceding workload managers, Load Sharing Facility v7 (LSF) installation is also possible. This is handled separately (section 8.5.5).

8.2 Forcing Jobs To Run In A Workload Management System

Another preliminary step is to consider forcing users to run jobs only within the workload management system. Having jobs run via a workload manager is normally a best practice.

For convenience, a Bright Cluster defaults to allowing users to login to a node and run their processes outside the workload management system without restriction. For clusters with a significant load this policy results in a sub-optimal use of resources, since such unplanned-for jobs disturb any already-running jobs.

Disallowing user logins to nodes, so that users have to run their jobs through the workload management system, means that jobs on the nodes are then disturbed only according to the planning of the workload manager. If planning is based on sensible assignment criteria, then resources use is optimized—which is the entire aim of a workload management system in the first place.

8.2.1 Disallowing User Logins To Nodes Via `cmsh`

The `usernodelogin` setting of `cmsh` restricts direct user logins from outside the workload manager, and is thus one way of preventing the user from using node resources in an unaccountable manner. The `usernodelogin` setting is applicable to node categories only, rather than to individual nodes.

In `cmsh` the attribute of `usernodelogin` is set from within `category` mode:

Example

```
[root@bright52 ~]# cmsh
[bright52]% category use default
[bright52->category[default]]% set usernodelogin onlywhenjob
[bright52->category*[default*]]% commit
```

The attributes for `usernodelogin` are:

- `always` (the default): This allows all users to `ssh` directly into a node from anywhere.

- `never`: This allows no user other than root to directly `ssh` into the node.
- `onlywhenjob`: This allows the user to `ssh` directly into the node when running a job on a node. It typically also prevents other users from doing a direct `ssh` into the same node during the job run, since typically the workload manager is set up so that only one job runs per node.

8.2.2 Disallowing User Logins To Nodes Via `cmgui`

In `cmgui`, user node login access is set from the `Settings` tab for a category selected in the `Node Categories` resource, in the section labeled "`User node login`" (figure 8.1).

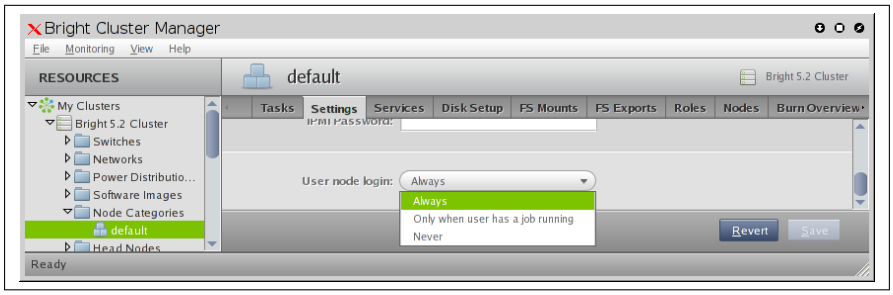

Figure 8.1: Disallowing User Logins To Nodes Via `cmgui`

8.2.3 Disallowing Other User Processes Outside Of Workload Manager User Processes

Besides disabling user logins, administrators may choose to disable interactive jobs in the workload management system as an additional measure to prevent users from starting jobs on other nodes.

Administrators may also choose to set up post-job scripts that terminate user processes outside the workload manager, as part of a policy, or for general administrative hygiene. Such scripts are usually called "`Epilog`" scripts and part of the workload manager.

The workload management system documentation has more on configuring these options.

8.3 Installation Of Workload Managers

Normally the administrator selects a workload manager to be used during Bright Cluster Manager installation (figure 2.17).

A workload manager may however also be added after Bright Cluster Manager has been installed. In that case, the command-line tool `wlm-setup` can be used to install and initialize a workload manager. It is also able to enable or disable the workload manager services. The options available can be viewed on running `wlm-setup -h`.

8.3.1 Setting Up, Enabling, And Disabling The Workload Manager With `wlm-setup`

The following options from `wlm-setup` must be chosen to set up, enable or disable a workload manager for Bright Cluster Manager:

- `-w` or `--wlmanager`: the **workload manager** option, together with <*name*>, the workload manager name. The value of <*name*> can be one of
  - `sge`
  - `torque` (Torque with its built-in scheduler)
  - `torquemaui` (Torque with the Maui scheduler)
  - `torquemoab` (Torque with the Moab scheduler)
  - `slurm`
  - `pbspro`

and for the named workload manager, either:

- `-s` or `--setup`: the **setup** option

  or

- `-p` or `--powersave`: the **powersave option**, which enables the power management options without enabling the workload manager. For `pbspro` this option only works if the PBS Pro server is already running.

  or

- `-e` or `--enable`: the **enable** option, which enables the workload manager
  - adding the `-p` or `--powersave` option to this also enables power management

  or

- `-d` or `--disable`: the **disable** option, which disables the workload manager
  - adding the `-p` or `--powersave` option to this disables power management without disabling the workload manager

  or

- `-i` or `--image`: the **image** option, which places the job execution daemons in a specific software image, for example when a new node category is added to the cluster.

The setup action of `wlm-setup` installs a workload manager package along with its scheduler. Roles, queues, and databases used by the workload manager are initialized on the head. Software images stored on the head node are also set up. The nodes themselves are only updated after manually running `imageupdate` (section 6.6.2)—the idea being to avoid an automatic update so that an administrator is encouraged to check via a dry-run if unintended consequences would happen.

The enable and disable actions of `wlm-setup` are the same as those that take place during role assignment (sections 8.4.1 and 8.4.2). These actions enable or disable the workload manager as a service.

For example, setting up SGE can done as follows:

Example

```
[root@bright52 ~]# wlm-setup -w sge -s
                  Disabling sge services  .....   [  OK  ]
                  Initializing sge setup  .....   [  OK  ]
                  Installing sge qmaster  .....   [  OK  ]
                 Updating image services  .....   [  OK  ]
                 Updating qmaster config  .....   [  OK  ]
               Creating default sge setup  .....   [  OK  ]
                      Setting permissions  .....   [  OK  ]
                   Enabling sge services  .....   [  OK  ]
```

For example, SLURM job daemons can be installed in the node image `new-image` as follows:

Example

```
wlm-setup -w slurm -i /cm/images/new-image
```

If there are provisioning nodes, the `updateprovisioners` command (section 6.2.4) should be run after the software image is changed. The nodes can then simply be rebooted to pick up the new image, or alternatively, to avoid rebooting, the `imageupdate` command (section 6.6.2) places a new image on the node from the provisioner.

8.3.2 Other Options With `wlm-setup`

Other options exist for `wlm-setup`, including:

- **offload** The offload option (`-o` or `--offload`) deals with setting up, enabling or disabling a workload manager server running on another specified node, other than the default head node:

  Example

  ```
  wlm-setup -w slurm -s -o node003
  ```

- **head as a compute node** This option (`-m` or `--mcompute`) sets the head node to join in as a compute node for job tasks in a workload management system. This can be significantly worthwhile on smaller clusters:

  Example

  ```
  wlm-setup -w slurm -m -s
  ```

- **slots** The slots option (`-n` or `--slots`) sets the value of slots (SGE terminology), or `np` (terminology used by other workload managers), and is typically set to the number of cores per node:

  Example

  ```
  wlm-setup -w torque -s -n 4
  ```

8.4 Enabling, Disabling, And Monitoring Workload Managers

After a workload manager package is installed and initialized with `wlm-setup` (section 8.3), it can also be enabled or (if already enabled) disabled, with `wlm-setup`. Enabling and disabling means the workload management services start or stop.

Alternatively, a workload manager can be enabled or disabled by the administrator with `cmgui` or `cmsh`. This is described further on in this section.

In Bright Cluster Manager 5.2, workload managers can even run concurrently. For example, SLURM, SGE, and Torque can run at the same time in the cluster. Using only one workload manager is however generally recommended for production environments.

For ease of use, the administrator can arrange it so that the skeleton file in `/etc/skel/.bashrc` loads only the appropriate workload manager environment module (`slurm`, `sge`, `torque`, or `pbspro`) as the preferred system-wide default for a category of users. Alternatively, users can adjust their personal `.bashrc` files.

From the `cmgui` or `cmsh` point of view a workload manager consists of

- a workload manager server, usually on the head node
- workload manager clients, usually on the compute nodes

Enabling or disabling the servers or clients is then simply a matter of assigning or unassigning a particular workload manager server or client role on the head or compute nodes, as deemed appropriate.

8.4.1 Enabling And Disabling A Workload Manager With `cmgui`

A particular workload manager package may be set up, but the workload manager may not be enabled. This can happen, for example, if using `wlm-setup` (section 8.3) to install the package without enabling it, or for example, if disabling a workload manager that was previously enabled.

The workload manager client and server can be enabled from `cmgui` using the `Roles` tab. Within the `Roles` tab, the properties of the workload manager may be further configured.

Workload Manager Role Assignment To An Individual Node With `cmgui`

Workload Manager Server Enabling the server on a node can be done by clicking on the "`Head Nodes`" or `Nodes` folder, selecting the node item, and selecting the `Roles` tab to display the possible roles. A workload manager server role is then chosen and its options set. For example, for Torque, the Maui or Moab schedulers must be set as options instead of Torque's built-in scheduler, when enabling Torque with Maui or Moab. The workload manager server role is then saved with the selected options (figure 8.2). For starting it up on non-head nodes (but not for a head node), the `imageupdate` command (section 6.6.2) is then run. The workload manager server process and any associated schedulers then automatically start up.

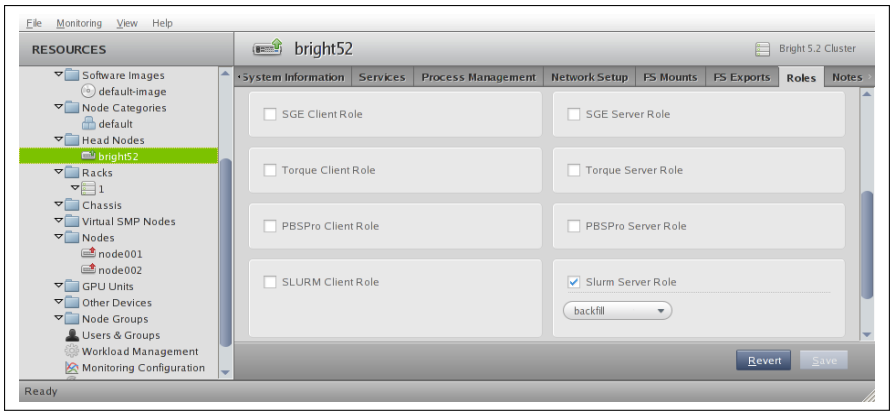

Figure 8.2: Workload Management Role Assignment On A Head Node

Workload Manager Client Similarly, the workload manager client process can be enabled on a node or head node by having the workload manager client role assigned and saved within the `Roles` tab. After running `imageupdate` (section 6.6.2), the client process then automatically starts up.

Workload Manager Role Assignment To A Category With `cmgui`

While workload manager role assignment can be done as described in the preceding text for individual non-head nodes, it is usually more efficient to assign roles using categories due to the large number of compute nodes in typical clusters.

All non-head nodes are by default placed in the `default` category. This means that by default roles in the category are automatically assigned to all non-head nodes, unless by way of exception an individual node configuration overrides the category setting and uses its own role setting instead.

Viewing the possible workload manager roles for the category `default` is done by clicking on the "`Node Categories`" folder, selecting the `default` category, and selecting the `Roles` tab. The appropriate workload manager role is then configured (figure 8.3).

For compute nodes, the role assigned is workload manager client, except for LSF, where the workload manager server role is used for compute nodes. The assigned role for a compute node allows queues and GPUs to be specified, and other parameters depending on the workload manager used.

For workload managers other than LSF, the workload manager server role can also be assigned to a non-head node. For example, a Torque server role can be carried out by a non-head node. This is the equivalent to the `offload` option of `wlm-setup`. As in the individual node assignment case, for Torque with Maui or Torque with Moab, the Maui scheduler or the Moab scheduler options must be set if these schedulers are to be used.

Saving the roles with their options and then running `imageupdate` (section 6.6.2) automatically starts up the newly-configured workload manager.

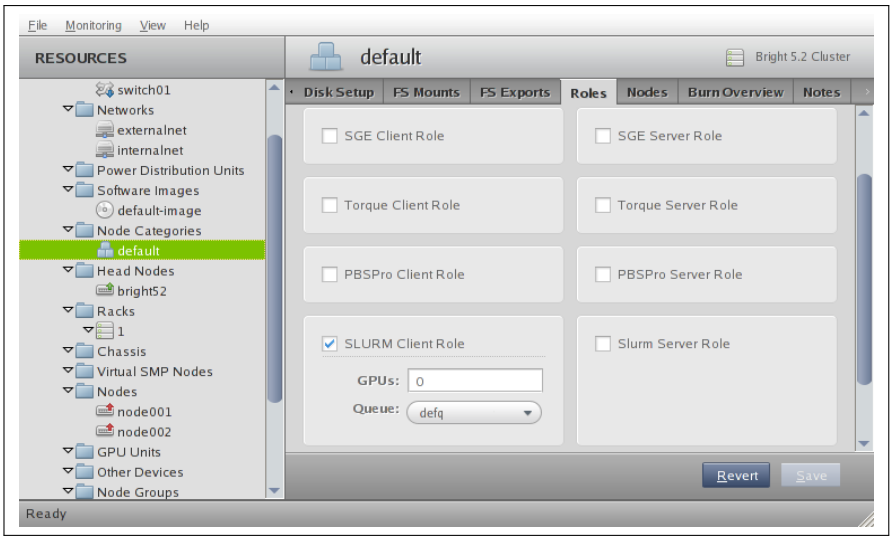

Figure 8.3: Workload Manager Role Assignment By Category For A Compute Node

Workload Manager Client Role Options With `cmgui`

Each compute node role (workload manager client role for non-LFS workload managers and workload manager server role for LFS) has options that can be set for `GPUs`, `Queues`, and `Slots`. "`Slots`", from SGE terminology, corresponds in Bright Cluster Manager to the "`np`" setting in Torque and PBS Pro terminology, and is normally set to the number of cores per node. `Queues` with a specified name are available in their associated role after they are created. The creation of queues is described in sections 8.6.2 (using `cmgui`) and 8.7.2 (using `cmsh`).

Overriding Category Settings Per Node With `cmgui`

If the role for the individual non-head node is set and saved then it overrides its corresponding category role. In `cmgui` this is done by selecting the particular node device from the `Nodes` folder, then selecting the `Roles` tab. The appropriate workload manager client role can then be configured (figure 8.4).

A useful feature of `cmgui` is that the role displayed for the individual node can be toggled between the category setting and the individual setting by clicking on the role checkbox (figure 8.5). Clicking on the `Save` button of the tabbed pane saves the displayed setting.

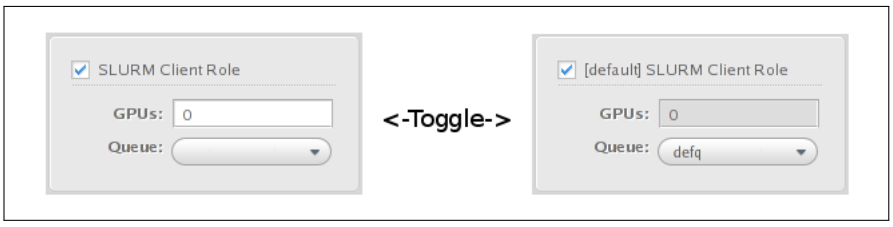

Figure 8.5: Workload Management Role Assignment Toggle States For An Individual Node

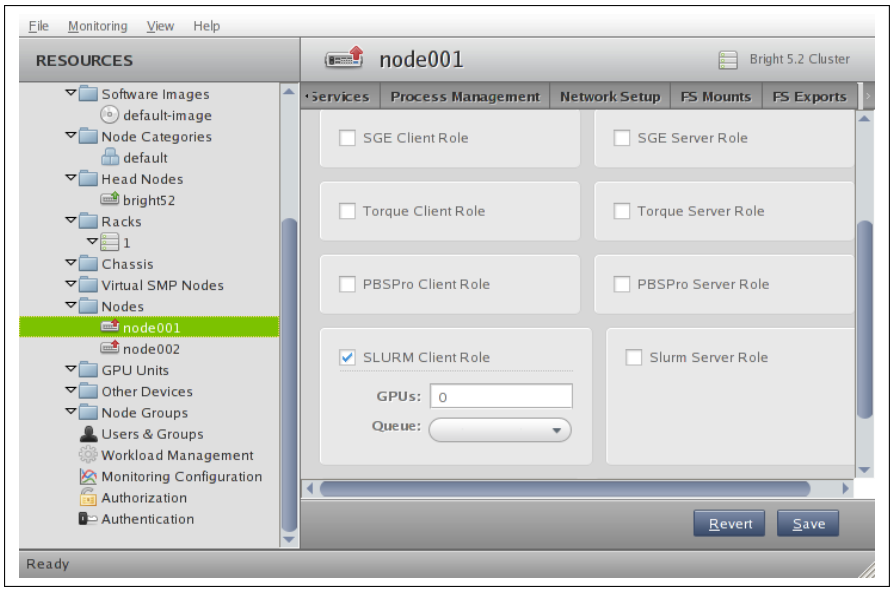

Figure 8.4: Workload Management Role Assignment For An Individual Node

8.4.2 Enabling And Disabling A Workload Manager With `cmsh`

A particular workload manager package may be set up, but not enabled. This can happen, for example, if using `wlm-setup` (section 8.3) on the package without enabling it. The workload manager client and server can be enabled from `cmsh` by assigning it from within the `roles` submode. Within the assigned role, the properties of the workload manager may be further configured.

Workload Manager Role Assignment To A Category With `cmsh`

Workload manager role assignment of a node category is done using `category` mode, using the category name, and assigning a role from the `roles` submode:

Example

```
[root@bright52 ~]# cmsh
[bright52]% category
[bright52->category]% use default
[bright52->category[default]]% roles
[bright52->category[default]->roles]% assign torqueclient
[bright52->category[default]->roles*[torqueclient*]]% commit
[bright52->category[default]->roles[torqueclient]]%
```

After workload manager roles are assigned or unassigned, and after running `imageupdate` (section 6.6.2) for non-head nodes, the associated workload manager services automatically start up or stop as appropriate.

Workload Manager Role Assignment To An Individual Node With `cmsh`

In `cmsh`, assigning a workload manager role to a head node is done in `device` mode, using `master` as the device, and assigning the workload manager role from the `roles` submode:

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->device]% use master
[bright52->device[bright52]]% roles
[bright52->device[bright52]->roles]% assign torqueserver
[bright52->device*[bright52*]->roles*[torqueserver*]]% commit
[bright52->device[bright52]->roles[torqueserver]]%
```

For regular nodes, role assignment is done via `device` mode, using the node name, and assigning a role from the `roles` submode:

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->device]% use node001
[bright52->device[node001]]% roles
[bright52->device[node001]->roles]% assign torqueclient
[bright52->device[node001]->roles*[torqueclient*]]% commit
[bright52->device[node001]->roles[torqueclient]]%
```

Role assignment values set in `device` mode have precedence over any role assignment values set in `category` mode for that node. This means, for example, that if a node is originally in a node category with a `torqueclient` role and queues set, then when the node is assigned a `torqueclient` role from `device` mode, its queue properties are empty by default.

Setting Options For Workload Manager Settings With `cmsh`

In the preceding text, it is explained how the workload manager client or server is assigned a role (such as `torqueclient` or `torqueserver`) within the `roles` submode. It is done from within a main mode of `category` or `devices`.

Whatever the main mode used, the workload manager settings can then be handled with the usual object commands introduced in section 3.5.3. For example, the number of slots can be set for Torque clients as follows:

Example

```
[bright52->category[default]->roles[torqueclient]]% show
Parameter                      Value
------------------------------ --------------------------------
All Queues                     yes
GPUs                           0
Name                           torqueclient
Queues                         shortq longq
Revision
Slots                          4
Type                           TorqueClientRole
[bright52->category[default]->roles[torqueclient]]% set slots 5
[bright52->category*[default*]->roles*[torqueclient*]]% commit
[bright52->category[default]->roles[torqueclient]]%
```

In particular, the `Scheduler` parameter can be set from the `torqueserver` role:

Example

```
[bright52->device[bright52]->roles]% set torqueserver scheduler
backfill  builtin   maui      moab      torque
[bright52->device[bright52]->roles]% use torqueserver
[bright52->device[bright52]->roles[torqueserver]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Name                           torqueserver
Revision
Scheduler                      torque
Type                           TorqueServerRole
[bright52->device[bright52]->roles[torqueserver]]% set scheduler moab
[bright52->device[bright52]->roles[torqueserver*]]% commit
[bright52->device[bright52]->roles[torqueserver]]%
```

In the preceding example, the Moab scheduler has been enabled. After running `imageupdate` (section 6.6.2) for non-head nodes, the built-in Torque scheduler stops and the Moab scheduler starts. It is possible to switch to the Maui scheduler with the same approach. The Moab or Maui schedulers do however need to be installed before starting them in this manner.

In `cmgui`, when the `Torque Server Role` is selected in the `Roles` tab of the node, a drop-down list of supported schedulers appears. Selecting a new scheduler and saving it enables the new scheduler. Running `imageupdate` (section 6.6.2) for non-head nodes stops the previous scheduler and starts the newly selected and enabled scheduler.

8.4.3 Monitoring The Workload Manager Services

By default, the workload manager services are monitored. The Bright Cluster Manager attempts to restart the services using the service tools (section 4.8), unless the role for that workload manager service is disabled, or the service has been stopped (using `cmsh` or `cmgui`). Workload manager roles and corresponding services can be disabled using `wlm-setup` (section 8.3.1), `cmgui` (section 8.4.1), or `cmsh` (section 8.4.2).

The daemon service states can be viewed for each node via the shell, `cmsh`, or `cmgui` (section 4.8).

Queue submission and scheduling daemons normally run on the head node. From `cmgui` their states are viewable by clicking on the node folder in the resources tree, then on the node name item, and selecting the `Services` tab (figures 4.16 and 10.5).

The job execution daemons run on compute nodes. Their states are viewable by clicking on the `Nodes` folder, then on the node name item, and selecting the `Services` tab.

From `cmsh` the services states are viewable from within `device` mode, using the `services` command. One-liners from the shell to illustrate this are (output elided):

Example

```
[root@bright52 ~]# cmsh -c "device services node001; status"
        sgeexecd[   UP   ]
[root@bright52 ~]# cmsh -c "device services master; status"
                    ...
             sge[   UP   ]
```

8.5 Configuring And Running Individual Workload Managers

Bright Cluster Manager deals with the various choices of workload managers in as generic a way as possible. This means that not all features of a particular workload manager can be controlled, so that fine-tuning must be done through the workload manager configuration files. Workload manager configuration files that are controlled by Bright Cluster Manager should normally not be changed directly because Bright Cluster Manager overwrites them. However, overwriting can be prevented by setting the directive:

`FreezeChangesTo<`*workload manager*`>Config = <true|false>`

in `cmd.conf` (Appendix C), where <*workload manager*> takes the value of `SLURM`, `SGE`, `Torque`, or `PBSPro`, as appropriate. The value of the directive defaults to `false`.

A list of configuration files that are changed by CMDaemon, the items changed, and the events causing such a change are listed in appendix L.

A very short guide to some specific workload manager commands that can be used outside of the Bright Cluster Manager 5.2 system is given in Appendix G.

8.5.1 Configuring And Running SLURM

Configuring SLURM

After package setup is done with `wlm-setup` (section 8.3), SLURM software components are installed in `/cm/shared/apps/slurm/current`.

SLURM documentation is available via `man` pages under `/cm/shared/apps/slurm/current/man`. HTML documentation is in the directory `/cm/shared/apps/slurm/current/share/doc/slurm-2.2.7/html`, as well as at the SLURM website at `https://computing.llnl.gov/linux/slurm/slurm.html`.

SLURM is set up with reasonable defaults, but administrators familiar with SLURM can reconfigure the configuration file `/cm/shared/apps/slurm/current/etc/slurm.conf` using the javascript-based configuration generator in `/cm/shared/apps/slurm/2.2.7/share/doc/slurm-2.2.7/html/configurator.html` in a webbrowser.

Running SLURM

SLURM can be disabled and enabled with the `wlm-setup tool` (section 8.3) during package installation itself. Alternatively, role assignment and role removal also enables and disables SLURM from `cmgui` (sections 8.4.1 or `cmsh` (section 8.4.2).

The SLURM workload manager runs these daemons:

1. as servers:
   (a) `slurmdbd`: The database that tracks job accounting. It is part of the `slurmdbd` service.
   (b) `slurmctld`: The controller daemon. Monitors SLURM processes, accepts jobs, and assigns resources. It is part of the `slurm` service.

(c) `munged`: The authentication (client-and-server) daemon. It is part of the `munge` service.

2. as clients:

(a) `slurmd`: The compute node daemon that monitors and handles tasks allocated by `slurmctld` to the node. It is part of the `slurm` service.

(b) `slurmstepd`: A temporary process spawned by the `slurmd` compute node daemon to handle SLURM job steps. It is not initiated directly by users or administrators.

(c) `munged`: The authentication (client-and-server) daemon. It is part of the `munge` service.

Logs for the daemons are saved on the node that they run on. Accordingly, the locations are:

- /var/log/slurmdbd
- /var/log/slurmd
- /var/log/slurmctld
- /var/log/munge/munged.log

8.5.2 Configuring And Running SGE

Configuring SGE

After installation and initialization, SGE has reasonable defaults.

Administrators familiar with SGE can reconfigure it using the template files in `$SGE_ROOT/cm/templates`, which define the queues, hostgroups and parallel environments. To configure the head node for use with SGE, the `install_qmaster` wrapper script under `$SGE_ROOT` is run. To configure a software image for use with SGE the `install_execd` wrapper script under `$SGE_ROOT` is run.

By default, the SGE application is installed in `/cm/shared/apps/sge/current`, the SGE documentation in `/cm/shared/doc/sge/current`, and job examples in `/cm/shared/docs/sge/current/job_examples`.

Running SGE

SGE can be disabled and enabled with the `wlm-setup tool` (section 8.3) during package installation itself. Alternatively, role assignment and role removal also enables and disables SGE from `cmgui` (sections 8.4.1) or `cmsh` (section 8.4.2).

The SGE workload manager runs the following two daemons:

1. an `sge_qmaster` daemon running on the head node. This handles queue submissions and schedules them according to criteria set by the administrator.

2. an `sge_execd` execution daemon running on each compute node. This accepts, manages, and returns the results of the jobs on the compute nodes.

Messages from the qmaster daemon are logged under:

```
/cm/shared/apps/sge/current/default/spool/
```

For the associated compute nodes the execution log exists in:

```
/cm/local/apps/sge/current/default/spool/node<number>/messages
```

where node<*number*> is the node name, for example: node001, node002 ...

8.5.3 Configuring And Running Torque

Torque is a resource manager controlling the jobs and compute nodes it talks with. Torque has its own built-in scheduler, but since this is quite basic, the open source Maui and the proprietary Moab schedulers are recommended alternatives.

Configuring Torque

The Torque package is installed, but not set up by default on Bright Cluster Manager 5.2. If it is not set up during installation (figure 2.17), it can be set up using the `wlm-setup` tool (section 8.3).

The execution daemon, `pbs_mom` is already in the software images by default and does not need to be installed, even if Maui or Moab are added.

The Torque services can be enabled and disabled via role assignment as described in section 8.4. Resources such as the number of GPUs are configured in that section too for the node or node category in order to set up the corresponding resource definitions in the Torque configuration files.

Torque software components are installed in `/cm/shared/apps/torque/current`, also referred to as the `PBS_HOME`. The `torque` environment module, which sets `$PBS_HOME` and other environment variables, must be loaded in order to submit jobs to Torque. The `man` pages for Torque are then accessible, with `$MANPATH` set to `$PBS_HOME/man`.

Torque documentation is available at the Adaptive Computing website at `http://www.adaptivecomputing.com/resources/docs/`, and in particular the Torque administrator manual is available at `http://www.adaptivecomputing.com/resources/docs/torque/index.php`.

Torque examples are available in `/cm/shared/docs/torque/2.5.12/examples`

Installing The Maui Scheduler Package

If Maui is to be installed, the Maui scheduler source version 3.3.1 is picked up from the Adaptive Computing website at `http://www.adaptivecomputing.com/resources/downloads/maui/index.php`. It is installed over the zero-sized placeholder file on the head node at `/usr/src/redhat/SOURCES/maui-3.3.1.tar.gz`.

Maui documentation is available at `http://www.adaptivecomputing.com/resources/docs/maui/index.php`.

The RPM file is built from the source on the head node for Bright Cluster Manager 5.2 using:

```
rpmbuild -bb /usr/src/redhat/SPECS/maui.spec
```

and the installation is done with:

```
rpm -i /usr/src/redhat/RPMS/x86_64/maui-3.3.1-58_cm5.2.x86_64.rpm
```

The exact version of the rpm file to install may differ from the version shown here. The version freshly generated by the `rpmbuild` process is what should be used.

CMDaemon needs to be aware the scheduler is Maui for nodes in a Torque server role. This can be configured using `wlm-setup` to enable the `torquemaui` option (as shown in section 8.3.1), or using `cmgui` to set the scheduler from the `Roles` tab (as shown in section 8.4.1), or using `cmsh` to assign the scheduler from within the assigned `torqueserver` role (as shown in section 8.4.2).

Installing The Moab Scheduler Package

Moab is not installed by default in Bright Cluster Manager 5.2. Moab and related products must be purchased from Adaptive Computing. Bright Cluster Manager 5.2 expects the init script for Moab to be located in the file `/etc/init.d/moab`. Other files such as the Moab binaries and libraries can be installed in `/cm/shared/apps/moab/<moab.version>` or any other preferred location.

CMDaemon needs to be aware the scheduler is Moab for nodes in a Torque server role. This can be configured using `wlm-setup` to enable the `torquemoab` option (section 8.3.1), or using `cmgui` to set the scheduler from the `Roles` tab (as shown in section 8.4.1) or using `cmsh` to assign the scheduler from within the assigned `torqueserver` role (as shown in section 8.4.2).

Running Torque And Schedulers

The Torque resource manager runs the following two daemons:

1. a `pbs_server` daemon. This handles submissions acceptance, and talks to the execution daemons on the compute nodes when sending and receiving jobs. It writes logs to the `/cm/shared/apps/torque/current/spool/server_logs` directory on its node. Queues for this service are configured with the `qmgr` command.

2. a `pbs_mom` execution daemon running on the nodes that are assigned the compute role. This accepts, manages, and returns the results of jobs on the compute nodes. It writes logs to the `/cm/local/apps/torque/current/spool/mom_logs` directory on the compute nodes.

Jobs are however not be executed unless the scheduler daemon is also running. This typically runs on the head node and schedules jobs for compute nodes according to criteria set by the administrator. The possible scheduler daemons for Torque are:

- `pbs_sched` if Torque's built-in scheduler itself is used. It writes logs to the `/cm/shared/apps/torque/current/spool/sched_logs` directory.

- `maui` if the Maui scheduler is used. It writes logs to `/cm/shared/apps/maui/current/spool/log`.

- moab if the Moab scheduler is used. Log files are written to the spool directory. For example: /cm/shared/apps/moab/moab.version/spool/moab.log if Moab is installed in /cm/shared/apps/moab/<moab.version>.

8.5.4 Configuring And Running PBS Pro

Configuring PBS Pro

PBS Pro can be selected for installation during Bright Cluster Manager 5.2 installation, at the point when a workload manager must be selected (figure 2.17). It can also be installed later on, when the cluster is already set up. In either case, it is offered under a 90-day trial license.

To install and initialize PBS Pro after Bright Cluster Manager has already been set up without PBS Pro, the wlm-setup tool (section 8.3) is used.

PBS Pro software components are then installed and initialized in /cm/shared/apps/pbspro/current, also referred to as the PBS_HOME. Users must load the pbspro environment module, which sets PBS_HOME and other environment variables, in order to use PBS Pro.

PBS Pro documentation is available at http://www.pbsworks.com/SupportDocuments.aspx.

By default, PBS Pro examples are available under the directory /cm/shared/docs/pbspro/current/examples/.

Further configuration of PBS Pro is done using its qmgr command and is covered in the PBS Pro documentation.

Running PBS Pro

PBS Pro runs the following three daemons:

1. a pbs_server daemon running, typically on the head node. This handles submissions acceptance, and talks to the execution daemons on the compute nodes when sending and receiving jobs. It writes logs to the /cm/shared/apps/pbspro/current/spool/server_logs/ directory on its node. Queues for this service are configured with the qmgr command.

2. a pbs_sched scheduler daemon, also typically running on the head node. It writes logs to the /cm/shared/apps/pbspro/current/spool/sched_logs/ directory.

3. a pbs_mom execution daemon running on each compute node. This accepts, manages, and returns the results of jobs on the compute nodes. It writes logs to /cm/local/apps/pbspro/current/spool/mom_logs on the compute nodes.

8.5.5 Installing, Configuring, And Running LSF

Installing LSF

The workload manager LSF, version 7, is integrated into Bright Cluster Manager 5.2 as follows:

1. LSF is installed in the usual way, using the lsfinstall script. In case of an existing failover setup, the installation is done on the active head node.

- Some configuration changes may be needed before running the `lsfinstall` script. The LSF `install.config` file requires edits for at least the following entries:

```
LSF_TOP="/cm/shared/apps/lsf"
LSF_CLUSTER_NAME="<cluster name>"
LSF_MASTER_LIST="<head node list>"
LSF_LICENSE="<license path>"
```

  For `LSF_TOP`, the path "`/cm/shared/apps/lsf`" is recommended.

  Another value can however be set for `LSF_TOP`. If it is set to "`<path>`" (and this must be done on both head nodes if using an existing failover configuration), then the entry

```
LSFProfileScript=<path>/conf/profile.lsf
```

  is set accordingly in `/cm/local/apps/cmd/etc/cmd.conf` (appendix C). CMDaemon is then restarted with:

```
service cmd restart
```

- The `lsfinstall` script is run with the preceding configuration changes:

```
./lsfinstall -f install.config
```

  and the instructions are followed.

2. - LSF is added to the `chkconfig` system on the head node by running:

```
./hostsetup --top=/cm/shared/apps/lsf/ --boot=y
```

     The same is run on the passive head node in case of an existing failover setup.

   - To verify that installation is proceeding as intended so far, the status can be checked with:

```
[root@bright52 ~]# /etc/init.d/lsf status
Show status of the LSF subsystem
lim (pid 21138) is running...
res (pid 17381) is running...
sbatchd (pid 17383) is running...
```

     while default queues can be seen by running:

```
[root@bright52 ~]# /etc/init.d/lsf status
. /cm/shared/apps/lsf/conf/profile.lsf
```

3. The LSF startup script is installed on to the software images, It should not be added via `hostsetup` or `chkconfig`. One way to copy over the files required is:

```
for image in $(find /cm/images/ -mindepth 1 -maxdepth 1 -type d)
do cp /etc/init.d/lsf $image/etc/init.d/; done
```

4. Optionally, the LSF environment can be added to `.bashrc` with:

```
. /cm/shared/apps/lsf/conf/profile.lsf
```

5. The LSF role is added to the head node (bright52 in the following) with:

```
cmsh -c "device roles bright52; assign lsfserver;\
set allqueues yes; set addtomasterlist yes; commit"
```

In case of an existing failover setup, role assignment should be repeated on the passive head node, say head2.

Example

```
cmsh -c "device roles head2; assign lsfserver;\
set allqueues yes; set addtomasterlist yes; commit"
```

6. The LSF role is added to each node category containing LFS nodes. If the only category is default, then the command run is:

```
cmsh -c "category roles default; assign lsfserver;\
set allqueues yes; commit"
```

7. Additional NFS entries for the export path of LFS are configured for the head node and for LSF node categories. If the path is already NFS exported, for example by using the shared directory /cm/shared, then this step is unnecessary. In cmgui the NFS path can be set from the "FS Exports" tab for a "Head Node", or "Node Category", or "Nodes" item (section 4.7.1).

8. The nodes are then rebooted, and the LSF command bhosts then shows a display like:

Example

```
[root@bright52 ~]# bhosts
HOST_NAME   STATUS  JL/U    MAX  NJOBS   RUN  SSUSP  USUSP   RSV
bright52        ok     -      2      0     0      0      0     0
head2           ok     -      0      0     0      0      0     0
node001         ok     -      1      0     0      0      0     0
node002         ok     -      1      0     0      0      0     0
node003    unavail     -      1      0     0      0      0     0
node004     closed     -      1      0     0      0      0     0
node005         ok     -      1      0     0      0      0     0
```

Configuring LSF

- **Queues** By default, the preceding installation procedure makes a node accept all queues in steps 5 and 6 by using "set allqueues yes". These default queues are displayed in figure 8.6.

  A restricted list of queues named "qname1", "qname2" and so on can be set using a command syntax like this instead:

```
set queues <qname1> [<qname2> ...]
```

Alternatively, these, and other queues can be added or deleted using `cmgui` (section 8.6.2) or `cmsh` (section 8.7.2).

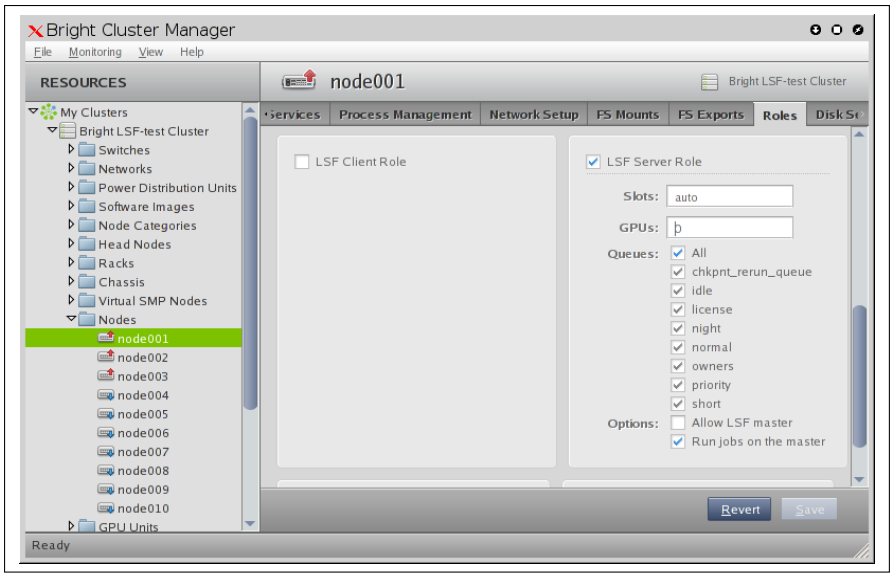

Figure 8.6: `cmgui` access to LSF configuration options via roles tab

- **Options** The options that can be set in figure 8.6 are:
  - **Slots**: The number of CPUs per node (by default LSF tries to determine the value automatically)
  - **GPUs**: The number of GPUs per node
  - **Allow LSF master**: When checked, allows the node, if it is an LSF compute node (a passive LSF-master) to become an active LSF-master if the existing active LSF-master fails
  - **Run jobs on the master**: Allow this LSF compute node to run jobs

  From `cmsh` these properties are accessible from within the appropriate node or category `roles` submode (section 8.4.2).

  Some care must be taken to avoid confusion between LSF terminology and more common terminology. What Bright Cluster Manager treats as LSF compute nodes, these are sometimes called passive LSF-masters, because any such node can become active and take over as an active LSF-master. In contrast, a node that has only the LSF client role in Bright Cluster Manager is simply a login node, not even able to run jobs, and is explicitly not a compute node.

- **Failover** If LSF is set up before setting up failover (section 13.2) for the Bright Cluster Manager head nodes, then nothing special or extra needs to be done when setting up failover for LSF.

- **Further Configuration** For further configuration the *Administering Platform LSF* manual provided with the LSF software should be consulted.

Running LSF

Role assignment and role removal enables and disables LSF from `cmgui` (sections 8.4.1) or `cmsh` (section 8.4.2).

An active LSF master (typically, but not necessarily on a head node) has the following LSF-related processes or services running on it:

Process/Service	Description
`res`	Remote Execution Server*
`sbatchd`	client batch job execution daemon*
`mbatchd`	master batch job execution daemon
`eauth`	External Authentication method
`lim`	Load Information Manager*
`pim`	Process Information Manager*
`pem`	Process Execution Manager*
`vemkd`	Platform LSF Kernel Daemon
`egosc`	Enterprise Grid Orchestrator service controller
`mbschd`	master batch scheduler daemon

*These services/processes run on compute nodes that are not active masters.

Non-active LSF-masters running as compute nodes run the processes marked with an asterisk only.

Logs for LSF processes and services are kept under `/cm/shared/apps/lsf/log/` (or `$LSF_TOP/log/` if the value of `$LSF_TOP` during installation is other than the recommended `/cm/shared/apps/lsf`).

8.6 Using `cmgui` With Workload Management

Viewing the workload manager services from `cmgui` is described in section 8.4.3.

Selecting the Bright Cluster Manager workload manager item from the resources tree displays tabs that let a cluster administrator change the states of:

- jobs
- queues
- nodes

These tabs are described next.

8.6.1 Jobs Display And Handling In `cmgui`

Selecting the `Jobs` tab displays a list of job IDs, along with the scheduler, user, queue, and status of the job (figure 8.7).

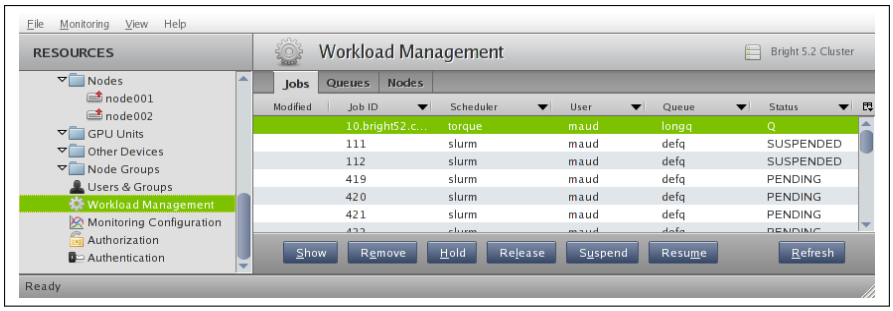

Figure 8.7: Workload Manager Jobs

Within the tabbed pane:

- The `Show` button allows further details of a selected job to be listed.
- The `Remove` button removes selected jobs from the queue.
- The `Hold` button stops selected queued jobs from being considered for running by putting them in a `Hold` state.
- The `Release` button releases selected queued jobs in the `Hold` state so that they are considered for running again.
- The `Suspend` button suspends selected running jobs.
- The `Resume` button allows selected suspended jobs to run again.
- The `Refresh` button refreshes the screen so that the latest available jobs list is displayed.

8.6.2 Queues Display And Handling In `cmgui`

Selecting the `Queues` tab displays a list of queues available, their associated scheduler, and the list of nodes that use each queue (figure 8.8).

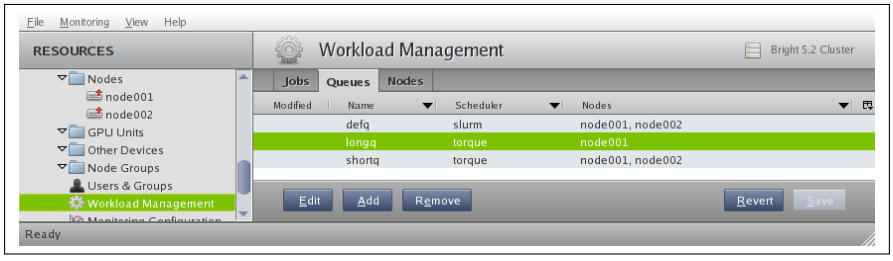

Figure 8.8: Workload Manager Queues

Within the tabbed pane:

- The `Edit` button allows an existing job queue of a workload manager to be edited. The particular values that can be edited for the queue depend upon the workload manager used (figures 8.9, 8.10 and 8.11).

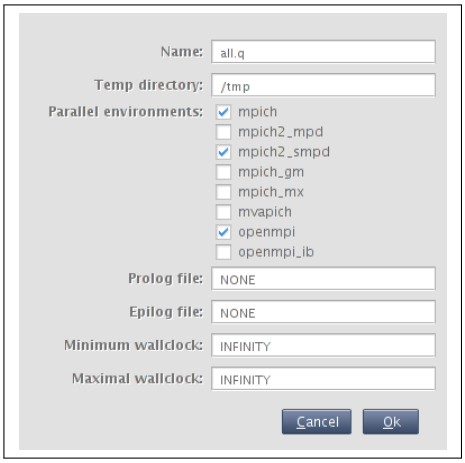

Figure 8.9: Workload Management Queues Edit Dialog For SGE

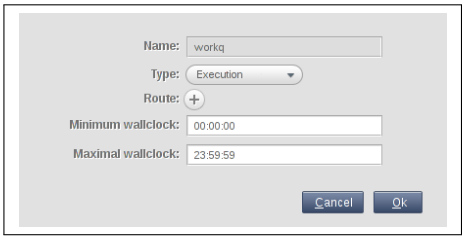

Figure 8.10: Workload Management Queues Edit Dialog For Torque And PBS Pro

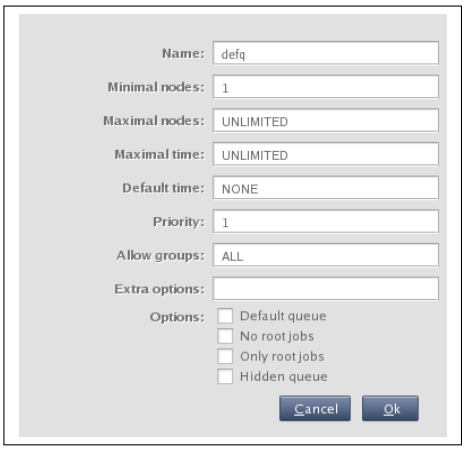

Figure 8.11: Workload Management Queues Edit Dialog For SLURM

In the edit dialog:

– the generic names "`Minimum wallclock`" and "`Maximum wallclock`" correspond respectively to the soft and hard wall-times allowed for the jobs in the queue. Specifically, these are `s_rt` and `h_rt` in SGE, or `resources_default.walltime`, and `resources_max.walltime` in Torque and PBS Pro. The `Maximum time` parameter for SLURM corresponds to `Maximum wallclock` and sets SLURM's `MaxTime` value in

/etc/slurm.conf.

– The Prolog and Epilog files that can be specified in the dialog are scripts run before and after the job is executed. However, for SGE, a default global Prolog configuration is used by Bright Cluster Manager if there is no local script in place. The global configuration ensures that Bright Cluster Manager's health check scripts flagged as prejob scripts (section 10.4.3) run as part of SGE's Prolog script. Administrators creating their own Prolog file may wish to refer to the global Prolog script (cm/prolog under SGE_ROOT), and in particular how it hooks into Bright Cluster Manager prejob checks with a call to cmprejobcheck.

  The Prolog and Epilog scripts for Torque and PBS Pro are set up for the software images and their paths cannot be altered via Bright Cluster Manager.

- The Add button allows a new job queue to be added to a workload manager.
- The Remove button removes a job queue from the workload manager.
- The Revert button reverts the Queues tabbed pane to its last saved state.
- The Save button saves the modified Queues tabbed pane.

8.6.3 Nodes Display And Handling In cmgui

Selecting the Nodes tab displays a list of nodes, along with their schedulers, queues, and whether they are in a status of Drained or Undrained (figure 8.12).

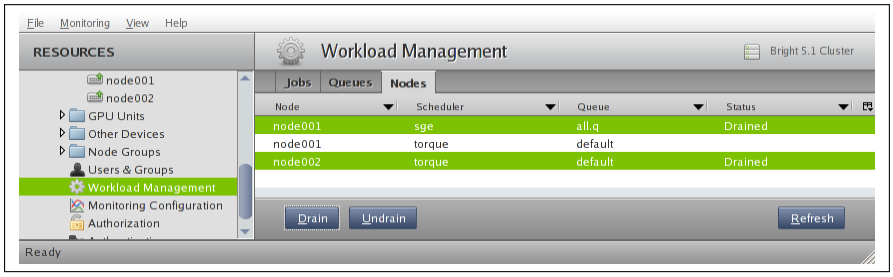

Figure 8.12: Node Drainage

- The Drain button sets the state of a node, scheduler, and queue combination to "Drained". The workload manager then stops jobs from starting to run for that combination.
- The Undrain button unsets a "Drained" state, allowing jobs to start running for that combination.
- The Refresh button refreshes the screen so that the latest available state is displayed.

8.7 Using cmsh With Workload Management

8.7.1 Jobs Display And Handling In cmsh: jobs Mode

jobs Mode In cmsh: Top Level

At the top level of jobs mode, the administrator can view all jobs regardless of scheduler type with the list command:

Example

```
[bright52->jobs]% list
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.bright52+ maud         hydroq       R
```

Also within the jobs mode, the hold, release, suspend, resume, show, and remove commands act on jobs when used with a specified scheduler type and job ID. Continuing with the example:

```
[bright52->jobs]% suspend torque 90.bright52.cm.cluster
Success
[bright52->jobs]% list
Type         jobid        User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.bright52+ maud         hydroq       S
```

While at the jobs mode top level, the suspended job here can be made to resume using suspend's complementary command—resume. However, resume along with the other commands can also be executed within a scheduler submode, as is shown shortly.

jobs Mode In cmsh: The scheduler Submode

Setting the scheduler type sets the scheduler submode, and can be done thus (continuing with the preceding example):

```
[bright52->jobs]% scheduler torque
[bright52->jobs(torque)]%
```

The submode restriction can be unset with: scheduler "".

The top level job mode commands executed within the scheduler submode then only apply to jobs running under that scheduler. The list and resume commands, for example, then only apply only to jobs running under torque (continuing with the example):

```
[bright52->jobs(torque)]% list; !#no sge jobs listed now - only torque
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
TorqueJob    90.bright52+ maud         hydroq       S
[bright52->jobs(torque)]% resume 90.bright52.cm.cluster; !#torque job
Success
[bright52->jobs(torque)]% list; !#only torque jobs
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
TorqueJob    90.bright52+ maud         hydroq       R
```

jobs Mode in cmsh: The show Command

The show command for a particular scheduler and job lists details of the job. Continuing with the preceding example:

```
[bright52->jobs(torque)]% show 90.bright52.cm.cluster;
Parameter                 Value
------------------------- ------------------------------------------
Arguments                 -q hydroq /home/maud/sleeper.sh
Executable
In queue
Job ID                    90.bright52.cm.cluster
Job name                  sleeper.sh
Mail list
Mail options              a
Maximum wallclock time    02:00:00
Memory usage              0
Nodes                     node001
Notify by mail            yes
Number of processes       1
Priority                  0
Queue                     hydroq
Run directory             /home/maud
Running time              809
Status                    R
Stderr file               bright52.cm.cluster:/home/maud/sleeper.sh.e90
Stdout file               bright52.cm.cluster:/home/maud/sleeper.sh.o90
Submission time           Fri Feb 18 12:49:01 2011
Type                      TorqueJob
User                      maud
```

8.7.2 Job Queues Display And Handling In cmsh: jobqueue Mode

Properties of scheduler job queues can be viewed and set in jobqueue mode.

jobqueue Mode In cmsh: Top Level

If a scheduler submode is not set, then the list, qstat, and listpes commands operate, as is expected, on all queues for all schedulers.

At the top level of jobqueue mode:

- list lists the queues associated with a scheduler.

  Example

  ```
  [root@bright52 ~]# cmsh
  [bright52]% jobqueue
  [bright52->jobqueue]% list
  Type         Name
  ------------ ------------------------
  sge          all.q
  torque       default
  torque       hydroq
  torque       longq
  torque       shortq
  ```

- qstat lists statistics for the queues associated with a scheduler.

Example

```
[bright52->jobqueue]% qstat
======================== sge ========================
Queue        Load      Total     Used      Available
------------ --------- --------- --------- ---------
all.q        0.1       1         0         1
======================= torque =======================
Queue        Running   Queued    Held      Waiting
------------ --------- --------- --------- ---------
default      0         0         0         0
hydroq       1         0         0         0
longq        0         0         0         0
shortq       0         0         0         0
======================= pbspro =======================
Queue        Running   Queued    Held      Waiting
------------ --------- --------- --------- ---------
```

- `listpes` lists the parallel environment available for schedulers

Example

(some details elided)

```
[bright52->jobqueue]% listpes
Scheduler    Parallel Environment
------------ ------------------------
sge          make
sge          mpich
...
sge          openmpi_ib
```

- `scheduler` sets the `scheduler` submode

Example

```
[bright52->jobqueue]% scheduler torque
Working scheduler is torque
[bright52->jobqueue(torque)]%
```

The submode can be unset using: `scheduler ""`

`jobqueue` Mode In `cmsh`: The `scheduler` Submode

If a `scheduler` submode is set, then commands under `jobqueue` mode operate only on the queues for that particular scheduler. For example, within the torque submode of `jobqueue` mode, the `list` command shows only the queues for torque.

Example

```
[bright52->jobqueue]% list
Type         Name
------------ ------------------------
sge          all.q
torque       default
```

```
torque       longq
torque       shortq
[bright52->jobqueue]% scheduler torque
Working scheduler is torque
[bright52->jobqueue(torque)]% list
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
```

jobqueue Mode In cmsh: Other Object Manipulation Commands

The usual object manipulation commands of section 3.5.3 work at the top level mode as well as in the scheduler submode:

Example

```
[bright52->jobqueue]% list torque
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[bright52->jobqueue]% show torque longq
Parameter                      Value
------------------------------ ------------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[bright52->jobqueue]% get torque longq maximalruntime
23:59:59
[bright52->jobqueue]%
[bright52->jobqueue]% scheduler torque
Working scheduler is torque
[bright52->jobqueue(torque)]% list
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[bright52->jobqueue(torque)]% show longq
Parameter                      Value
------------------------------ ------------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[bright52->jobqueue(torque)]% get longq maximalruntime
23:59:59
```

```
[bright52->jobqueue(torque)]% use longq
[bright52->jobqueue(torque)->longq]% show
Parameter                      Value
------------------------------ ----------------------------------------------
Maximal runtime                23:59:59
Minimal runtime                00:00:00
Queue type                     Execution
Routes
Type                           torque
name                           longq
nodes                          node001.cm.cluster node002.cm.cluster
[bright52->jobqueue(torque)->longq]% get maximalruntime
23:59:59
```

8.7.3 Nodes Drainage Status And Handling In cmsh

Running the device mode command drainstatus displays if a specified node is in a Drained state or not. In a Drained state jobs are not allowed to start running on that node.

Running the device mode command drain puts a specified node in a "Drained" state:

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->device]% drainstatus
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq
node002                  workq
[bright52->device]% drain node001
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq                    Drained
```

Both the drain and drainstatus commands have the same options. The options can make the command apply to not just one node, but to a list of nodes, a group of nodes, a category of nodes, or to a chassis. Continuing the example:

```
[bright52->device]% drain -c default; !# for a category of nodes
Node                     Queue                    Status
------------------------ ------------------------ ----------------
node001                  workq                    Drained
node002                  workq                    Drained
```

The help text for the command indicates the syntax:

```
[root@bright52 ~]# cmsh -c "device help drain"
Usage: drain ........................ Drain the current node
       drain <node> .................. Drain the specified node
       drain <-n|--nodes nodelist> ... Drain all nodes in the list
       drain <-g|--group group> ...... Drain all nodes in the group
       drain <-c|--category category> . Drain all nodes in the category
       drain <-h|--chassis chassis> .. Drain all nodes in the chassis

       nodelist
                e.g. node001..node015,node20..node028,node030
```

8.7.4 Launching Jobs With cm-launcher

Some MPI distributions occasionally leave processes behind that cannot be killed from the workload manager. To prevent this situation from occurring, Bright Cluster Manager provides the cm-launcher wrapper for the mpirun command, which tracks and kills such processes. The tracking relies on knowing what processes the workload manager launches, so it can only run from within a suitable workload manager. Currently, suitable workload managers are Torque or SGE.

In the following job script examples, instead of having users use:

Example

```
mpirun <...>
```

which may not have a job clean up properly after it ends, users can use:

Example

```
cm-launcher mpirun <...>
```

which cleans up properly after the job ends. Here, <...> is used to indicate the mix of options, programs and arguments used with mpirun.

For Torque cm-launcher can be used if the default Torque epilogue script provided by the Bright Cluster Manager Torque package is present, at /cm/local/apps/torque/current/spool/mom_priv/epilogue.

For SGE the procedure is as follows:

- A symlink to cm-launcher is created to the <arch> SGE functions directory library

  ```
  ln -s /cm/shared/apps/sge/current/cm/cm-launcher /cm/shared/apps/sge\
  /current/bin/<arch>
  ```

- SGE's epilog (spelt without the "-ue" ending) script is set either for a particular queue, or globally.

  - To set it for a particular queue, for example, for the default queue all.q, the following cmsh commands can be run:

    Example

    ```
    [root@bright52 ~]# cmsh
    [bright52]% jobqueue
    [bright52->jobqueue]% scheduler sge
    Working scheduler is sge
    [bright52->jobqueue(sge)]% set all.q epilog /cm/shared/apps/sge\
    /current/cm/epilog
    [bright52->jobqueue*(sge)]% commit
    Successfully committed 1 JobQueues
    [bright52->jobqueue(sge)]%
    ```

  - To set it globally for all queues, that is, not just the queue all.q but all the other queues as well, the following SGE configuration command is used:

```
qconf -mconf global
```

This starts up an editor acting on the `global` configuration setting. The epilog line entry is modified to:

```
epilog                /cm/shared/apps/sge/current/cm/epilog
```

8.8 Examples Of Workload Management Assignment

8.8.1 Setting Up A New Category And A New Queue For It

Suppose a new node with processor optimized to handle Shor's algorithm is added to a cluster that originally has no such nodes. This merits a new category, `shornodes`, so that administrators can configure more such new nodes efficiently. It also merits a new queue, `shorq`, so that users are aware that they can submit suitably optimized jobs to this category.

To create a new queue, the Workload Management item is selected, and the Queues tab selected. The `Add` button is used to associate a newly created queue with a scheduler and add it to the workload manager. The modification is then saved (figure 8.13).

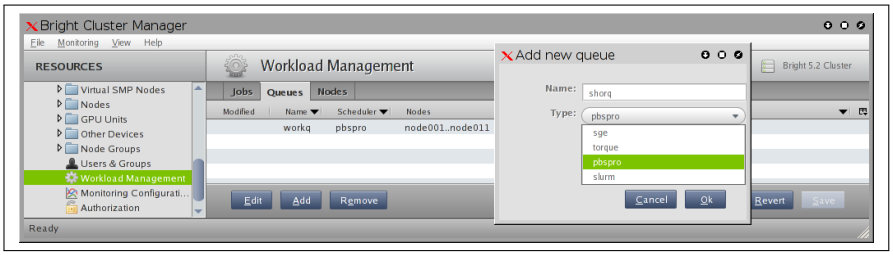

Figure 8.13: Adding A New Queue Via `cmgui`

A useful way to create a new category is to simply clone the old `default` category over to a new category, and then change parameters in the new category to suit the new machine (figure 8.14).

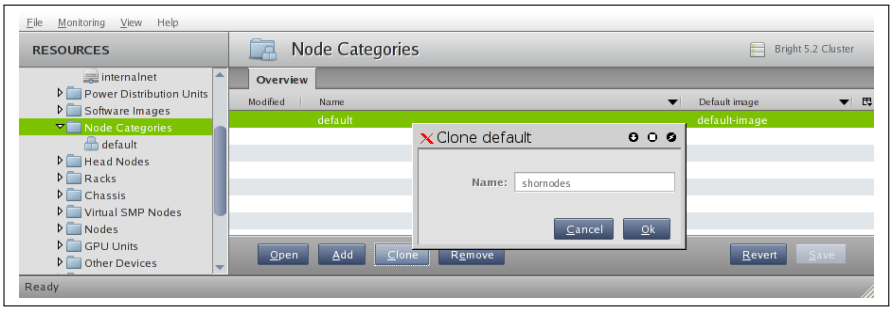

Figure 8.14: Cloning A New Category Via `cmgui`

Having cloned and saved the category, called `shornodes` in the example of figure 8.14, the configuration of the category may be altered to suit the new machine, perhaps by going into the settings tab and altering items there.

Next, the queue is set for this new category, `shornodes`, by going into the `Roles` tabbed pane of that category, selecting the appropriate workload manager client role and queues, and saving the setting (figure 8.15).

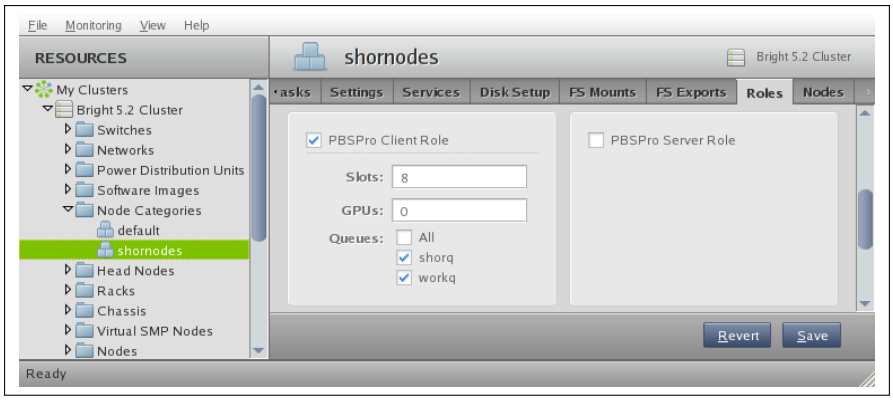

Figure 8.15: Setting A Queue For A New Category Via `cmgui`

Finally, a node in the `Nodes` folder that is to be placed in the new `shornodes` category must be placed there by changing the category value of that node in its `Settings` tab (figure 8.16).

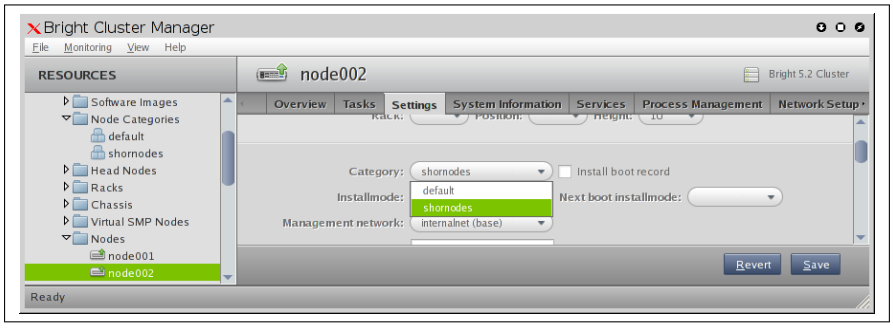

Figure 8.16: Setting A New Category To A Node Via `cmgui`

8.8.2 Setting Up A Prejob Health Check

How It Works

Health checks (section 10.2.4) by default run as scheduled tasks over regular intervals. They can optionally be configured to run as prejob health checks, that is, before a job is run. If the response to a `prejob` health check is `PASS`, then it shows that the node is displaying healthy behavior for that particular health aspect.

If the response to a `prejob` health check is `FAIL`, then it implies that the node is unhealthy, at least for that aspect. A consequence of this may be that a job submitted to the node may fail, or may not even be able to start. To disallow passing a job to such unhealthy nodes is therefore a good policy, and so for a cluster in the default configuration, the action (section 10.2.2) taken defaults to putting the node in a `Drained` state (sections 8.6.3 and 8.7.3), with Bright Cluster Manager arranging a rescheduling of the job.

A node that has been put in a `Drained` state with a health check is not automatically undrained. The administrator must clear such a state manually.

Configuration Using `cmgui`

To configure the monitoring of nodes as a prejob health check in `cmgui`, the `Monitoring Configuration` resource item is selected, and the `Health`

Check Configuration tabbed pane is opened. The default resource is chosen as a value for Health Check Configuration, and the Add button is clicked on to add the health check via a dialog (figure 8.17). In the dialog, the Health Check script value is set to the chosen health check. If the health check is already listed, then it can be edited as needed. The Sampling interval is set to prejob, which automatically sets the Fail action to the Drain node action, when the configuration is saved. After saving these settings, any node that is not in the Drained state in the default resource gets a pre-job check when a job is scheduled for the node, and the pre-job check puts the node in a Drained state if it is unhealthy.

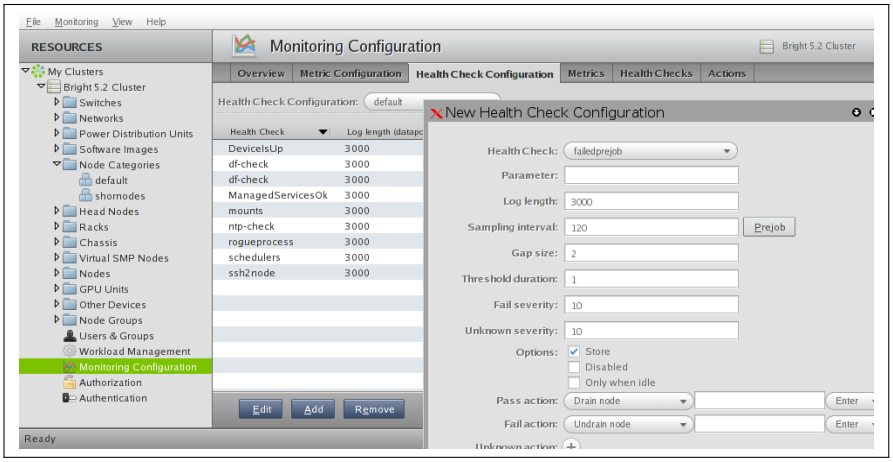

Figure 8.17: Configuring A Prejob Healthcheck Via cmgui

Configuration Using cmsh

To configure a prejob health check with cmsh, the healthconf submode (section 10.7.4) is entered, and the prejob health script object used. In the following example, where some text has been elided, the object is the smart script:

Example

