Bright Cluster Manager 5.1

Administrator Manual

Revision: 6775

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 5.1 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.

1

Introduction

1.1 What Is Bright Cluster Manager?

Bright Cluster Manager is a cluster management application built on top of a major Linux distribution. Bright Cluster Manager 5.1 is available on:

- Scientific Linux 5 (`x86_64` only)
- RedHat Enterprise Linux Server 5 (`x86_64` only)
- CentOS 5 (`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*, often also referred to as *slave* 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.

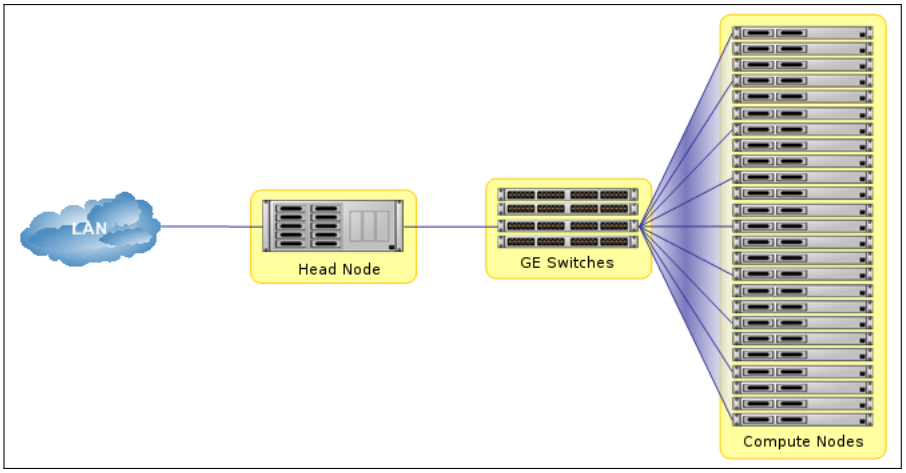

Figure 1.1: Cluster network

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:

Head Node

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

Compute Nodes

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

2.2 Supported Hardware

The following hardware is supported:

Compute Nodes

- SuperMicro
- Cray
- Dell
- IBM

- Asus
- Hewlett Packard

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

Ethernet Switches

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

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

Power Distribution Units

- APC (American Power Conversion) Switched Rack PDU

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

Management Controllers

- IPMI 1.5/2.0
- HP iLO 1/2

InfiniBand

- Most InfiniBand HCAs

2.3 Head Node Installation

This section describes the steps in installing a Bright Cluster Manager head node. To start the install, the head node is booted from the Bright Cluster Manager DVD.

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.

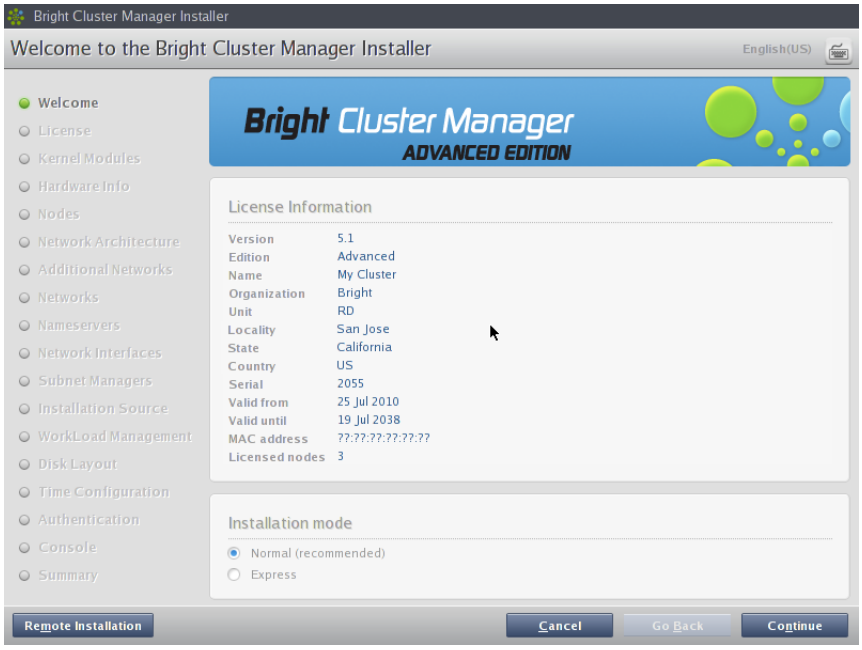

Figure 2.1: Installation welcome screen for Bright Cluster Manager

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 2.23), 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.

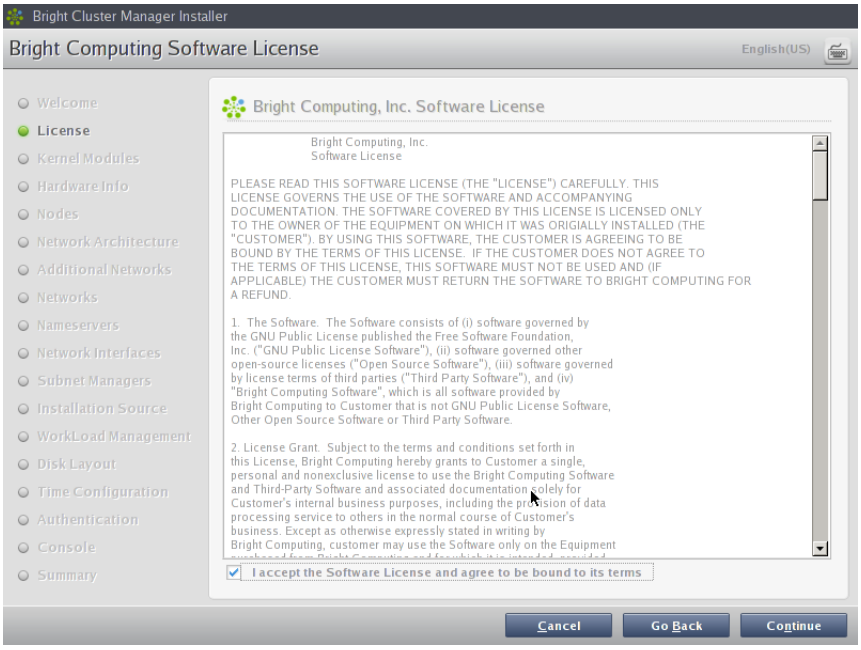

Figure 2.2: Bright Cluster Manager Software License

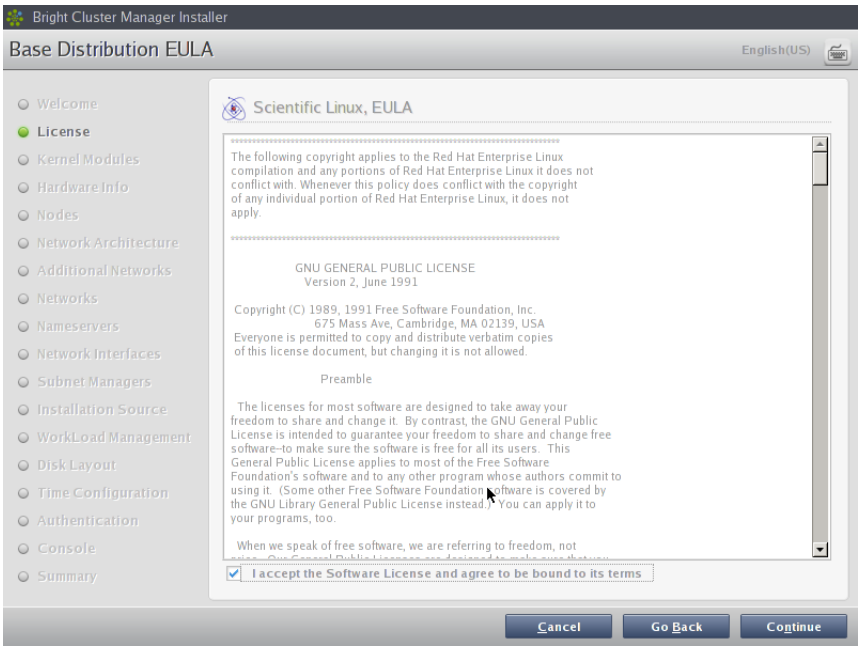

Figure 2.3: Base Distribution End User License Agreement

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.

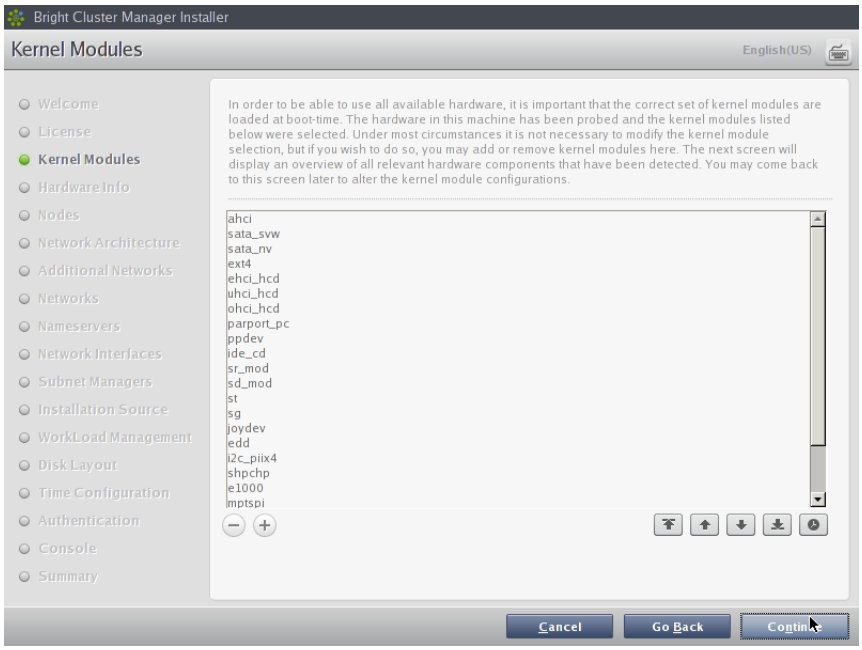

Figure 2.4: Kernel Modules Recommended For Loading After Probing

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.

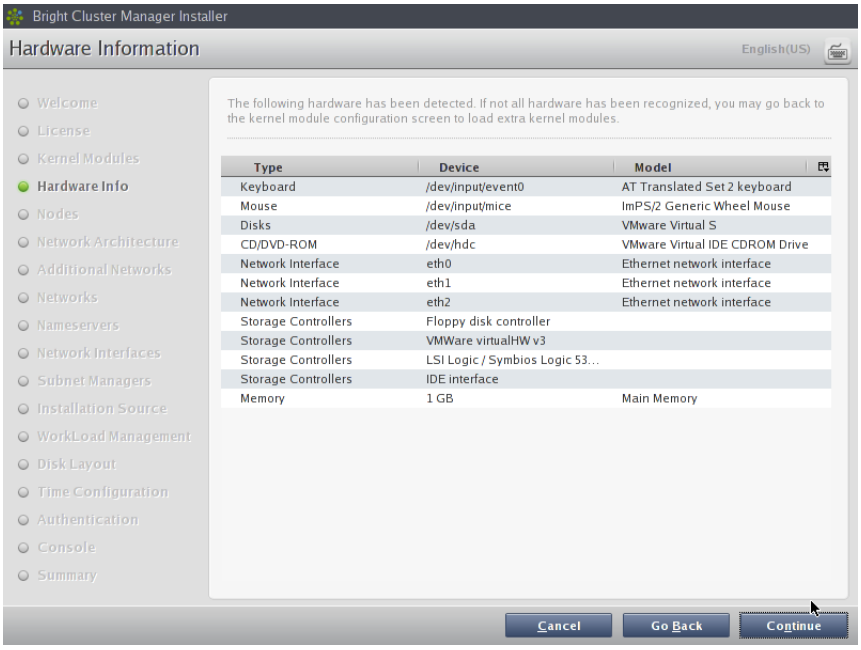

Figure 2.5: Hardware Overview Based On Loaded Kernel Modules

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 Architecture` selection screen, described next.

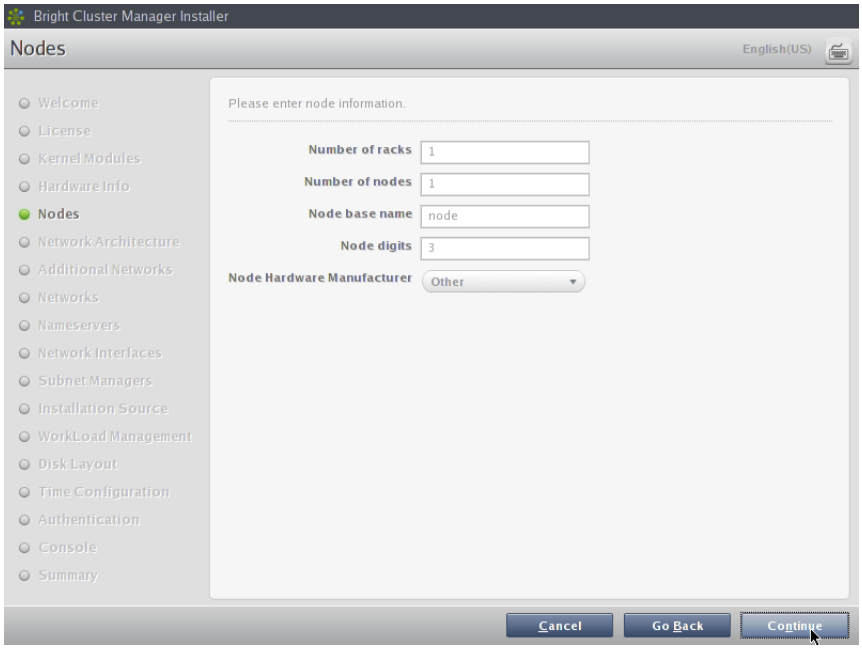

Figure 2.6: Nodes Configuration

Network Architecture

The `Network Architecture` screen allows selection of one of three different network architecture setups.

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 architecture 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.

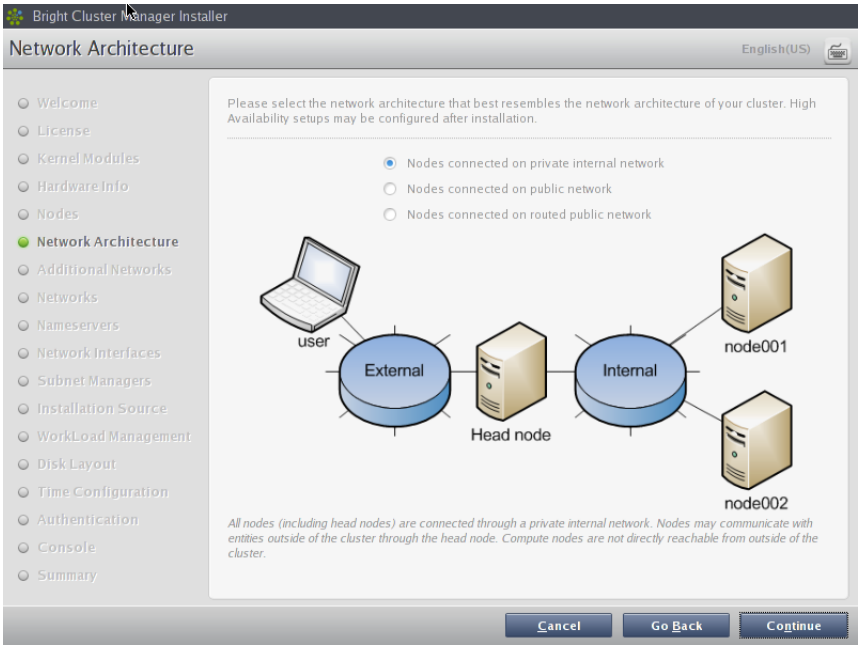

Figure 2.7: Networks Architecture: nodes connected on a private internal network

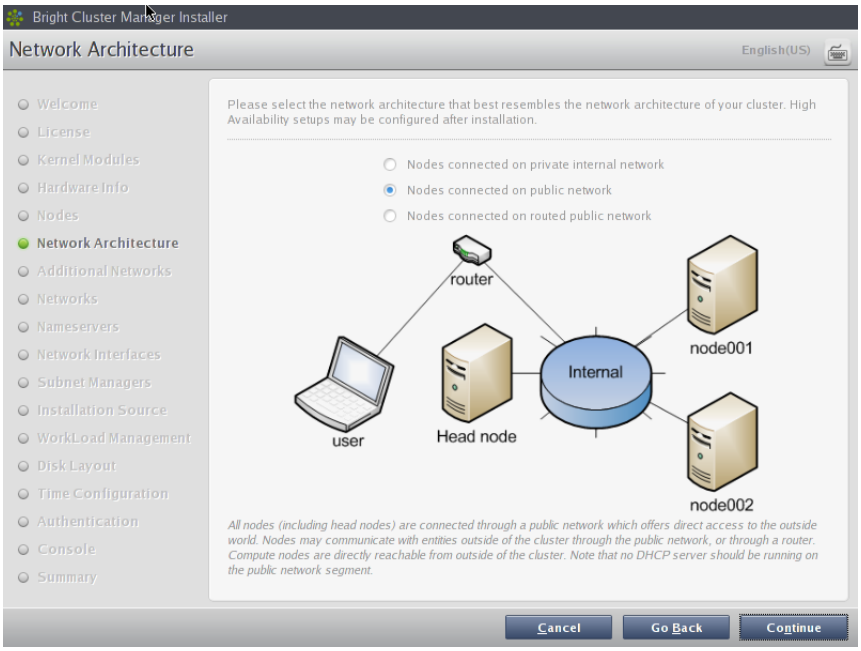

Figure 2.8: Networks Architecture: nodes connected on a public network

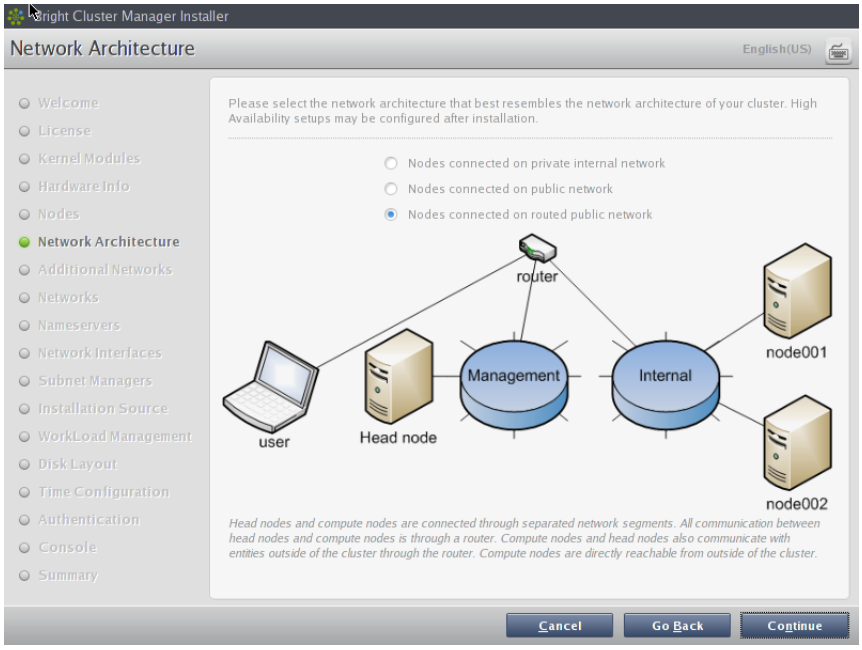

Figure 2.9: Network Architecture: nodes connected on a routed public network

Additional Network Configuration

The `Additional Network Configuration` screen (Figure 2.10) allows InfiniBand and IPMI/iLO networks to be configured. Clicking `Continue` here leads to the `Networks` configuration screen, described next.

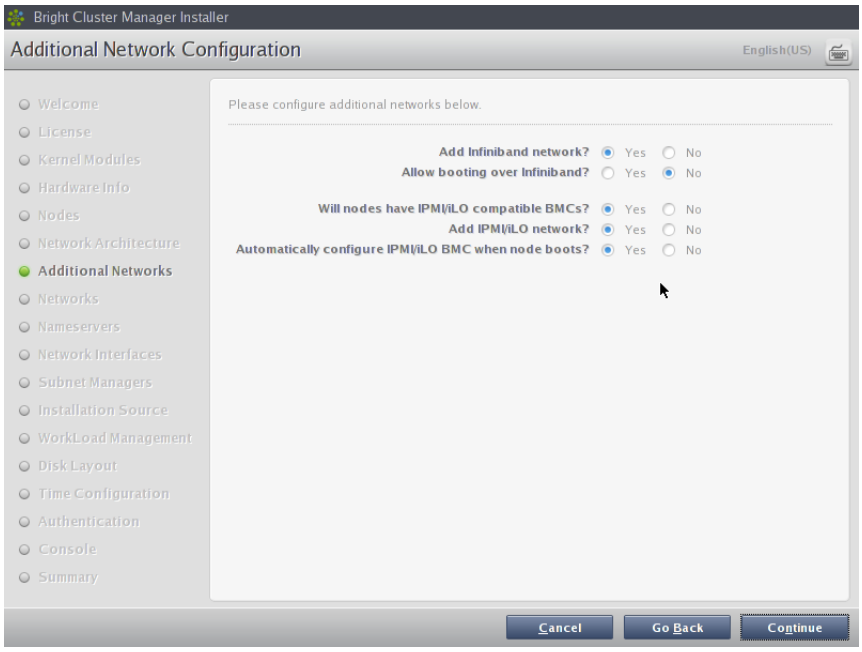

Figure 2.10: Additional Network Configuration

Networks Configuration

The `Networks` configuration screen (Figure 2.11) displays the predefined list of networks, based on the selected network architecture. 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.

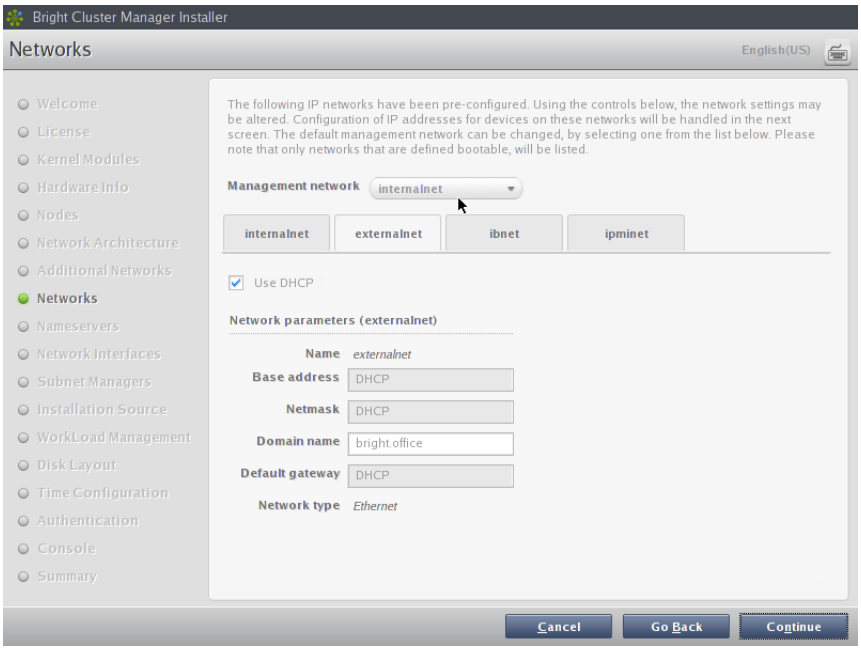

Figure 2.11: Networks Configuration

Nameservers and search domains

Nameservers and search domains can be added or removed using the `Nameservers` screen (Figure 2.12). Clicking on `Continue` leads to the `Network Interfaces` configuration screen, described next.

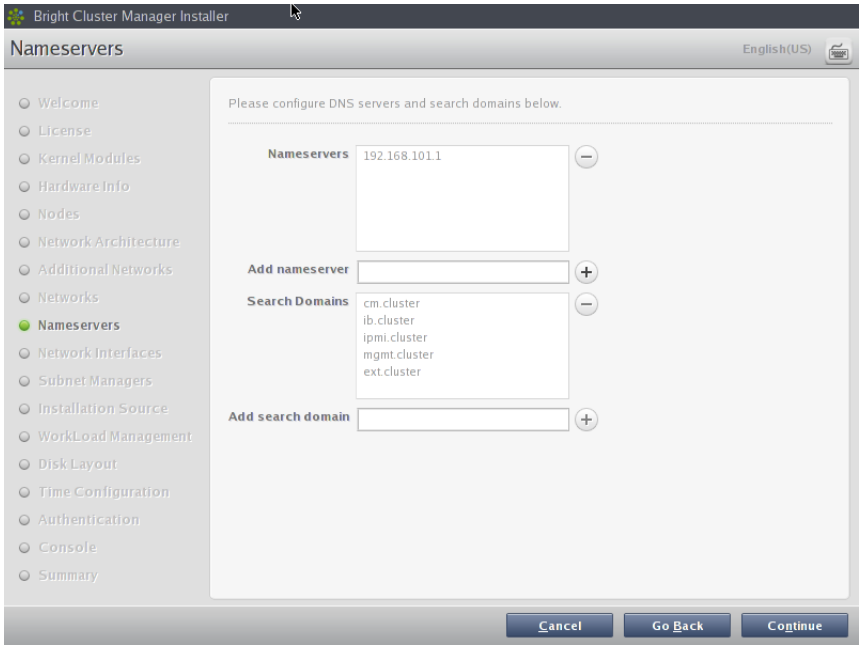

Figure 2.12: Nameservers and search domains

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 dropdown 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` on this screen leads to the `CD/DVD ROMs` selection screen (Figure 2.16).

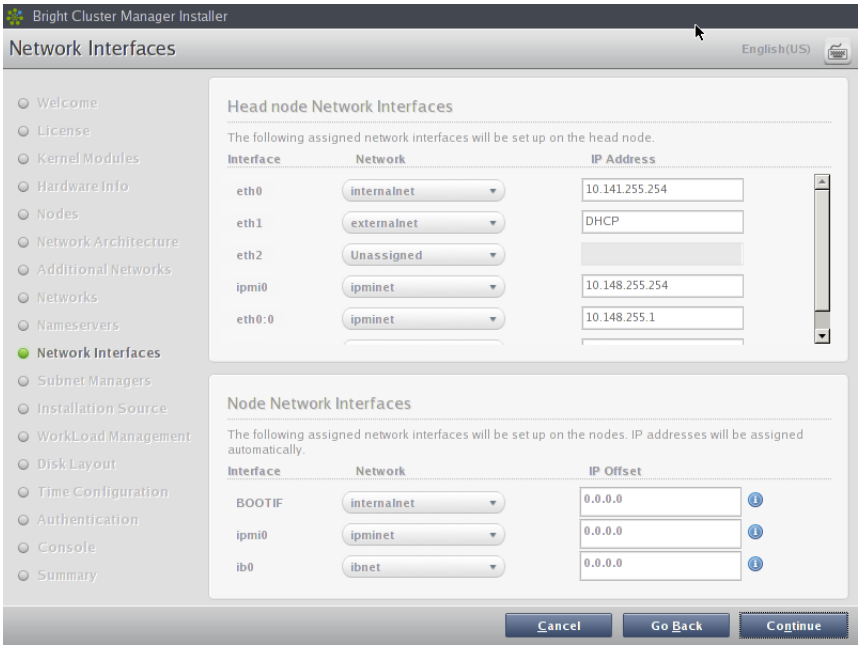

Figure 2.13: Network Interface Configuration: *type 1*

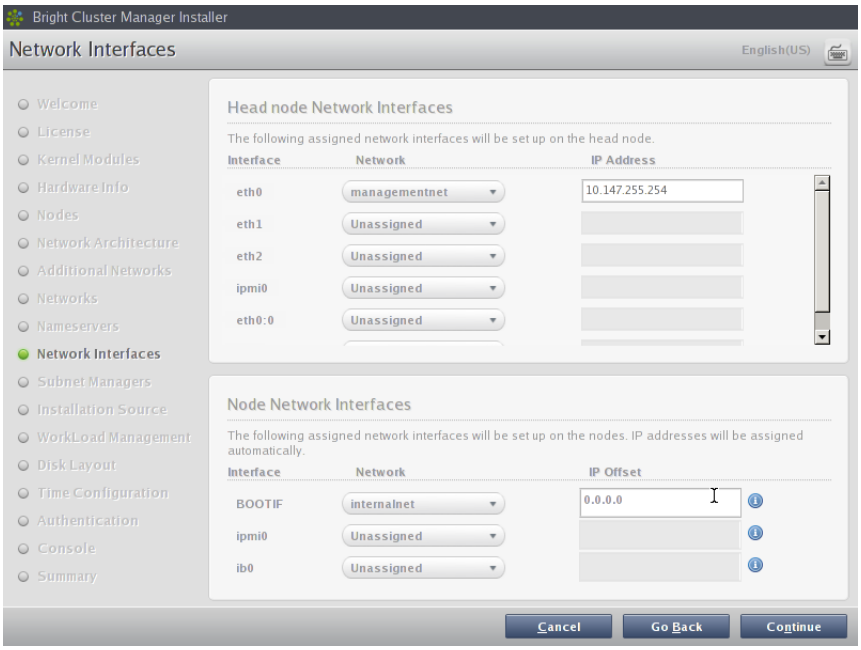

Figure 2.14: Network Interface Configuration: *type 3*

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.

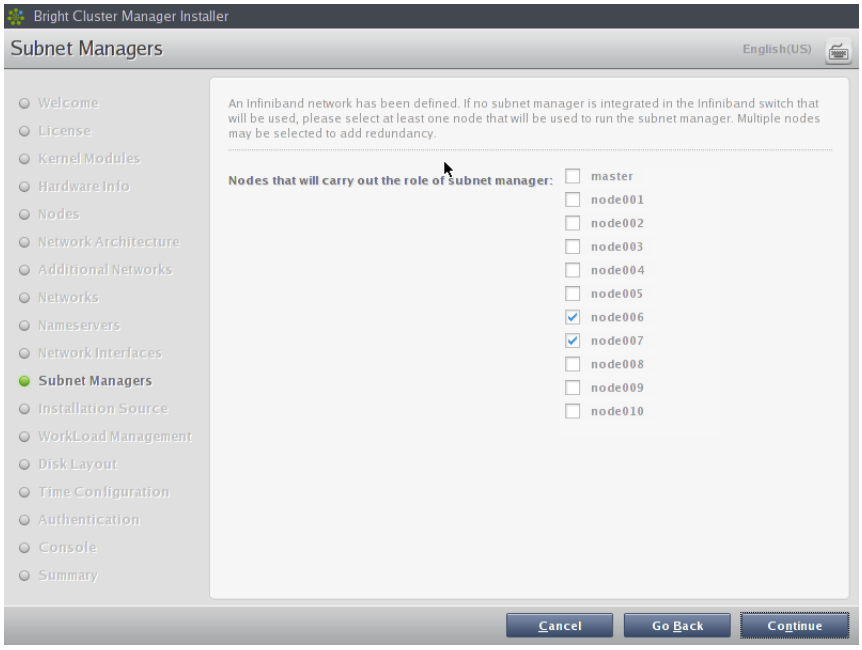

Figure 2.15: Subnet Manager Nodes

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.

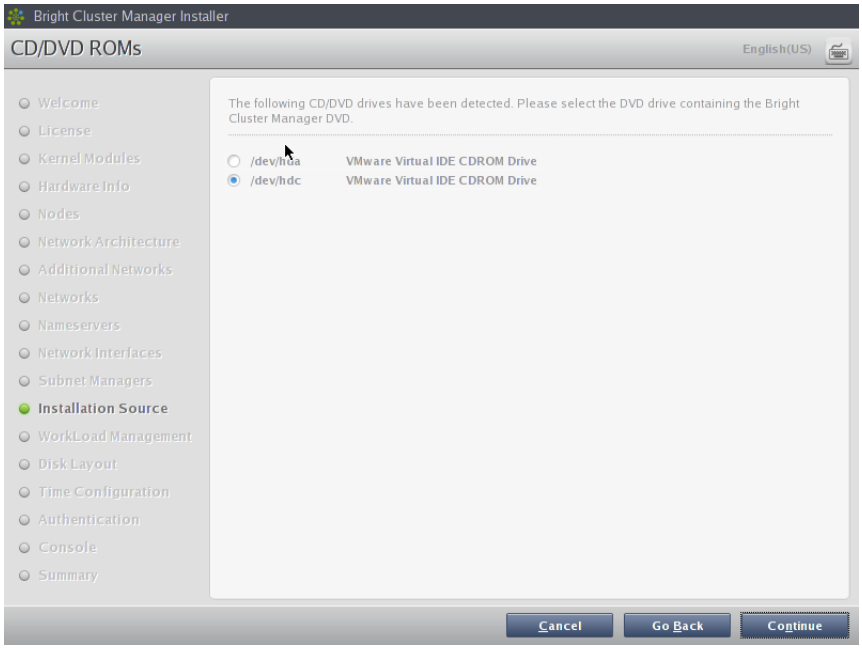

Figure 2.16: DVD Selection

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.

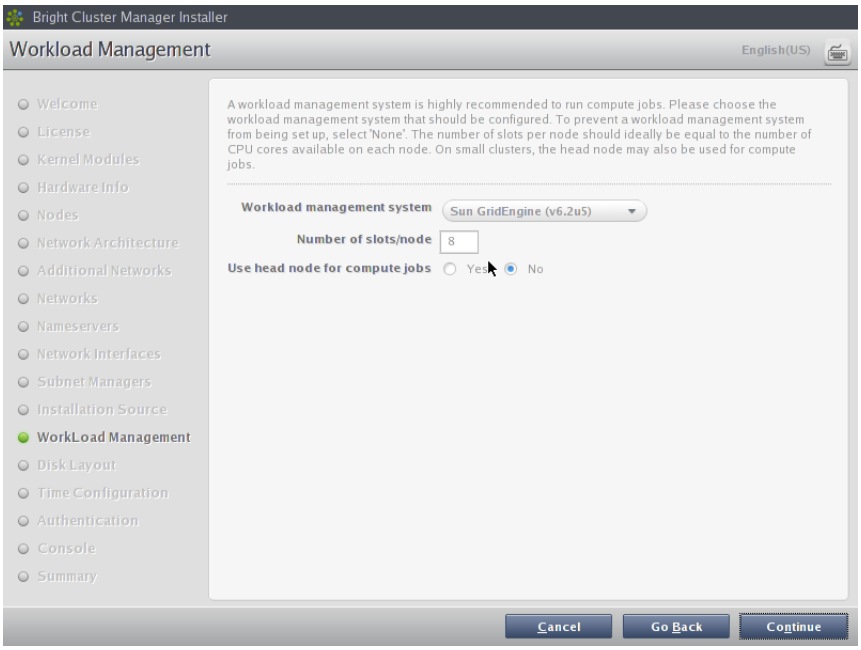

Figure 2.17: Workload Management Setup

Disk Partitioning and Layouts

The `Disk Partitioning and Layouts` configuration screen (Figure 2.18) consists of two options: `Head node disk layout` and `Node disk layout`.

For each option, a partitioning layout other than the default can be chosen by selecting it from the drop-down boxes. This will then be used for installation.

Also, for each option, a text editor box opens up when an option's edit button is clicked (Figure 2.19), and is useful for viewing and changing values. The `Save` and `Reset` buttons are enabled on editing, and will save or undo the text editor changes that were made. Once saved, the changes cannot be reverted.

