Bright Cluster Manager 5.0

Administrator Manual

Revision: 297

Date: Thu, 25 Mar 2010

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. 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 5.0 cluster environment. This manual is intended for those responsible for the administration of a cluster running Bright Cluster Manager and for those who need to be able to install and set up the cluster for use.

This manual covers administration topics which are specific to the Bright Cluster Manager environment. Although it does cover some aspects of general Linux system administration, it is by no means comprehensive in this area. Readers are advised to make themselves familiar with basic Linux system administration.

This manual is not intended for users who are solely interested in interacting with the cluster to run compute jobs. The User Manual is intended to get end-users up to speed with the user environment and workload management system.

Updated versions of both this Administrator Manual and the User Manual are always available on the cluster in the following location:

```
/cm/shared/docs/cm
```

Our manuals constantly evolve to match the development of the Bright Cluster Manager environment, the addition of new hardware and/or applications and the incorporation of customer feedback. Your input as an administrator and/or user is of great value to us and we would be very grateful if you could report any comments, suggestions or corrections to us at `manuals@brightcomputing.com`.

1

Introduction

Bright Cluster Manager is a cluster operating system and management environment built on top of a major Linux distribution. Bright Cluster Manager 5.0 is available with the following Linux distribitions:

- 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.

1.1 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*

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 *slave nodes*, most of which are normally *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 *slave nodes* as well (e.g. storage nodes and login nodes). Slave 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 synchronised 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 set-ups, 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 directly connected to the internal network. Instead, only the head node is connected to the the external network.

Figure 1.1 shows a graphical representation of a typical cluster network set up.

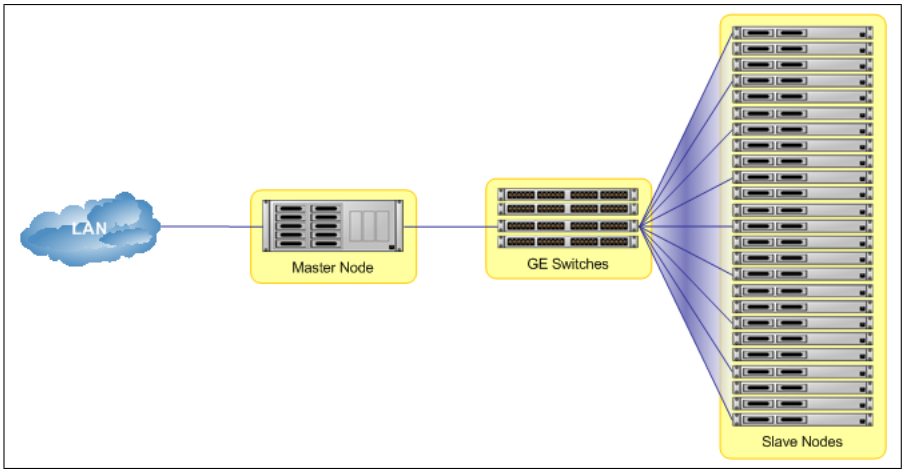

Figure 1.1: Cluster network

Most clusters are equipped with one, or several, 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.

1.2 Bright Cluster Manager Software 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 running applications that require extensive computational resources.

1.3 Organization of This Manual

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

In chapter 2, step-by-step instructions are given for 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. With the groundwork in place, chapter 4 will explain how to configure and further set up the cluster after the software installation of the head node.

Chapter 5 describes how power management within the cluster works and chapter 6 provides full details on node provisioning.

Chapter 7 demonstrates a number of techniques and tricks for working with software images and keeping images up to date. Several useful tips and tricks for day to day monitoring are summarized in chapter 8. In chapter 9 a number of third party software packages that play a role in Bright Cluster Manager are described.

For some clusters a high-availability set-up involving redundant head nodes can be useful. Details on high availability features provided in Bright Cluster Manager as well as set-up instructions, are provided in chapter 10.

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

2

Installing Bright Cluster Manager

This chapter shall describe the steps of installing Bright Cluster Manager on the head node of a cluster. Sections 2.1 and 2.2 list hardware requirements and supported hardware. Step-by-step instructions on installing Bright Cluster Manager from a DVD on a head node are provided in section 2.3.

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 (for diskless: at least 4GB recommended)
- 1 Gigabit Ethernet NIC

2.2 Supported Hardware

The following hardware is supported:

Compute Nodes

- SuperMicro
- Dell
- IBM
- Asus

- HP
- Cray CX1

Ethernet Switches

- HP Procurve
- Nortel
- Cisco
- Dell
- SuperMicro

Power Distribution Units

- APC Switched Rack PDU

Management Controllers

- IPMI 1.5/2.0
- HP iLO 2

InfiniBand

- Most Infiniband HCAs

2.3 Head Node Installation

This section describes the steps involved in installing a Bright Cluster Manager head node. To start the installation, the head node must be booted from the Bright Cluster Manager DVD.

Welcome screen

The welcome screen as seen in Figure 2.1 displays version and license information. Two installation modes are available, the normal mode and express mode. Selecting the express mode of installation, will install the head node with the predefined configuration that the DVD was created with. Click on `Continue` to proceed to the Bright Cluster Manager software license screen.

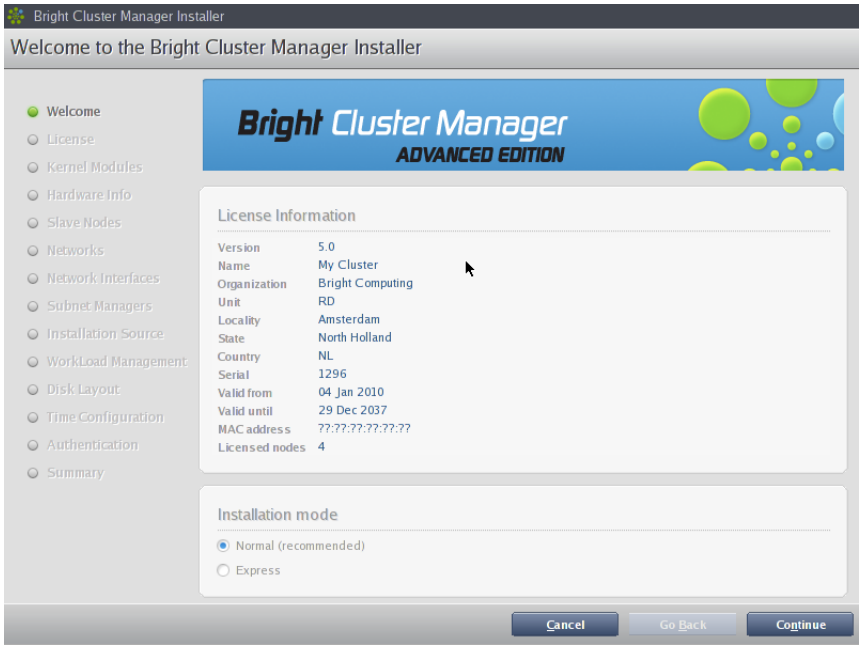

Figure 2.1: Welcome to Bright Cluster Manager head installation

Software License

The license screen as seen in Figure 2.2, explains the terms and conditions that apply to the Bright Cluster Manager software license. When the terms and conditions are accepted, then clicking `Continue` leads to the base linux distribution End User License Agreement (EULA), as seen in Figure 2.3. When the terms and conditions of the base distribution EULA are accepted, then clicking `Continue` leads to the kernel modules configuration screen, if you selected the nomal installation mode. If you selected the express mode of installation, it leads to the summary screen, as shown in Figure 2.17, showing an overview of the installation parameters, that were predefined.

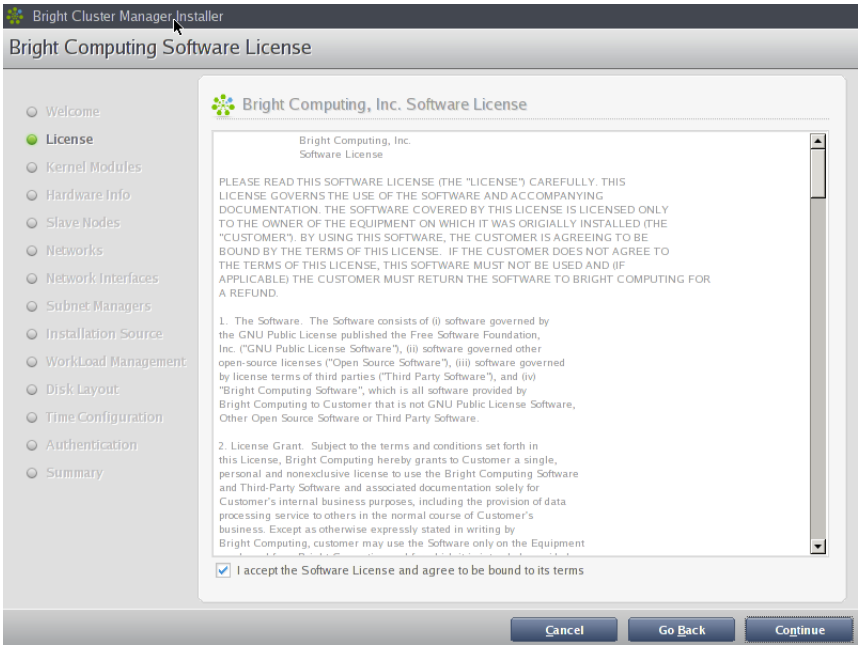

Figure 2.2: Bright Cluster Manager Software License

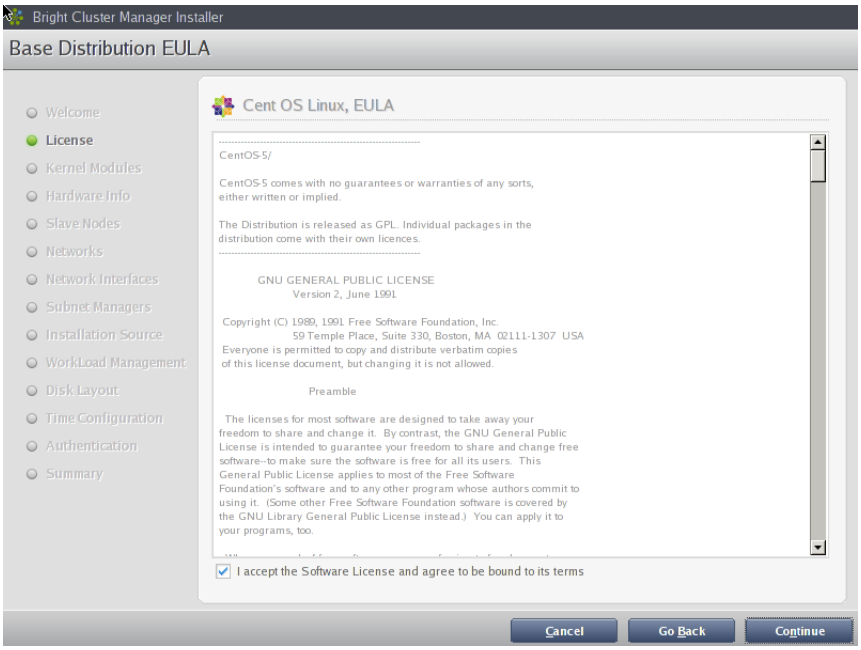

Figure 2.3: Base Distribution End User License Agreement

Kernel Modules Configuration

The kernel modules screen, as seen in Figure 2.4 shows the kernel modules that are recommended for loading based on hardware auto-detection. Clicking the `'plus'` button will open an input box, in which the module name and optionally module parameters can be entered. Clicking the `Add` button in the input box will add the kernel module. In order to remove a module, the module should be selected from the list and the `'minus'` button should be pressed. The arrow buttons may be used to move a ker-

nel module up or down in the list. The order in which kernel modules are loaded is relevant for the names that are assigned to devices (e.g. `sda`, `sdb`, `eth0`, `eth1`).

After optionally adding or removing kernel modules, clicking `Continue` leads to the hardware overview screen, described in the next paragraph.

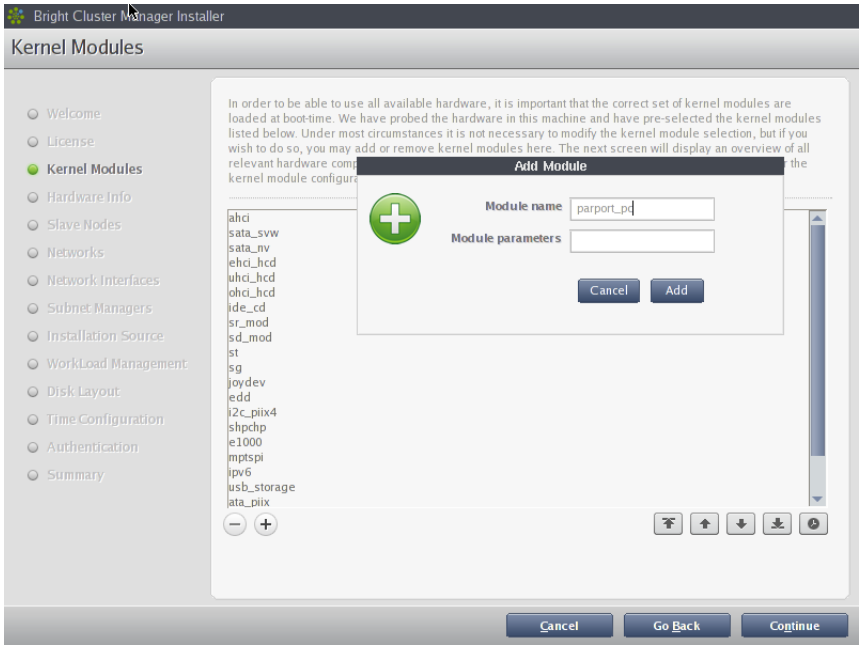

Figure 2.4: Kernel Modules

Hardware Overview

The hardware overview screen, as seen in Figure 2.5, provides an overview of hardware that has been detected depending on the kernel modules that have been loaded. If any hardware is not detected at this stage, click on the `GoBack` button to go back to the kernel modules screen (Figure 2.4) and add the appropriate modules, and then come back to this screen to see if the hardware has been detected. Clicking `Continue` in this screen leads to the slave nodes configuration screen (Figure 2.6), which is described in the following section.

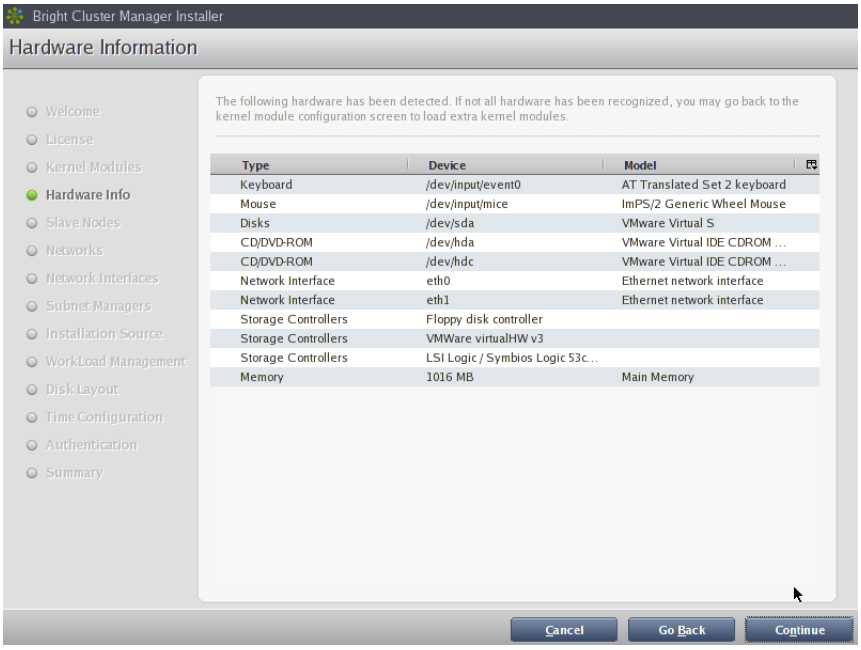

Figure 2.5: Hardware Overview

Slave Nodes Configuration

This screen allows you to configure your slave nodes. The number of racks, number of slave nodes, the slave basename and the number of slave digits, can all be configured on this screen. The maximum number of slave digits is limited to 5, in order to maintain a reasonably readable hostname. The number of slave nodes cannot exceed the number of licensed nodes. It is also possible to select to the node hardware manufacturer, if known. If the manufacturer is not known, then please select 'Other' from the list. This setting will be used to initialize monitoring parameters relevant for this kind of hardware. Clicking `Continue` in this screen leads to the network settings screen (Figure 2.7).

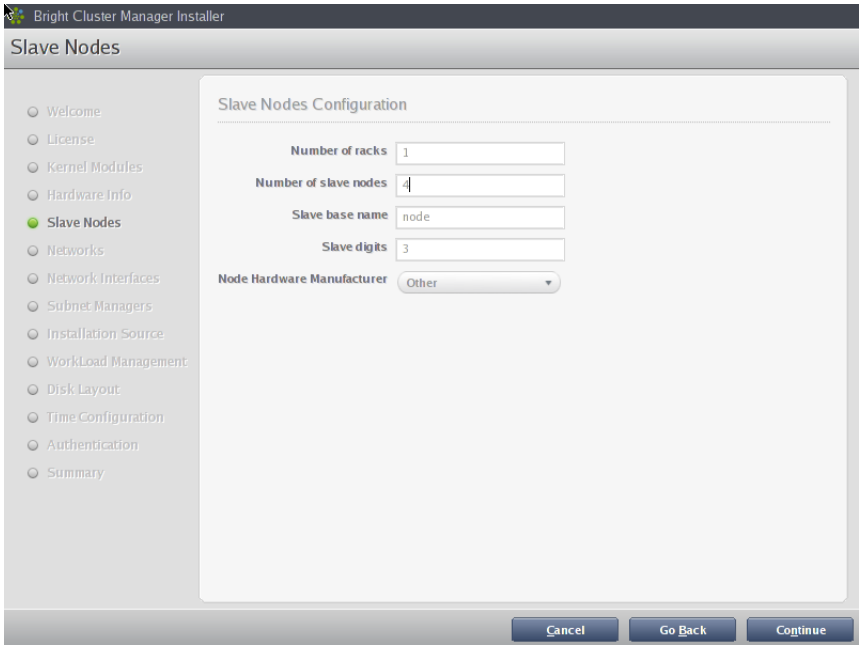

Figure 2.6: Slave Nodes Configuration

Networks Configuration

This screen displays the list of networks that have been predefined. The parameters of the network interfaces can be configured in this screen. It is possible to disable certain networks, and the corresponding master and slave interfaces on the network will be disabled as seen in Figure 2.8. Enabling a network will cause the corresponding interfaces on the network to be enabled (Figure 2.9). The networks can be one of three types: Ethernet | Infininand | IPMI. Additionally each network can be defined as an external network or a bootable network. A bootable network cannot be defined as an external network and vice-versa. Clicking `Continue` in this screen validates all network settings and if all settings are valid, leads to the master network interface configuration screen (Figure 2.9). If there are invalid settings for any of the enabled networks, an alert will be displayed, explaining the error, and it is not possible to proceed further, until all settings are valid.

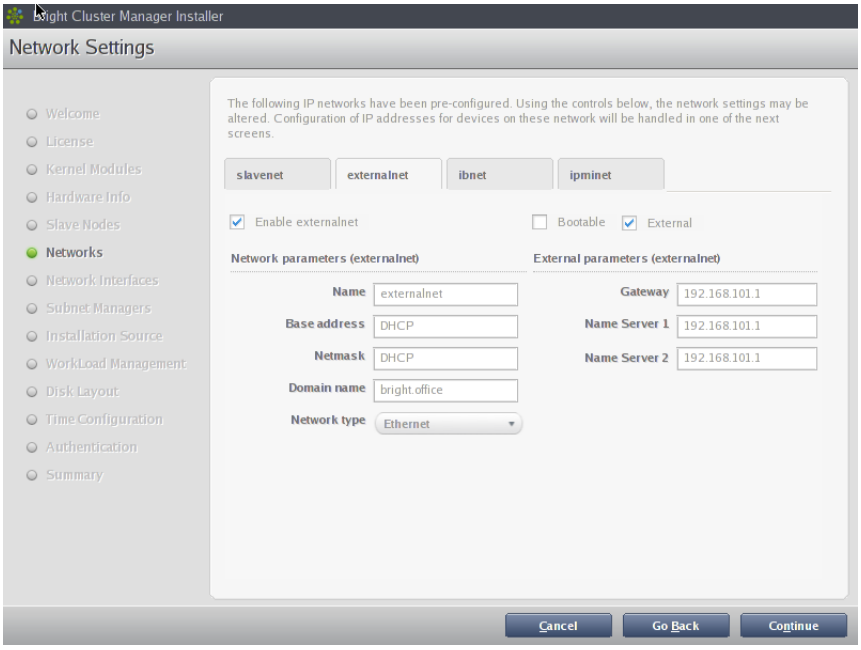

Figure 2.7: Networks Configuration

Network Interface Configuration

Figure 2.9 shows the list of network interfaces that have been predefined for the master and slave nodes. This screen consists of two sections, one for the master network interfaces configuration and one for the slave network interfaces configuration. Network interfaces can be disabled by unchecking the checkboxes corresponding to the interfaces. A different network can be selected for each interface, from the dropdown box in the 'Network' column. If the corresponding network settings had been changed (e.g., base address of the network) the IP address of the master node interface needs to be modified accordingly. If IP address settings are invalid, an alert will be displayed, explaining the error. Clicking `Continue` on this screen will validate IP address settings for all masternode interfaces, and if all setting are correct , and if infiniband networks were defined, leads to the subnet manager screen (Figure 2.10). If no infiniband networks are defined, or if infiniband networks were disabled on the networks settings screen, then clicking `Continue` on this screen leads to the DVD selection sceeen (Figure 2.11).

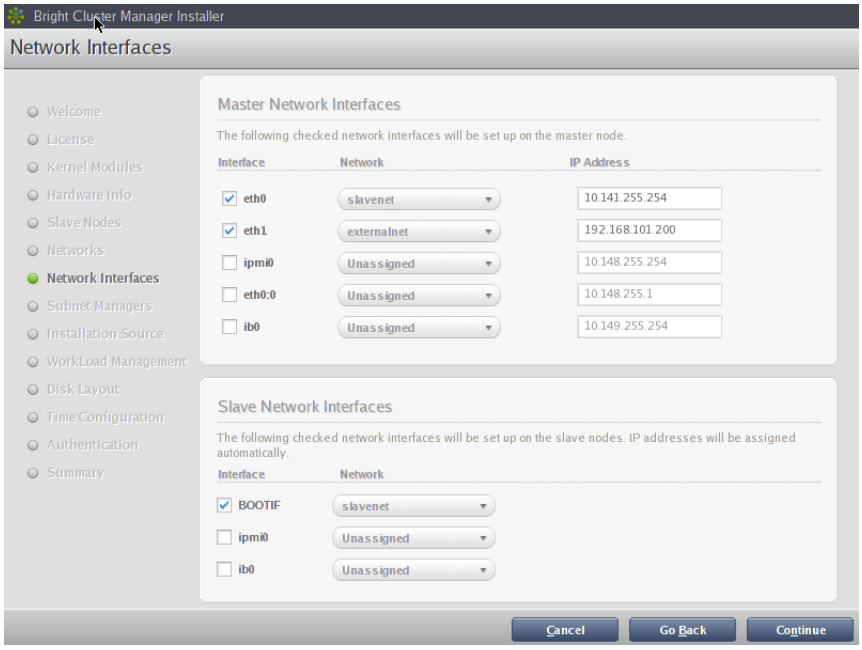

Figure 2.8: Network Interface Configuration

On the screen for configuring the network interfaces (Figure 2.9) , it is possible to choose a different network from the dropdown list in the 'Network' column. It is also possible to disable interfaces by unchecking the checkbox corresponding to the interface, or alternatively select 'Unassigned' from the dropdown box in the 'Network' column.