```
[bright52% monitoring setup healthconf default
[bright52->monitoring->setup[default]->healthconf]% use smart
[bright52->...->healthconf[smart]]% set checkinterval prejob
set checkinterval prejob
```

The failactions value automatically switches to "enter: Drain node()" when the value for the checkinterval parameter of the health check is set to prejob.

8.9 Power Saving Features

8.9.1 SLURM

SLURM provides a power saving mechanism that places idle nodes in power save mode. In Bright Cluster Manager, the SLURM power saving features can be enabled with the wlm-setup command from section 8.3:

```
wlm-setup -w slurm -e -p
```

This sets the following default values for power save configuration parameters in the SLURM configuration file:

```
SuspendTime=300
SuspendProgram=/cm/local/apps/cluster-tools/wlm/scripts/slurmpoweroff
ResumeProgram=/cm/local/apps/cluster-tools/wlm/scripts/slurmpoweron
SuspendTimeout=30
ResumeTimeout=60
```

Setting a negative value for SuspendTime disables power saving mode. A positive value is the time period (in seconds) after which a node that has remained idle becomes eligible for power saving mode.

SuspendProgram and ResumeProgram are scripts that act on eligible nodes, using the Bright power control mechanisms to power nodes on and off.

SuspendTimeout and ResumeTimeout are timeout values (in seconds) for when the nodes must respectively be set to a suspended state or a state available for use.

Additional power saving configuration file parameters are described in the SLURM power saving guide at https://computing.llnl.gov/linux/slurm/power_save.html

8.9.2 PBS Pro

PBS Pro provides a power saving mechanism through the Green Provisioning feature. This is installed as an RPM, pbspro-green, on Bright clusters. Green Provisioning integrates with the PBS Pro scheduling cycle, and nodes on the integrated system can then be powered on and off using Bright Cluster Manager power control mechanisms. Power saving features for PBS Pro can be enabled with the wlm-setup command from section 8.3:

```
wlm-setup -w pbspro -e -p
```

The command triggers the cluster management daemon to enable power management and set node level custom resources required for Green Provisioning. The PBS Pro server parameters that enable power saving are:

```
set server default_chunk.pwr_mgt_simpwr = True
set server resources_available.pwr_mgt_enabled = True
```

The following table shows the list of node level resource values that are defined when power saving is enabled:

Name (In PBS Pro)	Name (In CMDaemon)	Default Value
pwr_mgt_can_power_down	PBSProCanPowerDown	True
pwr_mgt_power_down_priority	PBSProPwrMgtPowerDownPriority	30
pwr_mgt_power_down_delay	PBSProPwrMgtPowerDownDelay	30
pwr_mgt_power_up_delay	PBSProPwrMgtPowerUpDelay	60
pwr_mgt_trying_to_power_up	PBSProTryingToPowerUp	False
pwr_mgt_simpwr	PBSProSimpwr	True

A delay time value in the preceding table is the period (in seconds) after issuing the node level resource command, after which the node power command is assumed to have completed its action (i.e., the node is assumed in a powered up or down state after the specified delay time).

The priority value is an arbitrary value per node that decides the order in which a node powers up and down. The PBS Pro documentation has more on this.

The default values for the parameters in the table can be changed by specifying them in the `AdvancedConfig` directive of the CMDaemon configuration file. An example is:

```
AdvancedConfig = {\
"PBSProCanPowerDown=True", "PBSProPwrMgtPowerDownPriority=30", \
"PBSProPwrMgtPowerDownDelay=30", "PBSProPwrMgtPowerUpDelay=60", \
"PBSProTryingToPowerUp=False", "PBSProSimpwr=True"\
}
```

Green Provisioning supports three power cycling strategies:

0. `No power cycling`
1. `Simulate power cycling`
2. `Full power cycling`

The numeric value in the list is used to set the strategy in the Green Provisioning configuration file. For example, to enable "`Full power cycling`", the following is set in the Green Provisioning configuration file at `/cm/shared/apps/pbspro-green/config`:

```
POWER_CYCLE_STRATEGY=2
```

By default, "`Simulate power cycling`" is used. This does not actually carry out the power parameter instruction on the node, but the node is treated as if the power parameter has been applied. So, for example, when power cycling is being simulated, jobs are no longer scheduled to run on the node after `PBSProPwrMgtPowerDownDelay` is run.

9

Post-Installation Software Management

Some time after Bright Cluster Manager has been installed, administrators may wish to manage other software on the cluster. This means carrying out software management actions such as installation, removal, updating, version checking, and so on.

Since Bright Cluster Manager is built on top of an existing Linux distribution, it is best that the administrator use distribution-specific package utilities for software management.

Packages managed by the distribution are hosted by distribution repositories. SUSE and Red Hat enterprise distributions require the purchase of their license in order to access their repositories.

Packages managed by the Bright Cluster Manager are hosted by the Bright Computing repository. Access to Bright repositories also requires a license (section 4.1).

There may also be software that the administrator would like to install that is outside the default packages collection. These could be source files that need compilation, or packages in other repositories.

A software image (section 3.1.2) is the file system that a node picks up from a provisioner (a head node or a provisioning node) during provisioning. A subtopic of software management on a cluster is software image management—the management of software on a software image. By default, a node uses the same distribution as the head node for its base image along with necessary minimal, cluster-mandated changes. A node may however deviate from the default, and be customized by having software added to it in several ways.

This chapter covers the techniques of software management for the cluster.

Section 9.1 describes the naming convention for a Bright Cluster Manager RPM package.

Section 9.2 describes how an RPM package is managed for the head node.

Section 9.3 describes how a kernel RPM package can be managed on a head node or image.

Section 9.4 describes how an RPM package can be managed on the software image.

Section 9.5 describes how a software other than an RPM package can be managed on a software image.

Section 9.6 describes how custom software images are created that are completely independent of the existing software image distribution and version.

Section 9.7 briefly describes how the FrozenFile configuration setting can be used to make all nodes function differently from standard cluster behavior.

Section 9.8 explains how different nodes can be made to load particular images.

Section 9.9 suggests alternatives that may achieve the same result as using different images for different nodes, especially when changes are minor.

9.1 Bright Cluster Manager RPM Packages And Their Naming Convention

Like the distributions it runs on top of, Bright Cluster Manager uses RPM (RPM Package Manager) packages. An example of a Bright Cluster Manager RPM package is:

```
mpich-ge-gcc-64-1.2.7-105_cm5.2.x86_64.rpm
```

The file name has the following structure:

package-*version*-*revision*_cm*x*.*y*.*architecture*.rpm where:

- *package* (mpich-ge-gcc-64) is the name of the package
- *version* (1.2.7) is the version number of the package
- *revision* (105) is the revision number of the package
- *x*.*y* (5.2) is the version of Bright Cluster Manager for which the RPM was built
- *architecture* (x86_64) is the architecture for which the RPM was built

To check whether Bright Computing or the distribution has provided a file that is already installed on the system, the package it has come from can be identified using "rpm -qf".

Example

```
[root@bright52 bin]# rpm -qf /usr/bin/zless
gzip-1.3.12-18.el6.x86_64
[root@bright52 bin]# rpm -qf /cm/local/apps/cmd/sbin/cmd
cmdaemon-5.2-11233_cm5.2.x86_64
```

In the example, /usr/bin/zless is supplied by the distribution, while /cm/local/apps/cmd/sbin/cmd is supplied by Bright Computing, as indicated by the "_cm" in the nomenclature.

More information about the RPM Package Manager is available at http://www.rpm.org.

9.2 Managing Packages On The Head Node

9.2.1 Managing RPM Packages On The Head Node

Once Bright Cluster Manager has been installed, distribution packages and Bright Cluster Manager software packages can be managed using the `rpm` command-line utility.

A more convenient way of managing RPM packages is to use either YUM or zypper. Both of these tools are repository and package managers. The zypper tool is recommended for use by the SUSE 11 distribution, while YUM is recommended for use by the other distributions that Bright Cluster Manager supports. YUM is not installed by default in SUSE 11, and it is better not to install and use it with SUSE 11 unless the administrator is familiar with configuring YUM.

For YUM and zypper, the following commands list all available packages:

```
yum list
or
zypper packages
```

For zypper, the short command option `pa` can also be used instead of `packages`.

The following commands can install a new package called <*package name*>:

```
yum install <package name>
or
zypper in <package name>
```

All installed packages can be updated with:

```
yum update
or
zypper up
```

Bright Computing maintains YUM and zypper repositories at:

`http://updates.brightcomputing.com/yum`

and updates are fetched by YUM and zypper for Bright Cluster Manager packages from there by default.

Accessing the YUM repositories manually (i.e. not through YUM or zypper) requires a username and password. Authentication credentials are provided upon request. For more information on this, `support@brightcomputing.com` should be contacted.

The repository managers use caches to speed up their operations. Occasionally these caches may need flushing to clean up the index files associated with the repository. This is done with:

```
yum clean all
or
zypper clean -a
```

As an extra protection to prevent Bright Cluster Manager installations from receiving malicious updates, all Bright Cluster Manager packages are signed with the Bright Computing GPG public key (0x5D849C16), installed by default in /etc/pki/rpm-gpg/RPM-GPG-KEY-cm for Red Hat, Scientific Linux, and CentOS packages. The Bright Computing public key is also listed in Appendix B.

The first time YUM or zypper are used to install updates, the user is asked whether the Bright Computing public key should be imported into the local RPM database. Before answering with a "Y", yum users may choose to compare the contents of /etc/pki/rpm-gpg/RPM-GPG-KEY-cm with the key listed in Appendix B to verify its integrity. Alternatively, the key may be imported into the local RPM database directly, using the following command:

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-cm
```

Installation of the following Bright Computing packages is described in Chapter 12. These are installed mostly on the head node, but some packages work as a server and client process, with the clients installed and running on the regular nodes:

- Modules (section 12.1)
- Shorewall (section 12.2)
- Compilers (section 12.3):
  - GCC (section 12.3.1)
  - Intel Compiler Suite (section 12.3.2)
  - PGI High-Performance Compilers (section 12.3.3)
  - AMD Open64 Compiler Suite (section 12.3.4)
  - FLEXlm License Daemon (section 12.3.5)
- Intel Cluster Checker (section 12.4)
- CUDA (section 12.5)
- Lustre (section 12.7)
- ScaleMP (section 12.8)

9.2.2 Managing Non-RPM Software On The Head Node

Sometimes a package is not packaged by Bright Computing for the head node. In that case, the software can be usually be treated as for installation onto a standard distribution. There may be special considerations on placement of components that the administrator may feel appropriate due to the particulars of a cluster configuration.

For example, for compilation and installation of the software, some consideration may be made of the options available on where to install parts of the software within the default shared filesystem. A software may have a compile option, say --prefix, that places an application <*application*> in a directory specified by the administrator. If the administrator decides that <*application*> should be placed in the shared directory, so that everyone can access it, the (hypothetical) option could perhaps be specified as: "--prefix=/cm/shared/apps/<*application*>".

Other commonly provided components of software for the applications that are placed in shared may be documentation, licenses, and examples. These may similarly be placed in the directories `/cm/shared/docs`, `/cm/shared/licenses`, and `/cm/shared/examples`. The placement may be done with a compiler option, or, if that is not done or not possible, it could be done by modifying the placement by hand later. It is not obligatory to do the change of placement, but it helps with cluster administration to stay consistent as packages are added.

Module files (sections 3.2 and 12.1) may sometimes be provided by the software, or created by the administrator to make the application work for users easily with the right components. The directory `/cm/shared/modulefiles` is recommended for module files to do with such software.

To summarize the above considerations on where to place software components, the directories under `/cm/shared` that can be used for these components are:

```
/cm/shared/
|-- apps
|-- docs
|-- examples
|-- licenses
`-- modulefiles
```

9.3 Kernel Management On A Head Node Or Image

Care should be taken when updating a head node or software image. This is particularly true when custom kernel modules compiled against a particular kernel version are being used.

9.3.1 Installing A Standard Distribution Kernel

A standard distribution kernel is treated almost like any other package in a distribution. This means that for head nodes, installing a standard kernel is done according to the normal procedures of managing a package on a head node (section 9.2), while for regular nodes, installing a standard distribution kernel onto a regular node is done according to the normal procedures of managing an RPM package inside an image (section 9.4).

An example standard kernel package name is "`kernel-2.6.18-274.3.1.el5`". The actual one suited to a cluster varies according to the distribution used. Packages with names that begin with "`kernel-devel-`" are development packages that can be used to compile custom kernels, and are not required when installing standard distribution kernels.

When installing a kernel, extra considerations for software images are:

- The kernel must also be explicitly set in CMDaemon (section 9.3.3) before it may be used by the regular nodes.
- Some GRUB-related errors show up during the installation of a kernel package in the software image. These occur due to a failure to find the partitions when running the post-install scripts in the chroot environment. They can simply be ignored because the nodes do not boot from GRUB.

As is standard for Linux, both head or regular nodes must be rebooted to use the new kernel.

9.3.2 Excluding Kernels And Other Packages From Updates

Sometimes it may be desirable to exclude the kernel from updates.

- When using `yum`, to prevent an automatic update of a package, the package is listed after using the `--exclude` flag. So, to exclude the kernel from the list of packages that should be updated, the following command can be used:

  ```
  yum --exclude kernel update
  ```

  To exclude a package such as `kernel` permanently from all YUM updates, without having to specify it on the command line each time, the package can instead be excluded inside the repository configuration file. YUM repository configuration files are located in the `/etc/yum.repos.d` directory, and the packages to be excluded are specified with a space-separated format like this:

  `exclude = <package 1> <package 2> ...`

- The `zypper` command can also carry out the task of excluding the kernel package from getting updated when updating. To do this, the kernel package is first locked (prevented from change) using the `addlock` command, then the `update` command is run, and finally the kernel package is unlocked again using the `removelock` command:

  ```
  zypper addlock kernel
  zypper update
  zypper removelock kernel
  ```

9.3.3 Updating A Kernel In A Software Image

For a software image, if the kernel is updated by the repository manager, then the kernel is not used on reboot until it is explicitly enabled with either `cmgui` or `cmsh`.

- To enable it in `cmgui`, the `Software Images` resource is selected, and the specific image item is selected. The `Settings` tabbed pane for that particular software image is opened, the new kernel version is selected from the "`Kernel version`" drop-down menu, and the `Save` button is clicked. Saving the version builds a new initial ramdisk containing the selected kernel (figure 9.1).

- To enable the updated kernel from `cmsh`, the `softwareimage` mode is used. The `kernelversion` property of a specified software image is then `set` and committed:

  Example

  ```
  [root@bright52 ~]# cmsh
  [bright52]% softwareimage
  [bright52]->softwareimage% use default-image
  [bright52->softwareimage[default-image]]% set kernelversion 2.6.32-
  131.2.1.el6.x86_64
  [bright52->softwareimage*[default-image*]]% commit
  ```

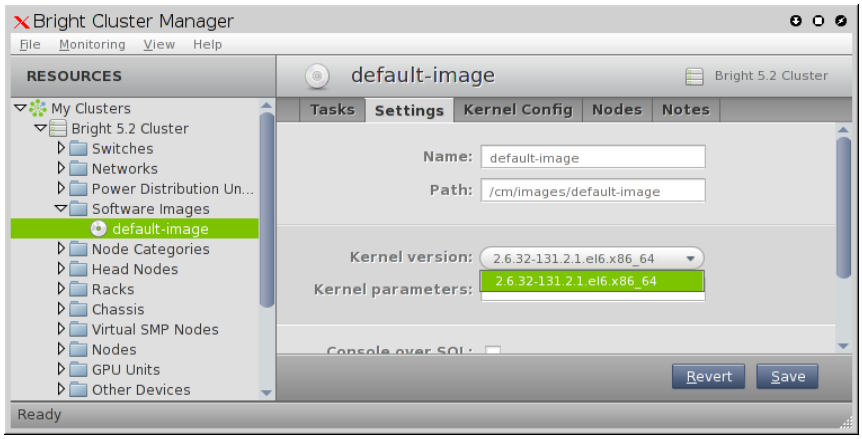

Figure 9.1: Updating A Software Image Kernel In `cmgui`

9.3.4 Setting Kernel Options For Software Images

A standard kernel can be booted with special options that alter its functionality. For example, a kernel can boot with `apm=off`, to disable Advanced Power Management, which is sometimes useful as a workaround for nodes with a buggy BIOS that may crash occasionally when it remains enabled.

To enable booting with this kernel option setting in `cmgui`, the "`Software Images`" resource is selected, and the specific image item is selected (figure 9.1). The `Settings` tabbed pane for that particular software image is opened, and the "`Kernel parameters`" value is set to `apm=off`.

In `cmsh` the value of "`kernel parameters`" in `softwareimage` mode for the selected image can be set as in the following example:

```
[root@bright52 ~]# cmsh
[bright52]% softwareimage
[bright52]->softwareimage% use default-image
[bright52->softwareimage[default-image]]% set kernelparameters apm=off
[bright52->softwareimage*[default-image*]]% commit
```

Often kernel options load up modules and their parameters. Making module loading persist after reboot and setting module loading order is covered in section 6.3.2

Some kernel options may require changes to be made in the BIOS settings in order to function.

9.3.5 Kernel Driver Modules

Bright Computing provides some packages which install new kernel drivers or update kernel drivers. Such packages generally require the `kernel-devel` package. In this section, the `kernel-devel-check` utility is first described, followed by the various drivers that Bright Computing provides.

Kernel Driver Modules: `kernel-devel-check` Compilation Check

The distribution's `kernel-devel` package is required to compile kernel drivers for its kernel. It must be the same version and release as the kernel running on the node.

To check the head node and software images for the installation status of the `kernel-devel` package, the Bright Cluster Manager utility

kernel-devel-check is run from the head node:

Example

```
[root@mycluster ~]# kernel-devel-check
Head node: mycluster
  Found kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.\
x86_64

Software image: default-image
  No kernel development directories found, probably no kernel developme\
nt package installed.
  Kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.x86_64\
not found
  If needed, try to install the kernel development package with:
  # chroot /cm/images/default-image yum install kernel-devel-2.6.32-131\
.2.1.el6.x86_64

Software image: default-image1
  No kernel development directories found, probably no kernel developme\
nt package installed.
  Kernel development rpm package kernel-devel-2.6.32-131.2.1.el6.x86_64\
 not found
  If needed, try to install the kernel development package with:
  # chroot /cm/images/default-image1 yum install kernel-devel-2.6.32-13\
1.2.1.el6.x86_64
```

As suggested by the output of kernel-devel-check, running a command on the head node such as:

```
[root@mycluster ~]# chroot /cm/images/default-image1 yum install kernel\
-devel-2.6.32-131.2.1.el6.x86_64
```

installs a kernel-devel package, to the software image called default-image1 in this case. The package version suggested corresponds to the kernel version set for the image, rather than necessarily the latest one that the distribution provides.

Kernel Driver Modules: Improved Intel Wired Ethernet Drivers

Improved Intel wired Ethernet drivers—what they are: The standard distributions provide Intel wired Ethernet driver modules as part of the kernel they provide. Bright Computing provides an improved version of the drivers with its own intel-wired-ethernet-drivers package. The package contains more recent versions of the Intel wired Ethernet kernel drivers: e1000, e1000e, igb, igbvf, ixgbe and ixgbevf. They often work better than standard distribution modules when it comes to performance, features, or stability.

Improved Intel wired Ethernet drivers—replacement mechanism: The improved drivers can be installed on all nodes.

For head nodes, the standard Intel wired Ethernet driver modules on the hard drive are overwritten by the improved versions during package installation. Backing up the standard driver modules before installation is recommended, because it may be that some particular hardware configurations are unable to cope with the changes, in which case reverting to the standard drivers may be needed.

For regular nodes, the standard distribution wired Ethernet drivers are not overwritten into the provisioner's software image during installation of the improved drivers package. Instead, the standard driver modules are removed from the kernel and the improved modules are loaded to the kernel during the `init` stage of boot.

For regular nodes in this "unwritten" state, removing the improved drivers package from the software image restores the state of the regular node, so that subsequent boots end up with a kernel running the standard distribution drivers from on the image once again. This is useful because it allows a very close-to-standard distribution to be maintained on the nodes, thus allowing better distribution support to be provided for the nodes.

If the software running on a fully-booted regular node is copied over to the software image, for example using the "`Grab to image`" button (section 9.5.2), this will write the improved driver module into the software image. Restoring to the standard version is then no longer possible with simply removing the improved drivers packages. This makes the image less close-to-standard, and distribution support is then less easily obtained for the node.

Thus, after the installation of the package is done on a head or regular node, for every boot from the next boot onwards, the standard distribution Intel wired Ethernet drivers are replaced by the improved versions for fully-booted kernels. This replacement occurs before the network and network services start. The head node simply boots from its drive with the new drivers, while a regular node initially starts with the kernel using the driver on the software image, but then if the driver differs from the improved one, the driver is unloaded and the improved one is compiled and loaded.

Improved Intel wired Ethernet drivers—installation: The drivers are compiled on the fly on the regular nodes, so a check should first be done that the `kernel-devel` package is installed on the regular nodes (section 9.3.5).

If the regular nodes have the `kernel-devel` package installed, then the following `yum` commands are issued on the head node, to install the package on the head node and in the `default-image`:

Example

```
[root@mycluster ~]# yum install intel-wired-ethernet-drivers
[root@mycluster ~]# chroot /cm/images/default-image
[root@mycluster /]# yum install intel-wired-ethernet-drivers
```

For SUSE, the equivalent `zypper` commands are used ("`zypper in`" instead of "`yum install`").

Kernel Driver Modules: CUDA Driver Installation

CUDA drivers are drivers the kernel uses to manage GPUs. These are compiled on the fly for nodes with GPUs in Bright Cluster Manager. The details of how this is done is covered in the CUDA software section (section 12.5).

Kernel Driver Modules: OFED Stack Installation

By default, the distribution provides the OFED stack used by the kernel to manage the InfiniBand or RDMA interconnect. Installing a Bright Cluster Manager repository OFED stack to replace the distribution version is covered in section 12.6. Some guidance on placement into initrd for the purpose of optional InfiniBand-based node provisioning is given in section 6.3.3.

9.4 Managing An RPM Package In A Software Image And Running It On Nodes

9.4.1 Installing From Head Via `chroot`: Installing The Image

Managing RPM packages (including the kernel) inside a software image is most easily done while on the head node, using a chroot mechanism with `rpm`, `yum`, or `zypper`.

The `rpm` command supports the `--root` flag. To install an RPM package inside the default software image while on the head node, the command is used as follows:

Example

```
rpm --root /cm/images/default-image -ivh /tmp/libxml2-2.6.16-6.x86_64.rpm
```

YUM and zypper implement the same functionality in slightly different ways. For example, all packages in the default image are updated with:

```
yum --installroot=/cm/images/default-image update
or
zypper --root /cm/images/default-image up
```

With the `chroot` command, the same result is accomplished by first chrooting into an image, and subsequently executing `rpm`, `yum`, or `zypper` commands without `--root` or `--installroot` arguments.

9.4.2 Installing From Head Via `chroot`: Updating The Image

Rebooting the nodes that use the software images then has those nodes start up with the new images. Alternatively, the nodes can usually simply be updated without a reboot (section 6.6), if no reboot is required by the underlying Linux distribution.

9.4.3 Installing From Head Via `rpm --root`, `yum --installroot` Or `chroot`: Possible Issues

- The `rpm --root` or `yum --installroot` command can fail if the versions between the head node and the version in the software image differ significantly. For example, installation from a Scientific Linux 5 head node to a Red Hat 6 software image is not possible with those commands, and can only be carried out with `chroot`.
- While installing software into a software image with an `rpm --root`, `yum --installroot` or with a `chroot` method is convenient, there can be issues if daemons start up in the image.

For example, installation scripts that stop and re-start a system service during a package installation may successfully start that service within the image's chroot jail and thereby cause related, unexpected changes in the image. Pre- and post- (un)install scriptlets that are part of RPM packages may cause similar problems.

Bright Computing's RPM packages are designed to install under chroot without issues. However packages from other repositories may cause the issues described. To deal with that, the cluster manager runs the `chrootprocess` health check, which alerts the administrator if there is a daemon process running in the image. The `chrootprocess` also checks and kills the process if it is a `crond`.

9.5 Managing Non-RPM Software In A Software Image And Running It On Nodes

Sometimes, packaged software is not available for a software image, but non-packaged software is. This section describes the installation of non-packaged software onto a software image in these two cases:

1. copying only the software over to the software image (section 9.5.1)
2. placing the software onto the node directly, configuring it until it is working as required, and syncing that back to the software image using Bright Cluster Manager's special utilities (section 9.5.2)

As a somewhat related aside, completely overhauling the software image, including changing the base files that distinguish the distribution and version of the image is also possible. How to manage that kind of extreme change is covered separately in section 9.6.

However, this current section (9.5) is about modifying the software image with non-RPM software while staying within the framework of an existing distribution and version.

In all cases of installing software to a software image, it is recommend that software components be placed under appropriate directories under `/cm/shared` (which is actually outside the software image).

So, just as in the case for installing software to the head node in section 9.2.2, appropriate software components go under:

```
/cm/shared/
|-- apps
|-- docs
|-- examples
|-- licenses
‘-- modulefiles
```

9.5.1 Managing The Software Directly On An Image

The administrator may choose to manage the non-packaged software directly in the correct location on the image.

For example, the administrator may wish to install a particular software to all nodes. If the software has already been prepared elsewhere and is known to work on the nodes without problems, such as for example library dependency or path problems, then the required files can simply be copied directly into the right places on the software image.

The chroot command may also be used to install non-packaged software into a software image. This is analogous to the chroot technique for installing packages in section 9.4:

Example

```
cd /cm/images/default-image/usr/src
tar -xvzf /tmp/app-4.5.6.tar.gz
chroot /cm/images/default-image
cd /usr/src/app-4.5.6
./configure --prefix=/usr
make install
```

Whatever method is used to install the software, after it is placed in the software image, the change can be implemented on all running nodes by running the updateprovisioners (section 6.2.4) and imageupdate (section 6.6.2) commands.

9.5.2 Managing The Software Directly On A Node, Then Syncing Node-To-Image

Why Sync Node-To-Image?

Sometimes, typically if the software to be managed is more complex and needs more care and testing than might be the case in section 9.5.1, the administrator manages it directly on a node itself, and then makes an updated image from the node after it is configured, to the provisioner.

For example, the administrator may wish to install and test an application from a node first before placing it in the image. Many files may be altered during installation in order to make the node work with the application. Eventually when the node is in a satisfactory state a new image can be created.

Administrators should be aware that until the new image is saved, the node loses its alterations and reverts back to the old image on reboot.

The node-to-image sync discussed in this section is called a "Grab to image" in cmgui or "grabimage" in cmsh, and is the converse of the image-to-node sync, imageupdate of section 6.6.2.

Node-To-Image Sync Using cmgui

In cmgui, saving the node state to a new image is done by selecting a node from the Nodes resource, and then selecting the Tasks tab. The "Software image" section of the tab has (amongst others) these two buttons that can carry out the sync from node to software image (figure 9.2):

1. the "Grab to image" button, which opens up a dialog offering an image to sync to. It creates a new image, wiping out whatever (if anything) is in the selected image. It uses the "Exclude list grabbing to a new image" exclude list, available under the "Node Categories" resource, under the Settings tab.

2. the "Synchronize image" button, which does a sync preserving system settings that are associated with image identity, and evaluates the remaining changes between the node and the selected image. It is thus a synchronization to an existing image. It uses the "Exclude list image grab" exclusion list, also available under the "Node Categories" resource, under the Settings tab.

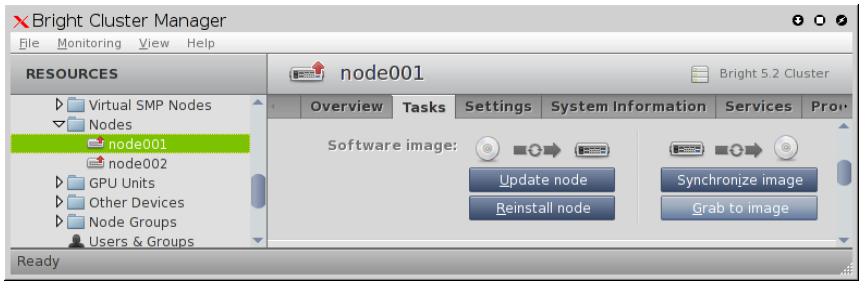

Figure 9.2: Synchronizing From A Node To A Software Image In `cmgui`

While both these exclude lists are simple by default, they conform in structure and patterns syntax in the same way that the exclude lists detailed in section 6.4.7 do and can therefore be quite powerful.

Both buttons open up a dialog allowing selection of the image to be used for syncing. In the dialog, a `dry-run` checkbox is marked by default to allow the administrator to see what would happen during the node-to-image sync, without actually having the files written over to the image. Logs of the run are viewable by clicking on the "`Provisioning Log`" button, also in the `Tasks` tab. If the administrator is sure that the run works as wanted, the node-to-image sync can be done with the `dry-run` checkbox unmarked.

The images that are available for selection with these buttons are existing images. If such existing images are known to work well with nodes, then overwriting them with a new image on a production system may be reckless. A wise administrator who has prepared a node that is to write an image would therefore follow a process similar to the following instead of simply overwriting an existing image:

1. A new image is created into which the node state can be written. This is easiest to do by using the "`Software Images`" resource to clone a new image. The node state with the software installed on it would then be saved using the "`Grab to image`" or "`Synchronize image`" buttons, and choosing the cloned image name as the image to save it to.

2. A new category is then cloned from the old category, and within the new category the old image is changed to the new image. This is easiest to do from the `Overview` tab of the "`Node Categories`" resource. Here the original category of the node is selected and cloned. The cloned category is then selected, opened and the old image within it—in the "`Software image`" section—is changed to the new, cloned, image, and the result is saved.

3. Some nodes are set to the new category to test the new image. This is done from the `Nodes` resource, selecting the node, and from its `Settings` tab, changing its category to the new one, and saving the change.

4. The nodes in the new category are made to pick up and run their new images. This can be done with a reboot.

5. After sufficient testing, all remaining nodes can be moved to using the new image, and the old image is removed if no longer needed.

Node-To-Image Sync Using cmsh

The preceding cmgui method can alternatively be carried out using cmsh commands. The cmsh equivalent to the "Grab to image" or "Synchronize image" buttons is the grabimage command, available from device mode. The grabimage command requires the -i option to specify what image it will write to. As before, that image must be created or cloned beforehand.

Cloning an image for use, setting a category of nodes that will use it, and then synchronizing a node that has the new software setup over to the new image on the provisioner might be carried out as follows via cmsh:

```
[root@bright52 ~]# cmsh
[bright52]% softwareimage
[bright52->softwareimage]% clone default-image default-image1
[bright52->softwareimage*[default-image1]]% commit
[bright52->softwareimage[default-image1]]% category
[bright52->category]% clone default default1
[bright52->category*[default1*]]% commit
[bright52->category[default1]]% set softwareimage default-image1
[bright52->category*[default1*]]% commit
[bright52->category[default1]]% device
[bright52->device]% grabimage -w -i default-image1 node001
[bright52->device]%
Mon Jul 18 16:13:00 2011 [notice] bright52: Provisioning started on\
 node node001
[bright52->device]%
Mon Jul 18 16:13:04 2011 [notice] bright52: Provisioning completed on\
 node node001
```

The grabimage command has some options. The -i option selects the image, while the -w option instructs CMDaemon to really write the image. Leaving the -w option out simply does a dry-run so that the user can see in the provisioning logs what should be grabbed, without having the changes actually carried out.

9.6 Creating A Custom Software Image

By default, the software image used to boot non-head nodes is based on the same version and release of the Linux distribution as used by the head node. However, sometimes an image based on a different distribution or a different release from that on the head node may be needed.

A custom software image is created typically by building an entire filesystem image from a node. The node is then called the *base host*, with the term "base" used to indicate that it has no additional cluster manager packages installed. The distribution on the base host, is called the *base distribution* and is a selection of packages derived from the *parent distribution* (Red Hat, Scientific Linux etc). A *base distribution package* is a package or rpm that is directly provided by the vendor of the parent distribution which the base distribution is based on, and is not provided by Bright Cluster Manager.

Creating a custom software image consists of two steps. The first step is to create a *base (distribution) archive* from an installed base host. The

second step is to create the image from the base archive using a special utility, cm-create-image.

9.6.1 Creating A Base Distribution Archive From A Base Host

The step of creating the base distribution archive is done by creating an archive structure containing the files that are needed by the non-head node. The archive can be a convenient and standard tar.gz file archive, or, taking the step a little further towards the end result, the archive can be a fully expanded archive file tree.

In the following example, a base distribution tar.gz archive /tmp/BASEDIST.tar.gz is created from the base host basehost64:

Example

```
ssh root@basehost64 \
"tar -cz \
--exclude /etc/HOSTNAME --exclude /etc/localtime \
--exclude /proc --exclude /lost+found --exclude /sys \
--exclude /root/.ssh --exclude /var/lib/dhcpcd/* \
--exclude /media/floppy --exclude /etc/motd \
--exclude /root/.bash_history --exclude /root/CHANGES \
--exclude /etc/udev/rules.d/*persistent*.rules \
--exclude /var/spool/mail/* --exclude /rhn \
--exclude /etc/sysconfig/rhn/systemid \
--exclude /var/spool/up2date/* \
--exclude /etc/sysconfig/rhn/systemid.save \
--exclude /root/mbox --exclude /var/cache/yum/* \
--exclude /etc/cron.daily/rhn-updates /" > /tmp/BASEDIST.tar.gz
```

Or alternatively, a fully expanded archive file tree can be created from basehost64 by rsyncing to an existing directory (here it is /cm/image/new-image):

Example

```
rsync -av --numeric-ids \
--exclude='/etc/HOSTNAME' --exclude='/etc/localtime' --exclude='/proc'\
--exclude='/lost+found' --exclude='/sys' --exclude='/root/.ssh' \
--exclude='/var/lib/dhcpcd/*' --exclude='/media/floppy' \
--exclude='/etc/motd' --exclude='/root/.bash_history' \
--exclude='/root/CHANGES' --exclude='/var/spool/mail/*'\
--exclude='/etc/udev/rules.d/*persistent*.rules' \
--exclude='/rhn' --exclude='/etc/sysconfig/rhn/systemid' \
--exclude='/etc/sysconfig/rhn/systemid.save'\
--exclude='/var/spool/up2date/*' \
--exclude='/root/mbox' --exclude='/var/cache/yum/*' \
--exclude='/etc/cron.daily/rhn-updates' \
root@basehost64:/ /cm/image/new-image/
```

The first step, that of building the base archive, is now done.

9.6.2 Creating The Software Image With cm-create-image

The second step, that of creating the image from the base archive, now needs to be done. This uses the cm-create-image utility, which is part of the cluster-tools package. The utility requires that base distribution repositories be accessible.

Repository access can be directly to the online repositories provided by the distribution, or it can be to a local copy. For RHEL, online repository access can be activated by registration with the Red Hat Network, as described in appendix M.1, and for SUSE, online repository access can be activated by registration with Novell, as described in appendix M.2. An offline repository can be constructed as described in section 9.6.3.

Usage Of The `cm-create-image` Command

The usage information for `cm-create-image` lists options and examples:

```
USAGE: cm-create-image <OPTIONS1> [OPTIONS2]

OPTIONS1:
---------
-a | --fromarchive <archive> Create software image from archive file
                             of supported base distribution. Supported
                             file formats are .tar, .tgz, .tar.gz, .tbz,
                             and .tar.bz2. The extension must match the
                             format.
-d | --fromdir   <dir path>  Create software image from existing
                             directory that already has valid base
                             distribution.
-h | --fromhost  <hostname>  Create software image from running host
-n | --imagename <name>      Name of software image to create in cluster
                             management daemon database.

OPTIONS2:
---------
-i | --imagedir <dir name>   Name of directory to be created in
                             /cm/images.
                             Contents of archive file are extracted into
                             this directory (default: name specified
                             with -n).
-r | --recreate              Recreate directory specified by -i or
                             default, if it exists.
                             Default behavior: directory is overwritten
-s | --skipdist              Skip install of base distribution packages
-e | --excludecmrepo         Do not copy cluster manager repo files. (Use
                             when the repo files already exist, and must
                             not be overwritten.)
-f | --forcecreate           Force non-interactive mode
-u | --updateimage           If image specified by -n already exists,
                             then it is updated, with the new parameters

EXAMPLES:
---------
1. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image
2. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/images/te\
st-image
3. cm-create-image -d /cm/images/SLES11-image -n sles11-image
4. cm-create-image -h node001 -n node001-image
5. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/image/new\
-image -r
6. cm-create-image -a /tmp/RHEL5.tar.gz -n rhel5-image -i /cm/image/new\
-image -u
7. cm-create-image -d /cm/image/new-image -n bio-image -s
```

Explanations Of The Examples In Usage Of `cm-create-image`

Explanations of the 7 examples in the usage text follow:

1. In the following, a base distribution archive file, `/tmp/RHEL5.tar.gz`, is written out to a software image named `rhel5-image`:

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image
   ```

   The image with the name `rhel5-image` is created in the CMDaemon database, making it available for use by `cmsh` and `cmgui`. If an image with the above name already exists, then `/cm/create-image` will exit and advise the administrator to provide an alternate name.

   By default, the image name specified sets the directory into which the software image is installed. Thus here the directory is `/cm/image/rhel5-image/`.

2. Instead of the image getting written into the default directory as in the previous item, an alternative directory can be specified with the `--imagedir` option. Thus, in the following, the base distribution archive file, `/tmp/RHEL5.tar.gz` is written out to the `/cm/image/test-image` directory. The software image is given the name `rhel5-image`:

   ```
   cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
   image --imagedir /cm/image/test-image
   ```

3. If the contents of the base distribution file tree have been transferred to a directory, then no extraction is needed. The `--fromdir` option can then be used with that directory. Thus, in the following, the archive has already been transferred to the directory `/cm/image/SLES11-image`, and it is that directory which is then used to place the image under a directory named `/cm/image/sles11-image/`. Also, the software image is given the name `sles11-image`:

   ```
   cm-create-image --fromdir /cm/images/SLES11-image --imagename sles\
   11-image
   ```

4. A software image can be created from a running node using the `--fromnode` option. This option makes `cm-create-image` behave in a similar manner to `grabimage` (section 9.5.2) in `cmsh`. It requires passwordless access to the node in order to work. An image named `node001-image` can then be created from a running node named `node001` as follows:

   ```
   cm-create-image --fromhost node001 --imagename node001-image
   ```

   By default the image goes under the `/cm/image/node001-image/` directory.

5. If the destination directory already exists, the `--recreate` option can be used to recreate the existing directory. The administrator should be aware that this means removal of the content of any existing directory of the same name. Thus, in the following, the content under `/cm/image/new-image/` is deleted, and new image content is taken from the base distribution archive file, `/tmp/RHEL5.tar.gz` and then placed under `/cm/image/new-image/`. Also, the software image is given the name `rhel5-image`:

```
cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
image --imagedir /cm/image/new-image --recreate
```

   If the `--recreate`, option is not used, then the contents are simply overwritten, that is, the existing directory contents are copied over by the source content. It also means that old files on the destination directly may survive unchanged because the new source may not have filenames matching those.

6. The destination directory can also just be updated without removing the existing contents, by using the option `--updateimage`. The option is almost the same as the "contents are simply overwritten" behavior described in example 5, but it actually works like an `rsync` command. Thus, in the following, the base distribution archive file, `/tmp/RHEL5.tar.gz`, is used to update the contents under the directory `/cm/image/new-image/`. The name of the image is also set to `rhel5-image`.

```
cm-create-image --fromarchive /tmp/RHEL5.tar.gz --imagename rhel5-\
image --imagedir /cm/image/new-image --updateimage
```

7. With the default Bright Cluster, the head node provisions a software image based on the parent distribution to the other nodes. The software image which runs on the nodes provides a selection of distribution packages from the parent distribution.

   The default software image is thus a selection of Red Hat packages, if the head node uses Red Hat, or a selection of SUSE packages if the head node uses SUSE, and so on. The other packages for the software image are supplied by Bright Computing.

   When creating a custom software image, and if using the `--skipdist` flag with `cm-create-image`, then Bright Cluster Manager packages are added to the software image, but no parent distribution packages are added. Thus in the following, the packages made available to `cm-create-image` in the directory `/cm/image/new-image`, are installed into the image named `bio-image`; however, no packages matching parent distribution packages are installed (because of the `--skipdist` option). Furthermore, transfer of the packages takes place only if they are newer than the files already in the `bio-image` image (because of the `--updateimage` option):

```
cm-create-image --fromdir /cm/image/new-image --imagename bio-imag\
e --skipdist --updateimage
```

So, only Bright Cluster Manager packages are updated to the image `bio-image` in the directory `/cm/image/bio-image`.

Package Selection Files In `cm-create-image`

Regarding explanation 7 in the preceding explanations text, the selection of packages on the head node is done using a *package selection file*.

Package selection files are available in `/cm/local/apps/cluster-tools/config/`. For example, if the base distribution of the software image being created is CentOS5, then the config file used is:

```
/cm/local/apps/cluster-tools/config/CENTOS5-config-dist.xml
```

The package selection file is made up of a list of XML elements, specifying the name of the package, architecture and image type. For example:

```
...
<package image="slave" name="apr" arch="x86_64"/>
<package image="slave" name="apr-util" arch="x86_64"/>
<package image="slave" name="atk-devel" arch="x86_64"/>
<package image="slave" name="autoconf" arch="noarch"/>
...
```

Additional packages to be installed in the image can be specified in the package selection file.

The package selection file also contains entries for the packages that can be installed on the head (`image="master"`) node. Therefore non-head node packages must have the `image="slave"` attribute.

Output And Logging During A `cm-create-image` Run

The `cm-create-image` run goes through several stages: validation, sanity checks, finalizing the base distribution, copying Bright Cluster Manager repository files, installing distribution package, finalizing image services, and installing Bright Cluster Manager packages. An indication is given if any of these stages fail.

Further detail is available in the logs of the `cm-create-image` run, which are kept in `/var/log/cmcreateimage.log.`*`<image name>`*, where *<image name>* is the name of the built image.

Default Image Location

The default-image is at `/cm/images/default-image`, so the image directory can simply be kept as `/cm/images/`.

During a `cm-create-image` run, the `--imagedir` option allows an image directory for the image to be specified. This must exist before the option is used.

More generally, the full path for each image can be set:

- In `cmgui` in the "`Software Images`" resource, by filling in the box for `Path` in the `Settings` tabbed pane for the image
- In `cmsh` within `softwareimage` mode, for example:

```
[bright60->softwareimage]% set new-image path /cm/higgs/new-images
```

- At the system level, the images or image directory can be symlinked to other locations for organizational convenience

9.6.3 Configuring Local Repositories For Linux Distributions

Having local instead of remote repositories for the cluster can be useful in the following cases:

- for clusters that have restricted or no internet access.
- for the RHEL and SUSE Linux distributions, which are based on a subscription and support model, and therefore do not have free access to their repositories.

The `cm-create-image` command introduced in section 9.6.2 requires access to the base distribution repositories.

The administrator can choose to access an online repository provided by the distribution itself via a subscription as described in appendix M. Another way to set up a repository is to set it up as a locally controlled repository, which may be offline, or perhaps set up as a locally controlled proxy with occasional, restricted, updates from the distribution repository.

In the three procedures that follow, the first two procedures explain how to create and configure a locally controlled offline zypper or zypper repository for the subscription-based base distribution packages. These first two procedures assume that the corresponding ISO/DVD has been purchased/downloaded from the appropriate vendors. The third procedure then explains how to create and configure a locally controlled offline YUM repository from a freely accessible CentOS repository and combine it with a local Bright repository so that a cluster that is completely offline still has a complete and consistent repository access.

Configuring Repositories For SLES

For SLES11 SP0 and SLES11 SP1, the required packages are spread across two DVDs, and hence two repositories must be created. The image directory is assumed to be `/cm/images/sles11sp1-image`, while the names of the DVDs are assumed to be `SLES-11-SP1-SDK-DVD-x86_64-GM-DVD1.iso` and `SLES-11-SP1-DVD-x86_64-GM-DVD1.iso`. The contents of the DVDs are copied as follows:

```
mkdir /mnt1 /mnt2
mkdir /cm/images/sles11sp1-image/root/repo1
mkdir /cm/images/sles11sp1-image/root/repo2
mount -o loop,ro SLES-11-SP1-SDK-DVD-x86_64-GM-DVD1.iso /mnt1
cp -ar /mnt1/* /cm/images/sles11sp1-image/root/repo1/
mount -o loop,ro SLES-11-SP1-DVD-x86_64-GM-DVD1.iso  /mnt2
cp -ar /mnt2/* /cm/images/sles11sp1-image/root/repo2/
```

The two repositories are added for use by `zypper`, as follows:

```
chroot /cm/images/sles11sp1-image
zypper addrepo /root/repo1 "SLES11SP1-SDK"
zypper addrepo /root/repo2 "SLES11SP1"
exit (chroot)
```

Configuring Repositories For RHEL-Based Distributions

For RHEL-based distributions, the procedure is almost the same. The required packages are contained in one DVD.

```
mkdir /mnt1
mkdir /cm/images/rhel-image/root/repo1
mount -o loop,ro RHEL-DVD1.iso /mnt1
cp -ar /mnt1/* /cm/images/rhel-image/root/repo1/
```

The repository is added to YUM by creating the repository file `/cm/images/rhel-image/etc/yum.repos.d/rhel-base.repo` with the contents:

```
[base]
name=Red Hat Enterprise Linux $releasever - $basearch - Base
baseurl=file:///root/repo1/Server
gpgcheck=0
enabled=1
```

Configuring CentOS And Bright Repositories For Local Access

An image directory can be assigned to `$imagedir` as follows:

```
imagedir=/cm/images/centosimage
```

and repository directories can then be created under it as follows:

```
mkdir $imagedir/root/bright-repo
mkdir $imagedir/root/centos5-extra
```

The `centos5-extra` repository contains a small number of extra packages, beyond the basic CentOS distribution, that are needed to make Bright able to work with CentOS. If the Bright DVD based on release version 5.2-29 is called `bright-centos5.iso`, then repository contents based on it can be created like this:

```
mkdir /mnt1
mount -o loop bright-centos5.iso /mnt1

cp -a /mnt1/data/cm-rpms/5.2-29/centos/5/* $imagedir/root/bright-repo/
chroot $imagedir createrepo /root/bright-repo
```

Similarly for the CentOS extra distribution, repository contents based on it can be created like this:

```
cp -a /mnt1/data/cm-rpms/dist/centos/5/extra/* $imagedir/root/centos5-ex\
tra/
chroot $imagedir createrepo /root/centos5-extra
```

The repository configuration files for the two repositories can then be written as:

```
cat >$imagedir/etc/yum.repos.d/cm.repo <<EOF
[bright-repo]
name=Bright Cluster Manager 5.2 Repo
baseurl=file:///root/bright-repo
enabled=1
gpgcheck=0
EOF
```

```
cat >$imagedir/etc/yum.repos.d/centos5-extra.repo <<EOF
[centos5-extra]
name=Centos5 Dist Repo Extra - Bright DVD
baseurl=file:///root/centos5-extra
enabled=1
gpgcheck=0
EOF
```

A central CentOS mirror can be enabled to allow additional distribution packages to be pulled in with YUM, to satisfy certain dependencies for Bright packages. A CentOS 5 repository configuration file can be created as follows:

```
cat  >$imagedir/etc/yum.repos.d/centos5.repo <<EOF
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basea\
rch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basea\
rch&repo=updates
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5

#additional packages that may be useful
[extras]
name=CentOS-$releasever - Extras
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$base\
arch&repo=extras
#baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5
EOF
```

The software image can now be created in the directory called, for example, `offlineimage`:

```
cm-create-image -d $imagedir -n offlineimage -e -s
```

The `-e` option prevents copying the default cluster manager `cm.repo` file. The `-s` option prevents installing additional distribution packages that might not be required.

9.7 Making All Nodes Function Differently From Normal Cluster Behavior With `FrozenFile`

Configuration changes carried out by `cmgui` or `cmsh` often generate, restore, or modify configuration files (Appendix A).

However, sometimes an administrator may need to make a direct change (without using `cmgui` or `cmsh`) to a configuration file to set up a special configuration that cannot otherwise be done.

The `FrozenFile` directive to CMDaemon (Appendix C) applied to such a configuration file stops CMDaemon from altering the file. The frozen configuration file is generally applicable to all nodes and is therefore a possible way of making all nodes function differently from their standard behavior.

Freezing files is however best avoided, if possible, in favor of a CMDaemon-based method of configuring nodes, for the sake of administrative maintainability.

9.8 Making Some Nodes Function Differently By Image

For minor changes, adjustments can often be made to node settings so that nodes or node categories function differently. This is covered in section 9.9.

For major changes, it is usually more appropriate to have nodes function differently from each other by simply setting different images per node category with CMDaemon.

This can be done as follows with `cmsh`:

1. The image on which the new one will be based is cloned. The cloning operation not only copies all the settings of the original (apart from the name), but also the data of the image:

   Example

   ```
   [root@bright52 ~]# cmsh
   [bright52]% softwareimage
   [bright52->softwareimage]% clone default-image imagetwo
   [bright52->softwareimage*[imagetwo*]]% commit
   Thu Aug 11 15:44:44 2011 [notice] bright52: Started to copy: /cm/\
   images/default-image -> /cm/images/imagetwo
   [bright52->softwareimage*[imagetwo*]]%
   Thu Aug 11 15:53:22 2011 [notice] bright52: Copied: /cm/images/de\
   fault-image -> /cm/images/imagetwo
   [bright52->softwareimage[imagetwo]]%
   ```

2. After cloning, the settings can be modified in the new object. For example, if the kernel needs to be changed to suit nodes with different hardware, kernel modules settings are changed (section 6.3.2) and committed. This creates a new image with a new ramdisk.

   Other ways of modifying and committing the image for the nodes are also possible, as discussed in sections 9.2–9.6 of this chapter.

3. The modified image that is to be used by the differently functioning nodes is placed in a new category in order to have the nodes be able to choose the image. To create a new category easily, it can simply be cloned. The image that the category uses is then set:

```
[bright52->softwareimage[imagetwo]]% category
[bright52->category]% clone default categorytwo
[bright52->category*[categorytwo*]]% set softwareimage imagetwo
[bright52->category*[categorytwo*]]% commit
[bright52->category[categorytwo]]%
```

4. - For just one node, or a few nodes, the node can be set from `device` mode to the new category (which has the new image):

     ```
     [bright52->category[categorytwo]]% device
     [bright52->device]% use node099
     [bright52->device[node099]]% set category categorytwo
     [bright52->device*[node099*]]% commit; exit
     ```

   - If there are many nodes, for example node100 sequentially up to node200, they can be set to that category using a `foreach` loop like this:

     Example

     ```
     [bright52->device]% foreach -n node100..node200 (set category\
      categorytwo)
     [bright52->device*]% commit
     ```

5. Rebooting restarts the nodes that are assigned to the new category with the new image.

Similarly, using `cmgui`, a default image can be cloned with the "`Clone`" button in the "`Software Images`" resource operating on an existing image. The new image is modified from the "`Software Images`" tab (section 9.3.4) or using the other image modifying methods in this chapter, to make nodes function in the way required, and any appropriate node or node category is assigned this image.

9.9 Making Some Nodes Function Differently By Configuration Adjustment

9.9.1 Configuration Adjustment: The Bigger Picture

From an administrative perspective, sections 9.2–9.6 of this chapter have covered managing (installing, maintaining, removing) software on nodes as a technique, while section 9.7 has covered using the `FrozenFile` directive as a way to override CMDaemon control of configuration files and is generally applicable to all nodes. How to make some nodes behave differently in the context of setting different images on some nodes is discussed in section 9.8.

How to make some nodes behave differently in the context of adjusting configuration settings on nodes instead of varying software images per node is discussed in the current section 9.9.

Since the subject of this chapter is about the post-installation management of software images, the current section is, strictly speaking, outside the scope of the chapter. It is nonetheless included here for completeness, because an administrator should consider configuration adjustment methods as a possibly more suitable alternative to setting different images on nodes.

9.9.2 Configuration Adjustment: When To Use It

Configuration adjustment is more appropriate for less complex changes and also avoids having to manage images.

For example, it is more appropriate to adjust nodes rather than install entire images on them when there are minor changes per node. The approach of installing software images would require a new image for each change, and tends to lead to a large number of images. Managing large numbers of images and their associated software can be a chore—each different image requires tracking from the administrator for all future changes. Adjustment techniques on the other hand may be able to achieve the same behavior as that achieved by the installation of different software images onto nodes, and, with no new image required for the change in behavior, less administrative labor is required when changes are needed in the future.

Therefore, adjustment techniques should be considered if the aim is to have nodes function differently without image changes, and usually with minor changes involved.

9.9.3 Configuration Adjustment: Techniques

The following adjustment techniques are considered next:

- node configuration adjustments using `cmgui`/`cmsh` (section 9.9.4)
- adding functionality via a `finalize` scripts (section 9.9.5)
- adding functionality by mounting a shared directory containing the software (section 9.9.6)

9.9.4 Configuring Nodes Using `cmgui` Or `cmsh`

A node or node category can often have its software configured in CMDaemon via `cmgui` or `cmsh`:

Example

Configuring a software for nodes using `cmgui` or `cmsh`: If the software under consideration is CUPS, then a node or node category can manage it from the `Services` tab with `cmgui` or `cmsh` as outlined in section 4.8.2.

A (counter-) example to the method discussed in this section can be considered here to make the difference clearer. Instead of configuring the software via CMDaemon, it could instead first be configured as required in a software image. It could then be loaded by the node category directly as done in section 9.9. The example analogous to the preceding one would then be:

Example

Configuring a software for nodes without using `cmgui` or `cmsh`[1]: Software images can be created with and without CUPS configured. Setting up nodes to load one of these two images via a node category is an alternative way of letting nodes run CUPS.

[1] except to link nodes to their appropriate image via the associated category

Whether node configuration for a particular functionality is done with CMDaemon, or directly with the software, depends on what an administrator prefers. In the preceding two examples, the first example with `cmgui` or `cmsh` setting the CUPS service is likely to be preferred over the second example where an entire separate image must be maintained. A new category must also be created in the second case.

Generally, sometimes configuring the node via Bright Cluster Manager, and not having to manage images is better, sometimes configuring the software and making various images to be managed out of it is better, and sometimes only one of these techniques is possible anyway.

Configuring Nodes Using `cmgui` Or `cmsh`: Category Settings

When configuring nodes using `cmgui` or `cmsh`, configuring particular nodes from a node category to override the state of the rest of its category (as in section 3.1.3) is sensible for a small number of nodes. For larger numbers it may not be organizationally practical to do this, and another category can instead be created to handle nodes with the changes conveniently.

The CUPS service in the next two examples is carried out by implementing the changes via `cmgui` or `cmsh` acting on CMDaemon.

Example

Setting a few nodes in a category: If only a few nodes in a category are to run CUPS, then it can be done by those few nodes having the CUPS service enabled in the `Nodes` resource, thereby overriding (section 3.1.3) the category settings.

Example

Setting many nodes to a category: If there are many nodes that are to be set to run CUPS, then a separate, new category can be created (cloning it from the existing one is easiest) and those many nodes are moved into that category, while the image is kept unchanged. The CUPS service setting is then set at category level to the appropriate value for the new category.

In contrast to these two examples, the software image method used in section 9.8 to implement a functionality such as CUPS would load up CUPS as configured in an image, and would not handle it via CMDaemon[1]. So, in section 9.8, software images prepared by the administrator are set for a node category. Since, by design, images are only selected for a category, a node cannot override the image used by the category other than by creating a new category, and using it with the new image. The administrative overhead of this can be inconvenient.

Administrators would therefore normally prefer letting CMDaemon track software functionality across nodes as in the last two examples, rather than having to deal with tracking software images manually.

9.9.5 Adding Functionality To Nodes Via A Finalize Script

CMDaemon can normally be used to allocate different images per node or node category as explained in section 9.8. However, some configuration files do not survive a reboot (Appendix A), sometimes hardware

issues can prevent a consistent end configuration, and sometimes drivers need to be initialized before provisioning of an image can happen. In such cases, a `finalize` script (Appendix E.6) can be used to initialize or configure nodes or node categories.

A `finalize` script is also useful because it can be used to implement minor changes across nodes.

Example

Supposing that some nodes with a particular network interface have a problem autonegotiating their network speed, and default to 100Mbps instead of the maximum speed of 1000Mbps. Such nodes can be set to ignore autonegotiation and be forced to use the 1000Mbps speed by using the `ETHTOOL_OPTS` configuration parameter in their network interface configuration file: `/etc/sysconfig/network-scripts/ifcfg-eth0` (or `/etc/sysconfig/network/ifcfg-eth0` in SUSE).

The `ETHTOOL_OPTS` parameter takes the options to the "`ethtool -s` *<device>*" command as options. The value of *<device>* (for example `eth0`) is specified by the filename that is used by the configuration file itself (for example `/etc/sysconfig/network-scripts/ifcfg-eth0`). The `ethtool` package is installed by default on Bright Clusters. Running the command:

```
ethtool -s autoneg off speed 1000 duplex full
```

turns out after some testing to be enough to reliably get the network card up and running at 1000Mbps on the problem hardware.

However, since the network configuration file is overwritten by node-installer settings during reboot, a way to bring persistence to the file setting is needed. One way to ensure persistence is to append the configuration setting to the file with a `finalize` script, so that it gets tagged onto the end of the configuration setting that the node-installer places for the file, just before the network interfaces are taken down again in preparation for `init`.

The script may thus look something like this for a Red Hat system:

```
#!/bin/bash

## node010..node014 get forced to 1000 duplex
if [[ $CMD_HOSTNAME = node01[0-4] ]]
then
echo 'ETHTOOL_OPTS="speed 1000 duplex full"'>>/localdisk/etc/sysconfig/\
network-scripts/ifcfg-eth0
fi
```

9.9.6 Adding Functionality To Nodes By Mounting A Shared Directory

Software that is on a shared directory can be run on a node or node category. Section 4.7 describes how to share a directory, thereby giving the node or node category the added functionality of that software.

10

Cluster Monitoring

The Bright Cluster Manager monitoring framework lets a cluster administrator:

- inspect monitoring data to the required level for existing resources;
- configure gathering of monitoring data for new resources;
- see current and past problems or abnormal behavior;
- notice trends that help the administrator predict likely future problems;
- handle current and likely future problems by
  - triggering alerts;
  - taking action if necessary to try to improve the situation or to investigate further.

Powerful features are accessible within an intuitive monitoring framework, and customized complex setups can be constructed to suit the requirements of the administrator.

In this chapter, the monitoring framework is explained with the following approach:

1. A basic example is first presented in which processes are run on a node. These processes are monitored, and are acted on when a threshold is exceeded.
2. With this easy-to-understand example as the base, the various features and associated functionality of the Bright Cluster Manager monitoring framework are described and discussed in depth. These include visualization of data, concepts, configuration, monitoring customization and `cmsh` use.

10.1 A Basic Example Of How Monitoring Works

In this section, a minimal basic example of monitoring a process is set up. The aim is to present a simple overview that covers a part of what the monitoring framework is capable of handling. The overview gives the reader a structure to keep in mind, around which further details are fitted and filled in during the coverage in the rest of this chapter.

In the example, a user runs a large number of pointless CPU-intensive processes on a head node which is normally very lightly loaded. An administrator would then want to monitor user mode CPU load usage, and stop such processes automatically when a high load is detected (figure 10.1).

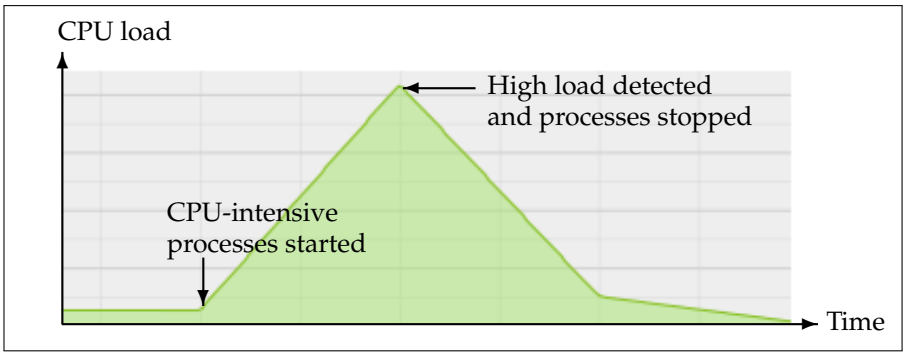

Figure 10.1: Monitoring Basic Example: CPU-intensive Processes Started, Detected And Stopped

The basic example illustrates a (very contrived) way for the Bright Cluster Manager monitoring framework to be used to do that.

10.1.1 Before Using The Framework—Setting Up The Pieces

Running A Large Number Of Pointless CPU-Intensive Processes

One way to simulate a user running pointless CPU-intensive processes is to run several instances of the standard unix utility, `yes`. The `yes` command sends out an endless number of lines of "y" texts. It is usually used to answer prompts for confirmation.

8 subshell processes are run in the background from the command line on the head node, with `yes` output sent to `/dev/null` as follows:

```
for i in {1..8}; do ( yes > /dev/null &); done
```

Running "`mpstat 2`" shows usage statistics for each processor, updating every 2 seconds. It shows that `%user`, which is user mode CPU usage, and which is reported as `CPUUser` in the Bright Cluster Manager metrics, is close to 100% on an 8-core or less head node when the 8 subshell processes are running.

Setting Up The Kill Action

To stop the pointless CPU-intensive `yes` processes, the command "killall yes" is used. It is made a part of a script `killallyes`:

```
#!/bin/bash
killall yes
```

and made executable with a `chmod 700 killallyes`. For convenience, it may be placed in the `/cm/local/apps/cmd/scripts/actions` directory where other action scripts also reside.

10.1.2 Using The Framework

Now that the pieces are in place, `cmgui`'s monitoring framework is used to add the action to its action list, and then set up a threshold level that triggers the action:

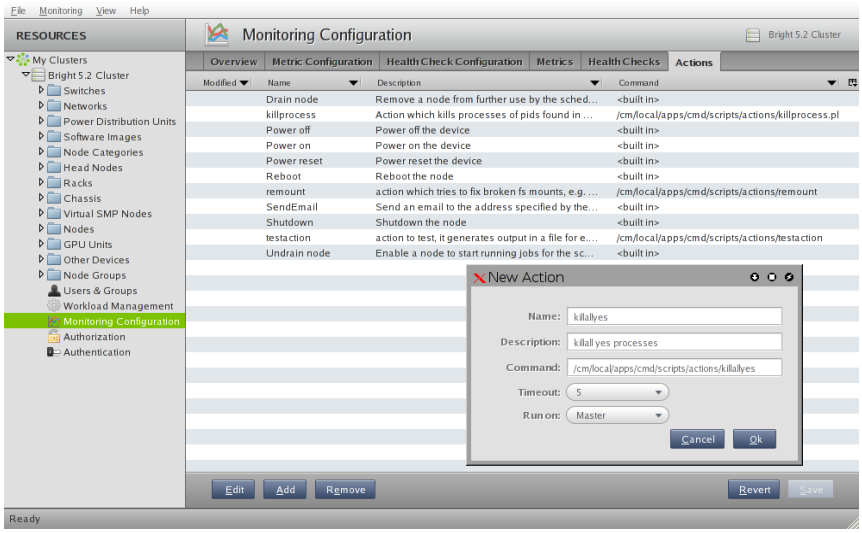

Figure 10.2: `cmgui` Monitoring Configuration: Adding An Action

Adding The Action To The Actions List

From the resources tree of `cmgui`, `Monitoring Configuration` is selected, and then the `Actions` tab is selected. A list of currently available actions is displayed. A new action is added by entering the following values in the `Add` dialog (figure 10.2):

- action name: `killallyes`
- description: `kill all yes processes`
- command: `/cm/local/apps/cmd/scripts/actions/killallyes`

The `Save` button adds the action `killallyes` to the list of possible actions, which means that the action can now be used throughout the monitoring framework.

Setting Up The Threshold Level For CPUUser On The Head Node(s)

Continuing on, the `Metric Configuration` tab is selected. Then within the selection box options for `Metric Configuration`, `All Master Nodes` is selected to confine the metrics being measured to the head node(s). The metric `CPUUser`, which is a measure of the user mode CPU usage as a percentage, is selected. The `Thresholds` button is clicked on to open a `Thresholds` dialog. Within the `Thresholds` dialog the `Add` is clicked button to open up a "`New Threshold`" dialog. Within the "`New Threshold`" dialog (figure 10.3), these values are set:

- threshold name: `killallyesthreshold`
- (upper) bound: `50`
- action name (first selection box in the action option): `killallyes`
- action state option (next selection box in the action option): `Enter`

Clicking on `Ok` exits the "`New Threshold`" dialog, clicking on `Done` exits the `Thresholds` dialog, and clicking on `Save` saves the threshold setting associated with `CPUUser` on the head node.

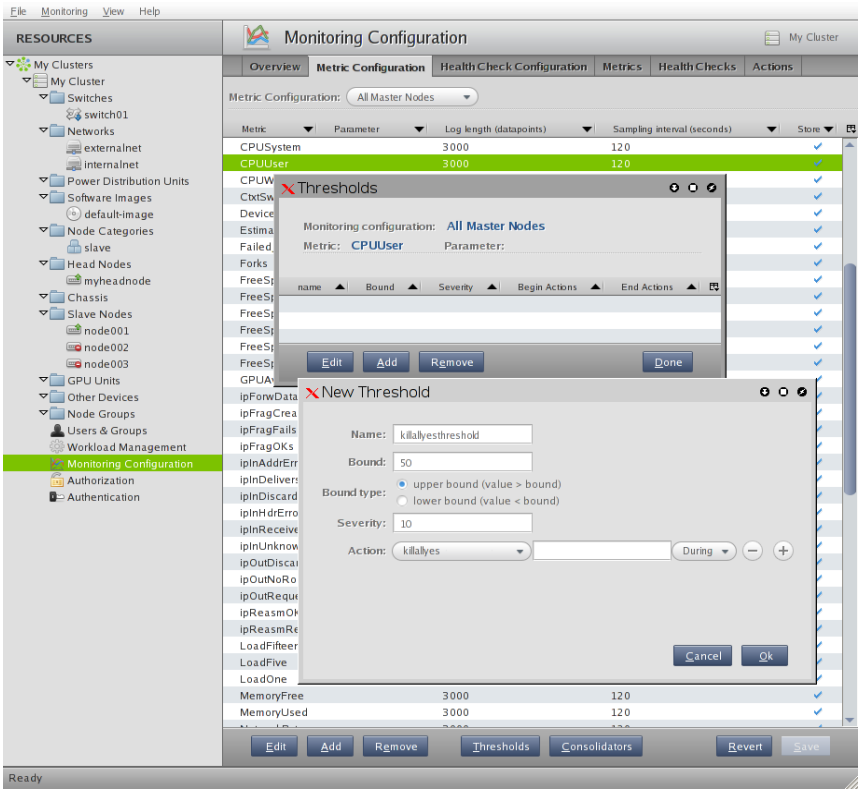

Figure 10.3: `cmgui` Monitoring Configuration: Setting A Threshold

The Result

In the preceding section, an action was added, and a threshold was set up with the monitoring framework.

With a default installation on a newly installed cluster, the measurement of CPUUser is done every 120s (the edit dialog of figure 10.22 shows how to edit this value). The basic example configured with the defaults thus monitors if `CPUUser` on the head node has crossed the bound of 50% every 120s.

If `CPUUser` is found to have entered, that is crossed over from below the value and gone into the zone beyond 50%, then the framework runs the `killallyes` script, killing all running `yes` processes. Assuming the system is trivially loaded apart from these `yes` processes, the `CPUUser` metric value then drops to below 50%.

After an `Enter` threshold condition has been met for a sample, the first sample immediately after that does not ever meet the `Enter` threshold condition, because an `Enter` threshold crossing condition requires the previous sample to be below the threshold.

The second sample can only launch an action if the `Enter` threshold condition is met and if the preceding sample is below the threshold.

Other non-`yes` CPU-intensive processes running on the head node can also trigger the `killallyes` script. Since the script only kills `yes` processes, leaving any non-`yes` processes alone, it would in such a case run unnecessarily. This is a deficiency due to the contrived and simple nature of the basic example being illustrated here, and is of no real concern.

10.2 Monitoring Concepts And Definitions

A discussion of the concepts of monitoring, along with definitions of terms used, is appropriate at this point. The features of the monitoring framework covered later on in this chapter will then be understood more clearly.

10.2.1 Metric

In the basic example of section 10.1, the metric value considered was `CPUUser`, measured at regular time intervals of 120s.

A metric is a property of a device that can be monitored. It has a numeric value and can have units, unless it is unknown, i.e. has a null value. Examples are:

- temperature (value in degrees Celsius, for example: 45.2 °C);
- load average (value is a number, for example: 1.23);
- free space (value in bytes, for example: 12322343).

A metric can be a built-in, which means it is an integral part of the monitoring framework, or it can be a standalone script.

The word metric is often used to mean the script or object associated with a metric as well as a metric value. The context makes it clear which is meant.

10.2.2 Action

In the basic example of section 10.1, the action script is the script added to the monitoring system to kill all yes processes. The script runs when the condition is met that `CPUUser` crosses 50%.

An *action* is a standalone script or a built-in command that is executed when a condition is met. This condition can be:

- health checking (section 10.2.4);
- threshold checking (section 10.2.3) associated with a metric (section 10.2.1);
- state flapping (section 10.2.9).

10.2.3 Threshold

In the basic example of section 10.1, a threshold is set to 50% of `CPUUser`, and an action set so that crossing this threshold runs the `killallyes` script.

A *threshold* is a particular value in a sampled metric. A sample can cross the threshold, thereby entering or leaving a zone that is demarcated by the threshold.

A threshold can be configured to launch an action (section 10.2.2) according to threshold crossing conditions. The "`New Threshold`" dialog of `cmgui` (figure 10.3) has three action launch configuration options:

1. `Enter`: if the sample has entered into the zone and the previous sample was not in the zone
2. `Leave`: if the sample has left the zone and the previous sample was in the zone

3. `During`: if the sample is in the zone, and the previous sample was also in the zone.

A threshold zone also has a settable severity (section 10.2.6) associated with it. This value is processed for the AlertLevel metric (section 10.2.7) when an action is triggered by a threshold event.

10.2.4 Health Check

A *health check* value is a state. It is the response to running a health check script at a regular time interval, with as possible response values: `PASS`, `FAIL`, or `UNKNOWN`. The state is recorded in the monitoring framework.

Examples of health checks are:

- checking if the hard drive still has enough space left on it and returning `PASS` if it has;
- checking if an NFS mount is accessible, and returning `FAIL` if it is not;
- checking if `CPUUser` is below 50%, and returning `PASS` if it is;
- checking if the `cmsh` binary is found, and returning `UNKNOWN` if it is not.

A health check has a settable severity (section 10.2.6) associated with a `FAIL` or `UNKNOWN` response. This value is processed for the AlertLevel metric (section 10.2.7) when the health check runs.

A health check can also launch an action based on any of the response values, similar to the way that an action is launched by a metric with a threshold condition.

10.2.5 Conceptual Overview: Health Checks Vs Threshold Checks

A health check is quite similar to a threshold state check with a metric. Conceptually, however, they are intended to differ as follows:

- A threshold state check works with numeric values.

  A health check on the other hand works with a response state of `PASS`, `FAIL`, or `UNKNOWN`.

- Threshold-checking does not specifically store a direct history of whether the threshold condition was met or not—it just calls the action script right away as its response. Admittedly, the associated metric data values are still kept by the monitoring framework, so that establishing if a threshold has been crossed historically is always possible with a little effort.

  A health check on the other hand stores its `PASS/FAIL/UNKNOWN` responses for the monitoring framework, making it easily accessible for viewing by default.

- The threshold-checking mechanism is intended to be limited to doing a numerical comparison of a metric value with a threshold value

  A health check on the other hand has more general checking capabilities.

With some inventiveness, a health check can be made to do the function of a metric's threshold/action sequence (as well as the other way round).

The considerations above should help decide what the appropriate tool (health check or metric threshold check) should be for the job.

10.2.6 Severity

Severity is a positive integer value that the administrator assigns to a threshold-crossing event or to a health check status event. It takes one of these 5 suggested values:

Value	Name	Icon	Description
0	`info`		informational message
10	`notice`		normal, but significant, condition
20	`warning`		warning conditions
30	`error`		error conditions
40	`alert`		action must be taken immediately

By default the value is 10. It is used in the `AlertLevel` metric (section 10.2.7).

10.2.7 AlertLevel

AlertLevel is a special metric. It is not sampled, but it is re-calculated when an event with an associated `Severity` (section 10.2.6) occurs. There are two types of AlertLevel metrics:

1. *AlertLevel (max)*, which is simply the maximum severity of the latest value of all the events. The aim of this metric is to alert the administrator to the severity of the *most important* issue.
2. *AlertLevel (sum)* which is the *sum* of the latest severity values of all the events. The aim of this metric is to alert the administrator to the *overall severity* of issues.

10.2.8 InfoMessages

InfoMessages are messages that inform the administrator of the reason for a health status event change in the cluster. These show up in the `Overview` tab of nodes, in the `Health Status` section.

10.2.9 Flapping

Flapping, or *State Flapping*, is when a state transition (section 10.2.10) occurs too many times over a number of samples. In the basic example of section 10.1, if the `CPUUser` metric crossed the threshold zone 7 times within 12 samples (the default values for flap detection), then it would by default be detected as flapping. A flapping alert would then be recorded in the event viewer, and a flapping action could also be launched if configured to do so. Flapping configuration for `cmgui` is covered for thresholds crossing events in section 10.4.2, when the metric configuration tab's `Edit` and `Add` dialogs are explained; and also covered for health check state changes in section 10.4.3, when the health check configuration tab's `Edit` and `Add` dialogs are explained.

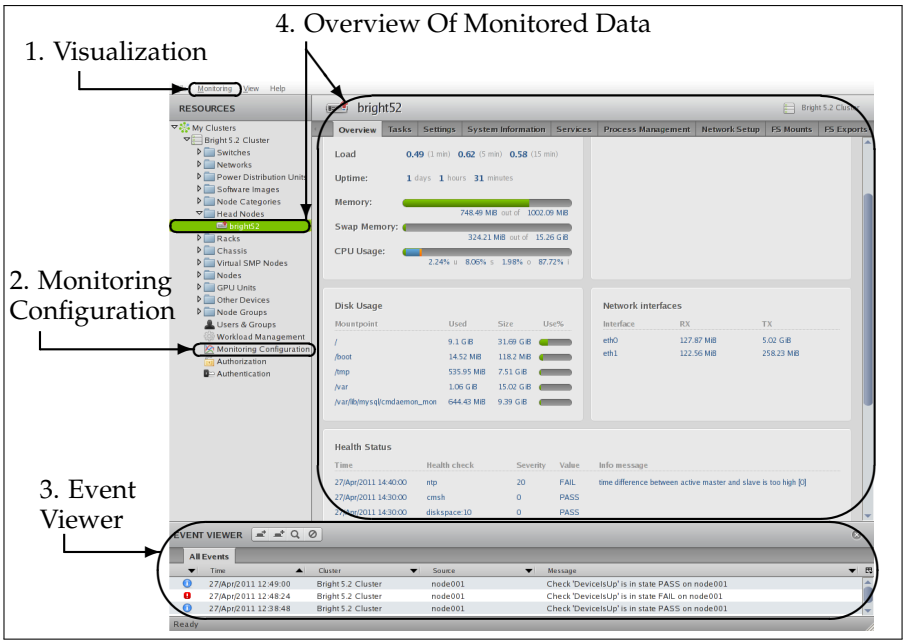

Figure 10.4: `cmgui` Conceptual Overview - Monitoring Types

10.2.10 Transition

A state transition is:

- a health check state change (for example, changing from `PASS` to `FAIL`, or from `FAIL` to `UNKNOWN`);
- a metric threshold (section 10.2.3) crossing event. This is only valid for values that `Enter` or `Leave` the threshold zone.

10.2.11 Conceptual Overview: `cmgui`'s Main Monitoring Interfaces

Monitoring information is presented in several places in `cmgui` for convenience during everyday use. The conceptual overview in figure 10.4 covers a commonly seen layout in `cmgui`, showing 4 monitoring-related viewing areas for the cluster administrator. These are:

1. **Visualization**

   Visualization of monitoring data is made available from `cmgui`'s monitoring menu, and launches a new window. Graphs are generated from metrics and health checks data, and these graphs are viewed in various ways within window panes.

   The use of the visualization tool is covered in section 10.3 using typical data from `CPUUser` from the basic example of section 10.1.

2. **Monitoring Configuration**

   Selecting the `Monitoring Configuration` resource in `cmgui` from the `Resources` list on the left hand side of the Bright Cluster Manager displays the monitoring configuration pane on the right hand side. Within this pane, the following tabs show up:

   - `Overview`: an overview of enabled actions

- `Metric Configuration`: allows configuration of device categories with metrics
- `Health Check Configuration`: allows configuration of device categories with health checks
- `Metrics`: allows configuration of metrics for devices
- `Health Checks`: allows configuration of health checks for devices
- `Actions`: allows actions to be set to run from metric thresholds and health check results

Some parts of `Monitoring Configuration` were used in the basic example of section 10.1 to set up the threshold for `CPUUser`, and to assign the action. It is covered more thoroughly in section 10.4.

3. **Event Viewer**

   The *Event Viewer* is a log of important events that are seen on the cluster(s). How the events are presented is configurable, with tools that allow filtering based on dates, clusters, nodes or a text string; and widgets that allow rearranging the sort order or detaching the pane.

4. **Overview Of Monitored Data**

   A dashboard in a car conveys the most important relevant information at a glance and attracts attention to items that are abnormal and merit further investigation.

   The same idea lies behind the `Overview` tab of Bright Cluster Manager. This gives a dashboard view based on the monitored data for a particular device such as a switch, a cluster (probably the most useful overview, and therefore also the default when first connecting to the cluster with `cmgui`), a node, a GPU unit, and so on.

   Neighboring tabs often allow a closer look at issues noticed in the `Overview`, and also sometimes a way to act on them.

   For example, if jobs are not seen in the `Overview` tab, then the administrator may want to look at the neighboring `Services` tab (figure 10.5), and see if the workload manager is running. The `Services` tab (section 4.8.2) allows the administrator to manage a service such as the workload manager.

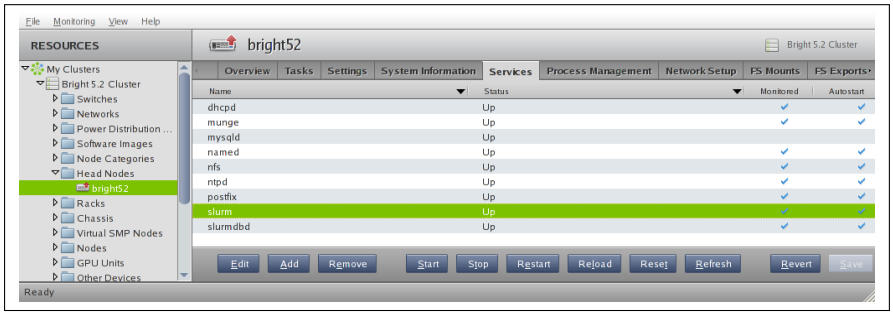

Figure 10.5: `cmgui`: Device Services Tab

10.3 Monitoring Visualization With cmgui

The Monitoring option in the menu bar of cmgui (item 1 in figure 10.4) launches an intuitive visualization tool that should be the main tool for getting a feel of the system's behavior over periods of time. With this tool the measurements and states of the system are viewed. Graphs for metrics and health checks can be looked at in various ways: for example, the graphs can be zoomed in and out on over a particular time period, the graphs can be laid out on top of each other or the graphs can be laid out as a giant grid. The graph scale settings can also be adjusted, stored and recalled for use the next time a session is started.

An alternative to cmgui's visualization tool is the command-line cmsh. This has the same functionality in the sense that data values can be selected and studied according to configurable parameters with it (section 10.8). The data values can even be plotted and displayed on graphs with cmsh with the help of unix pipes and graphing utilities. However, the strengths of monitoring with cmsh lie elsewhere: cmsh is more useful for scripting or for examining pre-decided metrics and health checks rather than a quick visual check over the system. This is because cmsh needs more familiarity with options, and is designed for text output instead of interactive graphs. Monitoring with cmsh is discussed in sections 10.7 and 10.8.

How cmgui is used for visualization is now described.

10.3.1 The Monitoring Window

The Monitoring menu is selected from the menu bar of cmgui and a cluster name is selected.

The Monitoring window opens (figure 10.6). The resources in the cluster are shown on the left side of the window. Clicking on a resource opens or closes its subtree of metrics and health checks.

The subsequent sections describe ways of viewing and changing resource settings. After having carried out such modifications, saving and loading a settings state can be done from options in the File menu.

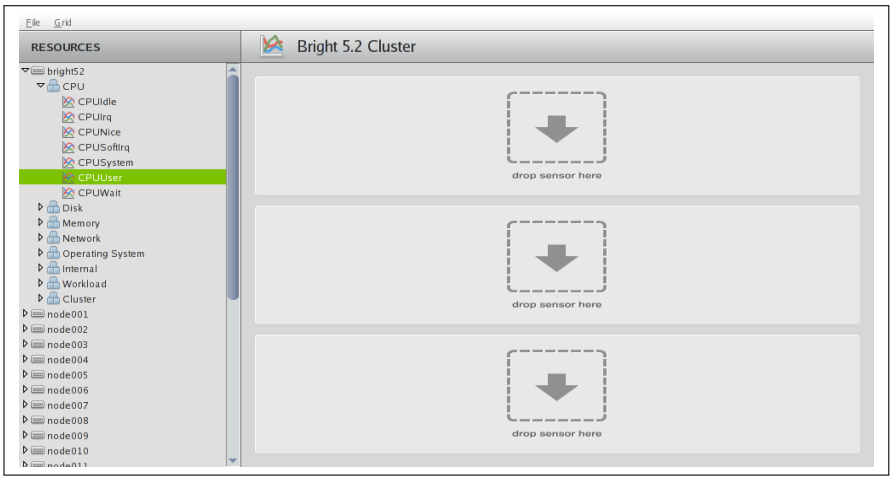

Figure 10.6: cmgui Monitoring Window: Resources View

Figure 10.6 shows the different resources of the head node, with the CPU resource subtree opened up in turn to show its metrics and health

checks. Out of these, the CPUUser metric (for user CPU usage) is shown selected for further display.

To display this metric, the selection is drag-and-dropped onto one of the 3 panes which has the text "drop sensor here".

10.3.2 The Graph Display Pane

Figure 10.7 shows the monitoring window after such a drag-and-drop. The graph of the metric CPUUser is displayed over 20 minutes (10th November 2010 08:04 to 08:24). On the y-axis the unit used by the metric is shown (0% to about 100%). This example is actually of data gathered when the basic example of 10.1 was run, and shows CPUUser rising as a number of yes processes are run, and falling when they end.

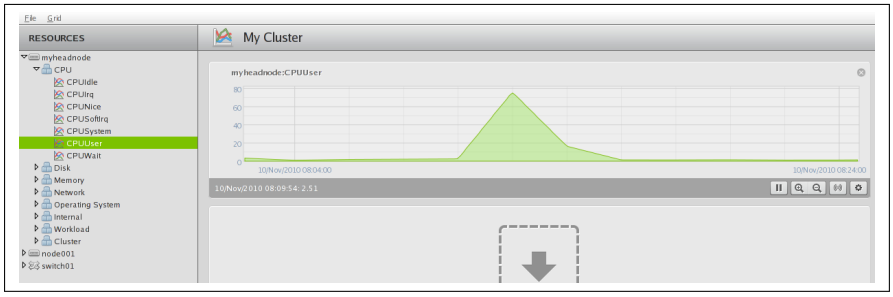

Figure 10.7: cmgui Monitoring Window: Graph Display Pane

Features of graph display panes are (figure 10.8):

1. **The close widget** which erases all graphs on the drawing pane when it is clicked. (Individual graphs are removed in the settings dialog discussed in section 10.3.5.)
2. **The (time, measurement) data values** in the graph are displayed on the graph toolbar by hovering the mouse cursor over the graph.
3. **The graph view adjustment buttons** are:
   - **play/pause**: by default the graph is refreshed with new data every 2 minutes. This is disabled and resumed by clicking on the pause/play button on the graph toolbar.

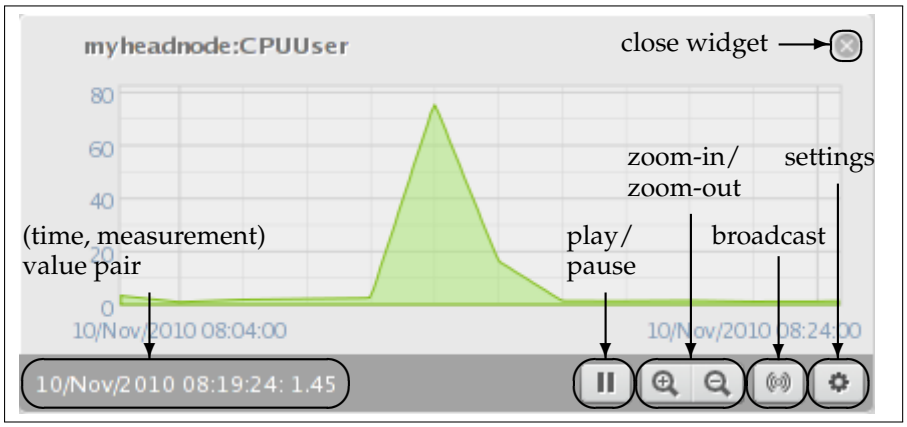

Figure 10.8: Graph Display Pane: Features

- **zoom-out/zoom-in**: Clicking on one of the magnifying glasses zooms-in or zooms-out on the graph in time. This way data values can be shown, even from many months ago. Zooming in with mouse gestures is also possible and is discussed in section 10.3.4.
- **broadcast**: A time-scale synchronizer. Toggling this button to a pressed state for one of the graphs means that scale changes carried out via magnifying glass zooms (preceding bullet point) or via mouse gestures (section 10.3.4) are done over all the other graph display panes too so that their x-ranges match. This is useful for large numbers of nodes.
- **settings**: Clicking on this button opens a dialog window to modify certain aspects of the graph. The settings dialog is discussed in section 10.3.5.

4. **A grid of graph display panes** can be laid out by using the `Grid` menu option of the main `Monitoring Pane` (figure 10.6). Among the menu options of the `Grid` menu (figure 10.9) are:

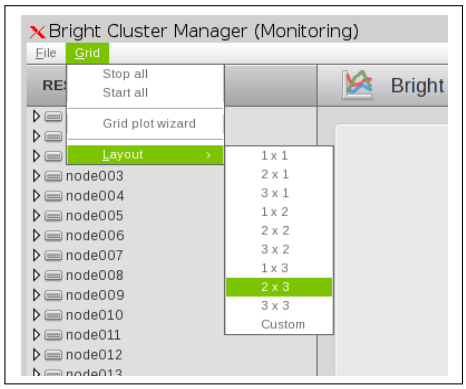

Figure 10.9: Grid Menu Options

   (a) **Layout**: a grid of dimensions $x \times y$ can be selected or specified with the `Layout` option. With the `Layout` option, metrics need to be added manually to each grid unit.

   (b) **Grid plot wizard**: For larger grids it is tedious to allocate devices to a grid and manually fill in the grid units with metrics. The `Grid plot wizard` can take care of the tedious aspects, and is described in section 10.3.3.

5. **Multiple graphs** are drawn in a single graph display pane by repeating the drag and drop for different metrics. For example, adding the CPUIdle metric with a drag-and-drop to the `CPUUser` graph of figure 10.7 gives a result as seen in figure 10.10, where both graphs lie on the same axis in the top pane.

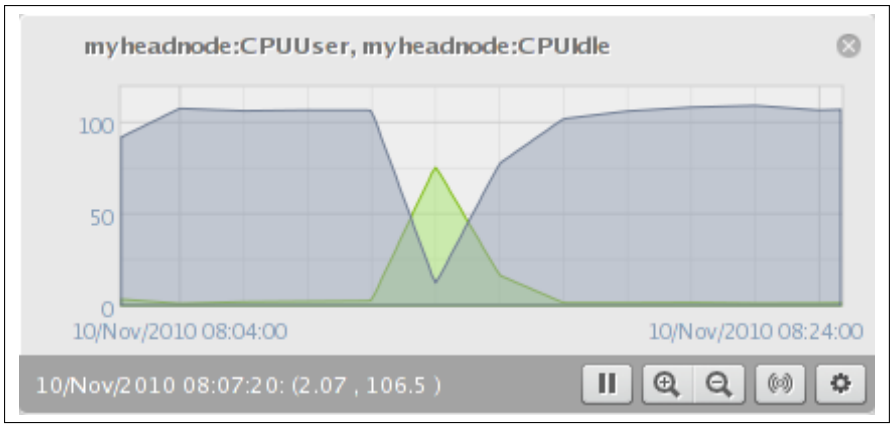

Figure 10.10: Graph Display Pane: Multiple Graphs On One Pane

10.3.3 Using The Grid Wizard

Within the `Monitoring` window (section 10.3.1), the `Grid plot wizard` sets up a grid for devices selected by the administrator, and allows metrics to be added automatically to each grid unit.

The first screen of the wizard allows devices to be selected from the group of all devices, and placed in a group of devices that are to have their metrics plotted (figure 10.11).

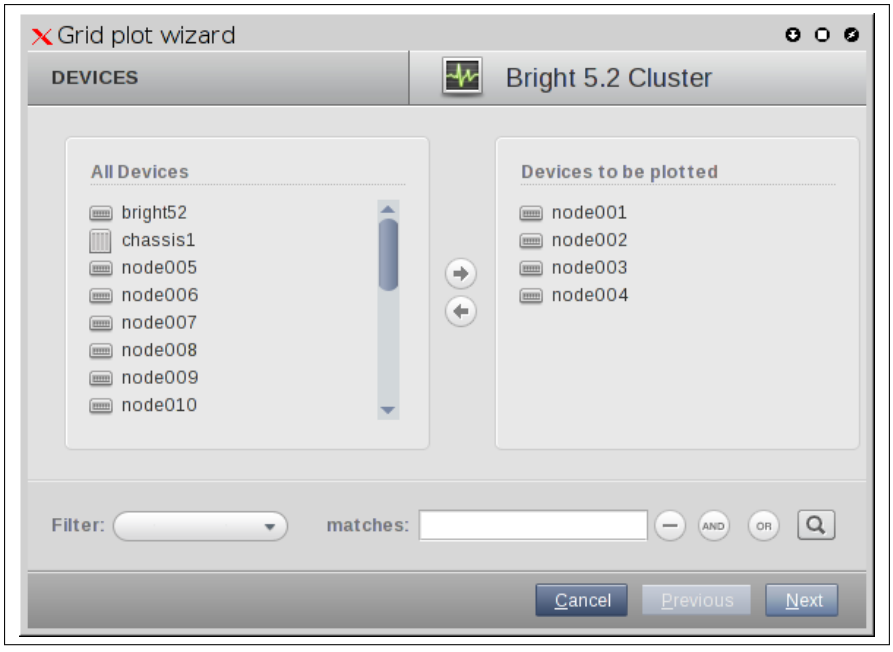

Figure 10.11: Grid Wizard: Devices Selection

The next screen of the wizard allows metrics to be drag-and-dropped from the available metrics into a group of metrics that are to be displayed for the devices in the previous screen (figure 10.12).

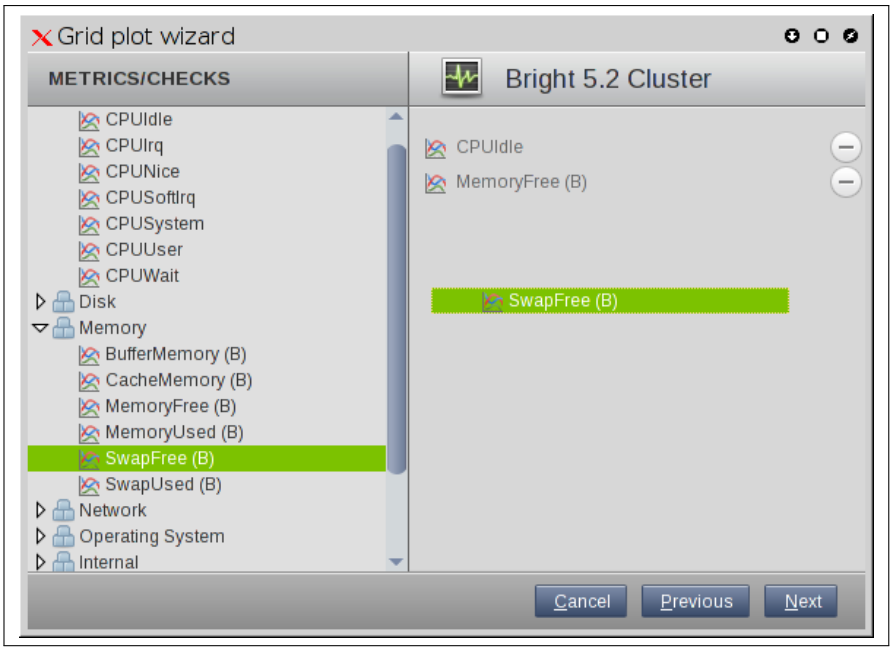

Figure 10.12: Grid Wizard: Metrics Drag-And-Drop

The last screen of the wizard allows several display options to be set for the selected devices and their metrics (figure 10.13).

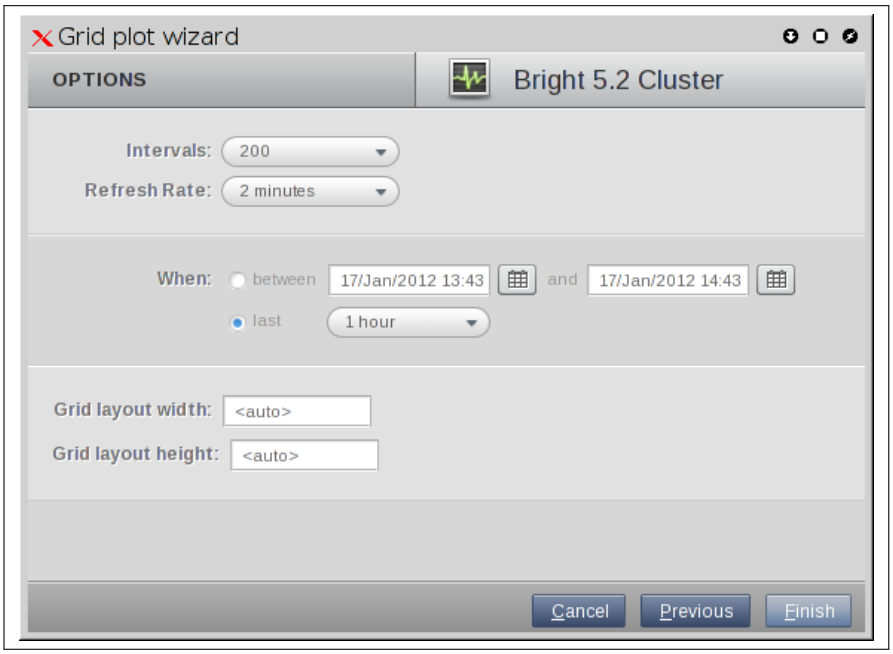

Figure 10.13: Grid Wizard: Display Options

One of these options is the specification of the layout width and height for the displayed grid of selected devices. For example, four nodes could be laid out in a grid of 4 wide by 1 high, 2 wide by 2 high, or 1 wide by 4 high. The meanings of the remaining display options are described in section 10.3.5.

Once the `Finish` button of the last screen is clicked, a graph display

pane is shown with a grid of graphs (figure 10.14).

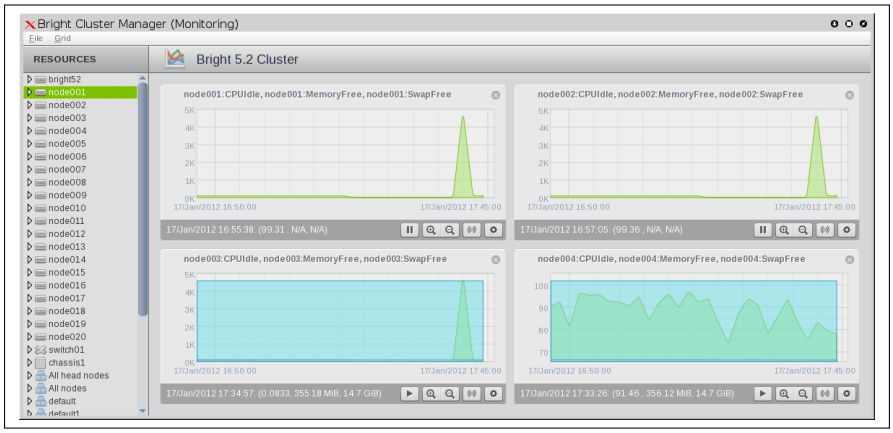

Figure 10.14: Grid Wizard: Grid Result

10.3.4 Zooming In With Mouse Gestures

Besides using a magnifying glass button there are two other ways to zoom in on a graph, based on intuitive mouse gestures:

X-Axis Zoom

The first way to zoom in is to draw a horizontal line across the graph by holding the left mouse button down on the graph. A guide line shows up while doing this (figure 10.15):

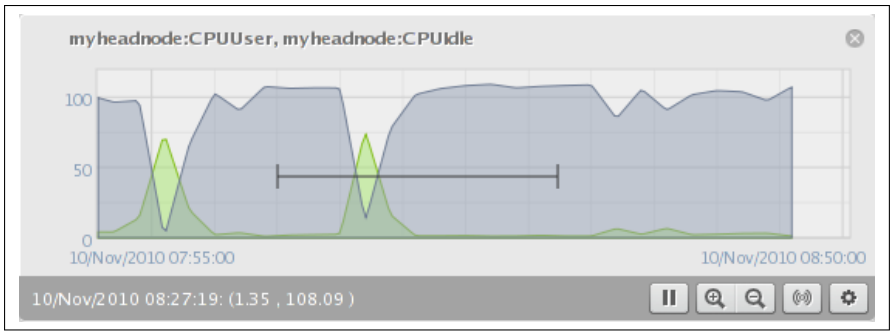

Figure 10.15: Graph Display Pane: X-axis Zoom Start

The x-axis range covered by this line is zoomed in on when the mouse button is released (figure 10.16):

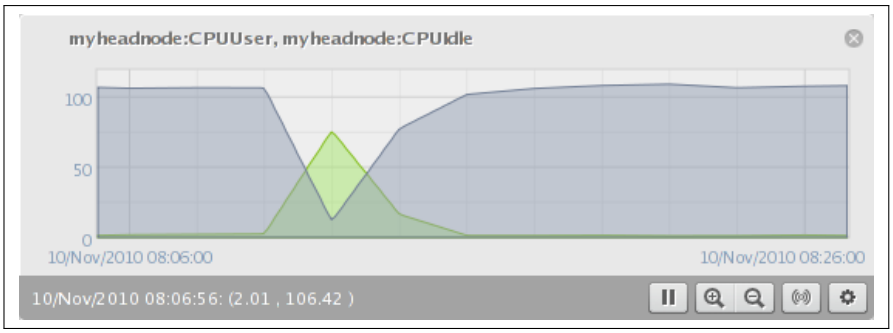

Figure 10.16: Graph Display Pane: X-axis Zoom Finish

Box Zoom

The second way to zoom in is to draw a box instead of a line across the graph by holding the left mouse button down and drawing a line diagonally across the data instead of horizontally. A guide box shows up (figure 10.17):

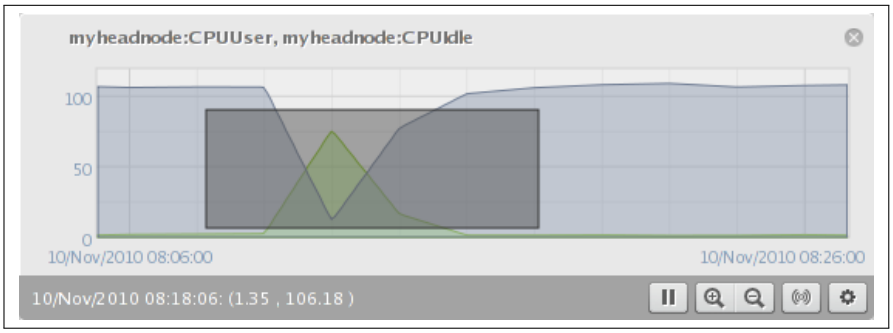

Figure 10.17: Graph Display Pane: Box Zoom Start

This is zoomed into when the mouse button is released (figure 10.18):

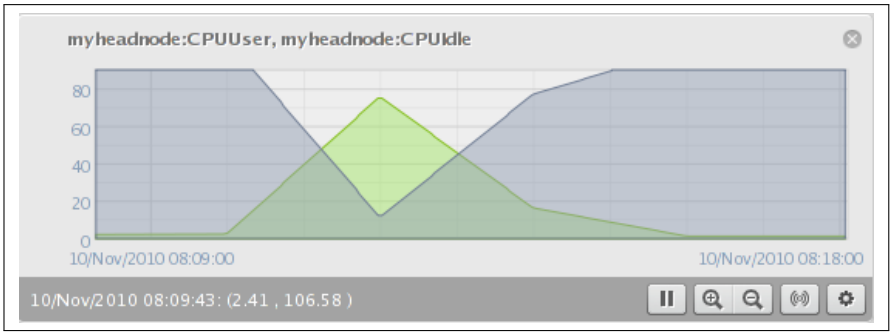

Figure 10.18: Graph Display Pane: Box Zoom Finish

10.3.5 The Graph Display Settings Dialog

Clicking on the settings button in the graph display pane (figure 10.8) opens up the graph display pane settings dialog (figure 10.19):

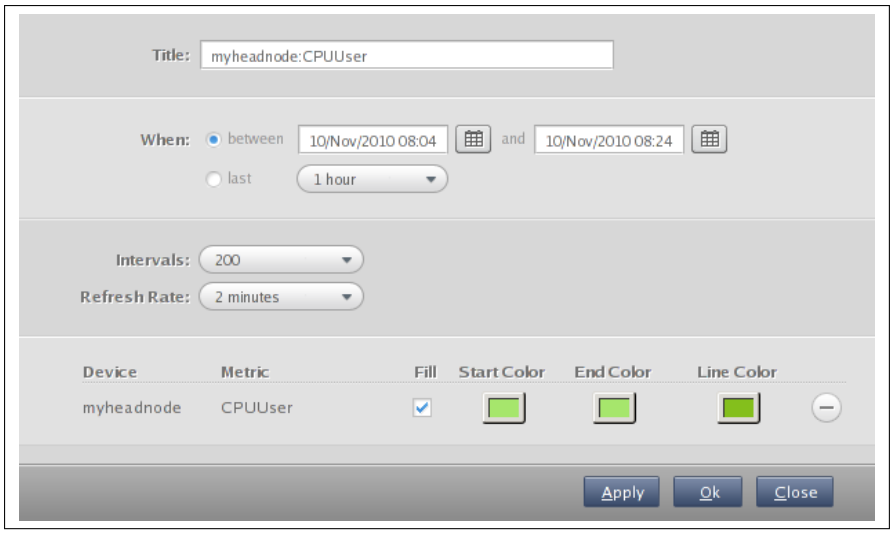

Figure 10.19: Graph Display Pane Settings Dialog

This allows the following settings to be modified:

- the `Title` shown at the top of the graph;
- over `When` the x-range is displayed;
- the `Intervals` value. This is the number of intervals (by default 200) used to draw the graph. For example, although there may be 2000 data points available during the selected period, by default only 200 are used, with each of the 200 an average of 10 real data points. This mechanism is especially useful for smoothing out noisier metrics to give a better overview of behavior.
- The `Refresh Rate`, which sets how often the graph is recreated;
- the visual layout of the graphs, which can be adjusted so that:
  - `Color` aspects of each graph are changed in the row of settings for that graph;
  - Each graph is deleted from its pane with the ⊖ button at the end of the row of settings for that graph.

10.4 Monitoring Configuration With `cmgui`

This section is about the configuration of monitoring for health checks and metrics, along with setting up the actions which are triggered from a health check or a metric threshold check.

Selecting `Monitoring Configuration` from the resources section of `cmgui` makes the following tabs available (figure 10.20):

- `Overview` (displays as the default)
- `Metric Configuration`
- `Health Check Configuration`
- `Metrics`

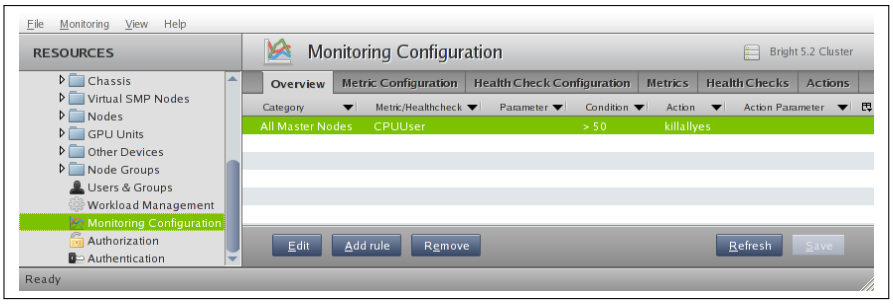

Figure 10.20: `cmgui` Monitoring Configuration Tabs

- `Health Checks`
- `Actions`

The tabs are now discussed in detail.

10.4.1 The Overview Tab

The `Overview` tab of figure 10.20 shows an overview of custom threshold actions and custom health check actions that are active in the system. Each row of conditions in the list that decides if an action is launched is called a rule. Only one rule is on display in figure 10.20, showing an overview of the metric threshold action settings which were set up in the basic example of section 10.1.

The `Add rule` button runs a convenient wizard that guides an administrator in setting up a condition, and thereby avoids having to go through the other tabs separately.

The `Remove` button removes a selected rule.

The `Edit` button edits aspects of a selected rule. It opens a dialog that edits a metric threshold configuration or a health check configuration. These configuration dialog options are also accessible from within the `Metric Configuration` and `Health Check Configuration` tabs.

The `Revert` button reverts a modified state of the tab to the last saved state.

The `Save` button saves a modified state of the tab.

10.4.2 The Metric Configuration Tab

The `Metric Configuration` tab allows device categories to be selected for the sampling of metrics. Properties of metrics related to the taking of samples can then be configured from this tab for the selected device category. These properties are the configuration of the sampling parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as thresholds, consolidation, actions launched when a threshold is crossed, and actions launched when a metric state is flapping.

The `Metric Configuration` tab is initially a blank tab until the device category is selected by using the `Metric Configuration` selection box. The selection box selects the device category from a list of built-in categories and user-defined node categories (node categories are introduced in section 3.1.3). On selection, the metrics of the selected device category are listed in the `Metric Configuration` tab. Properties of the metrics related to sampling are only available for configuration and manipula-

tion after the metrics list displays. Handling metrics in this manner, via groups of devices, is slightly awkward for just a few machines, but for larger clusters it keeps administration scalable and thus manageable.

Figure 10.21 shows an example of the Metric Configuration tab after All master nodes is chosen as the device category. This corresponds to the basic example of section 10.1, where All master nodes was the device category chosen because it was the CPUUser metric on a master node that was to be monitored. Examples of other device categories that could be chosen are All ethernet switches, if Ethernet switches are to have their metrics configured; or All Power Distribution Units, if power distribution units are to have their metrics configured.

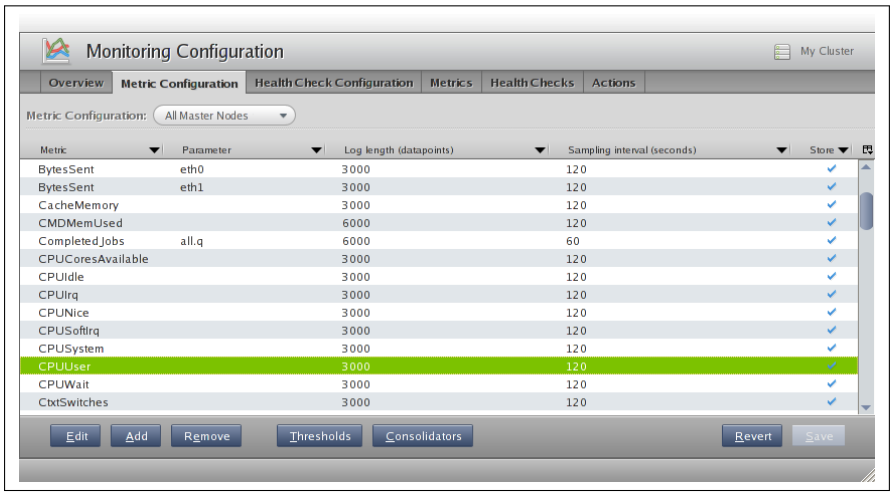

Figure 10.21: cmgui Monitoring: Metric Configuration Display After Category Selection

With the screen displaying a list of metrics as in figure 10.21, the metrics in the Metric Configuration tab can now be configured and manipulated. The buttons used to do this are: Edit, Add, Remove, Thresholds, Consolidators, Revert and Save.

The Save button saves as-yet-uncommitted changes made via the Add or Edit buttons.

The Revert button discards unsaved edits made via the Edit button. The reversion goes back to the last save.

The Remove button removes a selected metric from the metrics listed.

The remaining buttons, Edit, Add, Thresholds and Consolidators, open up options dialogs. These options are now discussed.

Metric Configuration Tab: Edit And Add Options

The Metric Configuration tab of figure 10.21 has Add and Edit buttons. The Add button opens up a dialog to add a new metric to the list, and the Edit button opens up a dialog to edit a selected metric from the list. The dialogs allow logging options for a metric to be set or adjusted. For example, a new metric could be set for sampling by adding it to the device category from the available list of all metrics, or the sampling frequency could be changed on an existing metric, or an action could be set for a metric that has a tendency to flap.

The Edit and Add dialogs for a metric have the following options (fig-

ure 10.22):

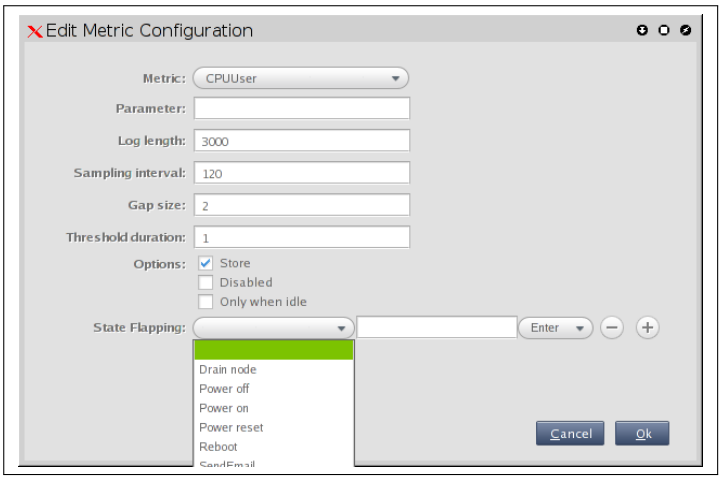

Figure 10.22: `cmgui` Monitoring: Metric Configuration Tab Edit Dialog

- `Metric`: The name of the metric.
- `Parameter`: Values that the metric script is designed to handle. For example:
  - the metric `FreeSpace` tracks the free space left in a file system, and is given a mount point such as `/` or `/var` as a parameter;
  - the metric `BytesRecv` measures the number of bytes received on an interface, and takes an interface name such as `eth0` or `eth1` as a parameter.

  For `CPUUser`, the parameter field is disallowed in the `Metric` tab, so values here are ignored.
- `Log length`: The maximum number of raw data samples that are stored for the metric. 3000 by default.
- `Sampling interval`: The time between samples. 120s by default.
- `Gap size`: The number of missing samples allowed before a null value is stored as a sample value. 2 by default.
- `Threshold duration`: Number of samples in the threshold zone before a threshold event is decided to have occurred. 1 by default.
- `Options` checkboxes:
  - `Store`: If ticked, the metric data values are saved to the database. Note that any threshold checks are still done, whether the samples are stored or not.
  - `Disabled`: If ticked, the metric script does not run, and no threshold checks are done for it. If `Store` is also ticked, no value is stored.
  - `Only when idle`: If ticked, the metric script is only run when the system is idling. A resource-hungry metric burdens the system less this way.

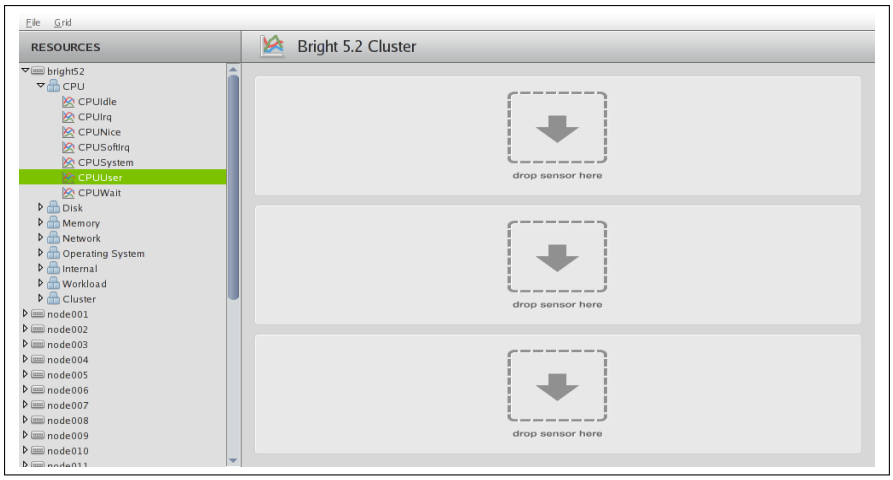

Figure 10.23: cmgui Monitoring: Thresholds Display

- State Flapping: The first selection box decides what action to launch if state flapping is detected. The next box is a plain text-entry box that allows a parameter to be passed to the action. The third box is a selection box again, which decides when to launch the action, depending on which of these following states is set:
  - Enter: if the flapping has just started. That is, the current sample is in a flapping state, and the previous sample was not in a flapping state.
  - During: if the flapping is ongoing. That is, the current and previous flapping sample are both in a flapping state.
  - Leave: if the flapping has just stopped. That is, the current sample is not in a flapping state, and the previous sample was in a flapping state.

Metric Configuration Tab: Thresholds Options

The Metric Configuration tab of figure 10.21 also has a Thresholds button associated with a selected metric.

Thresholds are defined and their underlying concepts are discussed in section 10.2.3. The current section describes the configuration of thresholds.

In the basic example of section 10.1, CPUUser was configured so that if it crossed a threshold of 50%, it would run an action (the killallyes script). The threshold configuration was done using the Thresholds button of cmgui.

Clicking on the Thresholds button launches the Thresholds display window, which lists the thresholds set for that metric. Figure 10.23, which corresponds to the basic example of section 10.1, shows a Thresholds display window with a threshold named killallyesthreshold configured for the metric CPUUser.

The Edit, and Remove buttons in this display edit and remove a selected threshold from the list of thresholds, while the Add button adds a new threshold to the list.

The Edit and Add dialogs for a threshold prompt for the following values (figure 10.24):

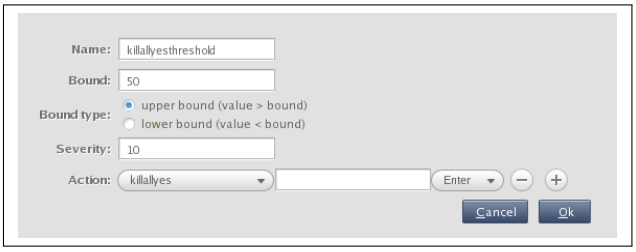

Figure 10.24: `cmgui` Metric Configuration: Thresholds Edit Dialog

- `Name`: the threshold's name.
- `Bound`: the metric value which demarcates the threshold.
- `Bound type`: If checked, the radio button for
  - `upper bound`: places the threshold zone above the bound;
  - `lower bound`: places the threshold zone below the bound.
- `Severity`: A value assigned to indicate the severity of the situation if the threshold is crossed. It is 10 by default. `Severity` is discussed in section 10.2.6.
- `Action`: The action field types decide how the action should be triggered and run. The field types are, from left to right:
  - `script`: a script selected from a drop-down list of available actions;
  - `parameter`: [optional] what parameter value to pass to the action;
  - `when`: when the action is run. It is selected from a drop-down choice of `Enter`, `During` or `Leave`, where:
    - `Enter` runs the action if the sample has entered the zone;
    - `Leave` runs the action if the sample has left the zone;
    - `During` runs the action if the sample is in the zone, and the previous sample was also in the zone.

Metric Configuration Tab: Consolidators Options

The `Metric Configuration` tab of figure 10.21 also has a `Consolidators` button associated with the selected metric.

Consolidators decide how the data values are handled once the initial log length quantity for a metric is exceeded. Data points that have become old are gathered and, when enough have been gathered, they are processed into consolidated data. Consolidated data values present fewer data values than the original raw data values over the same time duration. The aim of consolidation is to increase performance, save space, and keep the basic information still useful when viewing historical data.

The `Consolidators` button opens a window that displays a list of consolidators that have been defined for the selected metric (figure 10.25).

The `Edit` and `Remove` buttons in this display edit and remove a selected consolidator from the list of consolidators while the `Add` button in this display adds a new consolidator to the list of consolidators.

The `Edit` and `Add` dialogs for a consolidator prompt for the following values (figure 10.26):

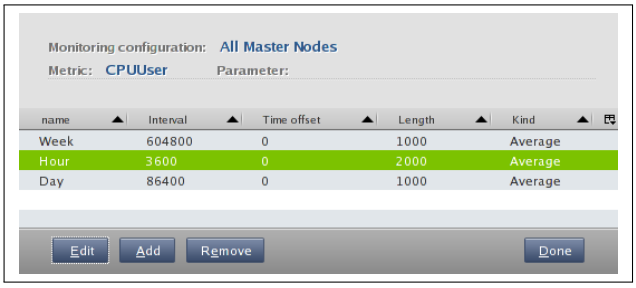

Figure 10.25: cmgui Metric Configuration: Consolidators Display

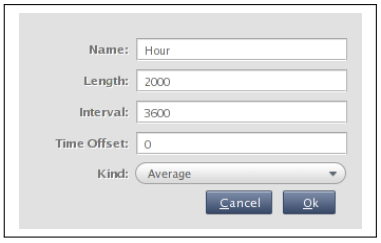

Figure 10.26: cmgui Metric Configuration: Consolidators Edit Dialog

- Name: the consolidator's name. By default Day, Hour Month are already set up, with appropriate values for their corresponding fields.
- Length: the number of intervals that are logged for this consolidator. Not to be confused with the metric log length.
- Interval: the time period (in seconds) associated with the consolidator. Not to be confused with the metric interval time period. For example, the default consolidator with the name Hour has a value of 3600.
- Time Offset: The time offset from the default consolidation time.

  To understand what this means, consider the Log length of the metric, which is the maximum number of raw data points that the metric stores. When this maximum is reached, the oldest data point is removed from the metric data when a new data point is added. Each removed data point is gathered and used for data consolidation purposes.

  For a metric that adds a new data point every Sampling interval seconds, the time $t_{\text{raw gone}}$, which is how many seconds into the past the raw log data point is removed, is given by:

  $$t_{\text{raw gone}} = (\text{Log length})_{\text{metric}} \times (\text{Sampling interval})_{\text{metric}}$$

  This value is also the default consolidation time, because the consolidated data values are normally presented from $t_{\text{raw gone}}$ seconds ago, to further into the past. The default consolidation time occurs when the Time Offset has its default, zero value.

  If however the Time Offset period is non-zero, then the consolidation time is offset, because the time into the past from which consolidation is presented to the user, $t_{\text{consolidation}}$, is then given by:

  $$t_{\text{consolidation}} = t_{\text{raw gone}} + \text{Time Offset}$$

The monitoring visualization graphs then show consolidated data from $t_{\texttt{consolidation}}$ seconds into the past, to further into the past[1].

- `Kind`: the kind of consolidation done on the raw data samples. The output result for a processed set of raw data—the consolidated data point—is an average, a maximum or a minimum of the input raw data values. `Kind` can thus have the value `Average`, `Maximum`, or `Minimum`.

10.4.3 Health Check Configuration Tab

The `Health Check Configuration` tab behaves in a similar way to the `Metric Configuration` tab of section 10.4.2, with some differences arising due to working with health checks instead of metric values.

The `Health Check Configuration` tab allows device categories to be selected for the evaluating the states of health checks. Properties of health checks related to the evaluating these states can then be configured from this tab for the selected device category. These properties are the configuration of the state evaluation parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as severity levels based on the evaluated state, the actions to launch based on the evaluated state, or the action to launch if the evaluated state is flapping.

The `Health Check Configuration` tab is initially a blank tab until the device category is selected by using the `Health Check Configuration` selection box. The selection box selects a device category from a list of built-in categories and user-defined node categories (node categories are introduced in section 3.1.3). On selection, the health checks of the selected device category are listed in the `Health Check Configuration` tab. Properties of the health checks related to the evaluation of states are only available for configuration and manipulation after the health checks list is displayed. Handling health checks in this manner, via groups of devices, is slightly awkward for just a few machines, but for larger clusters it keeps administration scalable and thus manageable.

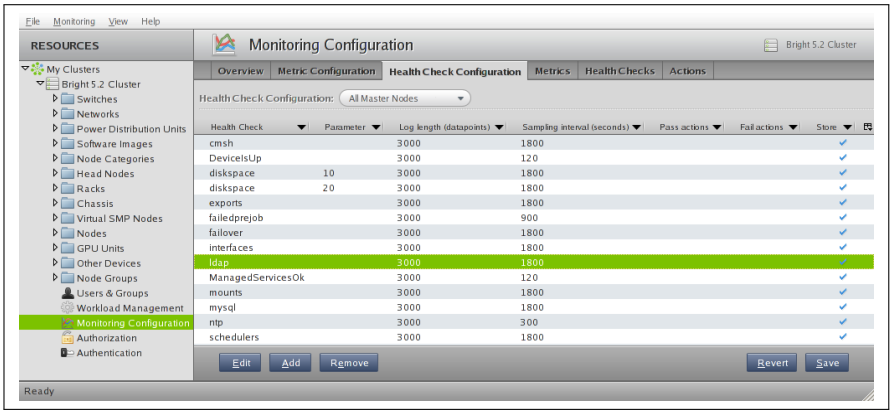

Figure 10.27: `cmgui` Monitoring: Health Check Configuration Display After Category Selection

[1] For completeness: the time $t_{\texttt{consolidation gone}}$, which is how many seconds into the past the consolidated data goes and is viewable, is given by an analogous equation to that of the equation defining $t_{\texttt{raw gone}}$:

$$t_{\texttt{consolidation gone}} = (\texttt{Log length})_{\texttt{consolidation}} \times (\texttt{Sampling interval})_{\texttt{consolidation}}$$

Figure 10.27 shows an example of the Health Check Configuration tab after All master nodes is chosen as the category. Examples of other categories that could be chosen to have health checks carried out on them are All ethernet switches and All Power Distribution Units.

With the screen displaying a list of health checks as in figure 10.27, the health checks in the Health Check Configuration tab can now be configured and manipulated. The buttons used to do this are: Edit, Add, Remove, Revert and Save.

These Health Configuration tab buttons behave just like the corresponding Metric Configuration tab buttons of section 10.4.2, that is:

The Save button saves as-yet-uncommitted changes made via the Add or Edit buttons.

The Revert button discards unsaved edits made via the Edit button. The reversion goes back to the last save.

The Remove button removes a selected health check from the health checks listed.

The remaining buttons, Edit and Add, open up options dialogs. These are now discussed.

Health Check Configuration Tab: Edit And Add Options

The Health Check Configuration tab of figure 10.27 has Add and Edit buttons. The Add button opens up a dialog to add a new health check to the list, and the Edit button opens up a dialog to edit a selected health check from the list. The dialogs are very similar to those of the Add and Edit options of Metric Configuration in section 10.4.2. The dialogs for the Health Check Configuration tab are as follows (figure 10.28):

- Health Check: The name of the health check.
- Parameter: The values that the health check script is designed to handle. For example:
  - the health check ldap checks if the ldap service is running. It tests the ability to look up a user on the LDAP server using cmsupport as the default user. If a value is specified for the parameter, it uses that value as the user instead;
  - the health check portchecker takes parameter values such as 192.168.0.1 22 to check the if host 192.168.0.1 has port 22 open.
- Log length: The maximum number of samples that are stored for the health check. 3000 by default.
- Sampling interval: The time between samples. 120s by default.
- Prejob: Clicking on this button sets the health check to run before a new job is run from the scheduler of the workload management system, instead of running at regular intervals.
- Gap size: The number of missing samples allowed before a null value is stored as a sample value. 2 by default.
- Threshold duration: Number of samples in the threshold zone before a health check state is decided to have changed. 1 by default.

- `Fail severity`: The severity value assigned to a FAIL response for a health check. 10 by default.
- `Unknown severity`: The severity value assigned to an UNKNOWN response for a health check. 10 by default.
- `Options` checkboxes:
  - `Store`: If ticked, the health check state data values are saved to the database. Note that health state changes and actions still take place, even if no values are stored.
  - `Disabled`: If ticked, the health state script does not run, and no health check state changes or actions associated with it occur. If `Store` is ticked, the value it stores while `Disabled` is ticked for this health check configuration is an `UNKNOWN` value
  - `Only when idle`: If ticked, the health check script is only run when the system is idling. This burdens a system less, and is useful if the health check is resource-hungry.
- `Pass action, Fail action, Unknown action, State Flapping`: These are all action launchers, which launch an action for a given health state (PASS, FAIL, UNKNOWN) or for a flapping state, depending on whether these states are true or false. Each action launcher is associated with three input boxes. The first selection box decides what action to launch if the state is true. The next box is a plain text-entry box that allows a parameter to be passed to the action. The third box is a selection box again, which decides when to launch the action, depending on which of the following conditions is met:
  - `Enter`: if the state has just started being true. That is, the current sample is in that state, and the previous sample was not in that state.
  - `During`: if the state is true, and ongoing. That is, the current and previous state sample are both in the same state.
  - `Leave`: if the state has just stopped being true. That is, the current sample is not in that state, and the previous sample was in that state.

10.4.4 Metrics Tab

The `Metrics` tab displays the list of metrics that can be set in the cluster. Some of these metrics are built-ins, such as `CPUUser` in the basic example of section 10.1. Other metrics are standalone scripts. New custom metrics can also be built and added as standalone commands or scripts.

Metrics can be manipulated and configured.

The `Save` button saves as-yet-uncommitted changes made via the `Add` or `Edit` buttons.

The `Revert` button discards unsaved edits made via the `Edit` button. The reversion goes back to the last save.

The `Remove` button removes a selected metric from the list.

The remaining buttons, `Edit` and `Add`, open up options dialogs. These are now discussed.

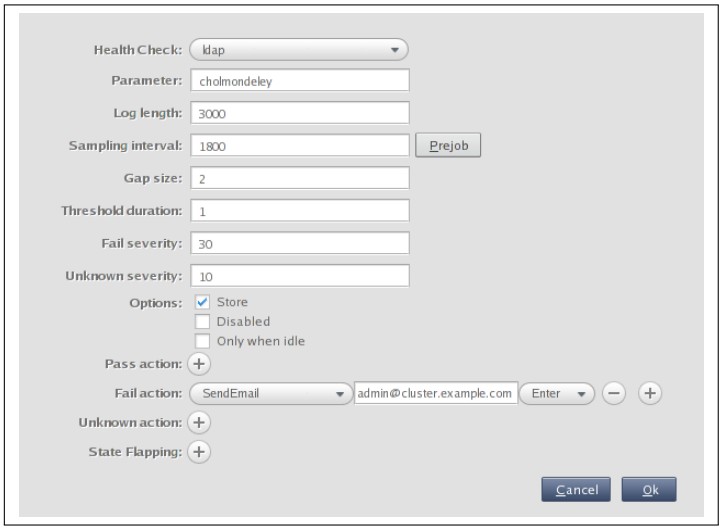

Figure 10.28: cmgui Monitoring: Health Check Configuration Edit Dialog

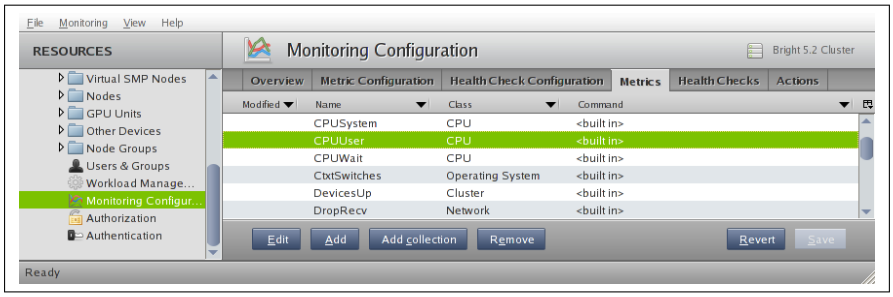

Figure 10.29: cmgui Monitoring: Metrics Tab

Metrics Tab: Edit And Add Options

The Metrics tab of figure 10.29 has Add and Edit buttons. The Add button opens up a dialog to add a new metric to the list, and the Edit button opens up a dialog to edit a selected metric from the list. Both dialogs have the following options (figure 10.30):

- Name: the name of the metric.
- Description: the description of the metric.
- Command: the command that carries out the script, or the full path to the executable script.
- Command timeout: After how many seconds the script should stop running, in case of no response.
- Parameter: an optional value that is passed to the script.
- Cumulative: whether the value is cumulative (for example, the bytes-received counter for an Ethernet interface), or non-cumulative (for example, temperature).
- Unit: the unit in which the metric is measured.
- When to run:
  - Disabled: if ticked, the metric script does not run.

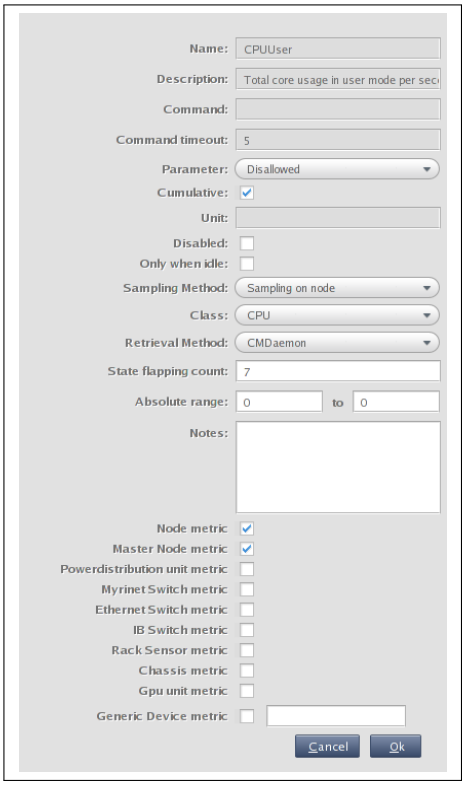

Figure 10.30: `cmgui` Monitoring: Metrics Tab, Edit Dialog

  - `Only when idle`: if ticked, the metric script only runs when the system is idling. This burdens the system less if the metric is resource-hungry.

- `Sampling Method`: the options are:
  - `Sampling on master`: The head node samples the metric on behalf of a device. For example, the head node may do this for a PDU, since a PDU does not have the capability to run the cluster management daemon at present, and so cannot itself pass on data values directly when `cmsh` or `cmgui` need them.
  - `Sampling on node`: The non-head node samples the metric itself. The administrator should ensure that the script is accessible from the non-head node.

- `Class`: An option selected from:
  - `Misc`
  - `CPU`
  - `GPU`
  - `Disk`
  - `Memory`
  - `Network`
  - `Environmental`
  - `Operating System`
  - `Internal`

  - Workload
  - Cluster
  - Prototype

  These options should not be confused with the device category that the metric can be configured for, which is a property of where the metrics can be applied. (The device category possibilities are listed in a bullet point a little further on).

- Retrieval Method:
  - cmdaemon: Metrics retrieved internally using CMDaemon (default).
  - snmp: Metrics retrieved internally using SNMP.
- State flapping count (default value 7): How many times the metric value must cross a threshold within the last 12 samples (a default setting, set in cmd.conf) before it is decided that it is in a flapping state.
- Absolute range: The range of values that the metric takes. A range of 0–0 implies no constraint is imposed.
- Notes: Notes can be made here.
- Which device category the metric is configured for, with choices out of:
  - Node metric
  - Master Node metric
  - Power Distribution Unit metric
  - Myrinet Switch metric
  - Ethernet Switch metric
  - IB Switch metric
  - Rack Sensor metric
  - Chassis metric
  - GPU Unit metric
  - Generic Device metric

  These options should not be confused with the class that the metric belongs to (the earlier Class bullet point), which is the property type of the metric.

Metrics Tab: Add Collection Option

The Add Collection button opens a dialog which is used to create a *metric collection* (figure 10.31). A metric collection is a special metric script, with the following properties:

- It is able to return several metrics of different types when it is run, not just one metric of one type like a normal metric script does—hence the name, "metric collection".

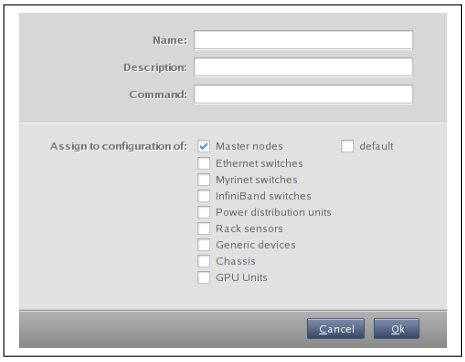

Figure 10.31: `cmgui` Monitoring: Metrics Tab, `Add collection` Dialog

- It autodetects if its associated metrics are able to run, and to what extent, and presents the metrics accordingly. For example, if the metric collection is run on a node which only has 3 CPUs running rather than a default of 4, it detects that and presents the results for just the 3 CPUs.

Further details on metric collections scripts are given in appendix I.

Because handling metric collections is just a special case of handling a metric, the `Add Collection` button dialog is merely a restricted version of the `Add` button dialog. Setting up a metric collection is therefore simplified by having most of the metric fields pre-filled and kept hidden. For example, the `Class` field for a metric collection would have the value `Prototype` in the `Add` button dialog, while this value is pre-filled and invisible in the `Add Collection` dialog. A metric collection can be created with the `Add` dialog, but it would be a little more laborious.

Whatever the method used to create the metric collection, it can always be edited with the `Edit` button, just like any other metric.

Viewing visualizations of a metric collection in `cmgui` is only possible through selection and viewing the separate graphs of its component metrics.

10.4.5 Health Checks Tab

The `Health Checks` tab lists available health checks (figure 10.32). These can be set to run from the system by configuring them from the `Health Check Configuration` tab of section 10.4.3.

What the listed health checks on a newly installed system do are described in appendix H.2.1.

The `remove`, `revert` and `save` buttons work for health checks just like they do for metrics in section 10.4.4

Also, the `edit` and `add` buttons start up dialogs to edit and add health checks. The dialog options for health checks are the same as for editing or adding metrics, with a few exceptions. The exceptions are for options that are inapplicable for health checks, and are elaborated on in appendix H.2.2.

10.4.6 Actions Tab

The `Actions` tab lists available actions (figure 10.33) that can be set to run on the system from metrics thresholds configuration, as explained in section 10.4.2, and as was done in the basic example of section 10.1. Actions

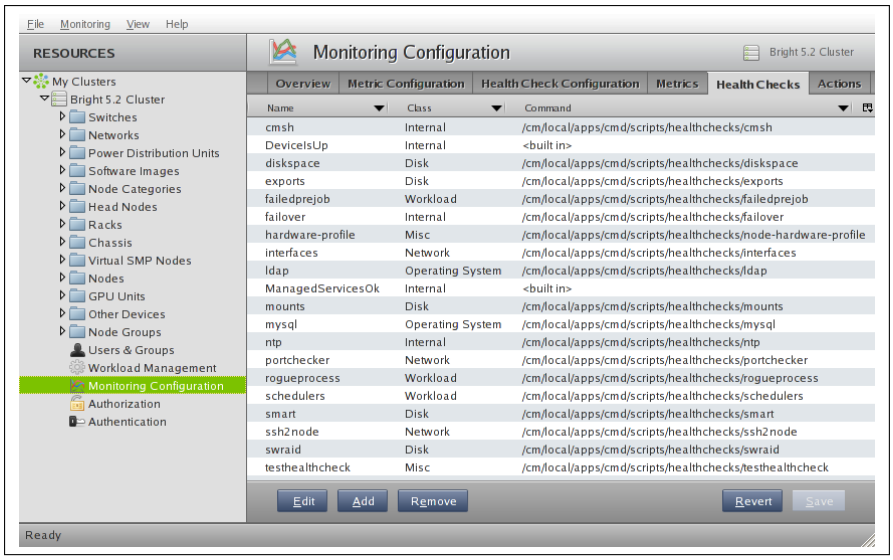

Figure 10.32: `cmgui` Monitoring: Health Checks Tab

can also be set to run from health check configuration action launcher options as described in section 10.4.3.

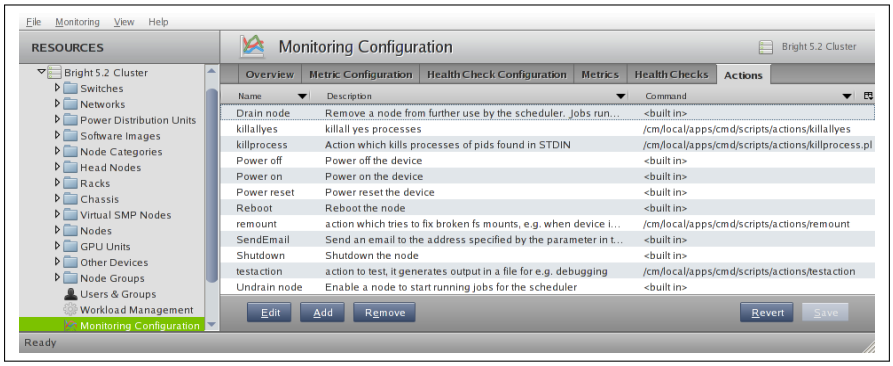

Figure 10.33: `cmgui` Monitoring: Actions Tab

What the listed actions on a newly installed system do are described in appendix H.3.1.

The `remove`, `revert`, and `save` buttons work as described for metrics in section 10.4.4.

The `edit` and `add` buttons start up dialogs to edit or add options to action parameters. Action parameters are described in appendix H.3.2.

10.5 Overview Of Monitoring Data For Devices

These views are set up under the `Overview` tab for various devices that are items under the resource tree in the cluster.

They are a miscellany of monitoring views based on the monitored data for a particular device. The views are laid out as part of an overview tab for that device, which can be a switch, cluster, node, GPU unit, and so on.

When first connecting to a cluster with `cmgui`, the `Overview` tab of the cluster is the default view. The `Overview` tab is also the default view first time a device is clicked on in a `cmgui` session.

Of the devices, the cluster(s), head node(s) and regular nodes have a relatively extensive `Overview` tab, with a pre-selected mix of information from monitored data. For example, in figure 10.4, a head node is shown with an `Overview` tab presenting memory used, CPU usage, disk usage, network statistics, running processes, and health status. Some of these values are presented with colors and histograms to make the information easier to see.

10.6 Event Viewer

This is a log view of events on the cluster(s). It is accessible from the `View` menu of the main window.

The logs can be handled and viewed in several ways.

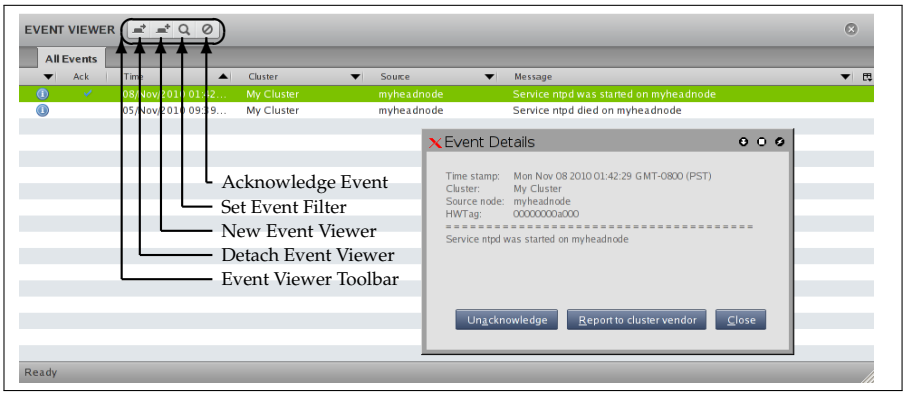

Figure 10.34: `cmgui` Monitoring: Event Viewer Pane

Double clicking on an event row starts up an `Event Details` dialog (figure 10.34), with buttons to:

- `Acknowledge` or `Unacknowledge` the event, as appropriate. Clicking on `Acknowledge` removes the event from the event view unless the `Show Acknowledged` checkbox has been checked. Any visible acknowledged events have their acknowledged status removed when the `Unacknowledge` button is clicked.
- `Report to cluster vendor`. The report option is used for sending an e-mail about the selected event to the cluster vendor in case troubleshooting and support is needed.

The event viewer toolbar (figure 10.34) offers icons to handle event logs:

- `detach event viewer`: Detaches the event viewer pane into its own window. Reattachment is done by clicking on the reattachment event viewer icon that becomes available in the detached window.
- `new event viewer filter dialog`: Loads or defines filters (figure 10.35). Filters can be customized according to acknowledgement status, time periods, cluster, nodes, severity, or message text. The filter settings can be saved for later reloading. If the dialog opened by this button is simply given an arbitrary name, and the `Ok` button clicked on to accept the default values, then a default event viewer tabbed pane is added to `cmgui`.

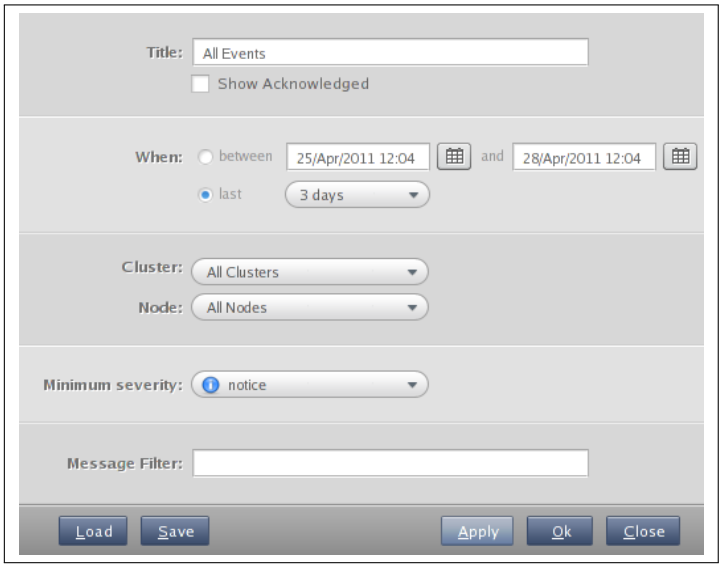

Figure 10.35: cmgui Monitoring: Event Viewer Filter Dialog

- `set event viewer filter dialog`: Adjusts an existing filter with a similar dialog to the `new event viewer filter dialog`.
- `acknowledge event`: Sets the status of one or more selected events in the log to "acknowledged". They are then no longer seen, unless the filter setting for the `show acknowledged` checkbox is checked in the `set event filter` option.

10.7 The monitoring Modes Of cmsh

This section covers how to use cmsh to configure monitoring. The monitoring mode in cmsh is how metrics and health checks are configured from the command line, and corresponds to the configuration carried out by cmgui in section 10.4.

Visualization of data similar to how cmgui does it in section 10.3 can also be done from cmsh's command line, via its device mode. Graphs can be obtained from cmsh by piping values returned by device mode commands such as latestmetricdata (section 10.8.1) and dumpmetricdata (section 10.8.2) into graphing utilities. These techniques are not covered in this chapter.

Familiarity is assumed with handling of objects as described in the introduction to working with objects (section 3.5.3). When using cmsh's monitoring mode, the properties of these objects—the details of the monitoring settings—are the parameters and values which are accessed and manipulated from the monitoring mode hierarchy within cmsh.

The monitoring "mode" of cmsh gives access to 4 modes under it.

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring help | tail -5
============================== Monitoring ==============================
actions ..................... Enter threshold actions mode
healthchecks ................ Enter healthchecks mode
metrics ..................... Enter metrics mode
```

```
setup ....................... Enter monitoring configuration setup mode
```