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

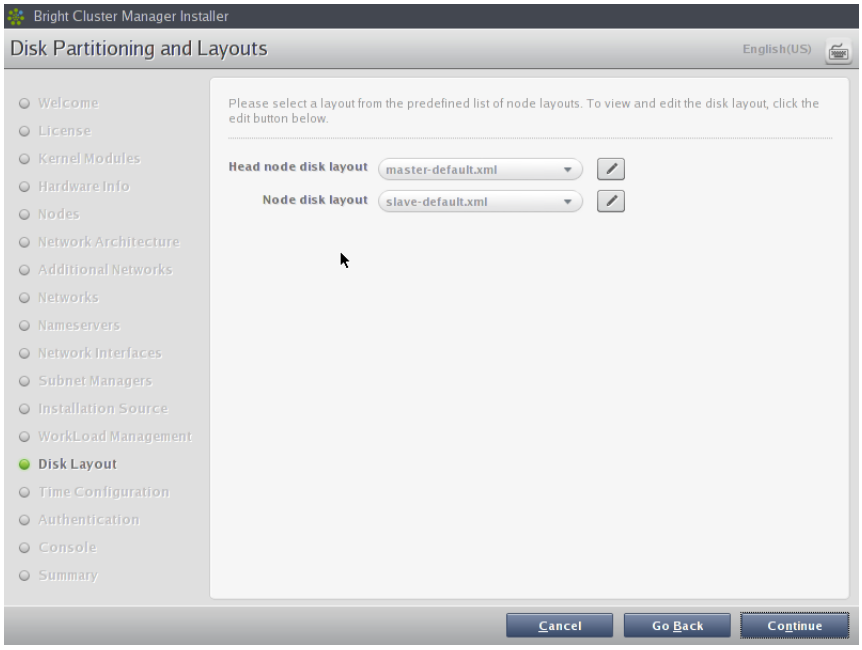

Figure 2.18: Disk Partitioning and Layouts

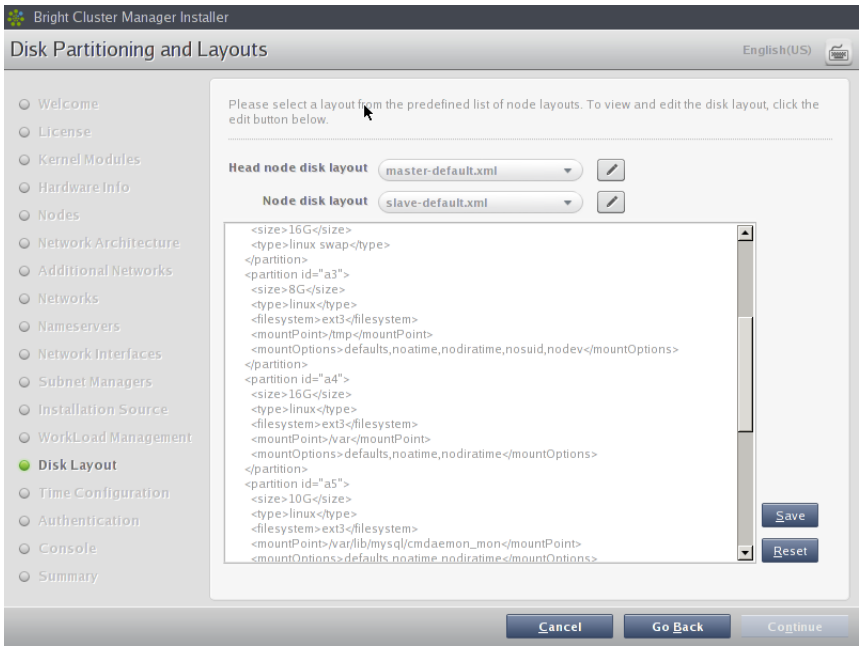

Figure 2.19: Edit Head Node Disk Partitioning

Time Configuration

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

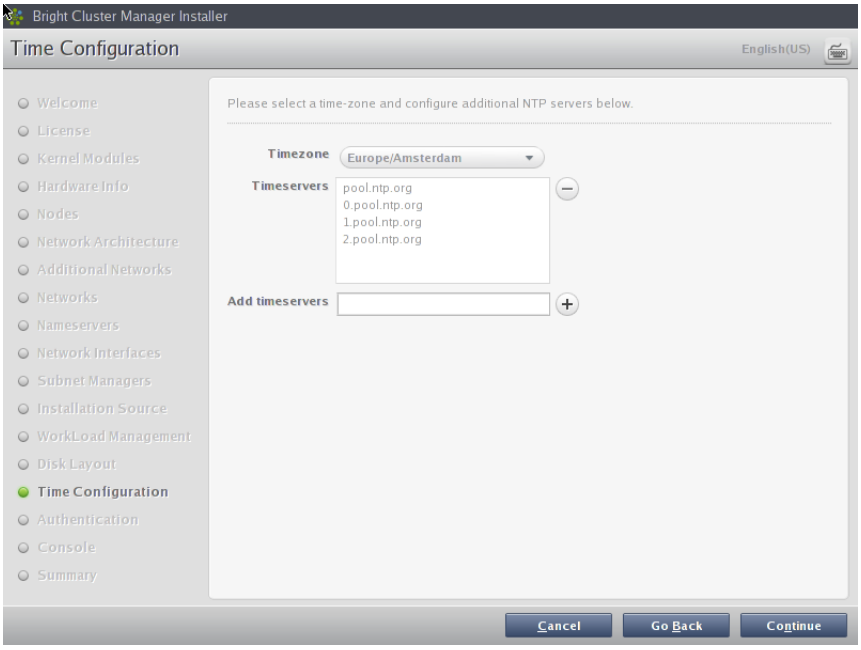

Figure 2.20: Time Configuration

Authentication

The `Authentication` screen (Figure 2.21) requires the password to be set twice for the cluster administrator. The hostname of the head node can also be modified in this screen. Clicking `Continue` validates the passwords that have been entered, and if successful, leads to the `Console` screen, described next.

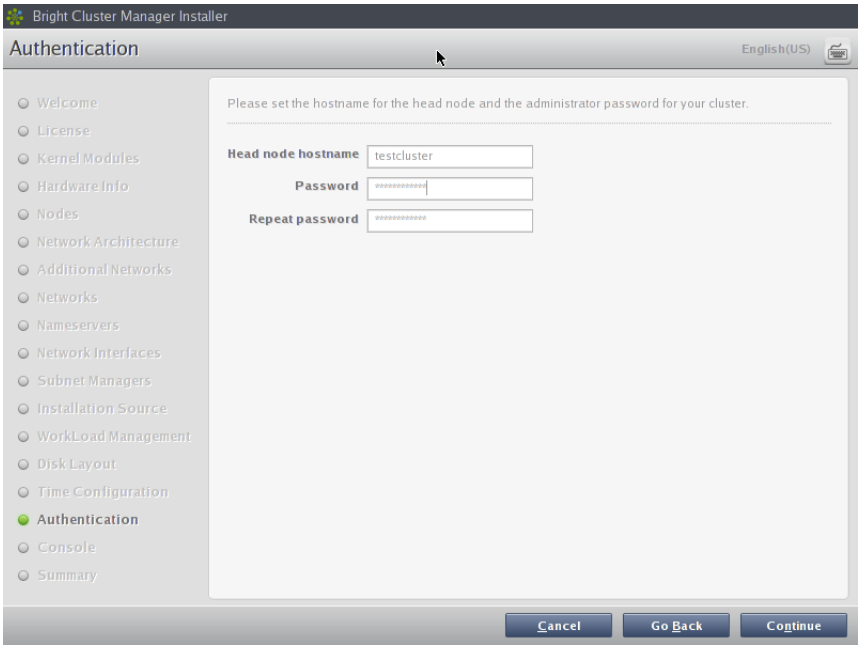

Figure 2.21: Authentication

Console

The Console screen (Figure 2.22) allows selection of a graphical mode or a text console mode for when the head node or ordinary nodes boot. Clicking Continue leads to the Summary screen, described next.

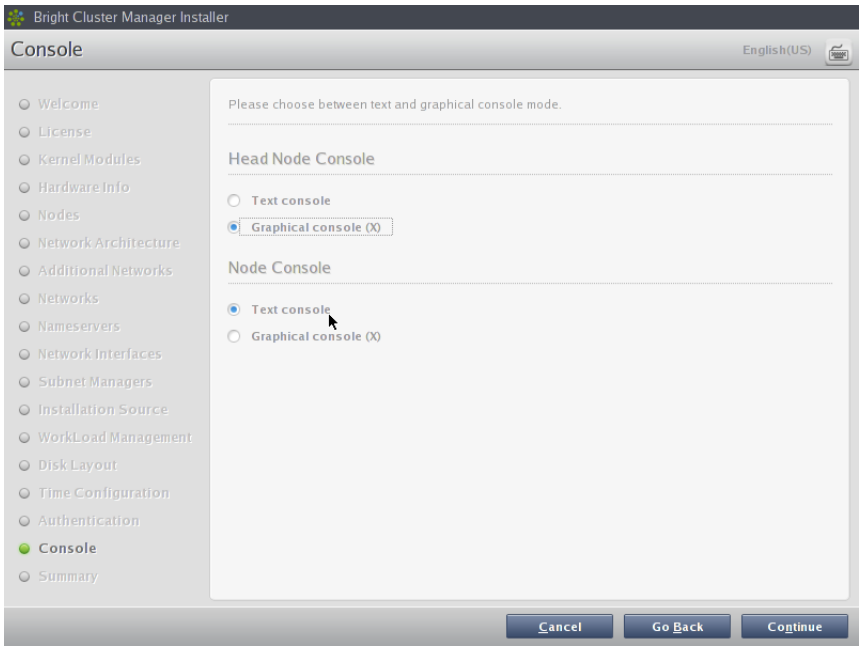

Figure 2.22: Console

Summary

The Summary screen (Figure 2.23), summarizes 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.

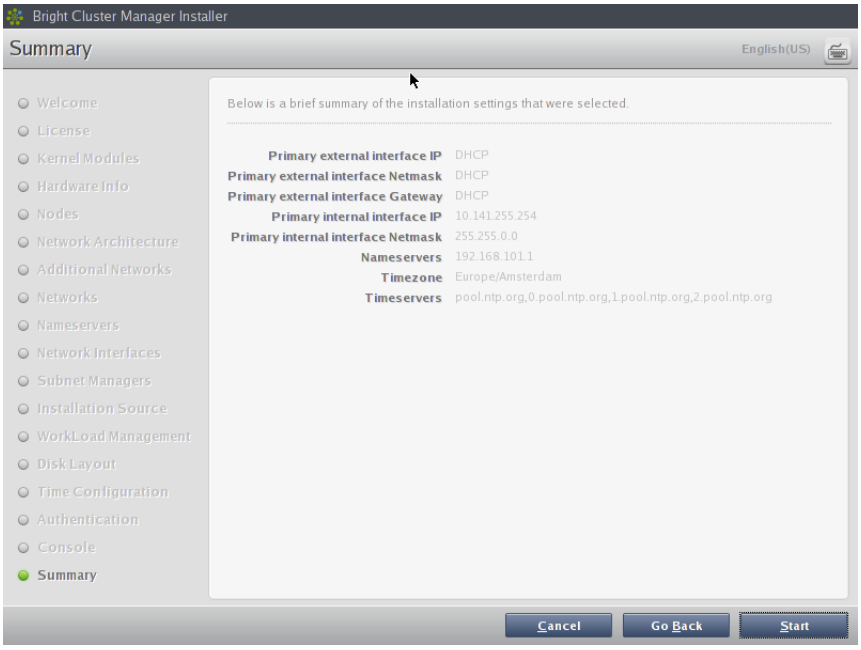

Figure 2.23: Summary of Installation Settings

Installation

The `Installation Progress` screen (Figure 2.24) 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 2.25), 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.

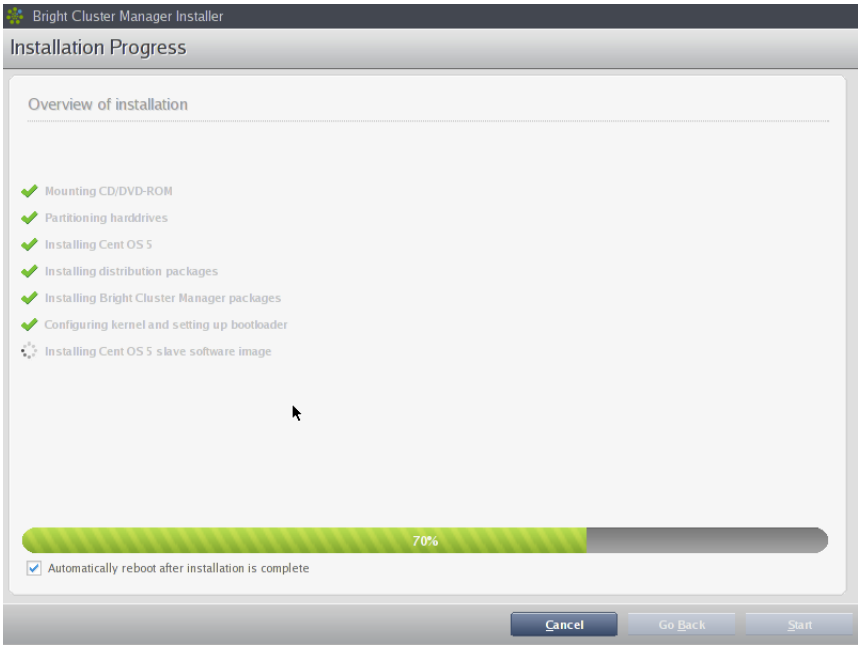

Figure 2.24: Installation Progress

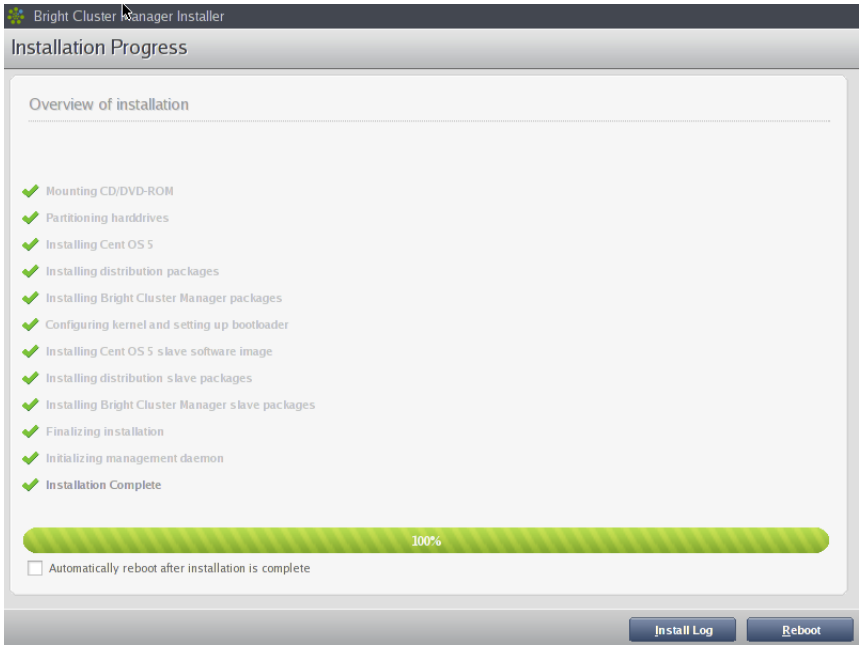

Figure 2.25: Installation Completed

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 3.6 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 3.7 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 a physical hardware component that is part of a cluster. A device can be any of the following types:

- Head Node
- Node
- Graphics Processing 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 Chapter 6 on node provisioning.

`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 an ordinary node. In practice, a software image is a directory on the head node containing a full Linux file-system. When an ordinary node boots, the node provisioning system sets up the node with a copy of the software 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.

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 performing package management can be found in chapter 9.

3.1.3 Node Categories

A *node category* is a group of ordinary nodes that share the same configuration. Node categories exist to allow an administrator to configure a large group of nodes at once. In addition, it is frequently convenient to perform certain operations (e.g. a reboot) on a number of nodes at a time.

A node is in exactly one category at all times, and is by default in the slave category.

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 (e.g. monitoring setup) for these nodes will differ from the rest of the nodes.

One of the parameters of a node category is the software image that is to be used for all of the nodes inside the category. 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.

Example

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

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 ordinary 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 login 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 per-node parameters to be set that influence the behavior of the role. For example, the `SGEClient` role (which turns a node into an Sun Grid Engine client) uses parameters to control how the node is configured within SGE in terms of queues and the number of queue slots.

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.

3.2 Modules Environment

The *modules environment* 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 that are currently installed are displayed by running:

```
module list
```

The modules available for loading are displayed by running:

```
module avail
```

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

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

Example

Here is how to load the environment using version 3.2 of the Pathscale compiler and version 1.2.7 of MPICH for Gigabit Ethernet, assuming these are already installed on the system. An MPI application can then be compiled with this environment:

```
module add shared
module add pathscale/3.2
module add mpich/ge/psc/64/1.2.7
mpicc -o myapp myapp.c
```

Note that specifying version numbers explicitly is typically only necessary when multiple versions of an application have been installed. 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.

The `shared` module is 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. The `shared` module is therefore not loaded by default for `root`.

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 loaded automatically with `module initadd` at login, using the full path specification.

More details on the modules environment from an administrator's perspective are given in section 12.1, or in the manual page (`man (1) module`).

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 node 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 6.3 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 3.7):

```
[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 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, more 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 Linux, `cmgui` is installed by untarring the `tar.bz2` file, and compiling it using `make`. A number of dependency packages (as listed in the accompanying README) may first have to be installed for `make` to complete successfully. After a successful `make`, the `cmgui` script can be run from the `cmgui` directory:

Example

```
user@desktop:~> tar -xjf cmgui-5.1-r2174-src.tar.bz2
user@desktop:~> cd cmgui-5.1-r2174
user@desktop:~/cmgui-5.1-r2174> make
[...]
user@desktop:~/cmgui-5.1-r2174> cd cmgui
user@desktop:~/cmgui-5.1-r2174/cmgui> ./cmgui
```

If the `cmgui` script reports unresolved symbols, then additional packages from the Linux distribution need to be installed, as listed in the accompanying README, 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
```

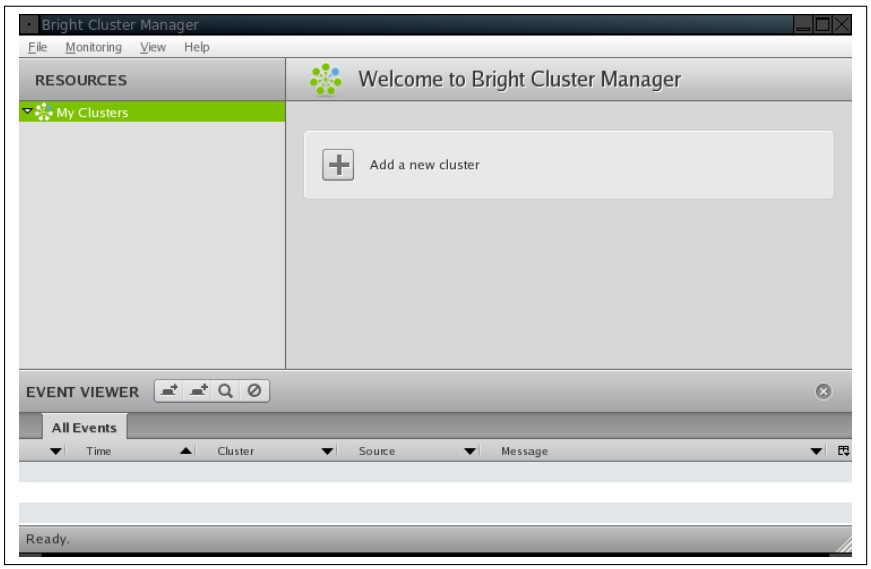

Figure 3.1: Cluster Management GUI welcome screen

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.

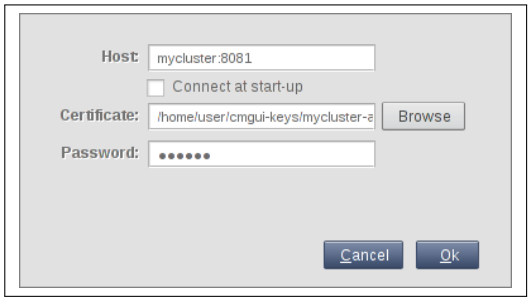

Figure 3.2: Edit Cluster dialog window

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).

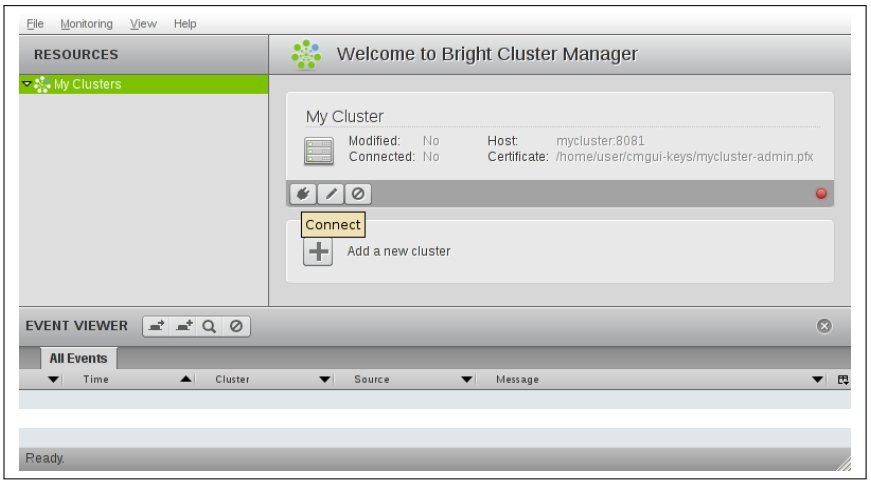

Figure 3.3: Connecting to a cluster

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

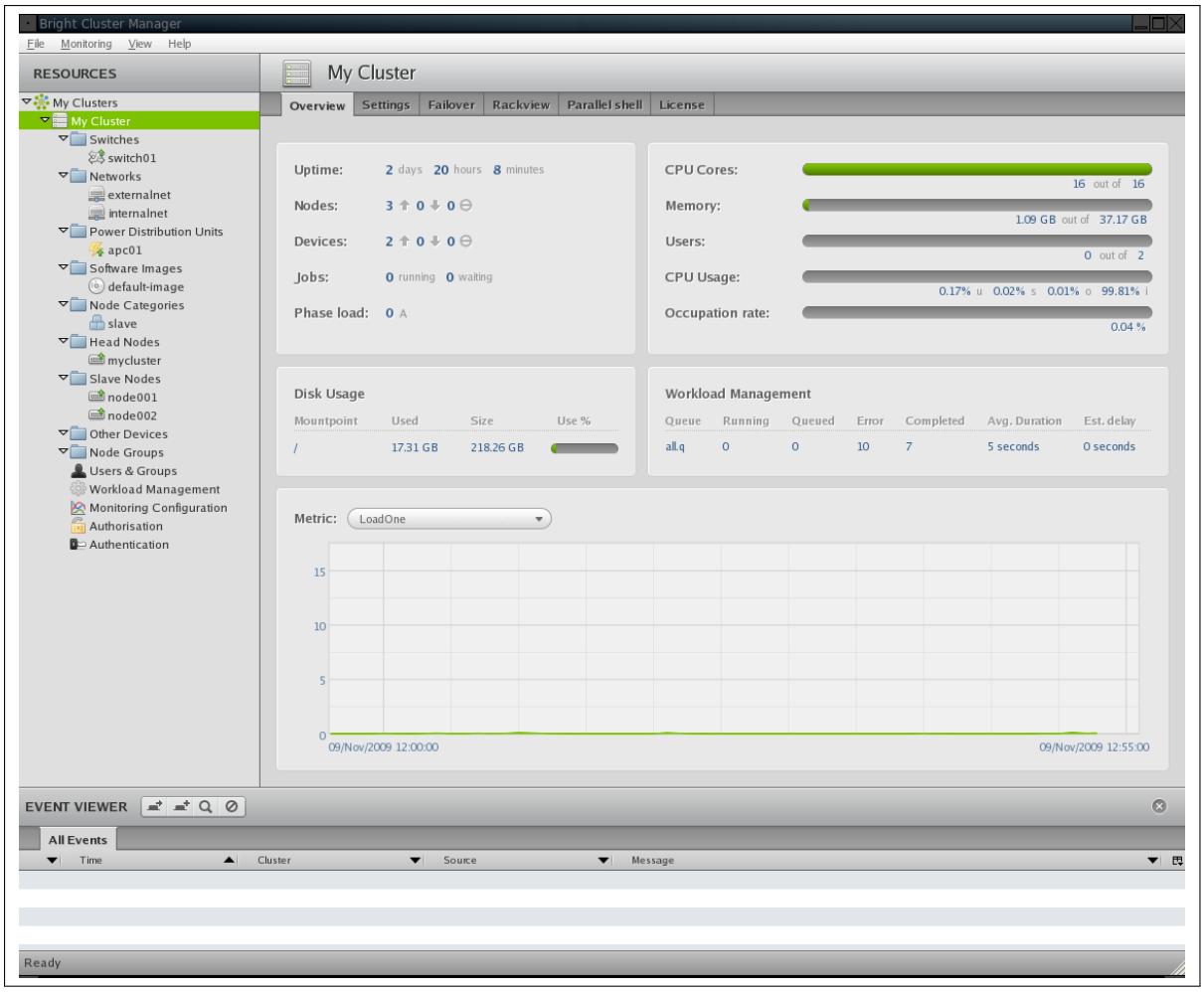

Figure 3.4: Cluster Overview

3.5 Navigating the Cluster Management GUI

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

The resource tree on the left side of the screen 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 that allows it to be 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.

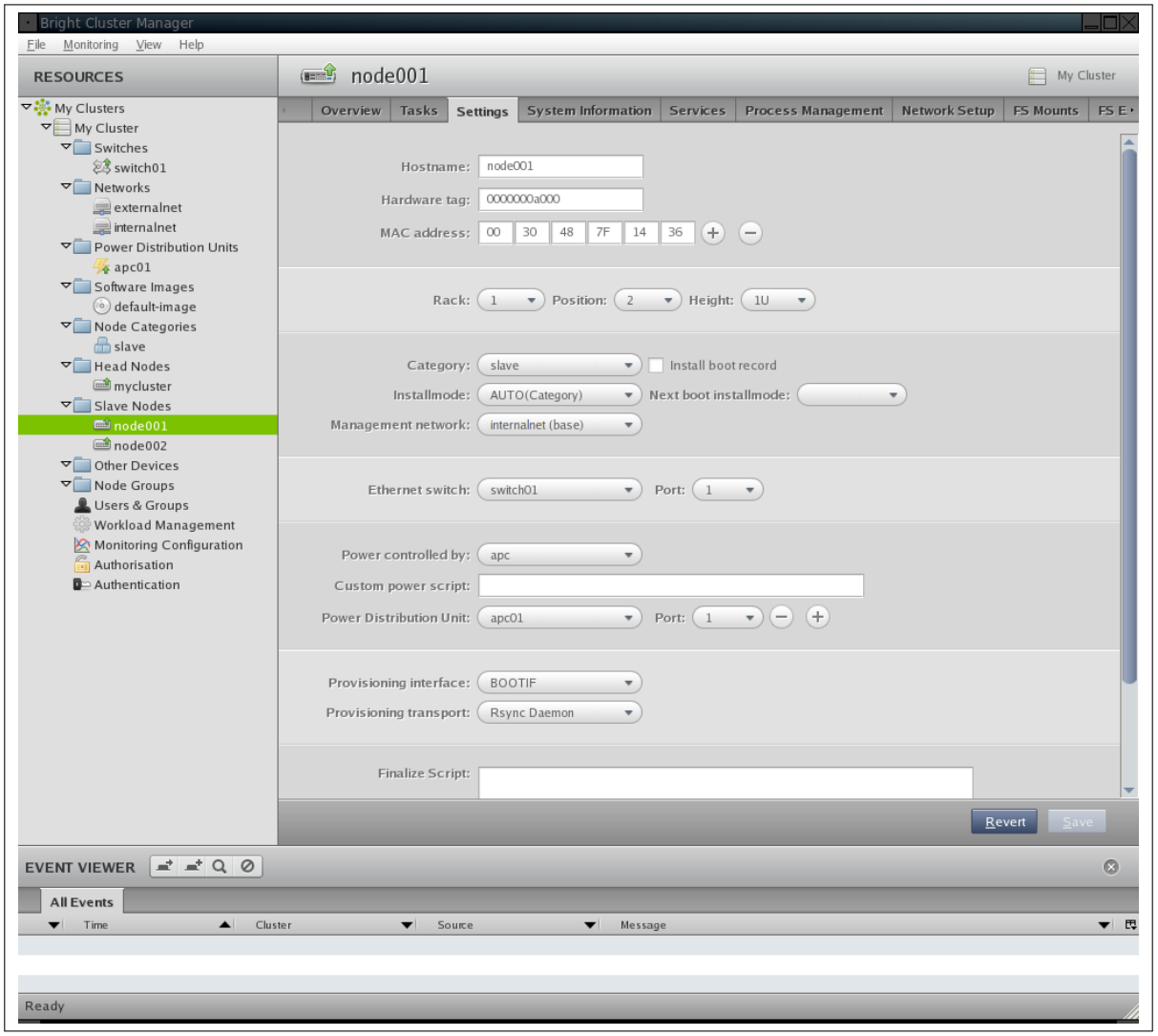

Figure 3.5: Node Settings

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.

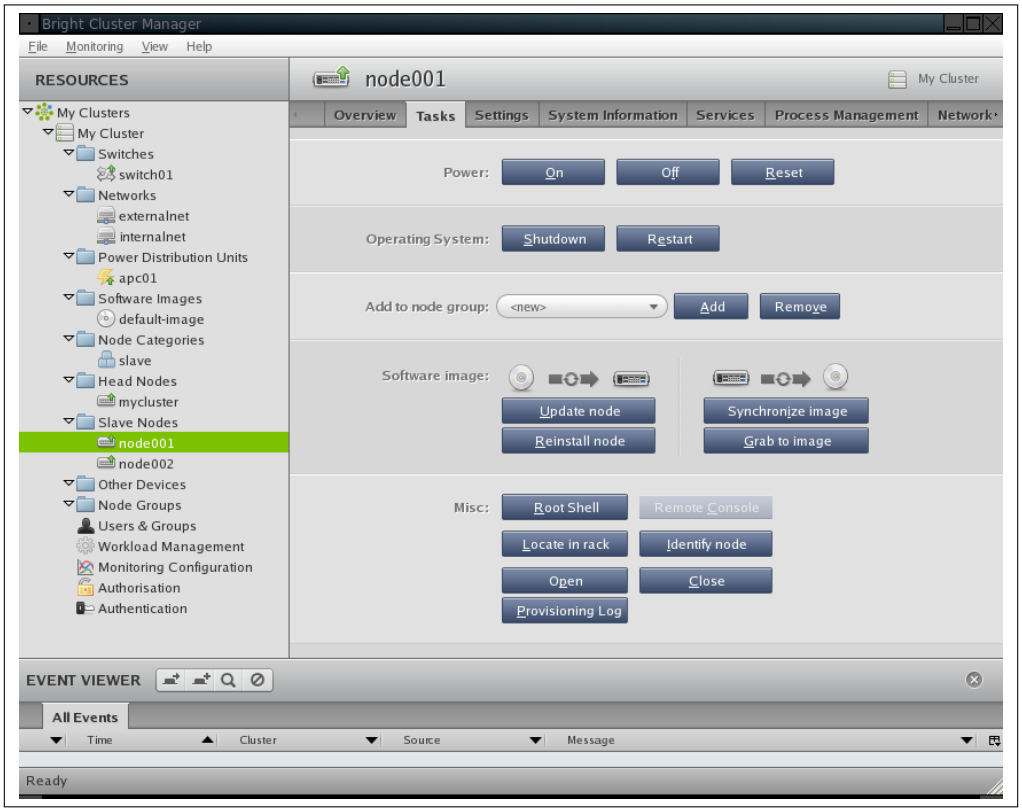

Figure 3.6: Node Tasks

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`, `Slave Nodes`, and `Networks`. Selecting a resource folder in the tree 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 the `Slave Nodes` resource, 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).

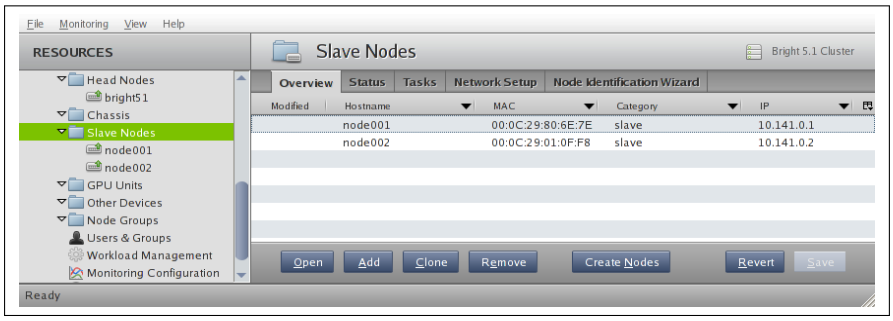

Figure 3.7: Nodes Overview

3.6 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.

3.6.1 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
       --nossl|-s .................... Do not use SSL
       --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

3.6.2 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:

```
connect ..................... Connect to cluster
disconnect .................. Disconnect from cluster
alias ....................... Set aliases
unalias ..................... Unset aliases
exit ........................ Exit from current object or mode
quit ........................ Quit shell
export ...................... Display list of aliases current list formats
help ........................ Display this help
history ..................... Display command history
list ........................ List state for all modes
modified .................... List modified objects
refresh ..................... Refresh all modes
events ...................... Manage events
run ......................... Execute cmsh commands from specified file
category .................... Enter category mode
cert ........................ Enter cert mode
device ...................... Enter device mode
jobqueue .................... Enter jobqueue mode
jobs ........................ Enter jobs mode
main ........................ Enter main mode
monitoring .................. Enter monitoring mode
network ..................... Enter network mode
nodegroup ................... Enter nodegroup mode
partition ................... Enter partition mode
process ..................... Enter process mode
profile ..................... Enter profile mode
session ..................... Enter session mode
softwareimage ............... Enter softwareimage mode
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
Usage: run [--echo|-x] [--quit|-q] <filename> [<filename2> ...]
           Execute all commands in the given file(s)

       --echo|-x ..................... Echo all commands
       --quit|-q ..................... 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 above 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, the `exit` command is used:

Example

```
[mycluster]% device
[mycluster->device]% list
Type           Hostname        MAC                Ip
-------------- --------------- ------------------ ---------------
EthernetSwitch switch01        00:00:00:00:00:00  10.142.253.1
MasterNode     mycluster       00:E0:81:34:9B:48  10.142.255.254
PowerDistribu+ apc01           00:00:00:00:00:00  10.142.254.1
SlaveNode      node001         00:E0:81:2E:F7:96  10.142.0.1
SlaveNode      node002         00:30:48:5D:8B:C6  10.142.0.2
[mycluster->device]% exit
[mycluster]%
```

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*s and *arg*s are optional.

Example

```
[mycluster->network]% device status node001; list
node001 ..... [   UP   ]
Name         Netmask bits  Base address     Broadcast address
------------ ------------- ---------------- -----------------
externalnet  29            195.73.194.136   195.73.194.143
internalnet  16            10.142.0.0       10.142.255.255
[mycluster->network]%
```

In the above example, while in `network` mode, the `status` command is executed in `device` mode and passed the argument `node001`, making it display the status of the `node001` device. The `list` command on the same line, after the semi-colon, runs as expected in `network` mode to display a list of network objects.

3.6.3 Working With Objects

Modes in `cmsh` work with associated *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:

Command	Description
`use`	Use the specified object. I.e.: Make the specified object the *current object*
`add`	Create the object and use it
`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
`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 specified property of the object. If no property is specified, clear every value of the object
`append`	Append a value to a specific property of the object for a property that can take more than one value at a time
`removefrom`	Remove a value from a specific property of the object, for a property that can take more than one value at a time
`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 above 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 slavenode node100 10.141.0.100
[mycluster->device*[node100*]]% category add test-slave
[mycluster->category*[test-slave*]]% device; use node100
[mycluster->device*[node100*]]% set category test-slave
[mycluster->device*[node100*]]% commit
[mycluster->device[node100]]% exit
[mycluster->device]%
```

In the above, within `device` mode, a new object `node100` is added, of type `slavenode`, and with IP address `10.141.0.100`. The `category` test-slave is then added, and the test-slave 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 the device node object still needs a property under it to be set. To return to device mode again, at the level it was left, a multiple command "`device; use node100`" is executed. The category property of the the `node100` object is set to the newly created category `test-slave` and the object is then committed to store it permanently. Note that 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 `slavenode`, 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
Usage: add <type> <hostname>
           Create a new device of the given type with specified hostname
       add <type> <hostname> <ip>
           Create a new device of the given type with specified hostname and
           boot interface with given ip

       type
            slavenode, 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
[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 above 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 above example, `node101` is assigned IP address `10.141.0.101`.

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

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-slave
[mycluster->device[node101]]% set category slave
[mycluster->device*[node101*]]% get category
slave
[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-slave
[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.

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

```
[mycluster->device]% list -f hostname:14,ethernetswitch:15,ip
hostname (key) ethernetswitch  ip
-------------- --------------- --------------------
apc01                          10.142.254.1
mycluster      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
[mycluster->device]%
```

Without an argument, the default format string for the mode is used. To display the default format string, the format command without parameters is used. Invoking the format command without arguments also displays all available properties including a description. To change the default format string, the desired format string can be passed as an argument to format.

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 slave
[mycluster->device*[node001*]]% validate
All good
[mycluster->device*[node001*]]% commit
[mycluster->device[node001]]%
```

3.6.4 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 can not 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              slave
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
```

```
Rack setup                      1 racks of 42 high
Search domains                  clustervision.com
Slave digits                    3
Slave name                      node
Time servers                    pool.ntp.org
Time zone                       America/Los_Angeles
```

3.6.5 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)           Ip
-------------- --------------      ------------------- -----------
SlaveNode      node001             00:E0:81:2E:F7:96    10.142.0.1
[mycluster]%
```

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

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]%
```

3.7 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.

CMDaemon is an application that is started automatically when any node boots and will continue running until the node is shut down. Should CMDaemon be stopped manually for whatever reason, its cluster management functionality will no longer be available, making it hard for administrators to manage the cluster. However, even with the daemon stopped, the cluster will remain fully usable for running computational jobs.

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 (see 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.

3.7.1 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 in `/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 to terminate...
Stopping CMDaemon:                                         [  OK  ]
Waiting for CMDaemon to start...
Starting CMDaemon:                                         [  OK  ]
[root@mycluster ~]#
```

3.7.2 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 ordinary 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.

3.7.3 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 manually!
# 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" }
```

4

Configuring The Cluster

After the 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. More elaborate aspects of cluster configuration such as power management and workload management will be 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. For example, 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. Another example: the maximum number of nodes is an attribute in the license file that specifies the condition that no more than the specified number of nodes may be used by the software.