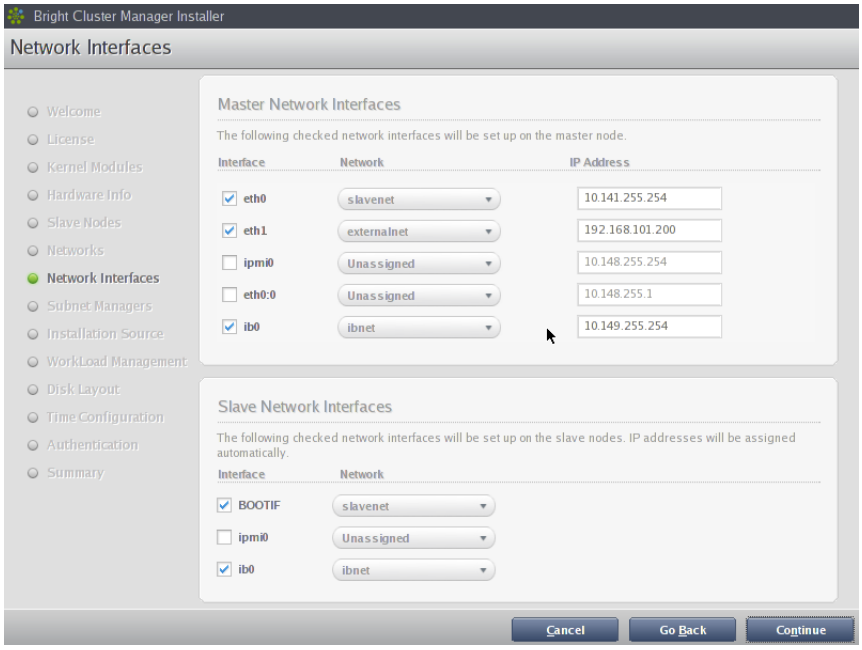

Figure 2.9: Network Interface Configuration - Infiniband interface enabled

Select subnet manager node

The screen in Figure 2.10 list all the nodes, that can be used to run the Infiniband subnet manager. This screen is only shown if an Infiniband network was defined. Select the nodes that should be assigned the role of a subnet manager. Click `Continue` to proceed to the DVD selection sceeen (Figure 2.11).

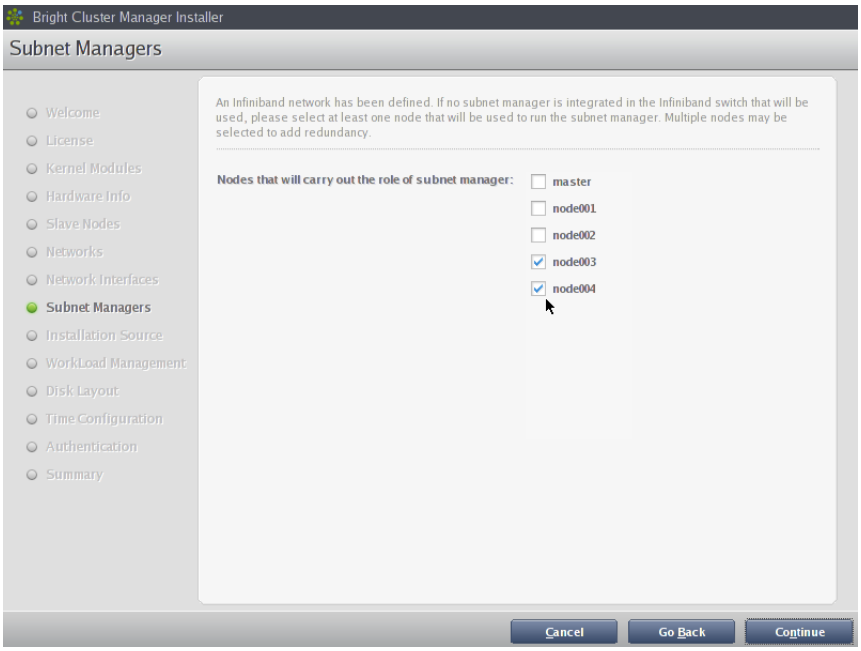

Figure 2.10: Subnet Manager Nodes

Select CD/DVD-ROM

The screen in Figure 2.11 lists all CD/DVD-ROM devices that have been detected. If multiple drives have been found, then select the appropriate drive that holds the Bright Cluster Manager DVD. After the correct drive has been selected, click `Continue` to proceed to the workload management setup screen.

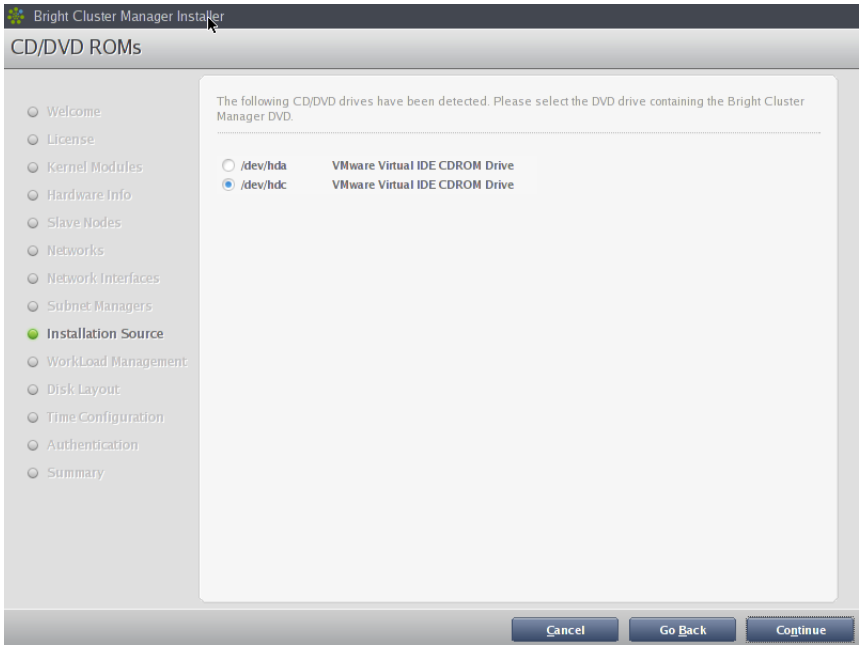

Figure 2.11: DVD Selection

Workload Management Configuration

Figure 2.12 shows the screen for configuring the workload management system. A workload management system is highly recommended to run compute jobs. A workload management system can be selected from the list of supported workload management systems. 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. The slots setting will be ignored if no workload management system is selected. If no changes are made, then the number of slots defaults to the CPU count on the head node. The masternode node may also be used as a compute node on small clusters. Please select if the master node should be used as a compute node. This setting will be ignored if no workload management system is selected. Clicking `Continue` on this screen leads to the disk partitioning and layouts screen (Figure 2.13).

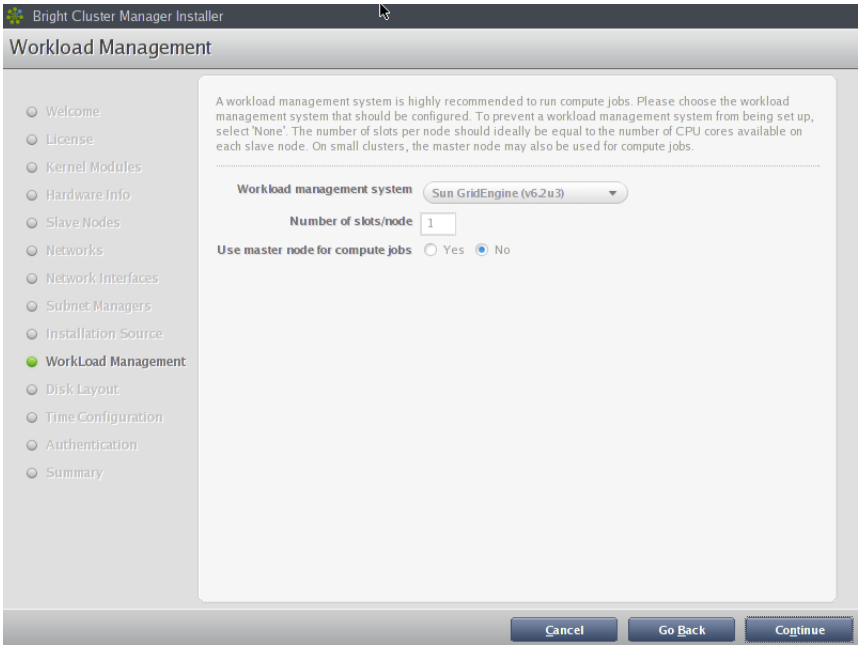

Figure 2.12: Workload Management Setup

Disk Partitioning and Layouts

This screen consists of two items, one for the master disk layout configuration and one for the slave disk layout configuration. When the edit button corresponding to the master or slave disk layout is clicked, the editor box is made visible (Figure 2.14). The `Save` and `Reset` buttons will be enabled upon edit. After making changes to the disk layout, click `Save` to save the changes that were made, or click `Reset`, to undo the changes that were made . Once saved, the changes cannot be reverted. A list of alternative layouts are available for both the master and the slave, and can be selected from the corresponding dropdown boxes. The selected disk layouts will be used for the installation. Clicking `Continue` on this screen leads to the time configuration screen.

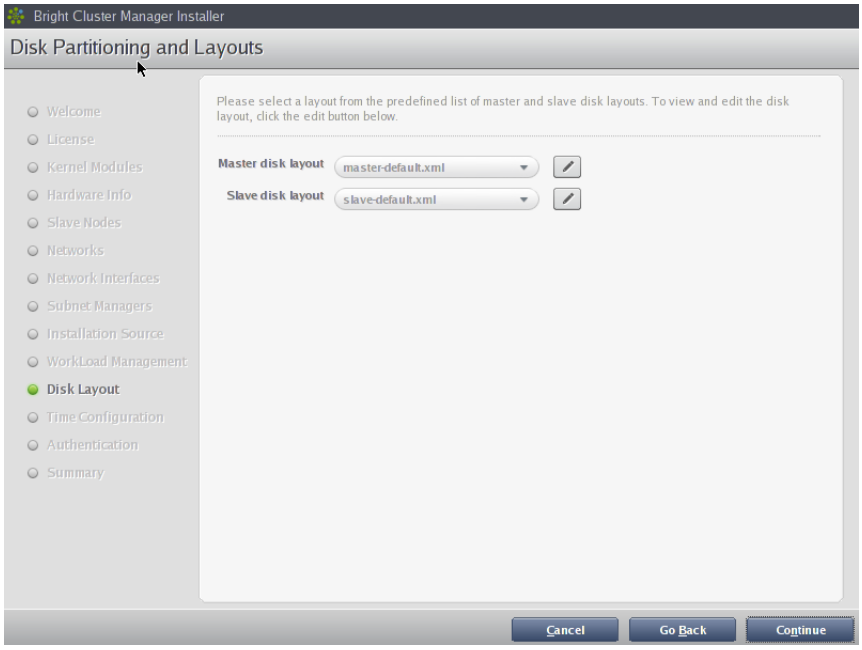

Figure 2.13: Disk Partitioning and Layouts

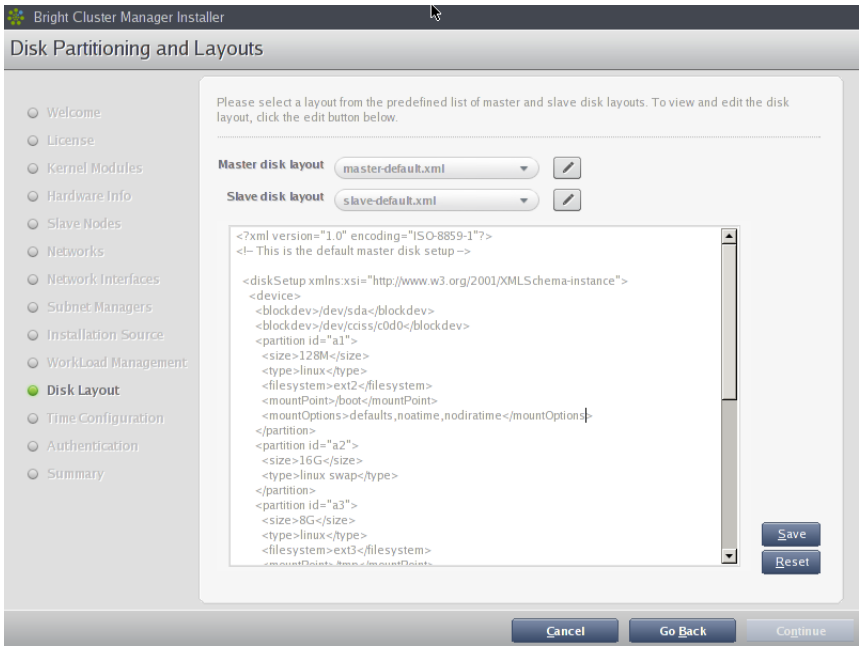

Figure 2.14: Edit Master Disk Partitioning

Time Configuration

Figure 2.15 shows the time configuration screen. The list of predefined timeservers are listed. Timeservers can be removed by selecting a timeserver from the list and clicking the minus button. Additional timeservers can be added by entering the name of the timeserver and clicking the plus button. Select appropriate timezone from the dropdown box if the timezone selected by default is not correct. Clicking `Continue` in this screen leads to the authentication screen (Figure 2.16).

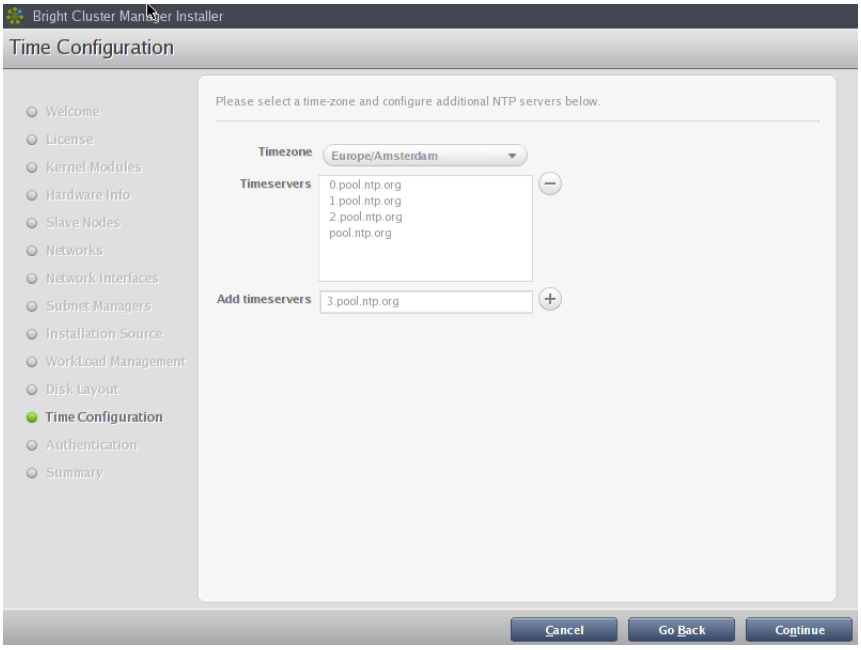

Figure 2.15: Time Configuration

Authentication

The authentication screen in Figure 2.16 requests the cluster administrator's password to be set. Please enter the password twice in order to proceed. The hostname of the head node can also be modified on this screen. Clicking `Continue` in this screen will validate the passwords that have been entered, and if validation succeeds, will lead to the summary screen.

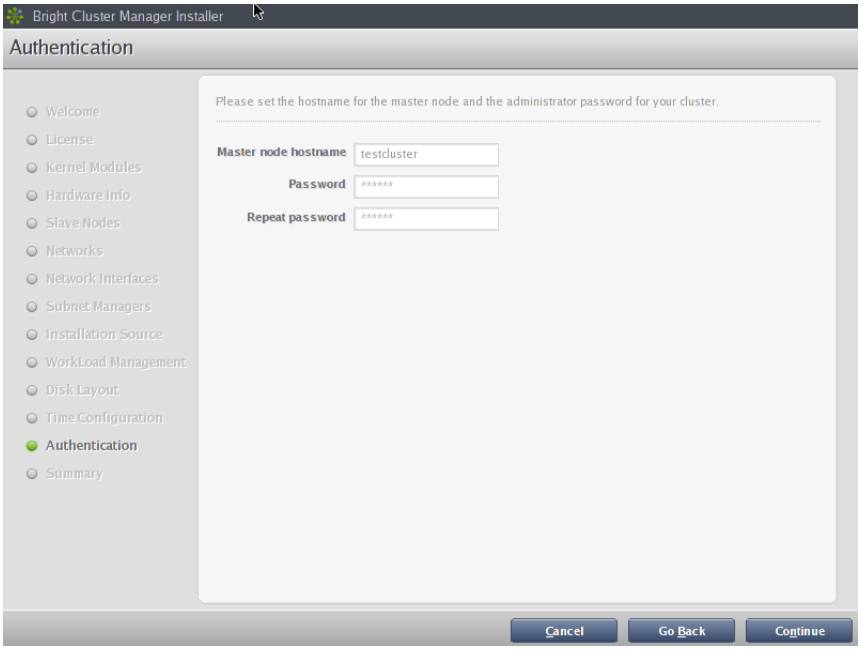

Figure 2.16: Authentication

Summary

The summary screen (Figure 2.17), shows a summary of installation settings and parameters configured during the previous stages or the predefined settings and parameters, if the express mode installation was selected. If any information in this screen is incorrect, navigate to previous screens and make necessary changes. When all information is correct, click `Start` to begin the installation. This will lead to the installation progress screen.

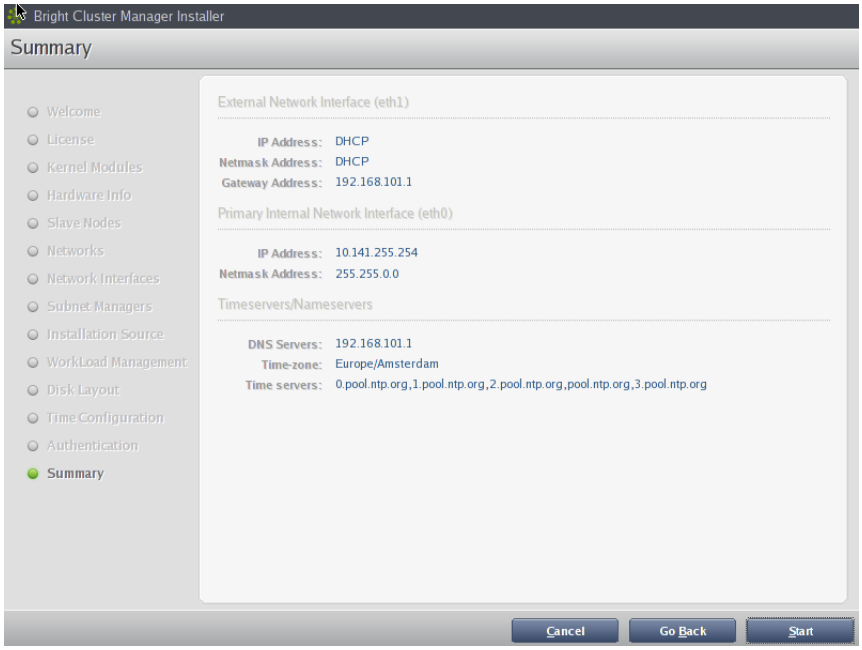

Figure 2.17: Summary of Installation Settings

Installation

The installation progress screen in Figure 2.18 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 2.19), click `Install Log` to view the installation log in detail, or click the `Reboot` button to restart the machine. Remember to remove the DVD or change the BIOS boot order, to boot from the hard drive on which Bright Cluster Manager has been installed.

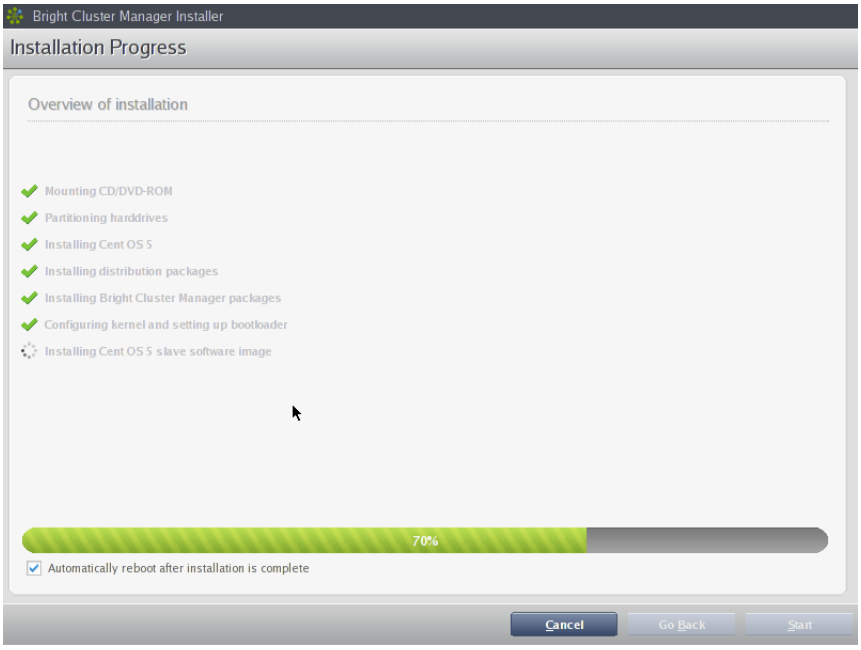

Figure 2.18: Installation Progress

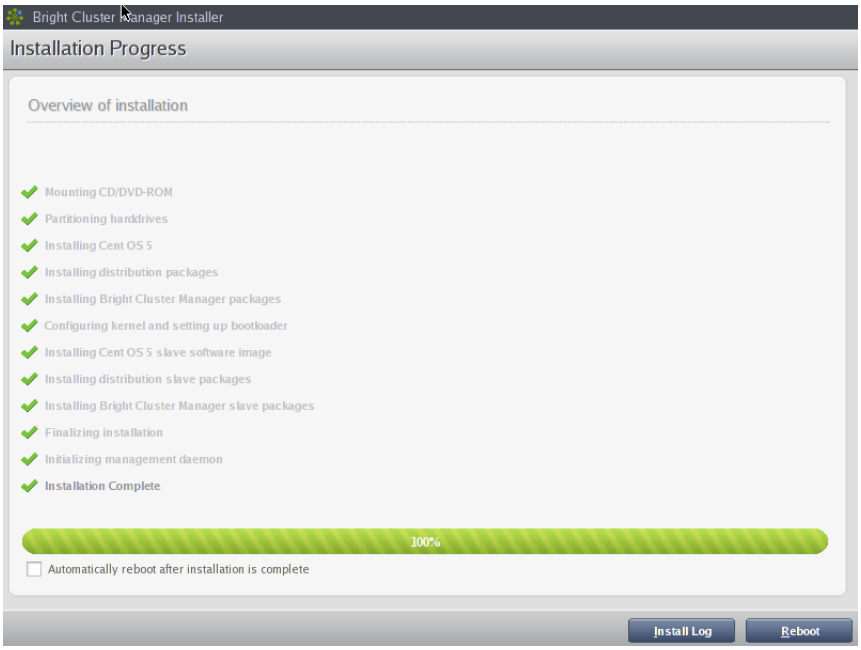

Figure 2.19: Installation Completed

After rebooting, the system will start and will present 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. The next chapter will introduce some of the tools and concepts that play a central role in Bright Cluster Manager. Chapter 4 will explain how to configure and further set up the cluster.