These 4 modes are regarded as being the top level monitoring related modes:

- `monitoring actions`
- `monitoring healthchecks`
- `monitoring metrics`
- `monitoring setup`

The word `monitoring` is therefore merely a grouping label prefixed inseparably to these 4 modes. The syntax of the 4 bulleted commands above is thus consistent with that of the other top level `cmsh` modes.

The sections 10.7.1, 10.7.2, 10.7.3, and 10.7.4 give examples of how objects are handled under these 4 monitoring modes. To avoid repeating similar descriptions, section 10.7.1 is relatively detailed, and is often referred to by the other sections.

10.7.1 The `monitoring actions` Mode In `cmsh`

The `monitoring actions` mode of `cmsh` corresponds to the `cmgui` `actions` tab of section 10.4.6.

The `monitoring actions` mode handles actions objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle action objects—the properties associated with an action script or action built-in—might be to view the actions available, or to add a custom action for use by, for example, a metric or health check.

This section continues the `cmsh` session started above, giving examples of how the `monitoring actions` mode is used.

The `monitoring actions` Mode In `cmsh`: `list`, `show`, And `get`

The `list` command by default lists the names and command scripts available in `monitoring actions` mode:

Example

```
[myheadnode]% monitoring actions
[myheadnode->monitoring->actions]% list
Name (key)               Command
------------------------ ----------------------------------------------
Drain node               <built-in>
Power off                <built-in>
Power on                 <built-in>
Power reset              <built-in>
Reboot                   <built-in>
SendEmail                <built-in>
Shutdown                 <built-in>
Undrain node             <built-in>
killprocess              /cm/local/apps/cmd/scripts/actions/killprocess.+
remount                  /cm/local/apps/cmd/scripts/actions/remount
testaction               /cm/local/apps/cmd/scripts/actions/testaction
```

The above shows the actions available on a newly installed system. The details of what they do are covered in appendix H.3.1.

The show command of cmsh displays the parameters and values of a specified action:

Example

```
[myheadnode->monitoring->actions]% show poweroff
Parameter                      Value
------------------------------ ------------------------------------------------
Command                        <built-in>
Description                    Power off the device
Name                           Power off
Revision
Run on                         master
Timeout                        5
isCustom                       no
[myheadnode->monitoring->actions]%
```

The meanings of the parameters are covered in appendix H.3.2.

Tab-completion suggestions with the show command suggest arguments corresponding to names of action objects:

Example

```
[myheadnode->monitoring->actions]% show
```

A double-tap on the tab key to get tab-completions suggestions for show in the above displays the following:

Example

```
drainnode    killprocess  poweron      reboot       sendemail    testaction
killallyes   poweroff     powerreset   remount      shutdown     undrainnode
```

The Power off action name, for example, corresponds with the argument poweroff. By default, the arguments are the action names in lower case, with the spaces removed. However, they are space- and case-insensitive, so typing in show "Power off" with the quotes included to pass the space on is also valid.

The get command returns the value of an individual parameter of the action object:

Example

```
[myheadnode->monitoring->actions]% get poweroff runon
master
[myheadnode->monitoring->actions]%
```

The monitoring actions Mode In cmsh: add, use, remove, commit, refresh, modified, set, clear, And validate

In the basic example of section 10.1, in "Adding The Action To The Actions List", the name, description and command for an action were added via a dialog in the Actions tab of cmgui.

The equivalent is done in cmsh with add and set commands. The add command adds an object, makes it the current object, and sets its name at the same time; while the set command sets values.

If there is no killallyes action already, then the name is added in the actions mode with the add command as follows:

Example

```
[myheadnode->monitoring->actions]% add killallyes
[myheadnode->monitoring->actions*[killallyes*]]%
```

The converse to the `add` command is the `remove` command, which removes the action.

The `use` command is the usual way of "using" an object, where "using" means that the object being used is referred to by default by any command run. So if the `killallyes` object already exists, then `use killallyes` drops into the context of an already existing object (i.e. it "uses" the object).

The `set` command sets the value of each parameter displayed by a `show` command:

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% set description "kill \
            all yes processes"
```

The `clear` command is the converse of `set`, and removes any value for a given parameter.

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% clear command
```

The `validate` command checks if the object has all required values set to sensible values. The commands `refresh`, `modified` and `commit` work as expected from the introduction to working with objects (section 3.5.3). So, for example, `commit` only succeeds if the `killallyes` object passes validation.

Example

```
[myheadnode->monitoring->actions*[killallyes*]]% validate
Code  Field                    Message
----- ---------------------- ------------------------------------------------
4     command                  command should not be empty
```

Here validation fails because the parameter `Command` has no value set for it yet. This is remedied with `set` acting on the parameter (some prompt text elided for display purposes):

Example

```
[...*]]% set command "/cm/local/apps/cmd/scripts/actions/killallyes"
[...*]]% commit
[...]]%
```

Validation then succeeds and the `commit` successfully saves the `killallyes` object.

Note that validation does not check if the script itself exists. It solely does a sanity check on the values of the parameters of the object, which is another issue. If the `killallyes` script does not yet exist in the location given by the parameter, it can be created as suggested in the basic example of section 10.1, in "Setting Up The Kill Action".

10.7.2 The monitoring healthchecks Mode in cmsh

The monitoring healthchecks mode of cmsh corresponds to the cmgui Health Checks tab of section 10.4.5.

The monitoring healthchecks mode handles health check objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle health check objects—the properties associated with an health check script or health check built-in—might be to view the health checks already available, or to add a health check for use by a device resource.

This section goes through a cmsh session giving some examples of how this mode is used and to illustrate what it looks like.

The monitoring healthchecks Mode in cmsh: list, show, And get

In monitoring healthchecks mode, the list command by default lists the names of the health check objects along with their command scripts:

Example

```
[bright52->monitoring->healthchecks]% format name:18 command:55
[bright52->monitoring->healthchecks]% list
name (key)         command
------------------ -------------------------------------------------------
DeviceIsUp         <built-in>
ManagedServicesOk  <built-in>
chrootprocess      /cm/local/apps/cmd/scripts/healthchecks/chrootprocess
cmsh               /cm/local/apps/cmd/scripts/healthchecks/cmsh
diskspace          /cm/local/apps/cmd/scripts/healthchecks/diskspace
...
```

The format command, introduced in section 3.5.3, is used here with the given column width values to avoid truncating the full path of the commands in the display.

The above example shows a truncated list of health checks that can be set for sampling on a newly installed system. The details of what these health checks do is covered in appendix H.2.1.

The show command of cmsh displays the parameters and values of a specified health check:

Example

```
[myheadnode->monitoring->healthchecks]% show deviceisup
Parameter                Value
------------------------ ------------------------------------------------
Class of healthcheck     internal
Command                  <built-in>
Description              Returns PASS when device is up, closed or insta+
Disabled                 no
Extended environment     no
Name                     DeviceIsUp
Notes                    <0 bytes>
Only when idle           no
Parameter permissions    disallowed
Revision
Sampling method          samplingonmaster
State flapping count     7
```

```
Timeout                 5
Valid for               node,master,pdu,ethernet,myrinet,ib,racksensor,+
[myheadnode->monitoring->healthchecks]%
```

The meanings of the parameters are covered in appendix H.2.2.

As detailed in section 10.7.1, tab-completion suggestions for the `show` command suggest arguments corresponding to names of objects that can be used in this mode. For `show` in `healthchecks` mode, tab-completion suggestions give the following as possible health check objects:

Example

```
[myheadnode->monitoring->healthchecks]% show
chrootprocess      failover           mysql              ssh2node
cmsh               hardware-profile   ntp                swraid
deviceisup         interfaces         portchecker        testhealthcheck
diskspace          ldap               rogueprocess
exports            managedservicesok  schedulers
failedprejob       mounts             smart
[myheadnode->monitoring->healthchecks]% show
```

The `get` command returns the value of an individual parameter of a particular health check object:

Example

```
[myheadnode->monitoring->healthchecks]% get deviceisup description
Returns PASS when device is up, closed or installing
[myheadnode->monitoring->healthchecks]%
```

The `monitoring healthchecks` Mode In `cmsh`: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, And `validate`

The remaining commands in `monitoring healthchecks` mode: `add`, `use`, `remove`, `commit`, `refresh`, `modified`, `set`, `clear`, and `validate`; all work as outlined in the introduction to working with objects (section 3.5.3). More detailed usage examples of these commands within a `monitoring` mode are given in `Cmsh Monitoring Actions` (section 10.7.1).

In the basic example of section 10.1, a metric script was set up from `cmgui` to check if thresholds were exceeded, and if so, to launch an action.

A functionally equivalent task can be set up by creating and configuring a health check, because metrics and health checks are so similar in concept. This is done here to illustrate how `cmsh` can be used to do something similar to what was done with `cmgui` in the basic example. A start is made on the task by creating a health check object and setting its values using the `monitoring healthchecks` mode of `cmsh`. The task is completed in the section on the `monitoring setup` mode in section 10.7.4.

To start the task, `cmsh`'s `add` command is used to create the new health check object:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring healthchecks
[myheadnode->monitoring->healthchecks]% add cpucheck
[myheadnode->monitoring->healthchecks*[cpucheck*]]%
```

The set command sets the value of each parameter displayed by a show command (some prompt text elided for layout purposes):

Example

```
[...]% set command /cm/local/apps/cmd/scripts/healthchecks/cpucheck
[...]% set description "CPUuser under 50%?"
[...]% set parameterpermissions disallowed
[...]% set samplingmethod samplingonmaster
[...]% set validfor master
[...]]% commit
```

Since the cpucheck script does not yet exist in the location given by the parameter command, it needs to be created:

```
#!/bin/bash

## echo PASS if CPUUser < 50
## cpu is a %, ie: between 0 and 100

cpu=‘mpstat 1 1 | tail -1 | awk ’{print $3}’‘
comparisonstring="$cpu"" < 50"

if (( $(bc <<< "$comparisonstring") )); then
  echo PASS
else
  echo FAIL
fi
```

The script should be placed in the location suggested by the object, /cm/local/apps/cmd/scripts/healthchecks/cpucheck, and made executable with a chmod 700.

The cpucheck object is handled further within the cmsh monitoring setup mode in section 10.7.4 to produce a fully configured health check.

10.7.3 The monitoring metrics Mode In cmsh

The monitoring metrics mode of cmsh corresponds to the cmgui metrics tab of section 10.4.4.

The monitoring metrics mode of cmsh handles metrics objects in the way described in the introduction to working with objects (section 3.5.3). A typical reason to handle metrics objects—the properties associated with a metrics script or metrics built-in—might be to view the configuration metrics already being used for sampling by a device category, or to add a metric for use by a device category.

This section goes through a cmsh session giving some examples of how this mode is used and to illustrate its behavior.

The monitoring metrics Mode In cmsh: list, show, And get

In metrics mode, the list command by default lists the names and command scripts available for setting for device categories:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring metrics
```

```
[myheadnode->monitoring->metrics]% list
Name (key)                  Command
--------------------------- ---------------------------------------------
AlertLevel                  <built-in>
AvgExpFactor                <built-in>
AvgJobDuration              <built-in>
...
```

The above shows a truncated list of the metrics that may be used for sampling on a newly installed system. What these metrics do is described in appendix H.1.1.

The `show` command of `cmsh` displays the parameters and values of a specified metric:

Example

```
[myheadnode->monitoring->metrics]% show cpuuser
Parameter                      Value
------------------------------ ----------------------------------------------
Class of metric                cpu
Command                        <built-in>
Cumulative                     yes
Description                    Total core usage in user mode per second
Disabled                       no
Extended environment           no
Maximum                        <range not set>
Measurement Unit
Minimum                        <range not set>
Name                           CPUUser
Notes                          <15 bytes>
Only when idle                 no
Parameter permissions          disallowed
Retrieval method               cmdaemon
Revision
Sampling method                samplingonnode
State flapping count           7
Timeout                        5
Valid for                      node,master
[myheadnode->monitoring->metrics]%
```

The meanings of the parameters above are explained in appendix H.1.2.

Tab-completion suggestions for the `show` command suggest arguments corresponding to names of objects (the names returned by the `list` command) that may be used in a monitoring mode. For `metrics` mode, `show`, followed by a double-tap on the tab key, displays a large number of possible metrics objects:

Example

```
[myheadnode->monitoring->metrics]% show
Display all 130 possibilities? (y or n)
alertlevel                droprecv                  ipoutrequests
avgexpfactor              dropsent                  ipreasmoks
avgjobduration            errorsrecv                ipreasmreqds
await_sda                 errorssent                loadfifteen
...
```

The get command returns the value of an individual parameter of a particular metric object:

Example

```
[myheadnode->monitoring->metrics]% get CPUUser description
Total core usage in user mode per second
```

The monitoring metrics Mode In cmsh: add, use, remove, commit, refresh, modified, set, clear, And validate

The remaining commands in monitoring metrics mode: add, use, remove, commit, refresh, modified, set, clear, and validate; all work as outlined in the introduction to working with objects (section 3.5.3). More detailed usage examples of these commands within a monitoring mode are given in Cmsh Monitoring Actions (section 10.7.1).

Adding a metric collections script to the framework is possible from this point in cmsh too. Details on how to do this are given in appendix I.

10.7.4 The monitoring setup Mode in cmsh

The cmsh monitoring setup mode corresponds to the cmgui Metric Configuration and Health Check Configuration tabs of sections 10.4.2 and 10.4.3.

The monitoring setup mode of cmsh, like the Metric Configuration and the Health Check Configuration tabs of cmgui, is used to select a device category. Properties of metrics or of health checks can then be configured for the selected device category. These properties are the configuration of the sampling parameters themselves (for example, frequency and length of logs), but also the configuration of related properties such as thresholds, consolidation, actions launched when a metric threshold is crossed, and actions launched when a metric or health state is flapping.

The setup mode only functions in the context of metrics or health checks, and therefore these contexts under the setup mode are called submodes. On a newly installed system, a list command from the monitoring setup prompt displays the following account of metrics and health checks that are in use by device categories:

Example

```
[root@myheadnode ~]# cmsh
[myheadnode]% monitoring setup
[myheadnode->monitoring->setup]% list
Category               Metric configuration   Health configuration
---------------------- ---------------------- ----------------------
Chassis                <2 in submode>         <1 in submode>
EthernetSwitch         <13 in submode>        <1 in submode>
GenericDevice          <2 in submode>         <1 in submode>
GpuUnit                <3 in submode>         <0 in submode>
IBSwitch               <2 in submode>         <1 in submode>
MasterNode             <90 in submode>        <14 in submode>
MyrinetSwitch          <2 in submode>         <1 in submode>
PowerDistributionUnit  <5 in submode>         <1 in submode>
RackSensor             <2 in submode>         <1 in submode>
default                <35 in submode>        <11 in submode>
```

```
[myheadnode->monitoring->setup]%
```

A device category must always be used when handling the properties of the metrics and health checks configurable under the submodes of `monitoring setup`. The syntax of a configuration submode, `metricconf` or `healthconf`, therefore requires the device category as a mandatory argument, and tab-completion suggestions become quite helpful at this point.

Examples are now given of how the metric configuration `metricconf` and health check configuration `healthconf` submodes are used:

The `monitoring setup` Mode in `cmsh`: `metricconf`

Continuing with the session above, the `metricconf` option can only be used with a device category specified. Tab-completion suggestions for metricconf suggest the following possible device categories:

Example

```
[myheadnode->monitoring->setup]% metricconf
chassis                 gpuunit                 powerdistributionunit
default                 ibswitch                racksensor
ethernetswitch          masternode
genericdevice           myrinetswitch
```

A category can be chosen with the `use` command, and `show` shows the properties of the category. With a category selected, the `metricconf` or `healthconf` submodes can then be invoked:

Example

```
[myheadnode->monitoring->setup]% use masternode
[myheadnode->monitoring->setup[MasterNode]]% show
Parameter                      Value
------------------------------ ----------------------------------------------
Category                       MasterNode
Health configuration           <14 in submode>
Metric configuration           <90 in submode>
Normal pickup interval         180
Revision
Scrutiny pickup interval       60
[myheadnode->monitoring->setup[MasterNode]]% metricconf
[myheadnode->monitoring->setup[MasterNode]->metricconf]%
```

Dropping into a submode—in the example given, the `metricconf` submode—could also have been done directly in one command: `metricconf mastermode`. The synopsis of the command in the example is actually `[[[monitoring] setup] metricconf] masternode`, where the optional parts of the command are invoked depending upon the context indicated by the prompt. The example below clarifies this (some prompt text elided for display purposes):

Example

```
[...->monitoring->setup[MasterNode]->metricconf]% exit; exit; exit; exit
[...]% monitoring setup metricconf masternode
[...->monitoring->setup[MasterNode]->metricconf]% exit; exit; exit
[...->monitoring]% setup metricconf masternode
[...->monitoring->setup[MasterNode]->metricconf]% exit; exit
[...->monitoring->setup]% metricconf masternode
[...->monitoring->setup[MasterNode]->metricconf]% exit
[...->monitoring->setup[MasterNode]]% metricconf
[...->monitoring->setup[MasterNode]->metricconf]%
```

A list of metrics that have been set to do sampling for the device category masternode is obtained with list. Since there are many of these, only 10 lines are displayed in the list shown below by piping it through head:

Example

```
[myheadnode->monitoring->setup[MasterNode]->metricconf]% list | head
Metric                  Metric Param       Samplinginterval
----------------------- ------------------ ------------------------
AlertLevel              max                0
AlertLevel              sum                0
AvgExpFactor                               120
AvgJobDuration          defq               60
BufferMemory                               120
BytesRecv               eth0               120
BytesRecv               eth1               120
BytesSent               eth0               120
BytesSent               eth1               120
CMDMemUsed                                 120
```

Besides list, an alternative way to get a list of metrics that are set to sample for masternode is to use the tab-completion suggestions to the use command.

The use command is normally used to drop into the configuration properties of the metric so that parameters of the metric object can be configured:

Example

```
[myheadnode->monitoring->setup[MasterNode]->metricconf]% use cpuuser
[myheadnode->monitoring->setup[MasterNode]->metricconf[CPUUser]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Consolidators                  <3 in submode>
Disabled                       no
GapThreshold                   2
LogLength                      3000
Metric                         CPUUser
MetricParam
Only when idle                 no
Revision
Sampling Interval              120
Stateflapping Actions
Store                          yes
ThresholdDuration              1
```

```
Thresholds                     <1 in submode>
[myheadnode->monitoring->setup[MasterNode]->metricconf[CPUUser]]%
```

The add command adds a metric to be set for sampling for the device category. The list of all possible metrics that can be added to the device category can be seen with the command monitoring metrics list, or more conveniently, simply with tab-completion suggestions to the add command at the [...metricconf]% prompt in the above example.

The above example indicates that there are two submodes for each metric configuration: Consolidators and Thresholds. Running the consolidators or thresholds commands brings cmsh into the chosen submode.

Consolidation and threshold manipulation only make sense in the context of a metric configuration, so at the metricconf prompt in the example above (before use cpuuser is executed), the commands thresholds cpuuser or consolidators cpuuser can be executed as more direct ways of getting to the chosen submode.

The thresholds submode If, continuing on from the above example, the thresholds submode is entered, then the list command lists the existing thresholds. If the basic example of section 10.1 has already been carried out on the system, then a threshold called killallyesthreshold is already there with an assigned action killallyes. The properties of each threshold can be shown (some prompt text elided for layout purposes):

Example

```
[...metricconf]% thresholds
[...metricconf[CPUUser]]% thresholds
[...metricconf[CPUUser]]->thresholds]% list
Name (key)              Bound           Severity
----------------------- --------------- -----------------------
killallyesthreshold     50              10
[...metricconf[CPUUser]]->thresholds]% show killallyesthreshold
Parameter                      Value
------------------------------ ------------------------------------------------
Actions                        enter: killallyes()
Bound                          50
Name                           killallyesthreshold
Revision
Severity                       10
UpperBound                     yes
```

The meanings of the parameters are explained in the GUI equivalent of the above example in section 10.4.2 in the section labeled "Metric Configuration: Thresholds Options". The object manipulation commands introduced in section 3.5.3 work as expected at this cmsh prompt level: add and remove add and remove a threshold; set, get, and clear set and get values for the parameters of each threshold; refresh and commit revert and commit changes; use "uses" the specified threshold, making it the default for commands; validate applied to the threshold checks if the threshold object has sensible values; and append and removefrom append an action to, and remove an action from, a specified threshold.

The append and removefrom commands correspond to the ⊕ and ⊖ widgets of cmgui in figure 10.24 and work with parameters that can have multiple values. For example, a sendemail action with a parameter root can be appended to the Actions parameter, which already has the killallyes action as a value. This sends an e-mail to the root mail account. A get command can be run to see the values for the threshold actions:

Example

```
[...->thresholds*]% append killallyesthreshold actions sendemail root
[...->thresholds*]% get killallyesthreshold actions
enter: killallyes()
enter: SendEmail(root)
```

By default, the actions are set to run on entering the threshold zone, with an implied flag of "-e|--enter". To run on leaving the threshold zone, or to run during the time the value is within the threshold zone, the flags "-l|--leave" or "-d|--during" must explicitly be applied to the actions command.

In the example, the "Actions" parameter now has the value of the the built-in action name, sendemail, as well as the value of the action script name, killallyes. This means that both actions run when the threshold condition is met.

The consolidators submode If, continuing on with the preceding example, the consolidators submode is entered, then the list command lists the consolidators running on the system. On a newly installed system there are three consolidators by default for each metric set for a device category. Each consolidator has an appropriately assigned time Interval, in seconds. The show command shows the parameters and values of a specific consolidator:

Example

```
[...metricconf[CPUUser]->thresholds*]% exit
[...metricconf[CPUUser]]% consolidators
[...metricconf[CPUUser]->consolidators]% list
Name (key)               Length     Interval
------------------------ ---------- ----------
Day                      1000       86400
Hour                     2000       3600
Week                     1000       604800
[...metricconf[CPUUser]->consolidators]% show day
Parameter                      Value
------------------------------ ------------------------------------------------
Interval                       86400
Kind                           AVERAGE
Length                         1000
Name                           Day
Offset                         0
Revision
```

The meanings of the parameters are explained in the GUI equivalent of the above example in section 10.4.2 in the section labeled "Metric Configuration: Consolidators Options".

The object manipulation commands introduced in section 3.5.3 work as expected at this `cmsh` prompt level: `add` and `remove` add and remove a consolidator; `set`, `get`, and `clear` set and get values for the parameters of each consolidator; `refresh` and `commit` revert and commit changes; `use` "uses" the specified consolidator, making it the default for commands; and `validate` applied to the consolidator checks if the consolidator object has sensible values.

The `monitoring setup` Mode in `cmsh`: `healthconf`

The `healthconf` submode is the alternative to the `metricconf` submode under the main `monitoring setup` mode. Like the `metricconf` option, `healthconf` too can only be used with a device category specified.

If the session above is continued, and the device category `masternode` is kept unchanged, then the `healthconf` submode can be invoked with:

```
[...metricconf[CPUUser]->consolidators]% exit; exit; exit
[myheadnode->monitoring->setup[MasterNode]]% healthconf
[...healthconf]%
```

Alternatively, the `healthconf` submode with the `masternode` device category could also have been reached from `cmsh`'s top level prompt by executing `monitoring setup healthconf masternode`.

The health checks set to do sampling in the device category `masternode` are listed:

Example

```
[myheadnode->monitoring->setup[MasterNode]->healthconf]% list
HealthCheck              HealthCheck Param  Check Interval
------------------------ ------------------ ------------------------
DeviceIsUp                                  120
ManagedServicesOk                           120
chrootprocess                               900
cmsh                                        1800
diskspace                2% 10% 20%         1800
exports                                     1800
failedprejob                                900
failover                                    1800
interfaces                                  1800
ldap                                        1800
mounts                                      1800
mysql                                       1800
ntp                                         300
schedulers                                  1800
smart                                       1800
```

The `use` command would normally be used to drop into the health check object. However `use` can also be an alternative to the `list` command, since tab-completion suggestions to the `use` command get a list of currently configured health checks for the `masternode` too.

The `add` command adds a health check into the device category. The list of all possible health checks that can be added to the category can be

seen with the command monitoring healthchecks list, or more conveniently, simply with tab-completion suggestions to the add command.

At the end of section 10.7.2 a script called cpucheck was built. This script was part of a task to use health checks instead of metric threshold actions to set up the functional equivalent of the behavior of the basic example of section 10.1. In this section the task is continued and completed, and on the way how to use the health checks configuration object methods to do this is shown.

First, the script is added, and as usual when using add, the prompt drops into the level of the added object. The show command acting on the object displays the following default values for its parameters (some prompt text elided for display purposes):

Example

```
[...[MasterNode]->healthconf]% add cpucheck
[...*[MasterNode*]->healthconf*[cpucheck*]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Check Interval                 120
Disabled                       no
Fail Actions
Fail severity                  10
GapThreshold                   2
HealthCheck                    cpucheck
HealthCheckParam
LogLength                      3000
Only when idle                 no
Pass Actions
Revision
Stateflapping Actions
Store                          yes
ThresholdDuration              1
Unknown Actions
Unknown severity               10
[...*[MasterNode*]->healthconf*[cpucheck*]]%
```

The details of what these parameters mean is covered in section 10.4.3 where the edit and add dialog options for a health check state shown in figure 10.28 are explained.

The object manipulation commands introduced in section 3.5.3 work as expected at the healthconf prompt level in the example above: add and remove add and remove a health check; set, get, and clear set and get values for the parameters of each health check; refresh and commit revert and commit changes; use "uses" the specified health check, making it the default for commands; and validate applied to the health check checks if the health check object has sensible values; and append and removefrom append an action to, and remove an action from, a specified health check action parameter.

The append and removefrom commands correspond to the ⊕ and ⊖ widgets of cmgui in figure 10.28 and work with parameters that can have multiple values:

The action killallyes was set up to be carried out with the metric CPUUser in the basic example of section 10.1. The action can also be car-

ried out with a FAIL response for the cpucheck health check by using append command:

Example

```
[...healthconf*[cpucheck*]]% append failactions killallyes
[...healthconf*[cpucheck*]]%
```

Sending an email to root can be done by appending further:

Example

```
[...healthconf*[cpucheck*]]% append failactions sendemail root
[...healthconf*[cpucheck*]]% get failactions
enter: SendEmail(root)
enter: killallyes()
[...healthconf*[cpucheck*]]%
```

10.8 Obtaining Monitoring Data Values

The monitoring data values that are logged by devices can be used to generate graphs using the methods in section 10.3. However, sometimes an administrator would like to have the data values that generate the graphs instead, perhaps in order to import them into a spreadsheet for further direct manipulation, or pipe them into a utility such as gnuplot.

The values can be obtained from within device mode in various ways.

10.8.1 The Latest Data Values—The latest*data Commands

Within device mode, the following commands display the latest metrics and health checks being monitored, along with their latest data:
For metrics:

- metrics: The metrics command in device mode lists the metrics that are currently configured to be monitored for a specified device. These correspond to the metrics shown in cmgui in the "Metric Configuration" tab for a specific device.
- latestmetricdata: The latestmetricdata command for a particular device displays the most recent value that has been obtained by the monitoring system for each item in the list of active metrics.

Similarly for health checks,

- healthchecks: The healthchecks command in device mode lists the health checks that are currently configured to be monitored for a device. These correspond to the health checks shown in cmgui in the "Health Check Configuration" tab for a device.
- latesthealthchecks: The latesthealthdata command for a particular device displays the most recent value that has been obtained by the monitoring system for each item in the list of active health checks.

Using The `metrics` And `healthchecks` Commands

When using the `metrics` or `healthchecks` command, the device must be specified (some output elided):

Example

```
[root@bright52 ~]# cmsh
[bright52]% device
[bright52->devices]% metrics node001
LoadOne
LoadFive
LoadFifteen
PageFaults
MajorPageFaults
Uptime
MemoryUsed
...
```

Using The `latestmetricdata` And `latesthealthdata` Commands

When using the `latestmetricdata` or `latesthealthdata` commands, the device must be specified (some output elided):

Example

```
[bright52->device]% use node001
[bright52->device[node001]]% latestmetricdata
Metric                  Value            Age (sec.) Info Message
----------------------- ---------------- ---------- ---------------
AlertLevel:max          30               76         FAIL schedulers
AlertLevel:sum          30               76         FAIL schedulers
BytesRecv:BOOTIF        690.733          76
BytesSent:BOOTIF        449.1            76
CPUIdle                 99.0333          76
CPUIrq                  0                76
CPUNice                 0                76
...
```

10.8.2 Data Values Over Time—The `dump*` Commands

Within `device` mode, the following commands display monitoring data values over a specified period:

- `dumpmetricdata`: The `dumpmetricdata` command for a particular device displays the values of metrics obtained over a specific period by the monitoring system for a particular metric.
- `dumphealthdata`: The `dumphealthdata` command for a particular device displays the values of health checks obtained over a specific period by the monitoring system for a particular health check.
- `dumpstatistics`: The `dumpstatistics` command for a particular device displays statistics obtained over a specific period by the monitoring system for a particular metric item in the list of checks.

Using The dumpmetricdata And dumphealthdata Commands

A concise overview of the dumpmetricdata or dumphealthdata commands can be displayed by, for example, typing in "help dumpmetricdata" in the device mode of cmsh.

The usage of the dumpmetricdata and dumphealthdata commands is:

```
dumpmetricdata [OPTIONS] <start-time> <end-time> <metric> [device]
```

The mandatory arguments for the times, the metric being dumped, and the device or devices being sampled, have values that are specified as follows:

- The metric item <metric> for which the data values are being gathered must always be given. Metrics currently in use can conveniently be listed by running the metrics command (section 10.8.1).
- If [device] is not specified when running the dumpmetricdata or dumphealthdata command, then it must either be set by specifying the object from the device mode of cmsh (for example, with "use node001"), or by specifying it in the options. The options allow specifying, amongst others, a list, a group, or a category of devices.
- The time pair <start-time> or <end-time> can conveniently be specified as follows:
  - now: That is, the time at which the dumpmetricdata or dumphealthdata command is run
  - for only one item in the time pair, its time can be set relative to the other (the fixed time item). The non-fixed time item (the relative time item) is specified as follows:
    - A <start-time> <*number*> value is prefixed with a "-"
    - an <end-time> <*number*> value is prefixed with "+"
    - The <*number*> values have suffix values indicating the units of time, as seconds (s), minutes (m), hours (h), or days (d).

    This is summarized in the following table:

Unit	<start-time>	<end-time>
seconds:	-<*number*>s	+<*number*>s
minutes:	-<*number*>m	+<*number*>m
hours:	-<*number*>h	+<*number*>h
days:	-<*number*>d	+<*number*>d

  - YY/MM/DD
  - HH:MM
  - HH:MM:SS
  - unix epoch time (seconds since 00:00:00 1 January 1970)

An example of how the preceding mandatory arguments to the dumpmetricdata command might be used is:

Example

```
[bright52->device]% dumpmetricdata  -10m now cpuidle node001
# From Thu Oct 20 15:32:41 2011 to Thu Oct 20 15:42:41 2011
Time                        Value            Info Message
--------------------------- ---------------- --------------
Thu Oct 20 15:32:41 2011    96.3481
Thu Oct 20 15:34:00 2011    95.3167
Thu Oct 20 15:36:00 2011    93.4167
Thu Oct 20 15:38:00 2011    93.7417
Thu Oct 20 15:40:00 2011    92.2417
Thu Oct 20 15:42:00 2011    93.5083
```

The options applied to the samples are specified as follows:

Option	Argument(s) (if any)	Description
`-n, --nodes`	<*list*>	list of nodes
`-c, --categories`	<*list*>	list of categories
`-g, --groups`	<*list*>	list of groups
`-r, --racks`	<*list*>	list of racks
`-h, --chassis`	<*list*>	list of chassis
`-i, --intervals`	<*number*>	number of samples to show
`-k, --kind`	`average, min, max`	show the average (default), minimum, or maximum of set of stores values, out of the list of device samples
`-u, --unix`		use a unix timestamp instead of using the default date format
`-v, --verbose`		show the rest of the line on a new line instead of cutting it off

The `-i` option interpolates the data that is to be displayed to a specified number <*number*> of interpolated samples over the given time range. Using "`-i 0`" outputs only the non-interpolated stored samples, and is the default.

Example

```
[bright52->device]% dumpmetricdata -i 0 -38m now loadone node001
# From Thu Oct 20 17:22:12 2011 to Thu Oct 20 17:59:12 2011
Time                        Value            Info Message
--------------------------- ---------------- --------------
Thu Oct 20 17:22:12 2011    0
Thu Oct 20 17:48:00 2011    0
Thu Oct 20 17:50:00 2011    0.03
Thu Oct 20 17:52:00 2011    0.02
Thu Oct 20 17:54:00 2011    0.08
Thu Oct 20 17:56:00 2011    0.07
```

```
Thu Oct 20 17:58:00 2011   0.69
[bright52->device]% dumpmetricdata -n node001..node002 -5m now cpuidle
# From Fri Oct 21 16:52:41 2011 to Fri Oct 21 16:57:41 2011
Time                       Value            Info Message
-------------------------- ---------------- ---------------
node001
Fri Oct 21 16:52:41 2011   99.174
Fri Oct 21 16:54:00 2011   99.3167
Fri Oct 21 16:56:00 2011   99.1333
node002
Fri Oct 21 16:52:41 2011   98.1518
Fri Oct 21 16:54:00 2011   99.3917
Fri Oct 21 16:56:00 2011   99.1417
[bright52->device]%
```

When a sample measurement is carried out, if the sample has the same value as the two preceding it in the records, then the "middle" sample is discarded from storage for performance reasons.

Thus, when viewing the sequence of output of non-interpolated samples, identical values do not exceed two entries one after the other.

Using The `dumpstatistics` Commands

The usage of the `dumpstatistics` command is:

```
dumpstatistics[OPTIONS] <start-time> <end-time> <metric> [device]
```

and it follows the pattern of earlier (page 294) for the `dumpmetricdata` and `dumphealthdata` commands.

There are two significant differences:

- The `-p` option sets percentile boundaries. By default, the statistics are divided into percentiles of 25%.
- The `-i` option is set by default to a value of 20. A setting of `i 0` (which displays non-interpolated values when used with `dumpmetricdata` and `dumphealthdata`) is not a valid value for `dumpstatistics`.

Example

```
[bright52->device]% dumpstatistics -i 5 -1h -p 100 now loadone -n node00\
1..node003
Start                      End                        100%
-------------------------- -------------------------- ------
Fri Oct 21 17:04:30 2011   Fri Oct 21 17:16:30 2011   0.11
Fri Oct 21 17:16:30 2011   Fri Oct 21 17:28:30 2011   0.08
Fri Oct 21 17:28:30 2011   Fri Oct 21 17:40:30 2011   0.09
Fri Oct 21 17:40:30 2011   Fri Oct 21 17:52:30 2011   0.27
Fri Oct 21 17:52:30 2011   Fri Oct 21 18:04:30 2011   0.22
[bright52->device]% dumpstatistics -i 5 -u -1h now loadone -n node001..n\
ode003
Start      End        0%      25%     50%     75%     100%
---------- ---------- ------  ------  ------  ------  ------
1319444362 1319445082 0       0       0.03    0.0475  0.07
1319445082 1319445802 0       0       0       0.0025  0.04
1319445802 1319446522 0       0       0       0.01    0.06
1319446522 1319447242 0       0       0       0.01    0.08
1319447242 1319447962 0       0       0       0.015   0.05
```

10.9 The User Portal

10.9.1 Accessing The User Portal

The user portal allows users to login via a browser and view the state of the cluster themselves. It is a read-only interface.

The first time a browser is used to login to the portal, a warning about the site certificate being untrusted appears.

The certificate is a self-signed certificate (the x509v3 certificate of section 4.1), generated and signed by Bright Computing, and the attributes of the cluster owner are part of the certificate. However, Bright Computing is not a recognized Certificate Authority (CA) according to the standard CAs that are recognized by a browser, which is why the warning appears.

Portals Isolated From The Outside

For a portal that is not accessible from the outside world, such as the internet, this warning is not an issue, and the user can simply accept the "untrusted" certificate.

Indeed, for such portals, the administrator may wish to do away with user authentication for the portal as well. A way to bypass authentication in such a case is to change the file `/var/www/html/header.php` so that one line is added as indicated:

```
...
Session()->set("username", "cmsupport"); // THIS LINE IS ADDED
if(!Session()->get("username")) {
$current = Input()->getCurrentPage();
$current = urlencode($current);
redirect("login.php?redirect=$current");
}
...
```

Portals Accessible From The Outside

For a portal that is accessible via the internet, some administrators may regard it as more secure to ask users to trust the self-signed certificate rather than external certificate authorities.

Alternatively the adminstrator can replace the self-signed certificate with one obtained by a standard CA certificate, if that is preferred.

External LDAP Service

The file `/var/www/html/login.php` has the following lines as default values:

```
$ldapHost = "ldap://localhost";
$ldapPort = "389";
$ldapBase = "dc=cm, dc=cluster";
```

These values are valid for the internal LDAP server provided by Bright Cluster Manager for LDAP authentication. However, the values in these lines must be changed appropriately if an external LDAP service (section 7.3) is used for authentication instead.

10.9.2 User Portal Home Page

The default user portal home page allows a quick glance to convey the most important cluster-related information for users (figure 10.36):

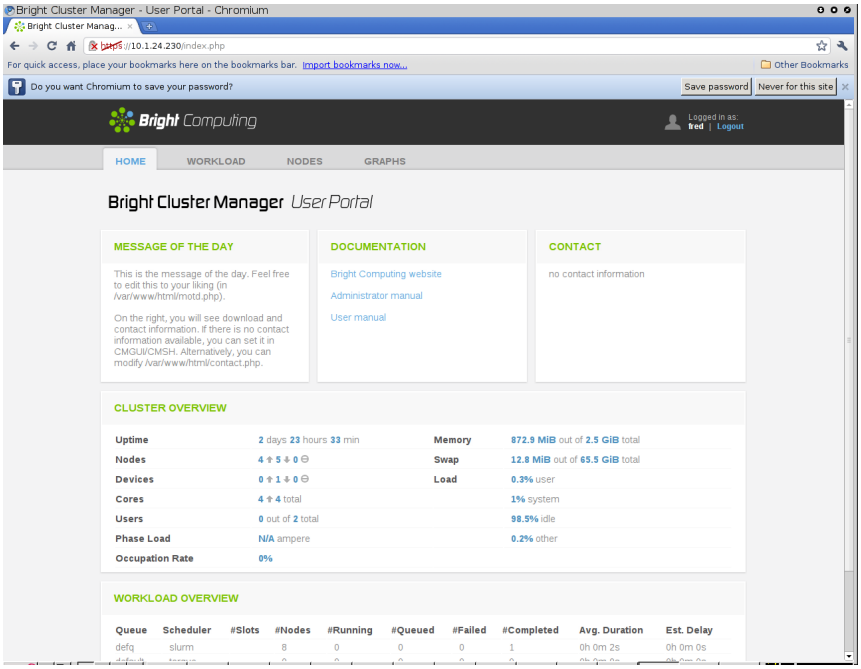

Figure 10.36: User Portal: Default Home Page

The following items are displayed on the home page:

- a Message Of The Day. This can be edited in `/var/www/html/motd.php`
- links to the documentation for the cluster
- contact information. This can be edited in `/var/www/html/contact.php`
- an overview of the cluster state, carried out by displaying the values of items in the cluster. These are a subset of the items seen in the `Overview` tab of the cluster in `cmgui` in figure 3.4
- a workload overview. This is a table displaying a summary of queues and their associated jobs

The user portal from the point of view of a user is described further in the *User Manual*.

11

Day-to-day Administration

This chapter discusses several tasks that may come up in day-to-day administration of a cluster running Bright Cluster Manager.

11.1 Parallel Shell

The cluster management tools include the parallel shell execution command, `pexec`. This allows `bash` commands to be run on a group of nodes simultaneously.

The `pexec` command can be run from the OS shell (`bash` by default), or from the CMDaemon (`cmsh` or `cmgui`) front ends. The name `pexec` is the same in both these cases, but the syntax differs slightly.

- In the OS shell, running `pexec` without any arguments or options displays the following help text:

```
------------------------------------------------------------------------
Cluster commands v1.3

------------------------------------------------------------------------

Options:
  \*                    use wildcard for file selection
  --                    end of parallel commandline options
  -v                    verbose
  -a                    do not broadcast to see which nodes are alive
  -b                    background execution (not supported by pcopy)
  -t=time(s)            timeout in seconds, default is 10 seconds, 0
                        disables timeout
  -d=directory          change the working directory
  -g=nodegroup          group selection
  -n=node               single node selection
  -n=node001,node010    node list selection (node001 and node010)
  -n=node001..node010   node range selection (node001 through node010)
  -c                    prevent colored output
  -e                    only execute on nodes running the installer
  -m                    execute on all nodes, including nodes running
                        the installer
  -u=url                alternative master CMDaemon URL

Commands:
```

```
pexec <command>         execute <command>

pcopy <f1..fn> <dir>    copy files and/or directories to <dir>. Add '/'
                        to the end of a directory in order to
                        transfer only the contents of a directory

pkilluser               as root: user-only command, it should not be
                                 used by root
                        as user: kill processes owned by user

------------------------//////////////////////////---------------------
```

Some of the less obvious options are explained here:

- *: Not actually an option, but a reminder to escape the asterisk from the OS shell if using wildcards. The following are equivalent:

  Example

  ```
  pexec ls "*"
  pexect ls \*
  ```

- --: After the list of parallel commands is run, -- is used as a marker to indicate the pexec command has ended, and that anything that continues on the same line is no longer part of the pexec just ended. Thus, in the following command:

  ```
  pexec ls -- ; ls
  ```

  it means that the first ls is executed in parallel as part of pexec, while the second ls is run only from the current OS session.

- -a: By default a broadcast ping is used to check what nodes are alive. The -a option tries to run the command on all nodes (head and regular) without running the check.

- -u=url: This can be useful in the odd case of CMDaemon not running on the default ports. Appendix C includes a description of the port directives which allow non-default ports to be used.

- In cmsh, the pexec command is run from device mode:

  Example

  ```
  [bright52->device]% pexec -n node001,node002 "cd ; ls"

  [node001] :
  anaconda-ks.cfg
  install.log
  install.log.syslog


  [node002] :
  anaconda-ks.cfg
  install.log
  install.log.syslog
  ```

- In `cmgui`, it is executed from the `Parallel Shell` tab after selecting the cluster from the resource tree (figure 11.1):

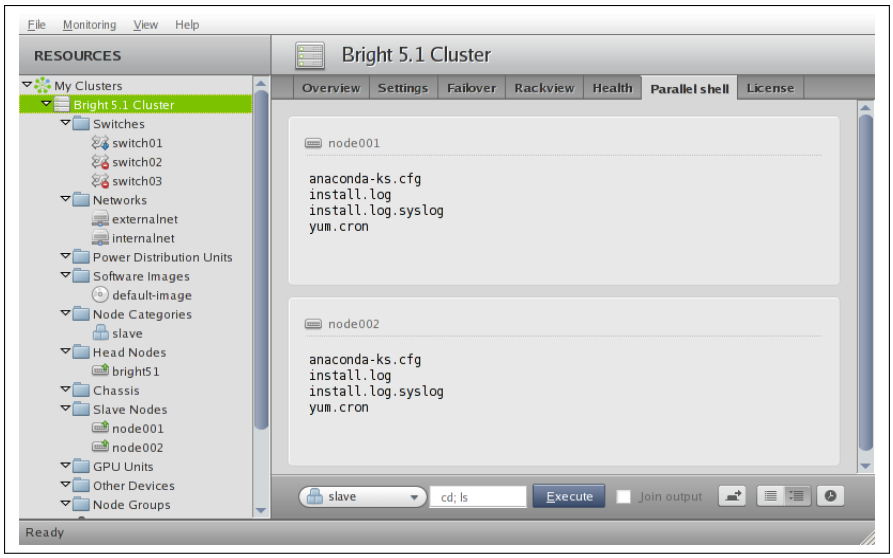

Figure 11.1: Executing Parallel Shell Commands

11.2 Getting Support With Cluster Manager Issues

Bright Cluster Manager is constantly undergoing development. While the result is a robust and well-designed cluster manager, it is possible that the administrator may run into a feature that requires help from Bright Computing or Bright Computing's resellers. This section describes how to get support for such issues.

11.2.1 Getting Support From The Reseller

If the Bright Cluster Manager software was obtained through a reseller or system integrator, then the first line of support is provided by the reseller or system integrator. The reseller or system integrator in turn contacts the Bright Computing support department if 2nd or 3rd level support is required.

11.2.2 Getting Support From Bright Computing

If the Bright Cluster Manager software was purchased directly from Bright Computing, then `support@brightcomputing.com` can be contacted for all levels of support.

Bright Computing provides the `cm-diagnose` and the `request-remote-assistance` utilities to help resolve issues.

If the issue appears to be a bug, then a bug report should be submitted. It is helpful to include as many details as possible to ensure the development team is able to reproduce the apparent bug. The policy at Bright Computing is to welcome such reports, to provide feedback to the reporter, and to resolve the problem.

Reporting Cluster Manager Diagnostics With `cm-diagnose`

The diagnostic utility `cm-diagnose` is run from the head node and gathers data on the system that may be of use in diagnosing issues. To view its

options and capabilities, it can be run as "`cm-diagnose --help`".

When run without any options, it runs interactively, and allows the administrator to send the resultant diagnostics file to Bright Computing directly. The output of a `cm-diagnose` session looks something like the following (the output has been made less verbose for easy viewing):

Example

```
[root@bright52 ~]# cm-diagnose
Describing the issue helps us diagnose it.
Do you want to enter a description? [Y/n]

End input with ctrl-d
I tried X, Y, and Z on the S2464 motherboard. When that didn't work, I
tried A, B, and C, but the florbish is grommicking.
Thank you.

If issues are suspected to be in the cmdaemon process, a gdb trace of
that process is useful.
In general such a trace is only needed if Bright Support asks for this.
Do you want to create a gdb trace of the running CMDaemon? [y/N]

Proceed to collect information? [Y/n]

Processing master
    Processing commands
    Processing file contents
    Processing large files and log files
    Collecting process information for CMDaemon
    Executing CMSH commands
    Finished executing CMSH commands

Processing default-image
    Processing commands
    Processing file contents

Creating log file: /root/bright52_Bright_5.2_Cluster_954.tar.gz

Cleaning up

Automatically submit diagnostics to
http://support.brightcomputing.com/cm-diagnose/ ? [Y/n] y

Uploaded file: bright52_Bright_5.2_Cluster_954.tar.gz
Remove log file (/root/bright52_Bright_5.2_Cluster_954.tar.gz)?  [y/N] y
[root@bright52 ~]
```

Requesting Remote Support With `request-remote-assistance`

The `request-remote-assistance` utility allows a Bright Computing engineer to securely tunnel into the cluster without a change in firewall or `ssh` settings of the cluster. For the utility to work, it should be allowed to access the `www` and `ssh` ports of Bright Computing's internet servers.

Administrators familiar with `screen` may wish to run it within a screen session. It is run as follows:

Example

```
[root@bright52 ~]# request-remote-assistance

This tool helps securely set up a temporary ssh tunnel to
sandbox.brightcomputing.com.

Allow a Bright Computing engineer ssh access to the cluster? [Y/n]

Enter additional information for Bright Computing (eg: related
ticket number, problem description)? [Y/n]

End input with ctrl-d
Ticket 1337 - the florbish is grommicking

Thank you.

Added temporary Bright public key.
```

The screen clears, and the secure tunnel opens up, displaying the following notice:

```
REMOTE ASSISTANCE REQUEST
############################################################
A connection has been opened to Bright Computing Support.
Closing this window will terminate the remote assistance
session.
------------------------------------------------------------

Hostname: bright52.NOFQDN
Connected on port: 7000

ctrl-c to terminate this session
```

Bright Computing support automatically receives an e-mail alert that an engineer can now securely tunnel into the cluster. The session activity is not explicitly visible to the administrator. When the engineer has ended the session, the administrator may remove the secure tunnel with a `ctrl-c`, and the display then shows:

```
Tunnel to sandbox.brightcomputing.com terminated.
Removed temporary Bright public key.
[root@bright52 ~]#
```

The Bright Computing engineer is then no longer able to access the cluster.

11.3 Backups

11.3.1 Cluster Installation Backup

Bright Cluster Manager does not include facilities to create backups of a cluster installation. When setting up a backup mechanism, it is recommended that the full file-system of the head node (i.e. including all software images) is backed up. Unless the node hard drives are used to store important data, it is not necessary to back up nodes.

If no backup infrastructure is already in place at the cluster site, the following open source (GPL) software packages may be used to maintain regular backups:

- **Bacula**: Bacula is a mature network based backup program that can be used to backup to a remote storage location. If desired, it is also possible to use Bacula on nodes to back up relevant data that is stored on the local hard drives. More information is available at `http://www.bacula.org`
- **rsnapshot**: rsnapshot allows periodic incremental file system snapshots to be written to a local or remote file system. Despite its simplicity, it can be a very effective tool to maintain frequent backups of a system. More information is available at `http://www.rsnapshot.org`.