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
```

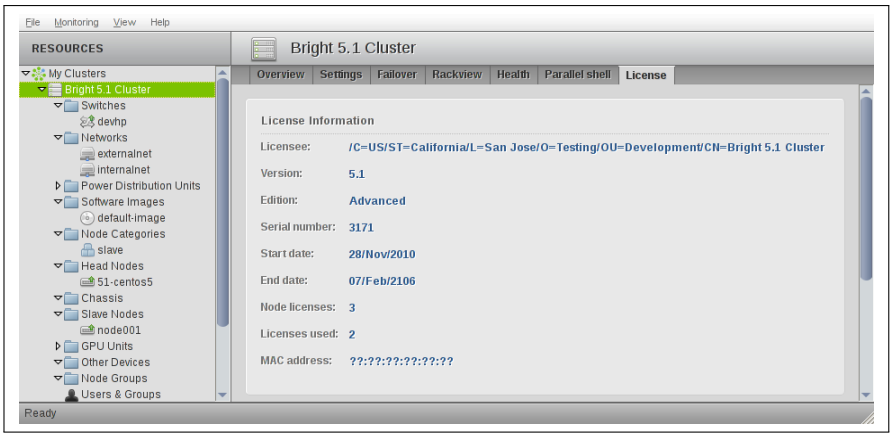

Figure 4.1: License Information

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). Alternatively the `licenseinfo` in `cmsh` `main` mode may be used:

Example

```
[root@51-centos5 ~]# cmsh
[51-centos5]% main licenseinfo
License Information
----------------------- ----------------------------------------------------
Licensee                /C=US/ST=California/L=San Jose/O=Testing/OU=Developm
                        ent/CN=Bright 5.1 Cluster
Serial Number           3171
Start Time              Sun Nov 28 00:00:00 2010
End Time                Tue Nov  2 23:59:59 2038
Version                 5.1
Edition                 Advanced
Licensed Nodes          3
Node Count              2
MAC Address             ??:??:??:??:??:??
[51-centos5]%
```

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@myheadnode ~]# /etc/init.d/cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
```

```
[root@myheadnode ~]# tail -1 /var/log/cmdaemon
Dec 30 15:57:02 myheadnode 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@myheadnode ~]# verify-license
Usage: verify-license <path to certificate> <path to keyfile> <verify|info>
[root@myheadnode ~]# cd /cm/local/apps/cmd/etc/
[root@myheadnode etc]# verify-license cert.pem cert.key info
========= Certificate Information ========
Version:             5.1
Edition:             Advanced
Common name:         Bright 5.1 Cluster
Organization:        Bright Computing
Organizational unit: Development
Locality:            San Jose
State:               California
Country:             US
Serial:              2603
Starting date:       29 Jun 2010
Expiration date:     29 Nov 2010
MAC address:         ??:??:??:??:??:??
Licensed nodes:      3
==========================================
[root@myheadnode 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@myheadnode etc]# verify-license cert.pem cert.key verify
    License has expired
    License verification failed.
    ```
  - If the certificate is not from Bright Computing:
    ```
    [root@myheadnode 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. Although the most convenient way to obtain such a license is with a cluster that is able to access the internet, the request can also be made regardless of cluster connectivity to outside networks, as will be elaborated upon shortly.

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

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.

3. If no web access is available to the administrator, the CSR data that was generated by the `request-license` command 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@mycluster ~]# 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 (bright51) 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 example above:

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

If the prompt was answered with a "`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 that was 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@bright51 ~]# install-license /cm/local/apps/cmd/etc/cert.pem.new
========= Certificate Information ========
Version:              5.1
Edition:              Advanced
Common name:          My Cluster
Organization:         Bright Computing, Inc.
Organizational unit: Development
Locality:             San Jose
State:                California
Country:              US
Serial:               3066
Starting date:        01 Jan 2000
```

```
Expiration date:     31 Dec 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

After the cluster is set up with the correct license, the next configuration step is to define the networks that are present. During the Bright Cluster Manager installation at least two default networks 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.

`externalnet`: the network connecting the cluster to the outside world (typically a corporate or campus network).

4.2.1 Configuring Networks

The `network` mode in `cmsh` gives access to all network-related operations using the standard object commands. See section 3.6.3 for more on `cmsh` modes and working with objects.

In `cmgui`, a network can be configured by selecting the `Networks` item in the resource tree (Figure 4.2).

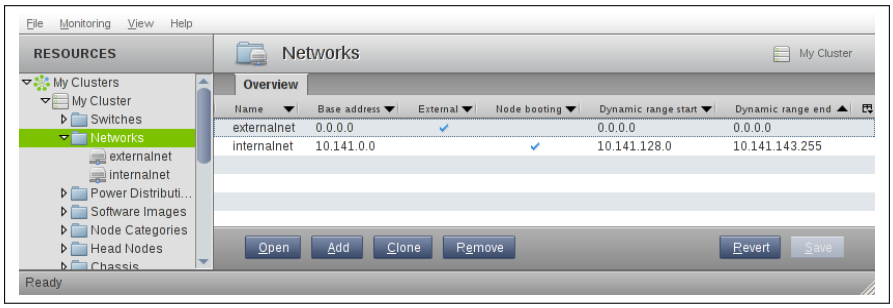

Figure 4.2: Networks

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. an Ethernet segment).

Selecting a network in the resource tree displays its tabbed pane. By default, the tab displayed is the `Overview` tab. This gives a convenient overview of all IP addresses assigned in the selected network (Figure 4.3).

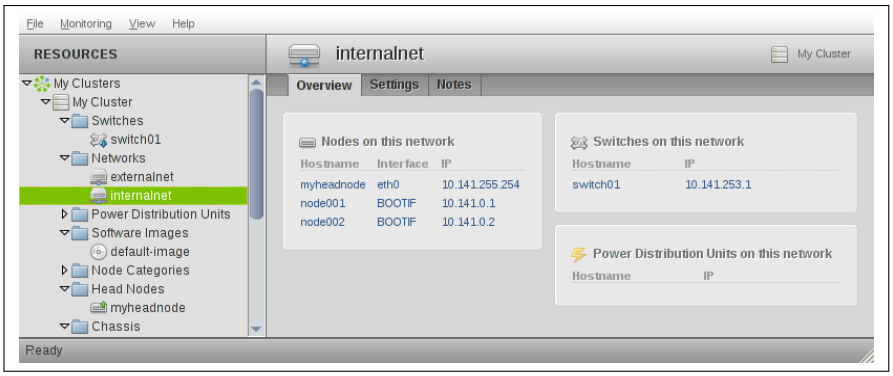

Figure 4.3: Network Overview

Selecting the `Settings` tab (Figure 4.4), allows a number of network properties (Figure 4.5) to be changed.

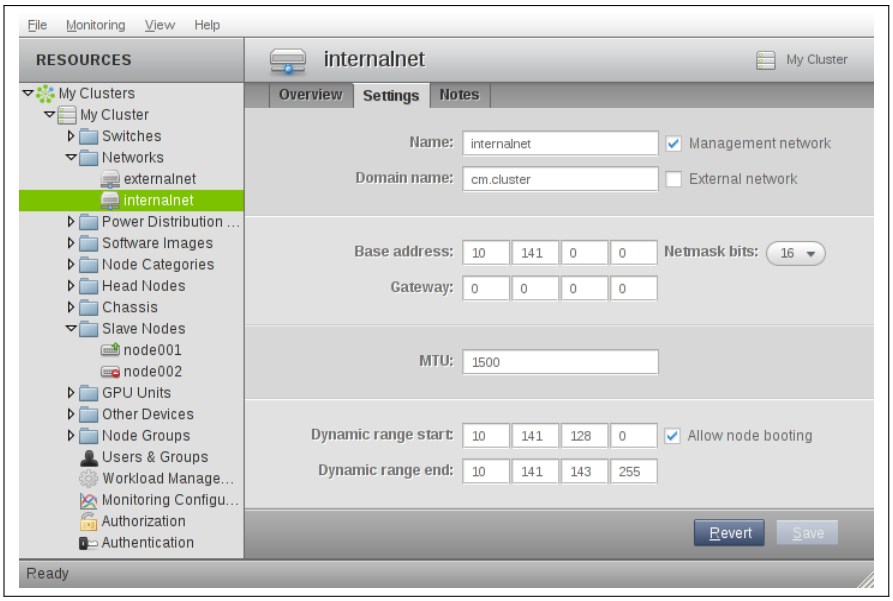

Figure 4.4: Network Settings

Property	Description
`Name`	Name of the network.
`Domain name`	DNS domain associated with the network.
`External network`	Switch to treat the network as an external network.
`Base address`	Base address of the network (also known as the `network address`)
`Netmask bits`	Prefix-length, or number of bits in netmask. The part after the "/" in CIDR notation.

Figure 4.5: Network properties

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`, one would use `node001.storage.cluster` 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 4.2 can be used to add a new network. After the new network has been added, the `Settings` tab (Figure 4.4) 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).

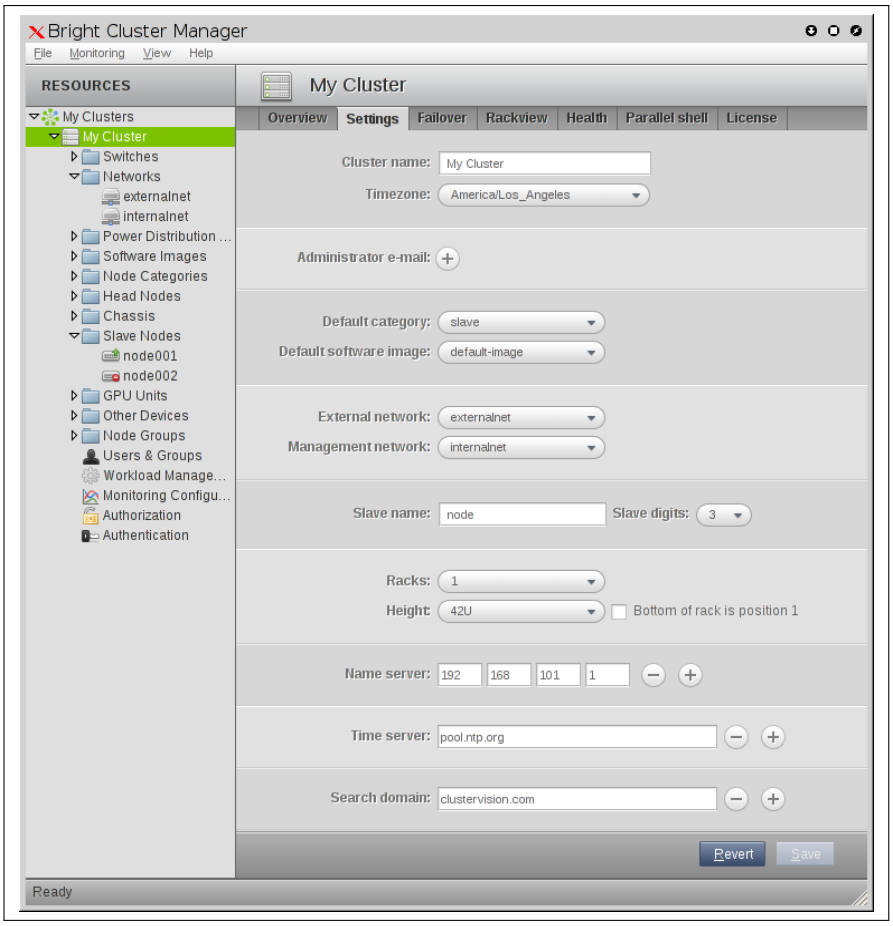

Figure 4.6: Cluster Settings

4.2.3 Configuring External Network Parameters

After both internal and external networks are defined, it may be necessary to change network parameters from their original 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.

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

Example

```
[root@mycluster ~]# cmsh
[mycluster]% device use master
[mycluster->device[mycluster]]% set hostname foobar
[foobar->device*[foobar*]]% commit
```

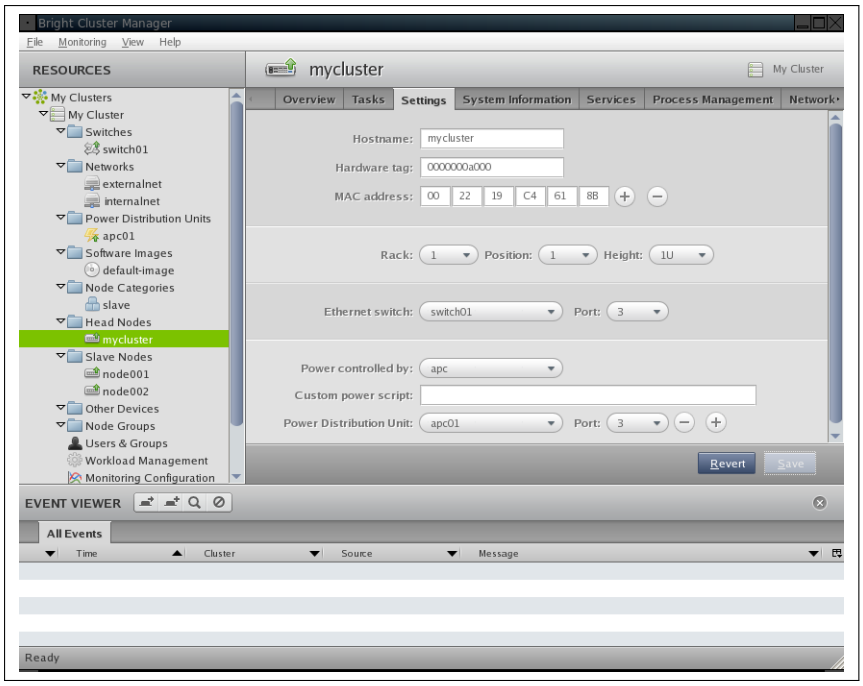

Figure 4.7: Head Node Settings

```
[foobar->device[foobar]]% quit
[root@mycluster ~]# sleep 30; hostname -f
foobar.cm.cluster
[root@mycluster ~]#
```

Note: the shell prompt still shows the hostname as `mycluster`, because its prompt string is only set when a new shell is started.

Changing External Network Parameters

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 object settings of the `Networks` resource involved, and secondly, by the cluster object network settings for connecting to the outside. In more detail:

1. The external network object configuration specifies network settings for nodes facing the outside, such as login nodes or head nodes. This means that network interface particulars associated with the external network for nodes on the external network, are all set here. These particulars are configured in the `Settings` tab of the `Networks` resource of `cmgui` (Figure 4.4) for the following parameters:

   - the IP address parameters:
     - base address
     - netmask
     - gateway
     - DHCP ranges, if using DHCP
   - the network domain (LAN domain, i.e. what domain machines on this network use as their domain),

- network name (what the external network itself is called),
- and MTU size (the maximum value for a TCP/IP packet before it fragments on this network—the default value is 1500).

2. The cluster object configuration sets the other network settings the cluster uses when connecting to the outside. These particulars are configured in the `Settings` tab of the cluster object resource in `cmgui` (Figure 4.6):

   - the nameservers 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).

Changing external IP parameters of a cluster therefore requires making changes in the settings of the two objects above. This can be done in `cmgui` by using the associated settings tabs, as specified in the above.

Changing the external network parameters of a cluster can also be done as follows using `cmsh`:

Example

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

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

To make the cluster use DHCP to obtain its external network settings, the IP address and `baseaddress` of `externalnet` are set to `0.0.0.0`. The gateway address, the nameserver(s), and the IP address of the external address are then obtained via DHCP. Timeserver configuration for `externalnet` is not picked up from the DHCP server, having been set during installation (Figure 2.20), and is changed manually if so desired using `cmgui` as in Figure 4.6, or using `cmsh` in `partition` mode as in the above example.

4.3 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.

4.3.1 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 was 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:

```
[bright51 ~]# for ((i=1; i<=150; i++)) do
> echo device interfaces node‘printf "%03d" $i‘
> echo add ipmi ipmi0
> echo set network ipminet
> echo set ip 10.148.0.$i
> echo 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`:

```
[bright51 ~]# 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:

```
[mc->device[mc]->interfaces]% add alias eth0:0
[mc->device[mc]->interfaces*[eth0:0*]]% set network ipminet
[mc->device[mc]->interfaces*[eth0:0*]]% set ip 10.148.255.254
[mc->device[mc]->interfaces*[eth0:0*]]% commit
[mc->device[mc]->interfaces[eth0:0]]%
Mon Dec  6 05:45:05 2010 mc: Reboot required: Interfaces have been modified
[mc->device[mc]->interfaces[eth0:0]]% quit
[root@mc ~]# /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.

4.3.2 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 was generated during the installation of the head node.

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

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).

Example

Changing IPMI username and password for the entire cluster:

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

4.4 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.

4.4.1 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. By default, all relevant OFED packages are installed on the head node and software images. It is possible to replace the native OFED with a custom OFED on the entire cluster, or on certain software images. Administrators may choose to switch to a different OFED version if the HCAs used are not supported by the native OFED version, or to increase performance using an OFED version that has been optimized for a particular HCA.

If the InfiniBand network was enabled during installation, the `openibd` service was scheduled to be started at boot-up for all nodes. The `openibd` service 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 `openibd` service to be started at boot-time on the head node and inside the software images.

4.4.2 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.

When the head node in a cluster is equipped with an InfiniBand HCA, it is a good candidate to run as a Subnet Manager. The following command can be used to configure the Subnet Manager to be started at boot-time on the head node:

```
chkconfig opensmd on
```

The following cmsh commands may be used to schedule the Subnet Manager to be started on one or more nodes:

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

On large clusters it is recommended to use a dedicated node to run the Subnet Manager.

4.4.3 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 was 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 4.3.1) as follows:

```
[root@bright51 ~]# echo "device
> addinterface -n node001..node150 pysical 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:

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

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

4.4.4 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@mycluster ~]# su - cmsupport
[cmsupport@mycluster ~]$ cd /cm/shared/apps/imb/current/
[cmsupport@mycluster current]$ ./setup.sh
[cmsupport@mycluster current]$ cd ~/BenchMarks/imb/3.2
[cmsupport@mycluster 3.2]$ module load openmpi/gcc
[cmsupport@mycluster 3.2]$ module initadd openmpi/gcc
[cmsupport@mycluster 3.2]$ make -f make_mpi2
[cmsupport@mycluster 3.2]$ mpirun -np 2 -machinefile ../nodes IMB-MPI2 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 different nodes 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.

4.5 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.

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

```
[mycluster]% device use switch01
[mycluster->device[switch01]]% get readstring
public
[mycluster->device[switch01]]% get writestring
private
[mycluster->device[switch01]]% set readstring public2
[mycluster->device*[switch01*]]% set writestring private2
[mycluster->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 4.8). 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.

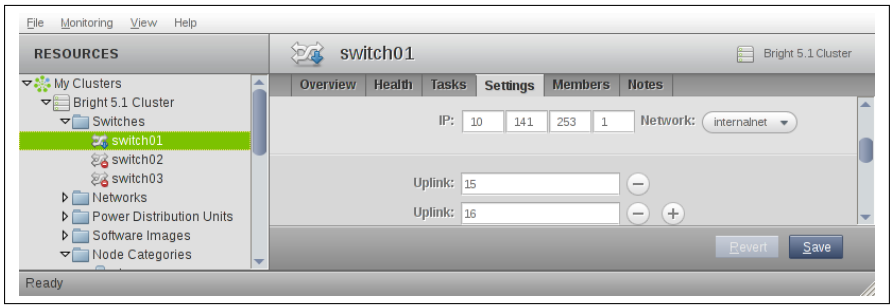

Figure 4.8: Notifying CMDaemon About Uplinks With `cmgui`

- 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@bright51 ~]# cmsh
[bright51]% device
[bright51->device]% set switch01 uplinks 15 16
[bright51->device*]% set switch02 uplinks 01
[bright51->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 6.3.2).

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@bright51 ~]# cmsh
[bright51]% device
[bright51->device]% showport 00:30:48:30:73:92
[bright51->device]% switch01:12
```

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

If the switch is known, as well as the MAC address, then the switch can also be specified with the "-s" option. If this is done, the query is carried out on that switch only. Continuing the earlier example:

```
[bright51->device]% showport -s switch01 00:30:48:30:73:92
[bright51->device]% switch01:12
```

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 mac address
00:30:48:30:73:92 bright51: 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 mac address
```

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 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, turn power off, reset the power, and retrieve the power status, and explains how these can be applied to devices in various ways. Section 5.3 briefly covers monitoring power.

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 will control it. Figure 5.1 shows the `Settings` tab for a head node.

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 will be ignored by default. Overriding the default is described in section 5.1.3.

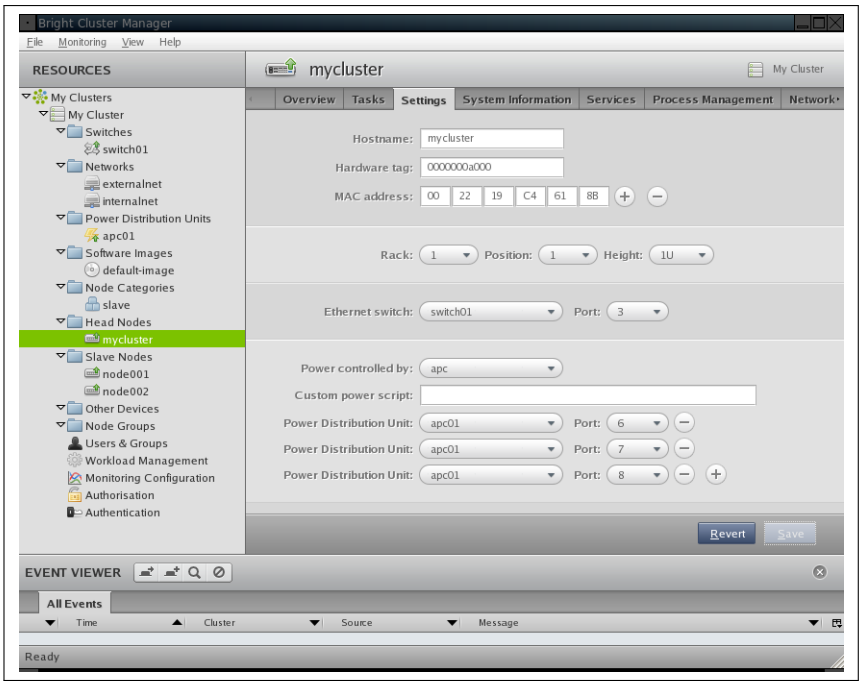

Figure 5.1: Head Node Settings

Since nodes may have multiple power feeds, there may be multiple PDU ports defined for a single device. The cluster management infrastructure will take 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 will apply to all nodes sharing the same PDU port.

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.

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 4.3.

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
```

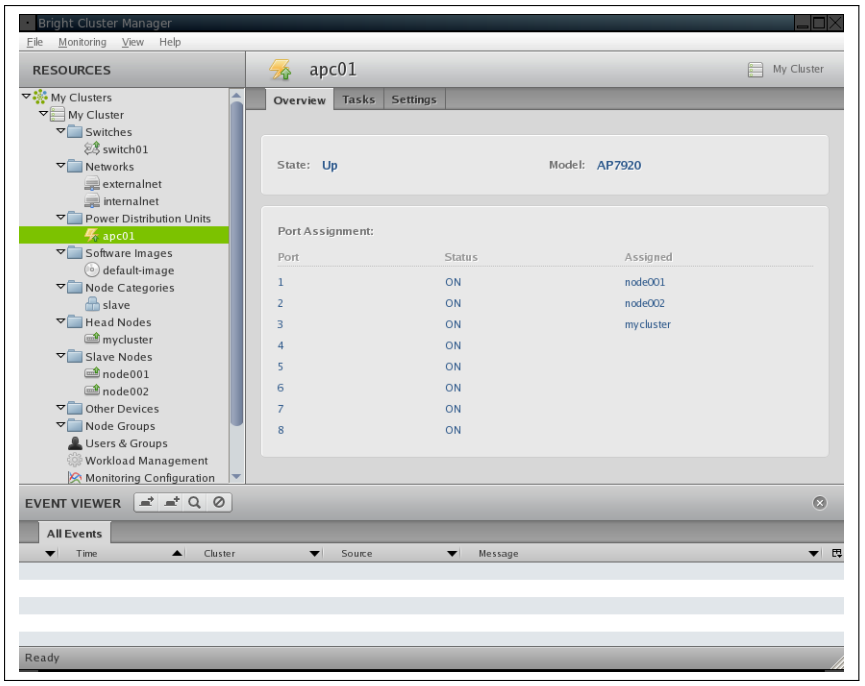

Figure 5.2: PDU Overview

```
[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` (see section 3.7.2, 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 sequence with `PowerOffPDUOutlet` set to `true` is: A power on instruction via `cmsh` or `cmgui` has CMDaemon power on the PDU port. This starts up the BMC in the node. Once the BMC is up, an IPMI-based power on instruction from CMDaemon powers on the node.

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/apps/cmd/scripts/powerscripts/power-example`. In it, setting `$3` to a natural number 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, it turns out that the goal of the task can be simplified considerably and achieved quicker by using the environment variables available in the cluster management daemon environment for this example. How to do this is examined in the next section.

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` package is in the Bright Cluster Manager rpm repository, so it is easily upgraded if needed for more recent hardware. The installation is done on the head node, the node image, and in the node-installer as follows:

```
yum install hponcfg
yum install hponcfg --installroot=/cm/images/default-image
yum install hponcfg --installroot=/cm/node-installer
```

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 `slave` category:

   Example

   ```
   [mycluster]% device foreach -c slave (set custompowerscript /cm/local/a\
   pps/cmd/scripts/powerscripts/ilo_power.pl)
   [mycluster]% device foreach -c slave (set powercontrol custom)
   [mycluster]% device commit
   ```

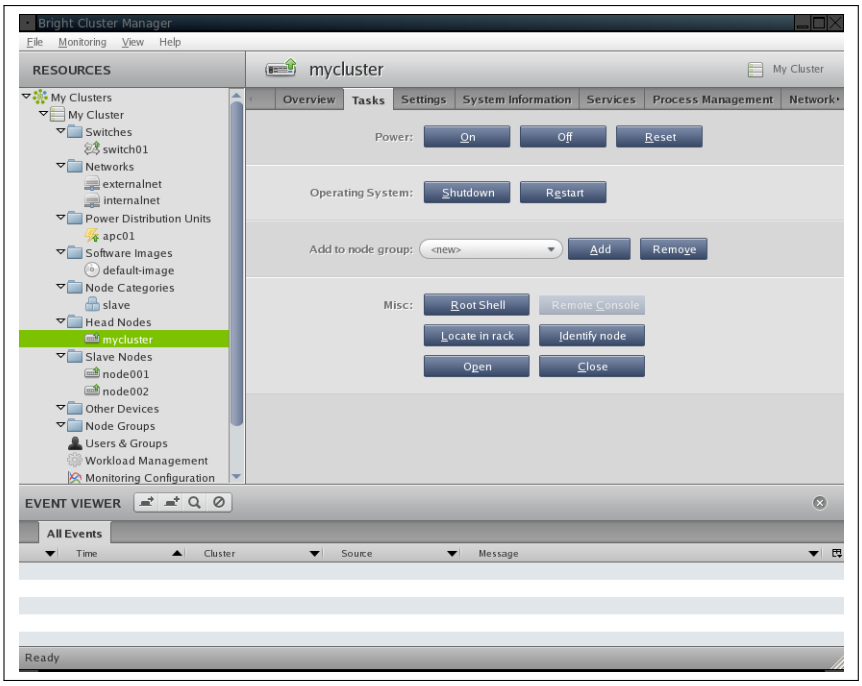

Figure 5.3: Head Node Tasks

5.2 Power Operations

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

- 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 `Slave 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 every 2 successive de-

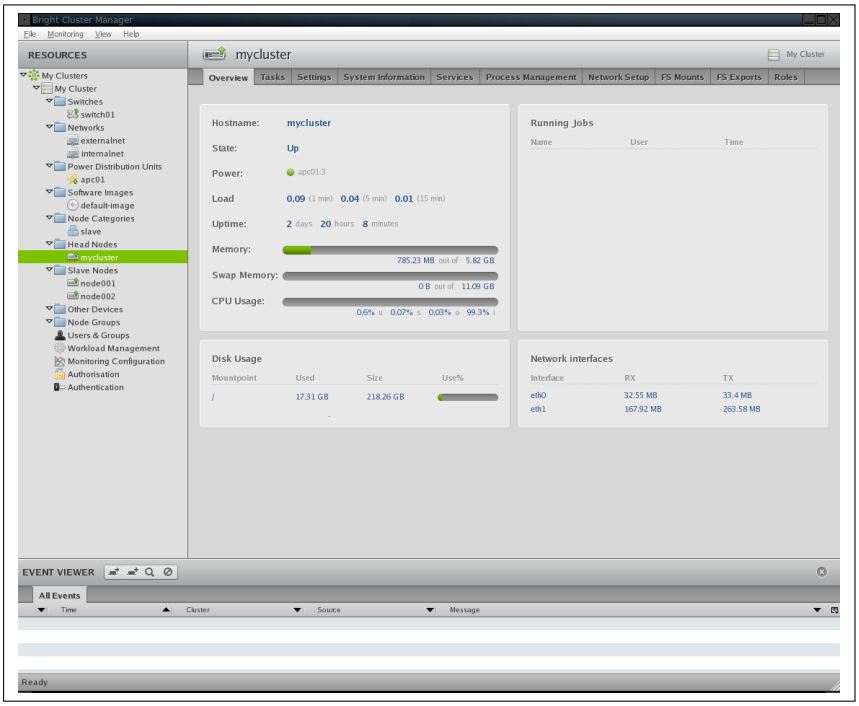

Figure 5.4: Head Node Overview

vices in the operating sequence, 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:

- Powering on `node001` and `node018`:

  Example

  ```
  [mycluster]% device power -n node001,node018 on
  apc01:1 ............. [   ON    ] node001
  apc02:8 ............. [   ON    ] node018
  ```

- Powering off all nodes in the `slave` category with a 100ms delay between nodes (some output elided):

  Example

  ```
  [mycluster]% device power -c slave -d 0.1 off
  apc01:1 ............. [   OFF   ] node001
  apc01:2 ............. [   OFF   ] node002
  ...
  apc23:8 ............. [   OFF   ] node953
  ```

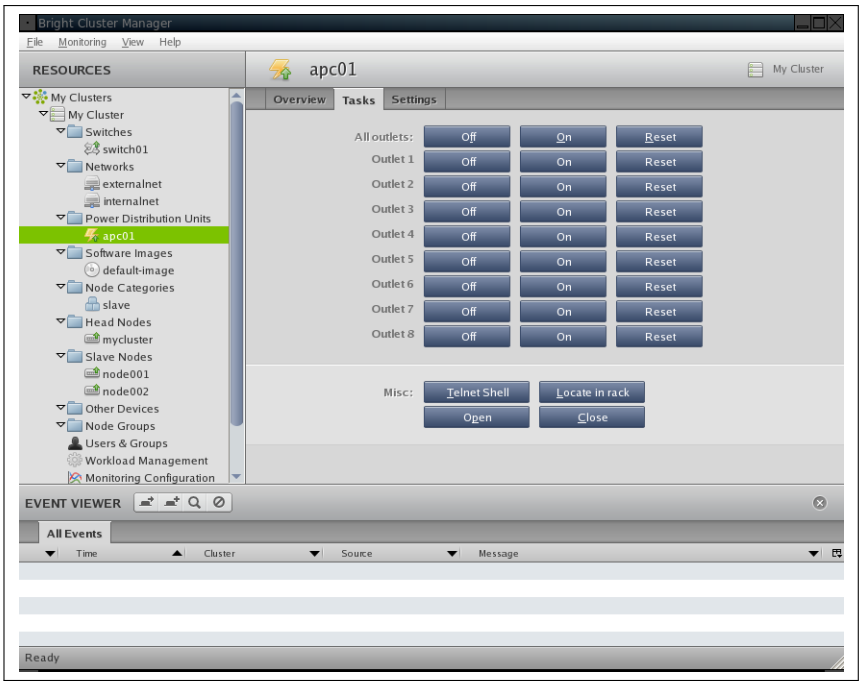

Figure 5.5: PDU Tasks

- 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:

```
power                           [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND
power -c|--category CATEGORY... [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND
power -g|--group GROUP...       [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND
power -h|--chassis CHASSIS...   [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND
power -n|--nodes NODES...       [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND
power -p|--pduport PDUPORT...   [-b|--background][-d|--delay [SECONDS]] POWERCOMMAND


POWERCOMMAND: one of:   on     ... Turn power on for current device
                        off    ... Turn power off for current device
                        reset  ... Reset power for current device
                        status ... Retrieve power status of the current device,
                                   (or all if none is selected)

CATEGORY      ... e.g. anycategory,anothercategory,yetanothercategory
GROUP         ... e.g. anygroupname,anothergroupname,yetanothergroupname
CHASSIS       ... e.g. anychassisname,anotherchassisname,yetanotherchassisname
NODES         ... e.g. node001..node015,node20..node028,node030
PDUPORT       ... e.g. apc01:* or apc01:8,apc01:5
SECONDS       ... e.g. 0.2. By default, 1 second
```

Figure 5.6: Synopsis Of `power` Command In `device` Mode Of `cmsh`

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.

6

Node Provisioning

By default, nodes boot from the network when using Bright Cluster Manager, and a *network boot*, sometimes called a *PXE boot*, is recommended as a BIOS setting for nodes.

On disked nodes, gPXE software is placed by default on the drive during node installation. If the boot instructions from the BIOS for PXE boot 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.

Besides network boot, a node can also be configured to boot entirely from its drive.

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.

The process of getting the software image onto the nodes and getting the nodes into a good state is called *node provisioning*, and ensures that a node is always restored to a known good state before cluster users use it.

The details of node provisioning are described in this chapter.

6.1 Provisioning Nodes

In simple clusters, node provisioning is done only by the head node. 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.

6.1.1 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 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.

6.1.2 Provisioning Nodes: Role Setup With cmsh

In the following cmsh example the administrator creates a new category called misc. The default category slave 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, login, and failover.

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

```
[mycluster]% category add misc
```

```
[mycluster->category*[misc*]]% roles
[mycluster->category*[misc*]->roles]% assign provisioning
[mycl...*]->roles*[provisioning*]]% set allimages false
[mycl...*]->roles*[provisioning*]]% set images default-image
[mycl...*]->roles*[provisioning*]]% set maxprovisioningnodes 20
[mycl...*]->roles*[provisioning*]]% show
Parameter                Value
------------------------ ------------------------------------------------
Name                     provisioning
Type                     ProvisioningRole
allImages                no
images                   default-image
maxProvisioningNodes     20
nodegroups
[mycluster->category*[misc*]->roles*[provisioning*]]% commit
[mycluster->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

```
[mycluster]% device use node001
[mycluster->device[node001]]% roles
[mycluster->device[node001]->roles]% assign provisioning
[mycluster->device*[node001*]->roles*[provisioning*]]%
...
```

After carrying out a role change, the `updateprovisioners` command described in section 6.1.4 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 6.1.4).

6.1.3 Provisioning Nodes: Role Setup With `cmgui`

The provisioning configuration outlined in `cmsh` mode in section 6.1.2 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).

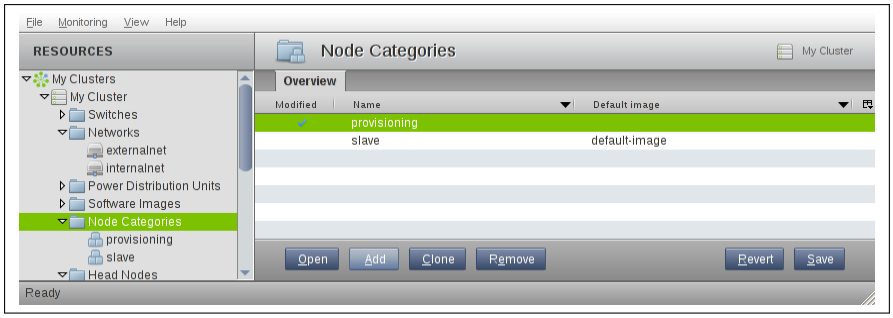

Figure 6.1: `cmgui`: Adding A `provisioning` Category

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).

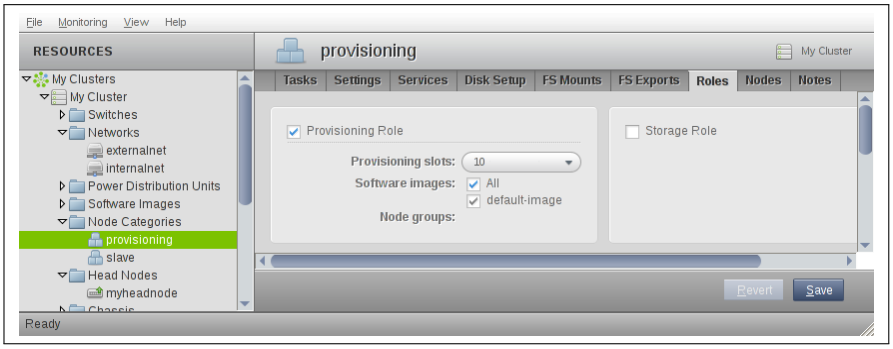

Figure 6.2: `cmgui`: Configuring A `provisioning` Role

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.

6.1.4 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 sequence 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 6.1.1), the head node also limits how many simultaneous provisioning tasks are allowed to run on the entire cluster. This is set using the `MaxNumberOfProvision-`

ingThreads 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):

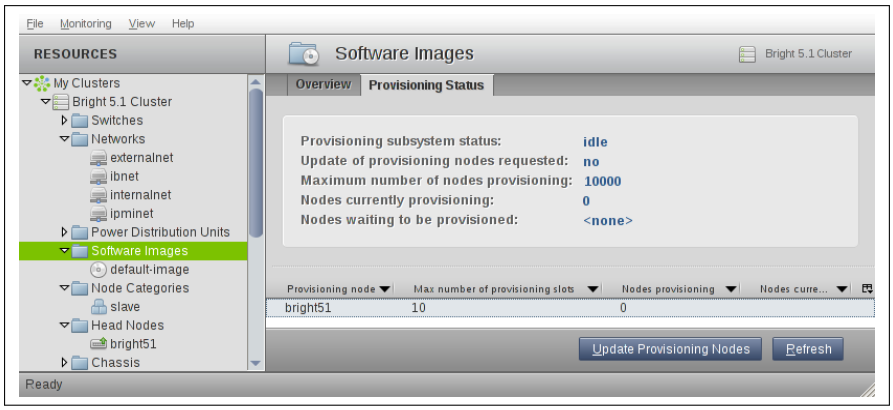

Figure 6.3: cmgui: A Button To Update Provisioning Nodes

In examples in Section 6.1.2, 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 images from the head node image to the provisioners according 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

```
[mycluster]% softwareimage updateprovisioners
Provisioning nodes will be updated in the background.