3

Cluster Management with Bright Cluster Manager

This chapter serves as an introduction to cluster management with Bright Cluster Manager. A cluster running Bright Cluster Manager exports a cluster management interface to the outside world, which can in principle be used by any application that is designed to communicate with the cluster. The primary applications that interact with the cluster through its management infrastructure, are the cluster management shell (`cmsh`) and cluster management GUI (`cmgui`) which will be introduced towards the end of this chapter (sections 3.4 and 3.6 respectively). However, in section 3.1 we start by introducing a number of concepts which are key to cluster management using Bright Cluster Manager. Section 3.7 will describe the basics of the cluster management daemon (cmdaemon) that is running on all nodes inside of a cluster.

Every user of a Bright Cluster Manager cluster will come across the *modules environment* at some point. The modules environment provides facilities to control many aspects of a users' interactive sessions and also the environment that compute jobs run in. Section 3.2 gives a short introduction on how the modules environment can be used by administrators.

Section 3.3 explains how authentication to the cluster management infrastructure works.

3.1 Concepts

In this section a number of concepts central to cluster management with Bright Cluster Manager will be 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:

- Master Node
- Slave Node
- 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 properties that may be set for a device and the operations that may be performed on a device, depend on the type of device. For example, it is possible to control the services running on a slave node device, which is not possible on a ethernet switch device.

Every device that is present in the cluster management infrastructure has a 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.

It is important to note the difference between the `DOWN` and `CLOSED` states. In the latter case the device is intentionally unavailable whereas in the former case the device was intended to be available, but instead is down.

3.1.2 Software Images

A *software image* is a blueprint for the contents of the local file-systems on a slave node. In practice, a software image is a directory on the head node containing a full Linux file-system. When a slave node boots, the node provisioning system is responsible for setting up the slave node with a copy of the software image. Once a node is fully booted, it is possible to instruct the slave node to re-synchronise 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 7.

3.1.3 Node Categories

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

Slave 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 set-up) 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 slave nodes inside the category. However, there is no requirement for a one-to-one correspondence between slave categories and software images. Therefore multiple slave categories may use the same software image.

Example

By default, all slave nodes are included in the `slave` category. Examples of alternative categories include:

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

3.1.4 Node Groups

A *node group* is one or more nodes that have been grouped for convenience. A node group can consist of any number of nodes frm any number of different node categories. Node groups are used primarily for performing operations on a number of nodes at a time. Since the nodes inside a node group do not necessarily share the same configuration, it is not possible to perform configuration changes through the node group. A node may be in 0 or more node groups at one time. In other words, nodes may be included in multiple node groups. In further contrast to node categories, node groups may also include head nodes.

Example

Node Group	Members
`broken`	`node087,node783,node917`
`headnodes`	`mycluster-m1,mycluster-m2`
`rack5`	`node212..node254`
`top`	`node042,node084,node126,node168,node210,node252`

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

Bright Cluster Manager aims to provide users with an optimal environment for running applications. Although the user environment is discussed in this manual, one particular aspect of the user environment is also relevant to administrators, and therefore requires a short introduction: the *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. Modules may be loaded and unloaded, and may be combined for greater flexibility. The `module avail` command can be used to obtain an overview of all available modules. Modules can be loaded with the following command:

```
module add modulename
```

Example

To compile an MPI application using version 2.5 of the Pathscale compiler and version 1.2.7 of MPICH for Gigabit Ethernet, one would use the following sequence of commands:

```
module add shared
module add pathscale/2.5
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 name prefixes may be used.

Loading the `shared` module is only required for the `root` user as this module is automatically added for normal users. Applications and their associated modules are divided in *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. Loading the `shared` module provides access to the modules belonging to the shared applications. On a cluster where a shared storage is not hosted on the head node, loading the `shared` module automatically for the `root` user is not recommended because it could obstruct `root` logins when the shared storage is unavailable. For this reason, the `shared` module is not loaded by default.

On clusters without external shared storage, it is safe for `root` to load the `shared` module automatically at every login. This can be set up by

using the following command as root:

```
module initadd shared
```

Similarly, using module initadd other modules may be added for automated loading at every login.

More details on the modules environment (from the perspective of an administrator) is provided in section 9.1.

3.3 Authentication

3.3.1 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 a person authenticating to the cluster management infrastructure must present his/her 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** the certificate and private key are stored as plain text in two separate PEM-encoded files.
- **PFX (a.k.a. PKCS12)** 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 for convenient password-based encryption of the private key data.

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 default password with which the admin.pfx has been encrypted is system. To change the password of a PFX file, the passwdpfx utility can be used which is part of the cmd module:

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

3.3.2 Profiles

A certificate that is used for authenticating to the cluster management infrastructure contains a so called *profile* which determines which cluster management operations the holder of the certificate 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 users of those certificates access to a pre-defined subset of the cluster management functionality.

3.4 Cluster Management GUI

The cluster management GUI (`cmgui`) is the graphical interface to cluster management in Bright Cluster Manager. Typically `cmgui` runs on the administrator's desktop computer, but alternatively it may be run on the head node or on a login node of the cluster using X11-forwarding. This section introduces the basics of `cmgui`.

3.4.1 Starting Cluster Management GUI

To start the cluster management GUI on the cluster, an administrator may connect to the cluster using X11-forwarding, load the `cmgui` module and run the `cmgui` command.

Example

```
user@desktop:~> ssh -Y root@mycluster
...
[root@mycluster ~]# module load shared cmgui
[root@mycluster ~]# cmgui
```

To install the cluster management GUI on a desktop computer running Linux or Windows, the binary installation package must be downloaded first. Redistributable installation packages of `cmgui` are available on any Bright Cluster Manager cluster in the following directory:

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

Installation packages are available for a number of different Linux distributions and for Windows XP/Vista [1]. Installing `cmgui` on a Windows desktop is a matter of executing the installer and going through the installation procedure. After the installation, `cmgui` can be started through the Start menu.

Installing the cluster management GUI on one of the supported Linux distributions involves untarring the `tar.bz2` file and if necessary installing a number of dependency packages. To start the cluster management GUI, simply change into the `cmgui` directory and execute the `cmgui` script.

Example

```
user@desktop:~> tar -xjf cmgui-4.0-r786-suse110-x86_64.tar.bz2
user@desktop:~> cd cmgui
user@desktop:~/cmgui> ./cmgui
```

[1] A MacOS X version will be available in the future.

If the cmgui script reports unresolved symbols, additional packages from the Linux distribution will have to be installed. For details on how to install these packages, please consult the distribution's documentation on installing additional software packages.

At least the following software packages 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, a certificate and private key are required to connect to a cluster. 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:admin.pfx \
                    ~/cmgui-keys/mycluster-admin.pfx
```

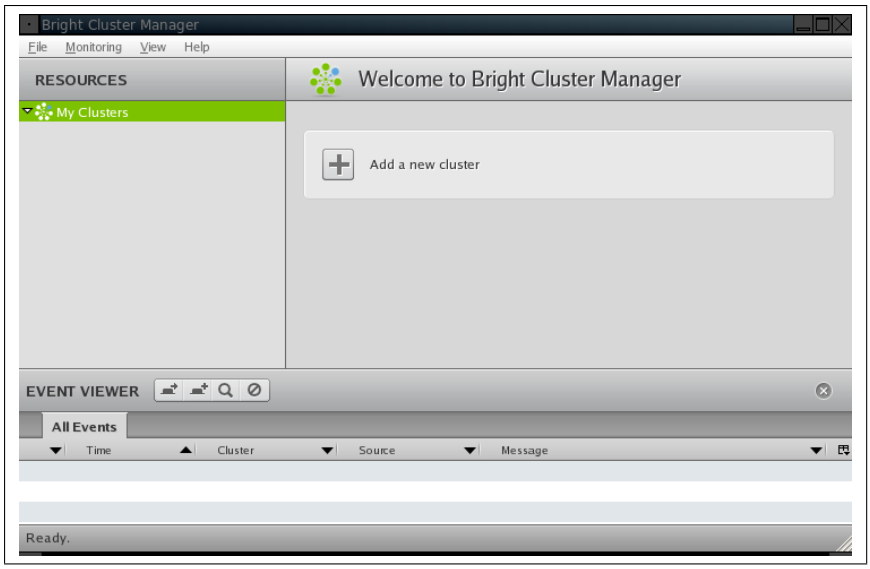

Figure 3.1: Cluster Management GUI welcome screen

When cmgui is started for the first time, the welcome screen (Figure 3.1) is displayed. The first step in configuring cmgui for connections to a new Bright Cluster Manager cluster, is to add the cluster to the cmgui by pressing the "Add a new cluster" button in the welcome screen. Figure 3.2 shows the dialog window in which the connection parameters can

be entered. As mentioned in section 3.3, the default password for the PFX certificate file is `system`.

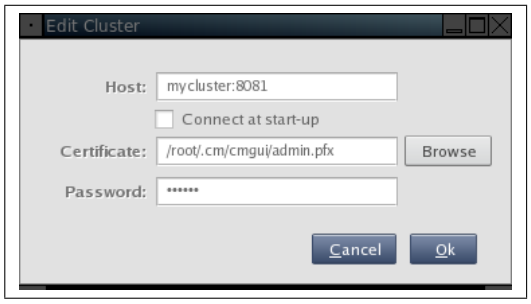

Figure 3.2: Edit Cluster dialog window

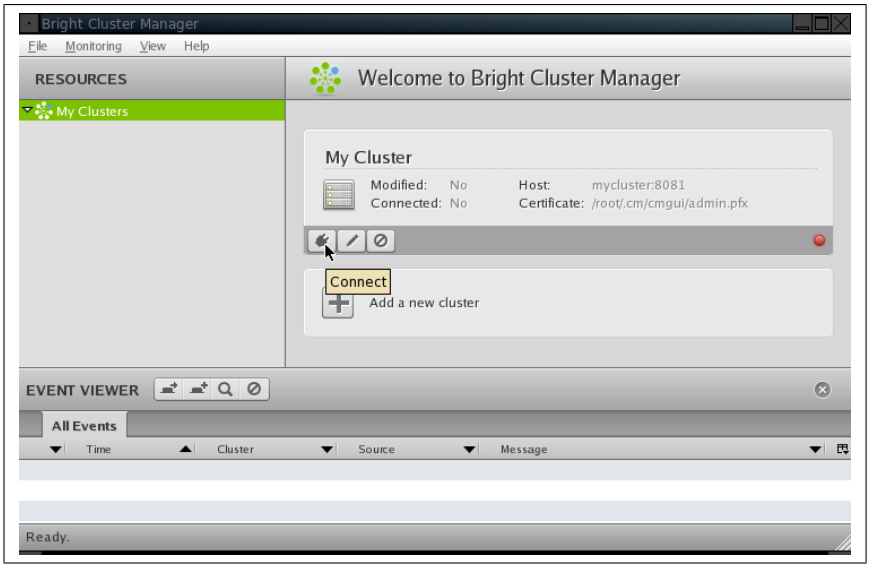

Figure 3.3: Connecting to a cluster

After the cluster has been added, clicking the Connect button (as displayed in figure 3.3) will set up a connection. After the connection to the cluster has been established, the `cmgui` display will look similar to figure 3.4.

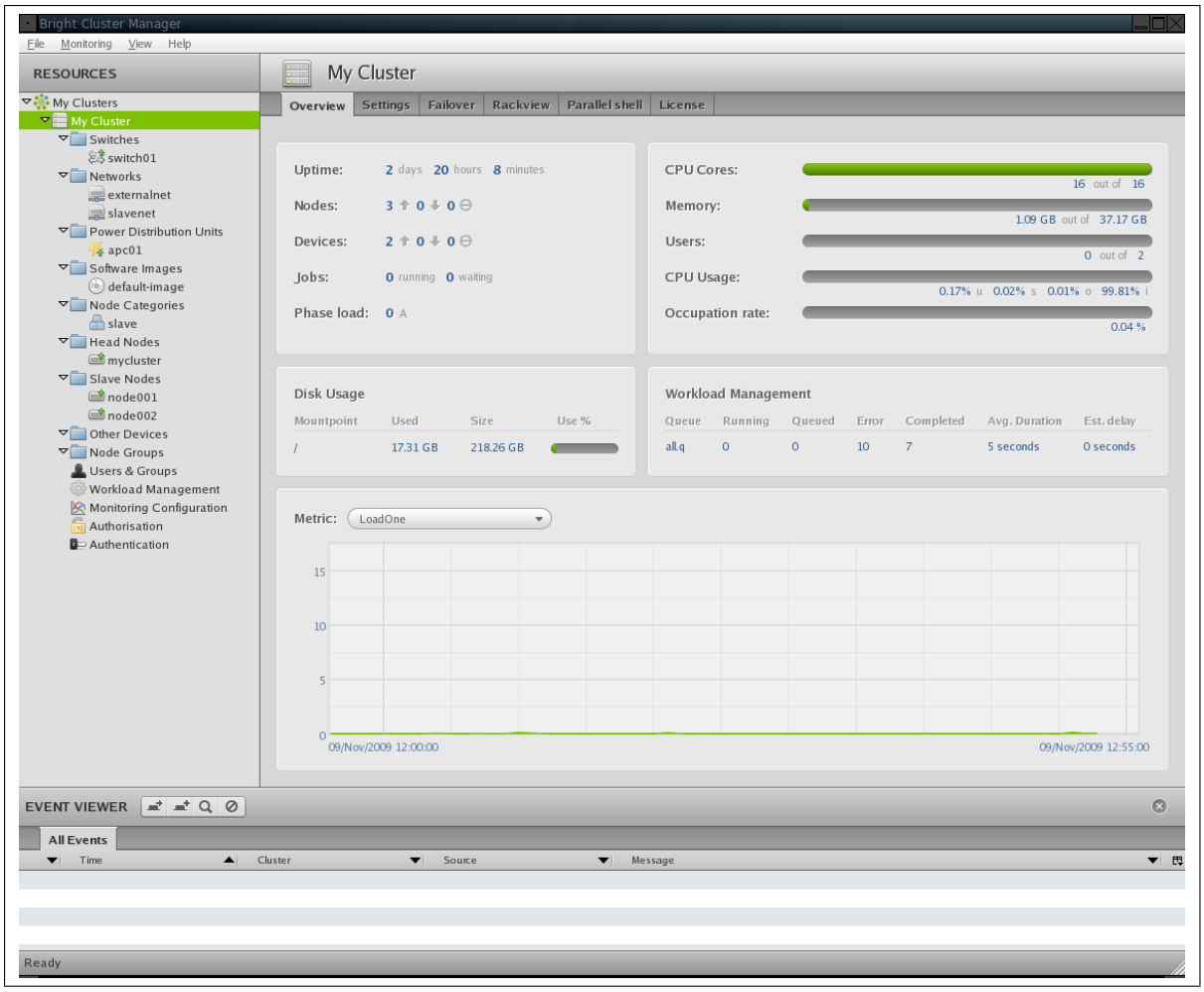

Figure 3.4: Cluster Overview

3.5 Navigating the Cluster Management GUI

The cluster management GUI (as displayed in figure 3.4) allows administrators to manage many aspects of their cluster. In the resource-tree on the left side of the screen, a user can select a particular resource and can subsequently manage that resource through the tab-pane on the middle-right part of the screen. In addition to hardware resources, the tree contains non-hardware resources such as `Users & Groups` and `Workload Management` which allow management of various aspects of the cluster.

The number of tabs displayed and the contents of these tabs depends on the type of resource selected in the tree. However, a number of standard tabs are available for most resources.

- `Overview`: this tab provides an overview containing the most important status details of the selected resource.
- `Tasks`: this tab gives access to controls that allow operations to be performed on the selected resource.
- `Settings`: this tab allows a user to configure properties of the selected resource.

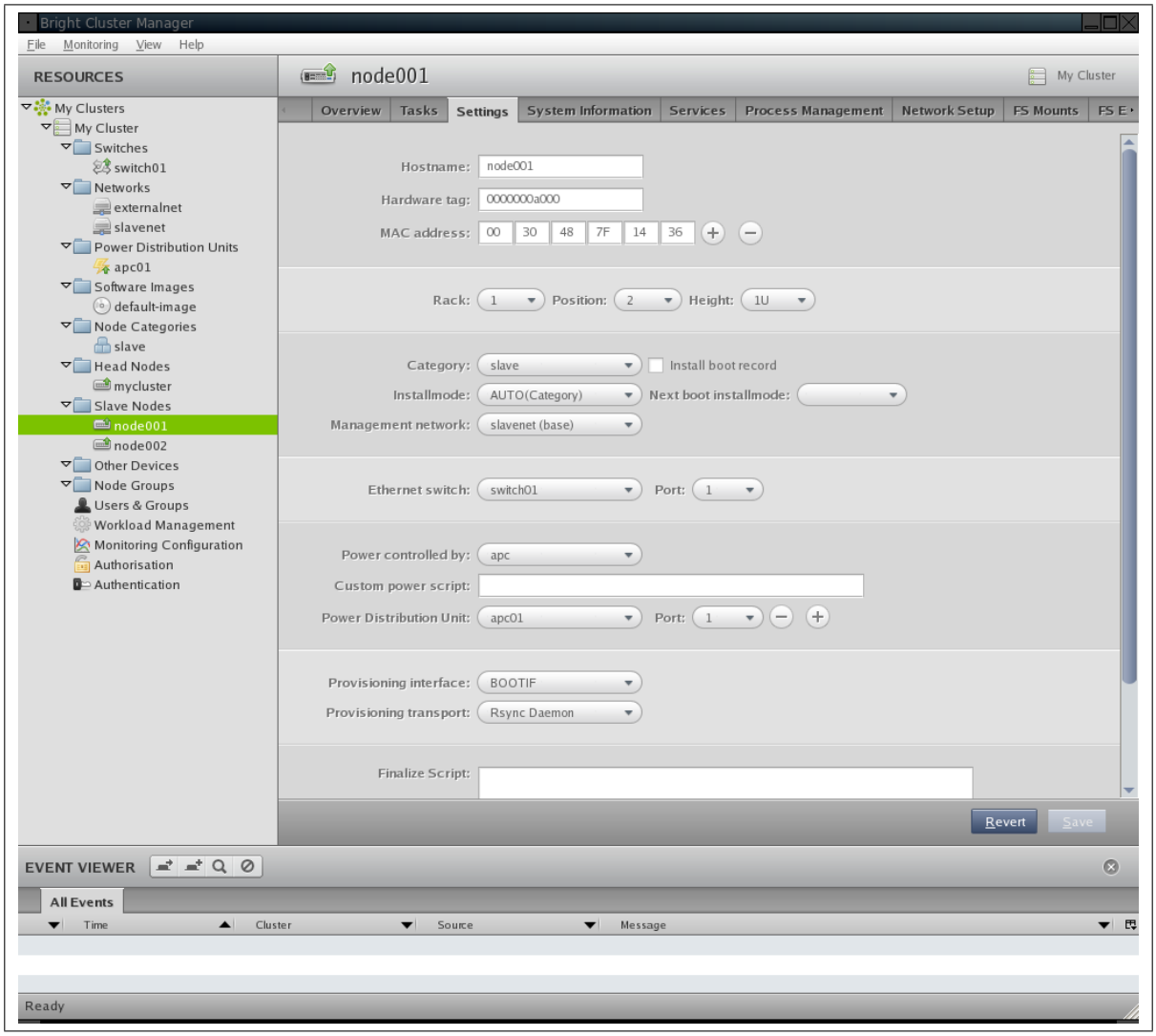

Figure 3.5: Node Settings

Figure 3.5 shows the `Settings` tab of the `node001` resource. The tab displays various properties that can be changed. The `Save` button on the bottom of the tab may be used to make changes active and permanent. Pressing the `Revert` will cause all unsaved changes to be undone.

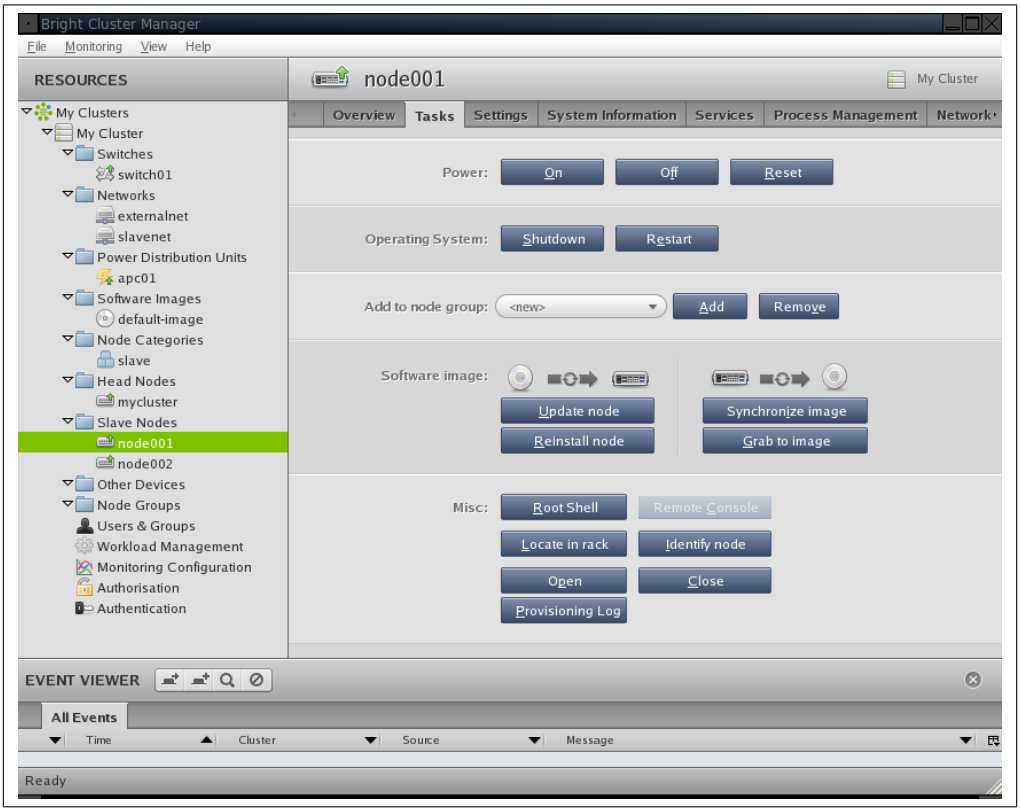

Figure 3.6: Node Tasks

Figure 3.6 shows the `Tasks` tab of the `node001` resource. The tab displays various operations that can be performed on the `node001` resource. Details on the semantics of the displayed node operations and settings are provided in the remaining chapters of this manual.

It is also possible to select a folder item in the tree (e.g. `Node Categories`, `Slave Nodes`, `Networks`). Selecting a folder item in the tree provides overviews for all resources inside the folder. In addition, it is possible to perform operations on groups of resources or selections of resources. Adding a new resource or removing a resource can be accomplished by selecting a folder and using the `Add` and `Remove` buttons in the `Overview` tab. Figure 3.7 shows the contents of the `Overview` tab for the `Slave Nodes` folder.