11.3.2 Local Database Backups And Restoration

The CMDaemon database is stored in the MySQL `cmdaemon` database, and contains most of the stored settings of the cluster.

The CMDaemon monitoring database is stored in a separate MySQL `cmdaemon_mon` database.

The administrator is expected to run a regular backup mechanism for the cluster to allow restores of all files from a recent snapshot. As an additional, separate, convenience:

- For the CMDaemon database, the entire database is also backed up nightly on the cluster filesystem itself ("local rotating backup") for the last 7 days.
- For the monitoring database, the monitoring database raw data records are not backed up locally, since these can get very large, but the rest of the database is backed up for the last 7 days too.

Database Corruption Messages And Repairs

A corrupted MySQL database is commonly caused by an improper shutdown of the node. To deal with this, when starting up, MySQL checks itself for corrupted tables, and tries to repair any such by itself. Detected corruption causes an event notice to be sent to `cmgui` or `cmsh`.

When there is database corruption, info messages in the `/var/log/cmdaemon` log may mention:

- "`Unexpected eof found`" in association with a table in the database,
- "`can't find file`" when referring to an entire missing table,
- locked tables,
- error numbers from table handlers,
- "`Error while executing`" a command.

An example of an event message on a head node `icarus`:

Example

```
[icarus]%
Fri Jan 18 10:33:15 2012 [notice] icarus: Error when reading data from m\
onitoring database. No monitoring data will be saved. (details: Incorrec\
t key file for table './cmdaemon_mon/MonData.MYI'; try to repair it)
[icarus]%
```

The associated messages logged in /var/log/cmdaemon may show something like:

Example

```
Jan 18 10:31:05 icarus CMDaemon: Info: Reconnect command processed.
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting main mysql \
buffer thread
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting mirroring t\
hread
Jan 18 10:31:05 icarus CMDaemon: Info: Preloading mysql key cache using \
query: LOAD INDEX INTO CACHE MonMetaData
Jan 18 10:31:05 icarus CMDaemon: Info: MysqlBuffer: starting MysqlBuffer\
 thread
Jan 18 10:31:05 icarus CMDaemon: Info: Database: Mirroring required to r\
remote master
Jan 18 10:31:05 icarus CMDaemon: Info: Database: generalQuery returned: \
cmdaemon_mon.MonMetaData preload_keys status OK
Jan 18 10:31:05 icarus CMDaemon: Info: Preloading mysql key cache using \
query: LOAD INDEX INTO CACHE MonData INDEX (MonDataIndex) IGNORE LEAVES
Jan 18 10:31:05 icarus CMDaemon: Info: Database: generalQuery returned: \
cmdaemon_mon.MonData preload_keys Error Unexpected eof found when readin\
g file '/var/lib/mysql/cmdaemon_mon/MonData.MYI' (Errcode: 0) cmdaemon_m\
on.MonData preload_keys status OK
```

Looking through the preceding messages, the conclusion is that the monitoring database has a corrupted MonData table. Being in MyISAM format (.MYI) means the myisamchk repair tool can be run on the table, for example as:

```
[root@icarus ~]# service cmd stop
[root@icarus ~]# myisamchk --recover /var/lib/mysql/cmdaemon_mon/MonData\
.MYI
[root@icarus ~]# service cmd start
```

If basic repair fails, more extreme repair options—man myisamchk(1) suggests what—can then be tried out next.

Another example: If CMDaemon is unable to start up due to a corrupted database, then messages in the /var/log/cmdaemon file might show something like:

Example

```
Oct 11 15:48:19 solaris CMDaemon: Info: Initialize cmdaemon database
Oct 11 15:48:19 solaris CMDaemon: Info: Attempt to set provisioningNetwo\
rk (280374976710700) not an element of networks
Oct 11 15:48:19 solaris CMDaemon: Fatal: Database corruption! Load Maste\
rNode with key: 280374976782569
Oct 11 15:48:20 solaris CMDaemon: Info: Sending reconnect command to all\
 nodes which were up before master went down ...
Oct 11 15:48:26 solaris CMDaemon: Info: Reconnect command processed.
```

Here it is CMDaemon's "Database corruption" message that the administrator should be aware of, and which suggests database repairs are required for the CMDaemon database. The severity of the corruption, in this case not even allowing CMDaemon to start up, may mean a restoration from backup is needed. How to restore from backup is explained in the next section.

Restoring From The Local Backup

MySQL has several database repair tools. If these do not fix the problem, then, for a failover configuration, the `dbreclone` option (section 13.3.2) should normally provide a database that is current.

If the head node is not part of a failover configuration, then a restoration from local backup can be done. The local backup directory is `/var/spool/cmd/backup`, with contents that look like (some text elided):

Example

```
[root@solaris ~]# cd /var/spool/cmd/backup/
[root@solaris backup]# ls -l
total 280
...
-rw------- 1 root root  1593 Oct 10 04:02 backup-monitor-Mon.sql.gz
-rw------- 1 root root  1593 Oct  9 04:02 backup-monitor-Sun.sql.gz
...
-rw------- 1 root root 33804 Oct 10 04:02 backup-Mon.sql.gz
-rw------- 1 root root 33805 Oct  9 04:02 backup-Sun.sql.gz
-rw------- 1 root root 33805 Oct 11 04:02 backup-Tue.sql.gz
...
```

The CMDaemon database snapshots are stored as `backup-<day of week>.sql.gz` while the monitoring database snapshots are stored as `backup-monitor-<day of week>.sql.gz`. In the example, the latest backup available in the listing for CMDaemon turns out to be `backup-Tue.sql.gz`

The latest backup can then be ungzipped and piped into the MySQL database for the user `cmdaemon`. The password, <*password*>, can be retrieved from `/cm/local/apps/cmd/etc/cmd.conf`, where it is configured in the DBPass directive (Appendix C).

Example

```
gunzip backup-Tue.sql.gz
service cmd stop #(just to make sure)
mysql -ucmdaemon -p<password> cmdaemon < backup-Tue.sql
```

Running "`service cmd start`" should have CMDaemon running again, this time with a restored database from the time the snapshot was taken.

A similar procedure for monitoring database backups can be used to restore the table structure and info messages of the data records from local backup. It does not restore the monitoring data records themselves. To restore records, the administrator would need to retrieve a snapshot from the regular backup that the administrator should be running for the cluster.

11.4 BIOS Configuration And Updates

Bright Cluster Manager includes a number of tools that can be used to configure and update the BIOS of nodes. All tools are located in the

/cm/shared/apps/cmbios/nodebios directory on the head node. The remainder of this section assumes that this directory is the current working directory.

Due to the nature of BIOS updates, it is highly recommended that these tools are used with great care. Incorrect use may render nodes unusable.

Updating a BIOS of a node requires booting it from the network using a specially prepared DOS image. From the autoexec.bat file, one or multiple automated BIOS operations can be performed.

11.4.1 BIOS Configuration

In order to configure the BIOS on a group of nodes, an administrator needs to manually configure the BIOS on a reference node using the conventional method of entering BIOS Setup mode at system boot time. After the BIOS has been configured, the machine needs to be booted as a node. The administrator may subsequently use the cmospull utility on the node to create a snapshot of the reference node's NVRAM contents.

Example

```
ssh node001 /cm/shared/apps/cmbios/nodebios/cmospull > node001.nvram
```

After the NVRAM settings of the reference node have been saved to a file, the settings need to be copied to the generic DOS image so that they can be written to the NVRAM of the other nodes.

The generic DOS image is located in /cm/shared/apps/cmbios/nodebios/win98boot.img. It is generally a good idea to copy the generic image and make changes to the copy only.

Example

```
cp -a win98boot.img flash.img
```

To modify the image, it is first mounted:

```
mount -o loop flash.img /mnt
```

When the DOS image has been mounted, the utility that writes out the NVRAM data needs to be combined with the NVRAM data into a single DOS executable. This is done by appending the NVRAM data to the cmosprog.bin file. The result is a DOS .COM executable.

Example

```
cat cmosprog.bin node001.nvram > cmosprog.com
```

The generated .COM is then copied to the image and should be started from the autoexec.bat file. Note that DOS text files require a carriage return at the end of every line.

Example

```
cp cmosprog.com /mnt
/bin/echo -e "A:\\\cmosprog.com\r" >> /mnt/autoexec.bat
```

After making the necessary changes to the DOS image, it is unmounted:

```
umount /mnt
```

After preparing the DOS image, it is booted as described in section 11.4.3.

11.4.2 Updating BIOS

Upgrading the BIOS to a new version involves using the DOS tools that were supplied with the BIOS. Similar to the instructions above, the flash tool and the BIOS image must be copied to the DOS image. The file `autoexec.bat` should be altered to invoke the flash utility with the correct parameters. In case of doubt, it can be useful to boot the DOS image and invoke the BIOS flash tool manually. Once the correct parameters have been determined, they can be added to the `autoexec.bat`.

After a BIOS upgrade, the contents of the NVRAM may no longer represent a valid BIOS configuration because different BIOS versions may store a configuration in different formats. It is therefore recommended to also write updated NVRAM settings immediately after flashing a BIOS image (section 11.4.1).

The next section describes how to boot the DOS image.

11.4.3 Booting DOS Image

To boot the DOS image over the network, it first needs to be copied to software image's `/boot` directory, and must be world-readable.

Example

```
cp flash.img /cm/images/default-image/boot/bios/flash.img
chmod 644 /cm/images/default-image/boot/bios/flash.img
```

An entry is added to the PXE boot menu to allow the DOS image to be selected. This can easily be achieved by modifying the contents of `/cm/images/default-image/boot/bios/menu.conf`, which is by default included automatically in the PXE menu. By default, one entry `Example` is included in the PXE menu, which is however invisible as a result of the `MENU HIDE` option. Removing the `MENU HIDE` line will make the BIOS flash option selectable. Optionally the `LABEL` and `MENU LABEL` may be set to an appropriate description.

The option `MENU DEFAULT` may be added to make the BIOS flash image the default boot option. This is convenient when flashing the BIOS of many nodes.

Example

```
LABEL FLASHBIOS
  KERNEL memdisk
  APPEND initrd=bios/flash.img
  MENU LABEL ^Flash BIOS
#  MENU HIDE
  MENU DEFAULT
```

The `bios/menu.conf` file may contain multiple entries corresponding to several DOS images to allow for flashing of multiple BIOS versions or configurations.

11.5 Hardware Match Check

Often a large number of identical nodes may be added to a cluster. In such a case it is a good practice to check that the hardware matches what is expected. This can be done easily as follows:

1. The new nodes, say node129 to node255, are committed to a newly created category newnodes as follows (output truncated):

```
[root@bright52 ~]# cmsh -c "category add newnodes; commit"
[root@bright52 ~]# for i in {129..255}
> do
> cmsh -c "device; set node00$i category newnodes; commit"
> done
Successfully committed 1 Devices
Successfully committed 1 Devices
```

2. The hardware profile of one of the new nodes, say node129, is saved into the category newnodes. This is done using the node-hardware-profile health check (Appendix H.2.1) as follows:

```
[root@bright52 ~]# /cm/local/apps/cmd/scripts/healthchecks/node-hardwar\
e-profile -n node129 -s newnodes
```

   The profile is intended to be the reference hardware against which all the other nodes should match.

3. The frequency with which the health check should run in normal automated periodic use is set as follows (some prompt text elided):

```
[root@bright52 ~]# cmsh
[bright52]% monitoring setup healthconf newnodes
[...->healthconf]% add hardware-profile
[...->healthconf*[hardware-profile*]]% set checkinterval 600; commit
```

4. The cmdaemon then automatically alerts the administrator if one of the nodes does not match the hardware of that category during the first automated check. In the unlikely case that the reference node is itself faulty, then that will also be obvious because all—or almost all, if more nodes are faulty—of the other nodes in that category will then be reported "faulty" during the first check.

12

Third Party Software

In this chapter, several third party software packages included in the Bright Cluster Manager repository are described briefly. For all packages, references to the complete documentation are provided.

12.1 Modules Environment

Package name: `env-modules` for head node

The *modules environment* (`http://modules.sourceforge.net/`) allows a user of a cluster to modify the shell environment for a particular application or even a particular version of an application. Typically, a module file defines additions to environment variables such as `PATH`, `LD_LIBRARY_PATH`, and `MANPATH`. Cluster users use the `module` command to load or remove modules from their environment. The `module(1)` man page has more detail on the command.

Aspects of the modules environment that are relevant for administrators are discussed in section 3.2, while the modules environment from a user's perspective is covered in the Bright Cluster Manager *User Manual*.

All module files are located in the `/cm/local/modulefiles` and `/cm/shared/modulefiles` trees. A module-file is a TCL script in which special commands are used to define functionality. The `modulefile(1)` man page has more on this.

Cluster administrators can use the existing modules files as a guide to creating their own modules for module environments, and can copy and modify a file for their own software if there is no environment provided for it already by Bright Cluster Manager.

12.2 Shorewall

Package name: `shorewall`

Bright Cluster Manager uses Shoreline Firewall (more commonly known as "Shorewall") package to provide firewall and gateway functionality on the head node of a cluster. Shorewall is a flexible and powerful high-level interface for the netfilter packet filtering framework inside the 2.4 and 2.6 Linux kernels. Behind the scenes, Shorewall uses the standard `iptables` command to configure netfilter in the kernel. All aspects of firewall and gateway configuration are handled through the configuration files located in `/etc/shorewall`. Shorewall does not run as a daemon

process, but rather exits immediately after configuring netfilter through `iptables`. After modifying Shorewall configuration files, Shorewall must be run again to have the new configuration take effect:

```
service shorewall restart
```

In the default setup, Shorewall provides gateway functionality to the internal cluster network on the first network interface (`eth0`). This network is known as the `nat` zone to Shorewall. The external network (i.e. the connection to the outside world) is assumed to be on the second network interface (`eth1`). This network is known as the `net` zone in Shorewall. Letting Bright Cluster Manager take care of the network interfaces settings is recommended for all interfaces on the head node (section 4.2). The `interfaces` file is generated by the cluster management daemon, and any extra instructions that cannot be added via `cmgui` or `cmsh` can be added outside of the file section clearly demarcated as being maintained by CMDaemon.

Shorewall is configured by default (through `/etc/shorewall/policy`) to deny all incoming traffic from the `net` zone, except for the traffic that has been explicitly allowed in `/etc/shorewall/rules`. Providing (a subset of) the outside world with access to a service running on a cluster, can be accomplished by creating appropriate rules in `/etc/shorewall/rules`. By default, the cluster responds to ICMP ping packets and allows SSH access from the whole world. Depending on site policy, access to port 8081 may also be enabled to allow access to the cluster management daemon.

To remove all rules, for example for testing purposes, the `clear` option should be used. This then allows all network traffic through:

```
shorewall clear
```

Administrators should be aware that in Red Hat distribution variants the "`service shorewall stop`" command corresponds to the "`shorewall stop`" command, and not to the "`shorewall clear`" command. The "`stop`" option blocks network traffic but allows a pre-defined minimal "safe" set of connections, and is not the same as completely removing Shorewall from consideration. This differs from Debian-like distributions where "`service shorewall stop`" corresponds to "`shorewall clear`" and removes Shorewall from consideration.

Administrators should also be aware that distributions such as Red Hat and SUSE run their own set of high-level iptable setup scripts if the standard distribution firewall is enabled. Since Bright Cluster Manager requires Shorewall, then, to avoid conflict, the standard distribution firewall must stay disabled. Shorewall can be configured to set up whatever iptable rules are installed by the standard distribution script instead.

Full documentation on Shorewall is available at `http://www.shorewall.net`.

12.3 Compilers

Bright Computing provides convenient RPM packages for several compilers that are popular in the HPC community. All of those may be installed through `yum` or `zypper` (section 9.2) but (with the exception of GCC) require an installed license file to be used.

12.3.1 GCC

Package name: `gcc-recent`

The GCC suite that the distribution provides is also present by default.

12.3.2 Intel Compiler Suite

Package names:

Intel Compiler Suite Versions

2011	2013
intel-compiler-common	intel-compiler-common-2013
intel-compiler-common-32	intel-compiler-common-2013-32
intel-cc	intel-cc-2013
intel-cc-32	intel-cc-2013-32
intel-fc	intel-fc-2013
intel-fc-32	intel-fc-2013-32
intel-idb	intel-idb-2013
intel-idb-32	intel-idb-2013-32
intel-ipp	intel-ipp-2013
intel-ipp-32	intel-ipp-2013-32
intel-itac	intel-itac-2013
intel-mkl	intel-mkl-2013
intel-mkl-32	intel-mkl-2013-32
intel-openmp-2013	intel-openmp-2013
intel-openmp-32	intel-openmp-2013-32
intel-sourcechecker	intel-sourcechecker-2013
intel-sourcechecker-32	intel-sourcechecker-2013-32
intel-tbb	intel-tbb-2013

The Intel compiler packages are provided as a suite, for example by Intel® C++ Composer XE. The 2013 version of the suite can be installed concurrently with the 2011 version. However, the administrator should be aware when doing this that if a short module (section 3.2) name is used, such as `intel/compiler`, then the compiler version with the highest character value folowing the short module name is the one that is used.

Typically, the suite includes the Intel Fortran (indicated by `fc`) and Intel C++ compilers (part of the C compiler package, indicated by `cc`). Along with the 64-bit (i.e. EM64T) version of both compilers, the 32-bit version may optionally be installed. The 32-bit packages have package names ending in "`-32`"

Both the 32-bit and 64-bit versions can be invoked through the same set of commands. The modules environment (section 3.2) provided when installing the packages can be loaded accordingly, to select one of the two versions. For the C++ and Fortran compilers the 64-bit and 32-bit modules are called `intel/compiler/64` and `intel/compiler/32` respectively.

The Intel compilers include a debugger, which can also be accessed by loading the `intel/compiler/64` or `intel/compiler/32` module. The fol-

lowing commands can be used to run the Intel compilers and debugger:

- `icc`: Intel C/C++ compiler
- `ifort`: Intel Fortran 90/95 compiler
- `idb`: Intel Debugger

Optional packages are:

- `intel-ipp`: Integrated Performance Primitives
- `intel-mkl`: Math Kernel Library
- `intel-itac`: Trace Analyzer And Collector
- `intel-sourcechecker`: Source Checker
- `intel-tbb`: Threading Building Blocks

A short summary of a package can be shown using, for example: "`yum info intel-fc`".

The compiler packages require a license, obtainable from Intel, and placed in `/cm/shared/licenses/intel`.

Full documentation for the Intel compilers is available at `http://software.intel.com/en-us/intel-compilers/`.

12.3.3 PGI High-Performance Compilers

Package name: `pgi`

The PGI compiler package contains the PGI C++ and Fortran 77/90/95 compilers.

- `pgcc`: PGI C compiler
- `pgCC`: PGI C++ compiler
- `pgf77`: PGI Fortran 77 compiler
- `pgf90`: PGI Fortran 90 compiler
- `pgf95`: PGI Fortran 95 compiler
- `pgdbg`: PGI debugger

Full documentation for the PGI High-Performance Compilers is available at `http://www.pgroup.com/resources/docs.htm`.

12.3.4 AMD Open64 Compiler Suite

Package name: `open64`

The Open64 Compiler Suite contains optimizing C++ and Fortran compilers.

- `opencc`: Open64 C compiler
- `openCC`: Open64 C++ compiler
- `openf90`: Open64 Fortran 90 compiler
- `openf95`: Open64 Fortran 95 compiler

Full documentation for the AMD Open64 Compiler Suite is available at: `http://www.amd.com`.

12.3.5 FLEXlm License Daemon

Package name: `flexlm`

For the Intel and PGI compilers a FLEXlm license must be present in the `/cm/shared/licenses` tree.

For workstation licenses, i.e. a license which is only valid on the head node, the presence of the license file is typically sufficient.

However, for floating licenses, i.e. a license which may be used on several machines, possibly simultaneously, the FLEXlm license manager, `lmgrd`, must be running.

The `lmgrd` service serves licenses to any system that is able to connect to it through the network. With the default firewall configuration, this means that licenses may be checked out from any machine on the internal cluster network. Licenses may be installed by adding them to `/cm/shared/licenses/lmgrd/license.dat`. Normally any FLEXlm license starts with the following line:

```
SERVER hostname MAC port
```

Only the first FLEXlm license that is listed in the `license.dat` file used by `lmgrd` may contain a `SERVER` line. All subsequent licenses listed in `license.dat` should have the `SERVER` line removed. This means in practice that all except for the first licenses listed in `license.dat` start with a line:

```
DAEMON name /full/path/to/vendor-daemon
```

The `DAEMON` line must refer to the vendor daemon for a specific application. For PGI the vendor daemon (called `pgroupd`) is included in the `pgi` package. For Intel the vendor daemon (called `INTEL`) must be installed from the `flexlm-intel`.

Installing the `flexlm` package adds a system account `lmgrd` to the password file. The account is not assigned a password, so it cannot be used for logins. The account is used to run the lmgrd process. The `lmgrd` service is not configured to start up automatically after a system boot, but can be configured to do so with:

```
chkconfig lmgrd on
```

The `lmgrd` service is started manually with:

```
/etc/init.d/lmgrd start
```

The `lmgrd` service logs its transactions and any errors to `/var/log/lmgrd.log`.

More details on FLEXlm and the `lmgrd` service are available at `http://www.rovicorp.com`.

12.4 Intel Cluster Checker

Package name: `intel-cluster-checker`

Intel Cluster Checker is a tool that checks the health of the cluster and verifies its compliance against the requirements defined by the Intel Cluster Ready Specification. This section lists the steps that must be taken to certify a cluster as Intel Cluster Ready.

For additional instructions on using Intel Cluster Checker and its test modules for a particular version <*version*>, the tool documentation located in the cluster at /opt/intel/clck/<*version*>/doc/ can be referred to. The URL http://software.intel.com/en-us/cluster-ready/ has more information on the Intel Cluster Ready (ICR) program.

12.4.1 Package Installation

Package Installation: Other Required Packages

The Intel Cluster Checker tool is provided by the intel-cluster-checker package. To meet all the Intel Cluster Ready specification requirements the following software packages also need to be installed on the head and regular nodes:

- intel-cluster-runtime
- cm-config-intelcompliance-master
- cm-config-intelcompliance-slave

Package Installation: Where The Packages Go

The intel-cluster-checker and intel-cluster-runtime packages are installed only on the head node, although libraries are available to the regular nodes through the shared file system. Packages cm-config-intelcompliance-master and cm-config-intelcompliance-slave are installed on the head node and software images respectively.

Package Installation: Installing The Packages With A Package Manager

The packages are normally already installed by default on a standard Bright Cluster Manager cluster. If they are not installed then the packages can be installed using yum (or zypper if using SLES 11).

Example

```
[root@mycluster ~]# yum install intel-cluster-runtime intel-cluster-che\
cker cm-config-intelcompliance-master
[root@mycluster ~]# chroot /cm/images/default-image
[root@mycluster /]# yum install cm-config-intelcompliance-slave
```

The packages guarantee through package dependencies that all Intel Cluster Ready package requirements are satisfied. If the package manager reports that any additional packages need to be installed, simply agreeing to install them is enough to satisfy the requirements.

Package Installation: Updating The Nodes

After installing the necessary packages the nodes need to be updated. This can be done with an updateprovisioners command (if there are node provisioners in the cluster) followed by an imageupdate command.

12.4.2 Preparing Configuration And Node List Files

The configuration and package list files are located in the /etc/intel/clck directory:

- config-ib.xml
- config-nonib.xml

- packagelist.head
- packagelist.node

The input files, containing a list of the nodes to be checked, are created in the /home/cmsupport/intel-cluster-ready directory:

- nodelist
- nodelist.ib

These files are used during the cluster checker execution. During the cluster checker preparation, the nodelist and nodelist.ib files must be generated. During the first boot of a head nodethe package list files packagelist.head and packagelist.node are generated.

Configuration Files

The config-nonib.xml and config-ib.xml files are default configuration files that have been included as part of the cm-config-intelcompliance-master package. Both configuration files may need small modifications based on the cluster for which certification is required.

The configuration file can be copied to the user's home directory and edited as needed. The adjusted configuration file name needs to be provided to the Intel cluster checker command as an argument. Otherwise, the tool uses /etc/intel/clck/config.xml by default.

For the certification run, two configuration files are available:

- config-nonib.xml
- config-ib.xml

During the first boot of the head node, the /etc/intel/clck/config.xml link is created.

- If no configuration file is provided when running the cluster check, then /etc/intel/clck/config.xml is used as the configuration file
- If the cluster has no InfiniBand interconnect, then /etc/intel/clck/config.xml links to the config-nonib.xml file
- If the cluster uses an InfiniBand interconnect, then /etc/intel/clck/config.xml links to the config-ib.xml file

The existence of a link and where it points to can be checked as follows:

```
[root@mycluster ~]# ls -l /etc/intel/clck/config.xml
```

The file or link /etc/intel/clck/config.xml can be changed if needed.

Although it is not required for an ICR certification, several performance thresholds can be defined which require tuning based on the hardware that is included in the cluster.

When in doubt, it can be useful to configure threshold values which are certainly too high in performance for the cluster to meet. For example, too high a throughput for disk I/O bandwidth, or too low a time in the

case of latency. After running the cluster checker, a (failed) value for the concerned performance parameters will be given, and the performance thresholds can then be adjusted to more realistic numbers based on the results obtained from the run.

Intel Cluster Checker can also be run with the `--autoconfigure` option for automatic configuration, in which case a basic configuration is written to an existing configuration file before the execution starts.

Node Lists

The `nodelist` and `nodelist.ib` files list the nodes which are considered by the Intel Cluster Checker. In the normal case `nodelist` is used. When an InfiniBand interconnect is used in the cluster, the `nodelist.ib` file can be used to run the cluster check entirely over InfiniBand. When the cluster changes, the node lists files must be regenerated with the `clck-prepare` command.

Updating the Node Lists

The `clck-prepare` command is used to generate or update the node lists files. The `cmsupport` account is used to generate the files, since the `cmsupport` account is used to perform the cluster check run. For clusters without InfiniBand interconnect, the `nodelist.ib` file is not generated.

Example

```
[root@mycluster ~]# su - cmsupport
[cmsupport@mycluster ~]$ clck-prepare
Created non InfiniBand node list file /home/cmsupport/intel-cluster-rea\
dy/nodelist

Created InfiniBand node list file /home/cmsupport/intel-cluster-ready/n\
odelist.ib
```

Package Lists

The package list files `packagelist.head` and `packagelist.node` contain lists of all packages installed on the head node and on the regular nodes. These lists are used to ensure that the same software is available on all nodes. The package lists are created on the first boot, and do not change unless explicitly regenerated.

Regenerating Package Lists

The package list files are generated during the first boot of the head node. The following example regenerates the package list files, if this is needed.

Example

```
[root@mycluster ~]# module load intel-cluster-checker
[root@mycluster ~]# clck-prepare
[root@mycluster ~]# cluster-check --packages
Packages installed in host mycluster.cm.cluster saved in file mycluster\
.cm.cluster-20111013.123523.list.
Packages installed in host node001.cm.cluster saved in file node001.cm.\
cluster-20111013.123523.list.
[root@mycluster ~]# mv mycluster.cm.cluster-20111013.123523.list /etc/i\
ntel/clck/packagelist.head
```

```
[root@mycluster ~]# mv node001.cm.cluster-20111013.123523.list /etc/int\
el/clck/packagelist.node
```

The "cluster-check --packages" command creates the package list of the head node and a regular node, in this case node001, in the current working directory. The package list of the head node is moved to /etc/intel/clck/packagelist.head, and the package list of a regular node is moved to /etc/intel/clck/packagelist.node.

12.4.3 Running Intel Cluster Checker

The cmsupport account, which is part of a default installation, is used to perform the cluster check run.

The following commands start the cluster checker:

```
[root@mycluster ~]# su - cmsupport
[cmsupport@mycluster ~]$ module initadd intel-cluster-runtime
[cmsupport@mycluster ~]$ module load intel-cluster-runtime
[cmsupport@mycluster ~]$ cluster-check --certification
```

The last line could instead be:

```
[cmsupport@mycluster ~]$ cluster-check --certification ~/custom_con\
fig.xml
```

if a configuration file config-ib.xml from the default location has been copied over to the cmsupport account directory, and then modified for use by the cluster checker.

Handling Test Failures

The cluster checker produces several output files, with .xml, .out, .debug suffixes, which include time stamps in the filenames. If tests fail, the output files can be consulted for details. The output files can be found in the ~/intel-cluster-ready/logs directory.

When debugging and re-running tests, the option

```
--include_only <test>
```

can be passed to cluster-check to execute only the test named "<*test*>" (and the tests on which it depends).

In a heterogeneous cluster the cluster check run fails as a result of hardware differences. To resolve the failures, it is necessary to create multiple groups of homogeneous hardware. For more information, the Intel Cluster Checker documentation can be consulted.

12.4.4 Applying For The Certificate

When the cluster check run has reported that the "Check has Succeeded", a certificate may be requested for the cluster. Requesting a certificate involves creating a "Bill of Materials", which includes software as well as hardware. This is then submitted together with the output files from Intel Cluster Checker runs and the packages lists to cluster@intel.com. The Intel Cluster Ready site contains interactive submissions forms that make the application process as easy as possible. For more details, http://software.intel.com/en-us/cluster-ready/ can be visited.

12.5 CUDA For GPUs

The optional CUDA packages should be installed in order to take advantage of the computational capabilities of NVIDIA GPUs.

12.5.1 GPUs And GPU Units

GPUs (Graphics Processing Units) are processors that are heavily optimized for executing certain types of parallel processing tasks. GPUs were originally used for rendering graphics, and one GPU typically has hundreds of cores. When used for general processing, they are sometimes called General Processing GPUs, or GPGPUs. In this manual the "GP" prefix is not used for convenience.

GPUs can be physically inside the node that uses them, or they can be physically external to the node that uses them. As far as the operating system on the node making use of the physically external GPUs is concerned, the GPUs are internal to the node.

If the GPUs are physically external to the node, then they are typically in a GPU Unit. A GPU unit is a chassis that hosts only GPUs. It can typically provide GPU access to several nodes, usually via PCIe extender connections. An example of a GPU Unit is the Dell PowerEdge C410x, which comes in a 3U chassis size, has upto 16 GPUs, and can allocate up to 4 GPUs per node.

12.5.2 Installing CUDA

A number of CUDA 4.0, 4.1, 4.2 and 5.0 packages exist in the YUM repository. The CUDA 4.0 packages are:

Package	Type	Description
`cuda40-toolkit`	shared	CUDA 4.0 math libraries and utilities
`cuda40-sdk`	shared	CUDA 4.0 software development kit
`cuda40-profiler`	shared	CUDA 4.0 profiler
`cuda40-tools`	shared	CUDA 4.0 sdk tools
`cuda40-driver`	local	CUDA 4.0 driver
`cuda40-libs`	local	CUDA 4.0 libraries
`cuda40-xorg`	local	CUDA 4.0 X.org driver and libraries

The CUDA 4.1+ (i.e. CUDA 4.1, CUDA 4.2, or CUDA 5.0) packages have the same names, except that the `cuda40-tools` package is replaced by a `cuda41-tdk`, `cuda42-tdk`, or `cuda50-tdk` package.

To update all packages from one CUDA version to another, the packages from the original CUDA version must first be removed. For example, to update from CUDA 4.0 to 4.1, the CUDA 4.0 packages must first be removed before installing CUDA 4.1 packages.

The installation procedure for CUDA 4.0 packages is described in this section; the installation for CUDA 4.1+ is similar.

The packages marked as "shared" in the preceding table should be installed on the head nodes of a cluster using CUDA-compatible GPUs. The packages marked as "local" should be installed to all nodes that access the GPUs. In most cases this means that the `cuda40-driver` and `cuda40-libs` packages should be installed in a software image (section 3.1.2).

If a head node also accesses GPUs, the `cuda40-driver` and `cuda40-libs` packages should be installed on it, too.

For packages of type local, only one package version out of the choice of CUDA 4.0, 4.1, 4.2, or 5.0 can be installed at a time. So, for example, considering the case where a cluster may have one node (or node category) that runs CUDA 4.0, and another node (or node category) that runs CUDA 4.1. Then the CUDA 4.0 node (or node category) can only run CUDA 4.1 jobs after CUDA 4.1 is reinstalled on the node (or category), and vice versa. This is because the driver is compiled and installed on the fly during boot.

For packages of type shared, multiple versions can be installed on the head node at one time, out of the choice of CUDA 4.0, 4.1, 4.2, or 5.0. The particular CUDA version that the node works with can be selected via a modules environment command:

Example

```
module add shared cuda41
```

The CUDA packages have additional dependencies that may require access to repositories besides the main repository in order to resolve them automatically. For example, for Red Hat, a subscription (Appendix M.1.2) is needed to the `rhel-x86_64-server-supplementary-5` channel for RHEL5, and to the `rhel-x86_64-server-optional-6` channel for RHEL6.

One of the dependencies of the `cuda40-driver` package is the `freeglut-devel` package, so it should be installed on a node that accesses a GPU. If the CUDA SDK source is to be compiled on the head node (with the head node not accessing a GPU, and with the cuda40-driver package not installed) then the `freeglut`, `freeglut-devel`, and `libXi-devel` packages can be installed on the head node.

The `cuda40-libs` package is used to compile `cuda40-driver`, the driver that manages the GPU. Therefore, when installing the `cuda40-driver` with `yum`, `cuda40-libs` and several other X11-related packages are installed too due to package dependencies.

The `cuda40-sdk` can be used to compile libraries and tools that are not part of the CUDA toolkit, but used by CUDA software developers, such as the `deviceQuery` binary (section 12.5.4).

The `cuda40-xorg` and `cuda40-tools` packages are optional. The `cuda40-xorg` package contains the driver and libraries for an X server. The `cuda40-tools` contains files for the CUDA profiling tools interface, CUDA debugging API and the nVidia Management Library.

Example

On a cluster where (some of) the nodes access GPUs but the head node does not access a GPU, the following commands are issued on the head node to install the packages through YUM:

```
yum install cuda40-toolkit cuda40-sdk cuda40-profiler
yum --installroot=/cm/images/default-image install cuda40-driver
```

The `cuda40-driver` package provides an init-script which is executed at boot-time to load the CUDA driver. Because the CUDA driver depends

on the running kernel, the script compiles the CUDA driver using the CUDA libaries on the fly, and subsequently loads the module into the running kernel.

This cuda40-driver can also be loaded on the fly by calling the init-script. Loading the driver also causes a number of diagnostic kernel messages to be logged:

Example

```
[root@mycluster ~]# /etc/init.d/cuda40-driver start
Compiling CUDA4.0 driver..installing..probe..              [  OK  ]
[root@mycluster ~]# dmesg
...
PCI: Setting latency timer of device 0000:0e:00.0 to 64
PCI: Setting latency timer of device 0000:10:00.0 to 64
NVRM: loading NVIDIA UNIX x86_64 Kernel Module  270.35  Fri Mar 18 11:\
48:56 PDT 2011
```

If there is a failure in compiling the CUDA module, usually indicated by a message saying "`Could not make module`" or "`Cannot determine kernel version`", then it is usually because compilation is not possible due to missing the correct kernel development package from the distribution. Section 12.5.3 explains how to check for and install the appropriate missing package.

12.5.3 Installing Kernel Development Packages

This section can be skipped if there is no CUDA compilation problem.

Typically, a CUDA compilation problem (section 12.5.2) is due to a missing or mismatched `kernel` package and `kernel-devel package`.

To check the head node and software images for the installation status of the `kernel-devel` package, the Bright Cluster Manager utility `kernel-devel-check` is used (section 9.3.5).

Alternatively, if a standard kernel is in use by the image, then simply upgrading CUDA, the standard kernel, and `kernel-devel`, to their latest versions may be a good tactic to fix a CUDA compilation problem because the `kernel` and `kernel-devel` package versions become synchronized during such an upgrade.

12.5.4 Verifying CUDA

An extensive method to verify that CUDA is working is to run the `verify_cuda40.sh` script, located in the CUDA SDK directory.

This script first copies the CUDA SDK source to a local directory under `/tmp` or `/local`. It then builds CUDA test binaries and runs them. It is possible to select which of the CUDA test binaries are run.

A help text showing available script options is displayed when "`verify_cuda40.sh -h`" is run.

Example

```
[root@cuda-test ~]# module load cuda40/toolkit

[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 4.0.11]# ./verify_cuda40.sh
Copy cuda40 sdk files to "/local/cuda40" directory.
```

```
make clean

make (may take a while)


Run all tests? (y/N)? y

Executing: /local/cuda40/C/bin/linux/release/alignedTypes

[alignedTypes]
[Tesla M2070] has 14 MP(s) x 32 (Cores/MP) = 448 (Cores)
> Compute scaling value = 1.00
> Memory Size = 49999872
Allocating memory...
Generating host input data array...
Uploading input data to GPU memory...
Testing misaligned types...
uint8...
Avg. time: 2.267250 ms / Copy throughput: 20.538542 GB/s.
        TEST OK
...
...
All cuda40 just compiled test programs can be found in the
 "/local/cuda40/C/bin/linux/release/" directory
They can be executed from the "/local/cuda40/C" directory.

The "/local/cuda40" directory may take up a lot of diskspace.
Use "rm -rf /local/cuda40" to remove the data.
```

Another method to verify that CUDA is working, is to build and use the `deviceQuery` command on a node accessing one or more GPUs. The `deviceQuery` command lists all CUDA-capable GPUs that a device can access, along with several of their properties.

Example

```
[root@cuda-test ~]# module load cuda40/toolkit
[root@cuda-test ~]# cd $CUDA_SDK/C
[root@cuda-test C]# make clean
...
[root@cuda-test C]# make
...
[root@cuda-test C]# bin/linux/release/deviceQuery
bin/linux/release/deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

There are 2 devices supporting CUDA

Device 0: "Tesla M2070"
  CUDA Driver Version:                           4.0
  CUDA Runtime Version:                          4.0
  CUDA Capability Major/Minor version number:    2.0
  Total amount of global memory:                 5636554752 bytes
  (14) Multiprocessors x (32) CUDA Cores/MP:     448 CUDA Cores
```

```
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Clock rate:                                    1.15 GHz
  Concurrent copy and execution:                 Yes
  # of Asynchronous Copy Engines:                2
  Run time limit on kernels:                     Yes
  Integrated:                                    No
  Support host page-locked memory mapping:       Yes
  Compute mode:                                  Default
  Concurrent kernel execution:                   Yes
  Device has ECC support enabled:                Yes
  Device is using TCC driver mode:               No

...
...
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 4.0,
 CUDA Runtime Version = 4.0, NumDevs = 2,
 Device = Tesla M2070, Device = Tesla M2070

PASSED
```

The CUDA user manual has further information on how to run compute jobs using CUDA.

12.5.5 Verifying OpenCL

CUDA 4.0 also contains an OpenCL compatible interface. To verify that the OpenCL is working, the `oclDeviceQuery` utility can be built and executed.

Example

```
[root@cuda-test ~]# module add cuda40/toolkit
[root@cuda-test ~]# cd $CUDA_SDK/OpenCL
[root@cuda-test OpenCL]# make clean
...
[root@cuda-test OpenCL]# make
...
[root@cuda-test OpenCL]# bin/linux/release/oclDeviceQuery
bin/linux/release/oclDeviceQuery Starting...

OpenCL SW Info:

 CL_PLATFORM_NAME:      NVIDIA CUDA
 CL_PLATFORM_VERSION:   OpenCL 1.0 CUDA 4.0.1
 OpenCL SDK Revision:   7027912


OpenCL Device Info:
```

```
 2 devices found supporting OpenCL:

 --------------------------------
 Device Tesla M2070
 --------------------------------
  CL_DEVICE_NAME:                        Tesla M2070
  CL_DEVICE_VENDOR:                      NVIDIA Corporation
  CL_DRIVER_VERSION:                     270.35
  CL_DEVICE_VERSION:                     OpenCL 1.0 CUDA
  CL_DEVICE_TYPE:                        CL_DEVICE_TYPE_GPU
  CL_DEVICE_MAX_COMPUTE_UNITS:           14
  CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS:    3
  CL_DEVICE_MAX_WORK_ITEM_SIZES:         1024 / 1024 / 64
  CL_DEVICE_MAX_WORK_GROUP_SIZE:         1024
  CL_DEVICE_MAX_CLOCK_FREQUENCY:         1147 MHz
  CL_DEVICE_ADDRESS_BITS:                32
  CL_DEVICE_MAX_MEM_ALLOC_SIZE:          1343 MByte
  CL_DEVICE_GLOBAL_MEM_SIZE:             5375 MByte
  CL_DEVICE_ERROR_CORRECTION_SUPPORT:    yes
  CL_DEVICE_LOCAL_MEM_TYPE:              local
  CL_DEVICE_LOCAL_MEM_SIZE:              48 KByte
  CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE:    64 KByte
  CL_DEVICE_QUEUE_PROPERTIES:            CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE
  CL_DEVICE_QUEUE_PROPERTIES:            CL_QUEUE_PROFILING_ENABLE
  CL_DEVICE_IMAGE_SUPPORT:               1
  CL_DEVICE_MAX_READ_IMAGE_ARGS:         128
...
...

oclDeviceQuery, Platform Name = NVIDIA CUDA,
Platform Version = OpenCL 1.0 CUDA 4.0.1,
SDK Revision = 7027912, NumDevs = 2,
Device = Tesla M2070, Device = Tesla M2070

System Info:

 Local Time/Date =  12:34:37, 04/05/2011
 CPU Name: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
 # of CPU processors: 8
 Linux version 2.6.18-194.32.1.el5 (mockbuild@builder10.centos.org)
 (gcc version 4.1.2 20080704
 (Red Hat 4.1.2-48)) #1 SMP Wed Jan 5 17:52:25 EST 2011


PASSED
```

12.5.6 Configuring The X server

The X server can be configured to use a CUDA GPU. To support the X server, the `cuda40-driver`, `cuda40-libs`, and `cuda40-xorg` need to be installed.

The following file pathname lines need to be added to the `Files` section of the X configuration file:

```
ModulePath "/usr/lib64/xorg/modules/extensions/nvidia"
```

```
    ModulePath "/usr/lib64/xorg/modules/extensions"
    ModulePath "/usr/lib64/xorg/modules"
```

The following dynamic module loading lines need to be added to the `Module` section of the X configuration:

```
    Load       "glx"
```

The following graphics device description lines need to be replaced in the `Device` section of the X configuration:

```
    Driver      "nvidia"
```

The default configuration file for X.org is `/etc/X11/xorg.conf`.

Example

```
Section "ServerLayout"
    Identifier     "Default Layout"
    Screen      0  "Screen0" 0 0
    InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
    ModulePath     "/usr/lib64/xorg/modules/extensions/nvidia"
    ModulePath     "/usr/lib64/xorg/modules/extensions"
    ModulePath     "/usr/lib64/xorg/modules"
EndSection

Section "Module"
    Load           "glx"
EndSection

Section "InputDevice"
    Identifier     "Keyboard0"
    Driver         "kbd"
    Option         "XkbModel" "pc105"
    Option         "XkbLayout" "us"
EndSection

Section "Device"
    Identifier     "Videocard0"
    Driver         "nvidia"
    BusID          "PCI:14:0:0"
EndSection

Section "Screen"
    Identifier     "Screen0"
    Device         "Videocard0"
    DefaultDepth   24
    SubSection     "Display"
        Viewport   0 0
        Depth      24
    EndSubSection
EndSection
```

12.6 OFED Software Stack

Bright Cluster Manager packages the OFED software developed by Mellanox and QLogic. The Bright Cluster Manager packages are usually more recent than the distribution packages and thus provide support for more recent hardware, as well as more features.

If there is no OFED kernel modules package available for the kernel in use, then the Bright Computing OFED install script tries to build the package from the source package provided by Mellanox or QLogic. However, very recent kernels may not yet be supported by the source package. If a build fails for such a kernel, then the OFED software stack will fail to install, and nothing is changed on the head node or the software image. OFED hardware manufacturers resolve build problems with their software shortly after they become aware of them, but in the meantime a supported kernel must be used.

12.6.1 Bright Repository Mellanox OFED Stack Installation

Package name: `mlnx-ofed`

Running a "`yum install mlnx-ofed`" unpacks several packages and scripts, but does not install them due to the fundamental nature of the changes it would carry out. The `mlnx-ofed-install.sh` install script is used to carry out the changes:

- On the head node, the default distribution OFED software stack can be replaced with the Bright Computing repository Mellanox OFED software stack as follows:

```
[root@bright52~]# /cm/local/apps/mlnx-ofed/current/bin/mlnx-ofed-ins\
tall.sh -h
```

  A reboot is recommended after the script completes the install.

- For a software image, for example `default-image`, used by the regular nodes, the default distribution OFED software stack can be replaced with the Bright Computing repository Mellanox OFED software stack as follows:

```
[root@bright52~]# /cm/local/apps/mlnx-ofed/current/bin/mlnx-ofed-inst\
all.sh -s default-image
```

  A reboot updates the software image on the regular node.

12.6.2 Bright Repository QLogic OFED Stack Installation

Package name: `qlgc-ofed`

Running a "`yum install qlgc-ofed`" package unpacks several packages and scripts, but does not install them due to the fundamental nature of the changes it would carry out. The `qlgc-ofed-install.sh` install script is used to carry out the changes:

- On the head node, the default distribution OFED software stack can be replaced with the Bright Computing repository QLogic OFED software stack as follows:

```
[root@bright52~]# /cm/local/apps/qlgc-ofed/current/bin/qlgc-ofed-ins\
tall.sh -h
```

A reboot is recommended after the script completes the install.

- For a software image, for example `default-image`, used by the regular nodes, the default distribution OFED software stack can be replaced with the Bright Computing repository QLogic OFED software stack as follows:

```
[root@bright52~]# /cm/local/apps/qlgc-ofed/current/bin/qlgc-ofed-inst\
all.sh -s default-image
```

A reboot updates the software image on the regular node.

12.7 Lustre

This section covers aspects of Lustre, a parallel distributed filesystem which can be used for clusters.

After a short architectural overview of Lustre, steps to set up a Lustre filesystem to work with Bright Cluster Manager are described.

Further details on Lustre can be found at `http://wiki.whamcloud.com`.

12.7.1 Architecture

There are four components to a Lustre filesystem:

1. One management service (MGS)
2. One metadata target (MDT) on the metadata server (MDS)
3. Multiple object storage target (OSTs), on an object storage server (OSS)
4. Clients that access and use the data on the Lustre filesystem

The management services run on the metadata server, and hold information for all Lustre filesystems running in a cluster.

Metadata values, like filenames, directories, permissions, and file layout are stored on the metadata target. The file data values themselves are stored on the object storage targets.

Among the supported Lustre networking types are TCP/IP over Ethernet and InfiniBand.

12.7.2 Server Implementation

Lustre servers, MDS, and OSSs, run on a patched kernel. The patched kernel, kernel modules, and software can be installed with RPM packages. The Lustre server software can also be compiled from source, but the kernel needs to be patched and recreated. Lustre supports one kernel version per Lustre version.

To use Lustre with Bright Cluster Manager, a Lustre server image and a Lustre client image are installed onto the head node so that they can provision the Lustre nodes.

Creating The Lustre Server Image

To create a Lustre server image, a clone is made of an existing software image, for example from `default-image`.

In cmgui this is done by selecting the Software Images resource to bring up the Overview tabbed pane display. Selecting the image to clone and then clicking on the Clone button prompts for a confirmation to build a clone image (figure 12.1):

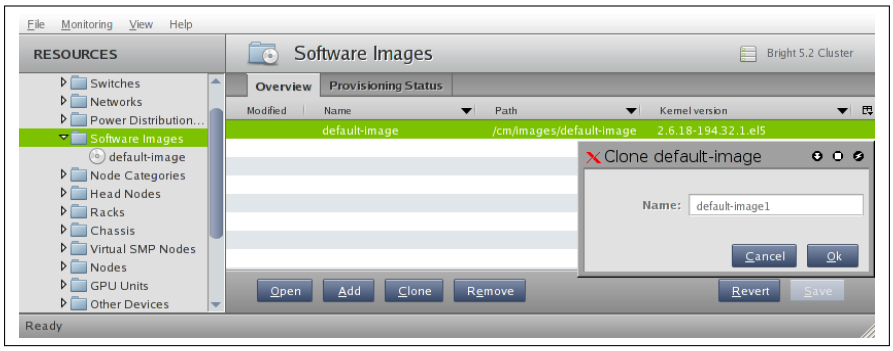

Figure 12.1: cmgui: Cloning An Image

Alternatively, cmsh on the head node can create a clone image:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-server-image
[mycluster->softwareimage*[lustre-server-image*]]% commit
```

The RPM Lustre packages can be downloaded from the Whamcloud website. It is best to first check which version of Lustre can be used for a particular distribution against the Lustre Support Matrix at http://wiki.whamcloud.com/display/PUB/Lustre+Support+Matrix.

After choosing a Lustre version from the Lustre Support Matrix, the appropriate distribution and platform can be chosen. For CentOS and Scientific Linux (SL), Red Hat packages can be used. Download links for Lustre releases, among others kernel, module, lustre and lustre-ldiskf packages, can be found at http://wiki.whamcloud.com/display/PUB/Lustre+Releases. Download links for Lustre tools, among others the e2fsprogs package, can be found at http://wiki.whamcloud.com/display/PUB/Lustre+Tools

The RPM packages to download are:

- kernel: Lustre-patched kernel (MDS/MGS/OSS only)
- lustre-modules: Lustre kernel modules (client and server for the Lustre-patched kernel)
- lustre: Lustre userland tools (client and server for the Lustre-patched kernel)
- lustre-ldiskfs: Backing filesystem kernel module (MDS/MGS/OSS only)
- e2fsprogs: Backing filesystem creation and repair tools (MDS/MGS/OSS only)
- e2fsprogs-libs: Backing filesystem creation and repair tools libraries (MDS/MGS/OSS and EL6 only)

- e2fsprogs-devel: Backing filesystem creation and repair tools development (MDS/MGS/OSS and EL6 only)

In most cases on EL5 nodes, the e2fsprogs package from the distribution is already installed, so the package only has to be upgraded. On EL5 nodes it is possible that the Lustre e2fsprogs package conflicts with the e4fsprogs package from the distribution, in which case the e4fsprogs package has to be removed. If the Lustre kernel version has a lower version number than the already installed kernel, then the Lustre kernel needs to be installed with the --force option. Warning and error messages that may display about installing packages in a software image can be ignored.

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-server-image/root/lustre
[root@mycluster ~]# cp kernel-* lustre-* e2fsprogs-* /cm/images/lustre-\
server-image/root/lustre
[root@mycluster ~]# chroot /cm/images/lustre-server-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -e e4fsprogs
[root@mycluster lustre]# rpm -Uvh e2fsprogs-1.41.90.wc3-0redhat.x86_64.\
rpm
[root@mycluster lustre]# rpm -ivh --force kernel-2.6.18-238.19.1.el5_lu\
stre.g65156ed.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-ldiskfs-3.3.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_\
64.rpm \
  lustre-2.1.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_64.rpm \
  lustre-modules-2.1.0-2.6.18_238.19.1.el5_lustre.g65156ed_g9d71fe8.x86_\
64.rpm
[root@mycluster lustre]# rm kernel-* lustre-* e2fsprogs-*
```

In most cases on EL6 nodes, the e2fsprogs package from the distribution is already installed, so the package only has to be upgraded. If the Lustre kernel version has a lower version number than the already installed kernel, then the Lustre kernel needs to be installed with the --force option. Warning and error messages that may display about installing packages in a software image can be ignored.

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-server-image/root/lustre
[root@mycluster ~]# cp kernel-* lustre-* e2fsprogs-* /cm/images/lustre-\
server-image/root/lustre/
[root@mycluster ~]# chroot /cm/images/lustre-server-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -Uvh \
  e2fsprogs-1.41.90.wc3-7.el6.x86_64.rpm \
  e2fsprogs-libs-1.41.90.wc3-7.el6.x86_64.rpm \
  e2fsprogs-devel-1.41.90.wc3-7.el6.x86_64.rpm
[root@mycluster lustre]# rpm -ivh --force kernel-2.6.32-131.6.1.el6_lus\
tre.g65156ed.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-ldiskfs-3.3.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71f\
e8.x86_64.rpm \
```

```
  lustre-2.1.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71fe8.x86_6\
4.rpm \
  lustre-modules-2.1.0-2.6.32_131.6.1.el6_lustre.g65156ed.x86_64_g9d71f\
e8.x86_64.rpm
[root@mycluster lustre]# rm kernel-* lustre-* e2fsprogs-*
```

The kernel version is set to the Lustre kernel version for the Lustre server image:

Example

```
[root@mycluster ~]# cd /cm/images/lustre-server-image/boot
[root@mycluster boot]# ls -1 vmlinuz-*
vmlinuz-2.6.18-238.19.1.el5_lustre.g65156ed
vmlinuz-2.6.18-274.7.1.el5
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% use lustre-server-image
[mycluster->softwareimage[lustre-server-image]]% set kernelversion 2.6.\
18-238.19.1.el5_lustre.g65156ed
[mycluster->softwareimage*[lustre-server-image*]]% commit
```

Creating The Lustre Server Category

A node category is cloned. For example, `default` to `lustre-server`. The software image is set to the Lustre server image, the `installbootrecord` option is enabled, and the `roles` option is cleared:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% clone default lustre-server
[mycluster->category*[lustre-server*]]% set softwareimage lustre-server\
-image
[mycluster->category*[lustre-server*]]% set installbootrecord yes
[mycluster->category*[lustre-server*]]% clear roles
[mycluster->category*[lustre-server*]]% commit
```

Creating Lustre Server Nodes

An MDS node is created with `cmsh`:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->[device]]% add physicalnode mds001 10.141.16.1
[mycluster->[device*[mds001*]]]% set category lustre-server
[mycluster->[device*[mds001*]]]% commit
```

One or multiple OSS node(s) are created with `cmsh`:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->[device]]% add physicalnode oss001 10.141.32.1
[mycluster->[device*[oss001*]]]% set category lustre-server
[mycluster->[device*[oss001*]]]% commit
```

After the first boot and initial installation, the MDS and OSS(s) are configured to boot from the local drive instead of the network, to preserve locally made changes. The BIOS of each server needs to be configured to boot from the local drive.

For nodes based on EL6 the Lustre initrd file needs to regenerated, after the first boot and initial installation. To regenerate the initrd image file, for the nodes in the `lustre-server` category:

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->device]% pexec -c lustre-server "mv \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.orig \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.old"
[mycluster->device]% pexec -c lustre-server "mkinitrd \
  /boot/initrd-2.6.32-131.6.1.el6_lustre.g65156ed.x86_64.orig \
  2.6.32-131.6.1.el6_lustre.g65156ed.x86_64"
```

Warning and error messages that display about write errors or broken pipes can be ignored.

Creating The Lustre Metadata Target

On the metadata server a metadata target must be created. To create the metadata target a free disk, partition, or logical volume is used. The disk device can also be an external storage device and/or a redundant storage device. The metadata server also acts as a management server.

To format a metadata target `mkfs.lustre` is used. For example, the following formats `/dev/sdb`, setting the Lustre filesystem name to "`lustre00`":

Example

```
[root@mds001 ~]# mkfs.lustre --fsname lustre00 --mdt --mgs /dev/sdb
```

The filesystem is mounted and the entry added to `/etc/fstab`:

Example

```
[root@mds001 ~]# mkdir /mnt/mdt
[root@mds001 ~]# mount -t lustre /dev/sdb /mnt/mdt
[root@mds001 ~]# echo "/dev/sdb /mnt/mdt lustre rw,_netdev 0 0" >> /etc\
/fstab
```

Creating The Lustre Object Storage Target

On the object storage server one or multiple object storage target(s) can be created. To create the object storage target, free disks, partitions or logical volumes are used. The disk devices can also be an external storage device and/or a redundant storage device.

To format a object storage target `mkfs.lustre` is used. For example, the following formats `/dev/sdb`, sets the management node to `10.141.16.1`, sets the filesystem name to `lustre00`, and sets the network type to TCP/IP:

Example

```
[root@oss001 ~]# mkfs.lustre --fsname lustre00 --ost --mgsnode=10.141.1\
6.1@tcp0 /dev/sdb
```

The filesystem is mounted and the entry added to /etc/fstab:

Example

```
[root@oss001 ~]# mkdir /mnt/ost01
[root@oss001 ~]# mount -t lustre /dev/sdb /mnt/ost01
[root@oss001 ~]# echo "/dev/sdb /mnt/ost01 lustre rw,_netdev 0 0" >> /e\
tc/fstab
```

After mounting the OST(s) the Lustre clients can mount the Lustre filesystem.

12.7.3 Client Implementation

There are several ways to install a Lustre client.

If the client has a supported kernel version, the lustre-client RPM package and lustre-client-modules RPM package can be installed. The lustre-client-modules package installs the required kernel modules.

If the client does not have a supported kernel, a Lustre kernel, Lustre modules, and Lustre userland software can be installed with RPM packages.

The client kernel modules and client software can also be built from source.

Creating The Lustre Client Image: Method 1

This method describes how to create a Lustre client image with Lustre client RPM packages. It requires that the lustre-client-module package have the same kernel version as the kernel version used for the image.

To create a starting point image for the Lustre client image, a clone is made of the existing software image, for example from default-image.

The clone software image is created via cmgui (figure 12.1), or using cmsh on the head node:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

The RPM Lustre client packages are downloaded from the Whamcloud website:

- lustre-client: Lustre client userland tools (client for unpatched vendor kernel)
- lustre-client-modules: Lustre client kernel modules (client for unpatched vendor kernel)

The same Lustre version which is used for the Lustre servers is used for the Lustre clients.

The kernel version of the lustre-client-modules package must also match that of the kernel used. It is 2.6.18-238.19.1.el5 in the following example:

Example

```
[root@mycluster ~]# ls lustre-client-modules-*
lustre-client-modules-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster ~]# cmsh -c "softwareimage; use lustre-client-image; ge\
t kernelversion"
2.6.18-238.19.1.el5
```

The installation can then be carried out:

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-client-image/root/lustre
[root@mycluster ~]# cp lustre-client-*.rpm /cm/images/lustre-client-ima\
ge/root/lustre/
[root@mycluster ~]# chroot /cm/images/lustre-client-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -ivh lustre-client-modules-2.0.0.1-2.6.18_\
164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
[root@mycluster lustre]# rpm -ivh \
  lustre-client-modules-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm \
  lustre-client-2.1.0-2.6.18_238.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# rm lustre-client-*.rpm
```

Creating The Lustre Client Image: Method 2

This method describes how to create a Lustre client image with a Lustre kernel package.

To create the image for the Lustre client image, a clone is made of the existing `lustre-server-image` software image.

A clone software image is created via `cmgui` (figure 12.1), or using `cmsh` on the head node.

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone lustre-server-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

Creating the `lustre-server-image` software image is described in the Lustre Server Implementation section (section 12.7.2).

Creating The Lustre Client Image: Method 3

This method describes how to create a Lustre client image by building Lustre from source.

As a starting point image for a Lustre client image, a clone is made of the existing software image, for example from `default-image`.

A clone software image is created via `cmgui` (figure 12.1), or using `cmsh` on the head node.

Example

```
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% clone default-image lustre-client-image
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

The source package can be downloaded from the Whamcloud website. The same Lustre version used for Lustre servers is used for the Lustre clients.

Instead of selecting a Linux distribution and architecture, a source package with the right version em<version> to download is chosen:

- `lustre-client-source-<version>.rpm`: Lustre source code

The source file is copied to the image:

Example

```
[root@mycluster ~]# mkdir /cm/images/lustre-client-image/root/lustre
[root@mycluster ~]# cp lustre-client-source-*.rpm /cm/images/lustre-cli\
ent-image/root/lustre
```

If the `kernel-devel` package is not installed on the client image, it is first installed so that the kernel can be compiled. To check the `lustre-client-image` software image among others, for the installation status of the `kernel-devel` package, the Bright Cluster Manager utility `kernel-devel-check` is used (section 9.3.5).

To determine the kernel version used by the software image:

Example

```
[root@mycluster ~]# cmsh -c "softwareimage get lustre-client-image kern\
elversion"
2.6.18-274.7.1.el5
```

The Lustre software is then built and installed:

Example

```
[root@mycluster ~]# export RPM_BUILD_NCPUS=$(grep -c "^processor" /proc\
/cpuinfo)
[root@mycluster ~]# chroot /cm/images/lustre-client-image
[root@mycluster /]# cd /root/lustre
[root@mycluster lustre]# rpm -ivh lustre-client-source-2.1.0-2.6.18_238\
.19.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# rm lustre-client-source-2.1.0-2.6.18_238.19.1.\
el5_g9d71fe8.x86_64.rpm
[root@mycluster lustre]# cd /usr/src
[root@mycluster src]# ln -s kernels/2.6.18-274.7.1.el5-x86_64 linux
[root@mycluster src]# cd lustre-2.1.0
[root@mycluster lustre-2.1.0]# ./configure --disable-server
[root@mycluster lustre-2.1.0]# make rpms
[root@mycluster lustre-2.1.0]# cd ..
[root@mycluster src]# rm linux
[root@mycluster src]# rpm -e lustre-client-source
[root@mycluster src]# rm -rf lustre-2.1.0
[root@mycluster src]# cd /usr/src/redhat/RPMS/x86_64
[root@mycluster x86_64]# rpm -ivh --nodeps lustre-modules-2.1.0-2.6.18_\
274.7.1.el5_g9d71fe8.x86_64.rpm
[root@mycluster x86_64]# rpm -ivh lustre-2.1.0-2.6.18_274.7.1.el5_g9d71\
fe8.x86_64.rpm
[root@mycluster x86_64]# rm lustre-*
[root@mycluster x86_64]# cd ../../SRPMS
[root@mycluster SRPMS]# rm lustre-*
[root@mycluster SRPMS]# cd ../BUILD
[root@mycluster BUILD]# rm -rf lustre-*
```

To configure the lnet kernel module to use TCP/IP interface eth1, the string "options lnet networks=tcp(eth1)" is added to the /etc/modprobe.conf file of the client image:

```
[root@mycluster ~]# echo "options lnet networks=tcp(eth1)" >> /cm/image\
s/lustre-client-image/etc/modprobe.conf
```

To specify that a Lustre node uses a TCP/IP interface and an InfiniBand interface, the string "options lnet networks=tcp0(eth0),o2ib(ib0)" is added to the /etc/modprobe.conf file of the client image:

```
[root@mycluster ~]# echo "options lnet networks=tcp0(eth0),o2ib(ib0)" >\
> /cm/images/lustre-client-image/etc/modprobe.conf
```

Creating The Lustre Client Category

A node category is cloned, for example default to a new category lustre-client. The software image in this category is set to the Lustre client image, lustre-client:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% clone default lustre-client
[mycluster->category*[lustre-client*]]% set softwareimage lustre-client\
-image
[mycluster->category*[lustre-client*]]% commit
```

The Lustre client category is configured to mount the Lustre filesystem (some text in the display here is elided):

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% use lustre-client
[mycluster->category[lustre-client]]% fsmounts
[mycl...fsmounts]% add /mnt/lustre00
[myc...fsmounts*[/mnt/lustre00*]]% set device 10.141.16.1@tcp0:/lustre00
[myc...fsmounts*[/mnt/lustre00*]]% set filesystem lustre
[myc...fsmounts*[/mnt/lustre00*]]% set mountoptions rw,_netdev
[myc...fsmounts*[/mnt/lustre00*]]% commit
```

The configured fsmounts device is the MGS, which in the example has IP address 10.141.16.1. The network type used in the example is TCP/IP.

Creating Lustre Client Nodes

A client node is created as follows:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device
[mycluster->device]% add physicalnode lclient001 10.141.48.1
[mycluster->device*[lclient001*]]% set category lustre-client
[mycluster->device*[lclient001*]]% commit
```

The Lustre client is booted and checked to see if the Lustre filesystem is mounted. The Lustre file stripe configuration of the filesystem can be checked with "`lfs getstripe`", and it can be set with "`lfs setstripe`":

Example

```
[root@lclient001 ~]# lfs getstripe /mnt/lustre00
[root@lclient001 ~]# lfs setstripe -s 1M -o -1 -c -1 /mnt/lustre00
```

The "`lfs setstripe`" command in the example sets the filesystem to use 1MB blocks, the start OST is chosen by the MDS, data is striped over all available OSTs.

12.8 ScaleMP

This section describes how to use ScaleMP's vSMP for Cloud product to create virtual SMP nodes in a cluster.

12.8.1 Installing vSMP For Cloud

Before virtual SMP nodes can be created, the ScaleMP vSMP for Cloud software needs to be installed on the head node of the cluster. The vSMP for Cloud software consists of two components:

- The `image_manager` utility
- The vSMP image

Both components have to be copied to the `/cm/local/apps/vsmp` directory on the head node. In addition, the `/cm/local/apps/vsmp/vSMP.img` symbolic link should point to the vSMP image that should be used.