Sun Dec 12 13:45:09 2010 myheadnode: Starting update of software image(s) o\
                                     n provisioning node(s). (user initiated).
[mycluster]% softwareimage updateprovisioners [mycluster]%
Sun Dec 12 13:45:41 2010 myheadnode: Updating image default-image on provis\
                                     ioning node node001.
[mycluster]%
Sun Dec 12 13:46:00 2010 myheadnode: Updating image default-image on provis\
                                     ioning node node001 completed.
Sun Dec 12 13:46:00 2010 myheadnode: Provisioning node node001 was updated.
Sun Dec 12 13:46:00 2010 myheadnode: Finished updating software image(s) on\
                                     provisioning node(s).
```

6.2 Software Images

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 greater 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 6.1.4).

6.2.1 Selecting Kernel Driver Modules To Load Onto Nodes

Each software image contains a Linux kernel and a ramdisk. The ramdisk is loaded after the kernel during early boot, and contains driver modules for the node's network card and local storage.

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 image and sends the updated image to all provisioning nodes.

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.

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).

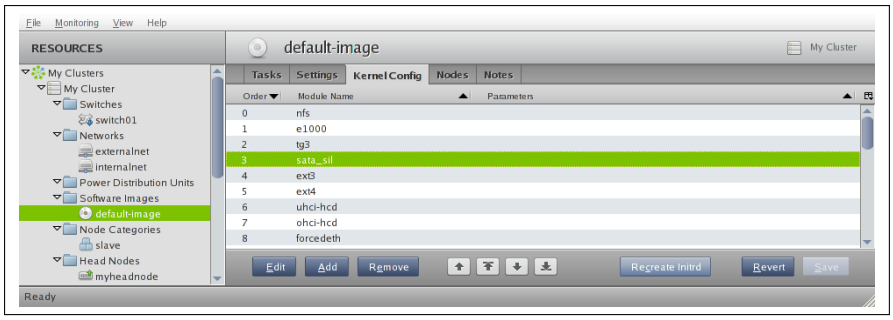

Figure 6.4: `cmgui`: Selecting Kernel Modules For Node Images

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.

6.3 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.
2. deciding or selecting node configuration.
3. starting up all network interfaces.
4. determining install-mode type and execution mode.
5. running `initialize` scripts
6. checking partitions, mounting filesystems
7. synchronizing the local drive with the correct software image.
8. writing network configuration files to the local drive.
9. creating an `/etc/fstab` file on the local drive.
10. installing GRUB bootloader if configured.
11. running `finalize` scripts.
12. unloading specific drivers no longer needed.
13. switching the root device to the local drive and calling `/sbin/init`.

These 13 node-installer steps and related matters are described in detail in the corresponding sections 6.3.1–6.3.13.

6.3.1 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).

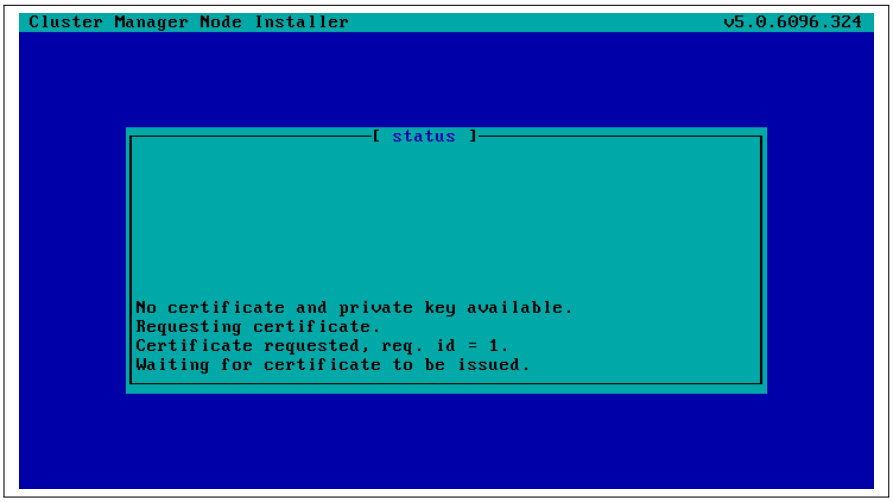

Figure 6.5: Certificate Request

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:

```
[mycluster]% cert autosign
on
[mycluster]% cert autosign off
off
[mycluster]% cert autosign
off
[mycluster]%
```

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.

6.3.2 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.

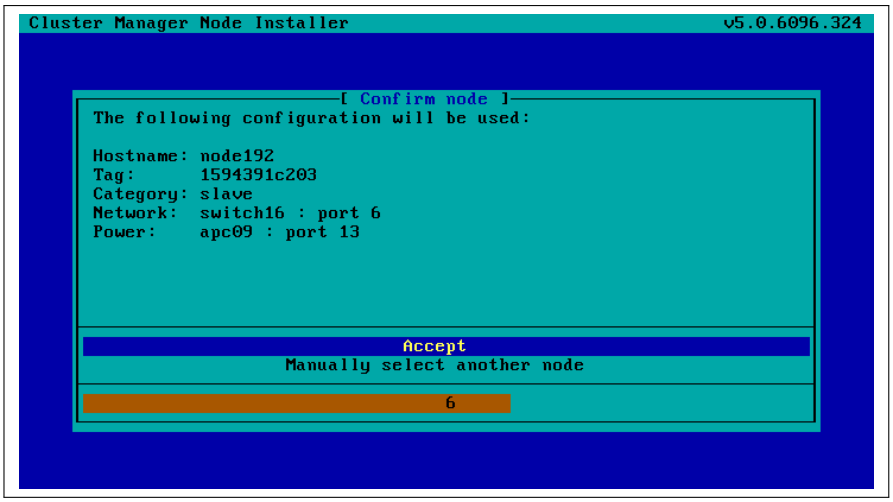

Figure 6.6: Scenarios: Configuration Found, Confirm Node Configuration

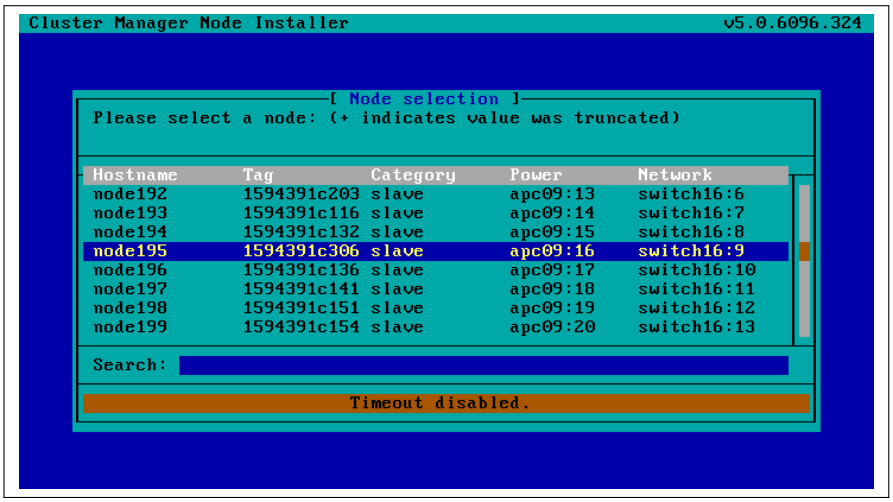

Figure 6.7: Scenarios: Node Selection Sub-Dialog

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.

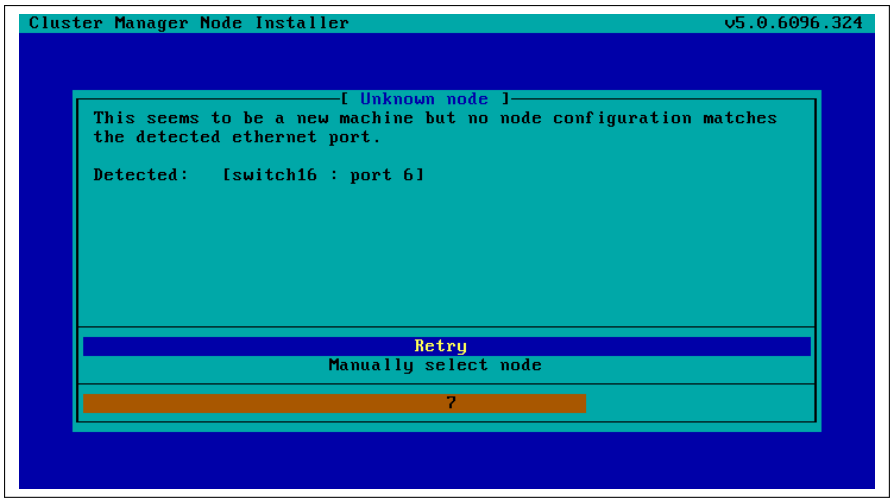

Figure 6.8: Scenarios: Unknown Node, Ethernet Port Detected

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.

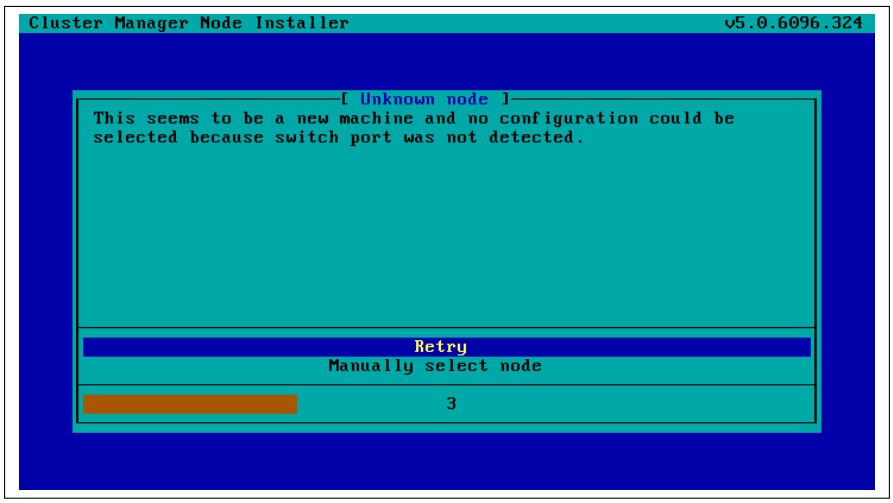

Figure 6.9: Scenarios: Unknown Node, No Ethernet Port Detected

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.

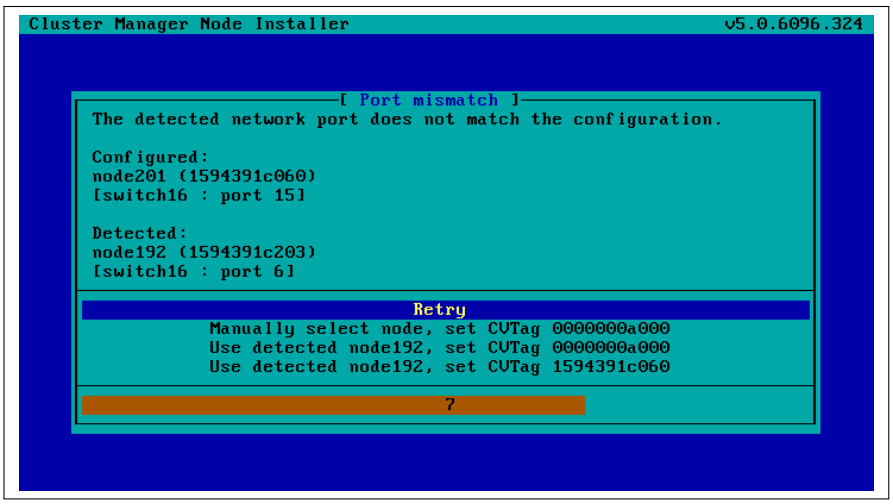

Figure 6.10: Scenarios: Port Mismatch Dialog

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.

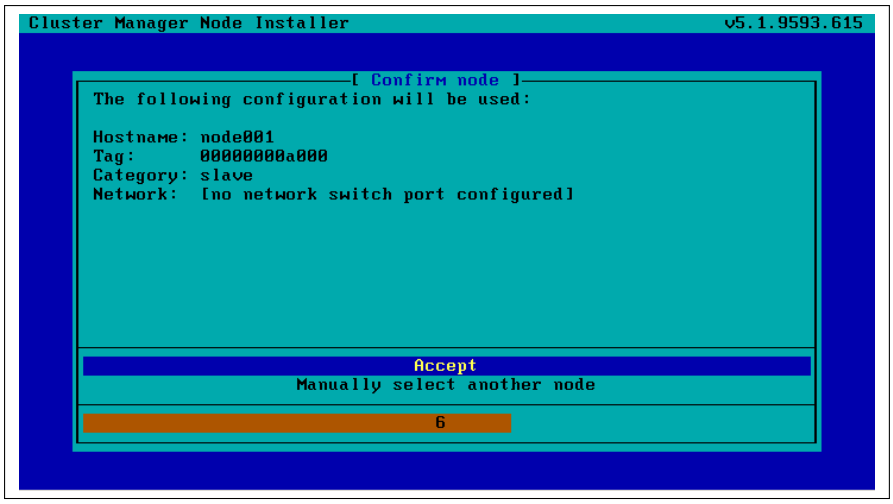

Figure 6.11: Scenarios: Port Unset Dialog

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

New nodes that have not been configured yet can be detected using the `newnodes` command from within the `device` mode in `cmsh`.

Example

```
[bright51->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]
[bright51->device]%
```

These 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

```
[bright51->device]% showport 00:0C:29:01:0F:F8
switch01:8
[bright51->device]% set node003 ethernetswitch switch01:8
[bright51->device*]% commit
```

node identification wizard

The *node identification wizard* tabbed pane, under the `Slave Nodes` resource (Figure 6.12) 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. Additionally, it can help assign a node name to the node, assuming the node name exists (for example by running the node creation wizard of Section 6.6.2). After assignment is done, the new status is saved with the `Save` button of the `Overview` tabbed pane.

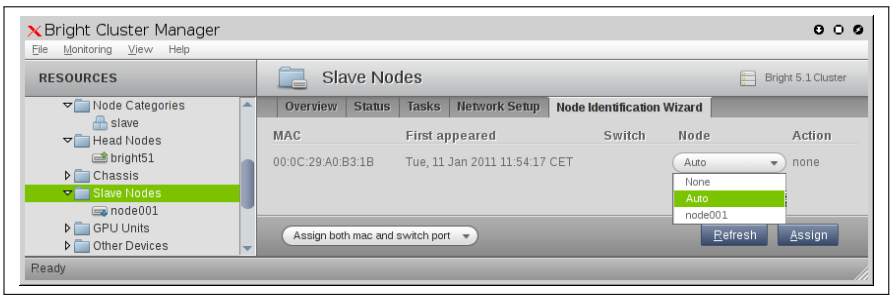

Figure 6.12: Node Identification Wizard

The most useful way of using the wizard is for node assignment in large clusters.

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 6.6.2). The nodes are also assumed to be set for net booting.

The physical nodes are then powered up in an arranged order. Because they are unknown new nodes, the nodes 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.

6.3.3 Starting Up All Network Interfaces

At the end of section 6.3.2, 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 interfaces required for the installer with IP addressing, 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.

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 6.3.3) the network interfaces are up. When the node installer has completed the remainder of its 13 steps (Sections 6.3.4–6.3.13), 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.

6.3.4 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.

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 6.3.7 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 or 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 6.13):

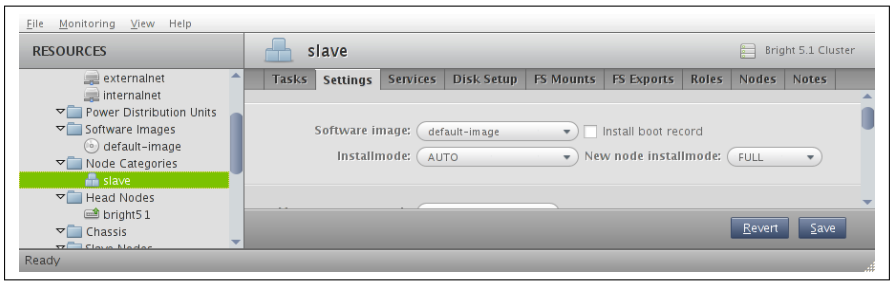

Figure 6.13: `cmgui` Install-Mode Settings Under Node Category

   or using `cmsh` with a one-liner like:

```
cmsh -c "category use slave; 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 6.14). This only affects the current boot. By default the PXE menu install mode option is set to `AUTO`.

Figure 6.14: 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 6.15):

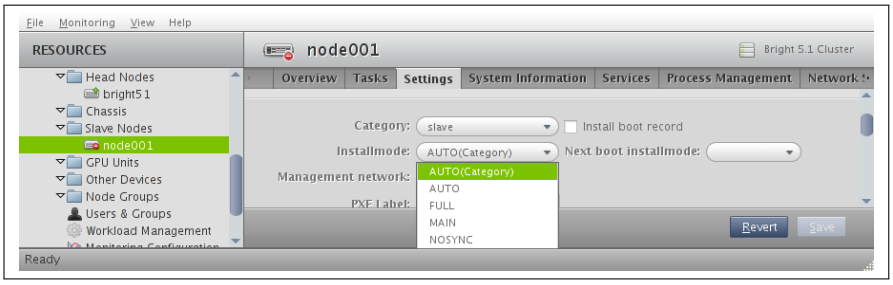

Figure 6.15: cmgui Install-Mode Settings For The Node

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 set in the node configuration. This can be set using cmgui (Figure 6.15), 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 6.13), or using `cmsh` with a one-liner like:

```
cmsh -c "category use slave; 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 6.16) 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`:

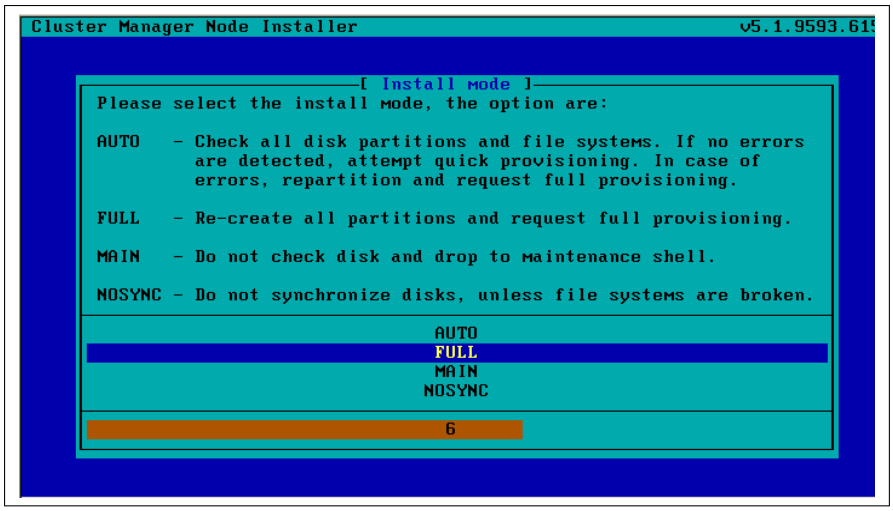

Figure 6.16: Install-Mode Setting Option During Node-Installer Run

6.3.5 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 6.3.11). 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 6.5.2).

6.3.6 Checking Partitions, Mounting File Systems

In the previous section, 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 has checked the drive(s) and, if required, recreated 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 show no errors, the node starts up without getting an image synced to it from the provisioning node. If the partition and filesystem checks show errors, then the node does get a known good image synced across.

The node-installer is capable of creating advanced drive layouts, including software RAID and LVM setups. Some drive layout examples, including documentation, are given in appendix D.

6.3.7 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

Image synchronization is done using rsync. 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 be excluded from consideration during synchronization. The excluded list that is passed on to rsync is decided by the type of synchronization chosen: full or sync:

- A sync type of synchronization uses the excludelistsyncinstall property to specify what files and directories to exclude from consideration when copying over the rest of the filesystem from the known good image. This list has sections of the filesystem that

should be retained across boots, such as log files. On the node that is being copied to, the remaining files and directories which undergo synchronization lose their original contents.

- A `full` type of synchronization uses the `excludelistfullinstall` property to specify what files and directories to exclude from consideration when copying over parts of the file system from a known good image. This is a small list of exclusions, containing items such as `/proc`. The default list allow a full filesystem to be copied over to the node. As with the preceding option, files and directories that are being synchronized on the node lose their original contents.

The exclude lists are passed to rsync using its `--exclude-from` option. The syntax of an exclude list can be quite involved. The rsync manual page, specifically the `INCLUDE/EXCLUDE PATTERN RULES` section, gives details on how such a list is built. A `cmsh` one-liner to get an exclude list is for a category is:

```
cmsh -c "category use slave; get excludelistfullinstall"
```

Similarly, to set the list:

```
cmsh -c "category use slave; set excludelistfullinstall; commit"
```

where a text-editor opens up to allow changes to be made to the list. Figure 6.17 illustrates how the setting can be modified via `cmgui`.

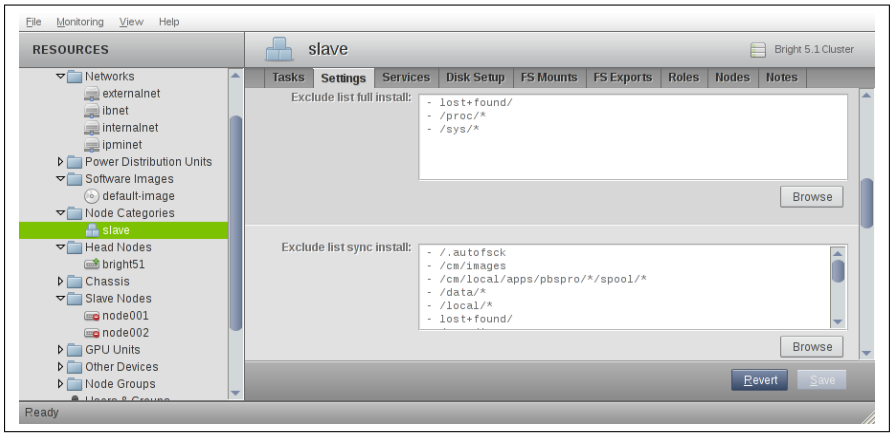

Figure 6.17: Setting up exclude lists with `cmgui` for provisioning

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:

```
bright51:~ # 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 bright51:
  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

```
[bright51->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 6.18):

Figure 6.18: `cmgui`: Provisioning Log Button For A Device Resource

6.3.8 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 6.3.9–6.3.13) 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.

6.3.9 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 both in the node's category and in the node configuration. From `cmsh` the extra mount points can be managed from the `fsmounts` submode of the `category` or `device` mode.

6.3.10 Installing GRUB Bootloader

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.

For this to work:

- network booting should have a lower priority in the BIOS of the node than hard drive booting

- the GRUB bootloader with a boot record must be installed in the MBR of the local drive, overwriting the default gPXE boot record.

  To do this 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 slave; set installbootrecord yes; commit"
  ```

This ensures that the next boot is from GRUB on the hard drive, instead of a boot from the head node image via the network.

Simply unsetting the "Install boot record" setting and rebooting the node does not restore its gPXE booting. To restore its gPXE booting, it 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.

6.3.11 Running Finalize Scripts

A *finalize script* is similar to an `initialize` script (section 6.3.5), only it runs a few stages later in the node-provisioning process.

A `finalize` script 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, or to supply a configuration file that cannot be added to the provisioned image because it needs node-specific settings. Such custom commands 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 6.3.5). 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 6.5.2).

6.3.12 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.

6.3.13 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.

6.4 Node States

6.4.1 Node States Indicating Regular Start Up

Throughout the boot process the node sends several state change messages to the head node CMDaemon. During a successful boot process the node goes through the following states:

- `INSTALLING`. This state is entered as soon as the node-installer has determined on which node the node-installer is running.
- `INSTALLER_CALLINGINIT`. This state is entered as soon as the node-installer has handed over control to the local `init` process.
- `UP`. This state is entered as soon as the CMDaemon of the node connects to the head node CMDaemon.

These states can be seen in the event viewer pane in `cmgui`, or in the console within `cmsh`, with messages indicating the name of the node that is in the "Installing", "Calling Init", or "Up" state.

6.4.2 Node States Indicating Problems

Several other node states are used to indicate problems in the boot process:

- `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 can not find the local drive, a network interface could not be started, etc. 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`.

6.5 Updating Running Nodes

6.5.1 Updating Running Nodes: `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 6.1.4). The image is then installed from the provisioning system onto a regular node when it (the regular node) reboots via a provisioning request (Section 6.3.7).

However, updating a running node with the latest changes from the software image is also possible without rebooting it. 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, as detailed in Section 6.3.7. This exclude list is defined in the `excludelistupdate` property of the node's category. The main difference between `excludelist`*update* from this section and `excludelistsync`*install*/`excludelistfull`*install* from Section 6.3.7 is indicated by the parts of their names emphasized here. Namely, the `excludelistupdate` property settings concern an update to a running system, while the other two are about an install during node start-up. The syntax for the exclude list in the update case remains the same as that of the install cases, i.e. defined by the syntax used by rsync's `--exclude-from` option.

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 slave; set excludelistupdate; commit"
```

The exclude list can be edited in `cmgui` as indicated in Figure 6.19.

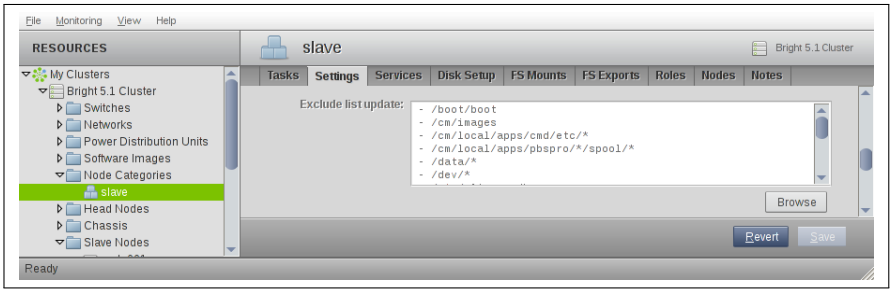

Figure 6.19: Setting up exclude lists with `cmgui` for node updates

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.

6.5.2 Updating Running Nodes: `imageupdate`

Using a defined `excludelistupdate` property, the `imageupdate` command of `cmsh` is used to start an update on a running node:

Example

```
[mycluster->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 bright51: Provisioning started on node node001
[bright51->device]% imageupdate -n node001: image update in progress ...
[bright51->device]%
Tue Jan 11 12:13:44 2011 bright51: 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 6.3.7).

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

```
[mycluster->device]% imageupdate -n node001 -w
Provisioning started on node node001
node001: image update in progress ...
[mycluster->device]% Provisioning completed on node node001
```

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 6.20). This opens up a dialog which has a dry-run checkbox marked by default.

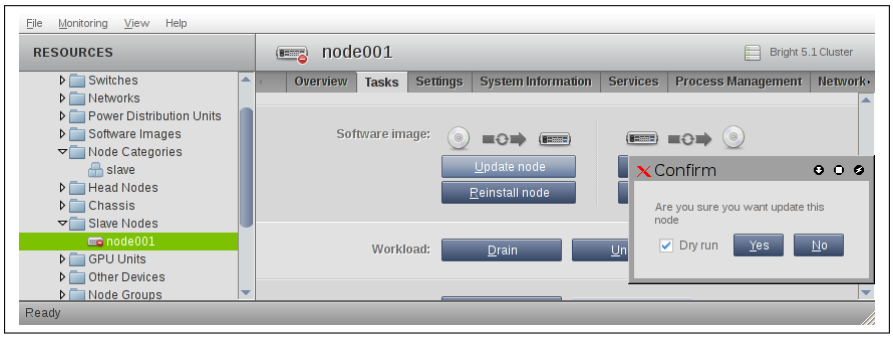

Figure 6.20: Updating A Running Node's Image With `cmgui`

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 an image via `cmsh` or `cmgui` automatically updates the provisioners first if the provisioners have not been updated in the last 5 minutes.

There are two scripts associated with the `imageupdate` command that may run as part of its execution:

- The *imageupdate_initialize script* runs before the node 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 node image has updated.

These differ from the `initialize` (section 6.3.5) and `finalize` (section 6.3.11) 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.

6.6 Adding New Nodes

6.6.1 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

```
[bright51->device]% add slavenode node002
[bright51->device*[node002*]]% commit
```

The `cmgui` equivalent of this is to go within the `Slave Nodes` resource, and after the `Overview` tabbed pane for the `Slave Nodes` resource comes up, to click on the `Add` button (Figure 6.21)

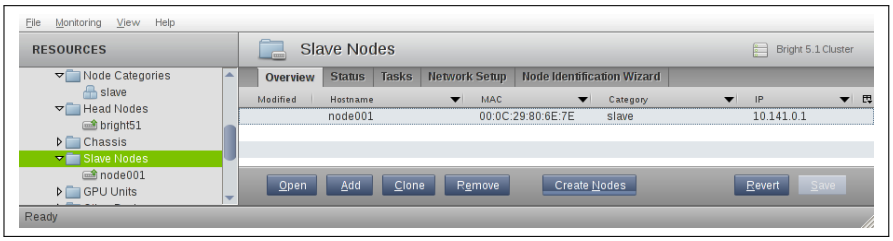

Figure 6.21: Node Creation Wizard Button

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.

6.6.2 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 `Slave Nodes` resource, by selecting the `Overview` tabbed pane and then the `Create Nodes` button (Figure 6.21).

This wizard should not be confused with the closely related node *identification* wizard described earlier in section 6.3.2, 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 6.22).

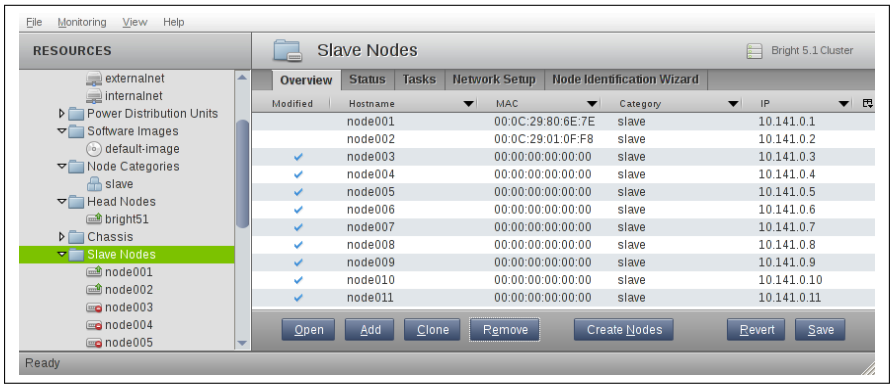

Figure 6.22: Node Creation Wizard: 10 Placeholders Created

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 6.3.2. 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 6.23).

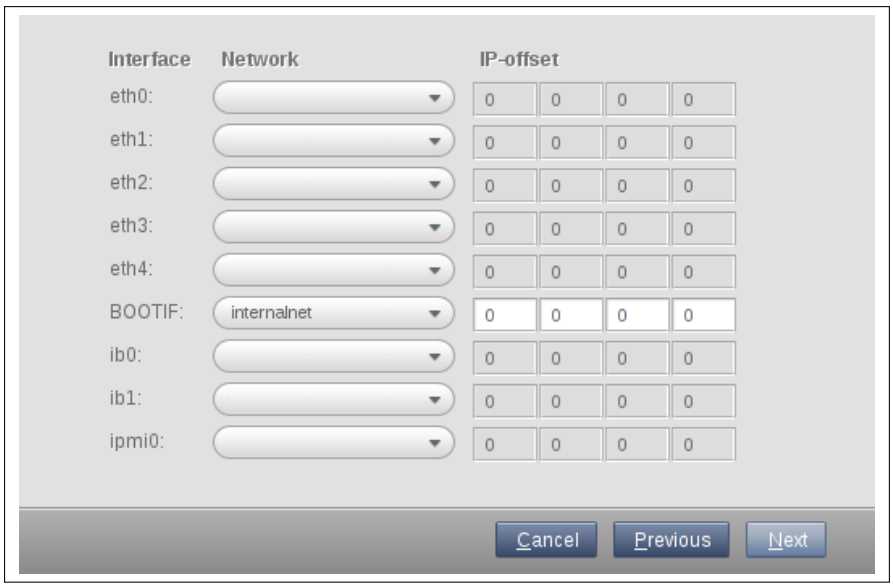

Figure 6.23: Node Creation Wizard: Setting Interfaces

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 *x*th 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.

6.7 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.

6.7.1 Node Fails To PXE Boot

Possible reasons to consider if a node is not even starting PXE boot in first place:

- There may a a bad cable connection. This can be due to moving the machine, or heat creep, or other 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 the driver.
- The interface may have a hardware failure. In that case, the interface should be replaced.

6.7.2 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 syslog. In a default Bright Cluster Manager syslog setup, the messages end up in `/var/log/node-installer` on the head node. Optionally extra log information can be written by enabling debug logging. To enable debug logging change the debug field in the node-installer configuration file `/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.

6.7.3 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 6.3.7). Hints on provisioning problems are often found by looking at the tail end of the log.

6.7.4 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 6.24.

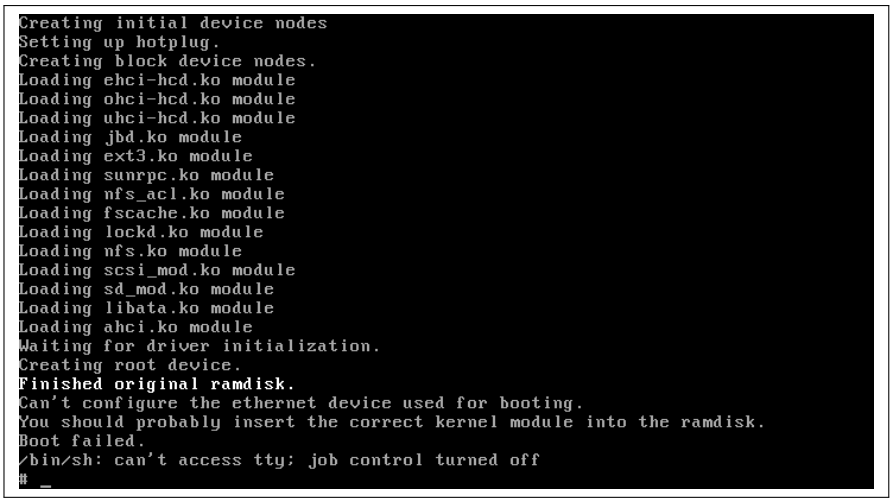

Figure 6.24: No Network Interface

To solve this issue the correct kernel module should be added to the software image's kernel module configuration. 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]]%
```

Note that after committing the change, it can take some time, typically a minute, before the ramdisk creation is done.

6.7.5 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 6.25. The node-installer log file 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.
```

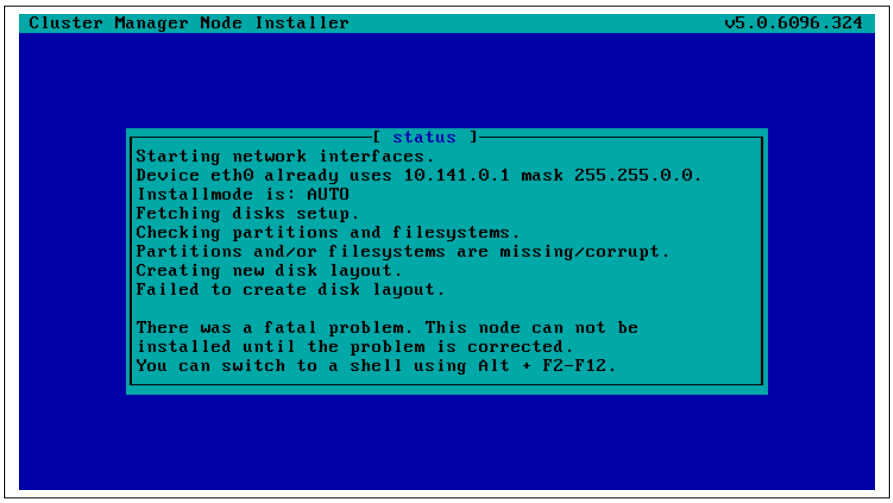

Figure 6.25: No Disk

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.