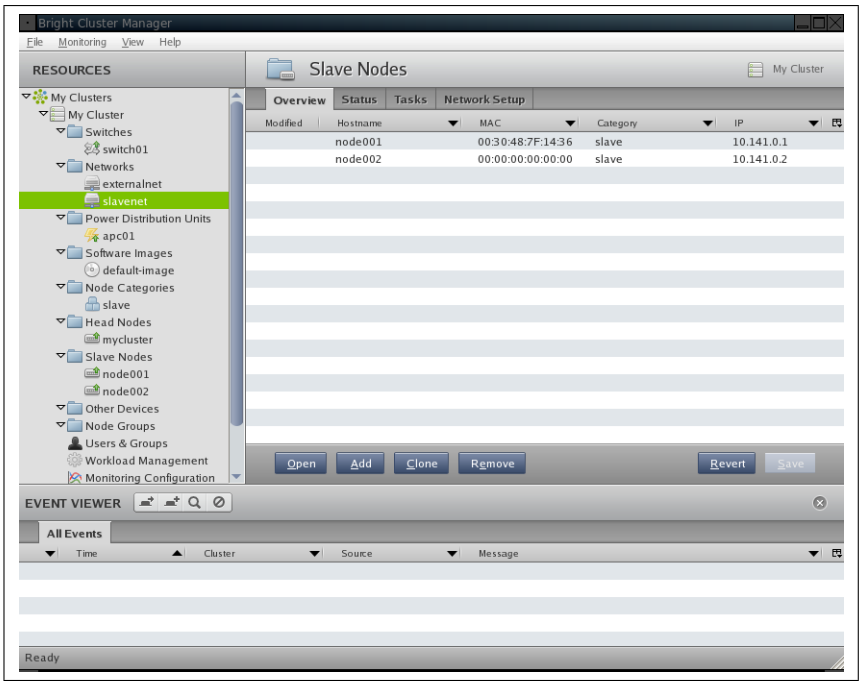

Figure 3.7: Nodes Overview

3.6 Cluster Management Shell

The cluster management shell (`cmsh`) is the command-line interface to cluster management in Bright Cluster Manager. Since the `cmsh` and `cmgui` give access to the same cluster management functionality, it is not necessary for an administrator to become familiar with both interfaces. Administrators only intending to manage a cluster through `cmgui` may safely skip this section.

Usually the `cmsh` is invoked from an interactive session (e.g. through `ssh`) on the head node, but in principle the `cmsh` can also be used to manage the cluster from outside. This section shall introduce the basics of `cmsh`.

3.6.1 Invoking `cmsh`

The `cmsh` can be invoked on the head node as follows:

```
[root@mycluster ~]# module load cmd
[root@mycluster ~]# cmsh
[mycluster]%
```

Running the `cmsh` without arguments starts an interactive cluster management session. To go back to the Unix shell, a user may enter the `quit` command:

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

The `cmsh` can also be used in batch mode by specifying a command using the `-c` flag.

```
[root@mycluster ~]# cmsh -c "main showprofile"
```

```
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
       --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

```
admin
[root@mycluster ~]#
```

Alternatively, commands can be piped to the cmsh:

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

Similar to Unix shells, cmsh sources ~/.cm/cmsh/.cmshrc upon start-up in both batch and interactive mode. This can be convenient for defining command aliases which may subsequently be used to abbreviate longer commands. Using the following contents for .cmshrc allows the ds command to be used as an alias for device status:

```
alias ds device status
```

Figure 3.8 shows complete usage information for cmsh.

3.6.2 Using cmsh

Through cmsh, an administrator can control many aspects of his/her cluster. In order not to overload a user with commands, the cluster management functionality has been grouped and placed in separate cmsh *modes*. When performing a cluster management operation, the first thing a user must do is switch to the appropriate mode. Figure 3.9 shows the top-level help which can be consulted inside cmsh by typing help. To enter

```
disconnect ................... Disconnect from cluster
connect ...................... Connect to cluster
quit ......................... Quit shell
exit ......................... Exit from current object or mode
help ......................... Display this help
run .......................... Execute cmsh commands from specified file
alias ........................ Set aliases
unalias ...................... Unset aliases
modified ..................... List modified objects
export ....................... Display list of aliases current list formats
events ....................... Manage events
list ......................... List state for all modes
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
test ......................... Enter test mode
user ......................... Enter user mode
```

Figure 3.9: Help in `cmsh` at top-level

a mode, a user must enter the name of a mode on the `cmsh` prompt. The `cmsh` prompt will change to reflect that the `cmsh` is in the requested mode. To leave a mode, the `exit` command can be 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]%
```

We have entered `device` mode, executed the `list` command and we have left `device` mode using the `exit` command

It is also possible to execute a command in a specific mode without entering that particular mode. This can be done by specifying the mode

before the command. Most commands also accept arguments which can be specified after the command. Multiple commands can be executed in one line by separating commands with semi-colons (i.e. ";")

Formally 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
slavenet     16            10.142.0.0       10.142.255.255
[mycluster->network]%
```

While we are in `network` mode, we execute the `status` command in `device` mode and pass argument `node001`. From the same command line we then execute the `list` command in the current mode (i.e. `network` mode).

The top-level commands listed in figure 3.9 are available regardless of which mode is active. Most notably the `help` command should be used to consult the help information for a particular command by passing the command as an argument. Invoking `help` without an argument provides a list of commands that may be used.

3.6.3 Working with objects

Most modes in `cmsh` work around a particular type of *objects*. For instance, `device` mode revolves around device objects whereas `network` mode is all about `network` objects. The commands that can be used for dealing with objects are the same in all modes. The following table lists the commands that may be used to deal with objects in a particular mode:

Command	Description
use	Make the specified object the *current object*
add	Create an object and make it the *current object*
clone	Clone an object and make it the *current object*
remove	Remove an object
commit	Commit local changes to an object to the cluster management infrastructure
refresh	Undo local changes to an object
list	List all objects
format	Set formating preferences for list output
show	Display all properties of an object
get	Display a particular property of an object
set	Set a particular property of an object
clear	Set empty value for a particular property of an object
append	Append a value to a particular list-property of an object
removefrom	Remove a given value from a particular list-property of an object
modified	Lists objects with uncommitted local changes
usedby	Lists objects that depend on a particular object
validate	Perform validation-check on the properties of an object

The commands listed above are demonstrated in a number of examples.

Example

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

We are in device mode and by issuing the use node001 command, node001 is made the *current object* (as can be seen in the prompt). We then issue the status command without passing an argument and receive status information for just node001. Making an object the *current object* makes all subsequent commands apply to this particular object. Using the exit command we unset the *current object*.

Example

```
[mycluster->device]% add slavenode node100 10.141.0.100
[mycluster->device[node100]]% set category test-slave
[mycluster->device[node100]]% commit
[mycluster->device[node100]]% exit
[mycluster->device]%
```

We are in device mode and we add a new object of slavenode type with name node100 and IP address 10.141.0.100. We set the category property of the object to test-slave and then commit the object to store it permanently. Note that until the newly added object has been committed, it remains a local change that is lost when the cmsh exits.

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 (slavenode) argument is also required, and an optional extra IP argument may also be specified. See help add in device mode for details.

Example

```
[mycluster->device]% clone node100 node101
[mycluster->device[node101]]% exit
[mycluster->device]% modified
Type                     Name
------------------------ --------
Cloned                   node101
[mycluster->device]% commit
Successfully commited 1 Device
[mycluster->device]%
[mycluster->device]% remove node100
[mycluster->device]%
```

The node object node100 that was created in the previous example is cloned to node101. We then check using the modified command what objects have uncommitted changes. We then commit the new object node101. The device node100 is subsequently removed by using the remove command. Note that while adding a device requires a commit to make the newly created object permanent, removing a device does not require a commit (i.e. the object is removed immediately).

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 this example, node101 was assigned IP address 10.141.0.101.

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
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]]%
```

We retrieve the category property of the node101 object by using the get command. We then change the property using the set command and confirm again using get that the value of the property has changed. We also use the modified command to confirm that the node101 has local

uncommitted changes. We then use the `refresh` command to undo the change that we made and use the `modified` command again to confirm that no local changes exist. Finally the `get` command is used to demonstrate that the local change has been undone.

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]%
```

We use the `set` and `get` commands to set and view the MAC address of `node101` without using the `use` command to make `node101` the *current object*. We then use the `clear` command to unset the value of the property. The result of `clear`ing a property depends on the type of the property. 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 used.

Example

```
[mycluster->device]% list -f hostname:10 ethernetswitch:15 ip
hostname   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]%
```

The `list` command is used to list all `device` objects. The `-f` flag is used to specify a format string. The format string is used to specify what properties need to be printed for each object and how many characters in the output lines should be used for each property. In this particular case we requested a list of the objects which displays the `hostname`, `ethernetswitch` and `ip` properties for each object. If we would have omitted the format argument, the default format string for `device` mode would have been used. To display the default format string, the `format` command may be used without parameters. 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`.

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]]%
```

When dealing with list-properties of an object, 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 example above we are appending and removing values 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 (see Chapter 5 for more information).

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]%
```

Removing an object is only possible if other objects do not have references to the object that is to be deleted. In order to display a list of objects that depend on a given object, the `usedby` command may be used. In the example above we are requesting which objects depend on device `apc01`. As can be seen, the `powerdistributionunits` property of device objects `node001` and `node002` contain references to object `apc01`. The fact that the `apc01` device has the `up` state is displayed as a dependency of `apc01` on its self. If we would want to remove the device `apc01`, we would have to remove the 2 references to it, and we would have to bring the device to the `closed` state by using the `close` command.

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]]%
```

Whenever committing changes to an object, the cluster management infrastructure will check the object being committed for consistency. If it is determined that one or more consistency requirements are not met, `cmsh` will report the violations that need to be resolved before the changes can be committed. The `validate` command allows an object to be checked for consistency without committing local changes.

3.6.4 Accessing Cluster Settings

The management infrastructure of Bright Cluster Manager has been 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 reflect on the way in which 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, which is 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 throughout the remainder of this manual.

Example

```
[root@mycluster ~]# cmsh
[mycluster]% partition use base
[mycluster->partition[base]]% show
Parameter                Value
------------------------ ----------------------------------------
Administrator e-mail
Default category         slave
Default gateway          192.168.101.1
Default software image   default-image
External address         0.0.0.0
External network         externalnet
FQDN
IPMI Password            *********
IPMI User ID             2
IPMI User name           ADMIN
Masternode               mycluster
Name servers             192.168.101.1
Primary network          slavenet
Search domains           brightcomputing.com
Slave digits             3
Slave name               node
Time servers             0.pool.ntp.org 1.pool.ntp.org 2.pool.ntp.+
Time zone                America/Los_Angeles
clusterName              My Cluster
defaultBurnConfig        default
name                     base
[mycluster->partition[base]]%
```

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 its self will start an interactive login sub-shell. By exiting the sub-shell, the user will return to the cmsh prompt.

In addition to executing commands from within cmsh, it is also possible to use the output of Unix shell commands as part of cmsh commands. This is done by using the "backtick syntax" that is also 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 through common operators such as >, >> and |.

Example

```
[mycluster]% device list > devices
[mycluster]% device status >> devices
[mycluster]% device list | grep node001
Type           Hostname      MAC                Ip
-------------- ------------- ------------------ -----------
SlaveNode      node001       00:E0:81:2E:F7:96  10.142.0.1
[mycluster]%
```

Looping over Objects

It is frequently convenient to be able to execute a cmsh command for several objects at once. In a number of cmsh modes, the foreach command is available to allow users to loop over lists of objects. A foreach command takes a list of space-separated object names and a list of commands that

must be enclosed by (and) characters. The foreach command will iterate over the specified objects making each of the objects the *current object*. For each loop iteration, the specified commands will be executed.
Formally, the foreach syntax is as follows:

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

Example

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

Using 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]%
```

3.7 Cluster Management Daemon

The *cluster management daemon* or *cmdaemon* is an server process that runs on all nodes of the cluster (including the head node). The cluster management daemons work together to make the cluster managable. 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 slave nodes.

The cluster management daemon is an application that is started automatically when a head node or slave node boots and will continue running until the node is shut down. Should the cluster management daemon be stopped manually for whatever reason, it will render the cluster unmanagable to a certain degree. However, in such a situation, 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, on which the cluster management daemon

accepts connections. All connections made to the cluster management daemon are SSL based. This is to ensure that all communication with the cluster management daemon is encrypted. In addition, authentication is also handled in the SSL layer by 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, when the cluster management daemon configuration file has been modified, a restart is necessary to activate the changes. The cluster management daemon can be controlled through its init-script:

```
/etc/init.d/cmd
```

When calling the init-script, a control operation must be passed as an argument. The following control operations are supported:

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

Through its configuration file, the cluster management daemon allows certain aspects of the configuration to be modified. The location of the configuration file is:

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

Under normal circumstances there is no need to modify the configuration settings on a default Bright Cluster Manager installation. Appendix C lists all recognized configuration file directives and describes how they can be used. After the configuration file is modified, the cluster management daemon must be restarted to activate the changes. The configuration file for the cluster management daemon running on a slave node is located inside of the software image that the slave is using.

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 will be overwritten, an event will be generated, and the original (modified) configuration file will be backed up to:

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

In some situations an administrator may have to override the contents of an automatically generated configuration file. In these cases the `FrozenFile` configuration file directive can be used in the CMDaemon configuration file.

Example

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

4

Configuring Your Cluster

After the Bright Cluster Manager software has been installed, 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 saved for 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 number of slave nodes that can be used on the cluster is limited by the maximum number of slave nodes specified in the license file. A license file is only usable on the machine for which it has been generated and can not be changed once it has been issued.

The license file is sometimes referred to as the *cluster certificate*, because it is in fact also used as the certificate of the head node (see section 3.3 for more information on certificate based authentication).

4.1.1 Displaying License Attributes

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

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

and the associated private key file is in:

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

To verify that the attributes of the license have been assigned the correct values, the `License` tab of the GUI can be used to display license details (see Figure 4.1). Alternatively the `licenseinfo` in `cmsh` `main` mode may be used.

Example

```
[root@mycluster ~]# module load cmsh
```

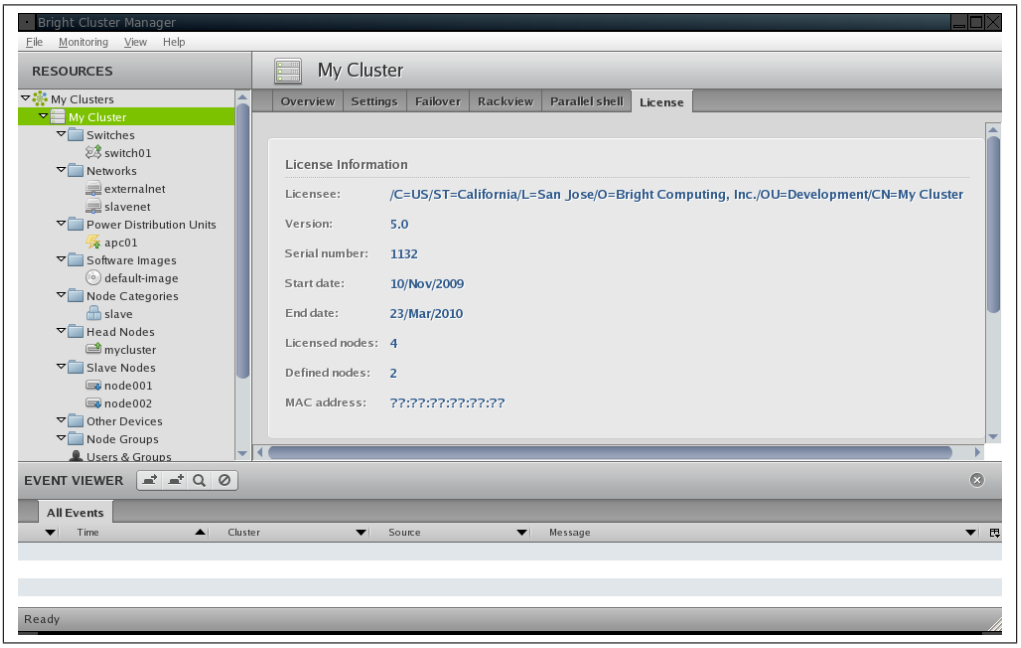

Figure 4.1: License Information

```
[root@mycluster ~]# cmsh
[mycluster]% main licenseinfo
License Information
---------------- ----------------------------------------------------
End Time         Tue Jul 23 23:59:59 2009
Licensed Nodes   4
Licensee         /C=US/ST=California/L=San Jose/O=Bright Computing,
                  Inc./OU=Development/CN=My Cluster
MAC Address      ??:??:??:??:??:??
Node Count       1
Serial Number    1249
Start Time       Mon Mar 23 00:00:00 2009
Version          5.0
[mycluster]%
```

The license in the example above can be considered a temporary license because it is valid for a short period of time and allows just 4 nodes to be used. Furthermore, the license is not tied to a specific MAC address, which means it can be used anywhere. For convenience, the `Node Count` field in the output of `licenseinfo` shows the current number of nodes being used.

4.1.2 Verifying a License

When an invalid license is being used, the cluster management daemon will fail to start. To confirm that there is a problem with the license, the cluster management daemon logfile should be consulted.

Example

```
[root@mycluster ~]# /etc/init.d/cmd start
Waiting for CMDaemon to start...
CMDaemon failed to start please see log file.
[root@mycluster ~]# tail -1 /var/log/cmdaemon
```

```
Dec 30 15:57:02 mycluster CMDaemon: Fatal: License has expired
```

To print license details and verify the license without having a running cluster management daemon, the `verify-license` utility may be used.

```
[root@mycluster ~]# verify-license
Usage: verify-license <path to certficate> <path to keyfile> <verify|info>
[root@mycluster ~]# cd /cm/local/apps/cmd/etc/
[root@mycluster etc]# verify-license cert.pem cert.key info
========= Certificate Information ========
Version:              5.0
Common name:          My Cluster
Organization:         Bright Computing
Organizational unit: Development
Locality:             San Jose
State:                California
Country:              US
Serial:               1249
Starting date:        23 Mar 2009
Expiration date:      23 Jul 2009
Allowed MAC address: ??:??:??:??:??:?? (good)
Licensed nodes:       4
==========================================
[root@mycluster etc]# verify-license cert.pem cert.key verify
License has expired
License verification failed.
```

By specifying the `verify` action, the license can be validated. If the license is valid, the verify-license utility will not produce any output and will exit with exit-code `0`. If the license is not valid, a message will be printed indicating why the license is not valid.

4.1.3 Requesting a License

It is important to verify that the license attributes are correct before proceeding with the configuration of a cluster. In particular, the license expiration date should be checked to make sure that the license being used is not a temporary license.

If the attributes of the license are correct, the remaining part of this section may be safely skipped. If a temporary license is being used or if the license attributes are not correct, a new license file must be requested. To request a new license file, the `request-cluster-certificate` script can be used.

Example

```
[root@mycluster ~]# module load cmd
[root@mycluster ~]# request-cluster-certificate
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
MAC Address for eth0 [00:0C:29:DA:CB:60]:
Filename to save private key to
```

```
  [/cm/local/apps/cmd/etc/cert.key.new]:
Filename to save certificate request to
  [/cm/local/apps/cmd/etc/cert.csr.new]:

Generated private key saved to
  /cm/local/apps/cmd/etc/cert.key.new
Please send the following certificate request to
  ca@brightcomputing.com:

-----BEGIN CERTIFICATE REQUEST-----
MIIB4jCCAUsCAQAwgaExCzAJBgNVBAYTAk5MMRYwFAYDVQQIEw1Ob3J0aCBIb2xs
YW5kMRIwEAYDVQQHEwlBbXN0ZXJkYW0xGTAXBgNVBAoTEENsdXN0ZXJWaXNpb24g
QlYxFDASBgNVBAsTC0RldmVsb3BtZW50MRMwEQYDVQQDEwpNeSBDbHVzdGVyMSAw
HgYJKoZIhvcNAQkCExEwMDowQzoyOTpEQTpDQjo2MDCBnzANBgkqhkiG9w0BAQEF
g/tpYO6HcJtOMbM7/BxHtPV+RpkEVpRbkls47cmQ/YFV9BajW+Xo0ism3KWIuUlp
AaAAMA0GCSqGSIb3DQEBBQUAA4GBAJiPPPIg5+oSpYIgYOB58unZQ497AvA8i7LU
AAOBjQAwgYkCgYEA2QdkOmP4bylSw2fvEONZxJhaVQum4dME7wFFCJ2ld4h7tUHI
5JihQFFVpEU7fYlyLy3Hh0VczgqMEBFqDSohC/sMMsdI1oDpriAN5Mx5DwECAwEA
J8f2uCMKMHuDnq1Tm+yOjjmq1e2UEGBOs0i0WKAhxzimyMsKOLv3qQ8O7qITB1ws
fQbT/oYyOaatJffJvofTsoz+LvaQXEfIxUaNTvvF8y6SO/35T6zhtFcEK4OnoBS/
huRV5yli
-----END CERTIFICATE REQUEST-----
[root@mycluster ~]#
```

The certificate request must be sent by e-mail to `ca@brightcomputing.com`. After the certificate request has been approved, a new certificate will be sent back by e-mail. The new certificate must then be copied to a temporary file on the cluster. To activate the new license, the `install-cluster-certificate` script in the `cmd` module can be used. This script takes the location of the new certificate temporary file as an argument and performs a number of steps to install the new license.

Example

Assuming the new certificate was saved as `/tmp/newcert.pem`:

```
[root@mycluster ~]# install-cluster-certificate /tmp/newcert.pem
Are you sure you want to change the license certificate? [y/N] y
Backup directory for certificates:
  /cm/node-installer/old-certificates/2009-03-24_19.06.42
Installed the new license!
Waiting for CMDaemon to terminate: OK!
Waiting for the CMDaemon to start again: OK!
```

After activating the license, it is a good idea to verify that the `licenseinfo` command in `cmsh` displays the correct license attribute values. Since the license is in fact the certificate that is used by the head node, any user or node certificates that were generated using the previous certificate will no longer be valid and must be regenerated. The `install-cluster-certificate` script has already taken care of renewing the administrator certificates for use with `cmsh` and `cmgui` (see section 3.3 for more information on certificate based authentication).

4.1.4 Licenses in a High Availability Setup

On a cluster with a High Availability setup (see chapter 10), the license file is present on both the primary as well as the secondary head node.

It is important that the MAC address field in the license holds both the MAC address for the primary head node, as well as the MAC address for the second head node. The two MAC addresses must be separated by a | character.

When using the `request-cluster-certificate` tool for requesting a new license file, the default value for the MAC address only includes the MAC address of the primary head node. It is up to the administrator to change this to also include the MAC address of the secondary head node:

Example