Example

Installing `image_manager` and version 3.5.155 of the vSMP image:

```
[root@mc ~]# cp image_manager /cm/local/apps/vsmp/
[root@mc ~]# cp vSMP-3.5.155.img /cm/local/apps/vsmp
[root@mc ~]# ln -sf vSMP-3.5.155.img /cm/local/apps/vsmp/vSMP.img
```

12.8.2 Creating Virtual SMP Nodes

After the vSMP for Cloud software has been installed, virtual SMP nodes may be created using `cmsh` or `cmgui`.

Creating a virtual SMP node in `cmgui` is done by clicking the `Virtual SMP Nodes` folder, clicking the `Add` button and entering a hostname (e.g. `vsmp001`). A virtual SMP node behaves like any other physical node, but has an extra configuration tab: `Virtual SMP`. This tab can be used to configure which physical nodes should be used as components of the virtual SMP node.

Nodes that are made members of a virtual SMP node, go into the `Aggregated` state and when booted load the vSMP kernel. After all members of a vSMP nodes have booted the vSMP kernel, the virtual SMP node boots as a single (large) node.

Example

Creating and powering up a virtual SMP node using `cmsh` is done as follows:

```
[mc]% device add virtualsmpnode vsmp001
[mc->device*[vsmp001*]]% set members node001 node002 node003
[mc->device*[vsmp001*]]% interfaces
[mc->device*[vsmp001*]->interfaces]% add physical BOOTIF
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% set ip 10.141.10.1
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% set network internalnet
[mc->device*[vsmp001*]->interfaces*[BOOTIF*]]% exit
[mc->device*[vsmp001*]->interfaces*]% exit
[mc->device*[vsmp001*]]% set provisioninginterface BOOTIF
[mc->device*[vsmp001*]]% commit
...
[mc->device[vsmp001]]% power reset -n vsmp001
```

After the virtual SMP node boots, it must be identified in the same way that a new physical node has to be identified at first boot. Section 6.4.2 has more information on node identification and selection.

13

High Availability

In a cluster with a single head node, the head node is a single point of failure for the entire cluster. It is often unacceptable that the failure of a single machine can disrupt the daily operations of a cluster.

The high availability (HA) feature of Bright Cluster Manager therefore allows clusters to be set up with two head nodes configured as a failover pair.

In this chapter:

- Section 13.1 describes the concepts behind HA, keeping the Bright Cluster configuration in mind.
- Section 13.2 describes how the Bright Cluster implementation of failover is set up.
- Section 13.3 describes how HA is managed on a Bright Cluster after it has been set up.

13.1 HA Concepts

13.1.1 Primary, Secondary, Active, Passive

In a cluster with an HA setup, one of the head nodes is named the *primary* head node and the other head node is named the *secondary* head node. Under normal operation, one of the two head nodes is in *active* mode, whereas the other is in *passive* mode.

The difference between naming versus mode is illustrated by realizing that while a head node which is primary always remains primary, the mode that the node is in may change. Thus, the primary head node can be in passive mode when the secondary is in active mode. Similarly the primary head node may be in active mode while the secondary head node is in passive mode.

13.1.2 Monitoring The Active Head Node, Initiating Failover

In HA the passive head node continuously monitors the active head node. If the passive finds that the active is no longer operational, it will initiate a *failover sequence*. A failover sequence involves taking over resources, services and network addresses from the active head node. The goal is to continue providing services to compute nodes, so that jobs running on these nodes keep running.

13.1.3 Services In Bright Cluster Manager HA Setups

There are several services being offered by a head node to the cluster and its users.

Services Running On Both Head Nodes

One of the design features of the HA implementation in Bright Cluster Manager is that whenever possible, services are offered on both the active as well as the passive head node. This allows the capacity of both machines to be used for certain tasks (e.g. provisioning), but it also means that there are fewer services to move in the event of a failover sequence.

On a default HA setup, the following key services for cluster operations are always running on both head nodes:

- **CMDaemon**: providing certain functionality on both head nodes (e.g. provisioning)
- **DHCP**: load balanced setup
- **LDAP**: running in replication mode (the active head node LDAP database is pulled by the passive)
- **MySQL**: running in multiple-master replication mode (the active head node MySQL database is pulled by the passive)
- **NTP**
- **DNS**

When an HA setup is created from a single head node setup, the above services are automatically reconfigured to run in the HA environment over two head nodes.

Provisioning role runs on both head nodes In addition, both head nodes also take up the *provisioning role*, which means that nodes can be provisioned from both head nodes.

The implications of running a cluster with multiple provisioning nodes are described in detail in section 6.2. One important aspect described in that section is how to make provisioning nodes aware of image changes.

From the administrator's point of view, achieving awareness of image changes for provisioning nodes in HA clusters is dealt with in the same way as for single-headed clusters. Thus, if using `cmsh`, the `updateprovisioners` command from within `softwareimage` mode is used, while if using `cmgui`, it is done from the `Software Images` resource, then selecting the `Provisioning Status` tab, and clicking on the `Update Provisioning Nodes` button (section 6.2.4).

Services That Migrate To The Active Node

Although it is possible to configure any service to migrate from one head node to another in the event of a failover, in a typical HA setup only the following services migrate:

- NFS
- Workload Management (e.g. SGE, Torque/Maui)

13.1.4 Failover Network Topology

A two-head failover network layout is illustrated in figure 13.1. In the il-

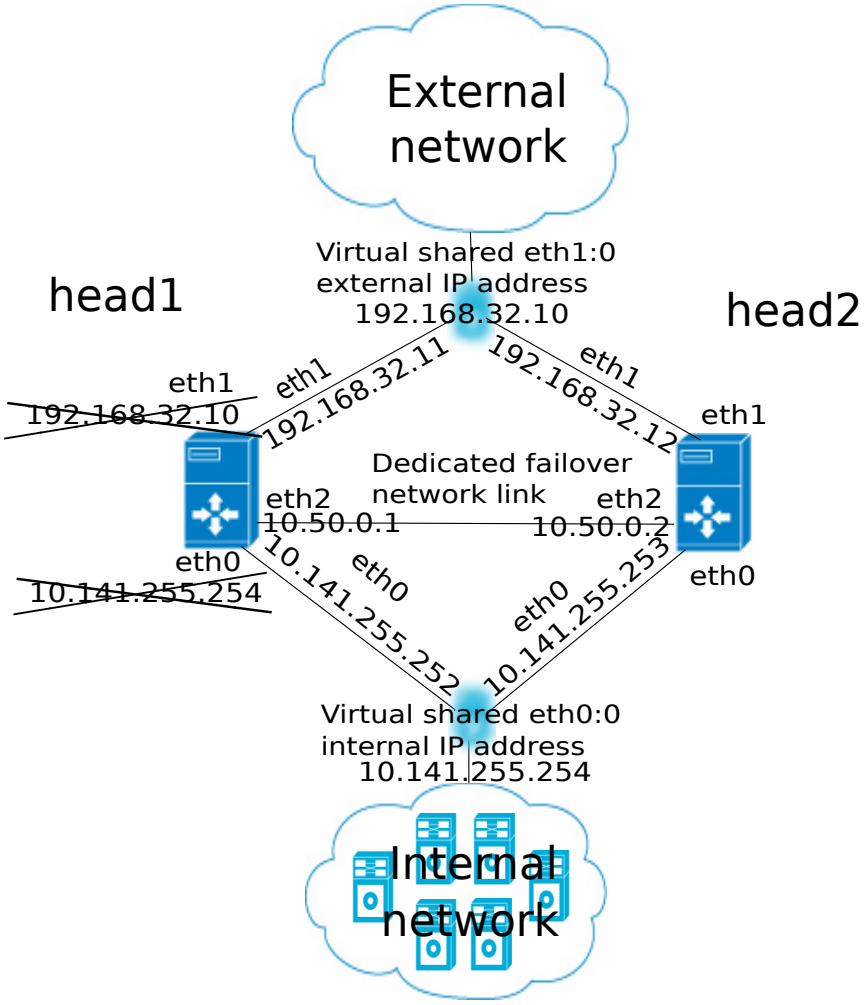

Figure 13.1: High Availability: Two-Head Failover Network Topology

lustration, the primary `head1` is originally a head node before the failover design is implemented. It is originally set up as part of a Type 1 network (section 2.3.6), with an internal interface eth0, and an external interface eth1.

When the secondary head is connected up to help form the failover system, several changes are made.

HA: Network Interfaces

Each head node in an HA setup typically has at least an external and an internal network interface, each configured with an IP address.

In addition, an HA setup uses two virtual IP interfaces, each of which has an associated virtual IP address: the external shared IP address and the internal shared IP address.

In a normal HA setup, both shared IP addresses are hosted on the head node that is operating in active mode, and during failover, the interfaces migrate and become part of the head node that then becomes active.

When head nodes are also being used as login nodes, users outside

of the cluster are encouraged to use the shared external IP address for connecting to the cluster. This ensures that they always reach whichever head node is active. Similarly, inside the cluster, nodes use the shared internal IP address wherever possible for referring to the head node. For example, nodes mount NFS filesystems on the shared internal IP interface so that the imported filesystems continue to be accessible in the event of a failover.

Shared interfaces are implemented as alias interfaces on the physical interfaces (e.g. `eth0:0`). They are activated when a head node becomes active, and deactivated when a head node becomes passive.

HA: Dedicated Failover Network

In addition to the normal internal and external network interfaces on both head nodes, the two head nodes are usually also connected using a direct dedicated network connection, `eth2` in figure 13.1. This connection is used between the two head nodes to monitor their counterpart's availability. It is called a *heartbeat* connection because the monitoring is usually done with a regular heartbeat-like signal between the nodes such as a ping, and if the signal is not detected, it suggests a head node is dead.

To set up a failover network, it is highly recommended to simply run a UTP cable directly from the NIC of one head node to the NIC of the other, because not using a switch means there is no disruption of the connection in the event of a switch reset.

13.1.5 Shared Storage

Almost any HA setup also involves some form of shared storage between two head nodes to preserve state after a failover sequence. For example, user home directories must always be available to the cluster in the event of a failover.

In the most common HA setup, the following two directories are shared:

- User home directories (i.e. `/home`)
- Shared tree containing applications and libraries that are made available to the nodes (i.e. `/cm/shared`)

The shared filesystems are only available on the active head node. For this reason, it is generally recommended that users login via the shared IP address, rather than ever using the direct primary or secondary IP address. End-users logging into the passive head node by direct login may run into confusing behavior due to unmounted filesystems.

Although Bright Cluster Manager gives the administrator full flexibility on how shared storage is implemented between two head nodes, there are generally three types being used: NAS, DAS and DRBD.

NAS

In a Network Attached Storage (NAS) setup, both head nodes mount a shared volume from an external network attached storage device. In the most common situation this would be an NFS server either inside or outside of the cluster.

Because imported mounts can typically not be re-exported (which is true at least for NFS), nodes typically mount filesystems directly from the NAS device.

DAS

In a Direct Attached Storage (DAS) setup, both head nodes share access to a block device that is usually acccessed through a SCSI interface. This could be a disk-array that is connected to both head nodes, or it could be a block device that is exported by a corporate SAN infrastructure.

Although the block device is visible and can physically be accessed simultaneously on both head nodes, the filesystem that is used on the block device is typically not suited for simultaneous access. Simultaneous access to a filesystem from two head nodes must therefore be avoided because it generally leads to filesystem corruption. Only special purpose parallel filesystems such as GPFS and Lustre are capable of being accessed by two head nodes simultaneously.

DRBD

In a setup with DRBD, both head nodes mirror a physical block device on each node device over a network interface. This results in a virtual shared DRBD block device. A DRBD block devices is effectively a DAS block device simulated via a network. DRBD is a cost-effective solution for implementing shared storage in an HA setup.

Custom Shared Storage With Mount And Unmount Scripts

The cluster management daemon on the two head nodes deals with shared storage through a *mount script* and an *unmount script*. When a head node is moving to active mode, it must acquire the shared filesystems. To accomplish this, the other head node first needs to relinquish any shared filesystems that may still be mounted. After this has been done, the head node that is moving to active mode invokes the *mount script* which has been configured during the HA setup procedure. When an active head node is requested to become *passive* (e.g. because the administrator wants to take it down for maintenance without disrupting jobs), the *unmount script* is invoked to release all shared filesystems.

By customizing the *mount* and *unmount* scripts, an administrator has full control over the form of shared storage that is used. Also an administrator can control which filesystems are shared.

Mount scripts paths can be set via `cmsh` or `cmgui` (section 13.3.6).

13.1.6 Guaranteeing One Active Head At All Times

Because of the risks involved in accessing a shared filesystem simultaneously from two head nodes, it is vital that only one head node is in active mode at any time. To guarantee that a head node that is about to switch to active mode will be the only head node in active mode, it must either receive confirmation from the other head node that it is in passive mode, or it must make sure that the other head node is powered off.

What Is A Split Brain?

When the passive head node determines that the active head node is no longer reachable, it must also take into consideration that there could be a communication disruption between the two head nodes. Because the "brains" of the cluster are communicatively "split" from each other, this is called a *split brain* situation.

Since the normal communication channel between the passive and active may not be working correctly, it is not possible to use only that chan-

nel to determine either an inactive head or a split brain with certainty. It can only be suspected.

Thus, on the one hand, it is possible that the head node has, for example, completely crashed, becoming totally inactive and thereby causing the lack of response. On the other hand, it is also possible that, for example, a switch between both head nodes is malfunctioning, and that the active head node is still up and running, looking after the cluster as usual, and that the head node in turn observes that the passive head node seems to have split away from the network.

Further supporting evidence from the dedicated failover network channel is therefore helpful. Some administrators find this supporting evidence an acceptable level of certainty, and configure the cluster to decide to automatically proceed with the failover sequence, while others may instead wish to examine the situation first before manually proceeding with the failover sequence. The implementation of automatic vs manual failover is described in section 13.1.7. In either implementation, *fencing*, described next, takes place until the formerly active node is powered off.

Going Into Fencing Mode

To deal with a suspected inactive head or split brain, a passive head node that notices that its active counterpart is no longer responding, first goes into *fencing* mode from that time onwards. While a node is fencing, it will try to obtain proof via another method that its counterpart is indeed inactive.

Fencing, incidentally, refers to the way all subsequent actions are tagged and effectively fenced-off as a backlog of actions to be carried out later. It does not refer to a thrust-and-parry imagery derived from fencing swordplay.

Ensuring That The Unresponsive Active Is Indeed Inactive

There are two ways in which "proof" can be obtained that an unresponsive active is inactive:

1. By asking the administrator to manually confirm that the active head node is indeed powered off

2. By performing a power-off operation on the active head node, and then checking that the power is indeed off. This is also referred to as a STONITH (Shoot The Other Node In The Head) procedure

Once a guarantee has been obtained that the active head node is powered off, the fencing head node (i.e. the previously passive head node) moves to active mode.

Improving The Decision To Initiate A Failover With A Quorum Process

While the preceding approach guarantees one active head, a problem remains.

In situations where the passive head node loses its connectivity to the active head node, but the active head node is communicating without a problem to the entire cluster, there is no reason to initiate a failover. It can even result in undesirable situations where the cluster is rendered unusable if, for example, a passive head node decides to power down an active head node just because the passive head node is unable to communicate

with any of the outside world (except for the PDU feeding the active head node).

One technique used to reduce the chances of a passive head node powering off an active head node unnecessarily is to have the passive head node carry out a quorum procedure. All nodes in the cluster are asked by the passive node to confirm that they also cannot communicate with the active head node. If more than half of the total number of nodes confirm that they are also unable to communicate with the active head node, then the passive head node initiates the STONITH procedure and moves to active mode.

13.1.7 Automatic Vs Manual Failover

Administrators have a choice between creating an HA setup with automatic or manual failover. In case of automatic failover, an active head node is powered off when it is no longer responding and a failover sequence is initiated automatically.

In case of manual failover, the administrator is responsible for initiating the failover when the active head node is no longer responding. No automatic power off is done, so the administrator is asked to confirm that the previously active node is powered off.

For automatic failover to be possible, power control should be defined for both head nodes. If power control is defined for the head nodes, automatic failover is used by default. However, it is possible to disable automatic failover. In `cmsh` this is done by setting the `disableautomaticfailover` property:

```
[root@bright52 ~]# cmsh
[bright52]% partition failover base
[bright52->partition[base]->failover]% set disableautomaticfailover yes
[bright52->partition*[base*]->failover*]% commit
```

With `cmgui` it is done by selecting the cluster resource, then selecting the `Failover` tab. Within the tab, the "`Disable automatic failover`" checkbox is ticked, and the change saved with a click on the "`Save`" button (figure 13.2).

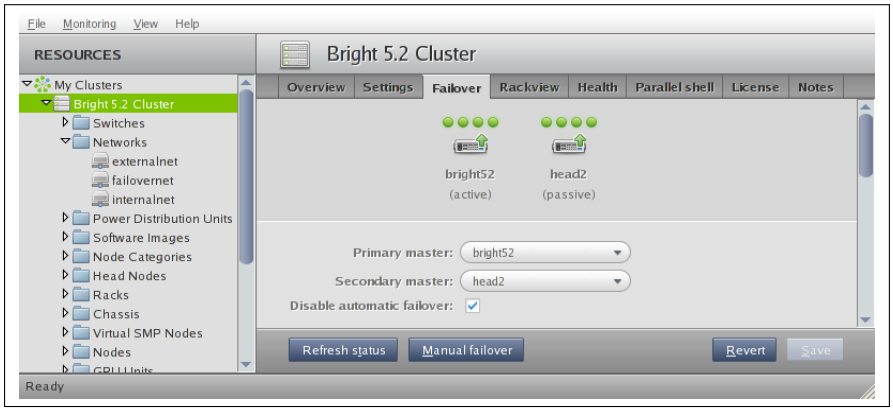

Figure 13.2: `cmgui` High Availability: Disable Automatic Failover

If no power control has been defined, or if automatic failover has been disabled, a failover sequence must always be initiated manually by the administrator.

13.2 HA Setup Procedure Using `cmha-setup`

After a cluster has been installed using the procedure described in Chapter 2, the administrator has the choice of running the cluster with a single head node or performing an HA setup. This section describes `cmha-setup`, a special tool that guides the administrator in building an HA setup.

The `cmha-setup` utility interacts with the cluster management environment using `cmsh` to create an HA setup. Although it is also possible to create an HA setup manually too, using either `cmgui` or `cmsh`, it is not recommended as it is error-prone.

A basic HA setup is created in three stages:

1. **Preparation** (section 13.2.1): the configuration parameters are set for the shared interface and for the secondary head node that is about to be installed.
2. **Cloning** (section 13.2.2): the secondary head node is installed by cloning it from the primary head node.
3. **Shared Storage Setup** (section 13.2.3): the method for shared storage is chosen and set up.

An optional extra stage is:

4. **Automated Failover Setup** (section 13.2.4): Power control to allow automated failover is set up.

13.2.1 Preparation

The following steps prepare the primary head node for the cloning of the secondary. The preparation is done only on the primary, so that the presence of the secondary is not actually needed during this stage.

0. All nodes except for the primary head node are powered off.
1. To start the HA setup, the `cmha-setup` command is run from a `root` shell on the primary head node.
2. `Setup` is selected from the main menu (figure 13.3).
3. `Configure` is selected from the `Setup` menu.
4. The virtual shared internal IP address and interface alias name are set.
5. The virtual shared external IP address and interface alias name are set.
6. The primary head node may have other network interfaces (e.g. InfiniBand interfaces, IPMI interface, alias interface on the IPMI network). These interfaces are also created on the secondary head node, but the IP address of the interfaces still need to be configured. For each such interface, when prompted, a unique IP address for the secondary head node is configured.
7. The hostname is entered for the secondary head node

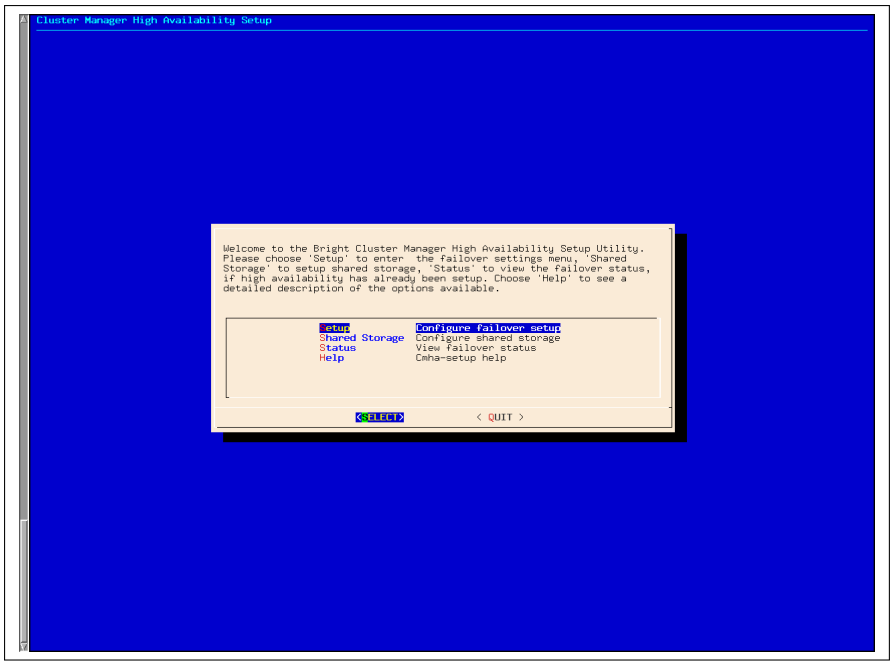

Figure 13.3: cmha-setup Main menu

8. The failover network parameters are created (network name, base address, netmask and domain name). This is the network used for heartbeat monitoring.
9. The failover network IP addresses and interface names are set for the primary and secondary head nodes.
10. The (direct, non-virtual) internal and external IP addresses for the secondary are set.
11. A summary screen displays the planned failover configuration. If alterations need to be made, they can be done via the next step.
12. The administrator is prompted to set the planned failover configuration. If it is not set, the main menu of cmha-setup is redisplayed.
13. If the option to set the planned failover configuration is chosen, then a password for the mysql server is requested. The procedure continues further after the password is entered.
14. Setup progress for the planned configuration is displayed.
15. Instructions on what to run on the secondary to clone it from the primary are displayed (figure 13.4).

13.2.2 Cloning

After the preparation has been done by configuring parameters as outlined in section 13.2.1, the cloning of the head nodes is carried out. In the cloning instructions that follow, the active node refers to the primary node and the passive node refers to the secondary node. However this correlation is only true for when an HA setup is created for the first time, and it is not necessarily true if head nodes are replaced later on by cloning.

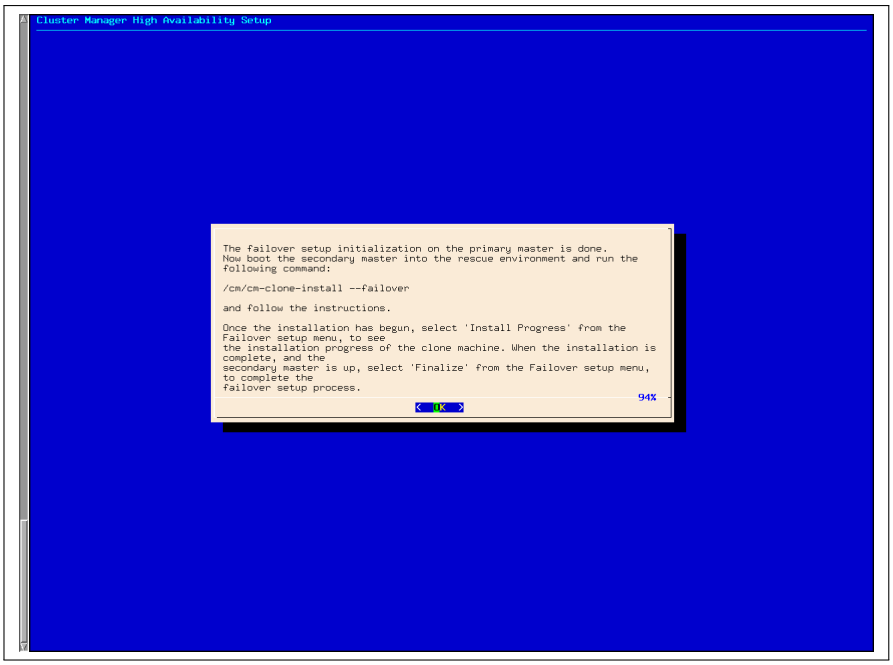

Figure 13.4: `cmha-setup` Instructions To Run On Secondary For Cloning

These cloning instructions may also be repeated later on if a passive head node ever needs to be replaced, for example, if the hardware is defective. In that case the active head node can be either the primary or secondary.

The process described PXE boots the passive from the active, thereby loading a special rescue image from the active that allows cloning from the active to the passive to take place.

1. The passive head node is PXE booted off the internal cluster network, from the active head node. It is highly recommended that the active and passive head nodes have identical hardware configurations. Typically, the BIOS of both head nodes is configured so that a hard disk boot is attempted first, and a PXE boot is attempted after a hard disk boot failure, leading to the `Cluster Manager PXE Environment` menu of options. This menu has a 5s time-out.

2. In the `Cluster Manager PXE Environment` menu, before the 5s time-out, "`Start Rescue Environment`" is selected to boot the node into a Linux ramdisk environment.

3. Once the rescue environment has finished booting, a login as root is done. No password is required (figure 13.5).

4. The following command is executed (figure 13.6):
   `/cm/cm-clone-install --failover`

5. When prompted to enter a network interface to use, the interface that was used to boot from the internal cluster network (e.g. `eth0`, `eth1`, ...) is entered. There is often uncertainty about what interface name corresponds to what physical port. This can be resolved by switching to another console and using "`ethtool -p <interface>`", which makes the NIC corresponding to the interface blink.

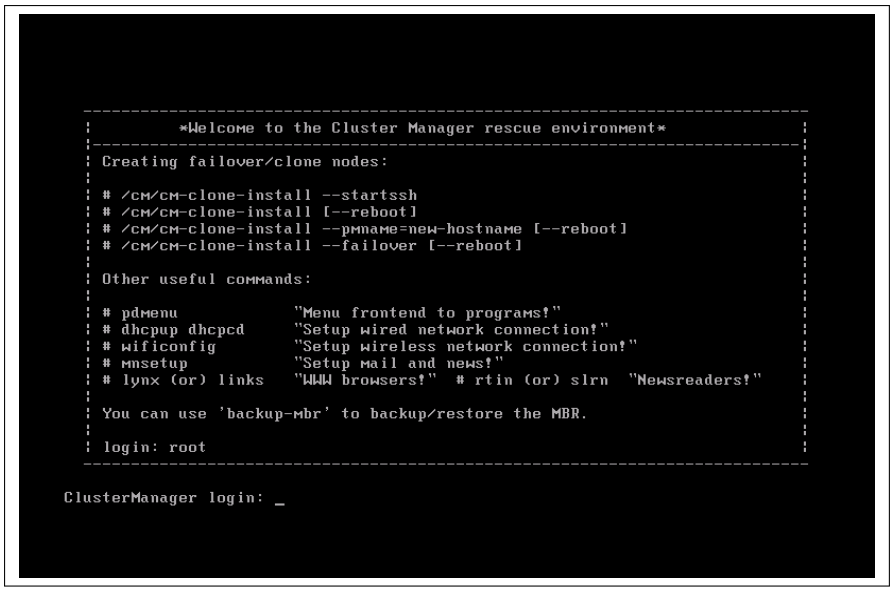

Figure 13.5: Login Screen After Booting Passive Into Rescue Mode From Active

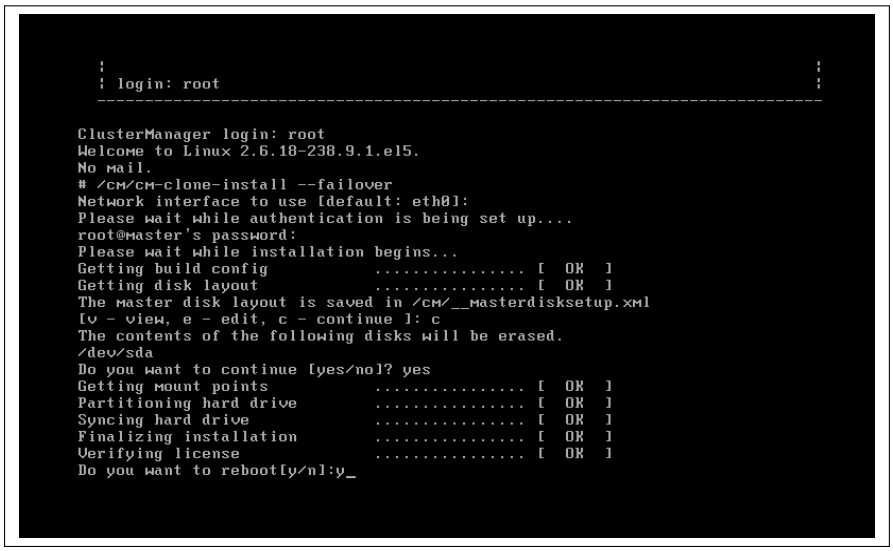

Figure 13.6: Cloning The Passive From The Active Via A Rescue Mode Session

6. If the provided network interface is correct, a `root@master's password` prompt appears. The administrator should enter the root password.

7. An opportunity to view or edit the master disk layout is offered.

8. A confirmation that the contents of the specified disk are to be erased is asked for.

9. The cloning takes place. The "syncing" stage usually takes the most time. Cloning progress can also be viewed on the active by selecting the "`Install Progress`" option from the `Setup` menu. When viewing progress using this option, the display is automatically updated

as changes occur.

10. After the cloning process has finished, a prompt at the console of the passive asks if a reboot is to be carried out. A "y" is typed in response to this. The passive node should be set to reboot off its hard drive. This may require an interrupt during reboot, to enter a change in the BIOS setting, if for example, the passive node is set to network boot first.

11. Continuing on now on the active head node, `Finalize` is selected from the `Setup` menu of `cmha-setup`.

12. The mysql root password is requested. After entering the mysql password, the progress of the `Finalize` procedure is displayed, and the cloning procedure continues.

13. The cloning procedure of `cmha-setup` pauses to offer the option to reboot the passive. The administrator should accept the reboot option. After reboot, the cloning procedure is complete. The administrator can then go to the `main` menu and `quit` from there or go on to configure "`Shared Storage`" (section 13.2.3) from there.

A check can already be done at this stage on the failover status of the head nodes with the `cmha` command, run from either head node:

Example

```
[root@bright52 ~]# cmha status
Node Status: running in active master mode

Failover status:
bright52* -> master2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
master2 -> bright52*
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

Here, the `mysql`, `ping` and `status` states indicate that HA setup completed successfully. The `backupping` (backup ping) state uses the dedicated failover network route for its checks, and starts working as soon as the passive head node has been rebooted.

13.2.3 Shared Storage Setup

After cloning the head node (section 13.2.2), the last basic stage of creating an HA setup is setting up shared storage. The available shared storage forms are NAS, DAS, and DRBD.

NAS

1. In the `cmha-setup` main menu, the "`Shared Storage`" option is selected.

2. NAS is selected.
3. The parts of the head node filesystem that are to be copied to the NAS filesystems are selected. The point in the filesystem where the copying is done is the future mount path where the NAS will share the shared filesystem to.
4. The NFS hostname is configured. Also, for each head node filesystem that is to be copied to the NAS filesystem, there is an associated path on the NAS filesystem where the share is to be served from. These NFS volume paths are now configured.
5. If the configured NFS filesystems can be correctly mounted from the NAS server, the process of copying the local filesystems onto the NAS server begins.

DAS

1. In the cmha-setup main menu, the "Shared Storage" option is selected.
2. DAS is selected.
3. The parts of the filesystem that should be placed on shared DAS filesystems are selected.
4. The hostnames of the primary and secondary head nodes and the physical disk partitions to use on both head nodes are entered.
5. That the contents of the listed partitions can be erased on both head nodes should be confirmed. After filesystems have been created, the current contents of the shared directories are copied onto the shared filesystems and the shared filesystems are mounted over the old non-shared filesystems.

DRBD

1. In the cmha-setup main menu, the "Shared Storage" option is selected.
2. DRBD is selected.
3. The parts of the filesystem that should be placed on DRBD filesystems are selected.
4. The hostnames of the primary and secondary head nodes and the physical disk partitions to use on both head nodes are entered.
5. That the contents of the listed partitions can be erased on both head nodes is confirmed. After DRBD based filesystems have been created, the current contents of the shared directories are copied onto the DRBD based filesystems and the DRBD based filesystems are mounted over the old non-shared filesystems.
6. Once the setup process has completed, "DRBD Status/Overview" is selected to verify the status of the DRBD block devices.

13.2.4 Automated Failover

For automatic failover to work, the two head nodes must be able to power off their counterpart. This is done by setting up power control (Chapter 5).

The "`device power status`" command in `cmsh` can be used to verify that power control is functional:

Example

```
[master1]% device power status -n mycluster1,mycluster2
 apc03:21 ..... [   ON    ] mycluster1
 apc04:18 ..... [   ON    ] mycluster2
```

If IPMI is used for power control, it is possible that a head node is not able to reach its own IPMI interface over the network. This is especially true when no dedicated IPMI network port is used. In this case, `cmsh -c "device power status"` reports a failure for the active head node. This does not necessarily mean that the head nodes cannot reach the IPMI interface of their counterpart. Pinging an IPMI interface can be used to verify that the IPMI interface of a head node is reachable from its counterpart.

Example

Verifying that the IPMI interface of `mycluster2` is reachable from `mycluster1`:

```
[root@mycluster1 ~]# ping -c 1 mycluster2.ipmi.cluster
PING mycluster2.ipmi.cluster (10.148.255.253) 56(84) bytes of data.
64 bytes from mycluster2.ipmi.cluster (10.148.255.253): icmp_seq=1
ttl=64 time=0.033 ms
```

Verifying that the IPMI interface of `mycluster1` is reachable from `mycluster2`:

```
[root@mycluster2 ~]# ping -c 1 mycluster1.ipmi.cluster
PING mycluster1.ipmi.cluster (10.148.255.254) 56(84) bytes of data.
64 bytes from mycluster1.ipmi.cluster (10.148.255.254): icmp_seq=1
ttl=64 time=0.028 ms
```

While testing an HA setup with automated failover, it can be useful to simulate a kernel crash on one of the head nodes. The following command crashes a head node instantly:

```
echo c > /proc/sysrq-trigger
```

After the active head node freezes as a result of the crash, the passive head node powers off the machine that has frozen and switches to active mode.

13.3 Managing HA

Once an HA setup has been created, the tools in this section can be used to manage the HA aspects of the cluster.

13.3.1 Changing An Existing Failover Configuration

Changing an existing failover configuration is usually done most simply by running through the HA setup procedure of section 13.2 again, with one exception. The exception is that the existing failover configuration must be removed by using the "`Undo Failover`" menu option between steps 2 and 3 of the procedure described in section 13.2.1.

13.3.2 cmha Utility

A major command-line utility for interacting with the HA subsystem is cmha. Its usage information is:

```
[root@mycluster1 ~]# cmha
Usage: /cm/local/apps/cmd/sbin/cmha status | makeactive | dbreclone <node>
```

The options do the following:

- **status**: query the status of the HA subsystem on the local machine
- **makeactive**: initiate failover manually, making the current machine active
- **dbreclone**: clone the cmdaemon database across head nodes

These options are looked at in greater detail next:

status: Querying HA Status

Information on the failover status is displayed thus:

Example

```
[root@mycluster1 ~]# cmha status
Node Status: running in active master mode

Failover status:
mycluster1* -> mycluster2
  backupping   [  OK  ]
  mysql        [  OK  ]
  ping         [  OK  ]
  status       [  OK  ]
mycluster2 -> mycluster1*
  backupping   [  OK  ]
  mysql        [  OK  ]
  ping         [  OK  ]
  status       [  OK  ]
```

The * in the output indicates the head node which is currently active. The status output shows 4 aspects of the HA subsystem from the perspective of each head nodes:

HA Status	Description
backupping	the other head node is reachable via the dedicated failover network. This backup ping uses the failover route instead of the internal net route.
mysql	MySQL replication status
ping	the other head node is reachable over the primary management network
status	CMDaemon running on the other head node responds to SOAP calls

By default, Bright Cluster Manager prepares to carry out the failover sequence when all three of ping, backupping and status are not OK on a head node. If automatic failover is enabled, the failover completes automatically; otherwise a manual failover must be done (section 13.1.7).

`makeactive`: Initiate Failover

Example

To initiate a failover manually:

```
[root@mycluster2 ~]# cmha makeactive
Proceeding will initiate a failover sequence which will make this node
(mycluster2) the active master.

Are you sure ? [Y/N]
y
Your session ended because: CMDaemon failover, no longer master
mycluster2 became active master, reconnecting your cmsh ...
```

The status information and makeactive functionalities of `cmha` are also accessible via `cmgui`, as described in section 13.3.7.

`dbreclone`: Cloning The CMDaemon Database

The `dbreclone` option of `cmha` clones the CMDaemon state database from the head node on which `cmha` runs to the other head node. This may be useful to run if the mysql CMDaemon state database tables are unsalvageably corrupted on the destination node, and the source node has a known good database state.

Example

```
[root@bright52 ~]# cmha status
Node Status: running in active master mode

Failover status:
bright52* -> head2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
head2 -> bright52*
  backupping    [  OK  ]
  mysql         [FAILED] (11)
  ping          [  OK  ]
  status        [  OK  ]
[root@bright52 ~]# cmha dbreclone head2
Proceeding will cause the contents of the cmdaemon state database on he\
ad2 to be resynchronized from this node (i.e. bright52 -> head2)

Are you sure ? [Y/N]
Y
Waiting for CMDaemon (3113) to terminate...
[  OK  ]
Waiting for CMDaemon (7967) to terminate...
                                                        [  OK  ]
cmdaemon.dump.8853.sql                      100%  253KB 252.9KB/s   00:00
slurmacctdb.dump.8853.sql                   100%   11KB  10.7KB/s   00:00
Waiting for CMDaemon to start...                        [  OK  ]
Waiting for CMDaemon to start...[  OK  ]
[root@bright52 ~]# cmha status
Node Status: running in active master mode
```

```
Failover status:
bright52* -> head2
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
head2 -> bright52*
  backupping    [  OK  ]
  mysql         [  OK  ]
  ping          [  OK  ]
  status        [  OK  ]
```

13.3.3 States

The state a head node is in can be determined in three different ways:

1 By looking at the message being displayed at login time.

Example

```
--------------------------------------------------------------------------------

Node Status: running in active master mode

--------------------------------------------------------------------------------
```

2 By executing `cmha status`.

Example

```
[root@mycluster ~]# cmha status
Node Status: running in active master mode
...
```

3 By examining `/var/spool/cmdaemon/state`.

There are a number of possible states that a head node can be in:

State	Description
INIT	Head node is initializing
FENCING	Head node is trying to determine whether it should try to become active
ACTIVE	Head node is in active mode
PASSIVE	Head node is in passive mode
BECOMEACTIVE	Head node is in the process of becoming active
BECOMEPASSIVE	Head node is in the process of becoming passive
UNABLETOBECOMEACTIVE	Head node tried to become active but failed
ERROR	Head node is in error state due to unknown problem

Especially when developing custom mount and unmount scripts, it is quite possible for a head node to go into the UNABLETOBECOMEACTIVE state. This generally means that the mount and/or unmount script are not working properly or are returning incorrect exit codes. To debug these situations, it is helpful to examine the output in /var/log/cmdaemon. The "cmha makeactive" shell command can be used to instruct a head node to become active again.

13.3.4 Failover Action Decisions

A table summarizing the scenarios that decide when a passive head should take over is helpful:

Event on active	Reaction on passive	Reason
Reboot	Nothing	Event is usually an administrator action. To make the passive turn active, an administrator would run "cmha makeactive" on it.
Shutdown	Nothing	As above.
Unusably sluggish or freezing system but still pingable	Nothing	1. Active may still unfreeze. 2. Shared filesystems may still be in use by the active. Concurrent use by the passive taking over therefore risks corruption.
Become passive in response to "cmha makeactive" run on passive	Become active when former active becomes passive	As ordered by administrator
Active dies	Quorum called, may lead to passive becoming new active	Confirms if active head is dead according to other nodes too. If so, then a "power off" command is sent to it. If the command is successful, the passive head becomes the new active head.

13.3.5 Keeping Head Nodes In Sync

What Should Be Kept In Sync?

If filesystem changes are made on an active head node without using CMDaemon (cmsh or cmgui), and if the changes are outside the shared

filesystem, then these changes should normally also be made on the passive head node. For example:

- RPM installations/updates (section 9.2)
- Applications installed locally
- Files (such as drivers or values) placed in the `/cm/node-installer/` directory and referred to by `initialize` (section 6.4.5) and `finalize` scripts (section 6.4.11)
- Any other configuration file changes outside of the shared filesystems

If the cluster is being built on bare metal, a sensible way to minimize the amount of work to be done is to install a single head cluster first. All packages and applications should then be placed, updated and configured on that single head node until it is in a satisfactory state. Only then should HA be set up as described in section 13.2, where the cloning of data from the initial head node to the secondary is described. The result is then that the secondary node gets a well-prepared system with the effort to prepare it having only been carried out once.

Avoiding Encounters With The Old Filesystems

It should be noted that when the shared storage setup is made, the contents of the shared directories (at that time) are copied over from the local filesystem to the newly created shared filesystems. The shared filesystems are then mounted on the mountpoints on the active head node, effectively hiding the local contents.

Since the shared filesystems are only mounted on the active machine, the old filesystem contents remain visible when a head node is operating in passive mode. Logging into the passive head node may thus confuse users and is therefore best avoided.

Updating Services On The Head Nodes And Associated Syncing

The services running on the head nodes described in section 13.1.3 should also have their packages updated on both head nodes.

For the services that run simultaneously on the head nodes, such as CMDaemon, DHCP, LDAP, MySql, NTP and DNS, their packages should be updated on both head nodes at about the same time. A suggested procedure is to stop the service on both nodes around the same time, update the service and ensure that it is restarted.

The provisioning node service is part of the CMDaemon package. The service updates images from the active head node to all provisioning nodes, including the passive head node, if the administrator runs the command to update provisioners. How to update provisioners is described in section 13.1.3.

For services that migrate across head nodes during failover, such as NFS and Workload Management, it is recommended (but not mandated) to carry out this procedure: the package on the passive node (called the secondary for the sake of this example) is updated to check for any broken package behavior. The secondary is then made active with "`cmha makeactive`" (section 13.3.2), which automatically migrates users cleanly off from being servced by the active to the secondary. The package is then

updated on the primary. If desired, the primary can then be made active again. The reason for recommending this procedure for services that migrate is that, in case the update has issues, the situation can be inspected somewhat better with this procedure.

13.3.6 High Availability Parameters

There are several HA-related parameters that can be tuned. Accessing these via `cmgui` is described in section 13.3.7. In `cmsh` the settings can be accessed in the `failover` submode of the `base` partition.

Example

```
[mycluster1]% partition failover base
[mycluster1->partition[base]->failover]% show
Parameter                      Value
------------------------------ ----------------------------------------
Dead time                      10
Disable automatic failover     no
Failover network               failovernet
Init dead                      30
Keep alive                     1
Mount script
Postfailover script
Prefailover script
Quorum time                    60
Revision
Secondary master
Unmount script
Warn time                      5
```

`Dead time`
When a passive head node determines that the active head node is not responding to any of the periodic checks for a period longer than the "`Dead time`" seconds, the active head node is considered dead and a quorum procedure starts. Depending on the outcome of the quorum, a failover sequence may be initiated.

`Disable automatic failover`
Setting this to yes disables automated failover. Section 13.1.7 covers this further.

`Failover network`
The "`Failover network`" setting determines which network is used as a dedicated network for the `backupping` (backup ping) heartbeat check. The heartbeat connection is normally a direct cable from a NIC on one head node to a NIC on the other head node. The network can be selected via tab-completion suggestions. By default, without a dedicated failover network, the possibilities are nothing, `externalnet` and `internalnet`.

`Init dead`
When head nodes are booted simultaneously, the standard "`Dead time`" might be too strict if one head node requires a bit more time for booting than the other. For this reason, when a head node boots (or more exactly, when the cluster management daemon is starting), a time of "`Init dead`"

seconds is used rather than the "Dead time" to determine whether the other node is alive.

Keep alive
The "Keep alive" value is the time interval, in seconds, over which the passive head node carries out a check that the active head node is still up. If a dedicated failover network is used, 3 separate heartbeat checks are carried out to determine if a head node is reachable.

Mount script
The script pointed to by the "Mount script" setting is responsible for bringing up and mounting the shared filesystems.

Postfailover script
The script pointed to by the "Postfailover script" setting is run by cmdaemon on both head nodes. The script first runs on the head that is now passive, then on the head that is now active. It runs as soon as the former passive has become active. It is typically used by scripts mounting an NFS shared storage so that no more than one head node exports a filesystem to NFS clients at a time.

Prefailover script
The script pointed to by the "Prefailover script" setting is run by cmdaemon on both head nodes. The script first runs on the (still) active head, then on the (still) passive head. It runs as soon as the decision for the passive to become active has been made, but before the changes are implemented. It is typically used by scripts unmounting an NFS shared storage so that no more than one head node exports a filesystem to NFS clients at a time.

Quorum time
When a node is asked what head nodes it is able to reach over the network, the node has "Quorum time" seconds to respond. If a node does not respond to a call for quorum within that time, it is no longer considered for the results of the quorum check.

Secondary master
The "Secondary master" setting is used to define the secondary head node to the cluster.

Unmount script
The script pointed to by the "Unmount script" setting is responsible for bringing down and unmounting the shared filesystems.

Warn time
When a passive head node determines that the active head node is not responding to any of the periodic checks for a period longer than "Warn time" seconds, a warning is logged that the active head node might become unreachable soon.

13.3.7 Handling And Viewing Failover Via cmgui

Accessing cmha Functionality Via cmgui

The equivalent functions of cmha (section 13.3.2) are available under cmgui by selecting a cluster from the resource tree, and then choosing the Failover tab (figure 13.7).

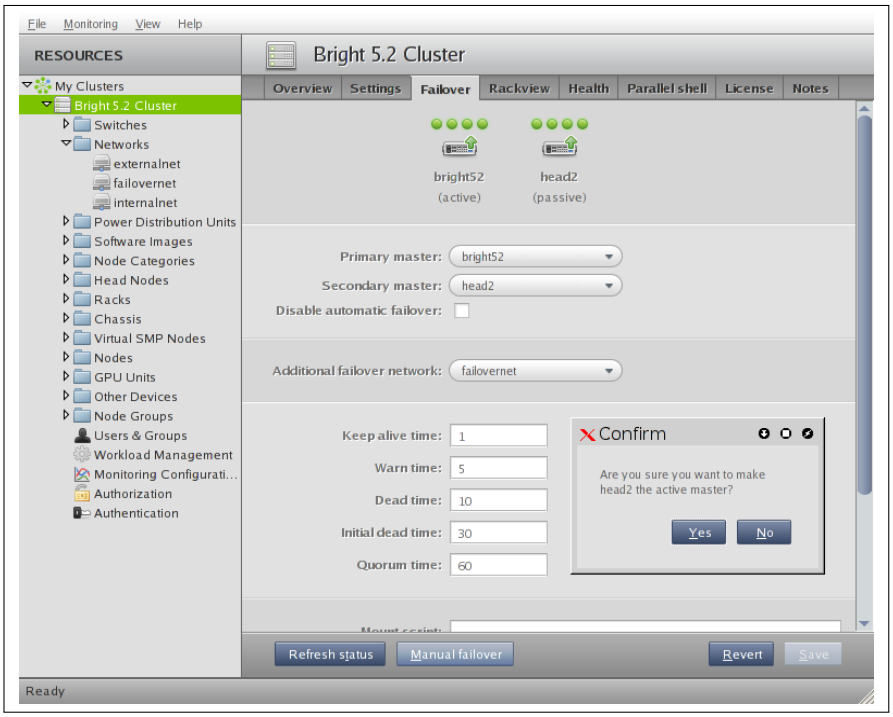

Figure 13.7: Accessing HA Cluster Parameters And Functions Via cmgui

The states of the head nodes are indicated in the first section of the tabbed pane, where the machines are shown along with their modes and states, and with LED lights. Hovering the mouse cursor over the LED lights causes hovertext to appear, indicating which check is associated with which light.

Manual failover can be initiated from cmgui by clicking on the "Manaul Failover" button (figure 13.7).

Accessing cmsh HA Parameters (partition failover base) Via cmgui

The cmgui equivalents of the cmsh HA parameters in section 13.3.6 are accessed from the same cmgui tab as described earlier in this section (13.3.7).

13.3.8 Re-cloning A Head Node

Some time after an HA setup has gone into production, it may become necessary to re-install one of the head nodes, for example if one of the head nodes were replaced due to hardware failure.

To re-clone a head node from an existing active head node, cmha-setup is entered, Setup is selected, and then "Clone Install" is selected. The displayed instructions are then followed (i.e. the instructions in section 13.2.2 are repeated).

If the MAC address of one of the head nodes has changed, it is typically necessary to request a new license. Section 4.1.3 has details on obtaining a new license.

A

Generated Files

This appendix contains a list of all system configuration files which are generated automatically.

Section 3.6.3 describes how system configuration files on all nodes are written out using the Cluster Management Daemon. The Cluster Management Daemon is introduced in section 3.6.3 and its configuration directives are listed in Appendix C.

All of these configuration files may be listed as `Frozen Files` in the Cluster Management Daemon configuration file to prevent them from being generated automatically.

A.1 Files Generated Automatically On Head Nodes

Files generated automatically on head nodes

File	Generated By	Method	Comment
/etc/resolv.conf	CMDaemon	Section	
/etc/HOSTNAME	CMDaemon	Entire file	SUSE only
/etc/localtime	CMDaemon	Entire file	Copied from zoneinfo
/etc/exports	CMDaemon	Section	
/etc/fstab	CMDaemon	Section	
/etc/hosts	CMDaemon	Section	
/etc/hosts.equiv	CMDaemon	Section	
/tftpboot/mtu.conf	CMDaemon	Entire file	Bright configuration
/etc/sysconfig/ipmicfg	CMDaemon	Entire file	Bright configuration
/etc/sysconfig/network/config	CMDaemon	Section	SUSE only
/etc/sysconfig/network/routes	CMDaemon	Section	SUSE only
/etc/sysconfig/network/ifcfg-*	CMDaemon	Section	SUSE only
/etc/sysconfig/network/dhcp	CMDaemon	Section	SUSE only
/etc/sysconfig/network	CMDaemon	Section	Red Hat only
/etc/sysconfig/network-scripts/ifcfg-*	CMDaemon	Section	Red Hat only
/etc/dhclient.conf	CMDaemon	Entire file	Red Hat only
/etc/dhcpd.conf	Entire file		
/etc/dhcpd.internalnet.conf	Entire file		
/etc/shorewall/interfaces	CMDaemon	Section	
/etc/shorewall/masq	CMDaemon	Section	
/etc/slurm/gres.conf	CMDaemon	Section	
/etc/sysconfig/clock	CMDaemon	Section	
/etc/postfix/canonical	CMDaemon	Section	
/etc/postfix/main.cf	CMDaemon	Section	
/etc/postfix/generic	CMDaemon	Section	
/etc/aliases	CMDaemon	Section	
/etc/ntp.conf	CMDaemon	Section	
/etc/ntp/step-tickers	CMDaemon	Section	Red Hat only
/etc/named.conf	CMDaemon	Entire file	For custom additions use /etc/named.conf.include
/var/named/*.zone	CMDaemon	Entire file	Red Hat only
/var/lib/named/*.zone	CMDaemon	Entire file	SUSE only

A.2 Files Generated Automatically In Software Images

Files generated automatically in software images

File	Generated By	Method	Comment
/etc/localtime	CMDaemon	Entire file	
/etc/hosts	CMDaemon	Section	
/etc/sysconfig/ipmicfg	CMDaemon	Entire file	
/etc/sysconfig/clock	CMDaemon	Section	
/etc/sysconfig/kernel	CMDaemon	Section	SUSE only
/etc/sysconfig/network/config	CMDaemon	Section	SUSE only
/etc/sysconfig/network/routes	CMDaemon	Section	SUSE only
/boot/vmlinuz	CMDaemon	Symlink	
/boot/initrd	CMDaemon	Symlink	
/boot/initrd-*	CMDaemon	Entire file	
/etc/modprobe.conf	CMDaemon	Section	
/etc/postfix/main.cf	CMDaemon	Section	
/etc/aliases	CMDaemon	Section	

A.3 Files Generated Automatically On Regular Nodes

Files generated automatically on regular nodes

File	Generated By	Method	Comment
/etc/hosts	Node-installer	Section	
/etc/exports	CMDaemon	Section	
/etc/fstab	Node-installer	Section	
/etc/ntp.conf	Node-installer	Entire file	
/etc/ntp/step-tickers	Node-installer	Entire file	
/etc/postfix/main.cf	Node-installer	Section	
/etc/resolv.conf	Node-installer	Entire file	
/etc/sysconfig/network	Node-installer	Entire file	
/etc/sysconfig/network/ifcfg-*	Node-installer	Entire file	SUSE only, not ifcfg-lo
/etc/sysconfig/network-scripts/ifcfg-*	Node-installer	Entire file	Red Hat only, not ifcfg-lo
/etc/HOSTNAME	Node-installer	Entire file	

B

Bright Computing Public Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.0 (GNU/Linux)

mQGiBEqtYegRBADStdQjn1XxbYorXbFGncF2IcMFiNA7hamARt4w7hjtwZoKGHbC
zSLsQTmgZO+FZs+tXcZa5OLjGwhpxT6qhCe8Y7zIh2vwKrKlaAVKj2PUU28vKj1p
2W/OIiG/HKLtahLiCkOL3ahPOevJHh8B7elClrZOTKTBB6qIUbC5vHtjiwCgydm3
THLJsKnwk4qZetluTupldOEEANCzJ1nZxZzN6ZAMkIBrct8GivWClT1nBG4UwjHd
EDcGlREJxpg/OhpEP8TY1eOYUKRWvMqSVChPzkLUTIsd/O4RGTwOPGCo6Q3TLXpM
RVoonYPR1tRymPNZyW8VJeTUEnOkdlCaqZykp1sRb3jFAiJIRCmBRc854i/jRXmo
foTPBACJQyoEH9Qfe3VcqR6+vR2tX9lPvkxS7A5AnJIRs3Sv6yM4oV+7k/HrfYKt
fyl6widtEbQ1870s4x3NYXmmne7lz1nGxBfAxzPG9rtjRSXyVxc+KGVd6gKeCV6d
o7kS/LJHRiOLb5G4NZRFy5CGqg64liJwp/f2J4uyRbC8b+/LQbQ7QnJpZ2hOIENv
bXB1dGluZyBEZXZlbG9wbWVudCBUZWFtIDxkZXZAYnJpZ2hOY29tcHVOaW5nLmNv
bT6IXgQTEQIAHgUCSq1h6AIbAwYLCQgHAwIDFQIDAxYCAQIeAQIXgAAKCRDvaS9m
+k3mOJOOAKCOGLTZiqoCQ6TRWW2ijjITEQ8CXACgg3o4oVbrG67VFzHUntcAOYTE
DXW5AgOESq1h6xAIAMJiaZI/OEqnrhSfiMsMT3sxz3mZkrQQL82Fob7s+S7nnMl8
A8btPzLlK8NzZytCglrIwPCYG6vfza/nkvyKEPh/f2it941bh7qiu4rBLqr+kGx3
zepSMRqIzW5FpIrUgDZOL9J+tWSSUtPWOYQ5jBBJrgJ8LQy9dK2RhAOLuHfbOSVB
JLIwNKxafkhMRwDoUNS4BiZKWyPFu47vd8fM67IPT1nMlOiCOR/QBn29MYuWnBcw
61344pd/IjOu3gM6YBqmRRU6yBeViOTxxbYYnWcts6tEGAlTjHUOQ7gxVp4RDia2
jLVtbee8H464wxkkC3SSkng216RaBBAoaAykhzcAAwUH/iG4WsJHFw3+CRhUqy51
jnmb1FTFO8KQXI8JlPXMOh6vvOPtP5rw5D5V2cyVe2i4ez9Y8XMVfcbf6OlptKyY
bRUjQq+9SNjt12ESU67YyLstSN68ach9Af03PoSZIKkiNwfAO+VBILv2Mhn7xd74
5LOM/eJ7lHSpeJA2Rzs6szc234Ob/VxGfGWjogaK3NElSYOzQo+/kOVMdMWsQm/8
Ras19IA9P5jlSbcZQlHlPjndS4x4XQ8P41ATczsIDyWhsJC51rTuw9/QO7fqvvPn
xsRz1pFmiiN7I4JLjwOnAlXexn4EaeVa7Eb+uTjvxJZNdShs7Td74OmlF7RKFccI
wLuISQQYEQIACQUCSq1h6wIbDAAKCRDvaS9m+k3mOC/oAJsHMmKrLPhjCdZyHbB1
e19+5JABUwCfUOPoawBNOHzDnfr3MLaTgCwjsEE=
=WJX7
-----END PGP PUBLIC KEY BLOCK-----
```

C

CMDaemon Configuration File Directives

This Appendix lists all configuration file directives that may be used in the cluster management daemon configuration file:

`/cm/local/apps/cmd/etc/cmd.conf`

To activate changes in a configuration file, the `cmd` service must be restarted, and is normally done with the command:

```
service cmd restart
```

Master directive

Syntax: `Master =` *hostname*
Default: `Master = master`

The cluster management daemon treats the host specified in the `Master` directive as the head node. A cluster management daemon running on a node specified as the head node starts in *head* mode. On a regular node, it starts in *node* mode.

Port directive

Syntax: `Port =` *number*
Default: `Port = 8080`

The *number* used in the syntax above is a number between 0 and 65535. The standard port is 8080.

The `Port` directive controls the non-SSL port that the cluster management daemon listens on. In practice all communication with the cluster management daemon is carried out over the SSL port.

SSLPort directive

Syntax: `SSLPort =` *number*
Default: `SSLPort = 8081`

The *number* used in the syntax above is a number between 0 and 65535. The standard port is 8081.

The SSLPort directive controls the SSL port that the cluster management daemon listens on.

SSLPortOnly directive

Syntax: SSLPortOnly = yes|no
Default: SSLPortOnly = no

The SSLPortOnly directive allows non-SSL port to be disabled. Normally both SSL and non-SSL ports are active although in practice only the SSL port is used.

CertificateFile directive

Syntax: CertificateFile = *filename*
Default: CertificateFile = "/cm/local/apps/cmd/etc/cmd.pem"

The CertificateFile directive specifies the certificate which is to be used for authentication purposes. On the master node, the certificate used also serves as a software license.

PrivateKeyFile directive

Syntax: PrivateKeyFile = *filename*
Default: PrivateKeyFile = "/cm/local/apps/cmd/etc/cmd.key"

The PrivateKeyFile directive specifies the private key which corresponds to the certificate that is being used.

CACertificateFile directive

Syntax: CACertificateFile = *filename*
Default: CACertificateFile = "/cm/local/apps/cmd/etc/cacert.pem"

The CACertificateFile directive specifies the path to the Bright Cluster Manager root certificate. It is normally not necessary to change the root certificate.

RandomSeedFile directive

Syntax: RandomSeedFile = *filename*
Default: RandomSeedFile = "/dev/urandom"

The RandomSeedFile directive specifies the path to a source of randomness.

DHParamFile directive

Syntax: DHParamFile = *filename*
Default: DHParamFile = "/cm/local/apps/cmd/etc/dh1024.pem"

The DHParamFile directive specifies the path to the Diffie-Hellman parameters.

SSLHandshakeTimeout directive

Syntax: `SSLHandshakeTimeout =` *number*
Default: `SSLHandshakeTimeout = 10`

The `SSLHandShakeTimeout` directive controls the time-out period (in seconds) for SSL handshakes.

SSLSessionCacheExpirationTime directive

Syntax: `SSLSessionCacheExpirationTime =` *number*
Default: `SSLSessionCacheExpirationTime = 300`

The `SSLSessionCacheExpirationTime` directive controls the period (in seconds) for which SSL sessions are cached. Specifying the value `0` can be used to disable SSL session caching.

DBHost directive

Syntax: `DBHost =` *hostname*
Default: `DBHost = "localhost"`

The `DBHost` directive specifies the hostname of the MySQL database server.

DBPort directive

Syntax: `DBPort =` *number*
Default: `DBHost = 3306`

The `DBPort` directive specifies the TCP port of the MySQL database server.

DBUser directive

Syntax: `DBUser =` *username*
Default: `DBUser = cmdaemon`

The `DBUser` directive specifies the username used to connect to the MySQL database server.

DBPass directive

Syntax: `DBPass =` *password*
Default: `DBPass = "<random string set during installation>"`

The `DBPass` directive specifies the password used to connect to the MySQL database server.

DBName directive

Syntax: `DBName =` *database*
Default: `DBName = "cmdaemon"`

The `DBName` directive specifies the database used on the MySQL database server to store CMDaemon related configuration and status information.

DBMonName directive

Syntax: `DBMonName =` *database*
Default: `DBMonName = "cmdaemon_mon"`

The `DBMonName` directive specifies the database used on the MySQL database server to store monitoring related data.

DBUnixSocket directive

Syntax: `DBUnixSocket =` *filename*

The `DBUnixSocket` directive specifies the named pipe used to connect to the MySQL database server if it is running on the same machine.

DBUpdateFile directive

Syntax: `DBUpdateFile =` *filename*
Default: `DBUpdateFile = "/cm/local/apps/cmd/etc/cmdaemon_upgrade.sql"`

The `DBUpdateFile` directive specifies the path to the file that contains information on how to upgrade the database from one revision to another.

EventBucket directive

Syntax: `EventBucket =` *filename*
Default: `EventBucket = "/var/spool/cmd/eventbucket"`

The `EventBucket` directive specifies the path to the named pipe that is created to listen for incoming events.

EventBucketFilter directive

Syntax: `EventBucketFilter =` *filename*
Default: `EventBucketFilter = "/cm/local/apps/cmd/etc/eventbucket.filter"`

The `EventBucketFilter` directive specifies the path to the file that contains regular expressions used to filter out incoming messages on the event-bucket.

LDAPHost directive

Syntax: `LDAPHost =` *hostname*
Default: `LDAPHost = "localhost"`

The `LDAPHost` directive specifies the hostname of the LDAP server to connect to for user management.

LDAPUser directive

Syntax: `LDAPUser =` *username*
Default: `LDAPUser = "root"`

The LDAPUser directive specifies the username used when connecting to the LDAP server.

LDAPPass directive

Syntax: LDAPPass = *password*
Default: LDAPPass = "*<random string set during installation>*"

The LDAPPass directive specifies the password used when connecting to the LDAP server.

LDAPSearchDN directive

Syntax: LDAPSearchDN = *dn*
Default: LDAPSearchDN = "dc=cm,dc=cluster"

The LDAPSearchDN directive specifies the Distinguished Name (DN) used when querying the LDAP server.

DocumentRoot directive

Syntax: DocumentRoot = *path*
Default: DocumentRoot = "/cm/local/apps/cmd/etc/htdocs"

The DocumentRoot directive specifies the directory mapped to the web-root of the CMDaemon. The CMDaemon acts as a HTTP-server, and can therefore in principle also be accessed by web-browsers.

SpoolDir directive

Syntax: SpoolDir = *path*
Default: SpoolDir = "/var/spool/cmd"

The SpoolDir directive specifies the directory which is used by the CMDaemon to store temporary and semi-temporary files.

CMDaemonAudit

Syntax: CMDaemonAudit = yes|no
Default: CMDaemonAudit = no

When the CMDaemonAudit directive is set to yes, and a value is set for the CMDaemon auditor file with the CMDaemonAuditorFile directive, then CMDaemon actions are time-stamped and logged in the CMDaemon auditor file.

CMDaemonAuditorFile

Syntax: CMDaemonAuditorFile = *filename*
Default: CMDaemonAuditorFile = "/var/spool/cmd/audit.log"

The CMDaemonAuditorFile directive sets where the audit logs for CMDaemon actions are logged. The log format is:

(*time stamp*) *profile* [*IP-address*] *action* (*unique key*)

Example

```
(Mon Jan 31 12:41:37 2011) Administrator [127.0.0.1] added Profile: arb\
itprof(4294967301)
```

DisableAuditorForProfiles

Syntax: `DisableAuditorForProfiles = {` *profile* [*,profile*]...`}`
Default: `DisableAuditorForProfiles = {node}`

The `DisableAuditorForProfiles` directive sets the profile for which an audit log for CMDaemon actions is disabled. A profile (section 3.3.3) defines the services that CMDaemon provides for that profile user. More than one profile can be set as a comma-separated list. Out of the profiles that are available on a newly-installed system: `node`, `admin`, `cmhealth`, and `readonly`; only the profile `node` is enabled by default. New profiles can also be created via the `profile` mode of `cmsh` or via the `Authorization` resource of `cmgui`, thus making it possible to disable auditing for arbitrary groups of CMDaemon services.