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@bright51 ~]# 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 node 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

```
[bright51]% softwareimage use default-image
[bright51->softwareimage[default-image]]% kernelmodules
[bright51->softwareimage[default-image]->kernelmodules]% add mpt2sas
[bright51->softwareimage[default-image]->kernelmodules*[mpt2sas*]]% com\
mit
[bright51->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:55:43 2011 bright51: Initial ramdisk for image default-im\
age is being generated
[bright51->softwareimage[default-image]->kernelmodules[mpt2sas]]%
Thu May 19 16:56:31 2011 bright51: Initial ramdisk for image default-im\
age was regenerated successfully
[bright51->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.

6.7.6 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 6.26. 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 can then be used to run the required commands instead.

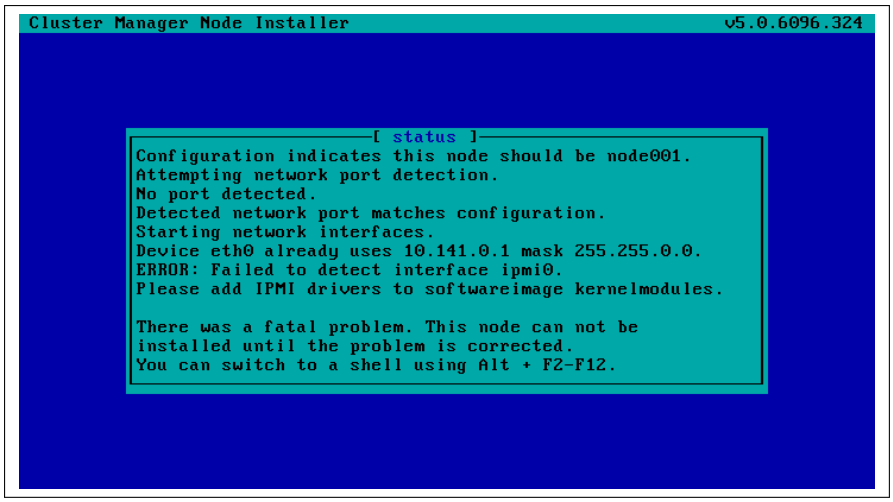

Figure 6.26: No IPMI Interface

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` (see Figure 7.1) will by default list the LDAP object entries for regular users. These entries are clickable and can be managed further.

By default, there will already be one user on a newly installed system: `cmsupport`. This is used to run various diagnostics utilities in Bright Cluster Manager, and should not be modified.

The following five buttons are available to manipulate the entries in

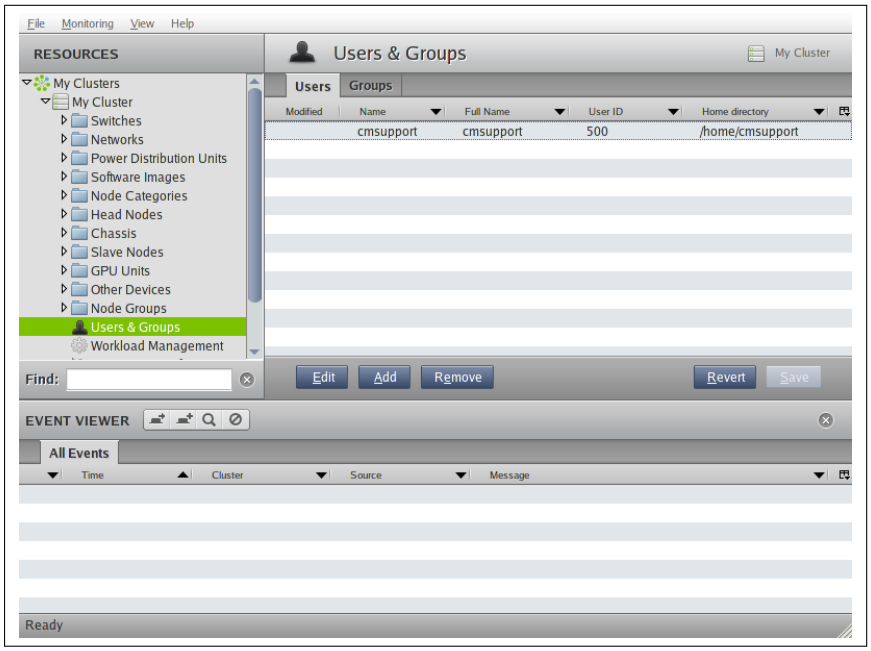

Figure 7.1: `cmgui` User Management

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 according to the policy of the underlying Linux distribution used. For example, 500 and onwards is used for regular UIDs in Red Hat, whereas 1000 and onwards is used in SuSE.
   - 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.
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 will then show up, similar to the user entries already covered above. 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
Group ID
Home directory
Login shell
Password                       < not set >
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 will store the modified state at the user `maureen` level:

Example

```
[mycluster->user*[maureen*]]% commit
[mycluster->user[maureen]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Common name                    maureen
Group ID                       502
Home directory                 /home/maureen
Login shell                    /bin/bash
Password                       *********
User ID                        502
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 will be covered next.

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 above 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, will suggest several parameters that can be set, one of which is password:

Example

```
[mycluster->user[maureen]]% set
Usage: set <parameter> <value> [<value> ...]
           Set value(s) of the specified parameter from the current user

       commonname ..................... Full user name
       groupid ........................ Base group of this user
       homedirectory .................. Home directory
       loginshell ..................... Login shell
       password ....................... Password
       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                       503
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                       503
Group members                  maureen
Group name                     printer
```

The clear command can also be used to clear members—but will clear all of the extras from the group:

Example

```
[mycluster->user[printer]]% clear groupmembers
[mycluster->user*[printer*]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
Group ID                       503
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 will revert 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 will then revert the properties of group printer back to their last committed state (and not affect other objects):

Example

```
[mycluster->user*[printer*]]% refresh
[mycluster->user*[printer]]% show
Parameter                      Value
------------------------------ ----------------------------------------------
Group ID                       503
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

(This corresponds roughly to the functionality of the Remove button operation in section 7.1.)

The remove command will remove a user or group. The useful "-r" flag added to the end of the username will remove the user's home directory too. For example, within user mode, the command "remove user maureen -r; commit" will do a removal of 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.
  - the `updateprovisioners` command (Section 6.1.4) 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 6.5.2) can be run to pick up the new image from a provisioner.
- 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.

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 (see 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.

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 will continue 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.

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.
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 5.1 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 6.5.2) then implements the changes to the non-head nodes.

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 will exist for LDAP authentication, which if left in there will allow 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
Node	node-related
Readonly	view-only
CMHealth	health-related

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:

```
[bright51->cert]% help createcertificate
Usage: createcertificate <key-length> <common-name> <organization> <org\
anizational-unit> <locality> <state> <country> <profile> <sys-login> <d\
```

```
ays> <key-file> <cert-file>

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] bright51: New certificate request wit\
h ID: 1
[bright51->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 7.2):

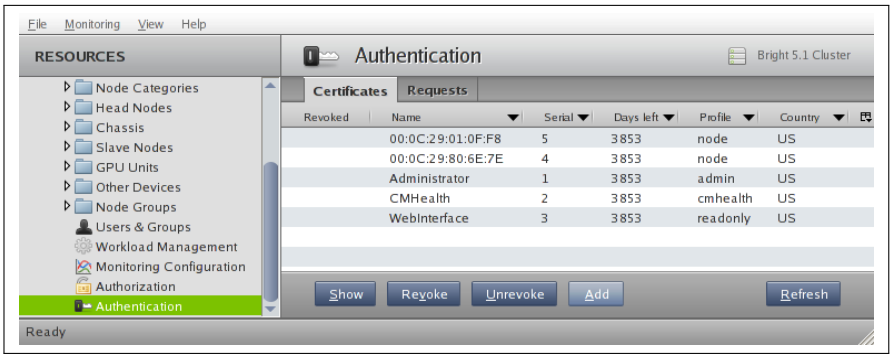

Figure 7.2: `cmgui` Certificates Tab

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 7.3):

Clicking on the `Add` button in Figure 7.3 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 7.4).

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 7.3).

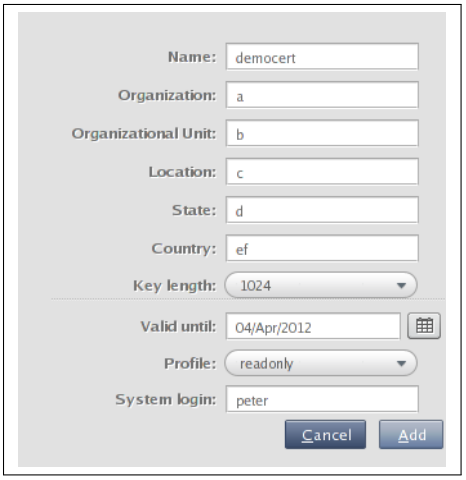

Figure 7.3: `cmgui` Add Certificate And Profile Dialog

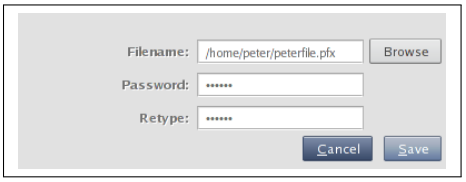

Figure 7.4: `cmgui` Password-protect Key And Save

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 overexploit 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 its estimates of the future availability of the cluster resources that the job is asking for. The workload manager 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.

The details of job submission from a user's perspective are covered in the *User Manual*.

Installing and setting up these choices is covered in this chapter in sections 8.1–8.4. How `cmgui` and `cmsh` are used to view and handle jobs, queues and node drainage is then covered in sections 8.5–8.6. The chapter finishes by giving various examples of how the workload manager can be used in Bright Cluster Manager in section 8.7.

8.1 Workload Managers Choices And Installation

During cluster installation, a workload manager can be chosen (Figure 2.17) for setting up. The choices are:

- None
- Sun Grid Engine (SGE). This is the default.
- Torque v2.4.8 and its built-in scheduler
- Torque v2.4.8 and the Maui scheduler
- Torque v2.4.8 and the Moab scheduler
- PBS Pro

Installation and set up of workload managers can also be done after the Bright Cluster Manager 5.1 installation.

Some workload manager packages are installed by default, others require registration from the distributor before installation.

During installation and set up of a workload manager package, the first time the workload manager is run is when its databases must be initialized. The installation and initialization procedure is described in the installation section for each workload manager in this chapter.

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.

Section 11.2 describes how to configure the cluster to disallow user logins to the nodes.

8.3 Enabling, Disabling, And Monitoring Workload Managers

After the corresponding workload manager package is installed and initialized, a workload manager can be enabled or disabled by the administrator with `cmgui` or `cmsh`. In Bright Cluster Manager 5.1, SGE can even run concurrently with Torque, or with PBS Pro.

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 (`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 the role.

8.3.1 Enabling And Disabling Workload Managers From `cmgui`

The workload manager server is typically enabled from the head node after the corresponding workload manager package is installed and initialized, as described in the installation section for each workload manager in this chapter. Enabling the server is done in `cmgui` by clicking on the

`Head Nodes` folder, selecting the head node, and selecting the `Roles` tab to display the possible roles. After a workload manager server role is chosen and saved (Figure 8.1), the workload manager process automatically starts up.

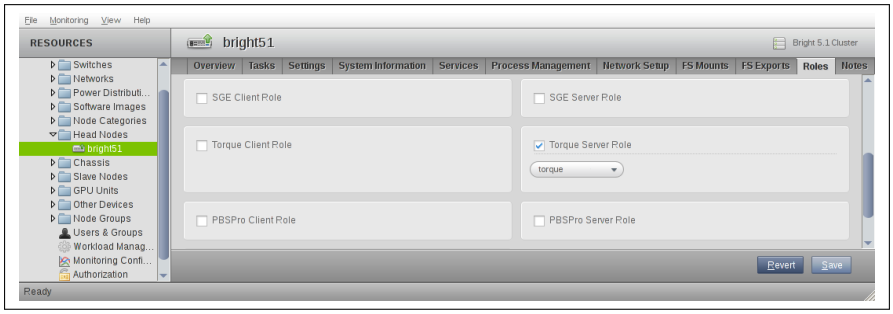

Figure 8.1: Workload Management Role Assignment On A Head Node

Similarly, the workload manager client process can be enabled on a node by having the workload manager client role assigned and saved for that node. The client process then automatically starts up.

While role assignment can be done as described for individual nodes, it is usually more efficient to do role assignment using categories due to the large number of compute nodes in typical clusters.

All non-head nodes are by default placed in the `slave` category. This means that roles assigned to the slave category are automatically assigned to all non-head nodes, unless by way of exception a node is individually configured to use its own role setting instead.

Setting the role in `Node Categories`, for the category `slave` is done by clicking on the `Node Categories` folder, selecting the `slave` category, and selecting the `Roles` tab. The appropriate workload manager client role is then configured (Figure 8.2).

Figure 8.2: Workload Manager Role Assignment By Category For Compute Nodes

Each workload manager client role has options that can be set for `Slots` and `Queues`. "`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 8.5.2 (using `cmgui`) and 8.6.2

(using cmsh).

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 default Slave Nodes folder, then selecting the Roles tab. The appropriate workload manager client role is then configured (Figure 8.3).

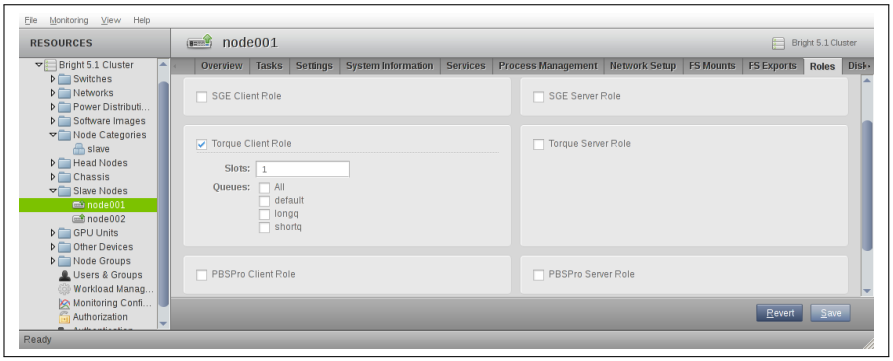

Figure 8.3: Workload Management Role Assignment For An Individual Node

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 8.4). Clicking on the Save button of the tabbed pane saves the displayed setting.

Figure 8.4: Workload Management Role Assignment Toggle States For An Individual Node

8.3.2 Enabling And Disabling Workload Managers From cmsh

In cmsh, assigning a workload manager role to a head node is done in devices mode, using master as the device, and assigning the workload manager role from the roles submode:

Example

```
[root@bright51 ~]# cmsh
[bright51]% device
[bright51->device]% use master
[bright51->device[bright51]]% roles
[bright51->device[bright51]->roles]% assign torqueserver
[bright51->device*[bright51*]->roles*[torqueserver*]]% commit
[bright51->device[bright51]->roles[torqueserver]]%
```

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@bright51 ~]# cmsh
[bright51]% category
[bright51->category]% use slave
[bright51->category[slave]]% roles
[bright51->category[slave]->roles]% assign torqueclient
[bright51->category[slave]->roles*[torqueclient*]]% commit
[bright51->category[slave]->roles[torqueclient]]%
```

For individual nodes, role assignment is done via `devices` mode, using the node name, and assigning a role from the `roles` submode:

Example

```
[root@bright51 ~]# cmsh
[bright51]% device
[bright51->device]% use node001
[bright51->device[node001]]% roles
[bright51->device[node001]->roles]% assign torqueclient
[bright51->device[node001]->roles*[torqueclient*]]% commit
[bright51->device[node001]->roles[torqueclient]]%
```

After workload manager roles are assigned or unassigned on the head and compute nodes, the associated workload manager services automatically start up or stop as appropriate.

Once a setting has been assigned for the workload manager within the `roles` submode, whether within a main mode of `category` or `devices`, the workload manager settings can be handled with the usual object commands introduced in section 3.6.3:

Example

```
[bright51->category[slave]->roles[torqueclient]]% show
Parameter                      Value
------------------------------ ------------------------------------------------
All Queues                     yes
Name                           torqueclient
Queues                         shortq longq
Slots                          4
Type                           TorqueClientRole
[bright51->category[slave]->roles[torqueclient]]% set slots 5
[bright51->category*[slave*]->roles*[torqueclient*]]% commit
[bright51->category[slave]->roles[torqueclient]]%
```

8.3.3 Monitoring The Workload Manager Services

The workload manager services are monitored. Restart attempts are made if the services stop, unless the role for that workload manager service is unassigned. As mentioned previously, role unassignment is how the workload manager service should be disabled.

The daemon service states can be viewed for each node via `cmgui` or `cmsh`.

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 (Figure 10.5).

The job execution daemons run on compute nodes. Their states are viewable by clicking on the `Slave 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@bright51 ~]# cmsh -c "device services node001; status"
        sgeexecd[   UP   ]
[root@bright51 ~]# cmsh -c "device services master; status"
                    ...
             sge[   UP   ]
```

8.4 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 will overwrite them. However, overwriting can be prevented by setting the directive:

`FreezeChangesTo<`*workload manager*`>Config`

in `cmd.conf` (see Appendix C), where *<workload manager>* takes the value of SGE, Torque, or PBS, as appropriate.

A very short guide to some specific workload manager commands that can be used outside of the Bright Cluster Manager 5.1 system is given in Appendix G.

8.4.1 SGE Installation, Initialization, And Configuration

Installing SGE

The SGE package comes with a Bright Cluster Manager 5.1 installation, even if another, or no, workload manager was chosen for configuration and set up. To set it up for use for the very first time, the workload manager server role is initialized, typically on the head node, using the `cm-install-qmaster` script with the "`-q`" option:

```
/cm/shared/apps/sge/current/cm/cm-install-qmaster -q
```

The "`-h`" option displays a help text listing the other options for this script. One of these options is `-c <image>`, where `<image>` is the path to the node image. The `-c` option can be used to place the execution daemon in the node image.

Example

```
/cm/shared/apps/sge/current/cm/cm-install-qmaster -c /cm/images/default-\
image
```

If there are provisioning nodes, the `updateprovisioners` command (section 6.1.4) should be run. The nodes can then simply be rebooted to pick up the new image, or alternatively, to avoid rebooting, the `imageupdate` command (section 6.5.2) can be run to pick up the new image from a provisioner.

After package installation, SGE software components are installed in `/cm/shared/apps/sge/current`, also referred to as `SGE_ROOT`.

SGE documentation is available via `man` pages, documentation in the directory `/cm/shared/docs/sge`, as well as at the SGE website at `http://wikis.sun.com/display/sungridengine/Home`.

Configuring SGE

After installation and initialization, SGE runs with 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 node image for use with SGE the `install_execd` wrapper script under `$SGE_ROOT` is run.

Running SGE

After initialization is carried out as described in the preceding text, SGE can be enabled and disabled as described in sections 8.3.1–8.3.2. 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.

SGE maintains several log files in:

`/cm/shared/apps/sge/current/default/spool`.

Messages from the qmaster daemon are logged to:

`/cm/shared/apps/sge/current/default/spool/messages`.

For the associated compute nodes the execution log exists in:

`/cm/shared/apps/sge/current/default/spool/node<`*number*`>/messages`.

where `node<`*number*`>` is the node name, for example: `node001`, `node002` ...

8.4.2 Torque Installation, Initialization, And Configuration

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.

Installing Torque

The Torque package is installed, but not set up by default on Bright Cluster Manager 5.1. If it is not set up during installation (Figure 2.17), then when it is set up later it must be initialized with the following script, using the "-q" flag:

```
/cm/shared/apps/torque/current/cm/cm-install-torqueserver -q
```

The execution daemon, `pbs_mom` is already in the node images by default and does not need to be installed, even if Maui or Moab are added.

The Torque services can be enabled via role assignment as described in section 8.3.

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.

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`.

Installing The Maui Scheduler

The Maui scheduler source version 3.2.6p21 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-3.2.6p21.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 5.1 using:

```
rpmbuild -bb /usr/src/redhat/SPECS/maui.spec
```

and the installation is done with:

```
rpm -i /usr/src/redhat/RPMS/x86_64/maui-3.2.6p21-44_cm5.1.x86_64.rpm
```

Installing The Moab Scheduler

Moab is installed by default in Bright Cluster Manager 5.1. Once the trial license has expired, a license must be obtained from Adaptive Computing.

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 will 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. It writes logs to `/cm/shared/apps/moab/current/spool/log`.

8.4.3 PBS Pro Installation, Initialization, And Configuration

PBS Pro Installation

PBS Pro can be selected for installation during Bright Cluster Manager 5.1 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, then the following script with the "-q" flag must be run:

```
/cm/shared/apps/pbspro/current/cm/cm-setup-pbspro -q
```

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`.

PBS Pro Configuration

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 Using cmgui With Workload Management

Viewing the workload manager services from cmgui is described in section 8.3.3.

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.

8.5.1 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 8.5).

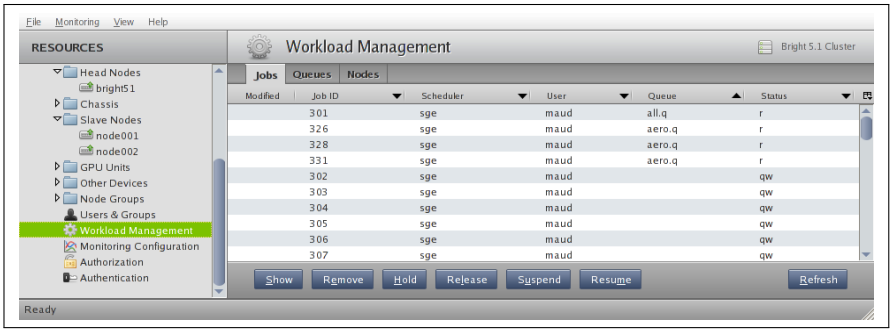

Figure 8.5: Workload Manager Jobs

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.

8.5.2 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 8.6).

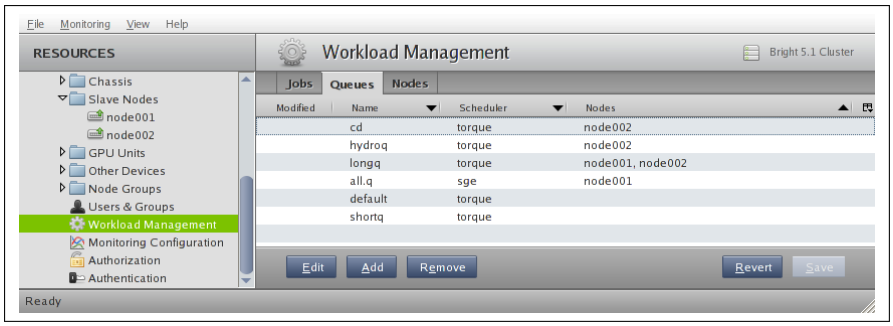

Figure 8.6: Workload Manager Queues

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 8.7 and 8.8).

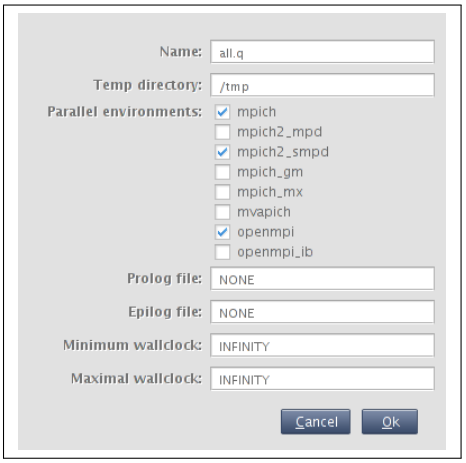

Figure 8.7: Workload Management Queues Edit Dialog For SGE

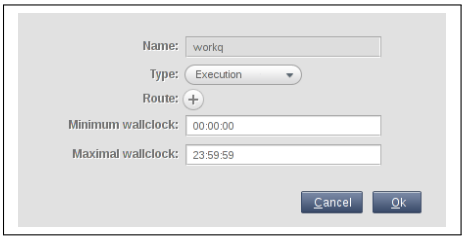

Figure 8.8: Workload Management Queues Edit Dialog For Torque And PBS Pro

In the edit dialog:

– the generic names "`Minimum wallclock`" and "`Maximum wallclock`" correspond respectively to the soft and hard walltimes 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 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 healthcheck 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 node images and their path 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.

8.5.3 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 8.9).

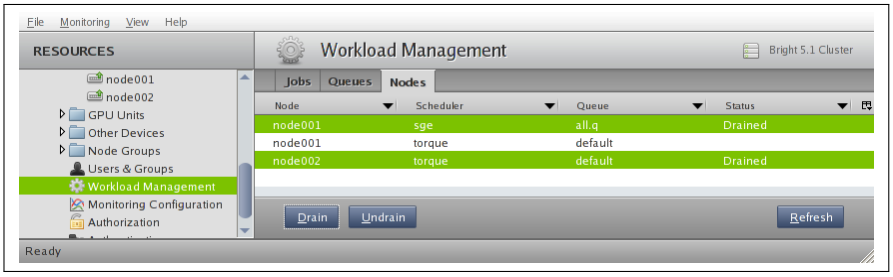

Figure 8.9: Node Drainage

- 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.

8.6 Using cmsh With Workload Management

8.6.1 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

```
[bright51->jobs]% list
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.bright51+ 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:

```
[bright51->jobs]% suspend torque 90.bright51.cm.cluster
Success
[bright51->jobs]% list
Type         jobid        User         Queue        Status
------------ ------------ ------------ ------------ ------------
SGEJob       620          maud         all.q        r
SGEJob       621          maud                      qw
TorqueJob    90.bright51+ 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):

```
[bright51->jobs]% scheduler torque
[bright51->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):

```
[bright51->jobs(torque)]% list; !#no sge jobs listed now - only torque
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
TorqueJob    90.bright51+ maud         hydroq       S
[bright51->jobs(torque)]% resume 90.bright51.cm.cluster; !#torque job
Success
[bright51->jobs(torque)]% list; !#only torque jobs
Type         Job ID       User         Queue        Status
------------ ------------ ------------ ------------ ------------
TorqueJob    90.bright51+ 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:

```
[bright51->jobs(torque)]% show 90.bright51.cm.cluster;
Parameter                      Value
------------------------------ ------------------------------------------
Arguments                      -q hydroq /home/maud/sleeper.sh
Executable
In queue
Job ID                         90.bright51.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                    bright51.cm.cluster:/home/maud/sleeper.sh.e90
Stdout file                    bright51.cm.cluster:/home/maud/sleeper.sh.o90
Submission time                Fri Feb 18 12:49:01 2011
Type                           TorqueJob
User                           maud
```

8.6.2 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 will 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@bright51 ~]# cmsh
  [bright51]% jobqueue
  [bright51->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

```
[bright51->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)

```
[bright51->jobqueue]% listpes
Scheduler    Parallel Environment
------------ ------------------------
sge          make
sge          mpich
...
sge          openmpi_ib
```

- `scheduler` sets the `scheduler` submode

  Example

```
[bright51->jobqueue]% scheduler torque
Working scheduler is torque
[bright51->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 will show only the queues for torque.

Example

```
[bright51->jobqueue]% list
Type         Name
------------ ------------------------
sge          all.q
torque       default
```

```
torque       longq
torque       shortq
[bright51->jobqueue]% scheduler torque
Working scheduler is torque
[bright51->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 3.6.3 work at the top level mode as well as in the scheduler submode:

Example

```
[bright51->jobqueue]% list torque
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[bright51->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
[bright51->jobqueue]% get torque longq maximalruntime
23:59:59
[bright51->jobqueue]%
[bright51->jobqueue]% scheduler torque
Working scheduler is torque
[bright51->jobqueue(torque)]% list
Type         Name
------------ ------------------------
torque       default
torque       longq
torque       shortq
[bright51->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
[bright51->jobqueue(torque)]% get longq maximalruntime
23:59:59
```

```
[bright51->jobqueue(torque)]% use longq
[bright51->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
[bright51->jobqueue(torque)->longq]% get maximalruntime
23:59:59
```

8.6.3 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@bright51 ~]# cmsh
[bright51]% device
[bright51->device]% drainstatus
Node                       Queue                      Status
-------------------------- -------------------------- ----------------
node001                    workq
node002                    workq
[bright51->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:

```
[bright51->device]% drain -c slave; !# for a category of nodes
Node                       Queue                      Status
-------------------------- -------------------------- ----------------
node001                    workq                      Drained
node002                    workq                      Drained
```

The help text for the command indicates the syntax:

```
[root@bright51 ~]# 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 Examples Of Workload Management Assignment

8.7.1 Setting up A New Category And A New Queue For It

Suppose a new node with a GPU is added to a cluster that originally has no nodes with GPUs. This merits a new category, `GPUnodes`, so that administrators can configure more new GPU nodes such as this efficiently. It also merits a new queue, `gpuq` so that users are aware that they can submit GPU-optimized jobs to the GPU queue.

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 8.10).

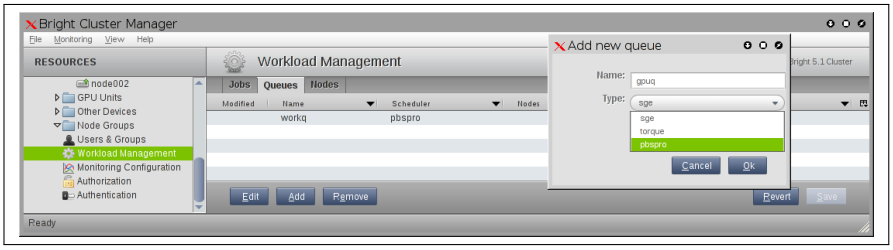

Figure 8.10: Adding A New Queue Via `cmgui`

A useful way to create a new category is to simply clone the old `slave` category over to a new category, and then change parameters in the new category to suit the new machine (Figure 8.11).

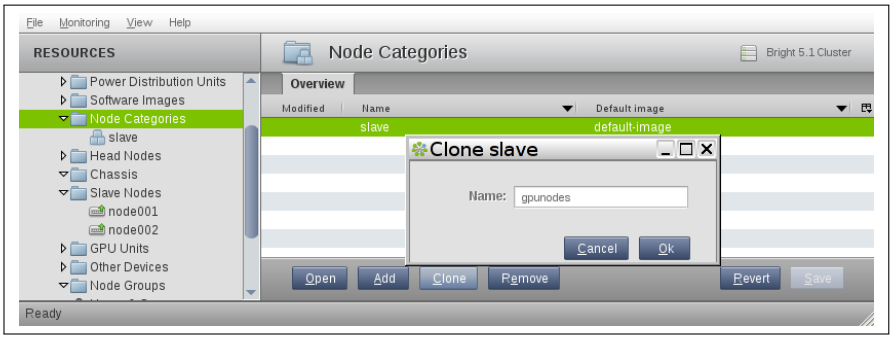

Figure 8.11: Cloning A New Category Via `cmgui`

Having cloned and saved the category, called `gpunodes` in the example of Figure 8.11, 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, `gpunodes`, by going into the `Roles` tabbed pane of that category, selecting the appropriate workload manager client role and queues, and saving the setting (Figure 8.12).

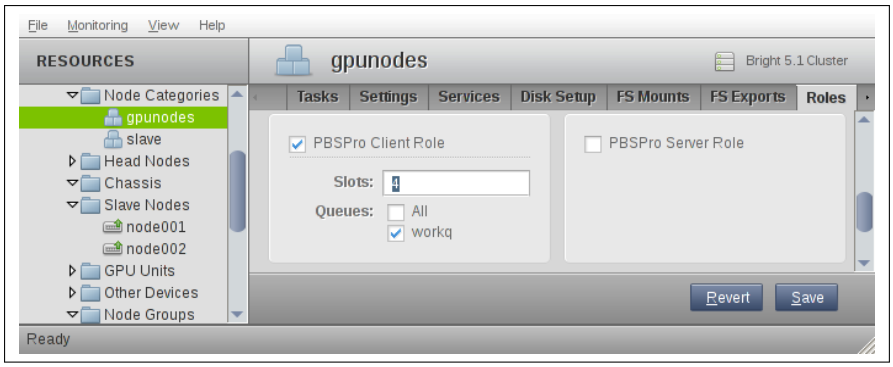

Figure 8.12: Setting A Queue For A New Category Via `cmgui`

Finally, a node in the Slave Nodes folder that is to be placed in the new `gpunodes` category must be placed there by changing the category value of that node in the settings tab (Figure 8.13).

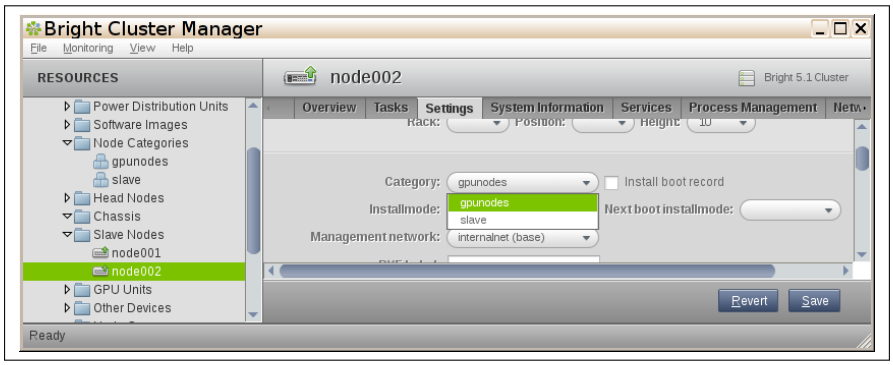

Figure 8.13: Setting A New Category To A Slave Node Via `cmgui`

8.7.2 Setting Up A Prejob Healthcheck

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 8.5.3 and 8.6.3), 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 8.15). In the dialog, the Health Check script value is set to the chosen health check, and the Sampling interval is set to prejob, which automatically sets the Fail action to Drain node. 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.

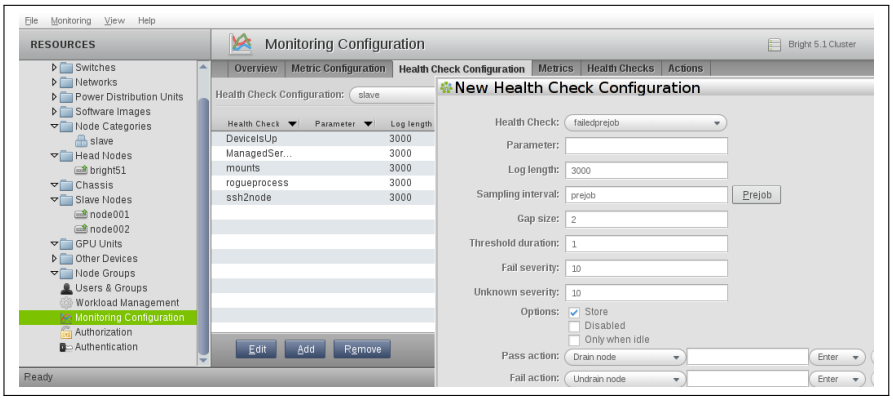

Figure 8.14: Configuring A Prejob Healthcheck Via cmgui

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.

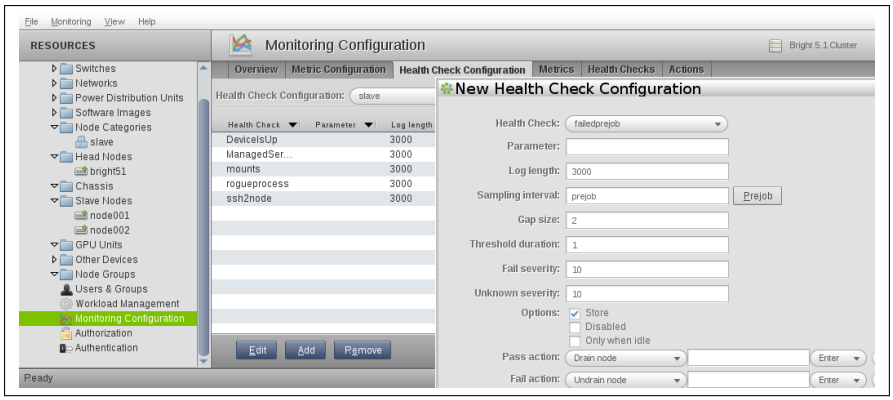

Figure 8.15: Configuring A Prejob Healthcheck Via `cmgui`

9

Software Image Management

Since Bright Cluster Manager is built on top of an existing Linux distribution, the administrator must use distribution-specific utilities for software package management. For Bright Cluster Manager related packages, a separate package management infrastructure has been set up, which is described in this chapter.

9.1 Bright Cluster Manager RPM Packages

Bright Cluster Manager relies on the RPM Package Manager (`rpm`) to manage its software packages. An example of such an RPM package is:

```
mpich-ge-gcc-64-1.2.7-40_cm5.1.x86_64.rpm
```

The file name has the folowing 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* (`40`) is the revision number of the package
- *x.y* (`5.1`) 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

More information about the RPM Package Manager is available at `http://www.rpm.org`.

9.2 Installing & Upgrading Packages

Once Bright Cluster Manager has been installed, Bright Cluster Manager software packages can be fetched/installed/upgraded by fetching/installing/upgrading the corresponding RPM packages using the `rpm` command-line utility. However, a more convenient way of managing packages is to use the YUM tool. For example, the following command lists all available packages:

```
yum list
```

The following command installs a new package:

```
yum install packagename
```

All installed packages are updated with:

```
yum update
```

Bright Computing maintains YUM repositories at:

```
http://updates.brightcomputing.com/yum
```

and updates are fetched by YUM in Bright Cluster Manager from there by default.

Accessing the YUM repositories manually (i.e. not through YUM) requires a username and password. Authentication credentials are provided upon request. For more information on this, support@brightcomputing.com should be contacted.

YUM uses caches to speed up its operations. Occasionally these caches may need flushing, to make YUM fetch fresh copies of all index files associated with a repository. This is done with:

```
yum clean all
```

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. The Bright Computing public key is also listed in Appendix B.

The first time YUM is 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", the administrator 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 also be imported into the local RPM database directly, the following command is used:

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-cm
```

9.3 Managing Packages Inside Images

Installing or updating packages inside a node image can be handled with rpm or yum.

The rpm command supports the --root flag. To install an RPM inside the default node image, the following command is used:

```
rpm --root /cm/images/default-image -ivh /tmp/libxml2-2.6.16-6.x86_64.rpm
```

Similarly, YUM uses the --installroot flag. For example, all packages in the image are updated with:

```
yum --installroot=/cm/images/default-image update
```

With the chroot command, the same result is accomplished by first chrooting into an image, and subsequently executing yum or rpm commands without --root or --installroot arguments.

The chroot command may also be used to install software which is not supplied as an RPM package into a node image. For 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
```

While `chroot` can be a useful tool for installing software into a node image, it can have issues if it starts up daemons in the image.

For example, installation scripts that stop and re-start a system service during a source install 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 RPM packages are however designed to install under chroot without issues.

9.4 Kernel Updates

In general it is a good idea to be careful about updating the kernel on a head node or in a node image. This is particularly true when custom kernel modules are being used that were compiled against a particular kernel.

To prevent an automatic update of a package, it is listed on the `yum` command line using the `--exclude` flag. To exclude the kernel from the list of packages that should be updated, the following command can thus be used:

```
yum --exclude kernel update
```

If a package (e.g. `kernel`) is to be excluded permanently from all YUM updates, it can be appended to the (space-separated) exclude list option for a repository configuration. Repository configuration files are located in the `/etc/yum.repos.d` directory.

An updated kernel in a node image is not used 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.

To enable the updated kernel from `cmsh`, the `softwareimage` mode is used. The `kernelversion` property of a specified software image is then `set`.

9.5 Creating Custom Software Images

By default, the node 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.

Creating a working node 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.

9.5.1 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 (actually taking the step a little further towards the end result) the archive can be a fully expanded archive file tree.

For example, a base distribution `tar.gz` archive (here it is `/cm/image/new-image/grab.tgz`) can be created from the base host `basehost64` as follows:

```
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/30-net_persistent_names.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 /" > /cm/image/new-image/grab.tgz
```

Or alternatively, a fully expanded archive file tree can be created from `basehost64` by `rsync`ing to an existing directory (here it is `/cm/image/new-image`):

```
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/30-net_persistent_names.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.