```
[root@mycluster ~]# module load cmd
[root@mycluster ~]# request-cluster-certificate
...
MAC Address for eth0 [00:0C:29:DA:CB:60]: 00:0C:29:DA:CB:60|00:0C:29:A3:C1:78
...
```

It is recommended that the correct license is installed on the primary head node before a high availability set-up is created. In situations where a new license needs to be installed on a cluster with a high availability in place, it is the administrator's responsibility to make sure that the new license is also installed on the secondary head node. This can be done by copying the following files from the primary head node to the secondary head node:

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

After the new license files have been installed on the secondary head node, the cluster management daemon must be restarted on that head node (see section 3.7.1 for more information).

4.2 Network Settings

After the cluster is set up with the correct license, the first step in the configuration process is to define the networks that are present. During the Bright Cluster Manager installation at least two networks were created:

- `slavenet`: the primary internal cluster network which is used for booting slave nodes and for all cluster management communication. In the absence of other internal networks, `slavenet` is also used for storage and communication between compute jobs.
- `externalnet`: the network which connects 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 information.

In the cluster management GUI, networks can be configured by selecting the `Networks` item in the resource tree (see figure 4.2). In the context of the OSI Reference Model, every network object represents a layer 3 (i.e. Network Layer) IP network. It should be noted that multiple layer

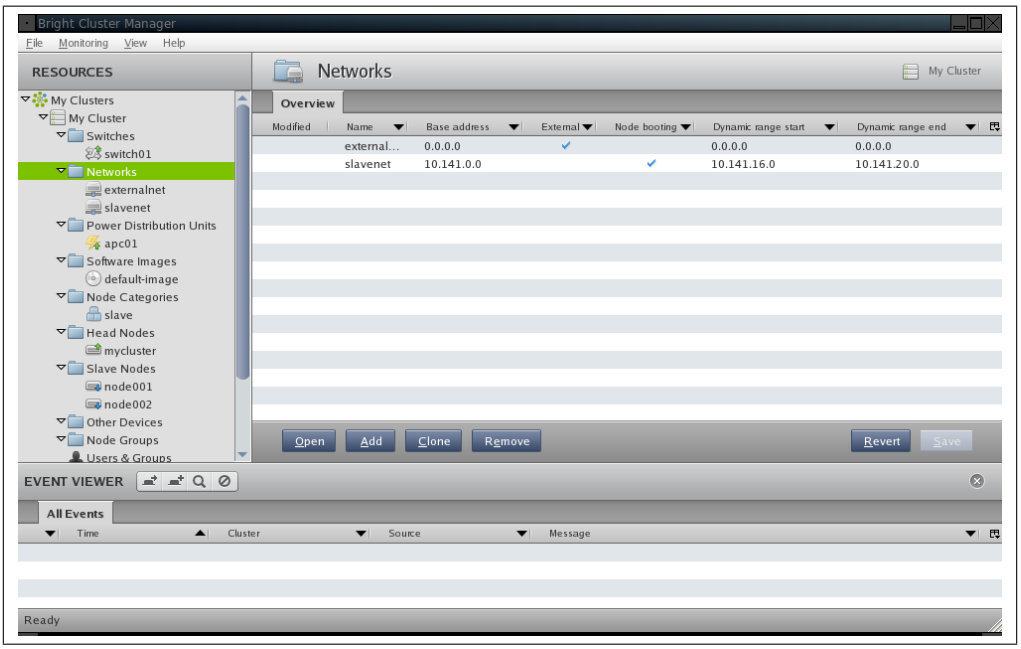

Figure 4.2: Networks

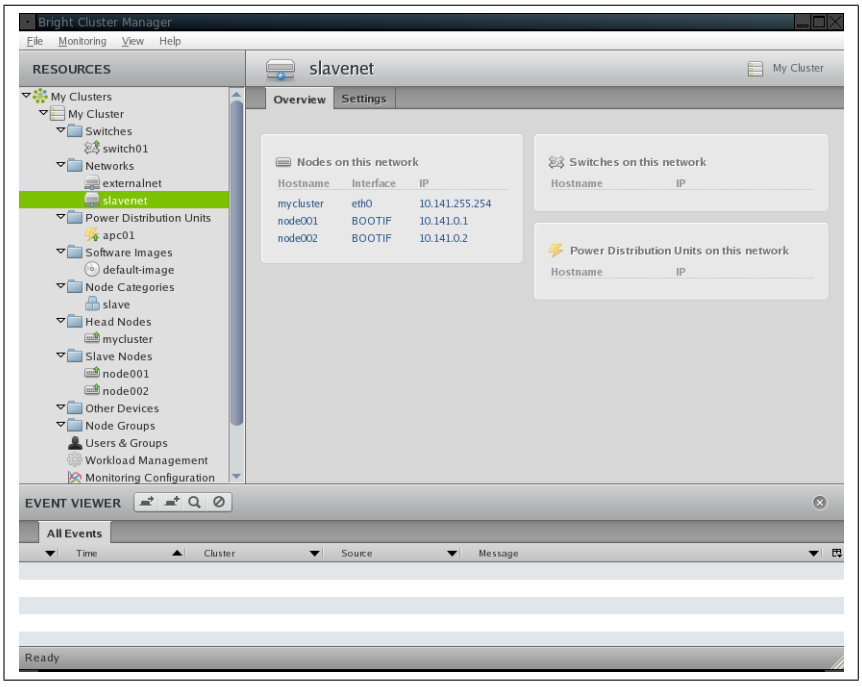

Figure 4.3: Network Overview

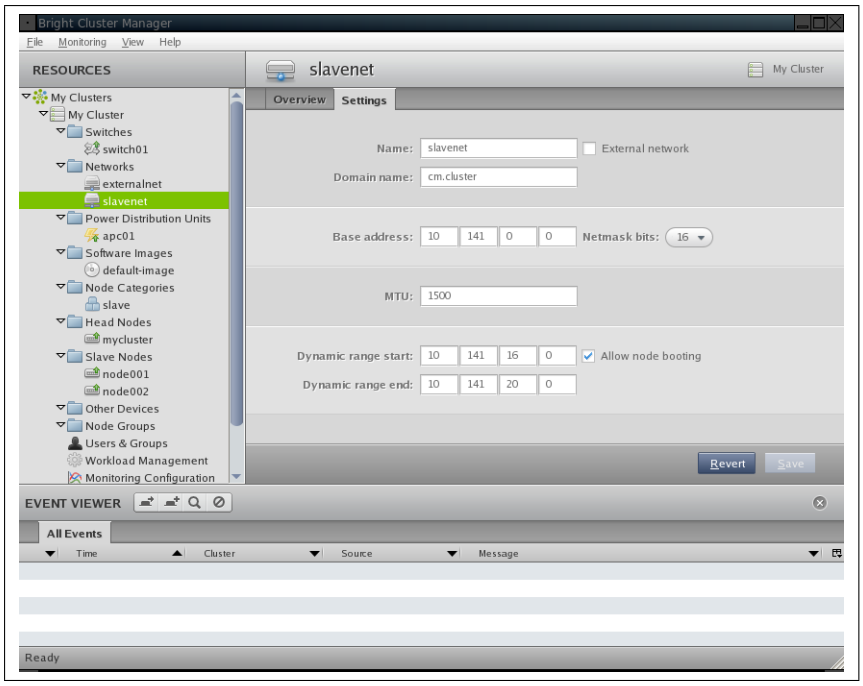

Figure 4.4: Network Settings

3 networks can be layered on a single layer 2 network (e.g. an Ethernet segment).

Selecting a network in the resource tree provides the user with a convenient overview of all IP addresses that have been assigned in the selected network (see Figure 4.3). In the `Settings` tab (see Figure 4.4), a number of properties can be changed.

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.
`Netmask bits`	Netmask in CIDR notation.

A network can be regarded as a range of IP addresses. The first address in the range is the *base address*. The length of the range is determined by the *netmask*, for which CIDR notation is used. For convenience, figure 4.5 lists a translation table between a number of traditional netmasks and CIDR notation. The `sipcalc` utility which is installed on the head node, is a useful tool for calculating IP subnets (see `man sipcalc` for more information).

Every network has an associated DNS domain which can be used to access a device through a particular network. The default DNS domain for the primary internal network is `cm.cluster`, which means that `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

Traditional Netmask	CIDR notation
255.255.0.0	16
255.255.128.0	17
255.255.192.0	18
255.255.224.0	19
255.255.240.0	20
255.255.248.0	21
255.255.252.0	22
255.255.254.0	23
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30
255.255.255.254	31
255.255.255.255	32

Figure 4.5: Traditional netmasks and CIDR notation counterparts

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 (see figure 4.2) can be used to add a new network. After the new network has been added, the `Settings` tab (see 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.

By default the network `slavenet` is configured as the primary internal network and the network `externalnet` is configured as the external network. To change this, the `Settings` tab of the cluster object (see Figure 4.6) lists two properties `Primary network` and `External network`. Setting these properties allows one to define respectively the primary internal network and the external network.

4.2.3 Configuring External Network Parameters

After both internal and external networks have been defined, it may be necessary to change how the cluster interfaces with the outside world. The initial settings were configured during installation but at some point these settings may need to be altered.

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

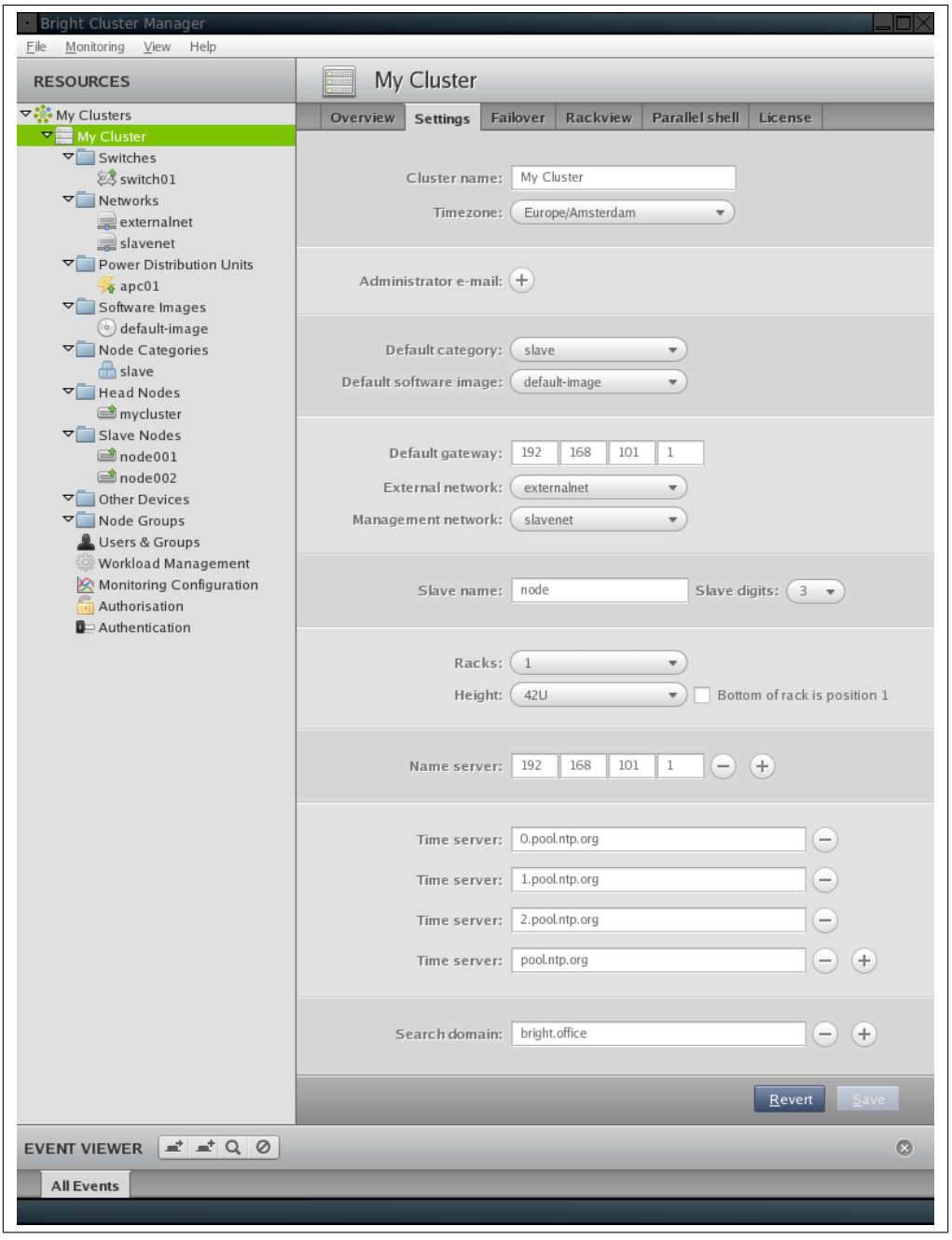

Figure 4.6: Cluster Settings

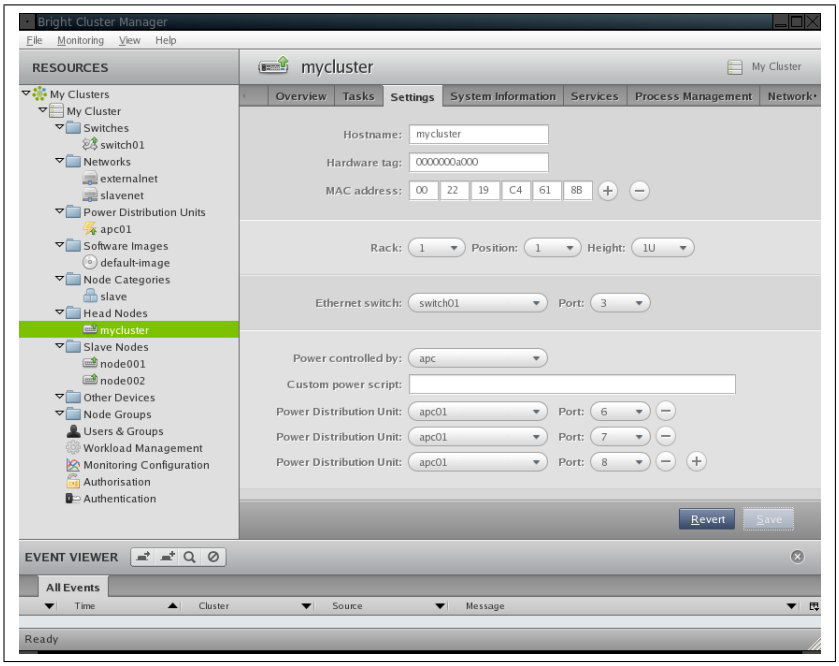

Figure 4.7: Head Node Settings

used at all times. Setting the hostname of the head node 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 the GUI, select the device listed under `Head Nodes` and subsequently select the `Settings` tab (see figure 4.7). Changing the hostname is done by modifying the `Hostname` property and pressing `Save`. When setting a hostname, do not include a domain.

Example

Changing the hostname of the head node through `cmsh`:

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

Note: the hostname in the shell prompt is only set upon login, which is why in the example it is still displayed as `mycluster`.

Changing External Network Parameters

Several entities inside the cluster management infrastructure are involved when it comes to how a cluster interacts with an external network (e.g. a company or university network). The networks defined as *external networks* need to be configured correctly. Then, the outward facing devices inside a cluster (e.g. head nodes, login nodes) need to be assigned IP addresses on these external networks. Changing external IP parameters of a cluster therefore involves making changes in three places.

1. To change IP range (e.g. base address, netmask), the `Settings` tab (figure 4.4) of the external network can be used. In addition the IP addresses of all outward facing devices must be changed through the `Network Setup` tabs of the corresponding devices.
2. The primary external IP address for a cluster (usually the IP address of the head node or login node) should be changed in the Settings tab of the cluster object (see figure 4.6).
3. Default gateway, external DNS name servers, DNS search domains and NTP time servers should also be configured through the Settings tab of the cluster object (see Figure 4.6).

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

Example

Changing external network parameters of the cluster through `cmsh`:

```
[mc]% network use externalnet
[mc->network[externalnet]]% set baseaddress 192.168.1.0
[mc->network[externalnet]]% set netmaskbits 24
[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 defaultgateway 192.168.1.1
[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]]%
```

To configure the cluster to use DHCP to obtain its external network settings, `baseaddress` for the `externalnet` and the IP address of the external address should be set to `0.0.0.0`. Name-server, time-server and default gateway settings are not inherited from DHCP, and must be set correctly manually.

4.3 Configuring IPMI Interfaces

Bright Cluster Manager also takes care of the initialisation and configuration of the baseboard management controller (BMC) that may be present on devices. The IPMI 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.

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 slave node device and then to clone that device several times.

Example

Alternatively, one may write a simple `bash` loop:

```
[root@mc ~]# (for i in `seq 1 150`;
> 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
```

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]->networkinterfaces]% add alias eth0:0
[mc->device[mc]->networkinterfaces[eth0:0]]% set network ipminet
[mc->device[mc]->networkinterfaces[eth0:0]]% set ip 10.148.255.254
[mc->device[mc]->networkinterfaces[eth0:0]]% commit
```

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 initialisation and configuration of the IPMI interface of a device. In addition to a number network-related settings, the node-installer also configures IPMI authentication credentials. By default IPMI interfaces are configured with username `ADMIN` and password `ADMIN`. To prevent unauthorized access to the IPMI interface of a device, it is recommended that the default username and/or password are changed.

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

The first step of configuring the Infiniband interconnect is to install a number of software packages on the head node and inside the software images. Bright Cluster Manager includes a script `install-ib-packages` which takes care of installing the required packages. The `install-ib-packages` should be executed once without an argument on the head node. The `install-ib-packages` should subsequently be called once for each software image that will be used on slave nodes with an Infiniband HCA. This is done by passing the full path to the software image as an argument.

Example

Installing software packages required for Infiniband on the head node and in a software image:

```
[root@mycluster ~]# module load cluster-tools
[root@mycluster ~]# install-ib-packages
...
[root@mycluster ~]# install-ib-packages /cm/images/default-image
...
```

The `install-ib-packages` will make calls to `yum` to download and install the relevant software packages. Packages include Infiniband drivers, several programming interface libraries, and management and diagnostic tools.

The `install-ib-packages` script also takes care of scheduling the Infiniband drivers to be started at system boot time. If this is not desired, the `chkconfig` command may be used on the `openibd` service to disable automatic loading of the Infiniband drivers.

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 equiped 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 slave nodes:

```
[mc]% device services node001
[mc->device[node001]->services]%
[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 slave node device and then to clone that device several times.

Example

Alternatively, one may write a simple `bash` loop:

```
[root@mc ~]# (for i in ‘seq 1 150‘;
> do echo device interfaces node‘printf "%03d" $i‘\;;
>    echo add physical ib0\;;
>    echo set network ibnet\;;
>    echo set ip 10.149.0.\$i\;;
>    echo commit;
> done;) | cmsh
```

When the head node is also equiped 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]->networkinterfaces]% add physical ib0
[mc->device[mc]->networkinterfaces[ib0]]% set network ibnet
[mc->device[mc]->networkinterfaces[ib0]]% set ip 10.149.255.254
[mc->device[mc]->networkinterfaces[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
[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

Network switches and PDUs that will be used as part of the cluster should be configured using the configuration interface that is described in the switch administration documentation. It is important that the switches and PDUs are configured with the correct IP settings. The IP configurations in the switch must match the IP settings that have been configured for the corresponding switch or PDU device in the cluster management software.

Moreover, in order to allow the cluster management software to communicate with the switch or PDU, 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
```

cd p

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.

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

In the case of PDU based power control, the power supply of a device is plugged in to a port on a PDU device. The device can then be turned on or off by changing the state of the PDU port. IPMI based power control relies on the baseboard management controller (BMC) inside a node. It is therefore only available for node devices. Blades inside of a blade chassis typically use IPMI for power management. For more information on IPMI and how to enable IPMI interfaces, see section 4.3.

When PDU based power control is used, the PDU must be added as a device first and must be reachable over the network. The `Settings` tab of a device object may subsequently be used to configure the PDU ports that may be used to control the device. Figure 5.1 shows the `Settings` tab for a head node. Using the + and - buttons, PDU ports can be added and removed. To enable power control through APC PDUs, the `Power controlled by` property should be set to `apc`.

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 performed 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

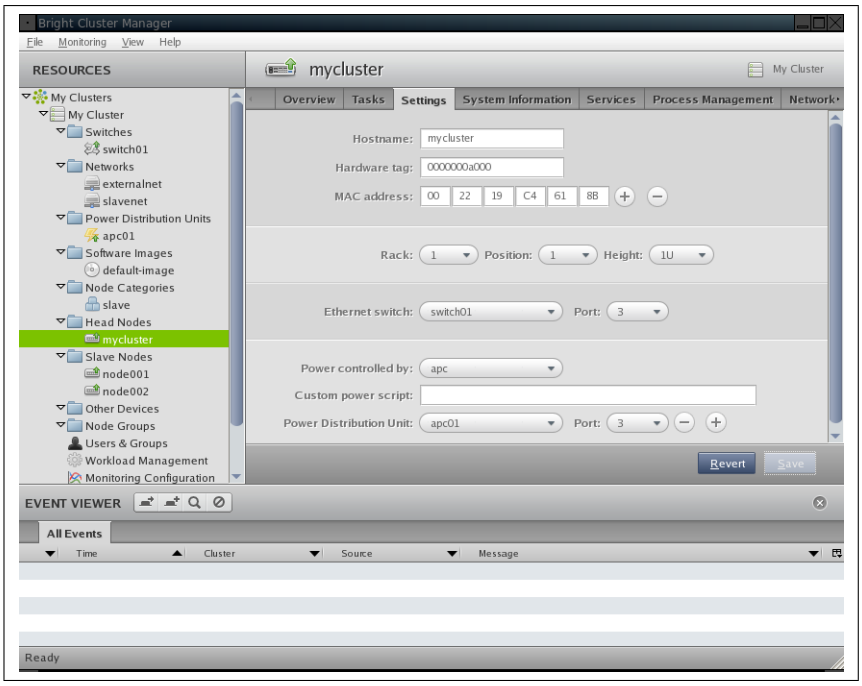

Figure 5.1: Head Node Settings

sharing a single power supply). In this case, all power operations on one device will apply to all nodes sharing the same PDU port.

5.1.2 IPMI Based Power Control

When IPMI based power control is desired, the `Power controlled by` property should be set to the IPMI interface through which power operations should be relayed. Normally this IPMI interface is `ipmi0`. When no PDUs are used, the list of configured PDU ports is ignored. IPMI based power control is available only for node devices.

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

Example

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

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

5.1.3 Combining PDU- and IPMI Based Power Control

By default when nodes are configured for IPMI Based Power Control, any configured PDU ports are ignored. However, in some situations it

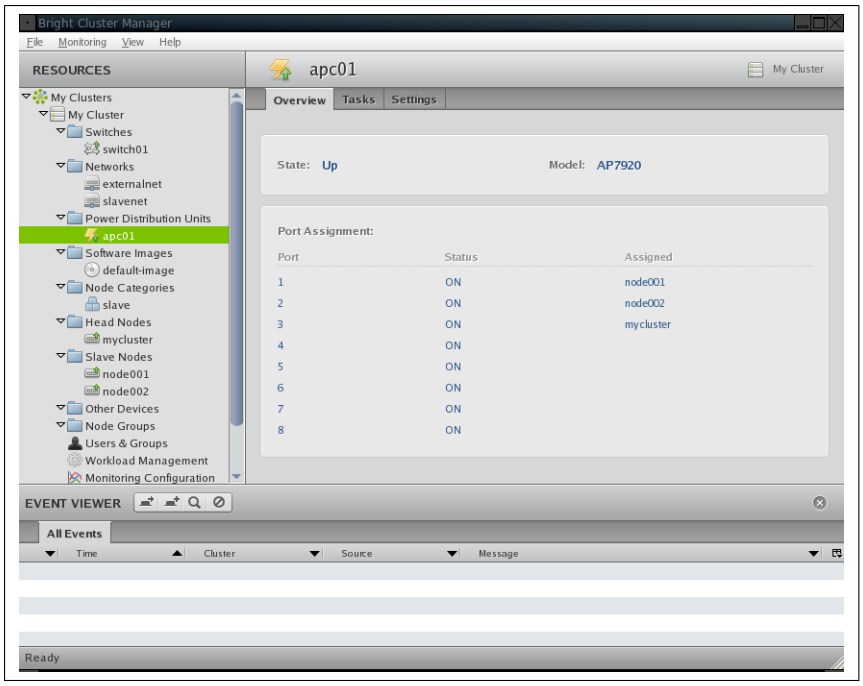

Figure 5.2: PDU Overview

can be useful to be able to change this behavior. The `PowerOffPDUOutlet` configuration file directive of the `cmdaemon` on the head node may be used to to instruct the cluster management infrastructure to power off the PDU ports after an IPMI based power off has been issued. This behavior can be used to shut down the BMC when nodes are powered off, which can be useful for power savings. When nodes are subsequently powered on again, the PDU ports are first powered on, after which an IPMI based power on operation is issued. When multiple nodes share the same PDU port, the PDU port is only powered off when all nodes served by that particular PDU port have been powered off.