PublicDNS

Syntax: `PublicDNS = true|false`
Default: `PublicDNS = false`

Setting the directive `PublicDNS` to true allows the head node to provide DNS services for any network, and not just the local one.

LockDownDhcpd directive

Syntax: `LockDownDhcpd = true|false`
Default: `LockDownDhcpd = false`

`LockDownDhcpd` is a deprecated legacy directive. If set to true, a global DHCP "deny unknown-clients" option is set. This means no new DHCP leases are granted to unknown clients for all networks. It is deprecated because its globality affects clients on all networks managed by Bright Cluster Manager, which is contrary to the general principle of segregating the network activity of networks.

The recommended way to deny letting new nodes boot up is now to set the option for specific networks by using `cmsh` or `cmgui`. In `cmsh` this is done via the network mode, selecting a network, and then setting a value for `lockdowndhcpd`. In `cmgui` this is done via the `Networks` resource, selecting a network item, and then choosing the `Settings` tabbed pane.

Setting the `cmd.conf` `LockDownDhcpd` directive overrides `lockdowndhcpd` values set by `cmsh` or `cmgui`.

MaxNumberOfProvisioningThreads directive

Syntax: `MaxNumberOfProvisioningThreads =` *number*
Default: `MaxNumberOfProvisioningThreads = 10000`

The MaxNumberOfProvisioningThreads directive specifies the cluster-wide total number of nodes that can be provisioned simultaneously. Individual provisioning servers typically define a much lower bound on the number of nodes that may be provisioned simultaneously.

IpmiSessionTimeout directive

Syntax: IpmiSessionTimeout = *number*
Default: IpmiSessionTimeout = 2000

The IpmiSessionTimeout specifies the time-out for IPMI calls in milliseconds.

SnmpSessionTimeout directive

Syntax: SnmpSessionTimeout = *number*
Default: SnmpSessionTimeout = 500000

The SnmpSessionTimeout specifies the time-out for SNMP calls in microseconds.

PowerOffPDUOutlet directive

Syntax: PowerOffPDUOutlet = true|false
Default: PowerOffPDUOutlet = false

On clusters with both PDU and IPMI power control, the PowerOffPDUOutlet directive when enabled, allows PDU ports to be powered off. Section 5.1.3 has more on this.

MetricAutoDiscover directive

Syntax: MetricAutoDiscover = true|false
Default: MetricAutoDiscover = true

Scan for new hardware components which are not monitored yet and schedule them for monitoring.

UseHWTags directive

Syntax: UseHWTags = true|false
Default: UseHWTags = false

When UseHWTags is set to true, the boot procedure for unknown nodes requires the administrator to enter a HWTag on the console.

DisableBootLogo directive

Syntax: DisableBootLogo = true|false
Default: DisableBootLogo = false

When DisableBootLogo is set to true, the Bright Cluster Manager logo is not displayed on the first boot menu.

StoreBIOSTimeInUTC directive

Syntax: `StoreBIOSTimeInUTC = true|false`
Default: `StoreBIOSTimeInUTC = false`

When `StoreBIOSTimeInUTC` is set to `true`, the system relies on the time being stored in BIOS as being UTC rather than local time.

FreezeChangesToSlurmConfig directive

Syntax: `FreezeChangesToSlurmConfig = true|false`
Default: `FreezeChangesToSlurmConfig = false`

When `FreezeChangesToSlurmConfig` is set to `true`, CMDaemon does not make any modifications to the SLURM configuration.

FreezeChangesToSGEConfig directive

Syntax: `FreezeChangesToSGEConfig = true|false`
Default: `FreezeChangesToSGEConfig = false`

When `FreezeChangesToSGEConfig` is set to `true`, CMDaemon does not make any modifications to the SGE configuration.

FreezeChangesToPBSProConfig directive

Syntax: `FreezeChangesToPBSProConfig = true|false`
Default: `FreezeChangesToPBSProConfig = false`

When `FreezeChangesToPBSProConfig` is set to `true`, CMDaemon does not make any modifications to the PBS Pro configuration.

FreezeChangesToTorqueConfig directive

Syntax: `FreezeChangesToTorqueConfig = true|false`
Default: `FreezeChangesToTorqueConfig = false`

When `FreezeChangesToTorqueConfig` is set to `true`, CMDaemon does not make any modifications to the Torque configuration.

FreezeChangesToLSFConfig directive

Syntax: `FreezeChangesToLSFConfig = true|false`
Default: `FreezeChangesToLSFConfig = false`

When `FreezeChangesToLSFConfig` is set to `true`, CMDaemon does not modify the LSF configuration in the file pointed to by the `LSFProfileScript` directive.

LSFProfileScript= directive

Syntax: `LSFProfileScript =` *filename*
Default: `LSFProfileScript = /cm/shared/apps/lsf/conf/profile.lsf`

LSFProfileScript sets the path to the LSF configuration profile. During LSF installation (section 8.5.5), the value of LSF_TOP used in LSF's install.config file must be set by the administrator to the directory level above /conf/profile.lsf as specified by LSFProfileScript. For the default value of LSFProfileScript, LSF_TOP must therefore be set to /cm/shared/apps/lsf.

ProvisioningNodeAutoUpdate directive

Syntax: ProvisioningNodeAutoUpdate = true|false
Default: ProvisioningNodeAutoUpdate = true

If ProvisioningNodeAutoUpdate is set to true, provisioning nodes are:

1. automatically updated every 24 hours
2. automatically updated when a provisioning request is made, if the ProvisioningNodeAutoUpdateTimer directive allows it

These updates are disabled if ProvisioningNodeAutoUpdate is set to false.

ProvisioningNodeAutoUpdateTimer directive

Syntax: ProvisioningNodeAutoUpdateTimer = *number*
Default: ProvisioningNodeAutoUpdateTimer = 300

When the head node receives a provisioning request, it checks if the last update of the provisioning nodes is more than *number* seconds ago. If this is the case an update is triggered. The update is disabled if ProvisioningNodeAutoUpdate is set to false.

FrozenFile directive

Syntax: FrozenFile = {*filename* [,*filename*]...}
Syntax: FrozenFile = { *filename1*, *filename2* }
Example: FrozenFile = {"/etc/dhcpd.conf","/etc/postfix/main.cf"}

The FrozenFile directive is used to prevent files from being automatically generated. This is useful when site-specific modifications to configuration files have to be made.

SyslogHost directive

Syntax: SyslogHost = *hostname*
Default: SyslogHost = "localhost"

The SyslogHost directive specifies the hostname of the syslog host.

SyslogFacility directive

Syntax: SyslogFacility = *facility*
Default: SyslogFacility = "LOG_LOCAL6"

The default value of LOG_LOCAL6 is set by default in /etc/syslog.conf to redirect messages to /var/log/cmdaemon. The value of *facility* must be one of: LOG_KERN, LOG_USER, LOG_MAIL, LOG_DAEMON, LOG_AUTH, LOG_SYSLOG or LOG_LOCAL0..7

ResolveToExternalName

Syntax: ResolveToExternalName = true|false
Default: ResolveToExternalName = false

The ResolveToExternalName directive determines under which domain name the primary and secondary head nodes are visible from within head nodes. The ResolveToExternalName directive has no effect on compute nodes. Compute nodes always resolve the hostname of any head node to an IP address located on the internal cluster network.

When ResolveToExternalName is set to true, the domain name of the head nodes is set to the domain name of the network that is specified as the "External network" in the base partition in cmsh (the output of "cmsh -c "partition use base; get externalnetwork"'). All head node host names (for example, head1, head2) resolved from within a head node resolve to the head node's IP address located on the external network. This also includes head node hostnames appended with the domain name of the internal cluster network. Only the master hostname and master.*<internalnet>* resolve to the shared IP address on the internal cluster network. The "hostname -f" command then returns a fully qualified domain name (FQDN) of the head node on which it was invoked.

The system configuration files affected by this directive include /etc/hosts and, on SLES systems, also the /etc/HOSTNAME.

A restart of CMDaemon implements any change set for ResolveToExternalName. However the change should not be done while important system services (for example, Torque) are running. Doing so is likely to cause problems with accessing such services due to them then running with a different domain name than the one with which they originally started.

D

Disk Partitioning

Bright Cluster Manager requires that disk partitionings are specified using the XML format that is described in section D.1.

Disk partitioning is initially implemented on the head node and regular nodes during installation (section 2.3.14).

For the head node it cannot be changed from within the Bright Cluster Manager after implementation.

For regular nodes partitioning can be changed after the initial implementation, by changing the XML file defining their partitioning scheme. Diskless operation can also be implemented by using an appropriate XML file.

D.1 Structure Of Partitioning Definition

In Bright Cluster Manager, partitioning setups have their global structure defined using an XML schema. The schema file is installed on the head node in `/cm/node-installer/scripts/disks.xsd`. This section shows the schema and gives some examples of the defined types, while the next sections contain examples of the schema in use:

XML schema for partitioning

```
<?xml version='1.0'?>

<!--
#
# Copyright (c) 2004-2010 Bright Computing, Inc. All Rights Reserved.
#
# This software is the confidential and proprietary information of
# Bright Computing, Inc.("Confidential Information").  You shall not
# disclose such Confidential Information and shall use it only in
# accordance with the terms of the license agreement you entered into
# with Bright Computing, Inc.

This is the XML schema description of the partition layout XML file.
It can be used by software to validate partitioning XML files.
There are however a few things the schema does not check:
- There should be exactly one root mountpoint (/), unless diskless.
- There can only be one partition with a 'max' size on a particular device.
- Something similar applies to logical volumes.
```

```
- The 'auto' size can only be used for a swap partition.
- Partitions of type 'linux swap' should not have a filesystem.
- Partitions of type 'linux raid' should not have a filesystem.
- Partitions of type 'linux lvm' should not have a filesystem.
- Partitions of type 'unspecified' should not have a filesystem.
- If a raid is a member of another raid then it can not have a filesystem.
- Partitions, which are listed as raid members, should be of type 'linux raid'.
- If diskless is not set, there should be at least one device.
-->

<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' elementFormDefault='qualified'>

  <xs:element name='diskSetup'>

    <xs:complexType>
      <xs:sequence>
        <xs:element name='diskless' type='diskless' minOccurs='0' maxOccurs='1'/>
        <xs:element name='device' type='device' minOccurs='0' maxOccurs='unbounded'/>
        <xs:element name='raid' type='raid' minOccurs='0' maxOccurs='unbounded'/>
        <xs:element name='volumeGroup' type='volumeGroup' minOccurs='0' maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>

    <xs:key name='partitionAndRaidIds'>
      <xs:selector xpath='.//raid|.//partition'/>
      <xs:field xpath='@id'/>
    </xs:key>

    <xs:keyref name='raidMemberIds' refer='partitionAndRaidIds'>
      <xs:selector xpath='.//raid/member'/>
      <xs:field xpath='.'/>
    </xs:keyref>

    <xs:keyref name='volumeGroupPhysicalVolumes' refer='partitionAndRaidIds'>
      <xs:selector xpath='.//volumeGroup/physicalVolumes/member'/>
      <xs:field xpath='.'/>
    </xs:keyref>

    <xs:unique name='raidAndVolumeMembersUnique'>
      <xs:selector xpath='.//member'/>
      <xs:field xpath='.'/>
    </xs:unique>

    <xs:unique name='deviceNodesUnique'>
      <xs:selector xpath='.//device/blockdev'/>
      <xs:field xpath='.'/>
    </xs:unique>

    <xs:unique name='mountPointsUnique'>
      <xs:selector xpath='.//mountPoint'/>
      <xs:field xpath='.'/>
    </xs:unique>

    <xs:unique name='assertNamesUnique'>
      <xs:selector xpath='.//assert'/>
```

```
    <xs:field xpath='@name'/>
  </xs:unique>

</xs:element>

<xs:complexType name='diskless'>
  <xs:attribute name='maxMemSize' type='memSize' use='required'/>
</xs:complexType>

<xs:simpleType name='memSize'>
  <xs:restriction base='xs:string'>
    <xs:pattern value='([0-9]+[MG])|100%|[0-9][0-9]%|[0-9]%|0'/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name='size'>
  <xs:restriction base='xs:string'>
    <xs:pattern value='max|auto|[0-9]+[MGT]'/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name='extentSize'>
  <xs:restriction base='xs:string'>
    <xs:pattern value='([0-9])+M'/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name='device'>
  <xs:sequence>
    <xs:element name='blockdev' type='xs:string' minOccurs='1' maxOccurs='unbounded'/>
    <xs:element name='vendor' type='xs:string' minOccurs='0' maxOccurs='1'/>
    <xs:element name='requiredSize' type='size' minOccurs='0' maxOccurs='1'/>
    <xs:element name='assert' minOccurs='0' maxOccurs='unbounded'>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base='xs:string'>
            <xs:attribute name='name' use='required'>
              <xs:simpleType>
                <xs:restriction base='xs:string'>
                  <xs:pattern value='[a-zA-Z0-9-_]+'/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name='args' type='xs:string'/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name='partition' type='partition' minOccurs='1' maxOccurs='unbounded'/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name='partition'>
  <xs:sequence>
    <xs:element name='size' type='size'/>
```

```
    <xs:element name='type'>
      <xs:simpleType>
        <xs:restriction base='xs:string'>
          <xs:enumeration value='linux'/>
          <xs:enumeration value='linux swap'/>
          <xs:enumeration value='linux raid'/>
          <xs:enumeration value='linux lvm'/>
          <xs:enumeration value='unspecified'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:group ref='filesystem' minOccurs='0' maxOccurs='1'/>
  </xs:sequence>
  <xs:attribute name='id' type='xs:string' use='required'/>
</xs:complexType>

<xs:group name='filesystem'>
  <xs:sequence>
    <xs:element name='filesystem'>
      <xs:simpleType>
        <xs:restriction base='xs:string'>
          <xs:enumeration value='ext2'/>
          <xs:enumeration value='ext3'/>
          <xs:enumeration value='ext4'/>
          <xs:enumeration value='xfs'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name='mountPoint' type='xs:string'/>
    <xs:element name='mountOptions' type='xs:string' default='defaults'/>
  </xs:sequence>
</xs:group>

<xs:complexType name='raid'>
  <xs:sequence>
    <xs:element name='member' type='xs:string' minOccurs='2' maxOccurs='unbounded'/>
    <xs:element name='level' type='xs:int'/>
    <xs:choice minOccurs='0' maxOccurs='1'>
      <xs:group ref='filesystem'/>
      <xs:element name='swap'><xs:complexType /></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name='id' type='xs:string' use='required'/>
</xs:complexType>

<xs:complexType name='volumeGroup'>
  <xs:sequence>
    <xs:element name='name' type='xs:string'/>
    <xs:element name='extentSize' type='extentSize'/>
    <xs:element name='physicalVolumes'>
      <xs:complexType>
        <xs:sequence>
          <xs:element name='member' type='xs:string' minOccurs='1' maxOccurs='unbounded'/>
        </xs:sequence>
      </xs:complexType>
```

```
      </xs:element>
      <xs:element name='logicalVolumes'>
        <xs:complexType>
          <xs:sequence>
            <xs:element name='volume' type='logicalVolume' minOccurs='1' maxOccurs='unbounded'/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='logicalVolume'>
    <xs:sequence>
      <xs:element name='name' type='xs:string'/>
      <xs:element name='size' type='size'/>
      <xs:group ref='filesystem' minOccurs='0' maxOccurs='1'/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Examples Of Element Types In XML Schema

Name Of Type	Example Values
size	10G, 128M, 1T, auto, max
device	/dev/sda, /dev/hda, /dev/cciss/c0d0
partition	linux, linux raid, linux swap, unspecified
fileystem	ext2, ext3, ext4, xfs

D.2 Example: Default Node Partitioning

The following example shows the default layout used for regular nodes:

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/sda</blockdev>
    <blockdev>/dev/hda</blockdev>
    <blockdev>/dev/cciss/c0d0</blockdev>
    <partition id="a1">
      <size>20G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a2">
      <size>2G</size>
      <type>linux</type>
```

```
      <filesystem>ext3</filesystem>
      <mountPoint>/var</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
    <partition id="a3">
      <size>2G</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/tmp</mountPoint>
      <mountOptions>defaults,noatime,nodiratime,nosuid,nodev</mountOptions>
    </partition>
    <partition id="a4">
      <size>16G</size>
      <type>linux swap</type>
    </partition>
    <partition id="a5">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/local</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
  </device>
</diskSetup>
```

The example assumes a single disk. Because multiple `blockdev` tags are used, the node-installer first tries to use `/dev/sda`, then `/dev/hda`, and finally `/dev/cciss/c0d0`.

For each `partition`, a `size` is specified. Sizes can be specified using megabytes (e.g. `500M`), gigabytes (e.g. `50G`) or terabytes (e.g. `2T`). Alternatively, a `max` size will use all remaining space. For swap partitions, a size of `auto` sets a swap partition to twice the node memory size.

In the example, all file systems are specified as `ext3`. Valid alternatives are `ext2` and `xfs`.

The `mount` man page has more details on mount options. If the `mountOptions` tag is left empty, its value defaults to `defaults`.

D.3 Example: Software RAID

The following example shows a simple software RAID setup:

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">

  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>25G</size>
      <type>linux raid</type>
    </partition>
  </device>
```

```
  <device>
    <blockdev>/dev/sdb</blockdev>
    <partition id="b1">
      <size>25G</size>
      <type>linux raid</type>
    </partition>
  </device>

  <raid id="r1">
    <member>a1</member>
    <member>b1</member>
    <level>1</level>
    <filesystem>ext3</filesystem>
    <mountPoint>/</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </raid>

</diskSetup>
```

The `level` tag specifies the RAID level used. The following are supported:

- `0` (striping without parity)
- `1` (mirroring)
- `4` (striping with dedicated parity drive)
- `5` (striping with distributed parity)
- `6` (striping with distributed double parity)

The `member` tags must refer to an `id` attribute of a `partition` tag, or an `id` attribute of a another `raid` tag. The latter can be used to create, for example, RAID 10 configurations.

The administrator must ensure that the correct RAID kernel module is loaded (section 6.3.2). Including the appropriate module from the following is usually sufficient: `raid0`, `raid1`, `raid4`, `raid5`, `raid6`.

D.4 Example: Software RAID With Swap

The `<swap></swap>` tag is used to indicate a swap partition in a RAID device specified in the XML schema of section D.1. For example, the following marks a 1GB RAID 1 partition as being used for swap, and the second partition for an ext3 filesystem:

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>1G</size>
      <type>linux raid</type>
    </partition>
    <partition id="a2">
```

```
      <size>max</size>
      <type>linux raid</type>
    </partition>
  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <partition id="b1">
      <size>1G</size>
      <type>linux raid</type>
    </partition>
    <partition id="b2">
      <size>max</size>
      <type>linux raid</type>
    </partition>
  </device>
  <raid id="r1">
    <member>a1</member>
    <member>b1</member>
    <level>1</level>
    <swap></swap>
  </raid>
  <raid id="r2">
    <member>a2</member>
    <member>b2</member>
    <level>1</level>
    <filesystem>ext3</filesystem>
    <mountPoint>/</mountPoint>
    <mountOptions>defaults,noatime,nodiratime</mountOptions>
  </raid>
</diskSetup>
```

D.5 Example: Logical Volume Manager

This example shows a simple LVM setup:

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>25G</size>
      <type>linux lvm</type>
    </partition>
  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <partition id="b1">
      <size>25G</size>
      <type>linux lvm</type>
    </partition>
  </device>
  <volumeGroup>
```

```
    <name>vg1</name>
    <extentSize>4M</extentSize>
    <physicalVolumes>
      <member>a1</member>
      <member>b1</member>
    </physicalVolumes>
    <logicalVolumes>
      <volume>
        <name>vol1</name>
        <size>35G</size>
        <filesystem>ext3</filesystem>
        <mountPoint>/</mountPoint>
        <mountOptions>defaults,noatime,nodiratime</mountOptions>
      </volume>
      <volume>
        <name>vol2</name>
        <size>max</size>
        <filesystem>ext3</filesystem>
        <mountPoint>/tmp</mountPoint>
        <mountOptions>defaults,noatime,nodiratime</mountOptions>
      </volume>
    </logicalVolumes>
  </volumeGroup>
</diskSetup>
```

The `member` tags must refer to an `id` attribute of a `partition` tag, or an `id` attribute of a `raid` tag.

The administrator must ensure that the `dm-mod` kernel module is loaded when LVM is used.

D.6 Example: Diskless

This example shows a node configured for diskless operation:

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <diskless maxMemSize="0"></diskless>
</diskSetup>
```

An example of the implementation of a diskless configuration is given in section 4.6.3.

In diskless mode the software image is transferred by the node-installer to a RAM-based filesystem on the node called `tmpfs`.

The obvious advantage of running from RAM is the elimination of the physical disk, cutting power consumption and reducing the chance of hardware failure. On the other hand, some of the RAM on the node is then no longer available for user applications.

Special considerations with diskless mode:

- **Recommended minimum RAM size:** The available RAM per node should be sufficient to run the OS and the required tasks. At least 4GB is recommended for diskless nodes.

- **Default `tmpfs` size is unrestricted:** By default the amount of RAM that may be used for a file system is unrestricted. This means that creating very large files can exceed the RAM physically available for the file system, causing a node to run out of memory and crash.
- **Limiting `tmpfs` size:** The amount of RAM used for a file system can be set with the `maxMemSize` attribute. The default value of `0` allows all of the RAM to be used. A limit does not however necessarily prevent the node from crashing, as some processes might not deal properly with a situation when there is no more space left on the filesystem.
- **Persistence issues:** While running as a diskless node, the node is unable to retain any non-shared data each time it reboots. For example the files in `/var/log/*`, which are normally preserved by the exclude list settings for disked nodes, are lost from RAM during diskless mode reboots.
- **Leftover disk issues:** Administrators in charge of sensitive environments should be aware that the disk of a node that is now running in diskless mode still contains files from the last time the disk was used, unless the files are explicitly wiped.
- **Reducing the software image size in `tmpfs` on a diskless node:** To make more RAM available for tasks, the software image size held in RAM can be reduced:
  - by removing unnecessary software from the image.
  - by mounting parts of the filesystem in the image over NFS during normal use. This is especially worthwhile for less frequently accessed parts of the image (section 4.7.3).

D.7 Example: Semi-diskless

Diskless operation (section D.6) can also be mixed with certain parts of the file system on the local physical disk. This frees up RAM which the node can then put to other use. In this example all data in `/local` is on the physical disk, the rest in RAM.

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <diskless maxMemSize="0"></diskless>
  <device>
    <blockdev>/dev/sda</blockdev>
    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/local</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>
```

```
  </device>
</diskSetup>
```

When nodes operate in semi-diskless mode the node-installer always uses `excludelistfullinstall` (section 6.4.7) when synchronizing the software image to memory and disk.

An alternative to using a local disk for freeing up RAM is to use NFS storage, as is described in section 4.7.3.

D.8 Example: Preventing Accidental Data Loss

Optional tags, `vendor` and `requiredSize`, can be used to prevent accidentally repartitioning the wrong drive. Such a tag use is shown in the following example.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <device>
    <blockdev>/dev/sda</blockdev>
    <vendor>Hitachi</vendor>
    <requiredSize>200G</requiredSize>

    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <vendor>BigRaid</vendor>
    <requiredSize>2T</requiredSize>

    <partition id="b1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/data</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
</diskSetup>
```

If a `vendor` or a `requiredSize` element is specified, it is treated as an assertion which is checked by the node-installer. If any assertion fails, no partitioning changes will be made to any of the specified devices. The node-installer reads the drive vendor string from `/sys/block/<drive name>/device/vendor`.

Specifying device assertions is recommended for machines that contain important data because it protects against a situation where a drive is assigned to an incorrect block device. This can happen, for example, when the first drive in a multi-drive system is not detected (e.g. due to a hardware failure) which could cause the second drive to become known as `/dev/sda`.

D.9 Example: Using Custom Assertions

The following example shows the use of the `assert` tag, which can be added to a `device` definition:

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<diskSetup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="schema.xsd">
  <device>
    <blockdev>/dev/sda</blockdev>
    <assert name="modelCheck" args="WD800AAJS">
      <![CDATA[
        #!/bin/bash
        if grep -q $1 /sys/block/$ASSERT_DEV/device/model; then
          exit 0
        else
          exit 1
        fi
      ]]>
    </assert>

    <partition id="a1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
  <device>
    <blockdev>/dev/sdb</blockdev>
    <vendor>BigRaid</vendor>
    <requiredSize>2T</requiredSize>

    <partition id="b1">
      <size>max</size>
      <type>linux</type>
      <filesystem>ext3</filesystem>
      <mountPoint>/data</mountPoint>
      <mountOptions>defaults,noatime,nodiratime</mountOptions>
    </partition>

  </device>
</diskSetup>
```

The `assert` tag is similar to the `vendor` and `size` tags described in section D.8.

It can be used to define custom assertions. The assertions can be implemented using any script language.

The script will have access to the environment variables `ASSERT_DEV` (i.e. sda) and `ASSERT_NODE` (i.e. /dev/sda).

Each assert needs to be assigned an arbitrary name and can be passed custom parameters. A non-zero exit code in the assertion causes the node-installer to halt.

E

Example initialize And finalize Scripts

The node-installer executes any initialize and finalize scripts at particular stages of its 13-step run during node-provisioning (section 6.4). They are sometimes useful for troubleshooting or workarounds during those stages. The scripts are stored in the CMDaemon database, rather than in the filesystem as plain text files, because they run before the node's init process takes over and establishes the final filesystem.

E.1 When Are They Used?

An initialize script is used well before the init process starts, to execute custom commands before partitions and mounting devices are checked. Typically, initialize script commands are related to partitioning, mounting, or initializing special storage hardware. Often an initialize script is needed because the commands in it cannot be stored persistently anywhere else.

A finalize script (also run before init) is used to set a file configuration or to initialize special hardware, sometimes after a hardware check. It is run in order to make software or hardware work before, or during the later init stage of boot. Thus, often a finalize script is needed because its commands must be executed before init, and the commands cannot be stored persistently anywhere else, or it is needed because a choice between (otherwise non-persistent) configuration files must be made based on the hardware before init starts.

E.2 Accessing From cmgui And cmsh

The initialize and finalize scripts are accessible for viewing and editing:

- In cmgui, using the "Node Categories" or Nodes resource, under the Settings tabbed pane for the selected item.
- In cmsh, using the category or device modes. The get command is used for viewing the script, and the set command to start up the default text editor to edit the script. Output is truncated in the two following examples at the point where the editor starts up:

Example

```
[root@bright52 ~]# cmsh
[bright52]% category use default
[bright52->category[default]]% show | grep script
Parameter                      Value
------------------------------ ------------------------------------------------
Finalize script                <1367 bytes>
Initialize script              <0 bytes>
[bright52->category[default]]% set initializescript
```

Example

```
[bright52]% device use node001
[bright52->device[node001]]%
[bright52->device[node001]]% set finalizescript
```

E.3 Analogous Scripts That Run During imageupdate

The imageupdate_initialize and imageupdate_finalize scripts are similar scripts, but run, as their name implies, when the imageupdate command is run, and not during node-provisioning. They are discussed further in section 6.6.2.

E.4 Environment Variables Available To initialize And finalize Scripts

For the initialize and finalize scripts, node-specific customizations can be made from a script using environment variables. The following table shows the available variables with some example values:

Table E: Environment Variables For The initialize *And* Finalize *Scripts*

Variable	Example Value
CMD_ACTIVE_MASTER_IP	10.141.255.254
CMD_CATEGORY	default
CMD_CHASSIS	chassis01
CMD_CHASSIS_IP	10.141.1.1
CMD_CHASSIS_PASSWORD	ADMIN
CMD_CHASSIS_SLOT	1
CMD_CHASSIS_USERNAME	ADMIN
CMD_CLUSTERNAME	Bright 5.2 Cluster
CMD_DEVICE_HEIGHT	1
CMD_DEVICE_POSITION	10
CMD_DEVICE_TYPE	SlaveNode
CMD_ETHERNETSWITCH	switch01:1
CMD_FSEXPORT__SLASH_cm_SLASH\	no

...continues

Table E: Environment Variables For The initialize *And* Finalize *Scripts...continued*

Variable	Example Value
_node-installer_ALLOWWRITE	
CMD_FSEXPORT__SLASH_cm_SLASH\	10.141.0.0/16
_node-installer_HOSTS	
CMD_FSEXPORT__SLASH_cm_SLASH\	/cm/node-installer
_node-installer_PATH	
CMD_FSEXPORTS	_SLASH_cm_SLASH_node-installer
CMD_FSMOUNT__SLASH_cm_SLASH\	master:/cm/shared
_shared_DEVICE	
CMD_FSMOUNT__SLASH_cm_SLASH\	nfs
_shared_FILESYSTEM	
CMD_FSMOUNT__SLASH_cm_SLASH\	/cm/shared
_shared_MOUNTPOINT	
CMD_FSMOUNT__SLASH_cm_SLASH\	rsize=32768,wsize=32768,\
_shared_OPTIONS	hard,intr,async
CMD_FSMOUNT__SLASH_dev_SLASH\	none
_pts_DEVICE	
CMD_FSMOUNT__SLASH_dev_SLASH\	devpts
_pts_FILESYSTEM	
CMD_FSMOUNT__SLASH_dev_SLASH\	/dev/pts
_pts_MOUNTPOINT	
CMD_FSMOUNT__SLASH_dev_SLASH\	gid=5,mode=620
_pts_OPTIONS	
CMD_FSMOUNT__SLASH_dev_SLASH\	none
_shm_DEVICE	
CMD_FSMOUNT__SLASH_dev_SLASH\	tmpfs
_shm_FILESYSTEM	
CMD_FSMOUNT__SLASH_dev_SLASH\	/dev/shm
_shm_MOUNTPOINT	
CMD_FSMOUNT__SLASH_dev_SLASH\	defaults
_shm_OPTIONS	
CMD_FSMOUNT__SLASH\	master:/home
_home_DEVICE	
CMD_FSMOUNT__SLASH_home\	nfs
_FILESYSTEM	
CMD_FSMOUNT__SLASH_home\	home
_MOUNTPOINT	
CMD_FSMOUNT__SLASH_home\	rsize=32768,wsize=32768,\
_OPTIONS	hard,intr,async

...continues

Table E: Environment Variables For The `initialize` *And* `Finalize` *Scripts...continued*

Variable	Example Value
CMD_FSMOUNT__SLASH_proc\	none
_DEVICE	
CMD_FSMOUNT__SLASH_proc\	proc
_FILESYSTEM	
CMD_FSMOUNT__SLASH_proc\	/proc
_MOUNTPOINT	
CMD_FSMOUNT__SLASH_proc\	defaults,nosuid
_OPTIONS	
CMD_FSMOUNT__SLASH_sys\	none
_DEVICE	
CMD_FSMOUNT__SLASH_sys\	sysfs
_FILESYSTEM	
CMD_FSMOUNT__SLASH_sys\	/sys
_MOUNTPOINT	
CMD_FSMOUNT__SLASH_sys\	defaults
_OPTIONS	
CMD_FSMOUNTS *	_SLASH_dev_SLASH_pts _SLASH_proc _SLASH_sys _SLASH_dev_SLASH_shm _SLASH_cm_SLASH_shared _SLASH_home
CMD_GATEWAY	10.141.255.254
CMD_HOSTNAME	node001
CMD_INSTALLMODE	AUTO
CMD_INTERFACE_eth0_IP **	10.141.0.1
CMD_INTERFACE_eth0_MTU **	1500
CMD_INTERFACE_eth0_NETMASK **	255.255.0.0
CMD_INTERFACE_eth0_TYPE **	physical
CMD_INTERFACES *	eth0 eth1 eth2 ipmi0
CMD_IP	10.141.0.1
CMD_MAC	00:00:00:00:00:01
CMD_PARTITION	base
CMD_PASSIVE_MASTER_IP	10.141.255.253
CMD_PDUS	
CMD_POWER_CONTROL	custom
CMD_RACK	rack01
CMD_RACK_HEIGHT	42

...continues

Table E: Environment Variables For The `initialize` *And* `Finalize` *Scripts...continued*

Variable	Example Value
CMD_RACK_ROOM	serverroom
CMD_ROLES	sgeclient storage
CMD_SHARED_MASTER_IP	10.141.255.252
CMD_SOFTWAREIMAGE_PATH	/cm/images/default-image
CMD_SOFTWAREIMAGE	default-image
CMD_TAG	00000000a000
CMD_USERDEFINED1	var1
CMD_USERDEFINED2	var2

* The value for this variable is a string with spaces, not an array. Eg:
CMD_FSMOUNTS="_SLASH_dev_SLASH_pts _SLASH_proc _SLASH_sys ..."

** The name of this variable varies according to the interfaces available. So, `eth0` can be replaced by `eth1`, `eth2`, `ipmi0`, and so on.

E.5 Using Environment Variables Stored In Multiple Variables

Some data values, such as those related to interfaces (`CMD_INTERFACES_*`), mount points (`CMD_FSMOUNT__SLASH_*`) and exports (`CMD_FSEXPORT__SLASH_cm__SLASH_node-installer_*`) are stored in multiple variables. The code example below shows how they can be used:

Example

```
#!/bin/bash

for interface in $CMD_INTERFACES
do
  eval type=\$CMD_INTERFACE_${interface}_TYPE
  eval ip=\$CMD_INTERFACE_${interface}_IP
  eval mask=\$CMD_INTERFACE_${interface}_NETMASK

  echo "$interface type=$type"
  echo "$interface ip=$ip"
  echo "$interface netmask=$mask"
done
```

For remotely mounted devices, the name of the environment variables for mount entries have the following naming convention:

Description	Naming Convention
volume	CMD_FSMOUNT_<*x*>_DEVICE
mount point	CMD_FSMOUNT_<*x*>_MOUNTPOINT
filesystem type	CMD_FSMOUNT_<*x*>_FILESYSTEM
mount point options	CMD_FSMOUNT_<*x*>_OPTIONS

For the names, the entries <*x*> are substituted with the local mount point path, such as "`/cm/shared`", but with the "`/`" character replaced with the text "`_SLASH_`". So, for a local mount point path "`/cm/shared`", the name of the associated volume environment variable becomes `CMD_FSMOUNT__SLASH_cm_SLASH_shared_DEVICE`.

A similar naming convention is applicable to the names of the environmental variables for the export entries:

Description	Naming Convention
exported system writeable?	`CMD_FSEXPORT_<y>_ALLOWWRITE`
allowed hosts or networks	`CMD_FSEXPORT_<y>_HOSTS`
path on exporter	`CMD_FSMOUNT_<y>_PATH`

Here, the entry <*y*> is replaced by the file path to the exported filesystem on the exporting node. This is actually the same as the value of "`CMD_FSMOUNT_<y>_PATH`", but with the "`/`" character replaced with the text "`_SLASH_`".

The entries for the local mount values and the export values in the table in section E.4 are the default values for a newly installed cluster. If the administrator wishes to add more devices and mount entries, this is done by configuring `fsexports` on the head node, and `fsmounts` on the regular nodes, using `cmgui` or `cmsh` (section 4.7).

E.6 Storing A Configuration To A Filesystem

E.6.1 Ways Of Writing A Finalize Script To Configure Nodes

The `initialize` script (section 6.4.5) runs after the install-mode type and execution have been determined (section 6.4.4), but before unloading specific drivers and before partitions are checked and filesystems mounted (section 6.4.6). Data output can therefore be written by it to writeable parts of the special NFS drive from which the node-installer is running.

For a `finalize` script (section 6.4.11), which runs just before switching from using the ramdrive to using the local hard drive, the local hard drive is mounted under `/localdisk`. Data can therefore be written to the local hard drive if needed. For example, predetermined configuration files can be written from the NFS drive for a particular node, or they can be written from an image prepared earlier and now running on the node at this stage, overwriting a node installer configuration:

Example

```
#!/bin/bash
cp /etc/myapp.conf.overwrite /localdisk/etc/myapp.conf
```

This technique is used in a `finalize` script example in section 9.9.5, except that an append operation is used instead of a copy operation, to overcome a network issue by modifying a network configuration file slightly.

Detection within a `finalize` script is another useful technique. The `finalize` script example of section 9.9.5 does detection too, to decide if a configuration change is to be done on the node or not.

A useful variation of a `finalize` script with detection is a script selecting from a choice of possible configurations. A symlink is set to one of the possible configurations based on hardware detection or detection of an environment variable. The environment variable can be a node parameter or similar, from the table in section E.4. If it is necessary to overwrite different nodes with different configurations, then the previous `finalize` script example might become something like:

Example

```
#!/bin/bash
if [[ $CMD_HOSTNAME = node00[1-7] ]]
  then ln -s /etc/myapp.conf.first /localdisk/etc/myapp.conf
fi
if [[ $CMD_HOSTNAME = node01[5-8] ]]
  then ln -s /etc/myapp.conf.second /localdisk/etc/myapp.conf
fi
if [[ $CMD_HOSTNAME = node02[3-6] ]]
  then ln -s /etc/myapp.conf.third /localdisk/etc/myapp.conf
fi
```

In the preceding example, the configuration file in the image has several versions: `/etc/myapp.conf.<first|second|third>`. Nodes node001 to node007 are configured with the first version, nodes node015 to node018 with the second version, and nodes node023 to node026 with the third version. More versions are convenient to add to this kind of decision mechanism.

E.6.2 Restricting The Script To Nodes Or Node Categories

As mentioned in section 3.1.3, node settings can be adjusted within a category. So the configuration changes to `ifcfg-eth0` can be implemented per node by accessing and adjusting the `finalize` script per node if only a few nodes in the category are to be set up like this. If all the nodes in a category are to be set up like this, then the changes are best implemented in a `finalize` script accessed and adjusted at the category level. Accessing the scripts at the node and category levels is covered in section E.2.

F

Quickstart Installation Guide

This appendix describes a basic installation of Bright Cluster Manager on a cluster as a step-by-step process. Following these steps allows cluster administrators to get a cluster up and running as quickly as possible without having to read the entire administrators manual. References to chapters and sections are provided where appropriate.

F.1 Installing Head Node

1. Set the local time in the BIOS of the head node.
2. Boot head node from Bright Cluster Manager DVD.
3. Select `Install Bright Cluster Manager` in the boot menu
4. Once the installation environment has been started, choose Normal installation mode and click `Continue`.
5. Accept the License Agreement for Bright Cluster Manager and click `Continue`.
6. Accept the License Agreement for the Linux base distribution and click `Continue`.
7. Click `Continue` on kernel modules screen.
8. Review the detected hardware and go back to kernel modules screen if additional kernel modules are required. Once all relevant hardware (Ethernet interfaces, hard drive and DVD drive) is detected, click `Continue`.
9. Specify the number of racks and the number of regular nodes, set the base name for the regular nodes and the number of digits to append to the base name. Select the correct hardware manufacturer and click `Continue`.
10. Choose a network layout and click `Continue`. The first layout is the most commonly used. The rest of this appendix assumes the first layout was chosen.
11. Optionally add an InfiniBand network and configure the use of IPMI/iLO BMCs on the nodes. Adding an IPMI/iLO network is

needed to configure IPMI/iLO interfaces in a different IP subnet (recommended). When done, click `Continue`.

12. Fill in the following settings for the network named `externalnet`:

    - Base Address (a.k.a. *network address*)
    - Netmask
    - Domain name
    - Default gateway

    The network `externalnet` corresponds to the site network that the cluster resides in (e.g. corporate or campus network). Note that assigning the cluster an IP address in this network is handled in one of the next screens. All networks besides the `externalnet` network use private IP ranges by default and normally do not have to be changed. Click `Continue`.

13. Add and remove DNS search domains and external DNS name servers as required, and click `Continue`

14. Assign an IP address for the head node on `externalnet`. This is the IP address that is used to access the cluster over the network.

15. If necessary, modify the node properties. When IPMI/iLO interfaces reside in the same IP subnet, an IP Offset is set for the `ipmi0` interface. Click `Continue` to continue.

16. If an InfiniBand network was enabled, select which nodes (if any) are to run the subnet manager for the InfiniBand network. Click `Continue` to continue.

17. Select the DVD drive containing the Bright Cluster Manager DVD and click `Continue`.

18. Select a workload management system and set the number of slots per node equal to the number of CPU cores per node. Click `Continue` to continue.

19. Optionally you may modify the disk layout for the head node by selecting a pre-defined layout. The layout may be fine-tuned by editing the XML partitioning definition. Click `Continue` to continue and confirm that the data on the listed drive(s) may be erased by clicking `Yes`.

20. Select a time-zone and optionally add NTP time-servers. Click `Continue` to continue.

21. By default there are no network-specific restrictions on access to the cluster (e.g. using `ssh` or `cmgui`). To accept the defaults, click `Continue`.

22. Enter a hostname for the head node. Enter a password that is to be used for system administration twice and click `Continue`.

23. Configure text or graphical consoles for the nodes in the cluster. Note that Cluster Management GUI can still be used remotely if the console of the head node is set to text mode.

24. Review the network summary screen, click the `Start` button to start the installation and click `Yes` to confirm that the data on the listed drive(s) may be erased.

25. Wait until installation has completed, click `Reboot` and click `Yes` to confirm that the head node may be rebooted.

F.2 First Boot

1. Ensure that the head node boots from the first hard drive by removing the DVD or altering the boot-order in the BIOS configuration.

2. Once the machine is fully booted, log in as `root` with the password that was entered during installation.

3. Confirm that the machine is visible on the external network. Ensure that the second NIC (i.e. `eth1`) is physically connected to the external network.

4. Verify that the license parameters are correct:
   `cmsh -c "main licenseinfo"`

   If the license being used is a temporary license (see `End Time` value), a new license should be requested well before the temporary license expires. The procedure for requesting and installing a new license is described in section 4.1.

F.3 Booting Nodes

1. Make sure the first NIC (i.e. `eth0`) on the head node is physically connected to the internal cluster network.

2. Configure the BIOS of nodes to boot from the network, and boot the nodes.

3. If everything goes well, the node-installer component starts on each node and a certificate request is sent to the head node.

   If a node does not make it to the node-installer, it is possible that additional kernel modules are needed. Section 6.8 contains more information on how to diagnose problems during the node booting process.

4. To identify the nodes (i.e. to assign a host name to each physical node), several options are available. Which option is most convenient depends mostly on the number of nodes and whether a (configured) managed Ethernet switch is present.

   Rather than identifying nodes based on their MAC address, it is often beneficial (especially in larger clusters) to identify nodes based on the Ethernet switch port that they are connected to. To allow nodes to be identified based on Ethernet switch ports, section 4.5 should be consulted.

   Any one of the following methods may be used to assign node identities when all nodes have been booted:

a. **Identify each node on the node console**: To manually identify each node, the "`Manually select node`" option is selected for each node. The node is then identified manually by selecting a node-entry from the list, chosing the `Accept` option. This option is easiest when there are not many nodes. It requires being able to view the console of each node and keyboard entry to the console.

b. **Identify nodes using** `cmgui`: The Node Identification Wizard (section 6.4.2) in `cmgui` automates the process of assigning identities so that nodes do not require manual identification on the console.

c. **Identify nodes using** `cmsh`: In `cmsh` the `newnodes` command in `device` mode (section 6.4.2) can be used to assign identities to nodes from the command line. When called without parameters, the `newnodes` command can be used to verify that all nodes have booted into the node installer and are all waiting to be assigned an identity.

Example

To verify that all nodes have booted into the node installer:

```
[root@mycluster ~]# cmsh
[mycluster]% device newnodes
MAC                   First appeared                  Detected on switch port
--------------------- ------------------------------- ---------------------------
00:0C:29:D2:68:8D     Mon, 05 Sep 2011 13:43:13 CEST  [no port detected]
00:0C:29:54:F5:94     Mon, 05 Sep 2011 13:49:41 CEST  [no port detected]
..
[mycluster]% device newnodes | wc -l
MAC                   First appeared                  Detected on switch port
--------------------- ------------------------------- ---------------------------
32
[mycluster]% exit
[root@mycluster ~]#
```

Example

Once all nodes have been booted in the proper order, we can use the order of their appearance on the network to assign node identities. To assign identities `node001` through `node032` to the first 32 nodes that were booted, the following commands may be used:

```
[root@mycluster ~]# cmsh
[mycluster]% device newnodes -s -n node001..node032
MAC                   First appeared                  Hostname
--------------------- ------------------------------- ----------------------
00:0C:29:D2:68:8D     Mon, 05 Sep 2011 13:43:13 CEST  node001
00:0C:29:54:F5:94     Mon, 05 Sep 2011 13:49:41 CEST  node002
..
[mycluster]% exit
[root@mycluster ~]#
```

5. Each node is now provisioned and eventually fully boots. In case of problems, section 6.8 should be consulted.

6. *Optional:* To configure power management, Chapter 5 should be consulted.

F.4 Running Cluster Management GUI

To run the Cluster Management GUI on the cluster from a workstation running X11:

1. From a Linux desktop PC, log in to the cluster with SSH X-forwarding:
   `ssh -X root@mycluster`
2. Start the Cluster Management GUI:
   `cmgui`
3. Click on the connect button (see figure 3.3 and enter the password that was configured during installation.)
4. *Optional:* For more information on how the Cluster Management GUI can be used to manage one or more clusters, consult section 3.4.

To run the Cluster Management GUI on a desktop PC:

1. Copy the appropriate package(s) from `/cm/shared/apps/cmgui/dist` to the desktop PC:

   ```
   scp root@mycluster:/cm/shared/apps/cmgui/dist/* /tmp
   ```

   Note: On windows use e.g. WinSCP.
2. Copy the PFX certificate file from the cluster to the desktop so that it can be used for authentication purposes:

   ```
   scp root@mycluster:admin.pfx ~/mycluster-admin.pfx
   ```

3. Install the package.
   On Windows: execute the installer and follow the steps.
   On Linux: extract using `tar -xvjf` *filename*
4. Start the cluster management GUI.
   On Windows: from the Start menu or by clicking the desktop icon.
   On Linux: change into the `cmgui` directory and execute:
   `./cmgui`
5. Click on `Add a new cluster` and enter the following parameters:
   Host: Hostname or IP address of the cluster
   Certificate: Click `Browse` and browse to the certificate file.
   Password: Password entered during installation
6. Click on the connect button (see figure 3.3)
7. *Optional:* For more information on how the Cluster Management GUI can be used to manage one or more clusters, consult section 3.4.

Your cluster should now be ready for running compute jobs. For more information on managing the cluster, please consult the appropriate chapters in this manual.

Please consult the *User Manual* provided in:

```
/cm/shared/docs/cm/user-manual.pdf
```

for more information on the user environment and how to start jobs through the workload management system.

G

Workload Managers Quick Reference

G.1 SLURM

SLURM (Simple Linux Utility for Resource Management) is a GPL-licensed workload management system and developed largely at Lawrence Livermore National Laboratory.

The SLURM service and outputs are normally handled using the `cmgui` or `cmsh` front-end tools for CMDaemon (section 8.4).

From the command line, direct SLURM commands that may sometimes come in useful include the following:

- `sacct`: used to report job or job step accounting information about active or completed jobs.
- `salloc`: used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.
- `sattach` used to attach standard input, output, and error plus signal capabilities to a currently running job or job step. One can attach to and detach from jobs multiple times.
- `sbatch`: used to submit a job script for later execution. The script typically contains one or more srun commands to launch parallel tasks.
- `sbcast`: used to transfer a file from local disk to local disk on the nodes allocated to a job. This can be used to effectively use diskless compute nodes or provide improved performance relative to a shared file system.
- `scancel`: used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.
- `scontrol`: the administrative tool used to view and/or modify SLURM state. Note that many scontrol commands can only be executed as user root.

- `sinfo`: reports the state of partitions and nodes managed by SLURM. It has a wide variety of filtering, sorting, and formatting options.
- `smap`: reports state information for jobs, partitions, and nodes managed by SLURM, but graphically displays the information to reflect network topology.
- `squeue`: reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.
- `srun`: used to submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared nodes within the job's node allocation.
- `smap`: reports state information for jobs, partitions, and nodes managed by SLURM, but graphically displays the information to reflect network topology.
- `strigger`: used to set, get or view event triggers. Event triggers include things such as nodes going down or jobs approaching their time limit.
- `sview`: a graphical user interface to get and update state information for jobs, partitions, and nodes managed by SLURM.

Full documentation on SLURM is available online at: `https://computing.llnl.gov/linux/slurm/documentation.html`.

G.2 Sun Grid Engine

Sun Grid Engine (SGE) is a workload management system that was originally made available under an Open Source license by Sun Microsystems. It forked off into various versions in 2010 and its future is unclear at the time of writing, but it remains in widespread use. Bright Cluster Manager 5.2 uses version 6.2 update 5 patch 2 of Grid Scheduler, which is a bugfix-patched version of the last SGE release from Sun Microsystems, and is made available on sourceforge at `http://gridscheduler.sourceforge.net/`.

SGE services should be handled using CMDaemon, as explained in section 8.4. However SGE can break in obtuse ways when implementing changes, so the following notes are sometimes useful in getting a system going again:

- The `sge_qmaster` daemon on the head node can be started or stopped using `/etc/init.d/sgemaster.sqe1 start|stop`, or alternatively via `qconf -{s|k}m`.

- The sge_execd execution daemon running on each compute node accepts, manages, and returns the results of the jobs on the compute nodes. The daemon can be started or stopped via /etc/init.d/sgeexecd start|stop, or alternatively deregistered from qmaster via qconf -{s|k}s.
- Queues in an error state are cleared with a qmod -c <queue name>.

SGE can be configured and managed generally with the command line utility qconf, which is what most administrators become familiar with. A GUI alternative, qmon, is also provided.

SGE commands are listed below. The details of these are in the man page of the command and the SGE documentation.

- qalter: modify existing batch jobs
- qacct: show usage information from accounting data
- qconf: configure SGE
- qdel: delete batch jobs
- qhold: place hold on batch jobs
- qhost: display compute node queues, states, jobs
- qlogin: start login-based interactive session with a node
- qmake: distributed, parallel make utility
- qmod: suspend/enable queues and jobs
- qmon: configure SGE with an X11 GUI interface
- qping: check sge_qmaster and sge_execd status
- qquota: list resource quotas
- qresub: create new jobs by copying existing jobs
- qrdel: cancel advance reservations
- qrls: release batch jobs from a held state
- qrsh: start rsh-based interactive session with node
- qrstat: show status of advance reservations
- qrsub: submit advanced reservation
- qselect: select queues based on argument values
- qsh: start sh interactive session with a node
- qstat: show status of batch jobs and queues
- qsub: submit new jobs (related: qalter, qresub)
- qtcsh: start csh-based interactive session with a node

G.3 Torque

The following commands are used to manage Torque:

Torque resource manager commands:

- `qalter`: alter batch job
- `qdel`: delete batch job
- `qhold`: hold batch jobs
- `qrls`: release hold on batch jobs
- `qstat`: show status of batch jobs
- `qsub`: submit job
- `qmgr`: batch policies and configurations manager
- `qenable`: enable input to a destination
- `qdisable`: disable input to a destination
- `tracejob`: trace job actions and states

Further information on these and other commands is available in the appropriate man pages and on-line documentation at `http://www.adaptivecomputing.com/resources/docs/`.

The Torque administrator manual is online at `http://www.adaptivecomputing.com/resources/docs/torque/index.php`.

G.4 PBS Pro

The following commands can be used in PBS Pro to view queues:

```
qstat             query queue status
qstat -a          alternate form
qstat -r          show only running jobs
qstat -q          show available queues
qstat -rn         only running jobs, w/ list of allocated nodes
qstat -i          only idle jobs
qstat -u username show jobs for named user
```

Other useful commands are:

```
tracejob id       show what happened today to job id
tracejob -n d id  search last d days

qmgr              administrator interface to batch system

qterm             terminates queues (but cm starts pbs_server again)

pbsnodes -a       list available worker nodes in queue
```

The commands of PBS Pro are documented in the *PBS Professional 11.0 Reference Guide*. There is further extensive documentation for PBS Pro administrators in the *PBS Professional 11.0 Administrator's Guide*. Both are available at the PBS Works website at `http://www.pbsworks.com/SupportDocuments.aspx`.

H

Metrics, Health Checks, And Actions

This appendix describes the metrics, health checks, and actions in a newly-installed cluster. Metrics, health checks, and actions can each be standalone scripts, or built-ins. Standalone scripts can be those supplied with the system, or they can be custom scripts built by the administrator. Scripts can use environmental values (the `CMD_*` variables of Appendix E.4).

H.1 Metrics And Their Parameters

H.1.1 Metrics

Table H.1.1: List Of Metrics

Name	Description
`AlertLevel`	Indicates the healthiness of a device, the lower the better
`Average`	Cluster-wide average of metric of choice
`AvgExpFactor`	Average Expansion Factor. This is by what factor, on average, jobs took longer to run than expected. The expectation is according to heuristics based on duration in past and current job queues, as well as node availability.
`AvgJobDuration`	Average Job Duration of current jobs
`BufferMemory`	System memory used for buffering
`BytesRecv`	Number of bytes received
`BytesSent`	Number of bytes sent
`CMDActiveSessions`	Managed active sessions count
`CMDCycleTime`	Time used by master to process picked up data

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
CMDMemUsed	Resident memory used by CMDaemon
CMDState	State in which CMDaemon is running (head: 0, node: 1, failover:2)
CMDStoreQueryTime	Time used by master to store monitoring data to database
CMDSystime	Time spent by CMDaemon in system mode
CMDUsertime	Time spent by CMDaemon in user mode
CPUCoresAvailable	Cluster-wide number of CPU cores
CPUIdle	Total core usage in idle tasks per second
CPUIrq	Total core usage in servicing interrupts per second
CPUNice	Total core usage in nice'd user mode per second
CPUSoftIrq	Total core usage in servicing soft interrupts per second
CPUSystem	Total core usage in system mode per second
CPUUser	Total core usage in user mode per second
CPUWait	Total core usage in waiting for I/O to complete per second
CacheMemory	System memory used for caching
CompletedJobs	Jobs completed
CtxtSwitches	Number of context switches per second
DevicesUp	Number of devices in status UP
DropRecv	Number of received packets which are dropped
DropSent	Number of packets sent which are dropped
ErrorsRecv	Number of received packets with error
ErrorsSent	Number of packets sent which have error
EstimatedDelay	Estimated Delay to execute jobs
FailedJobs	Failed jobs
Forks	Number of forks since boot per second
FrameErrors	Number of packet framing errors
FreeSpace	Free space for non-root. Takes mount point as a parameter
GPUAvailable	Cluster-wide number of GPUs
IOInProgress	Number of I/O operations currently in progress
IOTime	Number of milliseconds spent doing I/O
LoadFifteen	Load average on 15 minutes

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
LoadFive	Load average on 5 minutes
LoadOne	Load average on 1 minute
MajorPageFaults	Page faults that require I/O
Max	Cluster-wide maximum of metric of choice
MemoryFree	Free system memory
MemoryUsed	Used system memory
MergedReads	Total number of merged reads
MergedWrites	Total number of merged writes
Min	Cluster-wide minimum of metric of choice
NetworkBytesRecv	Cluster-wide number of bytes received on all networks
NetworkBytesSent	Cluster-wide number of bytes transmitted on all networks
NetworkUtilization	Network utilization estimation(%)
NodesUp	Number of nodes in status UP
OccupationRate	Cluster occupation rate—the load on the cluster as a whole. 100% means all cores on all nodes are fully loaded.
PDUBankLoad	Total PDU bank load
PDULoad	Total PDU phase load
PDUUptime	PDU uptime
PacketsRecv	Number of received packets
PacketsSent	Number of packets sent
PageFaults	Page faults
PhaseLoad	Cluster-wide phase load
ProcessCount	Total number of processes
QueuedJobs	Number of queued jobs
RackSensorHumidity	Rack sensor humidity
RackSensorTemp	Rack sensor Temperature
Range	Cluster-wide range of metric of choice (difference between min and max)
ReadTime	Total number of milliseconds spent by all reads
Reads	Total number of reads completed successfully
RunningJobs	Number of running jobs
RunningProcesses	Number of processes in runnable state
SMARTHDATemp	Temperature of a Hard Disk Assembly
SMARTReallocSecCnt	Number of remapped sectors
SMARTSeekErrRate	Frequency of errors appearance while positioning the head

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
SMARTSeekTimePerf	Average efficiency of operations whilst positioning the head
SMARTSoftReadErrRate	Frequency of program errors while reading data
SectorsRead	Total number of sectors read successfully
SectorsWritten	Total number of sectors written successfully
SensorFanSpeed	System or CPU fan speed sensor
SensorTemp	Temperature sensor(system and CPU)
SensorVoltage	Motherboard voltage sensor
Sum	Cluster-wide sum of metric of choice
SwapFree	Free swap space
SwapUsed	Used swap space
SwitchBroadcastPackets	Total number of good packets received and directed to the broadcast address
SwitchCPUUsage	Switch CPU utilization estimation(%)
SwitchCollisions	Total number of collisions on this network segment
SwitchDelayDiscardFrames	Number of frames discarded due to excessive transit delay through the bridge
SwitchFilterDiscardFrames	Number of valid frames received but discarded by the forwarding process
SwitchMTUDiscardFrames	Number of frames discarded due to an excessive size
SwitchMulticastPackets	Total number of good packets received and directed to a multicast address
SwitchOverSizedPackets	Total number of well-received packets longer than 1518 octets
SwitchUnderSizedPackets	Total number of packets received which are less than 64 octets long
SwitchUptime	Switch uptime
TotalCPUIdle	Cluster-wide core usage in idle tasks
TotalCPUSystem	Cluster-wide core usage in system mode
TotalCPUUser	Cluster-wide core usage in user mode
TotalMemoryUsed	Cluster-wide total memory used
TotalNodes	Total number of nodes
TotalSwapUsed	Cluster-wide total swap used
Uptime	System uptime
UsedSpace	Total used space by a mount point
WriteTime	Total number of milliseconds spent by all writes

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
Writes	Total number of writes completed successfully
gpu*	GPU measurements (file: sample_gpu)
ilo*	ILO measurements (file: sample_ilo)
ipForwDatagrams	Number of input datagrams to be forwarded
ipFragCreates	The number of IP datagram fragments generated
ipFragFails	Number of IP datagrams which needed to be fragmented but could not
ipFragOKs	Number of IP datagrams successfully fragmented
ipInAddrErrors	Number of input datagrams discarded because the IP address in their header was not a valid address
ipInDelivers	Total number of input datagrams successfully delivered
ipInDiscards	Number of input IP datagrams discarded
ipInHdrErrors	Number of input datagrams discarded due to errors in their IP headers
ipInReceives	Total number of input datagrams, including ones with errors, received from all interfaces
ipInUnknownProtos	Number of received datagrams but discarded because of an unknown or unsupported protocol
ipOutDiscards	Number of output IP datagrams discarded
ipOutNoRoutes	Number of IP datagrams discarded because no route could be found
ipOutRequests	Total number of IP datagrams supplied to IP in requests for transmission
ipReasmOKs	Number of IP datagrams successfully reassembled
ipReasmReqds	Number of IP fragments received needing reassembly
ipmi*	IPMI measurements (file: sample_ipmi)
responsiveness*	The average time (in ms) for I/O requests to be served from device (file: sample_responsiveness)
sdt*	SuperMicro hardware measurements (file: sample_sdt)
tcpCurrEstab	Number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
tcpInErrs	Total number of IP segments received in error
tcpRetransSegs	Total number of IP segments retransmitted
testcollection*	An example of a metric collection script (file: testmetriccollection)
testmetric*	An example of a metric script (file: testmetric)
udpInDatagrams	Total number of UDP datagrams delivered to UDP users
udpInErrors	Number of received UDP datagrams that could not be delivered for other reasons (no port excl.)
udpNoPorts	Total number of received UDP datagrams for which there was no application at the destination port

* standalone scripts, not built-ins. Located in directory:

/cm/local/apps/cmd/scripts/metrics/

H.1.2 Parameters For Metrics

Metrics have the parameters indicated by the left column in the following example:

Example

```
[myheadnode->monitoring->metrics]% show cpuuser
Parameter                      Value
------------------------------ ------------------------------------------------
Class of metric                cpu
Command                        <built-in>
Cumulative                     yes
Description                    Total core usage in user mode per second
Disabled                       no
Extended environment           no
Measurement Unit
Name                           CPUUser
Only when idle                 no
Parameter permissions          disallowed
Retrieval method               cmdaemon
Sampling method                samplingonnode
State flapping count           7
Timeout                        5
Valid for                      node,master
maximum                        <range not set>
minimum                        <range not set>
[myheadnode->monitoring->metrics]%
```

The meanings of the parameters are:

`Class of metric`: A choice assigned to a metric depending on its type. The choices and what they are related to are listed below:

- Misc (default): miscellaneous class of metrics, used if none of the other classes are appropriate, or if none of the other classes are chosen
- CPU: CPU activity
- GPU: GPU activity
- Disk: Disk activity
- Memory: Memory activity
- Network: Network activity
- Environmental: sensor measurements of the physical environment
- Operating System: operating system activity
- Internal: bright cluster manager utilities
- Workload: workload management
- Cluster: clusterwide measurements
- Prototype: metric collections class

Command: For a standalone metric script, it is the full path. For a built-in, the value cannot be set, and the command is simply the name of the metric.

Cumulative: If set to yes, then the value is cumulative (for example, the bytes-received counter for an Ethernet interface). If set to no (default), then the value is not cumulative (for example, temperature).

Description: Description of the metric. Empty by default.

Disabled: If set to no (default) then the script runs.

Extended environment: If set to yes, more information about the device is made part of the environment to the script. The default is no.

Measurement Unit: A unit for the metric. A percent is indicated with %.

Name: The name given to the metric.

Only when idle: If set to yes, the metric script runs only when the system is idling. Useful if the metric is resource hungry, in order to burden the system less. It is set to no by default.

Parameter permissions: Decides if parameters passed to the metric script can be used. The three possible values are:

- disallowed: parameters are not used
- required: parameters are mandatory
- optional (default): parameters are optional

Retrieval method:

- cmdaemon (default): Metrics retrieved internally using CMDaemon
- snmp: Metrics retrieved internally using SNMP

Sampling method:

- samplingonmaster: The head node samples the metric on behalf of a device. For example, the head node may do this for a PDU because the PDU does not have the capability to run the cluster management daemon at present, and so cannot itself pass on data values directly when cmsh or cmgui need them.
- samplingonnode (default): The non-head node samples the metric itself.

State flapping count: How many times the metric value must cross a threshold within the last 12 samples before it is decided that it is in a flapping state. Default value is 7.

Timeout: After how many seconds the command will give up retrying. Default value is 5 seconds.

Valid for: Which device category the metric can be used with. The choices being:

- Node (Default)
- Master Node (Also a default)
- Power Distribution Unit
- Myrinet Switch
- Ethernet Switch
- IB Switch
- Rack Switch
- Generic Switch
- Chassis
- GPU Unit

Maximum: the default minimum value the y-axis maximum will take in graphs plotted in cmgui.[1]

Minimum: the default maximum value the y-axis minimum will take in graphs plotted in cmgui.[1]

[1] To clarify the concept, if maximum=3, minimum=0, then a data-point with a y-value of 2 is plotted on a graph with the y-axis spanning from 0 to 3. However, if the data-point has a y-value of 4 instead, then it means the default y-axis maximum of 3 is resized to 4, and the y-axis will now span from 0 to 4.

H.2 Health Checks And Their Parameters

H.2.1 Health Checks

Table H.2.1: List Of Health Checks

Name	**Query** (response is PASS/FAIL/UNKNOWN)
`DeviceIsUp`*	Is the device up, closed or installing?
`ManagedServicesOk`*	Are CMDaemon-monitored services all OK?
`chrootprocess`	Are there daemon processes running using chroot in software images? (here: yes = fail). On failure, kill cron daemon processes running in the software images.
`cmsh`	Is `cmsh` available?
`diskspace`	Is there less disk space available to non-root users than any of the space parameters specified? *The space parameters can be specified as MB, GB, TB, or as percentages with %. The default severity of notices from this check is 10, when one space parameter is used. For more than one space parameter, the severity decreases by 10 for each space parameter, sequentially, down to 10 for the last space parameter. There must be at least one space parameter. An optional non-space parameter, the filesystem mount point parameter, can be specified after the last space parameter to track filesystem space, instead of disk space. A metric-based alternative to tracking filesystem space changes is to use the built-in metric* `freespace` *(page 410) instead.* Examples: • `diskspace 10%` less than 10% space = FAIL, severity 10 • `diskspace 10% 20% 30%` less than 30% space = FAIL, with severity levels as indicated: **space left** / **Severity** `10%` / 30 `20%` / 20 `30%` / 10

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is PASS/FAIL/UNKNOWN)
	• `diskspace 10GB 20GB` less than 20GB space = FAIL, severity 10 less than 10GB space = FAIL, severity 20 • `diskspace 10% 20% /var` For the file system `/var`: less than 20% space = FAIL, severity 10 less than 10% space = FAIL, severity 20
`exports`	Are all filesystems as defined by the cluster management system exported?
`failedprejob`	Are there failed prejob health checks? (here: yes = FAIL)
`failover`	Is all well with the failover system?
`interfaces`	Are the interfaces all up and OK?
`ldap`	Can the ID of the user be looked up with LDAP?
`mounts`	Are all mounts defined in the fstab OK?
`mysql`	Is the status and configuration of mysql correct?
`node-hardware-profile`	Is the specified node's hardware configuration during health check use unchanged? *The options to this script are described using the "-h" help option. Before this script is used for health checks, the specified hardware profile is usually first saved with the* `-s` *option. Eg: "*`node-hardware-profile -n node001 -s hardwarenode001`*"*
`ntp`	Is NTP synchronization happening?
`portchecker`	Is the specified port on the specified host open for TCP (default) or UDP connections?
`rogueprocess`	Are the processes that are running legitimate (ie, not 'rogue')? Illegitimate processes are processes that should not be running on the node. An illegitimate process is, by default: • not part of the workload manager service or its jobs • not a root- or system-owned process • in the state Z, T, W, or X. States are described in the `ps` man pages in the section on "PROCESS STATE CODES" Rogue process criteria can be configured in the file `rogueprocess.py`.

...continued

Table H.2.1: List Of Health Checks...continued

Name	**Query** (response is PASS/FAIL/UNKNOWN)
schedulers	Are the queue instances of all schedulers on a node healthy ?
smart	Is the SMART response healthy?
ssh2node	Is passwordless ssh login from head to node working?
swraid	Are the software RAID arrays healthy?
testhealthcheck	*A health check script example for creating scripts, or setting a mix of PASS/FAIL/UNKNOWN responses. The source includes examples of environment variables that can be used, as well as configuration suggestions.*

* built-ins, not standalone scripts. Standalone scripts are located in /cm/local/apps/cmd/scripts/healthchecks/

H.2.2 Parameters For Health Checks

Health checks have the parameters indicated by the left column in the example below:

Example