9.5.2 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:

```
USAGE: cm-create-image [-x <base tar>] -b <image> [-d] [-n <name>]

<base tar>  Path to gzipped base tar file (tar.gz, tgz)
<image>     Path to the directory containing base distribution image
            or empty directory where base tar should be extracted to.
```

```
<name>      Name of software image (by default, it will be the base
            name of the directory specified with option -b <image>)
-d          Install distribution packages
```

Examples of usage are:

Example

If the base distribution is in `/tmp/BASEDIST.tar.gz`, then the command that creates the node image `/cm/image/new-image` is:

```
cm-create-image -x /tmp/BASEDIST.tar.gz -b /cm/image/new-image
```

Example

If the contents of `basehost64` were `rsync`ed to an existing directory `/cm/image/new-image`, then no extraction is needed, and the command to create a node image is then simply:

```
cm-create-image -b /cm/image/new-image
```

This creates an image with the name `new-image` in the CM-daemon database.

Example

The same image with a different name, `bio-image`, can be created with the `-n` option:

```
cm-create-image -b /cm/image/new-image -n bio-image
```

Example

The `-d` flag is used to make the utility install distribution-specific packages into the image:

```
cm-create-image -b /cm/image/new-image -n bio-image -d
```

Package selection files are used from `/cm/local/apps/cluster-tools/config/`. If the base distribution of the node 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 node (`image="master"`). Therefore non-head node packages must have the `image="slave"` attribute.

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).

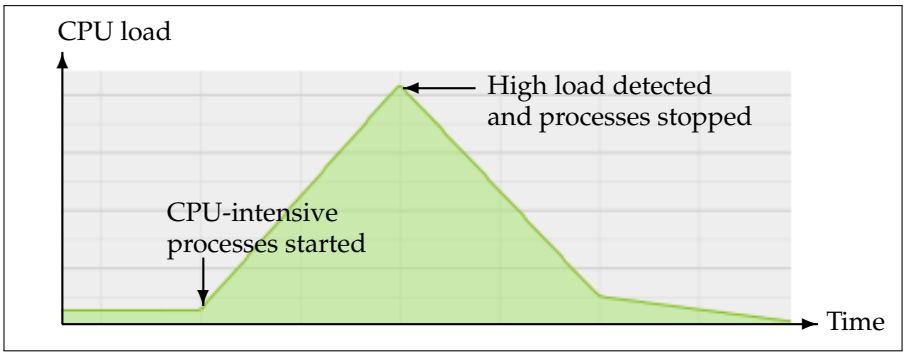

Figure 10.1: Monitoring Basic Example: CPU-intensive Processes Started, Detected And Stopped

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:

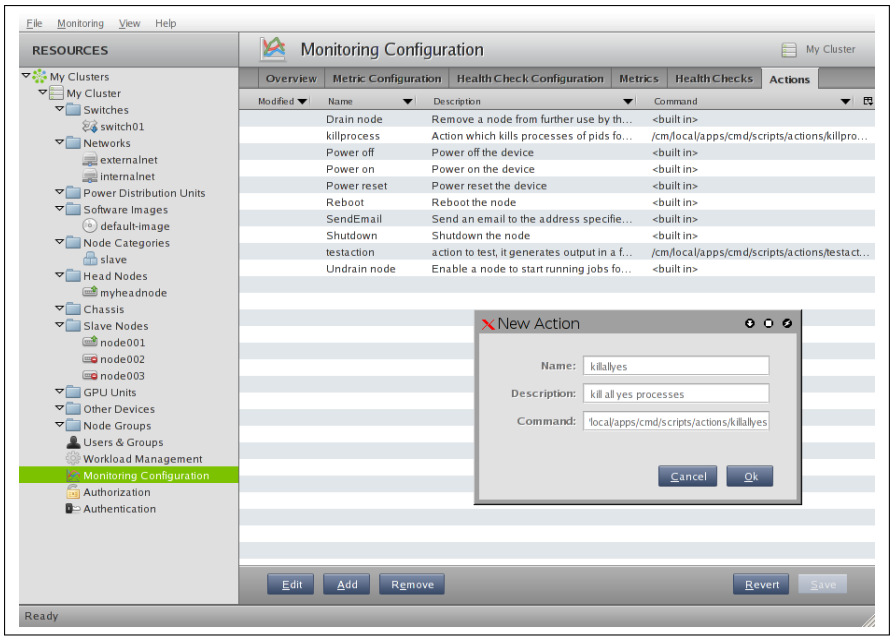

Figure 10.2: `cmgui` Monitoring Configuration: Adding An Action

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 Thresholds` 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`

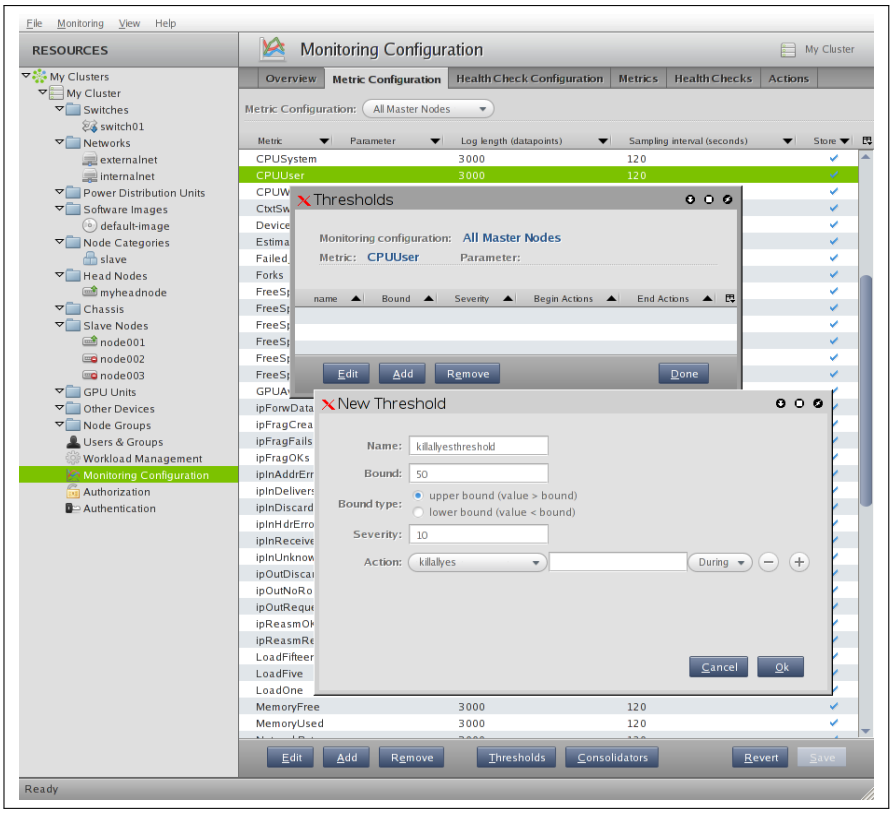

Figure 10.3: `cmgui` Monitoring Configuration: Setting A Threshold

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.

The Result

In the above, 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 basic example configured above therefore monitors if `CPUUser` on the head node has crossed 50% every 120s.

If `CPUUser` is found to have entered 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%.

Note that 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. `cmgui`'s `New Threshold` dialog (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 (see 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 store a 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 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. By default it 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 (see 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.

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.

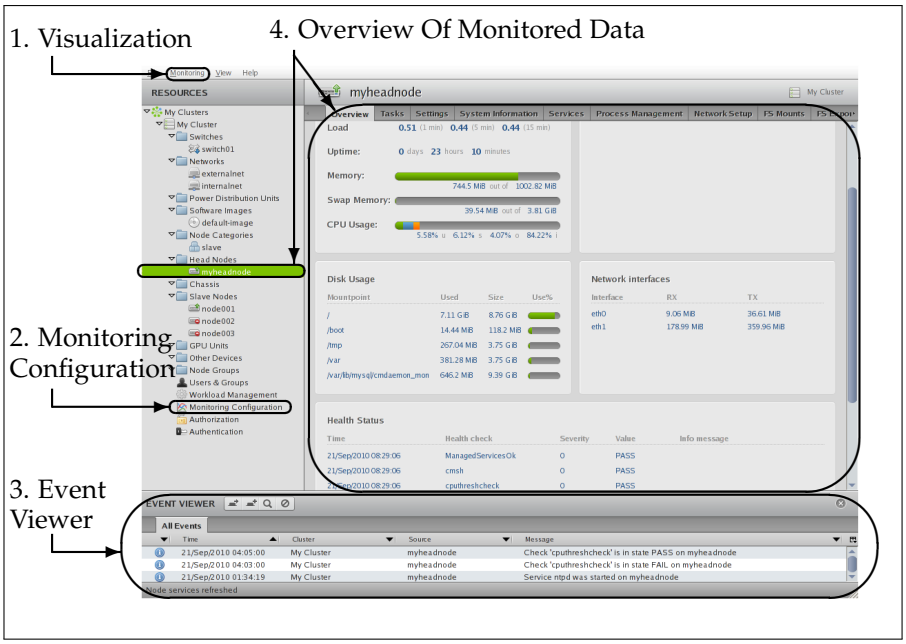

Figure 10.4: `cmgui` Conceptual Overview - Monitoring Types

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 here covers the layout:

There are 4 monitoring-related viewing areas for the user in `cmgui` (Figure 10.4).

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 later 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, sampling methods, data storage and threshold actions are configured and viewed.

   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.

   Neighbouring 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 allows the administrator to `Start`, `Stop`, `Restart` or `Reload` the service with the corresponding button if the backend `init.d` script for the service supports these commands. The `Reset` button is used to clear a "Failed" state of a service as seen by the monitoring system (the monitoring system sets the state of a service to the failed state if 20 restarts of the service in a row fail).

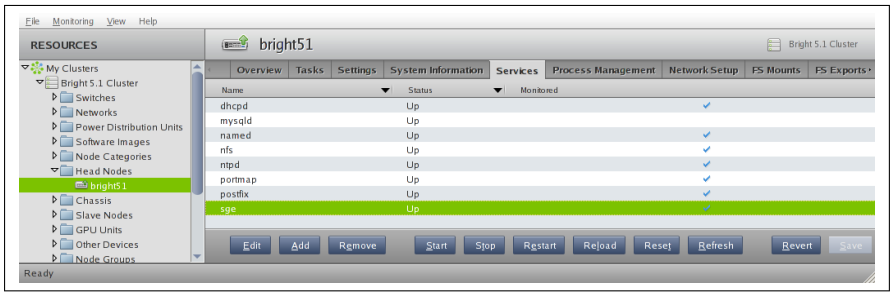

Figure 10.5: `cmgui`: Device Services Tab

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 are selected and studied according to configurable parameters with it. 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 moni-

toring 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 section 10.7.

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 will 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.

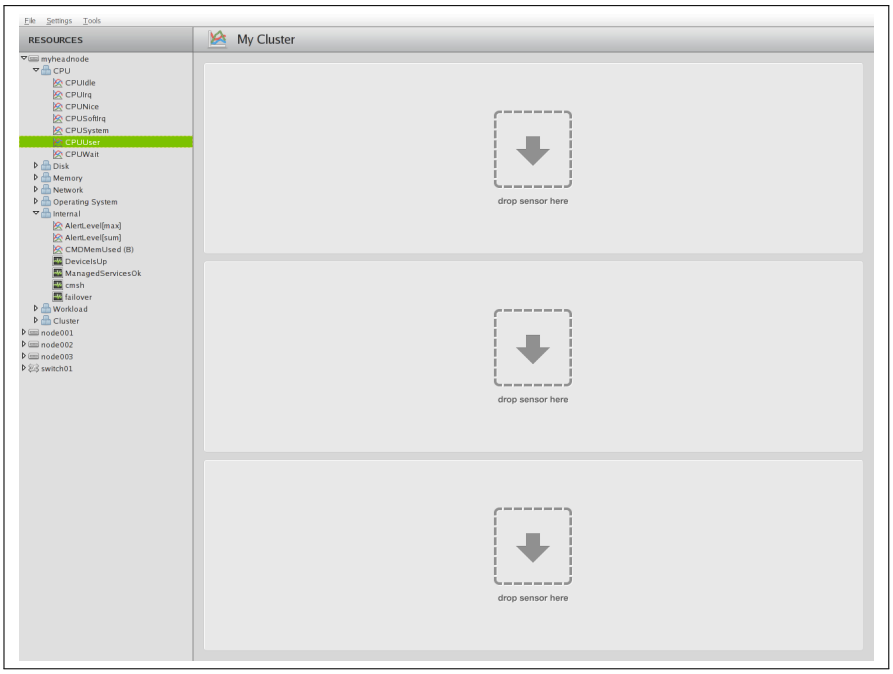

Figure 10.6: cmgui Monitoring Window: Resources View

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.

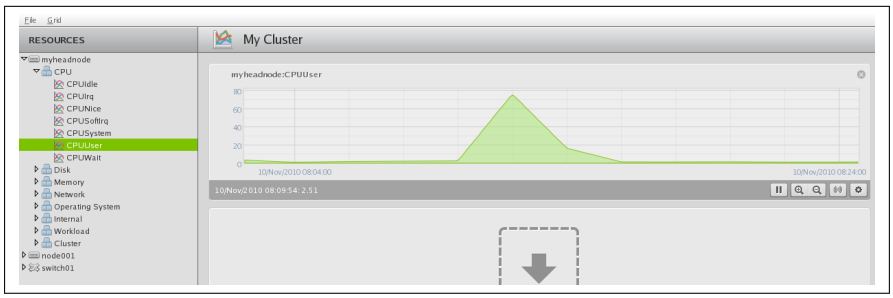

Figure 10.7: `cmgui` Monitoring Window: Graph Display Pane

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 10.3.4.)

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.
   - **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 10.3.3.
   - **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 (see bullet point above) or via mouse gestures (see section 10.3.3) are done over all the other graph display panes too so that their x-ranges match. This is useful for large numbers of nodes.

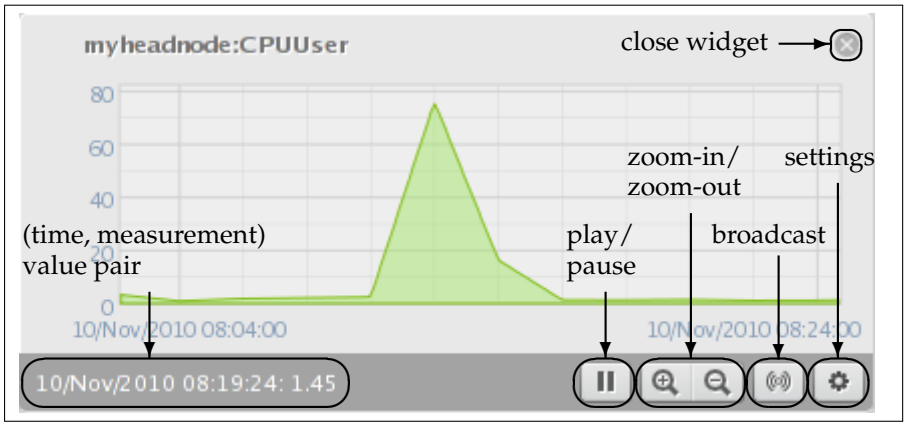

Figure 10.8: Graph Display Pane: Features

- **settings**: Clicking on this button opens a dialog window to modify certain aspects of the graph. The settings dialog is discussed in section 10.3.4.

4. **Any number of graph display panes** are laid out by using the `Grid` menu option of the main `Monitoring Pane` (Figure 10.6).

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 10.9, where both graphs lie on the same axis in the top pane.

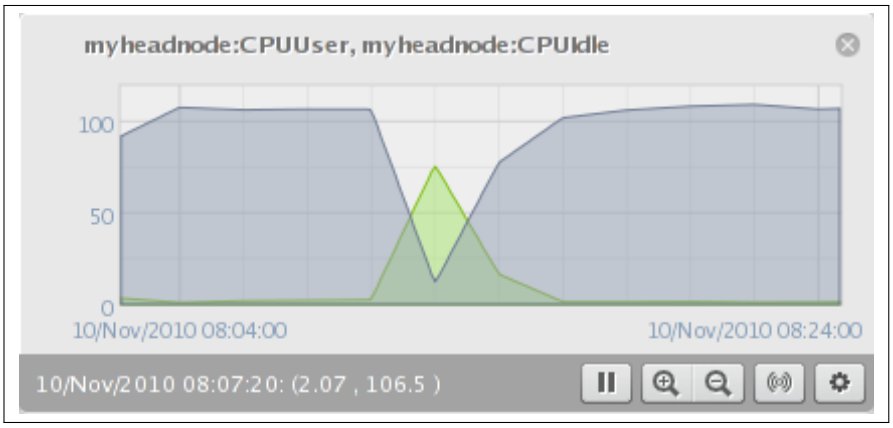

Figure 10.9: Graph Display Pane: Multiple Graphs On One Pane

10.3.3 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 10.10):

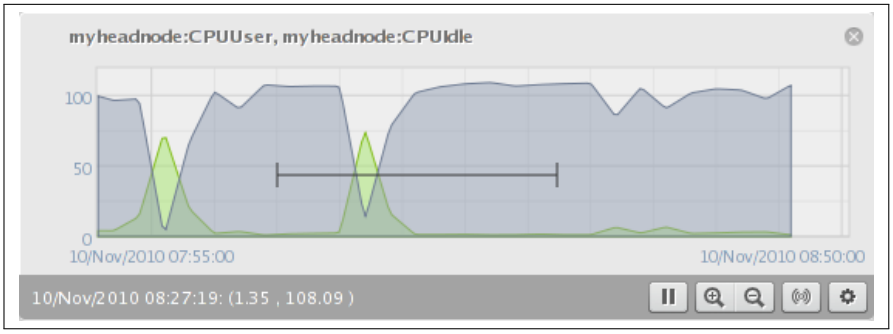

Figure 10.10: Graph Display Pane: X-axis Zoom Start

The x-axis range covered by this line is zoomed in on when the mouse button is released (Figure 10.11):

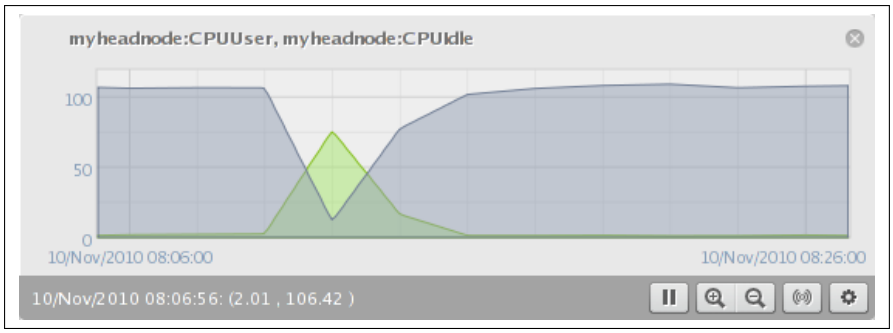

Figure 10.11: Graph Display Pane: X-axis Zoom Finish

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 10.12):

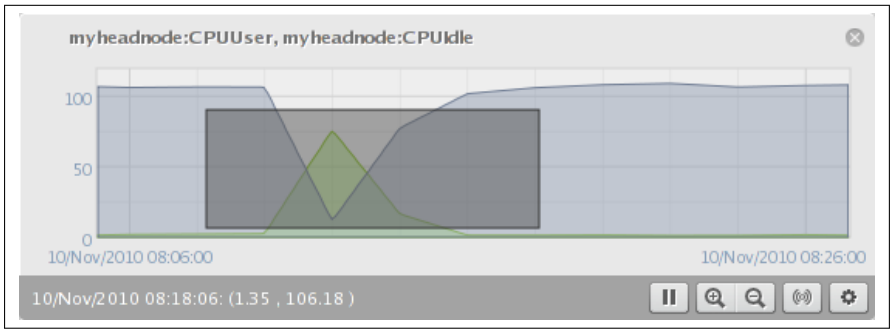

Figure 10.12: Graph Display Pane: Box Zoom Start

This is zoomed into when the mouse button is released (Figure 10.13):

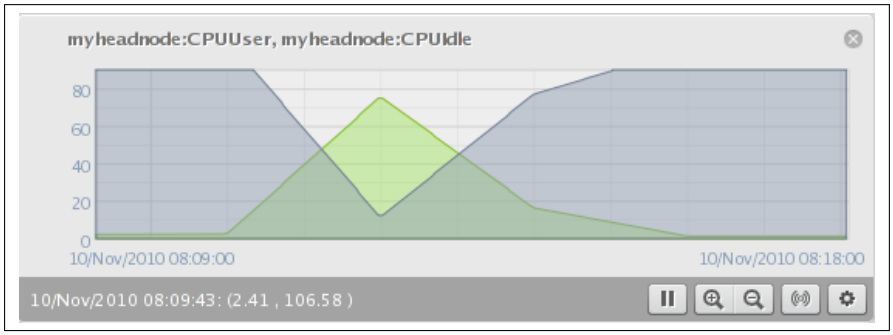

Figure 10.13: Graph Display Pane: Box Zoom Finish

10.3.4 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 10.14):

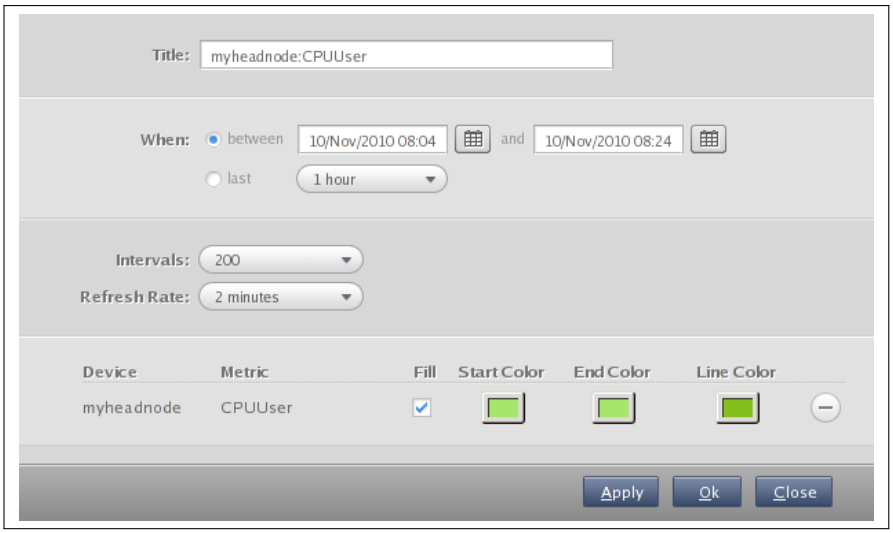

Figure 10.14: Graph Display Pane Settings Dialog

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 10.15):

- `Overview` (displays as the default)
- `Metric Configuration`
- `Health Check Configuration`
- `Metrics`

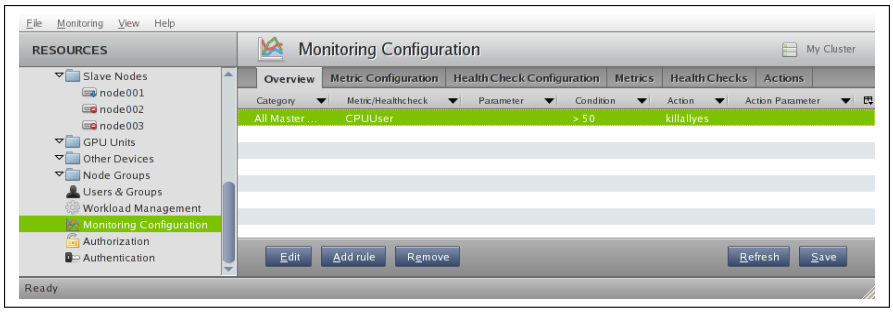

Figure 10.15: `cmgui` Monitoring Configuration Tabs

- `Health Checks`
- `Actions`

The tabs are now discussed in detail.

10.4.1 The Overview Tab

The `Overview` tab of Figure 10.15 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 10.15, 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 manipulation 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 10.16 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.

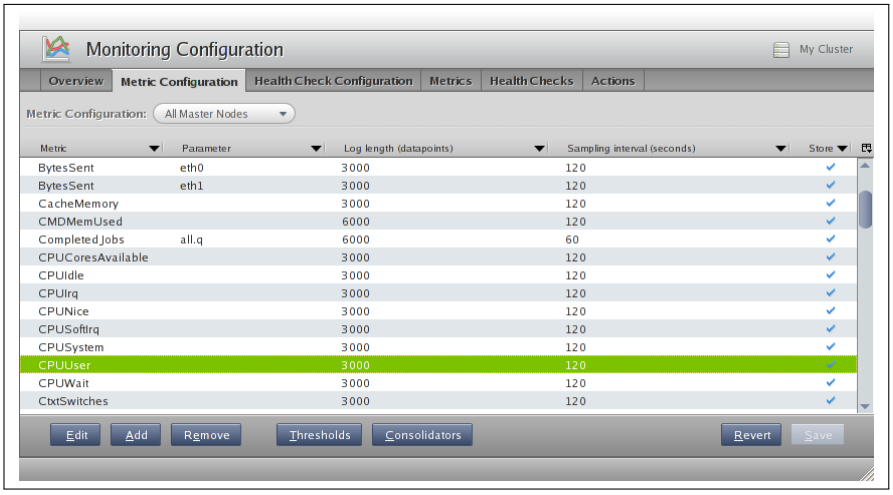

Figure 10.16: `cmgui` Monitoring: Metric Configuration Display After Category Selection

With the screen displaying a list of metrics as in Figure 10.16, 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: The Main Tab's Edit And Add Options

The `Metric Configuration` tab of Figure 10.16 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 (Figure 10.17):

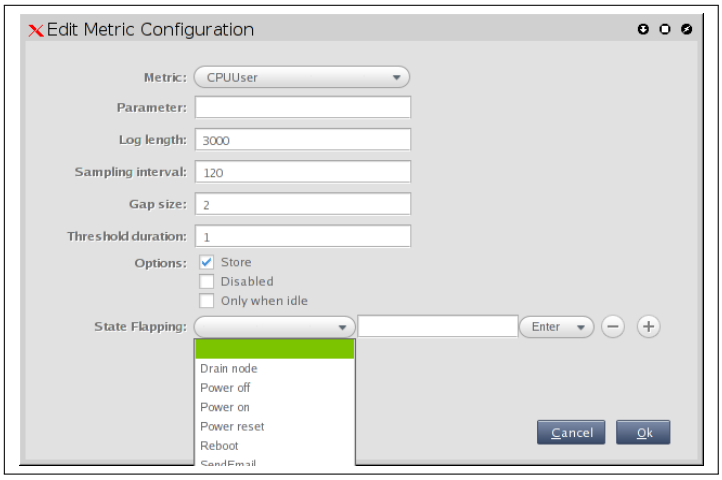

Figure 10.17: `cmgui` Monitoring: Metric Configuration Edit Dialog

- `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 will burden the system less this way.

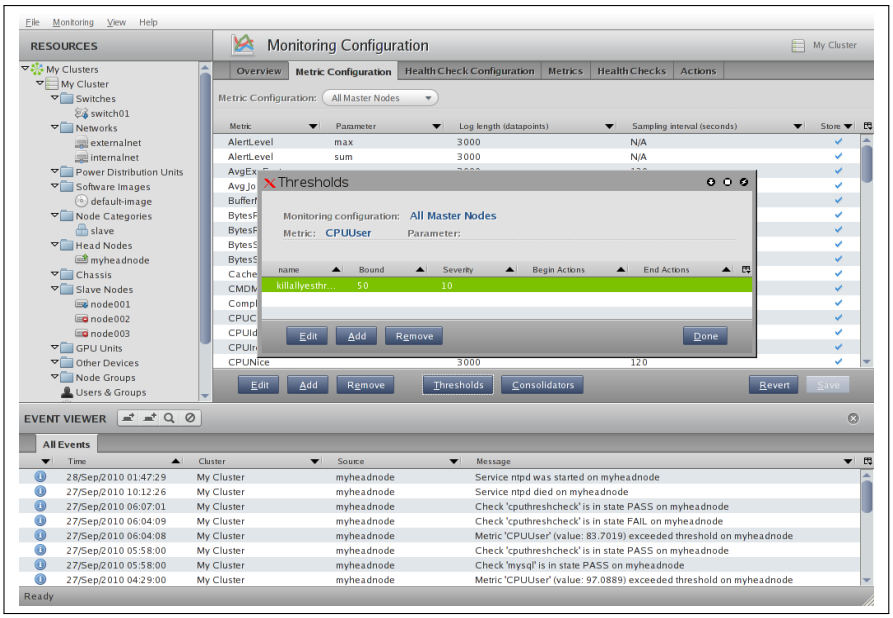

Figure 10.18: `cmgui` Monitoring: Thresholds Display

- `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: Thresholds Options

The `Metric Configuration` tab of Figure 10.16 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 10.18, 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

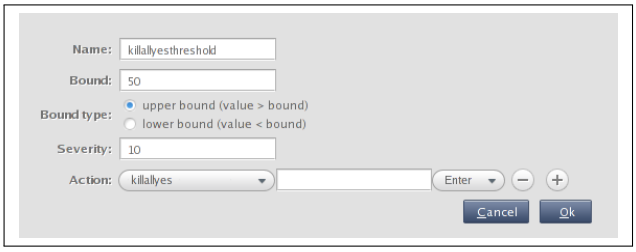

Figure 10.19: `cmgui` Metric Configuration: Thresholds Edit Dialog

new threshold to the list.

The `Edit` and `Add` dialogs for a threshold prompt for the following values (Figure 10.19):

- `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: Consolidators Options

The `Metric Configuration` tab of Figure 10.16 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 10.20).

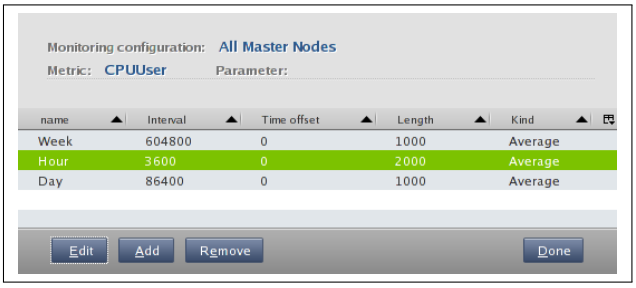

Figure 10.20: `cmgui` Metric Configuration: Consolidators Display

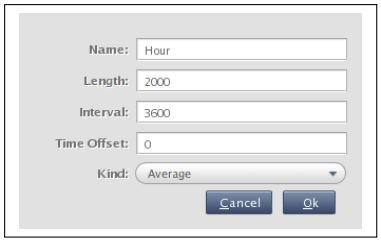

Figure 10.21: `cmgui` Metric Configuration: Consolidators Edit Dialog

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 10.21):

- `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 $t_{\texttt{raw gone}}$, which is how many seconds into the past the raw log data point is removed, is given by:

  $$t_{\texttt{raw gone}} = (\texttt{Log length})_{\texttt{metric}} \times (\texttt{Sampling interval})_{\texttt{metric}}$$

  This value is also the default consolidation time, because the consolidated data values are normally presented from $t_{\texttt{raw gone}}$ 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, $t_{consolidation}$, is then given by:

$$t_{consolidation} = t_{raw\ gone} + \text{Time Offset}$$

The monitoring visualization graphs then show consolidated data from $t_{consolidation}$ 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

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 10.22 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 10.22, 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:

[1] For completeness: the time $t_{consolidation\ gone}$, 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 $t_{raw\ gone}$:

$$t_{consolidation\ gone} = (\text{Log length})_{consolidation} \times (\text{Sampling interval})_{consolidation}$$

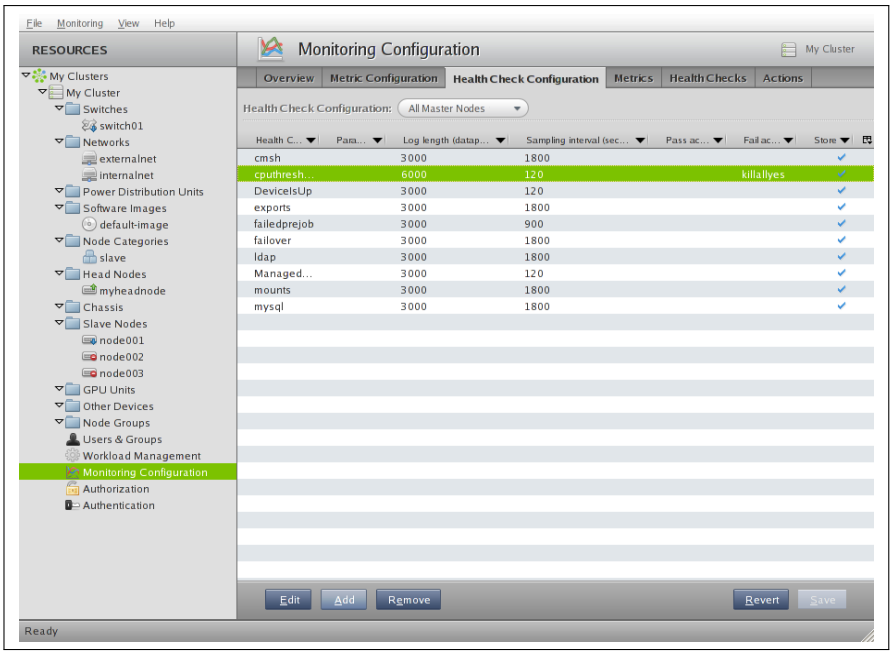

Figure 10.22: `cmgui` Monitoring: Health Check Configuration Display After Category Selection

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: The Main Tab's Edit And Add Options

The `Health Check Configuration` tab of Figure 10.22 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 10.23):

- `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.

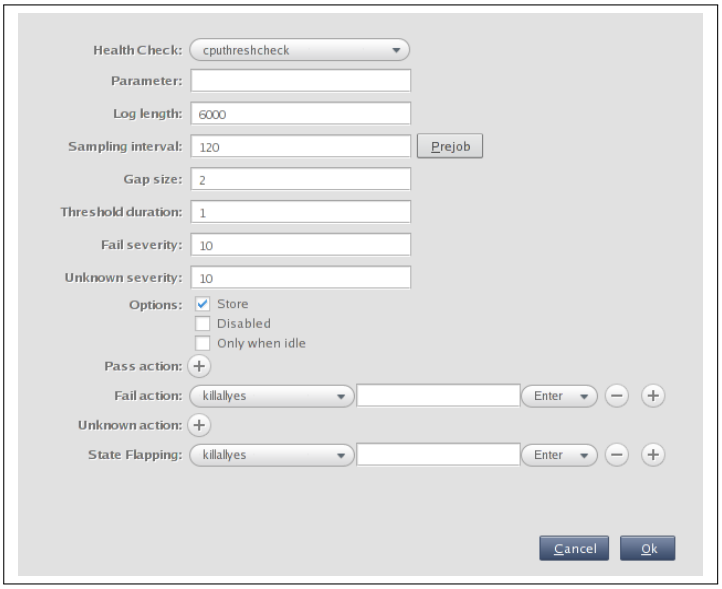

Figure 10.23: `cmgui` Monitoring: Health Check Configuration Edit Dialog

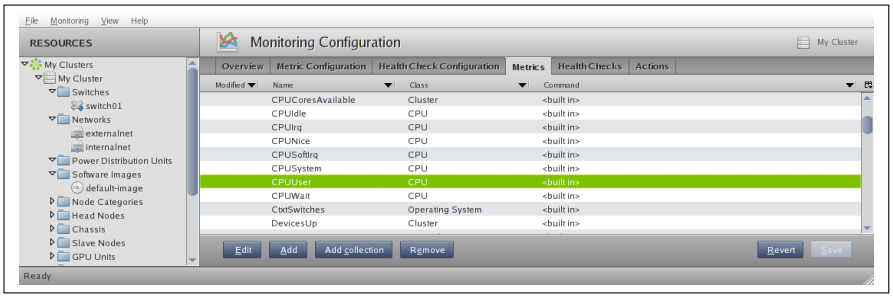

Figure 10.24: `cmgui` Monitoring: Main Metrics Tab

10.4.4 Metrics

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.

Metrics: The Main Tab's Edit And Add Options

The `Metrics` tab of Figure 10.24 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 10.25):

- `Name`: the name of the metric.

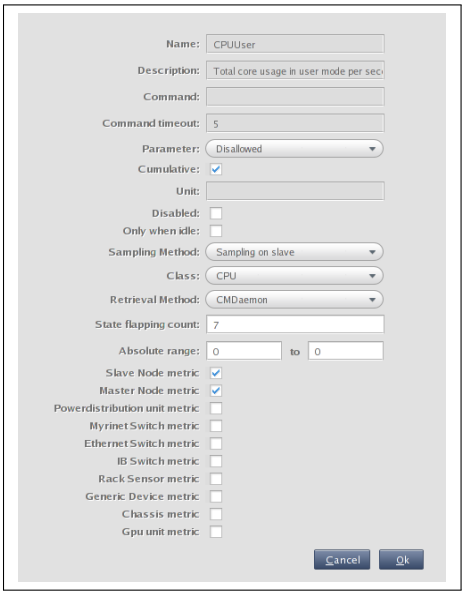

Figure 10.25: `cmgui` Monitoring: Main Metrics Tab, Edit Dialog

- `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.
  - `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 slave`: The non-head node samples the metric itself.
- `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 (see fourth bullet point from here further on), which is a property of where the metrics are applied.