By default the `PowerOffPDUOutlet` value is `false`. For more information on changing `cmdaemon` configuration file directives, see section 3.7.

5.1.4 Custom Power Control

For devices which cannot be controlled through any of the supported methods of power control, it is possible to set a custom power management script. Such a script will be invoked by the cluster management daemon on the head node whenever a power operation for a device is performed. To set a custom power management script for a device, one must set the `powercontrol` attribute to `custom` using either the cluster management GUI or `cmsh`, and specify a `custompowerscript`. The value for `custompowerscript` must be an executable script on the head node(s) of a cluster.

A custom power management script must will be invoked as follows:

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

where `device` is the name of the device on which a power operation is to be performed, and `operation` is one of the following:

- ON

- OFF
- RESET
- STATUS

which correspond to the power operations described in the next section. On success a custom power script should exit with exit-code 0. On failure, a custom power script should exit with a non-zero exit-code.

5.1.5 HP iLO Based Power Control

For HP nodes that are equiped with an iLO or iLO2 management interface, a number of steps must be performed to allow power management operations to be issued:

1. Set up `ipmi0` interfaces for all slave nodes (see section 5.1.2). Bright Cluster Manager treats HP iLO interfaces the same way as regular IPMI interfaces.
2. Download the `hponcfg` RPM from the HP web-site. This can usually be found by searching for "HP Lights-Out Online Configuration Utility for Linux". At time of writing, this RPM was available at the following location:
   `ftp://ftp.hp.com/pub/softlib2/software1/pubsw-linux/p1980791501/v54498/hponcfg-2.2.0-5.noarch.rpm`
3. Install the `hponcfg` on the head node, in the slave images and in the node-installer tree.

   Example

   ```
   rpm -i hponcfg-2.2.0-5.noarch.rpm
   rpm -i --root=/cm/images/default-image hponcfg-2.2.0-5.noarch.rpm
   rpm -i --root=/cm/node-installer hponcfg-2.2.0-5.noarch.rpm
   ```

4. Configure the `ilo_power.pl` custom power script for all slave nodes.

   Example

   To use `cmsh` to configure the `ilo_power.pl` custom power script for all slave nodes in the `slave` category:

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

5.2 Power Operations

Power operations may be performed on devices from either cluster management GUI 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

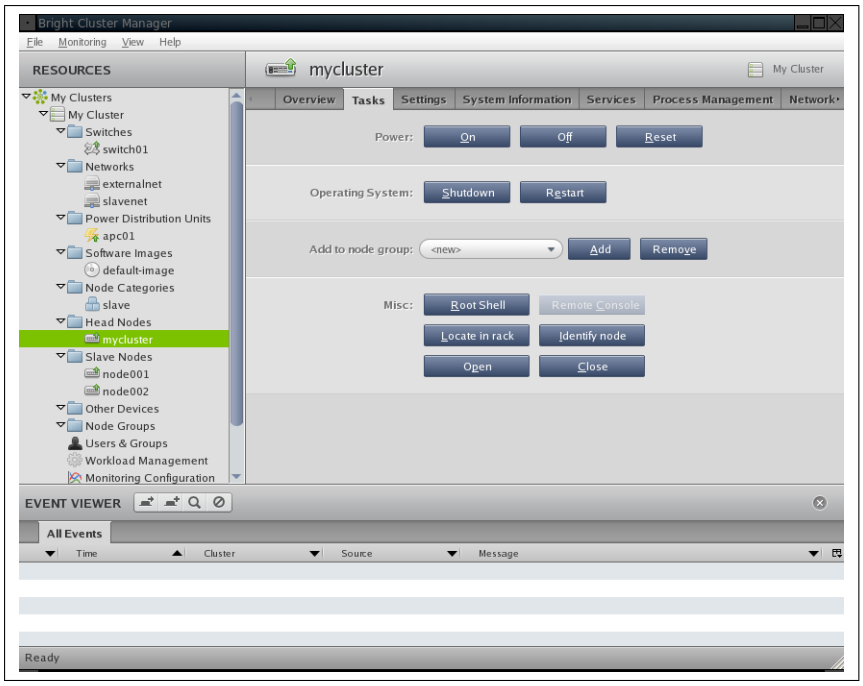

Figure 5.3: Head Node Tasks

5.2.1 Through GUI

In the cluster management GUI buttons for performing 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 the power status information for a device. Figure 5.4 shows the `Overview` tab of a head node. The green LEDs indicate that power on all three PDU ports is turned on. Power ports that have been turned off are visualised using red LEDs. Gray LEDs are used when the power status for a device is unknown.

Performing power operations on multiple devices at once is possible through the `Tasks` tabs of node categories and node groups. In addition it is also possible to perform power operations on ad-hoc groups through the `Slave Nodes` folder in the resource tree.

In the `Overview` tab the members of the ad-hoc group can be selected. The operation that is to be performed on the ad-hoc group defined by the selection in the `Overview` tab, can subsequently be invoked through the `Tasks` tab.

When performing a power operation on multiple devices, the power operation is performed on the devices in sequence with a 1 second delay between every 2 successive devices. This is done to prevent electrical current surges on the power infrastructure. Using `cmsh`, custom delays may be specified when performing power operations.

Through the `Overview` tab of a PDU object (see figure 5.5), it is also possible to perform power operations operations on PDU ports directly. This can be useful in situations where all ports on a particular PDU have to be powered on/off, or if temporary equipment that has been plugged into a particular PDU port needs to be powered on/off.

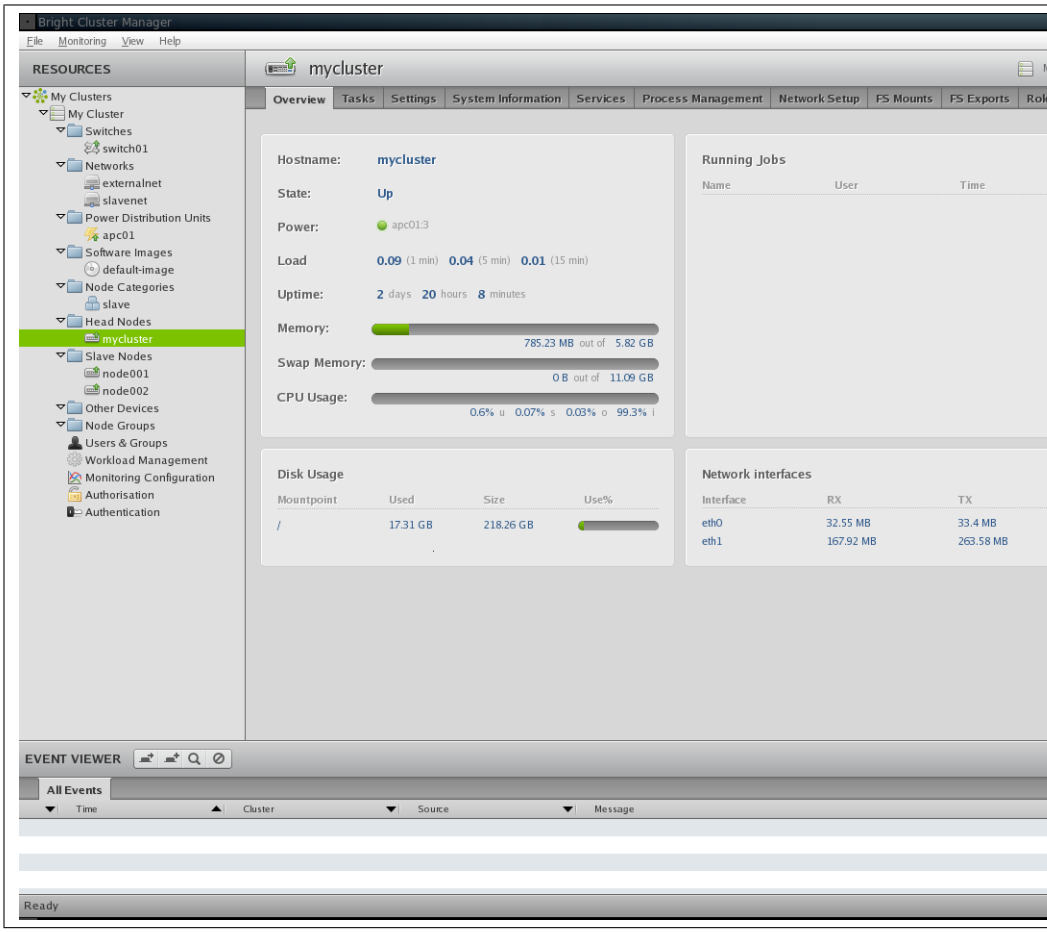

Figure 5.4: Head Node Overview

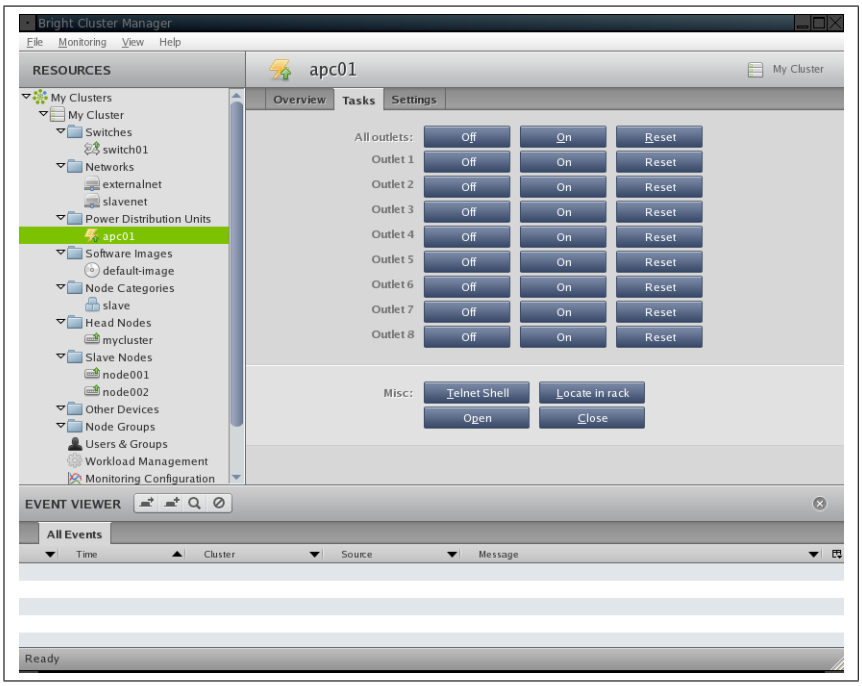

Figure 5.5: PDU Tasks

5.2.2 Through cmsh

All power operations in cmsh are performed through the power command in device mode. Figure 5.6 shows usage information for the power command.

Example

Powering on node001 and node018:

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

Example

Powering off all nodes in the slave category with a 100ms delay between nodes:

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

Example

Retrieving power status information for a group of nodes:

```
[mycluster]% device power -g mygroup status
apc01:3 ............. [   ON    ] node003
apc01:4 ............. [   OFF   ] node004
```

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 component of Bright Cluster Manager collects power related data from power distribution units. The following power related metrics are available:

- PDUBankLoad: Phase load (in Ampere) for one (specified) bank in a PDU
- PDULoad: Total phase load (in Ampere) for one PDU

```
Usage:
power status .................. Retrieve power status of the current device or all
                                if none is selected
power off ..................... Turn power on for current device
power on ...................... Turn power off for current device
power reset ................... Reset power for current device
power <-n|--nodes nodelist> [-b|--background] status
      Retrieve power status of selected devices
power <-g|--group node-groupname> [-b|--background] status
      Retrieve power status of devices inside the nodegroup
power <-p|--powerdistributionunitport powerdistributionunit> [-b|--background] status
      Retrieve power status for a powerdistributionunit
power <-c|--category category> status
      Retrieve power status of nodes in the specified category
power <-n|--nodes nodelist> [-b|--background] [-d|--delay secs] on
      Turn power on for list of devices using the given delay or default(seconds)
power <-g|--group node-groupname> [-b|--background] [-d|--delay secs] on
      Turn power on for all nodes in the group using the given delay or
      default(seconds)
power <-p|--powerdistributionunitport powerdistributionunit> [-b|--background]
      [-d|--delay secs] on
      Turn power on for a specific powerdistributionunit outlet using the given delay
      or default(seconds)
power <-c|--category category> [-b|--background] [-d|--delay secs] on
      Turn power on for nodes in the specified category using the given delay
      or default(seconds)
power <-n|--nodes nodelist> [-b|--background] [-d|--delay secs] off
      Turn power off for list of devices using the given delay or default(seconds)
power <-g|--group node-groupname> [-b|--background] [-d|--delay secs] off
     Turn power off for all nodes in the given nodegroup using the given delay
      or default(seconds)
power <-p|--powerdistributionunitport powerdistributionunit> [-b|--background]
      [-d|--delay secs] off
      Turn power off for a specific powerdistributionunit outlet using the given
      delay or default(seconds)
power <-c|--category category> [-b|--background] [-d|--delay secs] off
      Turn power off for nodes in the specified category using the given delay
      or default(seconds)
power <-n|--nodes nodelist> [-b|--background] [-d|--delay secs] reset
      Reset power for list of devices using the given delay or default(seconds)
power <-g|--group node-groupname> [-b|--background] [-d|--delay secs] reset
      Reset power for all nodes in the group using the given delay or default(seconds)
power <-p|--powerdistributionunitport powerdistributionunit> [-b|--background]
      [-d|--delay secs] reset
      Reset power for a specific powerdistributionunit outlet using the given delay
      or default(seconds)
power <-c|--category category> [-b|--background] [-d|--delay secs] reset
      Reset power for nodes in the specified category using the given delay
      or default(seconds)
nodelist ...................... e.g. node001..node015,node20..node028,node030
powerdistributionunit ......... e.g. apc01:* or apc01:8,apc01:5
```

Figure 5.6: Usage information for the `device power` command in `cmsh`

6

Node Provisioning

In a cluster running Bright Cluster Manager most slave nodes will be configured to boot from the network. On every boot up, the node's local hard disk will be checked and its contents will be updated. This ensures that after every reboot, a slave node is always restored to a *known good state*. This known good state is defined in a *software image*, which can be maintained from the head node. The process of getting the software image onto the slave node hard disks is called *provisioning*. Several elements are involved in the provisioning system, each of which will be described in the following sections.

6.1 Provisioning Nodes

Each cluster can have one or more *provisioning nodes*. Provisioning nodes are capable of sending a software image to a slave node. In simple clusters only the head node is a provisioning node. In more complex clusters there can be several provisioning nodes. This allows the network traffic to be distributed when many slave nodes are booting. Creating provisioning nodes is done by assigning the *provisioning role* to a node or category of nodes. The provisioning role has several parameters that can be set:

Property	Description
`allImage`	When set, the provisioning node will provide all available images regardless of the images property.
`images`	A list of images provided by the provisioning node. This property is only used when allImages is not set.
`maxProvisioningNodes`	The maximum number of nodes that can be provisioned in parallel. This property can be used to prevent the provisioning nodes network and disk from being overloaded.

The following `cmsh` example shows how to set up a category called `storage` with a provisioning role providing only the default-image and allowing a maximum of 20 nodes to be provisioned in parallel.

Note that a provisioning node will keep a copy of all images it provides on its local disk. The images will be stored in the same directory

as where the images are stored on the head node. Therefore the provisioning nodes should provide enough local file system space to store the relevant images. This may require changes to the nodes' disk layout.

Example

```
[mycluster]% category use storage
[mycluster->category[storage]]% roles
[mycluster->category[storage]->roles]% assign provisioning
[mycluster->category[storage]->roles[provisioning]]% set allimages false
[mycluster->category[storage]->roles[provisioning]]% set images default-image
[mycluster->category[storage]->roles[provisioning]]% show
Parameter                Value
------------------------ ------------------------------------------------
Name                     provisioning
Type                     ProvisioningRole
allImages                no
images                   default-image
maxProvisioningNodes     10
[mycluster->category[storage]->roles[provisioning]]% set maxprovisioningnodes 20
[mycluster->category[storage]->roles[provisioning]]% show
Parameter                Value
------------------------ ------------------------------------------------
Name                     provisioning
Type                     ProvisioningRole
allImages                no
images                   default-image
maxProvisioningNodes     20
[mycluster->category[storage]->roles[provisioning]]% commit
[mycluster->category[storage]->roles[provisioning]]%
```

The head node performs all housekeeping tasks for the entire provisioning system. When it receives a provisioning request from a slave node it will select a provisioning node to perform the actual provisioning. When multiple provisioning nodes are able to provide the requested software image, the provisioning node with the lowest number of already started provisioning tasks will be selected. The head node will also limit the total number of running provisioning tasks. This limit is defined using the `MaxNumberOfProvisioningThreads` setting in the head node CMDaemon configuration file (see Appendix C). The provisioning request will be deferred if the head node is not able to select a provisioning node. As soon as an ongoing provisioning task has finished, the head node will try to handle any deferred requests.

The head node needs to be notified whenever changes are made to the provisioning role setup of any categories or nodes. This can be done by using the `updateprovisioners` command from the `softwareimage` mode in the `cmsh` (see example). Whenever this command is invoked, the provisioning system will wait for all running provisioning tasks to end and will subsequently reinitialize its internal state with all provisioning role properties. Also it will update all images located on any provisioning nodes.

Example

```
[mycluster->softwareimage]% updateprovisioners
```

```
Provisioning nodes will be updated in the background.
[mycluster->softwareimage]% Updating image default-image on provisioning node storage01.
Updating image default-image on provisioning node storage01 completed.
Provisioning node storage01 was updated.
Updating image default-image on provisioning node storage02.
Updating image default-image on provisioning node storage02 completed.
Provisioning node storage02 was updated.
All provisioning nodes were updated.
[mycluster->softwareimage]%
```

6.2 Software Images

Software images contain a complete Linux file system to be installed on a slave node. See Chapter 7 for more details. The head node holds the main copy of a software image. Whenever a change has been made to the files inside an image, the changes need to be sent to all provisioning nodes. This can be done by running the `updateprovisioners` command from the `softwareimage` mode in the `cmsh`. Whenever this command is invoked, the provisioning system will wait for all running provisioning tasks to end, reinitialize it's internal state and update all images located on any provisioning nodes.

6.2.1 Kernel Drivers

Each software image will contain a Linux kernel and a ramdisk. The ramdisk is one of the first items that is loaded when a node boots from the network. It should contain drivers which enable the node's network card and local storage. To configure what drivers are loaded from the ramdisk, the `kernelmodules` submode of the `softwareimage` mode in the `cmsh` can be used. Whenever a change is made to the kernel module setup of a software image, the CMDaemon will automatically regenerate the ramdisk inside the image and send the updated image to all provisioning nodes. The same happens when a user forces recreation of the ramdisk using the `createramdisk` command.

6.3 Node Installer

When a slave node boots from the network it will load the kernel and a ramdisk. The ramdisk then loads all configured kernel modules and tries to launch the node installer. The node installer interacts with the CMDaemon on the head node and takes care of the rest of the boot process. Once the node installer has completed, the node's local hard disk will contain a complete Linux system. The node installer will then call `/sbin/init` from the local hard disk and the Linux boot process will proceed as a normal Linux boot. The main steps the node installer performs are listed below and described in detail in the following sections.

1. Request a certificate if needed.
2. Find the node's configuration.
3. Start all network interfaces.
4. Process the install-mode.
5. Check the local hard disk and re-partition if needed.

6. Synchronize the local hard disk with the correct software image.
7. Write network configuration files to the local drive.
8. Create an `/etc/fstab` file on the local drive.
9. Install grub bootloader if configured.
10. Run finalise scripts.
11. Unload specific drivers no longer needed.
12. Switch to local hard disk and call `/sbin/init`.

6.3.1 Node Certificate Management

Each node needs a certificate to be able to communicate with the CMDaemon on the head node. Checking if such a certificate is available is one of the first tasks the node installer performs. In case it can not find a certificate it will automatically request one and wait for the certificate to be issued by the head node CMDaemon. Figure 6.1 is a screen shot of a node installer waiting for its certificate to be issued.

When there is no risk that the cluster can be accessed from an untrusted network, the easiest way to allow a certificate to be issued is to enable certificate auto-signing. This can be done by issueing the `autosign on` command from `cert` mode in the `cmsh`. Enabling certificate auto-signing will allow the cluster management daemon to automatically issue a certificate to any node that requests a certificate. For safety reasons, certificate auto-signing is automatically turned off when the cluster management on the head node is restarted.

When certificate auto-signing is not possible (e.g. because the local network is not trusted), certificate requests will have to be approved manually.

Example

Enabling certificate auto-sign mode:

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

Once the node installer has received a valid certificate it will store 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. Note that the node installer will not request a new certificate if it finds a certificate in this directory even if the certificate may no longer be valid. When an invalid certificate is present this will result in a communication error. If this occurs the node's certificate directory should be removed to allow the node installer to request a new certificate.

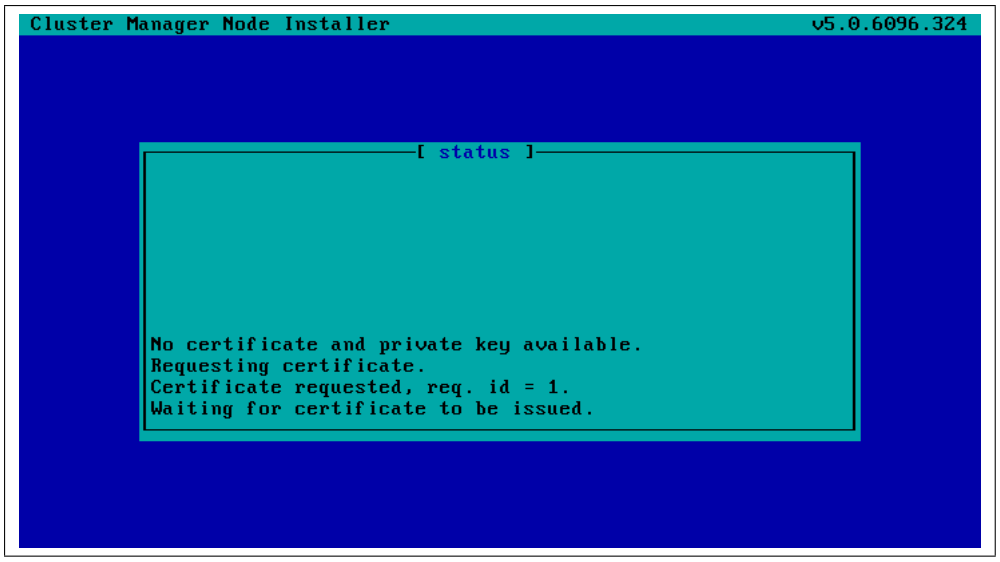

Figure 6.1: Certificate Request

6.3.2 Node Configuration Selection

Once communication with the head node CMDaemon is established, the node installer must determine on which node it is running. It needs this information to fetch the correct settings. To find the correct node configuration it will query the CMDaemon for a node configuration that is associated with the MAC address of the Ethernet interface used for booting the node. If the query matches a node configuration then the slave node has been booted before and is considered known. If no configuration is found the slave node is considered to be new. In either case the node installer now asks the CMDaemon to which Ethernet switch port the node is currently connected. For more details on how to set up Ethernet switches for port detection, consult section 4.5. If a port is found, the node installer will query the CMDaemon for a node configuration associated with the detected Ethernet switch port. There are now several scenarios:

1. The node is new and a configuration associated with the detected Ethernet switch port was found. The node installer will assume the configuration found using port detection should be used and displays a confirmation dialog. See figure 6.2.