```
[myheadnode->monitoring->healthchecks]% show cmsh
Parameter                      Value
------------------------------ ------------------------------------------------
Class of healthcheck           internal
Command                        /cm/local/apps/cmd/scripts/healthchecks/cmsh
Description                    Checks whether a the cmsh is available, i.e. we+
Disabled                       no
Extended environment           no
Name                           cmsh
Only when idle                 no
Parameter permissions          optional
Sampling method                samplingonnode
State flapping count           7
Timeout                        10
Valid for                      node,master,pdu,ethernet,myrinet,ib,racksensor+
```

The parameters have the same meaning as for metrics, with the following exceptions due to inapplicability:

Parameter	Reason For Inapplicability
class: prototype	only applies to metric collections
cumulative	only sensible for numeric values
measurementunit	only applies to numeric values
retrievalmethod	all health checks use CMDaemon internally for retrieval
maximum	only applies to numeric values
minimum	only applies to numeric values

The remaining parameters have meanings that can be looked up in section H.1.2.

H.3 Actions And Their Parameters

H.3.1 Actions

Table H.3.1: List Of Actions

Name	Description
`Drain node`	Allows no new processes on a compute node from the workload manager (Usage Tip: Plan for undrain from another node becoming active)
`killprocess`*	Kills a process with KILL (-9) signal
`Power off`	Powers off, hard
`Power on`	Powers on, hard
`Power reset`	Power reset, hard
`Reboot`	Reboot via the system, trying to shut everything down cleanly, and then start up again
`SendEmail`	Sends mail using the mailserver that was set up during server configuration. Format: sendemail [somebody@example.com]. Default destination is root@localhost
`Shutdown`	Power off via system, trying to shut everything down cleanly
`test action`*	An action script example for users who would like to create their own scripts. The source has helpful remarks about the environment variables that can be used as well as tips on configuring it generally
`Undrain node`	Allow processes to run on the node from the workload manager

* standalone scripts, not built-ins. Located in directory:

/cm/local/apps/cmd/scripts/actions/

H.3.2 Parameters For Actions

Actions have the parameters indicated by the left column in the example below:

Example

```
[myheadnode->monitoring->actions]% show drainnode
Parameter                      Value
------------------------------ ------------------------------------------------
Command                        <built-in>
Description                    Remove a node from further use by the scheduler+
Name                           Drain node
Run on                         master
Timeout                        5
isCustom                       no
```

The meanings of these parameters are:

`Command`: For a standalone metric script, it is the full path. For a built-in, the value cannot be set, and the command is simply the name of the metric.

`Description`: Description of the metric. Empty by default.

`Name`: The name given to the metric.

`Run on`: The node it will run on. For standalone actions it is usually a choice of head node, or the non-head node. For non-head nodes the action will run from the node that triggered it, if the node has sufficient permission to do that.

`Timeout`: After how many seconds the command will give up retrying. Default value is 5 seconds.

`isCustom`: Is this a standalone script?

I

Metric Collections

This appendix gives details on metric collections.

In section 10.4.4, metric collections are introduced, and how to add a metric collections script with `cmgui` is described.

This appendix covers how to add a metric collections script with `cmsh`. It also describes the output specification of a metric collections script, along with example outputs, so that a metric collections script can be made by the administrator.

I.1 Metric Collections Added Using `cmsh`

A metric collections script, `responsiveness`, is added in the `monitoring metrics` mode just like any other metric.

Example

```
[bright52]% monitoring metrics
[bright52->monitoring->metrics]% add responsiveness
[...[responsiveness]]% set command /cm/local/apps/cmd/scripts/metrics/s\
ample_responsiveness
[...*[responsiveness*]]% set classofmetric prototype; commit
```

For `classofmetric`, the value `prototype` is the class used to distinguish metric collections from normal metrics.

I.2 Metric Collections Initialization

When a metric collections script is added to the framework for the first time, it is implicitly run with the `--initialize` flag, which detects and adds component metrics to the framework.

The displayed output of a metric collections script when using the `--initialize` flag is a list of available metrics and their parameter values. The format of each line in the list is:

```
metric <name> [<unit> [<class> ["<description>" [<cumulative> [<min> <max>]]]]]
```

where the parameters are:

metric: A bare word.

name: The name of the metric.

unit: A measurement unit.

class: Any of: `misc cpu disk memory network environmental operatingsystem internal workload cluster`.

description: This can contain spaces, but should be enclosed with quotes.

cumulative: Either yes or no (default is no). This indicates whether the metric increases monotonically (e.g., bytes received) or not (e.g., temperature).

min and **max**: The minimum and maximum numeric values of this metric which still make sense.

Example

```
[root@myheadnode metrics]# ./sample_responsiveness --initialize
metric util_sda % internal "Percentage of CPU time during which I/O
requests were issued to device sda" no 0 100
metric await_sda ms internal "The average time (in milliseconds) for
I/O requests issued to device sda to be served" no 0 500
```

I.3 Metric Collections Output During Regular Use

The output of a metric collection script without a flag is a list of outputs from the available metrics. The format of each line in the list is:

```
metric <name> <value>
```

where the parameters are:

metric: A bare word.

name: The name of the metric.

value: The numeric value of the measurement.

Example

```
[root@myheadnode metrics]# ./sample_responsiveness
metric await_sda 0.00
metric util_sda 0.00
[root@myheadnode metrics]#
```

If the output has more metrics than that suggested by when the `--initialize` flag is used, then the extra sampled data is discarded. If the output has less metrics, then the metrics are set to `NaN` (not a number) for the sample.

I.4 Error Handling

As long as the exit code of the script is 0, the framework assumes that there is no error. So, with the `--initialize` flag active, despite no numeric value output, the script does not exit with an error.

If the exit code of the script is non-zero, the output of the script is assumed to be a diagnostic message and passed to the head node. This in turn will be shown as an event in `cmsh` or `cmgui`.

For example, the `sample_ipmi` script uses the `ipmi-sensors` binary internally. Calling the binary directly returns an error code if the device has

no IPMI configured. However, the `sample_ipmi` script in this case simply returns 0, and no output. The rationale here being that the administrator is aware of this situation and would not expect data from that IPMI anyway, let alone an error.

I.5 Environment Variables

The following environment variables are available for a metric collection script (as well as for custom scripts):

On all devices:

`CMD_HOSTNAME`: name of the device. For example:

```
CMDHOSTNAME=myheadnode
```

Only on non-node devices:

`CMD_IP`: IP address of the device. For example:

```
CMD_IP=192.168.1.33
```

Only on node devices:

Because these devices generally have multiple interfaces, the single environment variable `CMD_IP` is often not enough to express these. Multiple interfaces are therefore represented by these environment variables:

- `CMD_INTERFACES`: list of names of the interfaces attached to the node. For example:

  ```
  CMD_INTERFACES=eth0 eth1 ipmi0 BOOTIF
  ```

- `CMD_INTERFACE_<interface>_IP`: IP address of the interface with the name <interface>. For example:

  ```
  CMD_INTERFACE_eth0_IP=10.141.255.254
  CMD_INTERFACE_eth1_IP=0.0.0.0
  ```

- `CMD_INTERFACE_<interface>_TYPE`: type of interface with the name <interface>. For example:

  ```
  CMD_INTERFACE_eth1_TYPE=NetworkPhysicalInterface
  CMD_INTERFACE_ipmi0_TYPE=NetworkIpmiInterface
  ```

  Possible values are:

  - `NetworkIpmiInterface`
  - `NetworkPhysicalInterface`
  - `NetworkVLANInterface`
  - `NetworkAliasInterface`
  - `NetworkBondInterface`

- `CMD_IPMIUSERNAME`: username for the IPMI device at this node (if available).

- CMD_IPMIPASSWORD: password for the IMPI device at this node (if available).

To parse the above information to get the IPMI IP address of the node for which this script samples, one could use (in perl):

```
my $ip;
my $interfaces = $ENV{"CMD_INTERFACES"};
foreach my $interface ( split( " " ,  $interfaces ) ) {
  if( $ENV{"CMD_INTERFACE_" . $interface . "_TYPE"} eq
"NetworkIpmiInterface" ) {
    $ip = $ENV{"CMD_INTERFACE_" . $interface . "_IP"};
    last;
  }
}
# $ip holds the ipmi ip
```

I.6 Metric Collections Examples

Bright Cluster Manager has several scripts in the /cm/local/apps/cmd/scripts/metrics directory. Among them are the metric collections scripts testmetriccollection and sample_responsiveness. A glance through them while reading this appendix may be helpful.

I.7 iDataPlex And Similar Units

IBM's iDataPlex is a specially engineeered dual node rack unit. When the term iDataPlex is used in the following text in this section, it also implies any other dual node units that show similar behavior.

This section gives details on configuring an iDataPlex if IPMI metrics retrieval seems to skip most IPMI values from one of the nodes in the unit.

When carrying out metrics collections on an iDataPlex unit, Bright Cluster Manager should work without any issues. However, it may be that due to the special paired node design of an iDataPlex unit, most IPMI metrics of one member of the pair are undetectable by the sample_ipmi script sampling on that particular node. The missing IPMI metrics can instead be retrieved from the second member in the pair (along with the IPMI metrics of the second member).

The output may thus look something like:

Example

```
[root@master01 ~]# cmsh
[master01]% device latestmetricdata node181 | grep Domain
Metric                       Value
---------------------------- -----
Domain_A_FP_Temp             23
Domain_A_Temp1               39
Domain_A_Temp2               37
Domain_Avg_Power             140
Domain_B_FP_Temp             24
Domain_B_Temp1               40
Domain_B_Temp2               37
```

```
[master01]% device latestmetricdata node182 | grep Domain
Metric                         Value
------------------------------ -----
Domain_A_FP_Temp               no data
Domain_A_Temp1                 no data
Domain_A_Temp2                 no data
Domain_Avg_Power               170
Domain_B_FP_Temp               no data
Domain_B_Temp1                 no data
Domain_B_Temp2                 no data
[master01]%
```

Because there are usually many iDataplex units in the rack, the metrics retrieval response of each node pair in a unit should be checked for this behavior.

The issue can be dealt with by Bright Cluster Manager by modifying the configuration file for the `sample_ipmi` script in `/cm/local/apps/cmd/scripts/metrics/configfiles/sample_ipmi.conf`. Two parameters that can be configured there are `chassisContainsLeadNode` and `chassisContainsLeadNodeRegex`.

- Setting `chassisContainsLeadNode` to `on` forces the `sample_ipmi` script to treat the unit as an iDataPlex unit.

  In particular:

  - `auto` (recommended) means the unit is checked by the IPMI metric sample collection script for whether it behaves like an iDataPlex unit.
  - `on` means the unit is treated as an iDataplex node pair, with one node being a lead node that has all the IPMI metrics.
  - `off` means the unit is treated as a non-iDataPlex node pair, with each node having normal behavior when retrieving IPMI metrics. This setting may need to be used in case the default value of `auto` ever falsely detects a node as part of an iDataPlex pair.

- The value of `chassisContainsLeadNodeRegex` can be set to a regular expression pattern that matches the system information pattern for the name, as obtained by CMDaemon for an iDataPlex unit (or similar clone unit). The pattern that it is matched against is the output of:

  ```
  cmsh -c 'device ; sysinfo master | grep "^System Name"'
  ```

  If the pattern matches, then the IPMI sample collection script assumes the unit behaves like an iDataPlex dual node pair. The missing IPMI data values are then looked for on the lead node.

  By default, `chassisContainsLeadNodeRegex` is set to `iDataPlex`

J

Changing The Network Parameters Of The Head Node

J.1 Introduction

After a cluster physically arrives at its site, the administrator often has to change the network settings to suit the site. Details on this are given in section 4.2.1 of the Bright Cluster Manager *Administrator Manual*. However, it relies on understanding the material leading up to that section.

This document is therefore a quickstart document explaining how to change the IPv4 network settings while assuming no prior knowledge of Bright Cluster Manager and its network configuration interface.

J.2 Method

A cluster consists of a head node and one or more regular nodes. The head node of the cluster is assumed to face the internal network (the network of regular nodes) on one interface, say `eth0`. The external network leading to the internet is then on another interface, say `eth1`. This is referred to as a *type 1* configuration in the manual.

Typically, an administrator gives the head node a static external IP address before actually connecting it up to the external network. This requires logging into the physical head node with the vendor-supplied root password. The original network parameters of the head node can then be viewed and set. For example for `eth1`:

```
# cmsh -c "device interfaces master; get eth1 ip"
0.0.0.0
```

Here, `0.0.0.0` means the interface accepts DHCP server-supplied values.

Setting a static IP address value of, for example, `192.168.1.176` and checking the value once more:

```
# cmsh -c "device interfaces master; set eth1 ip 192.168.1.176; commit"
# cmsh -c "device interfaces master; get eth1 ip"
192.168.1.176
```

Other external network parameters can be viewed and set in a similar way, as shown in table J.1. A reboot implements the networking changes.

Table J.1: External Network Parameters And How To Change Them On The Head Node

Network Parameter	Description	Operation	Command Used
IP*	IP address of head node on eth1 interface	view	`cmsh -c "device interfaces master; get eth1 ip"`
		set	`cmsh -c "device interfaces master; set eth1 ip address; commit"`
baseaddress*	base IP address (network address) of network	view	`cmsh -c "network get externalnet baseaddress"`
		set	`cmsh -c "network; set externalnet baseaddress address; commit"`
broadcastaddress*	broadcast IP address of network	view	`cmsh -c "network get externalnet broadcastaddress"`
		set	`cmsh -c "network; set externalnet broadcastaddress address; commit"`
netmaskbits	netmask in CIDR notation (number after "/", or prefix length)	view	`cmsh -c "network get externalnet netmaskbits"`
		set	`cmsh -c "network; set externalnet netmaskbits bitsize; commit"`
gateway*	gateway (default route) IP address	view	`cmsh -c "network get externalnet gateway"`
		set	`cmsh -c "network; set externalnet gateway address; commit"`
nameservers*, **	nameserver IP addresses	view	`cmsh -c "partition get base nameservers"`
		set	`cmsh -c "partition; set base nameservers address; commit"`
searchdomains**	name of search domains	view	`cmsh -c "partition get base searchdomains"`
		set	`cmsh -c "partition; set base searchdomains hostname; commit"`
timeservers**	name of timeservers	view	`cmsh -c "partition get base timeservers"`
		set	`cmsh -c "partition; set base timeservers hostname; commit"`

* If *address* is set to 0.0.0.0 then the value offered by the DHCP server on the external network is accepted

** Space-separated multiple values are also accepted for these parameters when setting the value for *address* or *hostname*.

J.3 Terminology

A reminder about the less well-known terminology in the table:

- `netmaskbits` is the netmask size, or prefix-length, in bits. In IPv4's 32-bit addressing, this can be up to 31 bits, so it is a number between 1 and 31. For example: networks with 256 (2^8) addresses (i.e. with host addresses specified with the last 8 bits) have a netmask size of 24 bits. They are written in CIDR notation with a trailing "`/24`", and are commonly spoken of as "slash 24" networks.
- `baseaddress` is the IP address of the network the head node is on, rather than the IP address of the head node itself. The `baseaddress` is specified by taking `netmaskbits` number of bits from the IP address of the head node. Examples:
  - A network with 256 (2^8) host addresses: This implies the first 24 bits of the head node's IP address are the network address, and the remaining 8 bits are zeroed. This is specified by using "0" as the last value in the dotted-quad notation (i.e. zeroing the last 8 bits). For example: 192.168.3.0
  - A network with 128 (2^7) host addresses: Here `netmaskbits` is 25 bits in size, and only the last 7 bits are zeroed. In dotted-quad notation this implies "128" as the last quad value (i.e. zeroing the last 7 bits). For example: 192.168.3.128.

When in doubt, or if the preceding terminology is not understood, then the values to use can be calculated using the head node's `sipcalc` utility. To use it, the IP address in CIDR format for the head node must be known.

When run using a CIDR address value of 192.168.3.130/25, the output is (some output removed for clarity):

```
# sipcalc 192.168.3.130/25

Host address            - 192.168.3.130
Network address         - 192.168.3.128
Network mask            - 255.255.255.128
Network mask (bits)     - 25
Broadcast address       - 192.168.3.255
Addresses in network    - 128
Network range           - 192.168.3.128 - 192.168.3.255
```

Running it with the `-b` (binary) option may aid comprehension:

```
# sipcalc -b 192.168.3.130/25

Host address            - 11000000.10101000.00000011.10000010
Network address         - 11000000.10101000.00000011.10000000
Network mask            - 11111111.11111111.11111111.10000000
Broadcast address       - 11000000.10101000.00000011.11111111
Network range           - 11000000.10101000.00000011.10000000 -
                          11000000.10101000.00000011.11111111
```

K

Bright Cluster Manager Python API

This appendix introduces the Python API of the Bright Cluster Manager.

K.1 Installation

The Python cluster manager bindings are pre-installed on the head node.

K.1.1 Windows Clients

For windows clients, Python version 2.5.X is needed. Newer versions of Python do not work with the API.

For Windows a redistributable package is supplied in the `pythoncm-dist` package installed on the cluster. The file at `/cm/shared/apps/pythoncm/dist/windows-pythoncm.5.2.r11221.zip`—the exact version number may differ—is copied to the Windows PC and unzipped.

A Windows shell (`cmd.exe`) is opened to the directory where the Python bindings are. The `headnodeinfo.py` example supplied with the unzipped files has a line that has the following format:

```
cluster = clustermanager.addCluster(<parameters>);
```

where <*parameters*> is either:

```
'<URL>', '<PEMauth1>', '<PEMauth2>'
```
or
```
'<URL>', '<PFXauth>', '', '<password>'
```

The <*parameters*> entry is edited as follows:

- the correct hostname is set for the <*URL*> entry. By default it is set to `https://localhost:8081`
- If PEM key files are to be used for client authentication,
  - <*PEMauth1*> is set to path of `cert.pem`
  - <*PEMauth2*> is set to the path of `cert.key`
- If a PFX file is used for client authentication,

- <*PFXauth*> is set to path of `admin.pfx`
- <*password*> is set to the password

To verify everything is working, it can be run as follows:

```
c:\python25\python headnodeinfo.py
```

K.1.2 Linux Clients

For Linux clients, a redistributable source package is supplied in the `pythoncm-dist` package installed on the cluster. The file at `/cm/shared/apps/pythoncm/dist/pythoncm-5.2-r11221-src.tar.bz2`—the exact version number may differ—is copied and untarred to any directory.

The `build.sh` script is then run to compile the source. About 4GB of memory is usually needed for compilation, and additional packages may be required for compilation to succeed. A list of packages needed to build Python cluster manager bindings can be found in the `README` file included with the package.

The `headnodeinfo.py` example supplied with the untarred files is edited as for in the earlier windows client example, for the `clustermanager.addCluster` line.

The path to the remote cluster manager library is added:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:remotecm
```

To verify everything is working, the following can be run:

```
python ./headnodeinfo.py
```

K.2 Examples

A set of examples can be found in `/cm/local/examples/cmd/python/` on the head node of the cluster.

K.2.1 First Program

A Python script is told to use the cluster manager bindings by importing `pythoncm` at the start of the script:

```
import pythoncm
```

If not working on the cluster, the administrator needs to set the path where the shared libraries can be found (`pythoncm.so` in Linux, or `python.pyd` in windows). This is done by adding the following to the start of the script:

```
import sys
sys.path.append(".")    # path to pythoncm.so/python.pyd
```

Python cluster manager bindings allow for simultaneous connections to several clusters. For this reason the first thing to do is to create a ClusterManager object:

```
clustermanager = pythoncm.ClusterManager()
```

A connection to a cluster can now be made. There are two possible ways of connecting.

The first is using a certificate and private key file similar to what `cmsh` uses by default when it authenticates from the head node.

```
cluster = clustermanager.addCluster('https://mycluster:8081',\
'/cm/local/apps/cmd/etc/cert.pem', '/cm/local/apps/cmd/etc/cert.key');
```

The second way uses the password protected `admin.pfx` file similar to `cmgui`. A Python script could ask for the password and store it in a variable for increased security.

```
cluster = clustermanager.addCluster('https://mycluster:8081',\
'/root/.cm/cmgui/admin.pfx', '', '<password>');
```

Having defined the cluster, a connection can now be made to it:

```
isconnected = cluster.connect()
if !isconnected:
 print "Unable to connect"
 print cluster.getLastError()
 exit(1)
```

If a connection cannot be made, the Boolean `isconnected` is set to false. The function 'cluster.getLastError()' shows details about the problem. The two most likely problems are due to a wrong password setting or a firewall settings issue.

Similar to `cmgui` and `cmsh`, the cluster object contains a local cache of all objects. This cache will be filled automatically when the connection is established. All changes to properties will be done on these local copies and will be lost after the Python scripts exits, unless a `commit` operation is done.

The most common operation is finding specific objects in the cluster.

```
active = cluster.find('active')
if active == None:
 print "Unable to find active head node"
 exit(1)
else:
 print "Hostname of the active head node is %s" % master.hostname
```

If creating an automated script that runs at certain times, then it is highly recommended to check if objects can be found. During a failover, for instance, there will be a period over a few minutes in which the active head node will not be set.

It is good practice to disconnect from the cluster at the end of the script.

```
cluster.disconnect()
```

When connecting to a cluster with a failover setup, it is the shared IP address that should be connected to, and not the fixed IP address of either of the head nodes.

K.3 Methods And Properties

K.3.1 Viewing All Properties And Methods

All properties visible in `cmsh` and `cmgui` are also accessible from Python cluster manager bindings. The easiest way to get an overview of the methods and properties of an object is to define the following function:

```
import re
def dump(obj):
  print "--- DUMP ---"
  for attr in dir(obj):
    p = re.compile('^__.*__$')
    if not p.match(attr):
      print "%s = %s" % (attr, getattr(obj, attr))
```

An overview of all properties and methods for the active master node can be obtained with:

```
active = cluster.find('active')
dump(active)
```

K.3.2 Property Lists

Most properties are straightforward and their names are almost identical to the `cmsh` equivalent.

For instance:

```
node.mac = '00:00:00:00:00:00'
category.softwareimage = cluster.find('testimage')
```

Properties that contain lists, like `node.roles`, `node.interfaces`, `category.fsmounts` and several others, are trickier to deal with. While iterating over a list property is simple enough:

```
for role in node.roles:
  print role.name
```

because of an implementation restriction, adding a new role requires that a local copy of the roles list be made:

```
roles = node.roles
provisioningrole = pythoncm.ProvisioningRole()   # Create a new pro\
                                                 visioning role object
roles.append(provisioningrole)
node.roles = roles            # This will update the internal\
                                roles list with the local copy
```

K.3.3 Creating New Objects

Creating a new node can be done with:

```
node = pythoncm.Node()
```

This is valid command, but fairly useless because a node has to be a `MasterNode`, `PhysicalNode` or `VirtualSMPNode`. So to create a normal compute or login node, the object is created as follows:

```
node = pythoncm.PhysicalNode()
```

The first thing to do after creating a new object is to add it to a cluster.

```
cluster.add(node)
```

It is impossible to add one node to more than one cluster.

After the node has been added its properties can be set. In `cmsh` and `cmgui` this is semi-automated, but in Python cluster manager bindings it has to be done by hand.

```
node.hostname = 'node001'
node.partition = cluster.find('base')
node.category = cluster.find('default')
```

Similar to the node object, a NetworkInterface object has several subtypes: NetworkPhysicalInterface, NetworkVLANInterface, NetworkAliasInterface, NetworkBondInterface, and NetworkIPMIInterface.

```
interface = pythoncm.NetworkPhysicalInterface()
interface.name = 'eth0'
interface.ip = '10.141.0.1'
interface.network = cluster.find('internalnet')
node.interfaces = [interface]
node.provisioningInterface = interface
```

Having set the properties of the new node, it can now be committed.

```
cr = node.commit()
```

If a commit fails for some reason, the reason can be found:

```
if not cr.result:
  print "Commit of %s failed:" % node.resolveName()
  for j in range(cr.count):
    print cr.getValidation(j).msg
```

K.3.4 List Of Objects

In the following lists of objects:

- Objects marked with (*) cannot be used
- Trees marked with (+) denote inheritance

Roles

```
Role (*)
+ BackupRole
+ BootRole
+ DatabaseRole
+ EthernetSwitch
+ LoginRole
+ LSFClientRole
+ LSFServerRole
+ MasterRole
+ PbsProClientRole
+ PbsProServerRole
+ ProvisioningRole
+ SGEClientRole
+ SGEServerRole
+ SlurmClientRole
+ SlurmServerRole
+ SubnetManagerRole
+ TorqueClientRole
+ TorqueServerRole
```

Devices

```
Device (*)
+ Chassis
+ GpuUnit
+ GenericDevice
+ PowerDistributionUnit
+ Switch (*)
   + EthernetSwitch
   + IBSwitch
   + MyrinetSwitch
   Node (*)
   + MasterNode
   + SlaveNode (*)
     + PhysicalNode
     + VirtualSMPNode
```

Network Interfaces

```
NetworkInterface (*)
+ NetworkAliasInterface
+ NetworkBondInterface
+ NetworkIpmiInterface
+ NetworkPhysicalInterface
+ NetworkVLANInterface
```

Information Objects

```
ClusterSetup
GuiClusterOverview
GuiGpuUnitOverview
GuiNodeOverview
GuiNodeStatus
LicenseInfo
SysInfoCollector
VersionInfo
```

Monitoring Configuration Objects

```
MonConf
ConsolidatorConf
MonHealthConf
HealthCheck
MonMetricConf
ThreshActionConf
ThreshAction
Threshold
```

LDAP Objects

```
User
Group
```

Category Objects

```
Category
FSExport
FSMount
```

Miscellaneous Objects

```
SoftwareImage
```

```
KernelModule

Network

NodeGroup

Partition

Rack
```

K.3.5 Useful Methods

For The Cluster Object:

Name	Description
`find(<name>)`	Find the object with a given name, <*name*>
`find(<name>, <type>)`	Because it is possible to give a category and node the same name, sometimes the type <*type*> of the object needs to be specified too
`getAll(<type>)`	Get a list of all objects of a given type: e.g. device, category
`activeMaster()`	Get the active master object
`passiveMaster()`	Get the active master object
`overview()`	Get all the data shown in the `cmgui` cluster overview
`add(<object>)`	Add a newly created object <*object*> to the cluster. Only after an object is added can it be used
`pexec(<nodes>, <command>)`	Execute a command <*command*> on one or more nodes

For Any Object:

Name	Description
`commit()`	Save changes to the cluster
`refresh()`	Undo all changes and restore the object to its last saved state
`remove()`	Remove an object from the cluster
`clone()`	Make an identical copy. The newly created object is not added to a cluster yet

For Any Device:

Name	Description
close()	Close a device
open()	Open a device
powerOn()	Power on a device
powerOff()	Power off a device
powerReset()	Power reset a device
latestMonitoringData()	Return a list of the most recent monitoring data

For Any Node:

Name	Description
overview()	Get the data displayed in the cmgui node overview tab
sysinfo()	Get the data displayed in the cmgui node system information tab
pexec(*<command>*)	Execute a command

K.3.6 Useful Example Program

In the directory /cm/local/examples/cmd/python are some example programs using the python API.

One of these is printall.py. It displays values for objects in an easily viewed way. With all as the argument, it displays resource objects defined in a list in the program. The objects are 'Partition', 'MasterNode', 'SlaveNode', 'Category', 'SoftwareImage', 'Network', 'NodeGroup'. The output is displayed something like (some output elided):

Example

```
[root@bright60 ~]# cd /cm/local/examples/cmd/python
[root@bright60 python]# ./printall all
Partition base
 +- revision .....................
 |  name ......................... base
 |  clusterName ................... Bright 6.0b Cluster
...
 |  burnConfigs
 |   +- revision ..................
 |   |  name ...................... default
 |   |  description ............... Standard burn test.
 |   |  configuration ............. < 2780 bytes >
 |   +- revision ..................
 |   |  name ...................... long-hpl
...
 |  provisioningInterface ......... None
 |  fsmounts ...................... < none >
 |  fsexports
 |   +- revision ..................
 |   |  name ...................... /cm/shared@internalnet
```

```
 |   |  path ..................... /cm/shared
 |   |  hosts ..................... !17179869185!
...
Category default
 +- revision .....................
 |  name ......................... default
 |  softwareImage ................. default-image
 |  defaultGateway ................ 10.141.255.253
 |  nameServers ................... < none >
...
```

The values of a particular resource-level object, such as `nodegroup`, can be viewed by specifying it as the argument:

Example

```
[root@bright60 python]# ./printall.py nodegroup
nodegroup us-east-1-director-dependents
 +- revision .....................
 |  name ......................... us-east-1-director-dependents
 |  nodes ........................ < none >
 |  type .......................... CLOUDDIRECTOR
 |  firstSeenTime ................. 0
 |  lastSeenTime .................. 0
nodegroup us-east-1-director-dependents-2
 +- revision .....................
 |  name ......................... us-east-1-director-dependents-2
 |  nodes ........................ < none >
 |  type .......................... CLOUDDIRECTOR
 |  firstSeenTime ................. 1327593444
 |  lastSeenTime .................. 1327593444
[root@bright60 python]#
```

L

Workload Manager Configuration Files Updated By CMDaemon

This appendix lists workload manager configuration files changed by CMDaemon, events causing such change, and the file or property changed.

L.1 Slurm

File/Property	Updates What?	Updated During
`/etc/slurm.conf`	head node	Add/Remove/Update nodes, hostname change
`/etc/slurmdbd.conf`	head node	Add/Remove/Update nodes, hostname change

L.2 Grid Engine

File/Property	Updates What?	Updated During
`$ROOT/default/common/host_aliases`	head node	hostname/domain change, failover
`$ROOT/default/common/act_qmaster`	head node	hostname/domain change, failover
`queue hostlist`	head node	Add/Remove/Update nodes
`queue slots`	head node	Add/Remove/Update nodes
`queue ngpus`	head node	Add/Remove/Update nodes

L.3 Torque

File/Property	Updates What?	Updated During
`$ROOT/spool/server_name`	head node	hostname/domain change, failover
`$ROOT/spool/torque.cfg`	head node	hostname/domain change, failover
`$ROOT/server_priv/acl_svr/acl_hosts`	head node	hostname/domain change, failover
`$ROOT/spool/server_priv/acl_svr/operators`	head node	hostname change, failover
`$ROOT/spool/server_priv/nodes`	head node	Add/Remove/Update nodes
`$ROOT/mom_priv/config`	software image	hostname change, failover

L.4 PBS Pro

File/Property	Updates What?	Updated During
`/etc/pbs.conf`	head node, software image	hostname/domain change, failover
`$ROOT/server_priv/acl_svr/operators`	head node	hostname change, failover
`$ROOT/spool/server_priv/nodes`	head node	Add/Remove/Update nodes
`queue acl_hosts`	head node	Add/Remove/Update nodes

M

Linux Distributions That Require Registration

This appendix describes setting up registered access for the Bright Cluster Manager nodes with the Red Hat and SUSE distributions.

The head and regular nodes need registered access to the enterprise Linux distributions of Red Hat and SUSE so that updates from their repositories can take place on the cluster correctly. This allows the distributions to continue to provide support and security updates.

Registered access is also needed in order to create an up-to-date custom software image (section 9.6) if using Red Hat or SUSE as the base distribution.

M.1 Registering A Red Hat Enterprise Linux Based Cluster

To register a Red Hat Enterprise Linux (RHEL) system, Red Hat subscriptions are needed as described at `https://www.redhat.com/`. Registration with the Red Hat Network is needed to install new RHEL packages or receive RHEL package updates, as well as carry out some other tasks.

M.1.1 Registering A Head Node With RHEL

The `rhn_register` command can be used to register an RHEL head node. If the head node has no direct connection to the internet, an HTTP proxy can be configured as a command line option. The `rhn_register` man pages give details on configuring the proxy from the command line.

The head node can be registered in text mode by running:

```
[root@bright52 ~]# rhn_register --nox
```

When the `rhn_register` command is run, the following screens are gone through in a standard run:

- Connect to Red Hat Network
- Information
- Enter login information for Red Hat Network
- Register a System Profile—Hardware

- Register a System Profile—Packages
- Send Profile Information to Red Hat Network
- Review system subscription details
- Finish setup

Some of the screens may require inputs before going on to the next screen.

The Red Hat Network base software channel subscription configuration is displayed in the "`Review system subscription details`" screen.

The Red Hat Network software channels subscriptions configuration can also be viewed outside of the `rhn_register` command by using the `rhn-channel` command.

For RHEL5 it can be run as:

```
[root@bright52 ~]# rhn-channel -l
```

For RHEL6 it can be run as:

```
[root@bright52 ~]# rhn-channel -L -u <Red Hat Network username> -p \
<Red Hat Network password>
```

For a head node based on RHEL5, the following Red Hat Network software channels subscriptions need to be configured:

- rhel-x86_64-server-5
- rhel-x86_64-server-supplementary-5

For a head node based on RHEL6 these are:

- rhel-x86_64-server-6
- rhel-x86_64-server-optional-6

Typically, on an RHEL5-based head node, the `rhel-x86_64-server-supplementary-5` channel must still be added to the configuration. To do this, the `rhn-channel` command can be used as follows:

```
[root@bright52 ~]# rhn-channel -a -c rhel-x86_64-server-supplementary-5 \
-u <Red Hat Network username> -p <Red Hat Network password>
```

Similarly, for an RHEL6-based head node configuration the `rhel-x86_64-server-optional-6` channel is added with:

```
[root@bright52 ~]# rhn-channel -a -c rhel-x86_64-server-optional-6 -u \
<Red Hat Network username> -p <Red Hat Network password>
```

After registering the system with Red Hat Network the `rhnsd` daemon is enabled, and it then becomes active after a reboot. The `rhnsd` daemon synchronizes information regularly with the Red Hat Network.

The following commands are useful for handling `rhnsd` options at this stage:

Command	Description
`service rhnsd start`	make it active without a reboot
`chkconfig rhnsd off`	stop it becoming active on boot
`service rhnsd status`	see if it is currently running

After registration, the `yum-rhn-plugin` is enabled, so that `yum` can be used to install and update from the Red Hat Network repositories.

M.1.2 Registering A Software Image With RHEL

The `rhn_register` command can be used to register an RHEL software image. If the head node, on which the software image resides, has no direct connection to the internet, then an HTTP proxy can be configured as a command line option. The `rhn_register` man pages give details on configuring the proxy from the command line.

The default software image, `default-image`, can be registered by running the following on the head node:

```
[root@bright52 ~]# chroot /cm/images/default-image rhn_register --nox
```

When the `rhn_register` command is run, the following screens are gone through in a standard run:

- Connect to Red Hat Network
- Information
- Enter login information for Red Hat Network
- Register a System Profile—Hardware
- Register a System Profile—Packages
- Send Profile Information to Red Hat Network
- System subscription details
- Finish setup

Some of the screens may require inputs before going on to the next screen.

The Red Hat Network base software channel subscription configuration is displayed in the "`Review system subscription details`" screen.

The Red Hat Network software channels subscriptions configuration can also be viewed outside of the `rhn_register` command by using the `rhn-channel` command.

For RHEL5 it can be run from the head node as:

```
[root@bright52 ~]# chroot /cm/images/default-image rhn-channel -l
```

For RHEL6 it is:

```
[root@bright52 ~]# chroot /cm/images/default-image rhn-channel -L -u \
<Red Hat Network username> -p <Red Hat Network password>
```

For an RHEL5-based software image, the following Red Hat Network software channels subscriptions need to be configured:

- rhel-x86_64-server-5
- rhel-x86_64-server-supplementary-5

For an RHEL6-based software image these are:

- rhel-x86_64-server-6
- rhel-x86_64-server-optional-6

Typically, for an RHEL5-based software image, the `rhel-x86_64-server-supplementary-5` channel must still be added to the configuration. To do this, the `rhn-channel` command can be used as follows on the head node:

```
[root@bright52 ~]# chroot /cm/images/default-image rhn-channel -a -c \
rhel-x86_64-server-supplementary-5 -u <Red Hat Network username> -p \
<Red Hat Network password>
```

Similarly, for an RHEL6-based software image configuration the `rhel-x86_64-server-optional-6` channel must be added. This can be done with:

```
[root@bright52 ~]# chroot /cm/images/default-image rhn-channel -a -c \
rhel-x86_64-server-optional-6 -u <Red Hat Network username> -p \
<Red Hat Network password>
```

After registering the software image with the Red Hat Network, the `rhnsd` daemon is enabled, and it then becomes active when a node boots. The `rhnsd` daemon synchronizes information regularly with the Red Hat Network.

The `rhnsd` daemon can be turned off in the default `default-image` software image with:

```
[root@bright52 ~]# chroot /cm/images/default-image chkconfig rhnsd off
```

After registering, the `yum-rhn-plugin` is enabled within the software image, so `yum` can be used for the software image to install and update from the Red Hat Network repositories.

M.2 Registering A SUSE Linux Enterprise Server Based Cluster

To register a SUSE Linux Enterprise Server system, SUSE Linux Enterprise Server subscriptions are needed as described at `http://www.suse.com/`. Registration with Novell is needed to install new SLES packages or receive SLES package updates, as well as to carry out some other tasks.

M.2.1 Registering A Head Node With SUSE

The `suse_register` command can be used to register a SUSE head node. If the head node has no direct connection to the internet, then the `HTTP_PROXY` and `HTTPS_PROXY` environment variables can be set, to access the internet via a proxy. Running the command with the help option, as "`suse_register --help`", provides further information about the command and its options.

The head node can be registered as follows:

```
[root@bright52 ~]# suse_register -a email=<e-mail address> -a regcode-\
sles=<activation code> --restore-repos
```

The e-mail address used is the address that was used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting "SUSE Linux Enterprise Server".

After registering, the SLES and SLE SDK repositories are added to the repository list and enabled.

The repository list can be viewed with:

```
[root@bright52 ~]# zypper lr
```

and the head node can be updated with:

```
[root@bright52 ~]# zypper refresh
[root@bright52 ~]# zypper update
```

M.2.2 Registering A Software Image With SUSE

The suse_register command can be used to register a SUSE software image. If the head node on which the software image resides has no direct connection to the internet, then the HTTP_PROXY and HTTPS_PROXY environment variables can be set to access the internet via a proxy. Running the command with the help option, as "suse_register --help", provides further information about the command and its options.

The default software image default-image can be registered by running the following on the head node:

```
[root@bright52 ~]# chroot /cm/images/default-image suse_register -n -a \
email=<e-mail address> -a regcode-sles=<activation code> --restore-repos
```

The e-mail address is the address used to register the subscription with Novell. When logged in on the Novell site, the activation code or registration code can be found at the products overview page after selecting "SUSE Linux Enterprise Server".

When running the suse_register command, warnings about the /sys or /proc filesystems can be ignored. The command tries to query hardware information via these filesystems, but these are empty filesystems in a software image, and only fill up on the node itself after the image is provisioned to the node.

Instead of registering the software image, the SLES repositories can be enabled for the default-image software image with:

```
[root@bright52 ~]# cp /etc/zypp/repos.d/*novell* /cm/images/default-ima\
ge/etc/zypp/repos.d/
[root@bright52 ~]# cp /etc/zypp/credentials.d/NCCcredentials /cm/images\
/default-image/etc/zypp/credentials.d/
```

The repository list of the default-image software image can be viewed with the chroot option, -R, as follows:

```
[root@bright52 ~]# zypper -R /cm/images/default-image lr
```

and the software image can be updated with:

```
[root@bright52 ~]# export PBL_SKIP_BOOT_TEST=1
[root@bright52 ~]# zypper -R /cm/images/default-image refresh
[root@bright52 ~]# zypper -R /cm/images/default-image update
[root@bright52 ~]# zypper -R /cm/images/default-image clean --all
```

N

Burning Nodes

The *burn framework* is a component of Bright Cluster Manager 5.2 that can automatically run test scripts on specified nodes within a cluster. The framework is designed to stress test newly built machines and to detect components that may fail under load.

N.1 Test Scripts Deployment

The framework requires power management to be running so that the node can be power cycled by the scripts used. In modern clusters power management is typically achieved by enabling a baseboard managment controller such as IPMI or iLO. Details on power management are given in chapter 5.

The framework can run any executable script. The default test scripts are mostly `bash` shell scripts and Perl scripts. Each test script has a directory in `/cm/shared/apps/cmburn` containing the script. The directory and test script must have the same name. For example: `/cm/shared/apps/cmburn/disktest/disktest` is the default script used for testing a disk. More on the contents of a test script is given in section N.3.3.

N.2 Burn Configurations

A *burn configuration* file specifies the order of the tests that are run. Within the burn configuration the tests are normally grouped into sequences, and several sequences typically make up a phase. Phases in turn are grouped in either a pre-install section or post install section. A simple example of such a burn configuration could therefore look like:

Example

```
<?xml version="1.0"?>
<burnconfig>

  <mail>
    <address>root@master</address>
    <address>some@other.address</address>
  </mail>

  <pre-install>
```

```
    <phase name="01-hwinfo">
      <test name="hwinfo"/>
      <test name="hwdiff"/>
      <test name="sleep" args="10"/>
    </phase>

    <phase name="02-disks">
      <test name="disktest" args="30"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </pre-install>

  <post-install>

    <phase name="03-hpl">
      <test name="hpl"/>
      <test name="mce_check" endless="1"/>
    </phase>

    <phase name="04-compile">
      <test name="compile" args="6"/>
      <test name="mce_check" endless="1"/>
    </phase>

  </post-install>

</burnconfig>
```

N.2.1 Mail Tag

The *mail tag* is an optional tag to add a sequence of e-mail addresses. These addresses receive burn failure and warning messages, as well as a notice when the burn run has completed.

N.2.2 Pre-install And Post-install

The pre-install part of a burn configuration is run from inside a node-installer environment. This environment is a limited Linux environment and allows some simpler tests to run before loading up the full Linux node environment.

The post-install part of a burn configuration is run from inside the full Linux node environment. This environment allows more complex tests to run.

N.2.3 Phases

The phases sections must exist. If there is no content for the phases, the phases tags must still be in place ("must exist"). Each phase must have a unique name and must be written in the burn configuration file in alphanumerical order. By default, numbers are used as prefixes. The phases are executed in sequence.

N.2.4 Tests

Each phase consists of one or more test tags. The tests can optionally be passed arguments using the args property of the burn configuration file (section N.2). If multiple arguments are required, they should be a space separated list, with the (single) list being the args property.

Tests in the same phase are run simultaneously.

Most tests test something and then end. For example, the disk test tests the performance of all drives and then quits.

Tests which are designed to end automatically are known as *non-endless* tests.

Tests designed to monitor continuously are known as *endless tests*. Endless tests are not really endless. They end once all the non-endless tests in the same phase are ended, thus bringing an end to the phase. Endless tests typically test for errors caused by the load induced by the non-endless tests. For example the `mce_check` test continuously keeps an eye out for Machine Check Exceptions while the non-endless tests in the same phase are run.

N.3 Running A Burn Configuration

Burn configurations can be viewed and executed from `cmsh` or `cmgui`.

N.3.1 Burn Configuration And Execution In `cmgui`

From within `cmgui` the configuration can be selected for a particular node by selecting a node from the `Nodes` resource, then selecting the `Burn` tab. Clicking on the "`Start New Burn`" button opens up the burn configuration file selection dialog (figure N.1).

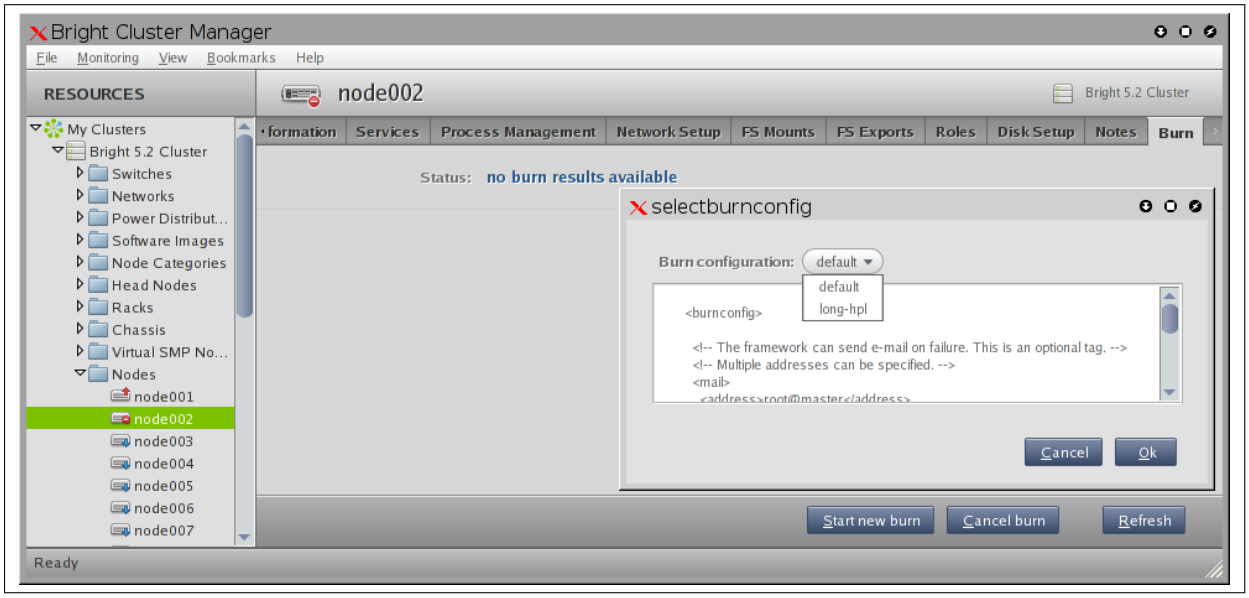

Figure N.1: `cmgui`: Starting A New Burn

Clicking on the `OK` button then means the burn setting is turned on, as well as sending a power reset signal to the node via the Baseboard Management Controller or the APC. The burn setting being on means the node starts up in burn mode when starting up from now on (section 6.5.1).

The "Cancel burn" button can be used to turn the burn setting off from cmgui, and then also sends a power reset signal.

The burn status of a node can be monitored for each individual node (figure N.2).

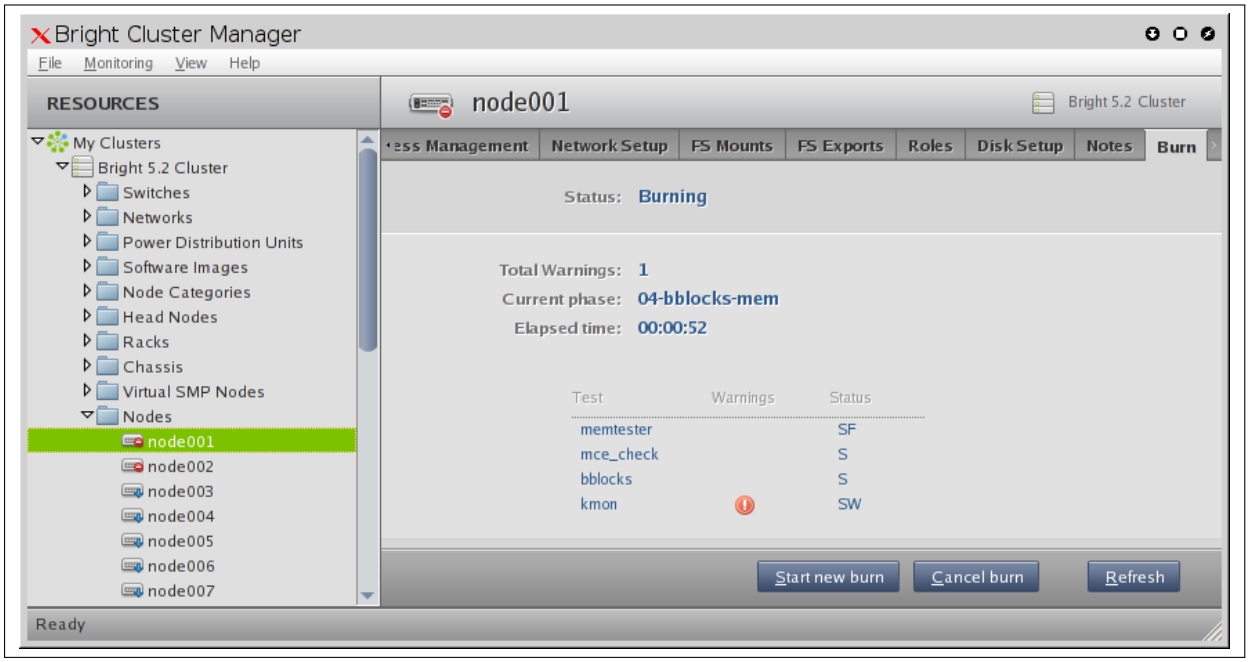

Figure N.2: cmgui: Status Of A Burn Run—Node View

An overview of the burn status of nodes in a particular category can also be viewed for all nodes in the category (figure N.3). The category view is accessible when selecting the Nodes resource folder, and then selecting the "Burn Overview" tab. It can also be accessed by selecting the category item associated with the node from "Node Categories" resource, and then selecting the "Burn Overview" tab.

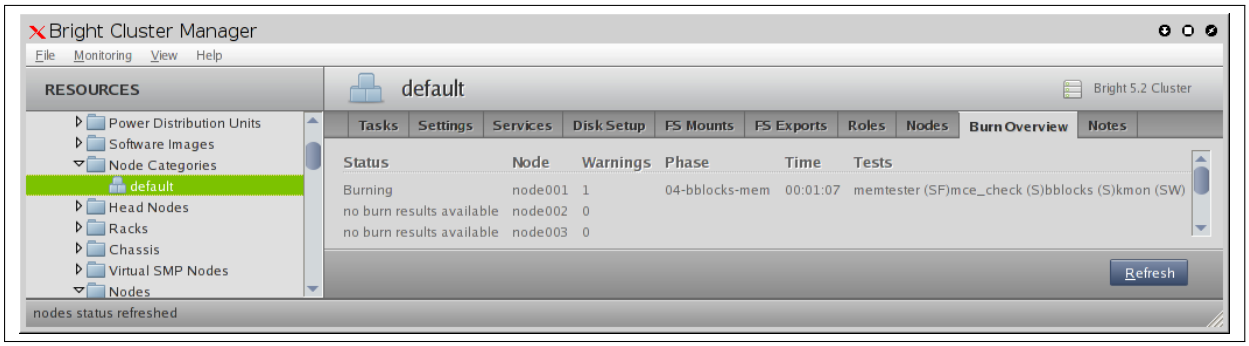

Figure N.3: cmgui: Status Of A Burn Run—Category View

N.3.2 Burn Configuration And Execution In cmsh

Burn configuration and execution can be carried out using cmgui (section N.3.1), or using cmsh. Using cmsh has some extra features, including the ability to create or modify a new burn configuration file, and also the ability to have the burn execution wait for a separate manual power reset.

Burn Configuration File Settings

From cmsh, the burn configurations can be accessed from partition mode as follows:

Example

```
[bright52]% partition use base
[bright52->partition[base]]% burnconfigs
[bright52->partition[base]->burnconfigs]% list
Name (key)    Description              XML
------------- ------------------------ -------------
default       Standard burn test.      <2780 bytes>
long-hpl      Run HPL test for a long+ <879 bytes>
```

The values of a particular burn configuration (default in the following example) can be viewed as follows:

Example

```
[bright52->partition[base]->burnconfigs]% use default
[bright52->partition[base]->burnconfigs[default]]% show
Parameter                      Value
------------------------------ --------------------------
Description                    Standard burn test.
Name                           default
Revision
XML                            <2780 bytes>
```

The set command can be used to modify existing values of the burn configuration, that is: Description, Name and XML. XML is the burn configuration file itself, and the default text editor opens up when using the set command on it, thus allowing the burn configuration to be modified.

A new burn configuration can also be added with the add command. The new burn configuration can be created from scratch with the set command. However, an XML file can also be imported to the new burn configuration by specifying the full path of the XML file to be imported:

Example

```
[bright52->partition[base]->burnconfigs]% add boxburn
[bright52->partition[base]->burnconfigs*[boxburn*]]% set xml /tmp/im.xml
```

The burn configuration can also be edited when carrying out burn execution with the burn command.

Executing A Burn

A burn as specified by the burn configuration file can be executed using cmgui (section N.3.1), or using cmsh. Execution using cmsh has some extra features, including the ability to have the burn execution wait for a separate manual power reset.

Burn-related properties: To execute a burn configuration file on a node in cmsh, the node object is accessed from device mode in cmsh. Among its properties, a node has

- Burn config: the selected burn configuration file

- `Burning`: the burn setting of the node. When its value is "`on`", and if the node has been power reset, then the node PXE boots into an image that runs burn tests according to the specifications of the burn configuration file

These properties can be viewed in `device` mode with the `show` command:

Example

```
[bright52->device[node001]]% show | grep ^Burn
Burn config                              custom <2780 bytes>
Burning                                  no
```

Burn commands: The burn commands can modify these properties, as well as execute other burn-related operations.

The burn commands are executed within device mode, and are:

- `burn start`
- `burn stop`
- `burn status`
- `burn log`

The burn help text lists the detailed options (figure N.4). Next, operations with the burn commands illustrate how the options may be used along with some features.

Burn command operations: Burn commands allow the following operations, and have the following features:

- The basic burn operations allow a burn to be started or stopped, and the status of a burn to be viewed.
  - The "`burn start`" command always needs a configuration file name (here it is "`default`"), and also always needs to be given the nodes it operates on. For example:

    ```
    burn start -o default -n node001
    ```

  - The "`burn stop`" command only needs to be given the nodes it operates on, for example:

    ```
    burn stop -n node001
    ```

  - The "`burn status`" command may be given the nodes for which the status is to be found, for example:

    ```
    burn status -n node001
    ```

  - The "`burn status`" command need not have any nodes specified for it, for example

    ```
    burn status
    ```

```
[bright52->device]% burn

Name:          burn - Node burn control

Usage:    burn [OPTIONS] status
          burn [OPTIONS] start
          burn [OPTIONS] stop
          burn [OPTIONS] log

Options:  -n, --nodes node(list)
               List of nodes, e.g. node001..node015,node20..node028,nod\
               e030 or ^/some/file/containing/hostnames

          -g, --group group(list)
               Include all nodes that belong to the node group, e.g. te\
               stnodes or test01,test03

          -c, --category category(list)
               Include all nodes that belong to the category, e.g. defa\
               ult or default,gpu

          -r, --rack rack(list)
               Include all nodes that are located in the given rack, e.\
               g rack01 or rack01..rack04

          -h, --chassis chassis(list)
               Include all nodes that are located in the given chassis,\
               e.g chassis01 or chassis03..chassis05

          -o, --config <name>
               Burn with the specified burn configuration. See in parti\
               tion burn configurations for a list of valid names

          -l, --later
               Do not reboot nodes now, wait until manual reboot

          -e, --edit
               Open editor for last minute changes

          -p, --path
               Show path to the burn log files. Of the form: /var/spool\
               /burn/<mac>.

Examples: burn -o default start -n node001
```

Figure N.4: Usage information for `burn`

in which case the burn status is shown for all nodes.

- Advanced options allow burn commands can be executed over various node groupings, that is: node lists, a group, category, rack, and chassis.

- By default a "`burn start`" or "`burn stop`" command carries out a power reset on the node or nodes, but this can be disabled
- The burn configuration file can not only be selected, but also be edited for the "`burn start`" command. This is an alternative to editing the burn configuration file in `partition` mode.
- Each node with a boot MAC address of <*mac*> has an associated burn log file, by default under `/var/spool/burn/`<*mac*>.

Burn command output examples: The `burn status` command has a compact one-line output per node:

Example

```
[bright52->device]% burn -n node001 status
node001 (00000000a000) - W(0) phase 02-disks 00:02:58 (D:H:M) FAILED, m\
ce_check (SP), disktest (SF,61), kmon (SP)
```

The fields in the preceding output example are:

Description	Value	Meaning Here
The node name	`node001`	
The node tag	`(00000000a000)`	
Warnings since start of burn	`(0)`	
The current phase name	02-disks	Burn configuration phase being run is `02-disks`
Time since phase started	`00:02:58 (D:H:M)`	2 hours 58 minutes
State of current phase	`FAILED`	Failed in `02-disks`
burn test for MCE	`mce_check (SP)`	Started and Passed
burn test for disks	`disktest (SF,61)`	Started and Failed 61 is the speed and is custom information
burn test kernel log monitor	`kmon (SP)`	Started and Passed

The "`burn log`" command output looks like the following (some output elided):

```
[bright52->device]% burn -n node001 log
Thu ... 2012: node001 - burn-control: burn framework initializing
Thu ... 2012: node001 - burn-control: e-mail will be sent to: root@master
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 01-hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/hwinfo
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/sleep
Thu ... 2012: node001 - sleep: sleeping for 10 seconds
Thu ... 2012  node001 - hwinfo: hardware information:
Thu ... 2012: node001 - hwinfo: CPU1: vendor_id = AuthenticAMD
...
Thu ... 2012: node001 - burn-control: test hwinfo has ended, test passed
Thu ... 2012: node001 - burn-control: test sleep has ended, test passed
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating endless tests
```

```
Thu ... 2012: node001 - burn-control: phase 01-hwinfo passed
Thu ... 2012: node001 - burn-control: finding next pre-install phase
Thu ... 2012: node001 - burn-control: starting phase 02-disks
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/disktest
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/mce_check
Thu ... 2012: node001 - burn-control: starting test /cm/shared/apps/cmburn/kmon
Thu ... 2012: node001 - disktest: starting, threshold = 30 MB/s
Thu ... 2012: node001 - mce_check: checking for MCE's every minute
Thu ... 2012: node001 - kmon: kernel log monitor started
Thu ... 2012: node001 - disktest: detected 1 drives: sda
...
Thu ... 2012: node001 - disktest: drive sda wrote 81920 MB in 1278.13
Thu ... 2012: node001 - disktest: speed for drive sda was 64 MB/s -> disk passed
Thu ... 2012: node001 - burn-control: test disktest has ended, test FAILED
Thu ... 2012: node001 - burn-control: all non-endless test are done, terminating endless tests
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/kmon/kmon to terminate
Thu ... 2012: node001 - kmon: kernel log monitor terminated
Thu ... 2012: node001 - burn-control: test kmon has ended, test passed
Thu ... 2012: node001 - burn-control: asking test /cm/shared/apps/cmburn/mce_check/mce_check to terminate
Thu ... 2012: node001 - mce_check: terminating
Thu ... 2012: node001 - mce_check: waiting for mce_check to stop
Thu ... 2012: node001 - mce_check: no MCE's found
Thu ... 2012: node001 - mce_check: terminated
Thu ... 2012: node001 - burn-control: test mce_check has ended, test passed
Thu ... 2012: node001 - burn-control: phase 02-disks FAILED
Thu ... 2012: node001 - burn-control: burn will terminate
```

N.3.3 Writing A Test Script

This section describes a sample test script for use within the burn framework. The script is typically a shell or Perl script. The sample that follows is a Bash script, while the `hpl` script is an example in Perl.

Section N.1 describes how to deploy the script.

Non-endless Tests

The following example test script is not a working test script, but can be used as a template for a non-endless test:

Example

```
#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=‘basename $0‘

# This first argument passed to a test script by the burn framework is a path
# to a spool directory. The directory is created by the framework. Inside the
# spool directory a sub-directory with the same name as the test is also created.
# This directory ($spooldir/$me) should be used for any output files etc. Note
# that the script should possibly remove any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
```

```
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be halted
# $warningfile _and_ $passedfile should be created. Any warnings should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the stresscpu
# test outputs something like 13/60 to this file to indicate time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# A test script can be passed arguments from the burn configuration. It is
# recommended to supply default values and test if any values have been overriden
# from the config file. Set some defaults:
option1=40
option2=some_other_value

# Test if option1 and/or option2 was specified (note that $1 was to spooldir parameter):
if [ ! x$2 = "x" ]; then
  option1=$2
fi
if [ ! x$3 = "x" ]; then
  option2=$3
fi

# Some scripts may require some cleanup. For instance a test might fail and be
# restarted after hardware fixes.
rm -f $spooldir/$me/*.out &>/dev/null

# Send a message to the burn log file, syslog and the screen.
# Always prefix with $me!
blog "$me: starting, option1 = $option1 option2 = $option2"

# Run your test here:
run-my-test
if [ its_all_good ]; then
  blog "$me: w00t, it's all good! my-test passed."
  touch $passedfile
  exit 0
elif [ was_a_problem ]; then
  blog "$me: WARNING, it did not make sense to run this test. You don't have special device X."
  echo "some warning" >> $warningfile  # note the append!
  touch $passedfile
  exit 0
else
  blog "$me: Aiii, we're all gonna die! my-test FAILED!"
  echo "Failure message." > $failedfile
  exit 0
fi
```

Endless Tests

The following example test script is not a working test, but can be used as a template for an endless test.

Example

```
#!/bin/bash

# We need to know our own test name, amongst other things for logging.
me=‘basename $0‘

# This first argument passed to a test script by the burn framework is a path
# to a spool directory. The directory is created by the framework. Inside the
# spool directory a sub-directory with the same name as the test is also created.
# This directory ($spooldir/$me) should be used for any output files etc. Note
# that the script should possibly remove any previous output files before starting.
spooldir=$1

# In case of success, the script should touch $passedfile before exiting.
passedfile=$spooldir/$me.passed

# In case of failure, the script should touch $failedfile before exiting.
# Note that the framework will create this file if a script exits without
# creating $passedfile. The file should contain a summary of the failure.
failedfile=$spooldir/$me.failed

# In case a test detects trouble but does not want the entire burn to be halted
# $warningfile _and_ $passedfile should be created. Any warnings should be written to this file.
warningfile=$spooldir/$me.warning

# Some short status info can be written to this file. For instance, the stresscpu
# test outputs something like 13/60 to this file to indicate time remaining.
# Keep the content on one line and as short as possible!
statusfile=$spooldir/$me.status

# Since this in an endless test the framework needs a way of stopping it once
# all non-endless test in the same phase are done. It does this by calling
# the script once more and passing a "-terminate" argument.
if [ "$2" == "-terminate" ]; then
  blog "$me: terminating"

  # remove the lock file the main loop is checking for
  rm $spooldir/$me/running

  blog "$me: waiting for $me to stop"
  # wait for the main loop to die
  while [ -d /proc/‘cat $spooldir/$me/pid‘ ]
  do
    sleep 1
  done
  blog "$me: terminated"

else
  blog "$me: starting test, checking every minute"

  # Some scripts may require some cleanup. For instance a test might fail and be
  # restarted after hardware fixes.
  rm -f $spooldir/$me/*.out &>/dev/null

  # create internal lock file, the script will remove this if it is requested to end
```

```
  touch $spooldir/$me/running

  # save our process id
  echo $$ > "$spooldir/$me/pid"

  while [ -e "$spooldir/$me/running" ]
  do

    run-some-check
    if [ was_a_problem ]; then
      blog "$me: WARNING, something unexpected happened."
      echo "some warning" >> $warningfile  # note the append!
    elif [ failure ]; then
      blog "$me: Aiii, we're all gonna die! my-test FAILED!"
      echo "Failure message." > $failedfile
    fi
    sleep 60

  done

  # This part is only reached when the test is terminating.
  if [ ! -e "$failedfile" ]; then
    blog "$me: no problem detected"
    touch $passedfile
  else
    blog "$me: test ended with a failure"
  fi

fi
```