- 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.
- Which device category the metric is configured for, with choices out of:
  - Slave Node
  - Master Node
  - Power Distribution Unit
  - Myrinet Switch
  - Ethernet Switch
  - IB Switch
  - Rack Switch
  - Generic Switch
  - Chassis
  - GPU Unit

  These options should not be confused with the class that the metric belongs to (see fourth bullet point from here earlier on), which is the property type of the metric.

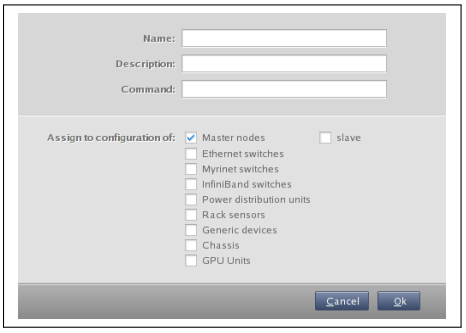

Figure 10.26: `cmgui` Monitoring: Main Metrics Tab, `Add collection` Dialog

Metrics: The Main Tab's Add Collection Option

The `Add Collection` button opens a dialog which is used to create a *metric collection* (Figure 10.26). 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".
- 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

The `Health Checks` tab lists available health checks (Figure 10.27). 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

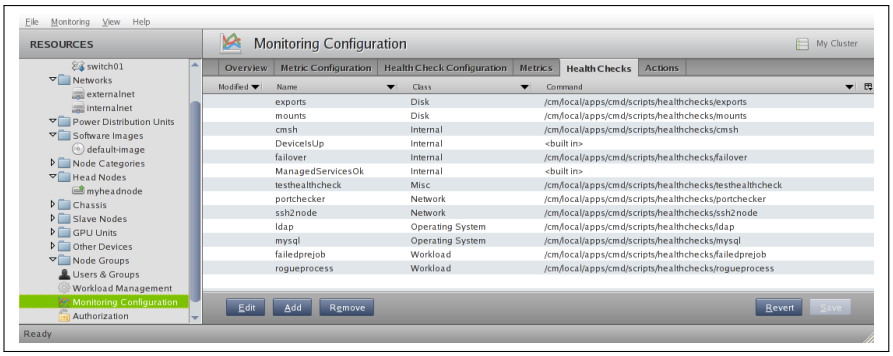

Figure 10.27: `cmgui` Monitoring: Main Health Checks Tab

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

The `Actions` tab lists available actions (Figure 10.28) 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 can also be set to run from health check configuration action launcher options as described in section 10.4.3.

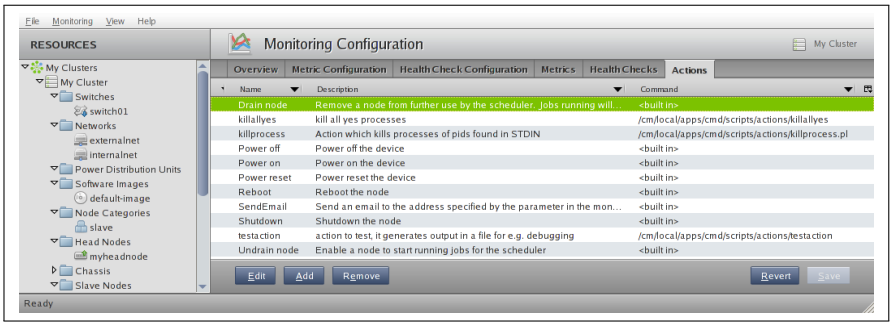

Figure 10.28: `cmgui` Monitoring: Main Actions Tab

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

These views are set up under the `Overview` tab for various devices within a resource.

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 within a resource is clicked on in a `cmgui` session.

Of the devices, the cluster, head node and regular node resources 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). The logs can be handled and viewed in several ways.

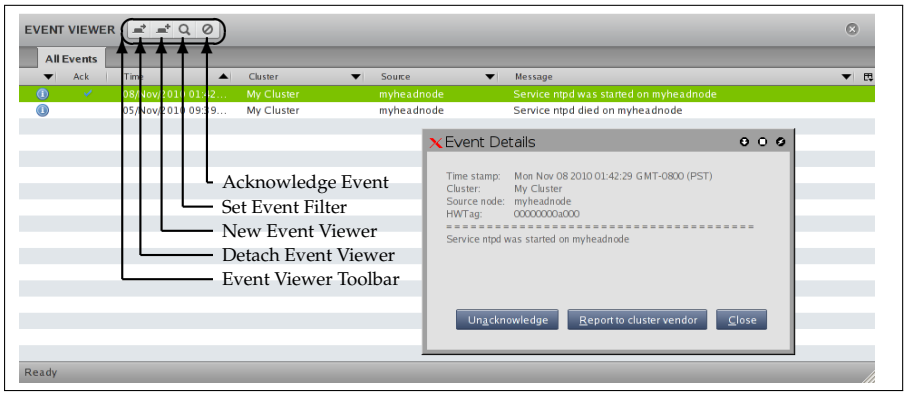

Figure 10.29: `cmgui` Monitoring: Event Viewer Pane

Double clicking on an event row starts up an `Event Details` dialog (Figure 10.29), with buttons to:

- `Acknowledge` or `Unacknowledge` the event, as appropriate. Clicking on `Acknowledge` will remove the event from the event view unless the `Show Acknowledged` checkbox has been checked. Any visible acknowledged events will 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 10.29) 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 10.30). Filters can be customized according to acknowledgement status, time periods, cluster, nodes or message text. The filter settings can be saved for later reloading.
- `set event viewer filter dialog`: Adjusts an existing filter with a similar dialog to the `new event viewer filter dialog`.

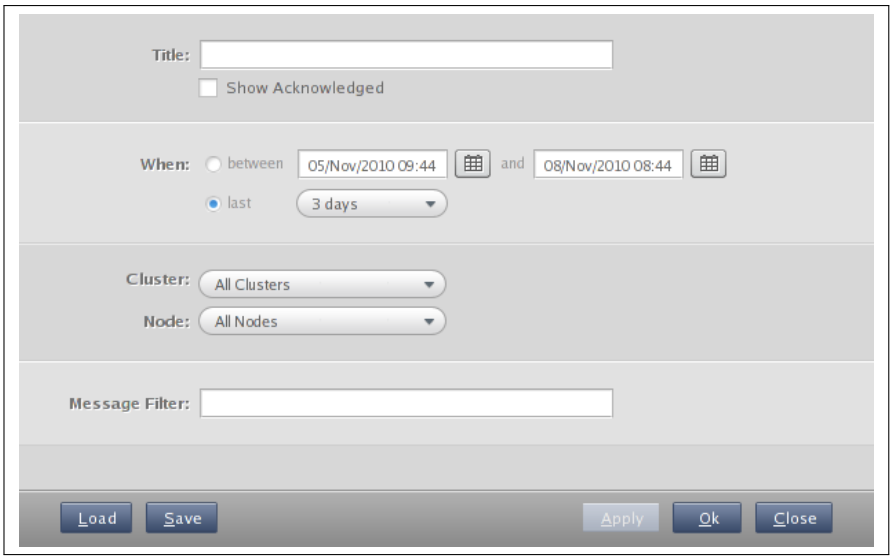

Figure 10.30: `cmgui` Monitoring: Event Viewer Filter Dialog

- `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 Monitoring Modes With Cmsh

This section covers how to use `cmsh` to configure monitoring. The `cmsh` `monitoring` mode 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 `dumpmetricdata` and `latestmetricdata` into graphing utilities. These techniques will not be covered in this chapter.

Familiarity is assumed with handling of objects as described in the introduction to working with objects (section 3.6.3). 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 will be relatively detailed, and will often be referred to by the other sections.

10.7.1 Cmsh Monitoring: Actions

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 3.6.3). 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.

cmsh monitoring actions: 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.+
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
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 will display the following:

Example

```
drainnode    killprocess  poweron      reboot       shutdown     undrainnode
killallyes   poweroff     powerreset   sendemail    testaction
```

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]%
```

cmsh monitoring actions: 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 will drop into the context of an already existing object (i.e. it will "use" 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 3.6.3). So, for example, commit will only succeed 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 Cmsh Monitoring: Healthchecks

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 3.6.3). 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.

cmsh monitoring healthchecks: 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

```
[myheadnode->monitoring->healthchecks]% format name:18 command:55
[myheadnode->monitoring->healthchecks]% list
name (key)         command
------------------ -------------------------------------------------------
DeviceIsUp         <built-in>
ManagedServicesOk  <built-in>
cmsh               /cm/local/apps/cmd/scripts/healthchecks/cmsh
exports            /cm/local/apps/cmd/scripts/healthchecks/exports
...
```

The `format` command, introduced in section 3.6.3, 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
Only when idle                 no
Parameter permissions          disallowed
Sampling method                samplingonmaster
State flapping count           7
Timeout                        5
Valid for                      slave,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
cmsh               failover           mounts             ssh2node
deviceisup         cpucheck           mysql              testhealthcheck
exports            ldap               portchecker
failedprejob       managedservicesok  rogueprocess
[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]%
```

cmsh Monitoring healthchecks: 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 3.6.3). 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 Cmsh Monitoring: Metrics

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 3.6.3). 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.

cmsh monitoring metrics: 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
Measurement Unit
Name                           CPUUser
Only when idle                 no
Parameter permissions          disallowed
Retrieval method               cmdaemon
Sampling method                samplingonslave
State flapping count           7
Timeout                        5
Valid for                      slave,master
maximum                        <range not set>
minimum                        <range not set>
[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 122 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
```

cmsh monitoring metrics: 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 3.6.3). 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 Cmsh Monitoring: Setup

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               <13 in submode>        <1 in submode>
MasterNode             <88 in submode>        <9 in submode>
MyrinetSwitch          <0 in submode>         <0 in submode>
PowerDistributionUnit  <5 in submode>         <1 in submode>
RackSensor             <2 in submode>         <1 in submode>
slave                  <25 in submode>        <5 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 healthcheck configuration `healthconf` submodes are used:

cmsh monitoring setup: 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                 ibswitch                racksensor
ethernetswitch          masternode              slave
genericdevice           myrinetswitch
gpuunit                 powerdistributionunit
```

A category can be chosen with the `use` command, and `show` will show 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           <9 in submode>
Metric configuration           <88 in submode>
Normal pickup interval         180
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          all.q              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
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.

thresholds If, continuing on from the above example, the thresholds submode is entered, then the `list` command will list 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
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 3.6.3 will work as expected at this `cmsh` prompt level: `add` and `remove` will add and remove a threshold; `set`, `get`, and `clear` will set and get values for the parameters of each threshold; `refresh` and `commit` will revert and commit changes; `use` will "use" the specified threshold, making it the default for commands; `validate` applied to the threshold will check if the threshold object has sensible values; and `append` and `removefrom` will 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 10.19 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 will send an e-mail to the root mail account. A `get` command can be run to see the values for the threshold actions:

Example

```
[...metricconf[CPUUser]->thresholds*]% append killallyesthreshold actions sendemail root
[...metricconf[CPUUser]->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 will run when the threshold condition is met.

consolidators If, continuing on with the preceding example, the `consolidators` mode is entered, then the `list` command will list 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 will show 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
```

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 3.6.3 will work as expected at this `cmsh` prompt level: `add` and `remove` will add and remove a consolidator; `set`, `get`, and `clear` will set and get values for the parameters of each consolidator; `refresh` and `commit` will revert and commit changes; `use` will "use" the specified consolidator, making it the default for commands; and `validate` applied to the consolidator will check if the consolidator object has sensible values.

cmsh monitoring setup: 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
cmsh                                        1800
exports                                     1800
failedprejob                                900
failover                                    1800
ldap                                        1800
mounts                                      1800
mysql                                       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 will 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 will be continued and completed, and on the way how to use the health checks configuration object methods to do this will be 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
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 10.23 are explained.

The object manipulation commands introduced in section 3.6.3 will work as expected at the `healthconf` prompt level in the example above: `add` and `remove` will add and remove a health check; `set`, `get`, and `clear` will set and get values for the parameters of each health check; `refresh` and `commit` will revert and commit changes; `use` will "use" the specified health check, making it the default for commands; and `validate` applied to the health check will check if the health check object has sensible values; and `append` and `removefrom` will 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 10.23 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 carried 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*]]%
```

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 parallel shell allows `bash` commands to be run on a group of nodes simultaneously.

- In `cmsh`, it is run from `device` mode by using the `pexec` command:

  Example

  ```
  [bright51->device]% pexec -n node001,node002 "cd ; ls"
  ```

- In `cmgui`, it is executed from the `Parallel Shell` tab after selecting the cluster from the resource tree (Figure 11.1):

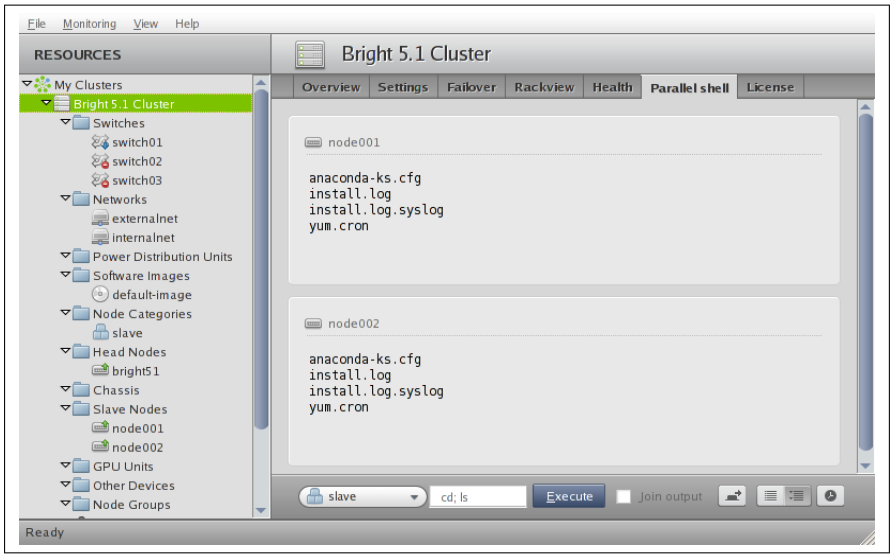

Figure 11.1: Executing Parallel Shell Commands

11.2 Disallowing User Logins To Nodes

Users run their computations on the cluster by submitting jobs to the workload management system. However workload management is only effective if cluster users do not run jobs outside the workload management system.

This can be enforced as a policy by disabling user logins to the nodes from a head node by adding a line like the following to the `/etc/ssh/sshd_config` file in the node image:

Example

```
AllowUsers root@bright51.cm.cluster *@node*.cm.cluster
```

In the example the domain `cm.cluster` or the `node` prefix may need to be changed according to the actual settings of the cluster.

After modifying the image, if there are provisioning nodes, the `updateprovisioners` command (section 6.1.4) should be run. The nodes can then simply be rebooted to pick up the new image, or alternatively, to avoid rebooting, the imageupdate command (section 6.5.2) can be run to pick up the new image. To make the new setting take effect, the following command is run using a parallel shell:

```
/etc/init.d/sshd restart
```

After this change is implemented, only the `root` user can log in to a node from the head node. Users may however still log in from any node to any other node as this is needed for a number of MPI implementations to function properly. Administrators may choose to disable interactive jobs in the workload management system as a measure to prevent users from starting jobs on other nodes. The workload management system documentation has more on configuring this.

11.3 Getting Help With Bugs And Other 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 bug or other issue with it that requires help from Bright Computing or Bright Computing's resellers. This section describes how to report such problems and get help for them.

11.3.1 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.

11.3.2 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. In the bug report it is helpful to include as many details as possible to ensure the development team is able to reproduce the bug. The policy at Bright Computing is to welcome such reports, to provide feedback to the reporter, and to work towards resolving bugs.

Bright Computing provides the cm-diagnose and the request-remote-assistance utilities to help resolve problems.

Reporting Cluster Manager Diagnostics With cm-diagnose

A diagnostic utility to help resolve bugs is cm-diagnose. To view its options, it can be run as "cm-diagnose --help". If it is 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 this:

```
[root@bright51 ~]# cm-diagnose
Collecting kernel version
Collecting top 5 processes
Collecting cmsh commands
Collecting network setup
Collecting uptime
Collecting CMDaemon configuration
Collecting node-installer configuration
Collecting CMDaemon database backups
Collecting RPM database information for image: default-image
Collecting base installation release
Collecting CM version
Collecting core trace information
Collecting process information
Collecting RPM database information
Collecting CMDaemon log
Collecting node-installer log
Collecting system log files log
Collecting mysql log file
Collecting mce log file
Collecting workload management log files
Collecting filesystem mount information
Collecting process information
Collecting license information
Preparing diagnostics file
Diagnostics saved in: /root/cm/cm-diagnose_bright51.cm.cluster-000007_27-4-11_12152.tar.gz


Submit diagnostics to http://support.brightcomputing.com/cm-diagnose/ ? [Y/n]

Uploaded file: cm-diagnose_bright51.cm.cluster-000007_27-4-11_12152.tar.gz

[root@bright51 ~]#
```

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.

The tool is run by the cluster administrator:

Example

```
[root@bright51 ~]# 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 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: bright51.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 tunnel into the cluster. When the engineer has ended the session, the administrator may remove the tunnel with a `ctrl-c`, and the display then shows:

```
Tunnel to sandbox.brightcomputing.com terminated.
Removed temporary Bright public key.
[root@bright51 ~]#
```

11.4 Backups

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 node 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`.

11.5 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.

11.5.1 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 11.5.3.

11.5.2 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 (see previous section).

The next section describes how to boot the DOS image.

11.5.3 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.

11.6 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@bright51 ~]# cmsh -c "category add newnodes; commit"
   [root@bright51 ~]# 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 (see Appendix H.2.1) as follows:

   ```
   [root@bright51 ~]# /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@bright51 ~]# cmsh
   [bright51]% 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 Bright Cluster Manager are described briefly. For all packages, references to the complete documentation are provided.

12.1 Modules Environment

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. Details on the modules environment from a user's perspective can be found 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)` manpage has more detail on this.

Modules can be combined in *meta-modules*. By default, the `default-environment` meta-module exists, which allows a user to load a number of other modules at once. 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.

12.2 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. The `interfaces` file is generated by the cluster management daemon.

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.

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` but (with the exception of GCC) require an installed license file to be used.

12.3.1 GCC

Package names: `gcc-recent`

12.3.2 Intel Fortran and C++ Compilers

Package names: `intel-fc` and `intel-cc`

The Intel compiler packages include the Intel Fortran and Intel C++ compilers. For both compilers two versions are installed: the 32-bit version, and the 64-bit (i.e. EM64T) version. Both versions can be invoked through the same set of commands, so the modules environment (see section 12.1) must be used to select one of the two versions. For the C++ compiler the 32-bit and 64-bit modules are called `intel/cc` and `intel/cce`

respectively. The modules for the Fortran compiler are called `intel/fc` and `intel/fce`. The Intel compilers also include a debugger which can be used by loading the `intel/idb` or `intel/idbe` module. The following 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

Full documentation for the Intel compilers is availble 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 licences, 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 can not 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

The Intel Cluster Checker is a tool that verifies if a cluster complies with all of the requirements of the Intel Cluster Ready Specification. This section lists the steps that must be taken to certify a cluster as Intel Cluster Ready.

12.4.1 Preparing Cluster

The Intel Cluster Ready specification requires a number of packages to be installed on the head and regular nodes:

The `cm-config-intelcompliance-master` and `cm-config-intelcompliance-slave` packages are installed on the head node and software images respectively. The `intel-cluster-runtime` package is installed on both the head node and the software images.

These packages guarantee through package dependencies that all Intel Cluster Ready package requirements are satisfied. Both packages are normally installed by default on a standard Bright Cluster Manager cluster. If they are not installed then the following commands install the complete suite:

Example

```
yum install cm-config-intelcompliance-master intel-cluster-runtime
yum --installroot=/cm/images/default-image install cm-config-intelcompl\
iance-slave intel-cluster-runtime
```

If `yum` reports that any additional packages need to be installed, simply agreeing to install them is enough to satisfy the requirements.

The Intel Cluster Ready specification also requires the `/etc/dat.conf` file. The file can be copied from the `/etc/ofed` directory, and has to be changed. This has to be done for both the head and the software images. The lines in the file that mention devices that are not used can be removed:

Example

For the head node:

```
cp /etc/ofed/dat.conf /etc/
```

For the default-image:

```
cp /cm/images/default-image/etc/ofed/dat.conf /cm/images/default-image/\
etc/
```

The `ibstat` command can be used to check if an InfiniBand device is used, and if so, what kind.

If the `mlx4_0` is used, the following lines are needed in the `dat.conf` file:

```
ofa-v2-mlx4_0-1u u2.0 nonthreadsafe default libdaploucm.so.2 dapl.2.0 "mlx4_0 1" ""
ofa-v2-mlx4_0-2u u2.0 nonthreadsafe default libdaploucm.so.2 dapl.2.0 "mlx4_0 2" ""
```

If no InfiniBand is present, the `ifconfig` command can be used to check which network interface is used. If the `eth0` device is used, then the following line is needed in `dat.conf` file:

```
ofa-v2-iwarp u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth0 0" ""
```

After installing the necessary packages and modifying the `dat.conf` file of the software images, 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 Input Files

Three runs of the Intel Cluster Checker are necessary for Intel Cluster Ready certification:

- As a privileged cluster user (i.e. `root`) to generate files for the package test
- As a regular cluster user
- As a privileged cluster user (i.e. `root`)

Each user requires an input file, which is called a *recipe*. Two other input files are important for certification: the node list and the file exclude list. These files are located in the `/home/cmsupport/intel-cluster-ready` directory:

- `recipe-user-ib.xml`
- `recipe-user-nonib.xml`
- `recipe-root-ib.xml`
- `recipe-root-nonib.xml`
- `nodelist`
- `excludefiles`

Recipes

The `recipe-user-ib.xml`, `recipe-user-nonib.xml`, `recipe-root-ib.xml` and `recipe-root-nonib.xml` files are default recipes that have been included as part of the `cm-config-intelcompliance-master` package. Both recipes may need small modifications based on the cluster for which certification is required.

For the user certification run, two recipes are available:

- `recipe-user-ib.xml`
- `recipe-user-nonib.xml`

For the privileged user certification run, two recipes are available:

- `recipe-root-ib.xml`
- `recipe-root-nonib.xml`

When an InfiniBand interconnect is used in the cluster, the `recipe-user-ib.xml` and `recipe-root-ib.xml` recipe should be used. For clusters without InfiniBand interconnect, the `recipe-user-nonib.xml` and `recipe-root-nonib.xml` should be used.

Throughout the recipe files, 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 values which are certainly too high (or too low in case of latency). After running the cluster checker, the performance thresholds can be adjusted to more realistic numbers based on the results that were obtained in practice. The cluster checker can be run with the `--auto` option, for automatic configuration, but this can give problems for clusters without InfiniBand interconnect.

A description of all test modules and parameters, is available in the Intel Cluster Checker documentation at: `http://software.intel.com/en-us/cluster-ready/`

Node List

The `nodelist` file lists the nodes which should be considered by the Intel Cluster Checker. By default only the head node and the first three nodes are included. However, for a full certification of the cluster, all nodes should be included.

Example

Rather than manually creating the `nodelist` file, the following command may be used to generate a nodes list consisting of `master` and `node001` through `node150`:

```
[root@mycluster ~]# (echo "   master  # type:head"; for i in {1..150};
> do echo "   node‘printf "%03d" $i‘ # type:compute";
> done) > /home/cmsupport/intel-cluster-ready/nodelist
```

File Exclude List

The `excludefiles` file lists all files which should be skipped when scanning for differences between nodes. Modifying the `excludefiles` list is normally not necessary.

12.4.3 Generating Fingerprint Files

Before running the Intel Cluster Checker, a list of all packages installed on the head node and regular nodes must be generated. These lists are used to ensure that the same software is available on all nodes. Generating the lists can be done by passing the `--packages` flag to the cluster checker. The cluster checker uses the first node defined in the `nodes` file as the standard.

Two output files with a time stamp in the filename are produced by the cluster checker. The two output files must subsequently be copied to the following location:

```
/home/cmsupport/intel-cluster-ready/head.package.list
/home/cmsupport/intel-cluster-ready/node.package.list
```

Example

```
module load shared intel-cluster-checker intel-cluster-runtime
cluster-check --packages  \
             /home/cmsupport/intel-cluster-ready/recipe-root.xml
mv master-20090522.173125.list \
             /home/cmsupport/intel-cluster-ready/head.package.list
mv node001-20090522.173125.list \
             /home/cmsupport/intel-cluster-ready/node.package.list
```

12.4.4 Running Intel Cluster Checker

Regular User Run

The `cmsupport` account is used to perform the regular user run. The following commands start the cluster checker:

```
su - cmsupport
module initadd intel-cluster-checker intel-cluster-runtime
module load intel-cluster-checker intel-cluster-runtime
cluster-check --certification 1.1 \
             /home/cmsupport/intel-cluster-ready/recipe-user-ib.xml
```

The cluster checker produces two output files (one `.xml` and one `.out`) which include time stamps in the filenames. In the event of failing tests, the output files should be consulted for details as to why the test failed.

When debugging and re-running tests, the `--include_only` *test* parameter can be passed to `cluster-check` to execute just the specified test (and the tests on which it depends).

Privileged User Run

The privileged user run should be started as the `root` user. The following commands start the cluster checker:

```
module load shared intel-cluster-checker intel-cluster-runtime
cluster-check --certification 1.1 \
            /home/cmsupport/intel-cluster-ready/recipe-root-ib.xml
```

In a heterogeneous cluster the privileged user 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.

12.4.5 Applying for Certificate

When both the regular user run as well as the privileged user run have reported that the *Check has Succeeded*, a certificate may be requested for the cluster. Requesting a certificate involves creating a `Bill of Materials` and submitting it along with the two output files of the two cluster checker runs to `cluster@intel.com`. The Intel Cluster Ready site contains interactive submissions forms that make the application process as easy as possible.

12.5 CUDA

In order to take advantage of the computational capabilities of NVIDIA GPUs that may be present in the nodes of a cluster, the optional CUDA packages should be installed.

12.5.1 Installing CUDA

A number of CUDA 3.2 packages exist in the YUM repository:

Package	Type	Description
`cuda32-toolkit`	shared	CUDA 3.2 math libraries and utilities
`cuda32-sdk`	shared	CUDA 3.2 software development kit
`cuda32-profiler`	shared	CUDA 3.2 profiler
`cuda32-driver`	local	CUDA 3.2 driver
`cuda32-libs`	local	CUDA 3.2 libraries

The packages marked as "shared" in the table above should be installed on the head nodes of a cluster containing CUDA-compatible GPUs. The packages marked as "local" should be installed to all nodes that have direct access to the GPUs. In most cases this means that the `cuda32-driver` and `cuda32-libs` packages should be installed in a software image. If the head nodes also contain GPUs, the `cuda32-driver` and `cuda32-libs`

packages should also be installed on the head nodes. Note that as a result of package dependencies, the `cuda32-libs` package are also installed on the head node. The reason for this is because the files are needed for compilation. Installing the `cuda32-driver` and `cuda32-libs` packages to a software image also causes several X11-related packages to be installed.

Example

On a cluster where (some of) the nodes contain GPUs but the head node does not contain a GPU, the following commands are issued on the head node to install the packages through YUM:

```
yum install cuda32-toolkit cuda32-sdk cuda32-profiler
yum --installroot=/cm/images/default-image install cuda32-driver cuda32\
-libs
```

The `cuda32-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 on the fly, and subsequently loads the module into the running kernel.

This cuda32-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/cuda32-driver start
Compiling CUDA3.2 driver..installing..probe..              [  OK  ]
[root@mycluster ~]# dmesg
...
PCI: Setting latency timer of device 0000:07:00.0 to 64
PCI: Setting latency timer of device 0000:09:00.0 to 64
NVRM: loading NVIDIA UNIX x86_64 Kernel Module  260.19.21 Thu Nov  4 21\
:16:27 PDT 2010
```

12.5.2 Verifying CUDA

An extensive method to verify that CUDA is working is to run the `verify_cuda32.sh` script, located in the CUDA SDK directory.

This script first copies the CUDA SDK source to a local directory under `/tmp`. 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_cuda32.sh -h`" is run.

Example

```
[root@cuda-test ~]# module load cuda32/toolkit
[root@cuda-test ~]# cd $CUDA_SDK
[root@cuda-test 3.2]# ./verify_cuda32.sh
Copy cuda32 sdk files to "/tmp/cuda32" directory.

make clean

make (can take a while)
```

```
Run all tests? (y/N)? y

Executing: /tmp/cuda32/C/bin/linux/release/alignedTypes

[alignedTypes]
CUDA device [Tesla T10 Processor] has 30 Multi-Processors
SM scaling value = 1.00
> Memory Size = 49999872
Allocating memory...
...
...
All cuda32 just compiled test programs can be found in
the "/tmp/cuda32/C/bin/linux/release/" directory.
They can be executed from the "/tmp/cuda32/C" directory.

The "/tmp/cuda32" directory can take up a lot of diskspace.
Use "rm -rf /tmp/cuda32" to remove the data.
```

Another method to verify that CUDA is working, is to build and use the `deviceQuery` command on a node containing one or more GPUs. The `deviceQuery` command lists all CUDA-capable GPUs in a device, along with several of their properties.

Example

```
[root@cuda-test ~]# module load cuda32/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 T10 Processor"
  CUDA Driver Version:                           3.20
  CUDA Runtime Version:                          3.20
  CUDA Capability Major/Minor version number:    1.3
  Total amount of global memory:                 4294770688 bytes
  Multiprocessors x Cores/MP = Cores:            30 (MP) x 8 (Cores/MP) \
                                                 = 240 (Cores)
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       16384 bytes
  Total number of registers available per block: 16384
  Warp size:                                     32
  Maximum number of threads per block:           512
  Maximum sizes of each dimension of a block:    512 x 512 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             256 bytes
  Clock rate:                                    1.30 GHz
```

```
  Concurrent copy and execution:                 Yes
  Run time limit on kernels:                     No
  Integrated:                                    No
  Support host page-locked memory mapping:       Yes
  Compute mode:                                  Default
  Concurrent kernel execution:                   No
  Device has ECC support enabled:                No
  Device is using TCC driver mode:               No
...
...
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 3.20,
 CUDA Runtime Version = 3.20, NumDevs = 2,
 Device = Tesla T10 Processor,
 Device = Tesla T10 Processor

PASSED
```

The CUDA user manual has further information on how to run compute jobs using CUDA.

12.5.3 Verifying OpenCL

CUDA 3.2 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 cuda32/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
oclDeviceQuery.exe Starting...

OpenCL SW Info:

 CL_PLATFORM_NAME:      NVIDIA CUDA
 CL_PLATFORM_VERSION:   OpenCL 1.0 CUDA 3.2.1
 OpenCL SDK Revision:   7027912


OpenCL Device Info:

 2 devices found supporting OpenCL:

 --------------------------------
 Device Tesla T10 Processor
 --------------------------------
  CL_DEVICE_NAME:                        Tesla T10 Processor
  CL_DEVICE_VENDOR:                      NVIDIA Corporation
  CL_DRIVER_VERSION:                     260.19.21
  CL_DEVICE_VERSION:                     OpenCL 1.0 CUDA
  CL_DEVICE_TYPE:                        CL_DEVICE_TYPE_GPU
  CL_DEVICE_MAX_COMPUTE_UNITS:           30
```

```
  CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS:   3
  CL_DEVICE_MAX_WORK_ITEM_SIZES:        512 / 512 / 64
  CL_DEVICE_MAX_WORK_GROUP_SIZE:        512
  CL_DEVICE_MAX_CLOCK_FREQUENCY:        1296 MHz
  CL_DEVICE_ADDRESS_BITS:               32
  CL_DEVICE_MAX_MEM_ALLOC_SIZE:         1023 MByte
  CL_DEVICE_GLOBAL_MEM_SIZE:            4095 MByte
  CL_DEVICE_ERROR_CORRECTION_SUPPORT:   no
  CL_DEVICE_LOCAL_MEM_TYPE:             local
  CL_DEVICE_LOCAL_MEM_SIZE:             16 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
  CL_DEVICE_MAX_WRITE_IMAGE_ARGS:       8
  CL_DEVICE_SINGLE_FP_CONFIG:           INF-quietNaNs round-to-nearest

  CL_DEVICE_IMAGE <dim>                 2D_MAX_WIDTH     4096
                                        2D_MAX_HEIGHT    32768
                                        3D_MAX_WIDTH     2048
                                        3D_MAX_HEIGHT    2048
                                        3D_MAX_DEPTH     2048

  CL_DEVICE_EXTENSIONS:                 cl_khr_byte_addressable_store
                                        cl_khr_icd
                                        cl_khr_gl_sharing
                                        cl_nv_compiler_options
                                        cl_nv_device_attribute_query
                                        cl_nv_pragma_unroll
                                        cl_khr_global_int32_base_atomics
                                        cl_khr_global_int32_extended_atomics
                                        cl_khr_local_int32_base_atomics
                                        cl_khr_local_int32_extended_atomics
                                        cl_khr_fp64


  CL_DEVICE_COMPUTE_CAPABILITY_NV:      1.3
  NUMBER OF MULTIPROCESSORS:            30
  NUMBER OF CUDA CORES:                 240
  CL_DEVICE_REGISTERS_PER_BLOCK_NV:     16384
  CL_DEVICE_WARP_SIZE_NV:               32
  CL_DEVICE_GPU_OVERLAP_NV:             CL_TRUE
  CL_DEVICE_KERNEL_EXEC_TIMEOUT_NV:     CL_FALSE
  CL_DEVICE_INTEGRATED_MEMORY_NV:       CL_FALSE
  CL_DEVICE_PREFERRED_VECTOR_WIDTH_<t>  CHAR 1, SHORT 1, INT 1, LONG 1, FLOAT
...
...
oclDeviceQuery, Platform Name = NVIDIA CUDA,
Platform Version = OpenCL 1.0 CUDA 3.2.1,
SDK Revision = 7027912, NumDevs = 2,
Device = Tesla T10 Processor, Device = Tesla T10 Processor

System Info:
```

```
Local Time/Date =  19:04:20, 01/13/2011
CPU Name: Intel(R) Xeon(R) CPU 5130 @ 2.00GHz
# of CPU processors: 4
Linux version 2.6.18-194.26.1.el5 (mockbuild@builder10.centos.org)
(gcc version 4.1.2 20080704
(Red Hat 4.1.2-48)) #1 SMP Tue Nov 9 12:54:20 EST 2010

PASSED
```

12.6 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.lustre.org/index.php/Main_Page`.

12.6.1 Architecture

There are four components per 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.

12.6.2 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):

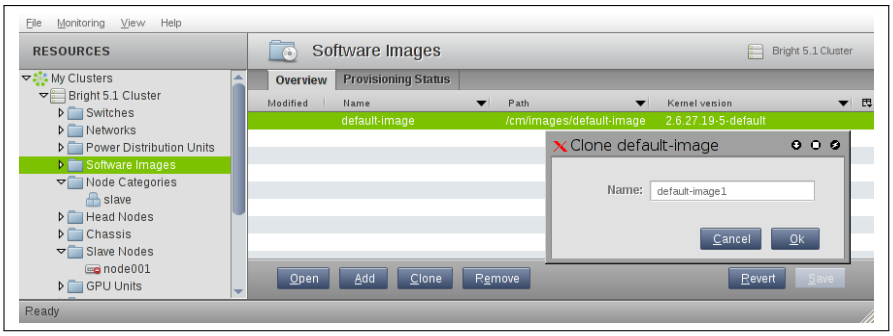

Figure 12.1: `cmgui`: Cloning An Image

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 Lustre website. It is best to first check which version of Lustre can be used for a particular distribution against the Lustre Test Matrix at the Lustre wiki at `http://wiki.lustre.org/index.php/Lustre_Release_Information#Lustre_Test_Matrix`.

After choosing a Lustre version from the Lustre Test Matrix, the appropriate distribution and platform can be chosen. For CentOS and Scientific Linux (SL), RedHat packages can be used. To download the packages an account is required.

The RPM packages to download are:

- kernel: Lustre-patched kernel (MDS/MGS/OSS only)
- kernel-modules: Lustre 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)

In most cases the e2fsprogs distribution package is already installed, so the package has to be upgraded. It is possible that the Lustre e2fsprogs package conflicts with the e4fsprogs distribution package, 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. Opening `/sys/block` and GRUB error messages can be ignored.

Example

```
[root@mycluster ~]# rpm --root /cm/images/lustre-server-image -e e4fspr\
ogs
[root@mycluster ~]# rpm --root /cm/images/lustre-server-image -Uvh e2fs\
progs-1.41.10.sun2-0redhat.rhel5.x86_64.rpm
[root@mycluster ~]# rpm --root /cm/images/lustre-server-image --force -\
ivh kernel-2.6.18-164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
[root@mycluster ~]# rpm --root /cm/images/lustre-server-image -ivh lust\
re-ldiskfs-3.2.0-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
[root@mycluster ~]# rpm --root /cm/images/lustre-server-image -ivh lust\
re-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm lustre-modules\
-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
```

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 ~]# ls -1 vmlinuz-*
vmlinuz-2.6.18-164.11.1.el5_lustre.2.0.0.1
vmlinuz-2.6.18-194.17.1.el5
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% use lustre-server-image
[mycluster->softwareimage[lustre-server-image]]% set kernelversion 2.6.\
18-164.11.1.el5_lustre.2.0.0.1
[mycluster->softwareimage*[lustre-server-image*]]% commit
```

Creating The Lustre Server Category

A node category is cloned, for example, slave 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 slave lustre-server
[mycluster->category*[lustre-server*]]% commit
[mycluster->category[lustre-server]]% set softwareimage lustre-server-i\
mage
[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 slavenode 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 slavenode 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.