2. The node is new and no configuration associated with the detected Ethernet switch port was found. The node installer will display a dialog that allows the administrator to either retry the Ethernet switch port detection or manually select a node configuration. See figure 6.3 and 6.5.

3. The node is new and no Ethernet switch port was detected. The node installer will display a dialog that allows the user to either retry the Ethernet switch port detection or manually select a node configuration. See figure 6.4 and 6.5.

4. The node is known and the configuration associated with the detected Ethernet port is the same as the configuration associated with

the node's MAC address. The node installer displays a confirmation dialog. See figure 6.2.

5. The node is known and the configuration associated with the detected Ethernet port is not the same as the configuration associated with the node's MAC address. There is a port mismatch and, to prevent configuration mistakes, the node installer displays a dialog allowing the user to retry, manually select another node configuration or accept the node configuration that is associated with the detected Ethernet port. See figure 6.6 and 6.5.

6. The node is known and the configuration associated with the node's MAC address also has an Ethernet port associated with it, but no Ethernet switch port was detected. This is also considered a port mismatch and, to prevent configuration mistakes, the node installer displays a dialog similar to the dialog in figure 6.6.

7. The node is known and the configuration associated with the node's MAC address is not associated with any Ethernet switch port, but an Ethernet switch port was detected. This is also considered a port mismatch and, to prevent configuration mistakes, the node installer displays a dialog similar to the dialog in figure 6.6.

Note that whenever the user decides to manually select a node configuration, the process of attempting to detect an Ethernet switch port is repeated. If a port mismatch occurs it is handled the same way as it would have been handled when the user did not manually select a configuration.

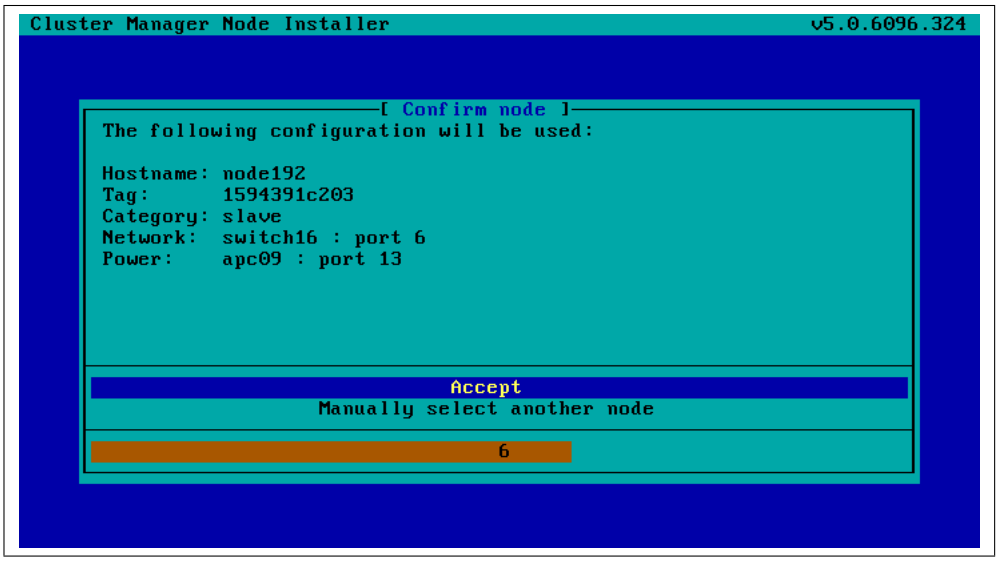

Figure 6.2: Node Configuration Confirm

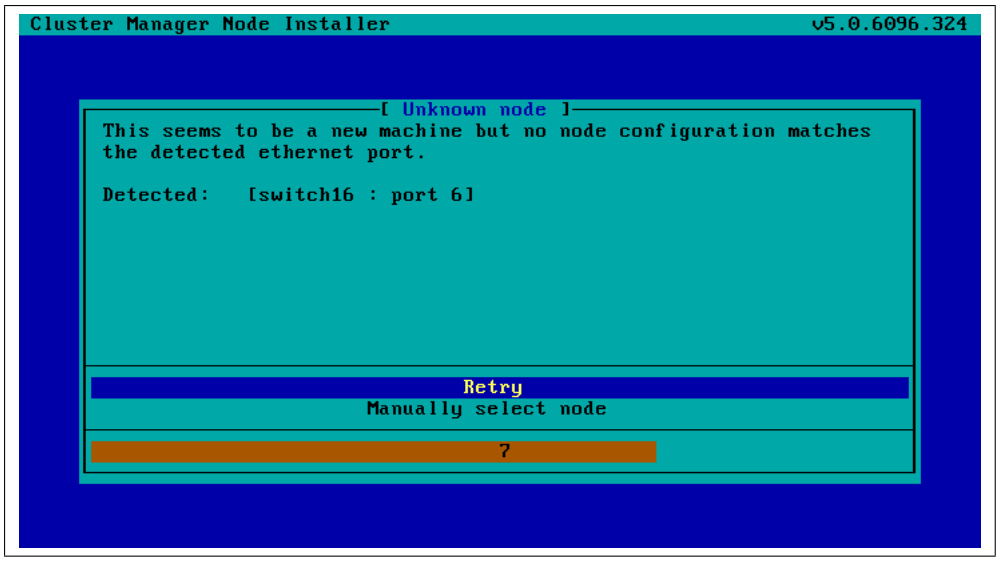

Figure 6.3: Unknown Node

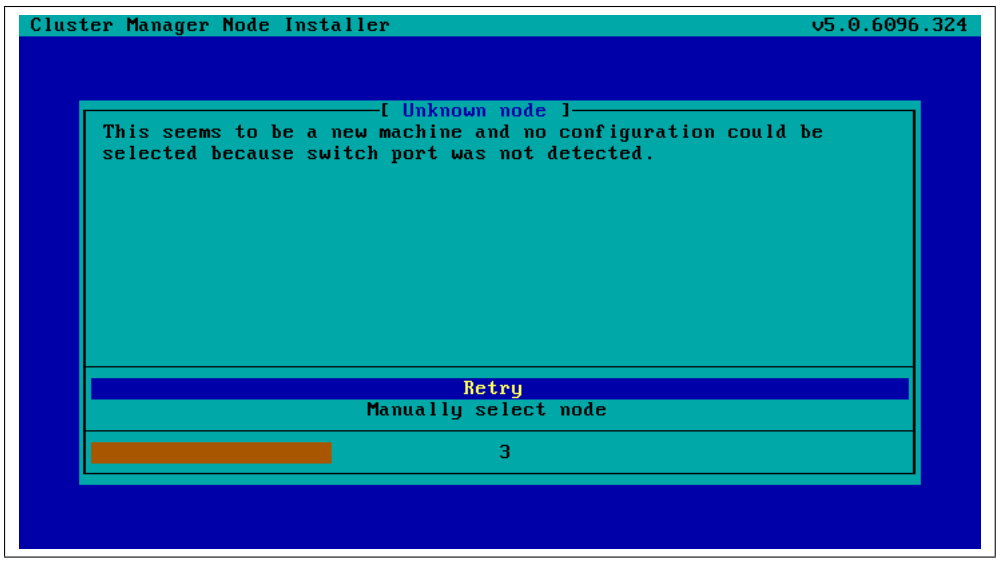

Figure 6.4: Unknown Node

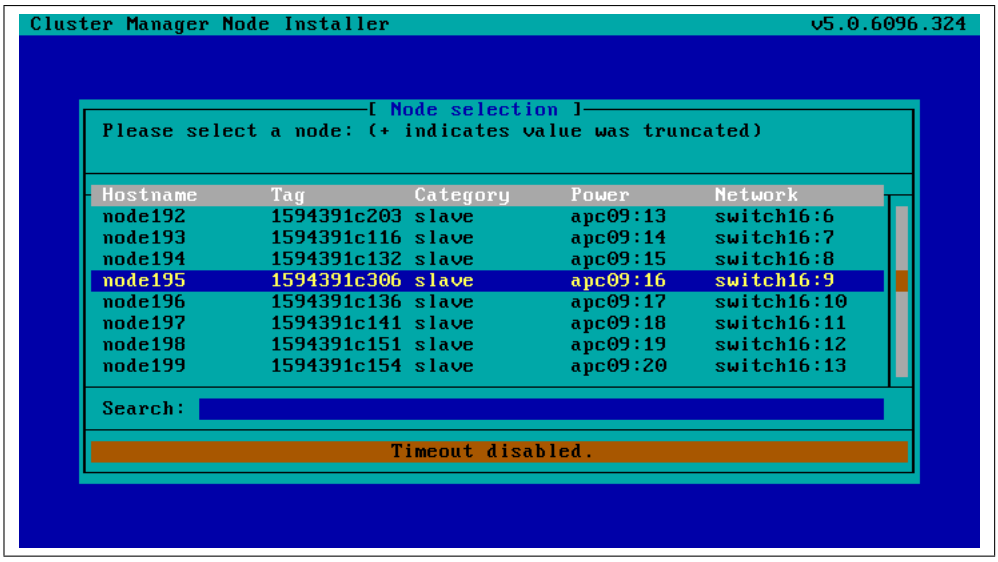

Figure 6.5: Node Selection

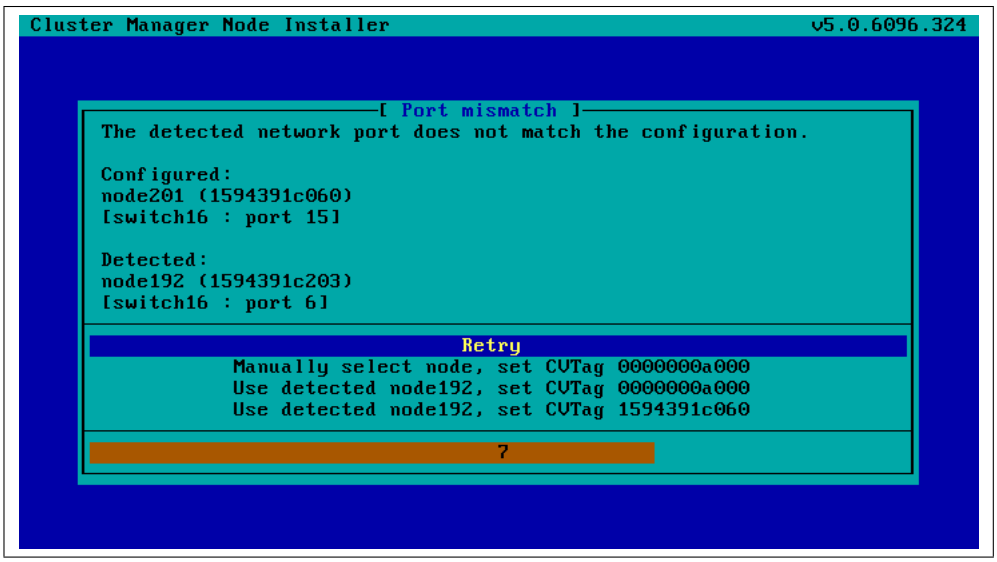

Figure 6.6: Port Mismatch

6.3.3 Network Interfaces

Now that the node installer knows on which node it is running, it will continue and bring up all network interfaces configured for the node. Before starting each interface, the node installer will check if the configured IP address is not already in use by something else. If so, it will display a warning dialog and enter a retry loop. It will not continue until the IP conflict is resolved.

One of the interfaces can have the special name `BOOTIF`. The node installer will automatically translate this into the name of the device used for network booting. This can be useful if a machine has multiple network links of which only one is physically attached to the network. When connecting the hardware it is often unclear which interface will show up as eth0 in Linux. When configuring a single physical interface called `BOOTIF`,

it no longer matters which interface is actually used.

For some interface types like VLAN and channel bonding, the node installer will halt 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 try again may be necessary.

IPMI Interfaces, if present and set up in the node's configuration will also be initialized with the correct IP address, net mask and user/password settings. Just before the node installer ends and hands over control to the local init process it will bring down all network devices again. They will be brought back up again by the network init scripts from the local hard disk.

6.3.4 Install-Mode

The *install-mode* determines to a large extent what the node installer does. The install-mode can have one of three values; `AUTO`, `FULL` and `MAIN`. If the install-mode is set to `FULL`, the node installer will wipe all contents from the local hard disk and completely repartition the drive. After that it will create new file systems and synchronize a full image onto the drive.

When the install-mode is set to `AUTO`, the node installer will check the local hard disks partition table and file systems against the node's configuration. Only if the partition table or file systems do not match the configuration, or they have become corrupted, will the node installer wipe the disk and recreate the partitions and file systems. If the hard disk partitions and file systems are healthy, the node installer will perform an incremental software image synchronization. Usually this means the process is fairly quick because there is little difference between the software image and the local drive.

Setting the install-mode to `MAIN` will just halt the installer in maintenance mode. It will not touch the local hard disk and allows for manual investigation of specific problems.

The install-mode can be set in several ways, both permanently or just temporarily for the next boot. To determine the `install-mode`, the node installer examines the following properties in order of precedence until it finds an install-mode:

1. Install-Mode selected from PXE menu shown on the console of the node before it loads the kernel and ramdisk (see figure 6.7). (This only affects the current boot.)
2. The `nextinstall-mode` property of the node configuration. (Usually empty, if set the property will be cleared when the node installer ends.)
3. The `install-mode` property of the node configuration. (Usually empty.)
4. The `install-mode` property of the node's category.
5. A dialog (see figure 6.8) on the console of the node gives the user a last opportunity to overrule the install-mode as determined by the node installer.

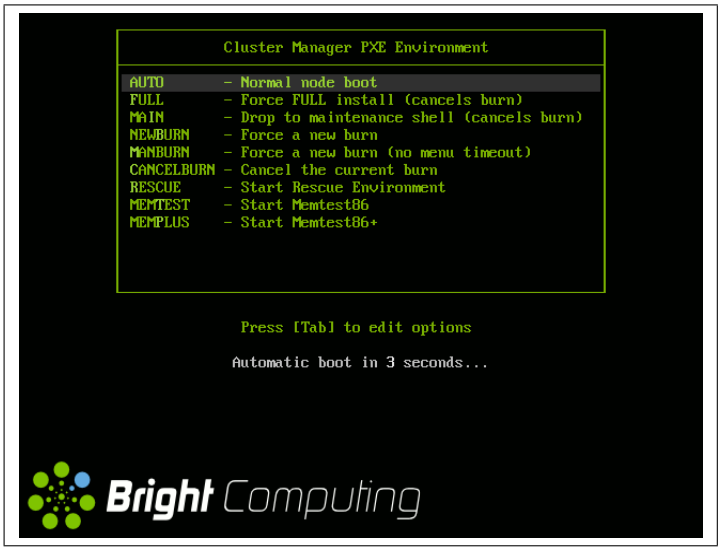

Figure 6.7: PXE Menu

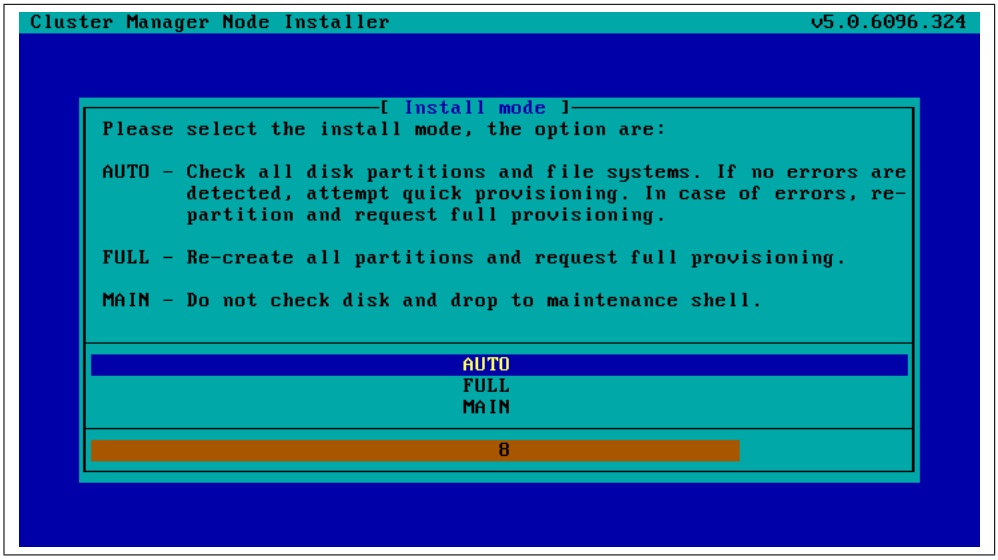

Figure 6.8: Install-Mode

6.3.5 Local Hard Disk Partitions And File Systems

In most cases the install-mode will be set to `AUTO`, which means the node installer will check the local hard disk(s) partitions and file systems and recreate partitions and file systems in case of errors. To do this the node installer will compare the contents of the local hard disk(s) against the hard disk layout as configured in the node's category. The node installer is capable of creating advanced hard disk layouts, including software raid and LVM setups. Some example hard disk layouts, including documentation, are given in appendix D. Once the node installer has checked the drive(s) and, if required recreated the layout, it will mount all file systems to allow the hard disk contents to be synchronized with the contents of the software image.

6.3.6 Synchronize Local Hard Disk With The Software Image

Synchronizing the local file systems with the contents of the software image associated with the node (through its category) is not done by the node installer. All the node installer does is send a provisioning request to the head node CMDaemon. Depending on the install-mode and (in case of AUTO install-mode) the status of the local drive, the node installer will request either full provisioning or sync provisioning.

The provisioning system inside the head node CMDaemon will receive the provisioning request and try to assign the provisioning task to one of the provisioning nodes. As soon as a provisioning node is selected and the image synchronization has started, the node installer gets notified. When the image synchronization has completed, whether successful or not, the node installer will also be notified.

The actual image synchronization is performed using the rsync utility. Depending on the type of synchronization, full or sync, a different exclude list will be passed to the rsync process. The exclude lists are defined as properties of a node's category. The `excludelistsyncinstall` property is used for sync provisioning and the `excludelistfullinstall` property is used for full provisioning.

The exact syntax of these exclude lists can be quite peculiar. For a complete description, please refer to the rsync manual pages, specifically the `INCLUDE/EXCLUDE PATTERN RULES` section. The exclude lists are passed to rsync using its `-exclude-from` option.

When a slave node has more than one network interface, it can sometimes be useful to send the image data over a specific interface. This is particularly useful when one interface is faster than the other. To determine which network to use, the provisioning system will look at the `provisioninginterface` property of the node configuration. By default this is set to `BOOTIF`.

To transfer the image data, the provisioning system can use encrypted or unencrypted protocols. This is determined by the `provisioningtransport` property of the node configuration. Setting this to `rsyncdaemon` causes the data to be sent unencrypted, setting it to `rsyncssh` causes the data to be sent encrypted. Although encrypting the data may seem beneficial, it also severely increases the load on the provisioning node(s). It is therefore recommended to only use the `rsyncssh` mode in cases where the network can not be trusted. The default is to use rsyncdaemon based transfer.

6.3.7 Write Network Configuration Files

After synchronizing the local hard disk with the node's software image, the node installer sets up configuration files for each configured network interface. It then creates relevant files such as

`/etc/sysconfig/network-scripts/ifcfg-eth0`

on the local hard disk. When the node installer ends it brings down all network devices and hand over control to the local `/sbin/init` process. Eventually the local init scripts uses the network configuration files to bring the interfaces back up.

6.3.8 Create Local `/etc/fstab` File

The `/etc/fstab` file on the local hard disk should contain most partitions in order to have the init process mount them. As the actual hard disk

layout is configured in the category, the node installer is able to generate a valid local `/etc/fstab` file. In addition to all mount points defined in the disk layout several extra mount points can be added. These extra mount points, for example NFS imports, can be defined both in the node's category and in the node configuration. From the `cmsh` the extra mount points can be managed from the `fsmounts` submode of the `category` or `device` mode.

6.3.9 Install Grub Bootloader

Optionally the node installer can install a boot record on the local hard drive. This allows the slave node to be started without the network during future startups. The node installer will install the GRUB bootloader in the MBR of the local hard disk if the `installbootrecord` property of the node configuration or node category is set.

6.3.10 Run Finalise Scripts

In some occasions some specific custom commands need to be executed before handing over control to the local init process. An example could be some piece of unsupported hardware that needs to be initialized. Another reason could be the need for some configuration file which can not be added to the image because it needs node-specific settings. In these cases the custom commands can be added to a finalise script. A finalise script can be added to both a node's category and the node configuration. The node installer will first run the finalise script from the node's category and then, if defined, the finalise script from the node's configuration. The node installer will set several environment variables which can be used by the finalise script. Appendix E contains an example script which documents all variables as well.

6.3.11 Unload Specific Drivers

Some kernel drivers are only required during the installation of the node. After that, they just cause unnecessary interruptions and could degrade node performance. An example of such an issue are the IPMI drivers. They are required in order to have the node installer configure the IP address of any IPMI cards. But once configured, the driver is no longer needed and just consumes CPU cycles.

The node installer can be configured to unload a specified set of drivers just before it hands over control to the local init process. To configure the list of drivers, edit the `removeModulesBeforeInit` setting in the node installer configuration file `/cm/node-installer/scripts/node-installer.conf`.

6.3.12 Switch To Local Init Process

At this point the node installer is done. The node's local hard disk 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 will continue the boot process and start all runlevel services. From here on the boot process continues as if the machine was started from the hard disk just like any other regular Linux machine.

6.4 Node States

Throughout the boot process the slave node will send several state change messages to the head node CMDaemon. During a successful boot process the slave node will go trough 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 slave CMDaemon connects to the head node CMDaemon.

Several additional 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 hard disk, 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 slave CMDaemon. 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 slave. It could indicate the node has crashed while running the node installer.
- INSTALLER_REBOOTING - In some cases the node installer will have to reboot the slave node to load the correct kernel. Before rebooting it will set this state. If the reboot takes too long, the head node CMDaemon will set the state to INSTALLER_FAILED.