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 a 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, to format `/dev/sdb`, and set 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 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 a external storage device and/or a redundant storage device.

To format a object storage target `mkfs.lustre` is used. For example, to format `/dev/sdb`, set the management node to "10.141.16.1", the filesystem name to "lustre00", and 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.

12.6.3 Client Implementation

There are several ways to install a Lustre client.

If the client has a supported kernel version, the lustre-client RPM and lustre-client-modules RPM 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 Lustre website:

- lustre-client: Lustre client userland tools (client for unpatched vendor kernel)
- lustre-client-modules: Lustre client 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_164.11.1.el5 in the following example:

Example

```
[root@mycluster ~]# ls lustre-client-modules-*
lustre-client-modules-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64\
.rpm
[root@mycluster ~]# ls /cm/images/lustre-client-image/boot/vmlinuz-*
/cm/images/lustre-client-image/boot/vmlinuz-2.6.18_164.11.1.el5
```

The installation can then be carried out:

Example

```
[root@mycluster ~]# rpm --root /cm/images/lustre-client-image -ivh lust\
re-client-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm lustre-\
client-modules-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.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 a starting point image for the 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 RPM packages can be downloaded from the Lustre website. The same Lustre version which is used for the Lustre servers must be used for the Lustre clients.

The RPM packages to download are:

- kernel: Lustre-patched kernel
- kernel-modules: Lustre modules (client and server for Lustre-patched kernel)
- lustre: Lustre userland tools (client and server for Lustre-patched kernel)

If the Lustre kernel has a lower version number than the installed kernel, then the Lustre kernel needs to be installed with the `--force` option. Opening `/sys/block` and GRUB error messages can be ignored:

Example

```
[root@mycluster ~]# rpm --root /cm/images/lustre-client-image --force -\
ivh kernel-2.6.18-164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
[root@mycluster ~]# rpm --root /cm/images/lustre-client-image -ivh lust\
re-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm lustre-modules\
-2.0.0.1-2.6.18_164.11.1.el5_lustre.2.0.0.1.x86_64.rpm
```

Any ldiskfs warnings can be ignored since ldiskfs is not used by a Lustre client.

The kernel version used is set for the Lustre image to the Lustre kernel:

Example

```
[root@mycluster ~]# cd /cm/images/lustre-client-image/boot
[root@mycluster ~]# ls -1 vmlinuz-*
vmlinuz-2.6.18-164.11.1.el5_lustre.2.0.0.1
vmlinuz-2.6.18-194.17.1.el5
[root@mycluster ~]# cmsh
[mycluster]% softwareimage
[mycluster->softwareimage]% use lustre-client-image
[mycluster->softwareimage[lustre-client-image]]% set kernelversion 2.6.\
18-164.11.1.el5_lustre.2.0.0.1
[mycluster->softwareimage*[lustre-client-image*]]% commit
```

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 Lustre 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 to download is chosen:

- lustre-<version>.tar.gz: Lustre source code

The source file is copied to the image:

Example

```
[root@mycluster ~]# cp lustre-2.0.0.1.tar.gz /cm/images/lustre-client-i\
mage/usr/src
```

If the kernel-devel package is not installed on the client image, it is first installed so that the kernel can be compiled:

```
[root@mycluster ~]# rpm --root /cm/images/lustre-client-image -q kernel\
-devel
[root@mycluster ~]# yum install --installroot=/cm/images/lustre-client-\
image kernel-devel
```

The Lustre software is then built and installed:

Example

```
[root@mycluster ~]# chroot /cm/images/lustre-client-image
[root@mycluster /]# cd /usr/src
[root@mycluster src]# ln -s kernels/‘uname -r‘-x86_64 linux
[root@mycluster src]# tar zxvf lustre-2.0.0.1.tar.gz
[root@mycluster src]# cd lustre-2.0.0.1
[root@mycluster lustre-2.0.0.1]# ./configure --disable-server
[root@mycluster lustre-2.0.0.1]# make
[root@mycluster lustre-2.0.0.1]# make install
[root@mycluster lustre-2.0.0.1]# depmod -a
[root@mycluster lustre-2.0.0.1]# cd /usr/src
[root@mycluster src]# rm -rf lustre-2.0.0.1
[root@mycluster src]# rm linux
[root@mycluster src]# exit
```

To configure the lnet kernel module to use TCP/IP, the string `options lnet networks=tcp` is added to the `/etc/modprobe.conf` file of the client image.

```
[root@mycluster ~]# echo "options lnet networks=tcp" >> /cm/images/lust\
re-client-image/etc/modprobe.conf
```

Creating The Lustre Client Category

A node category is cloned, for example slave to lustre-client. The software image is set to the Lustre client image:

Example

```
[root@mycluster ~]# cmsh
[mycluster]% category
[mycluster->category]% clone slave 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 slavenode 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`. The Lustre file striping can be set with `lfs setstripe`.

Example

```
[root@lclient001 ~]# lfs getstripe /mnt/lustre00
[root@lclient001 ~]# lfs setstripe -s 4M -o -1 -c -1 /mnt/lustre00
```

The `lfs setstripe` command in the example sets the filesystem to use 4MB blocks, the start OST is chosen by the MDS and stripes data over all available OSTs.

13

High Availability

In a cluster with a single head node, the head node is a single point of failure for the entire cluster. In certain environments it is unacceptable that a single machine failure can cause a disruption to the daily operations of a cluster. Bright Cluster Manager includes high availability (HA) features which allow clusters to be set up with two head nodes.

13.1 HA Concepts

In a cluster with an HA setup, one of the head nodes is called the *primary* head node and the other head node is called the *secondary* head node. Under normal operation, one of the two head nodes is in *active* mode, whereas the other is in *passive* mode.

It is important to distinguish between the concepts of primary/secondary and active/passive mode. The difference between the two concepts is that while a head node which is *primary* always remains *primary*, the mode that the node is in may change. It is possible for the primary head node to 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.

The central concept of HA is that 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 to allow jobs running on these nodes to keep running.

13.1.1 Services

There are several services being offered by a head node to the cluster and its users. One of the key aspects 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 for the capacity of both machines to be used for certain tasks (e.g. provisioning slave nodes), 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 services key 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
- **MySQL**: running in multi-master replication mode
- **NTP**
- **DNS**

When an HA setup is created, the above services are automatically reconfigured for an HA environment with two head nodes.

In addition, both head nodes will also receive the *Provisioning role*, which means that slave nodes can be provisioned from both head nodes. The implications of running a cluster with multiple provisioning nodes are described in section 6.1. Most importantly, every time a change has been applied to a software image, the `updateprovisioners` command in the `cmsh softwareimage` mode has to be executed to propagate changes to the other provisioning nodes. Alternatively in CMGUI the `Update Provisioning Nodes` button in the Provisioning Nodes tab may be pressed when the Software Images folder is selected in the resource tree.

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 will be migrated:

- **NFS**
- **Workload Management (e.g. SGE, Torque/Maui)**

13.1.2 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 involves two virtual IP interfaces which migrate in the event of a failover: 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.

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 will always reach whichever head node is active. Similarly, inside the cluster slave nodes will use the shared internal IP address wherever possible for referring to the head node. For example, slave nodes mount NFS filesystems on the shared internal IP interface so that the imported filesystems will continue to be accessible in the event of a failover.

Shared interfaces are normally implemented as alias interfaces on the physical interfaces (e.g. `eth0:0`).

13.1.3 Dedicated Failover Network

In addition to the internal and external network interfaces on both head nodes, the two head nodes are usually also connected using a direct dedicated network connection. This connection is used between the two

head nodes to monitor their counterpart's availability. It is highly recommended to run a UTP cable directly from the NIC of one head node to the NIC of the other. Not using a switch means there is no disruption of the connection in the event of a switch reset.

13.1.4 Shared Storage

Almost any HA setup also involves some form of shared storage between two head nodes. The reason for this is that state must be preserved after a failover sequence. It would be unacceptable for user home directories to be unavailable to the cluster in the event of a failover.

In the most common HA setup, the following three directories are shared:

- User home directories (i.e. `/home`)
- Shared tree containing applications and libraries that are made available to the slave nodes (i.e. `/cm/shared`)
- Node certificate store (i.e. `/cm/node-installer/certificates`)

The shared filesystems are only available on the active head node. For this reason, it is generally not recommended for end-users to login to the secondary head node.

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 be accessed simultaneously on both head nodes, the filesystem that is used on the block device is typically not suited for simultaneous access. In fact, simultaneous access to a filesystem from two head nodes must be avoided at all cost because it will almost certainly lead 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 are mirroring 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

simulated DAS block device. DRBD is a cost-effective solution for implementing shared storage in an HA setup.

Custom Shared Storage

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 needs to 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.

13.1.5 Handling a Split Brain

Because of the risks involved in accessing a shared filesystem simultaneously from two head nodes, it is of the highest importance only one head node is in active mode at any point in 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.

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. This is generally referred to as a *split brain* situation.

Since detecting a split brain situation is impossible, the passive head node may not assume that the active node is no longer up if it finds the active node to be unresponsive. It is quite possible that the active head node is still up and running, and observes that the passive head node has disappeared (i.e. a split brain).

To resolve these situations, a passive head node that notices that its active counterpart is no longer responding will first go into *fencing mode*. While a node is fencing, it will try to obtain proof that its counterpart is indeed powered off.

There are two ways in which such proof can be obtained:

a By asking the administrator to manually confirm that the active head node is indeed powered off

b 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.

13.1.6 Quorum

There is only one problem: in situations where the passive head node loses its connectivity to the active head node, but the active head node is doing fine communicating with the entire cluster, there is no reason to initiate a failover. In fact, this could even result in undesirable situations where the cluster is rendered unusable because a passive head node might decide to power down an active head node just because the passive head node is unable to communicate with the outside world (except the PDU feeding the active head node).

To prevent a passive head node from powering off an active head node unnecessarily, the passive head node will first initiate a quorum by contacting all nodes in the cluster. The nodes will be asked to confirm that they also cannot communicate with the active head node. If more than half of the total number of slave nodes confirm that they are also unable to communicate with the active head node, the passive head node will initiate the STONITH procedure and move 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 will be asked to certify 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 has been 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.

Example

```
[root@bright51 ~]# cmsh
[bright51]% partition failover base
[bright51->partition[base]->failover]% set disableautomaticfailover yes
[bright51->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

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 Set Up Procedure

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 will describe how to

create an HA setup using the `cmha-setup` utility which was specifically created for guiding the process of building an HA setup. During the process of setting up HA, the `cmha-setup` utility will interact with the cluster management environment (using `cmsh`) to create the setup. Although it is also possible to create an HA setup manually using either CMGUI or `cmsh`, this approach is not recommended as it is error-prone.

Globally the process of creating an HA setup involves three stages:

Preparation: setting up configuration parameters for the shared interface and for the secondary head node that is about to be installed.

Cloning: installing the secondary head node is done by creating a clone of the primary head node.

Shared Storage Setup: setting up the method for shared storage

13.2.1 Preparation

The following steps will prepare for the cloning of a new head node.

0 Power off all slave nodes.

1 To start the HA setup, run the `cmha-setup` command from a `root` shell on the primary head node and choose `Setup Failover` in the main menu.

2 Enter the MySQL root password. On a new installation this is the administrator password that was configured during cluster installation.

3 Configure parameters for the virtual shared **internal** IP address. By selecting `Create` the shared interface will be created.

4 Configure parameters for the virtual shared **external** IP address. By selecting `Create` the shared external interface will be created.

5 Configure the hostname and internal and external primary network interfaces for the secondary head node.

6 The primary head node may have other network interfaces (e.g. InfiniBand interfaces, IPMI interface, alias interface on the IPMI network). These interfaces will also be created on the secondary head node, but the IP address of the interfaces will need to be configured. For each such interface, when prompted configure a unique IP address for the secondary head node.

7 Configure the dedicated failover network that will be used between the two head nodes for heartbeat monitoring.

8 Assign a network interface and IP address on both head nodes that will be used for the dedicated failover network.

13.2.2 Cloning

After the parameters have been configured in the **Preparation** stage, the secondary head node should be cloned from the primary head node. This procedure may also be repeated later on if a head node ever needs to be replaced (e.g. as a result of defective hardware).

1 Boot the secondary head node off the internal cluster network. It is highly recommended that the primary and secondary head nodes have identical hardware configurations.

2 In the `Cluster Manager PXE Environment` menu, before the timeout of 5s expires, select "Start Rescue Environment" to boot the node into a Linux ramdisk environment.

3 Once the rescue environment has finished booting, login as root. No password is required

4 Execute the following command:
`/cm/cm-clone-install --failover`

5 When prompted to enter a network interface to use, enter the interface that was used to boot from the internal cluster network (e.g. `eth0`, `eth1`, ...). When unsure about the interface , switch to another console and use `ethtool -p <interface>` to make the NIC corresponding to an interface blink.

6 If the provided network interface is correct, a `root@master's password` prompt will appear. Enter the root password.

7 After the cloning process has finished, press `Y` to reboot and let the machine boot off its harddrive.

8 Once the secondary head node has finished booting from its harddrive, go back to the primary head node and select `Finalize`.

9 Enter the MySQL root password.

10 Verify that the `mysql`, `ping` and `status` are listed as `OK` for both head nodes. This confirms that the HA setup was completed successfully. The `backupping` will initially report `FAILED`, but will start working as soon as the secondary head node has been rebooted. Press `OK` and then `Reboot` to reboot the secondary head node.

11 Wait until the secondary head node has fully booted, and select "Failover Status" from the main menu. After that, select "View failover status" and confirm that `backupping` is also reported as `OK`

13.2.3 Shared Storage Setup

The last stage of creating an HA setup involves setting up a shared storage solution.

NAS

1 In the `cmha-setup` main menu, select the `Setup Shared Storage` option.

2 Select `NAS`.

3 Select the parts of the filesystem that should be copied to NAS filesystems.

4 Configure the NFS server and the paths to the NFS volume for each of the chosen mountpoints.

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 will begin.

DAS

1 In the `cmha-setup` main menu, select the `Setup Shared Storage` option.

2 Select `DAS`.

5 Select the parts of the filesystem that should be placed on shared DAS filesystems.

6 Enter the hostnames of the primary and secondary head nodes and the physical disk partitions to use on both head nodes.

7 Confirm that the contents of the listed partitions can be erased on both head nodes. After filesystems have been created, the current contents of the shared directories will be copied onto the shared filesystems and the shared filesystems will be mounted over the old non-shared filesystems.

DRBD

1 In the `cmha-setup` main menu, select the `Setup Shared Storage` option.

2 Select `DRBD`.

3 Select `Install DRBD` to install the `drbd` RPMs if they have not been installed yet.

4 Select `DRBD Setup`.

5 Select the parts of the filesystem that should be placed on DRBD filesystems.

6 Enter the hostnames of the primary and secondary head nodes and the physical disk partitions to use on both head nodes.

7 Confirm that the contents of the listed partitions can be erased on both head nodes. After DRBD based filesystems have been created, the current contents of the shared directories will be copied onto the DRBD based filesystems and the DRBD based filesystems will be mounted over the old non-shared filesystems.

8 Once the setup process has completed, select `DRBD Status/Overview` to verify the status of the DRBD block devices.

13.2.4 Automated Failover

If automatic failover is desired, the two head nodes must be able to power off their counterpart. This is done by setting up power control (see chapter 5 for details).

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 esspecially true when no dedicated IPMI network port is used. In this case, the `device power status` will report a failure for the active head node. This does not necessarily mean that the head nodes can not 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

On `mycluster1` verify that the IPMI interface of `mycluster2` is reachable:

```
[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
```

On `mycluster2` verify that the IPMI interface of `mycluster1` is reachable:

```
[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 can be used to crash 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 will power off the machine that has frozen and will then proceed to switch to active mode.

13.3 Managing HA

Once an HA setup has been created, there are several things to be aware of while managing the cluster.

13.3.1 `cmha` utility

The main utility for interacting with the HA subsystem is `cmha`. Using `cmha` an administrator may query the state the HA subsystem is in on the local machine. An administrator may also manually initiate a failover sequence to make the current machine active.

Example

Usage information:

```
[root@mycluster1 ~]# cmha
Usage: /cm/local/apps/cmd/sbin/cmha status | makeactive | dbreclone <node>
```

Example

To display failover status informtation:

```
[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 both head nodes:

HA Status	Description
`backupping`	the other head node is visible over the dedicated failover network
`mysql`	MySQL replication status
`ping`	the other head node is visible over the primary management network
`status`	CMDaemon running on the other head node responds to SOAP calls

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 ...
```

13.3.2 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

Esspecially 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 can be helpful to examine the output in `/var/log/cmdaemon`. The `cmha makeactive` command can be used to instruct a head node to become active again.

13.3.3 Keeping Head Nodes in Sync

It is important that relevant filesystem changes outside of the shared directories that are made to active head node, are also made on the passive head node. For example:

- RPM installations/updates
- Applications installed locally
- Configuration file changes outside of the filesystems that are shared

It is also useful to realize that when the shared storage setup was made, the contents of the shared directories (at that time) were copied from the local filesystem to the newly created shared filesystems. The shared filesystems were then mounted over the mountpoints, effectively hiding the local contents.

Since the shared filesystems are only mounted on the active machine, it is normal that the old data is still visible when a head node is operating in passive mode. This is not harmful, but may surprise users logging in to the passive head node. For this reason, logging in to a passive head node is not recommended for end-users.

13.3.4 High Availability Parameters

There are several HA-related parameters that can be tuned. In the cluster management GUI this can be done through the `Failover` tab while selecting the cluster in the resource tree. In `cmsh` the settings can be accessed in the `failover` sub-mode of the `base` partition.

Example

```
[mycluster1]% partition failover base
[mycluster1->partition[base]->failover]% show
Parameter                Value
------------------------ ------------------------------------------------
Dead time                10
Failover network         failovernet
Init dead                30
Keep alive               1
Mount script             /cm/local/apps/cmd/scripts/drbd-mount.sh
Quorum time              60
Secondary master         mycluster2
Unmount script           /cm/local/apps/cmd/scripts/drbd-unmount.sh
Warn time                5
[mycluster1->partition[base]->failover]%
```

Keep alive

The passive head node will use the value specified as `Keep alive` as a frequency for checking that the active head node is still up. If a dedicated failover network is being used, there will be 3 separate heartbeat checks for determining that a head node is reachable.

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 the `Warn time`, a warning is logged that the active head node might become unreachable soon.

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`, the active head node is considered dead and a quorum is initialized. Depending on the outcome of the quorum, a failover sequence may be initiated.

Failover network

The `Failover network` setting determines that network that will be used as a dedicated network for the `backupping` heartbeat check. This is normally a direct cable from a NIC on one head node to a NIC on the other head node.

Init dead

When boot 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 the node boots (or rather when the cluster management daemon is starting, the `Init dead` time is used rather than the `Dead time` to determine whether the other node is alive.

Mount script

The script pointed to by the `Mount script` setting is responsible for bringing up and mounting the shared filesystems.

Unmount script

The script pointed to by the `Unmount script` setting is responsible for bringing down and unmounting the shared filesystems.

Quorum time

When a node is being asked what head nodes it is able to reach over the network, the node has a certain time within which it must respond. If a node does not respond to a quorum within the configured `Quorum time` it is no longer considered for the results of the quorum.

Secondary master

The `Secondary master` setting is used to define the secondary head node to the cluster.

13.3.5 Re-cloning a Head Node

After an HA setup has gone into production, it may become necessary to re-install one of the head nodes at some point. This would be necessary if one of the head nodes was replaced due to hardware failure.

To re-clone a head node out of an existing active head node, enter the `cmha-setup`, select `Failover Status` and subsequently `View clone installation instructions`. Then follow the instructions as displayed on the screen (i.e. repeat the instructions in section 13.2.2).

Note that if the MAC address of one of the head nodes has changed, it is typically necessary to request a new license. See section 4.1.3 for more information on obtaining a new license.

A

Generated Files

Section 3.7.3 describes how system configuration files on all nodes are written out. This appendix contains a list of all system configuration files which are generated automatically. All of these files may be listed as `Frozen Files` in the Cluster Management Daemon configuration file to prevent them from being generated automatically (see section 3.7.3 and Appendix C).

Files generated automatically on head nodes

File	Generated By	Method	Comment
/etc/resolv.conf	CMDaemon	Entire file	
/etc/localtime	CMDaemon	Entire file	
/etc/exports	CMDaemon	Section	
/etc/fstab	CMDaemon	Section	
/etc/hosts	CMDaemon	Section	
/etc/hosts.allow	CMDaemon	Section	
/tftpboot/mtu.conf	CMDaemon	Entire file	
/etc/sysconfig/ipmicfg	CMDaemon	Entire file	
/etc/sysconfig/network/config	CMDaemon	Section	SuSE only
/etc/sysconfig/network/routes	CMDaemon	Entire file	SuSE only
/etc/sysconfig/network/ifcfg-*	CMDaemon	Entire file	SuSE only
/etc/sysconfig/network/dhcp	CMDaemon	Section	SuSE only
/etc/sysconfig/network	CMDaemon	Entire file	RedHat only
/etc/sysconfig/network-scripts/ifcfg-*	CMDaemon	Entire file	RedHat only
/etc/dhclient.conf	CMDaemon	Entire file	RedHat only
/etc/dhcpd.conf	Entire file		
/etc/dhcpd.slavenet.conf	Entire file		
/etc/shorewall/interfaces	CMDaemon	Section	
/etc/shorewall/masq	CMDaemon	Section	
/etc/sysconfig/clock	CMDaemon	Entire file	
/etc/postfix/main.cf	CMDaemon	Section	
/etc/postfix/generic	CMDaemon	Section	
/etc/aliases	CMDaemon	Section	
/etc/ntp.conf	CMDaemon	Entire file	
/etc/ntp/step-tickers	CMDaemon	Entire file	RedHat only
/etc/named.conf	CMDaemon	Entire file	
/var/named/*	CMDaemon	Entire file	RedHat only
/var/lib/named/*	CMDaemon	Entire file	SuSE only

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	Entire file	
/etc/sysconfig/kernel	CMDaemon	Section	SuSE only
/etc/sysconfig/network/config	CMDaemon	Section	SuSE only
/etc/sysconfig/network/routes	CMDaemon	Entire file	SuSE only
/boot/vmlinuz	CMDaemon	Symlink	
/boot/initrd	CMDaemon	Symlink	
/boot/initrd-*	CMDaemon	Entire file	
/etc/modprobe.conf	CMDaemon	Entire file	
/etc/postfix/main.cf	CMDaemon	Section	
/etc/postfix/generic	CMDaemon	Section	
/etc/aliases	CMDaemon	Section	

Files generated automatically on nodes

File	Generated By	Method	Comment
/etc/hosts	Node installer	Section	
/etc/exports	CMDaemon	Section	
/etc/fstab	Node installer	Section	
/etc/sysconfig/network	Node installer	Entire file	
/etc/sysconfig/network/ifcfg-*	Node installer	Entire file	SuSE only
/etc/sysconfig/network-scripts/ifcfg-*	Node installer	Entire file	RedHat only
/etc/ntp.conf	Node installer	Entire file	
/etc/ntp/step-tickers	Node installer	Entire file	
/etc/postfix/main.cf	Node installer	Section	
/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 will start in *head* mode. On a regular node, it will start 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 that will be 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 that will be used to connect to the MySQL database server.

DBName directive

Syntax: `DBName =` *database*
Default: `DBName = "cmdaemon"`

The `DBName` directive specifies the database that will be 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 that will be used on the MySQL database server to store monitoring related data.

DBUnixSocket directive

Syntax: `DBUnixSocket =` *filename*

The `DBUnixSocket` directive specifies the named pipe that will be 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 will be 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 which will be 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 that will be 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 that will be 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) that will be 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 that will be 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`

When set to true, DHCP's "deny unknown-clients" option will be set. This means no new DHCP leases are granted to unknown clients. In Bright 5.1 this flag is used for all networks. In 5.2 particular networks can be specified.

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` allows (when enabled) for PDU ports to be powered off as well to conserve power. See section 5.1.3 for more information.

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 will require 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 will not be displayed on the first boot menu.

StoreBIOSTimeInUTC directive

Syntax: `StoreBIOSTimeInUTC = true|false`
Default: `StoreBIOSTimeInUTC = false`

When StoreBIOSTimeInUTC is set to true, the BIOS time in nodes will be stored in UTC rather than local time.

FreezeChangesToSGEConfig directive

Syntax: FreezeChangesToSGEConfig = true|false
Default: FreezeChangesToSGEConfig = false

When FreezeChangesToSGEConfig is set to true, CMDaemon will not make any modifications to the SGE configuration.

FreezeChangesToPBSConfig directive

Syntax: FreezeChangesToPBSConfig = true|false
Default: FreezeChangesToPBSConfig = false

When FreezeChangesToPBSConfig is set to true, CMDaemon will not make any modifications to the PBS configuration.

FreezeChangesToTorqueConfig directive

Syntax: FreezeChangesToTorqueConfig = true|false
Default: FreezeChangesToTorqueConfig = false

When FreezeChangesToTorqueConfig is set to true, CMDaemon will not make any modifications to the Torque configuration.

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 value of *facility* must be `LOG_KERN`, `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_SYSLOG` or `LOG_LOCAL0..7`

D

Disk Partitioning

Bright Cluster Manager requires that disk partitionings are specified using the XML format that is described below. Partitioning is relevant when the disk-layout for nodes is being configured, but also when the head node is initially installed. For nodes, the XML format also allows diskless operation.

D.1 Structure of Partitioning Definition

The global structure of a file that describes a partitioning setup is 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, the next sections contain a few examples with an explanation of all elements.

```
<?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.
- 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: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='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>
```

D.2 Example: Default Node Partitioning

The following example shows the default layout used for regular nodes. This example assumes a single disk. Because multiple `blockdev` tags are used, the node-installer will first try to use `/dev/sda` and then `/dev/hda`. For each `partition`, a `size` is specified. Sizes can be specified using megabytes (`500M`), gigabytes (`50G`) or terabytes (`2T`). Alternatively, a `max` size will use all remaining space. For swap partitions a size of `auto` will result in twice the nodes memory size. In this case all file systems are specified as `ext3`, valid alternatives are `ext2` and `xfs`. For details on mount options, please refer to the `mount` man-page. Note that if the `mountOptions` tag is left empty, its value will default to `defaults`.

```
<?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>
    <blockdev>/dev/hda</blockdev>

    <partition id="a1">
      <size>5G</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>auto</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>
```

D.3 Example: Preventing Accidental Data Loss

The following example shows the use of the `vendor` and `requiredSize` tags. These are optional tags which can be used to prevent accidentally repartitioning the wrong drive. 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. Note that the node-installer reads a drives vendor string from `/sys/block/<drive name>/device/vendor`. Specifying device assertions is recommended for machines that contain important data as it will serve as a protection against situations where drives are assigned to incorrect block devices. This could 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`.

```
<?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>
```

D.4 Example: Using custom assertions

The following example shows the use of the `assert` tag, which can be added to a `device` definition. The `assert` tag is similar to the `vendor` and `size` tags described before. 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. The exit code should be non zero when the assert should trigger the node-installer to halt.

```
<?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>
```

D.5 Example: Software RAID

This example shows a simple software RAID setup. The `level` tag specifies what type of RAID is used. The following RAID-levels are supported: `0` (striping without parity), `1` (mirroring), `4` (striping with dedicated parity drive), `5` (striping with distributed parity) and `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. Note that when RAID is used, the administrator is responsible for ensuring that the correct kernel modules are loaded. Normally including one of the following modules should be sufficient: `raid0`, `raid1`, `raid4`, `raid5`, `raid6`.

```
<?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>
```

D.6 Example: Logical Volume Manager

This example shows a simple LVM setup. The `member` tags must refer to an `id` attribute of a `partition` tag, or an `id` attribute of a `raid` tag. Note that when LVM is used, the administrator is responsible for ensuring that the `dm-mod` kernel module is loaded.

```
<?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>
```

D.7 Example: Diskless

This example shows how nodes can be configured for diskless operation. In diskless mode all data from the software image will be transferred into the nodes memory by the node-installer. The obvious advantage is the elimination of the physical disk, cutting power consumption and reducing the chance of hardware failure. On the other hand some of the nodes memory will no longer be available for user applications. By default the amount of memory used for holding all file system data is unlimited. This means that creating very large files could cause a node to run out of memory and crash. If required, the maximum amount of memory used for the file system can be limited. This can be done by setting a maximum using the `maxMemSize` attribute. The default value of `0` results in no limitations for the file system. Note that setting a limit will not necessarily prevent the node from crashing as some processes might not deal properly with situations when there is no more free space on the filesystem.

```
<?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>
</diskSetup>
```

D.8 Example: Semi diskless

It is also possible to mix diskless operation as described above with certain parts of the file system on physical disk. In this example all data in `/local` will be on the physical disk, the rest will be in memory. Note that when nodes operate in semi diskless mode the node-installer will always use the `excludelistfullinstall` when synchronizing the software image to memory and disk.

```
<?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>
```

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 6.3). These are sometimes useful when building a cluster for carrying out troubleshooting or workarounds with non-standard hardware for these stages. The scripts are stored in the CMDaemon database, rather than in the filesystem as plain text files, because they run before the init process for the node takes over.

They are accessible for viewing and editing:

- In cmgui, using the Node Categories or Slave 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. The output is truncated in the two following examples at the point where the editor starts up:

Example

```
[root@bright51 ~]# cmsh
[bright51]% category use slave
[bright51->category[slave]]% show | grep script
Parameter                      Value
------------------------------ ------------------------------------------------
Finalize script                <1367 bytes>
Initialize script              <0 bytes>
[bright51->category[slave]]% set initializescript
```

Example

```
[bright51]% device use node001
[bright51->device[node001]]%
[bright51->device[node001]]% set finalizescript
```

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 6.5.2.

For the `initialize` and `finalize` scripts, node-specific customizations can be made from a script using environment variables. The following script does not actually do anything useful, but does show available variables:

```
#!/bin/bash

echo "HOSTNAME=$HOSTNAME"
echo "HWTAG=$HWTAG"
echo "MAC=$MAC"
echo "PARTITION=$PARTITION"
echo "RACKINDEX=$RACKINDEX"
echo "DEVICEPOSITION=$DEVICEPOSITION"
echo "DEVICEHEIGHT=$DEVICEHEIGHT"
echo "INSTALLMODE=$INSTALLMODE"
echo "CATEGORY=$CATEGORY"
echo "POWERCONTROL=$POWERCONTROL"
echo "PARTITION=$PARTITION"
echo "GATEWAY=$GATEWAY"
echo "PDUS=$PDUS"
echo "ETHERNETSWITCH=$ETHERNETSWITCH"

for interface in $INTERFACES
do
  eval type=\$INTERFACE_${interface}_TYPE
  eval ip=\$INTERFACE_${interface}_IP
  eval mask=\$INTERFACE_${interface}_NETMASK

  echo "$interface type=$type"
  echo "$interface ip=$ip"
  echo "$interface netmask=$mask"
done
```

The `initialize` script runs after the install mode type and execution have been determined, but before unloading specific drivers and before partitions are checked and filesystems mounted. Data output can be written by it to writeable parts of the NFS drive.

For a `finalize` script, 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 there if needed, for example predetermined configuration files from the NFS drive for a particular node:

Example

```
#!/bin/bash
ln -sf /etc/myapp.conf.$HOSTNAME /localdisk/etc/myapp.conf
```

One way of writing the environment out to the local disk is to redirect and append the variables to the drive. For example, as illustrated by the following:

Example

```
#!/bin/bash
echo "HOSTNAME=$HOSTNAME" > /localdisk/env
echo "HWTAG=$HWTAG" >> /localdisk/env
echo "MAC=$MAC" >> /localdisk/env
```

Data stored earlier by an `initialize` script can be copied over by a `finalize` script from the NFS drive to the local hard drive, and placed alongside the output of a `finalize` script. This is useful for comparison purposes after the node is fully running.

Writing to the drive of the local node means that the directory being written to may need to be added to the `excludelistsyncinstall` and/or `excludelistfullinstall` exclude lists, to prevent it being overwritten by a "known good state" directory during provisioning.

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. Boot head node from Bright Cluster Manager DVD.
2. Select `Install Bright Cluster Manager` in the boot menu
3. Once the installation environment has been started, choose Normal installation mode and click `Continue`.
4. Accept the License Agreements for Bright Cluster Manager and the Linux distribution and click `Continue`.
5. Click `Continue` on kernel modules screen.
6. 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`.
7. 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`.
8. Choose a network layout and click `Continue`. The first layout is the most commonly used. The rest of this appendix will assume the first layout was chosen.
9. Optionally add an InfiniBand network and configure the use of IPMI/iLO BMCs on the nodes. Adding an IPMI/iLO network is only necessary when the IPMI/iLO interfaces should be configured in a different IP subnet. When done, click `Continue`.
10. 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 will be handled in one of the next screens. Click `Continue`.

11. Add and remove DNS nameservers and DNS search domains, and click `Continue`
12. Assign an IP address for the head node on `externalnet`. This is the IP address that will be used to access the cluster over the network.
13. If necessary, modify the node properties. When IPMI/iLO interfaces will reside in the same IP subnet, set an IP Offset for the `ipmi0` interface. Click `Continue` to continue.
14. 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.
15. Select the DVD drive containing the Bright Cluster Manager DVD and click `Continue`.
16. 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.
17. 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.
18. Select a time-zone and optionally add NTP time-servers. Click `Continue` to continue.
19. Enter a hostname for the head node. Enter a password that will be used for system administration twice and click `Continue`.
20. Configure text or graphical consoles for the nodes in the cluster.
21. Review the network summary screen, and click the `Start` button to start the installation.
22. Wait until installation has completed, and click `Reboot`.

F.2 First Boot

1. Ensure that the head node boots from the first harddrive 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 will be started and a certificate request will be sent to the head node.

   If the node does not make it to the Node Installer, it is possible that additional kernel modules are needed. Section 6.7 contains more information on how to diagnose problems during the node booting process.

4. To manually identify each node, select `Manually select node` on each node, and identify the node manually by selecting a node-entry from the list, and choosing `Accept`.

   Optional: To allow nodes to be identified based on Ethernet switch ports, consult section 4.5

   Optional: For larger clusters, assigning identities to nodes can be tedious to do manually. The Node Identification Wizard (Section 6.3.2) running from `cmgui` automates the process so that nodes do not require manual identification at the console.

5. The node will now be provisioned and will eventually boot. In case of problems, consult section 6.7

6. *Optional:* To configure power management, consult chapter 5.

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 -Y 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 that will 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 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. Bright Cluster Manager 5.1 uses SGE version 6.2 update 5, which was the last release from Sun Microsystems, and remains in widespread use.

SGE services should be handled using CMDaemon, as explained in section 8.3. 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 (see related: qalter, qresub)
- `qtcsh`: start `csh`-based interactive session with a node

G.2 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.3 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               sets individual server type stuff

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 10.4 Reference Guide*. There is further extensive documentation for PBS Pro administrators in the *PBS Professional 10.4 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.

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
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
CMDMemUsed	Resident memory used by CMDaemon
CMDState	State in which CMDaemon is running (head: 0, node: 1, failover:2)
CMDSystime	Time spent by CMDaemon in system mode
CMDUsertime	Time spent by CMDaemon in user mode

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
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
FreeSpace	Free space for non-root on a mount point
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
LoadFive	Load average on 5 minutes
LoadOne	Load average on 1 minute
MemoryFree	Free system memory
MemoryUsed	Used system memory
MergedReads	Total number of merged reads
MergedWrites	Total number of merged writes
NetworkBytesRecv	Cluster-wide number of bytes received on all networks

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
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
PDUBankLoad	Total PDU bank load
PDULoad	Total PDU phase load
PDUUptime	PDU uptime
PacketsRecv	Number of received packets
PacketsSent	Number of packets sent
PhaseLoad	Cluster-wide phase load
ProcessCount	Total number of processes
QueuedJobs	Number of queued jobs
RackSensorHumidity	Rack sensor humidity
RackSensorTemp	Rack sensor Temperature
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
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
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

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
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
Writes	Total number of writes completed successfully
await_sda*	The average time (in milliseconds) for I/O requests issued to device sda to be served
gpu*	sample_gpu
ilo*	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

...continues

Table H.1.1: List Of Metrics...continued

Name	Description
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
responsiveness*	sample_responsiveness
sdt*	sample_sdt
tcpCurrEstab	Number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT
tcpInErrs	Total number of IP segments received in error
tcpRetransSegs	Total number of IP segments retransmitted
testcollection*	testmetriccollection
testmetric*	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
util_sda*	Percentage of CPU time during which I/O requests were

* 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                samplingonslave
State flapping count           7
Timeout                        5
Valid for                      slave,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.
- `samplingonslave` (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:

- `Slave 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?
`cmsh`	Is `cmsh` available?
`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?
`ldap`	Can the ID of the user be looked up with LDAP?
`mounts`	Are all mounts defined in the cluster manager 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`*"*
`portchecker`	Is the specified port on the specified host open for TCP (default) or UDP connections?
`rogueprocess`	Are the user processes that are running legitimate (ie, not 'rogue')?
`ssh2node`	Is passwordless ssh login from head to node working?
`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                samplingonslave
State flapping count           7
Timeout                        10
Valid for                      slave,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

```
[...]% add responsiveness
[...[responsiveness]]% set command /cm/local/apps/cmd/scripts/metrics/sample_responsiveness
[...]% set classofmetric prototype; commit
```

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.

J

Changing The Network Parameters Of The Head Node

J.1 Introduction

After a cluster physically arrives at its site, the first configuration task that an administrator usually faces is to change the network settings to suit the network at the site. How to configure network interfaces is detailed in section 4.2.1 of the *Bright Cluster Manager Administrator Manual*, and is easy to do. However, there is some reliance on having understood the material leading up to that section, which can take a while.

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.

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
```