6.5 Updating Running Slave Nodes

Changes made to the contents of a software image can be rolled out to the slave nodes by rebooting them. However, updating running slave nodes with the latest changes from the software image is also possible without rebooting. Such an update can be requested using cmsh or cmgui and will be queued and delegated to a provisioning node, just like an ordinary provisioning request. A separate exclude list, defined in the excludelistupdate property of the slave node's category, is used when updating. The exact syntax of the exclude list can be quite peculiar. For a complete description, please refer to the rsync manual pages, specifically the INCLUDE/EXCLUDE PATTERN RULES section. In addition to the paths excluded using the excludelistupdate property, the provisioning system will automatically add all NFS, Lustre and GPFS imported file systems on the slave node (if applicable). If this would not be done one would run the risk of wiping all data on the network volumes since that data is not part of the software image! To start an update using the cmsh:

Example

```
[mycluster->device]% imageupdate -n node001
Performing dry run (use -w to perform real update) ....
Provisioning started on node node001
node001: image update in progress ...
[mycluster->device]% Provisioning completed on node node001
```

By default the imageupdate command will perform a dry run, which means no data on the slave node is actually changed. Before passing the -w switch, it is highly recommended to carefully analyse the rsync output using the synclog command:

Example

```
[mycluster->device]% synclog node001
Wed, 25 Mar 2009 11:27:45 CET - Starting rsync ssh based provisioning. \
Mode is UPDATE.

building file list ... done
./
cm/local/apps/cmd/etc/
dev/
<more paths listed>
var/lock/
var/log/
deleting etc/resolv.conf.predhclient

sent 1021174 bytes  received 99 bytes  157118.92 bytes/sec
total size is 1590423950  speedup is 1557.30 (DRY RUN)

Wed, 25 Mar 2009 11:27:51 CET - DRY RUN Rsync completed, \
NO DATA WAS ACTUALLY WRITTEN!
[mycluster->device]%
```

If the user is now happy with the changes that will be made, invoke the imageupdate command again and pass -w:

Example

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

6.6 Troubleshooting The Slave Node Boot Process

During the slave node boot process there are several common issues that could lead to an unsuccessful boot. This section will describe some of these issues and their solutions. It will also provide general hints on how to analyse boot problems.

6.6.1 Node installer logging

The first place to look for hints on why a slave node fails to boot is usually the node installer log file. The node installer sends logging output to syslog. In a default Bright Cluster Manager syslog set up, the messages will

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 slave node the log file is also accessible by pressing Alt+F7 on the keyboard.

6.6.2 Provisioning Logging

The provisioning system sends log information to the CMDaemon log file. By default this is in /var/log/cmdaemon. It is also possible to retrieve the log file of the image synchronization using the synclog command in the cmsh. In most cases of provisioning problems the last bit of the log file contains hints. This is why we pipe the log output trough the tail command:

Example

```
[mycluster]% device
[mycluster->device]% synclog node001 | tail
var/tmp/
var/yp/
var/yp/nicknames
var/yp/binding/

sent 1593970435 bytes  received 801090 bytes  10389391.04 bytes/sec
total size is 1591044712  speedup is 1.00

Tue, 24 Mar 2009 15:59:44 CET - Rsync completed.

[mycluster->device]%
```

6.6.3 Ramdisk Can Not Start Network

The ramdisk needs to activate the slave node's network interface in order to fetch the node installer. In order to activate the network device, the correct kernel module needs to be loaded. If this does not happen, it will result in boot failures. The console of the slave node will display something similar to figure 6.9.

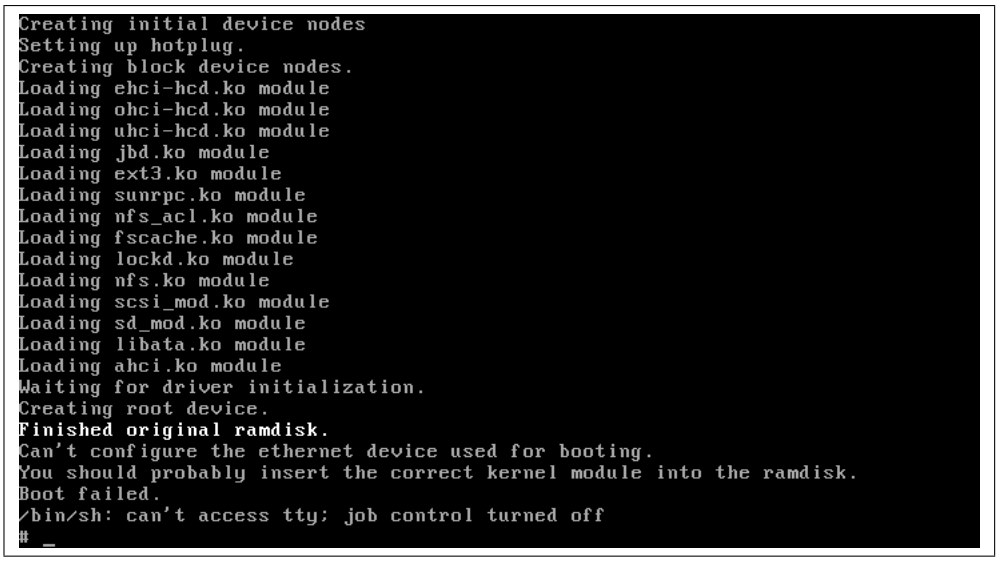

Figure 6.9: No Network Interface

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 the `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 a minute before the ramdisk creation event comes in.

6.6.4 Node Installer Can Not Create Disk Layout

When the node installer is not able to create a disk layout it will display a message similar to figure 6.10. The node installer log file will contain 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. This
node can not be installed until the problem is corrected.
```

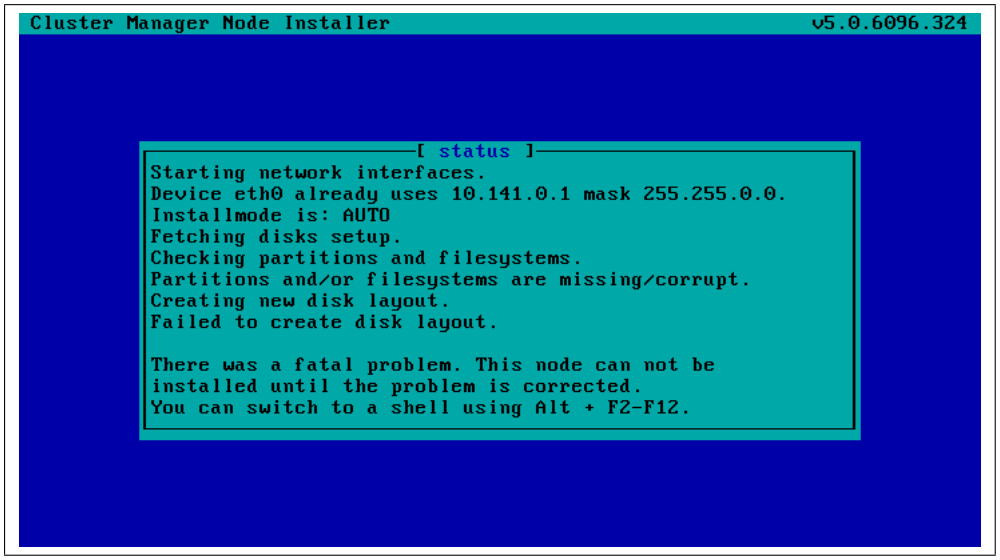

Figure 6.10: No Disk

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. For example, to add the `ahci` module to the default image using the `cmsh`:

Example

```
[mycluster]% softwareimage use default-image
[mycluster->softwareimage[default-image]]% kernelmodules
[mycluster->softwareimage[default-image]->kernelmodules]% add ahci
[mycluster->softwareimage[default-image]->kernelmodules[ahci]]% commit
Initial ramdisk for image default-image was regenerated successfully
[mycluster->softwareimage[default-image]->kernelmodules[ahci]]%
```

Note that after committing the change, it can take a few minutes before the ramdisk creation event comes in.

6.6.5 Node Installer Can Not Start IPMI Interface

In some cases the node installer is not able to configure a slave node's IPMI interface. It will display an error message similar to figure 6.11. 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 will still not be 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 finalise script could then be used to run the required commands instead.

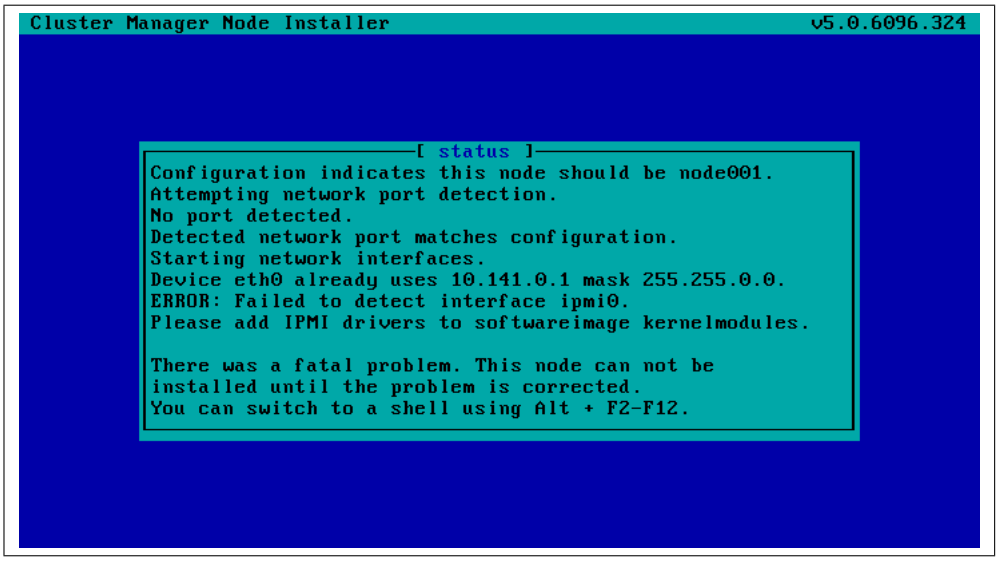

Figure 6.11: No IPMI Interface

7

Software Image Management

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

7.1 Bright Cluster Manager RPM Packages

Bright Cluster Manager relies on the RPM Package Manager (`rpm`) to manage its software packages. The filename of a Bright Cluster Manager RPM file typically has the following form:

package-version-revision`_cm`*x.y*`.`*architecture*`.rpm`

- *package* is the name of the package (e.g. `mpich-ge-gcc-64`).
- *version* is the version number of the package (e.g. `1.2.7`).
- *revision* is the revision number of the package (e.g. `40`).
- *architecture* is the architecture for which the RPM was built (e.g. `x86_64`).
- *x.y* is the version of Bright Cluster Manager for which the RPM was built (e.g. `5.0`).

Hence, an RPM may have the name:

```
mpich-ge-gcc-64-1.2.7-40_cm5.0.x86_64.rpm
```

For more information about the RPM Package Manager, please refer to:

`http://www.rpm.org`.

7.2 Installing & Upgrading Packages

Once Bright Cluster Manager has been installed, Bright Cluster Manager software packages can be installed and/or upgraded by fetching the corresponding RPM packages and using the `rpm` command-line utility to install and/or upgrade them. However, a more convenient way of managing packages is by using the YUM tool. YUM can be used to install extra

packages or update installed packages. The following command can be used to obtain a list of all available packages:

```
yum list
```

The following command will install a new package:

```
yum install packagename
```

Installing updates for all installed packages can be done through the following command:

```
yum update
```

Bright Computing maintains YUM repositories containing package updates at:

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

By default, YUM will be configured to fetch updates from this repository (see `/etc/yum.repos.d/cm.repo`).

Accessing the YUM repositories manually (i.e. not through YUM) will require a username and password. If you have not already received a username or password, please contact `support@brightcomputing.com` to receive these.

Occassionaly it may be desirable to flush the caches that YUM uses to speed up its operations. Flushing caches will cause YUM to fetch fresh copies of all index files associated with a repository. To flush all caches the following command may be used:

```
yum clean all
```

As an extra protection to prevent Bright Cluster Manager installations from receiving malicious updates, all Bright Cluster Manager packages have been signed with the Bright Computing GPG public key (`0x5D849C16`). This key is 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 will be asked whether the Bright Computing public key should be imported into the local RPM database. Before answering `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, by using the following command:

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

7.3 Managing Packages Inside Images

Installing or updating packages inside a slave-image can be handled in a similar fashion. The `rpm` command supports the `--root` flag which can

be useful when dealing with slave-images. To install an RPM inside the default slave-image, one would use the following command:

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

A similar construct is supported by YUM through the `--installroot` flag. Updating all packages in the slave image can be done by issuing the following command:

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

Using the `chroot` command, it is possible to accomplish the same by first "chroot-ing" 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 has not been packaged in an RPM file in an image. For example, the following sequence of commands would install an application into a slave-image:

```
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 slave-image, it is important to realize that `chroot` only changes the file system that is visible to processes. Running installation scripts that e.g. attempt to kill or re-start a system service, may result in unexpected behaviour on a head node. Note that post-install scriptlets included in RPM packages could potentially cause similar problems.

7.4 Kernel Updates

In general it is a good idea to be careful about updating the kernel on a head node and in particular in a slave-image. Packages that should not be updated automatically can be 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 be used:

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

To exclude a package (e.g. `kernel`) permanently from all YUM updates, the package may be included in the (space-separated) exclude list for a repository. Exclude lists can be found in the files in the `/etc/yum.repos.d` directory.

When a kernel in a slave-image has been updated, the new kernel will not automatically be used. It is important to use either `cmgui` or `cmsh` to enable the kernel for the software image. In `cmgui` this is done by going to the `Settings` tab for the software image, selecting the new kernel ver-

sion from the drop-down menu, and clicking save. This will cause a new initial ramdisk to be generated. Using `cmsh`, this is done by modifying the `kernelversion` property of a software image.

8

Day-to-day Administration

This chapter discusses several topics relevant for day-to-day administration of a cluster running Bright Cluster Manager.

8.1 Parallel Shell

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

8.2 Disallowing User Logins on Slave Nodes

Users are able to run computations on the cluster by submitting jobs to the workload management system. Workload management is only effective if cluster users do not run jobs outside the workload management system. In certain environments it may be desirable to enforce this policy by disabling user-logins on the computational nodes. A simple way to accomplish this, is by adding the following line to the `/etc/ssh/sshd_config` file **in the slave image**:

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

When the internal domain is not `cm.cluster` or when the node hostnames do not have the `node` prefix, the line above should be changed accordingly. After modifying the image, the change should be propagated to the nodes by either rebooting or by using the `pupdate` command and subsequently issueing the following command on all nodes using `pexec`:

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

This change will allow only the `root` user to log in to a node from the head node. Users may 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 (see workload management system documentation).

8.3 Bright Cluster Manager Bug Reporting

Bright Cluster Manager is constantly undergoing development. While we are doing our best to produce a rock solid clustering environment, it is possible that you may run into a bug at some point. To report a bug to the Bright Cluster Manager development team, please send a bug-report to:

```
support@brightcomputing.com
```

Please make sure to include as many details as possible that would help the development team to reproduce the situation in which the problem occurred.

8.4 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 slave-images) is backed up. Unless the slave hard drives are used to store important data, it is not necessary to back up slave 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.For more information, please refer to `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. For more information, please refer to `http://www.rsnapshot.org`.

8.5 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 shall assume 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 involves 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.

8.5.1 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 con-

ventional method of entering BIOS Setup mode at system boot time. After the BIOS has been configured, the machine needs to be booted as a slave 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 slave 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
```

In order to modify the image, it must first be mounted using the following command:

```
mount -o loop flash.img /mnt
```

When the DOS image has been mounted, the utility that will write 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` subsequently needs to be 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 must be unmounted by using the following command:

```
umount /mnt
```

After the DOS image has been prepared, it should be booted using the procedure described in section 8.5.3.

8.5.2 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 will describe how to boot the DOS image.

8.5.3 Booting DOS Image

In order to be able 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 needs to be 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 large amounts of 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.

9

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.

9.1 Modules Environment

The *modules environment* (`http://modules.sourceforge.net/`) allows a user of a cluster to modify his 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 Wiki.

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. Please refer to the `modulefile(1)` man-page for details.

Modules can be combined in *meta-modules*. By default, the `default-environment` meta-modules exists, which will allow a user to load a number of other modules at once. Cluster administrators are encouraged to customise the `default-environment` meta-module to set up a recommended environment for their users. The `default-environment` meta-module is empty by default.

9.2 Shorewall

Bright Cluster Manager uses the 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 the Shorewall configuration files have been modified,

Shorewall must be restarted to let the new configuration become effective. The easiest way to restart Shorewall is by issuing the following command:

```
/etc/init.d/shorewall restart
```

In the default set-up, Shorewall is configured to provide 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 has been configured (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 will respond to ICMP ping packets and will allow SSH access from the whole world. Depending on site-policy, one may choose to enable access to port 8081 as well, to allow access to the cluster management daemon.

Full documentation on Shorewall can be obtained at:
`http://www.shorewall.net`

9.3 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) will require an installed license file to be used.

9.3.1 GCC

Package names: `gcc-recent`

9.3.2 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 9.1) 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 can be obtained at the following URLs:

- For Fortran:
  `http://support.intel.com/support/performancetools/fortran/linux/`
- For C++:
  `http://support.intel.com/support/performancetools/c/linux/`

9.3.3 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 can be obtained at:

`http://www.pgroup.com/resources/docs.htm`

9.3.4 Pathscale Compiler Suite

Package name: `pathscale`

The Pathscale Compiler Suite contains optimizing C++ and Fortran compilers and a debugger:

- `pathcc`: Pathscale C compiler
- `pathCC`: Pathscale C++ compiler
- `pathf90`: Pathscale Fortran 90 compiler
- `pathf95`: Pathscale Fortran 95 compiler
- `pathdb-x86_64`: Pathscale debugger

Full documentation for the Pathscale Compiler Suite can be obtained at:

`http://www.pathscale.com/docs.html`

Note: Pathscale versions prior to 3.0 will not be able to compile C++ code on a GCC4 based system (e.g. OpenSuSE 10, Fedora Core 5).

9.3.5 FLEXlm License Daemon

Package name: flexlm

For the Intel and PGI compilers a FLEXlm license must be present in the /cm/shared/licenses tree. While the presence of the license file is typically sufficient for workstation licenses (i.e. a license which is only valid on the head-node), floating licenses (i.e. a license which may be used on several machines, possibly simultaneously) will require the FLEXlm license manager to be running: lmgrd.

The lmgrd service will serve 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 will start 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 will 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 will add a system account lmgrd to the password file (since the account will not be assigned a password, it can not be used for logins), which will be used to run the lmgrd process. The lmgrd service can be configured to be started automatically at system boot-up (this is not configured by default) through the following command:

```
chkconfig lmgrd on
```

The lmgrd service can be started manually by running:

```
/etc/init.d/lmgrd start
```

The lmgrd service will log its transactions and any errors to /var/log/lmgrd.log.

More details on FLEXlm and the lmgrd service can be found at:

```
http://www.macrovision.com
```

9.3.6 Pathscale Subcription Server

In contrast to the Intel and PGI compilers, Pathscale does not use FLEXlm licenses. Instead, it uses its own license format and license manager: `pathscale-sub`. Enabling the `pathscale-sub` server at boot-time is done by:

```
chkconfig pathscale-sub on
```

The `pathscale-sub` service can be started manually by running:

```
/etc/init.d/pathscale-sub start
```

Pathscale license files will be picked up by the subscription server if they are placed in the `/cm/shared/licenses/pathscale` directory. More information on the Pathscale Subscription Server is available at:

```
http://www.pathscale.com/subscriptionmgr.html
```

9.4 Intel Cluster Checker

Package name: `intel-cluster-checker`

The Intel Cluster Checker is a tool that can be used to verify that a cluster complies with all of the requirements as set out in the Intel Cluster Ready Specification. This section lists a number of steps that must be taken to certify a cluster as Intel Cluster Ready.

9.4.1 Preparing Cluster

The Intel Cluster Ready specification requires a number of packages to be installed on the head node and slave nodes. To verify that all required packages have been installed, the `cm-config-intelcompliance-master` and `cm-config-intelcompliance-slave` packages must be installed on respectively the head node and software images. These packages guarantee through package dependencies that all Intel Cluster Ready requirements are satisfied. Both packages are normally installed by default on a normal Bright Cluster Manager cluster. If they are not installed for whatever reason, the following commands will install the packages and any packages needed for compliance.

Example

```
yum install cm-config-intelcompliance-master
yum --installroot=/cm/images/default-image install \
                 cm-config-intelcompliance-slave
```

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

9.4.2 Preparing Input Files

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

- As a regular cluster user

- As a privileged cluster user (i.e. root)

Each run 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.xml
- nodelist
- excludefiles

Recipes

The recipe-user-ib.xml, recipe-user-nonib.xml and recipe-root.xml files are 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

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

Throughout both recipe files, several performance thresholds are 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.

For a description of all test modules and parameters, please refer to the Intel Cluster Checker documentation available from:

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 slave 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 # head"; for i in ‘seq 1 150‘;
> do echo node‘printf "%03d" $i‘;
> 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 slave nodes. Modifying the `excludefiles` list is normally not necessary.

9.4.3 Generating Fingerprint Files

Before running the Intel Cluster Checker, a list of all packages installed on the head node and slave nodes must be generated. These lists will be 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 will use the first slave node defined in the `nodes` file as the standard.

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

```
/home/cmsupport/intel-cluster-ready/head-packages.list
/home/cmsupport/intel-cluster-ready/slave-packages.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-packages.list
mv node001-20090522.173125.list /home/cmsupport/intel-cluster-ready/node-packages.list
```

9.4.4 Running Intel Cluster Checker

Regular User Run

The `cmsupport` account is used to perform the regular user run. The following commands will 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 will produce 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 will 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.xml
```

In a heterogeneous cluster the privileged user run will fail as a result of hardware differences. To resolve the failures, it is necessary to create multiple groups of homogeneous hardware. For more information, please consult the Intel Cluster Checker documentation.

9.4.5 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.

10

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.

10.1 HA Concepts

In a cluster with an HA set-up, 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.

10.1.1 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 less services to move in the event of a failover sequence.

On a default HA set-up, 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 set-up
- **LDAP**: running in replication mode
- **MySQL**: running in multi-master replication mode
- **NTP**
- **DNS**

When a HA set-up is created, the above services are automatically reconfigured for an HA environment with two head nodes.

In addition, both head nodes will also receive 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 set-up only the following services will be migrated:

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

10.1.2 Network Interfaces

Each head node in an HA set-up typically has at least an external and an internal network interface, each configured with an IP address. In addition, an HA set-up 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 set-up, 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`).

10.1.3 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.

10.1.4 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 set-up, 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, SAN and DRBD.

NAS

In a NAS set-up, 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.

SAN

In a SAN set-up, 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 GFS and OCFS are capable of being accessed by two head nodes simultaneously.

DRBD

In a set-up with DRBD, both head nodes are mirroring a physical block 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 SAN 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 set-up 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 flexibility over the form of shared storage that is used. Also an administrator can control which filesystems are shared.

10.1.5 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.

10.1.6 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.

10.1.7 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 will be used. If no power control has been defined, a failover sequence must always be initiated manually by the administrator.

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

Globally the process of creating an HA set-up 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

10.2.1 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 set-up, 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.

10.2.2 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`

10.2.3 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.

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 may 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 filesystems.

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

10.2.4 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 set-up 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.

10.3 Managing HA

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

10.3.1 `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	the 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 ...
```

10.3.2 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.

10.3.3 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 set-up 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.

10.3.4 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.

A

Generated Files

Section 3.7.3 describes how system configuration files on head nodes and slave 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	Entire file	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	Entire file	SuSE only
/etc/sysconfig/kernel	CMDaemon	Entire file	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	
/etc/shorewall/interfaces	CMDaemon	Entire file	
/etc/shorewall/masq	CMDaemon	Entire file	
/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	
/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	
/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	
/etc/ntp.conf	CMDaemon	Entire file	
/etc/ntp/step-tickers	CMDaemon	Entire file	

Files generated automatically on slave nodes

File	Generated By	Method	Comment
/etc/exports	CMDaemon	Section	
/etc/fstab	Node installer	Section	
/etc/sysconfig/network/ifcfg-*	Node installer	Entire file	SuSE only
/etc/sysconfig/network-scripts/ifcfg-*	Node installer	Entire file	RedHat only
/etc/postfix/main.cf	Node installer	Section	

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
```

Master directive

Syntax: `Master =` *hostname*

When the `Master` directive is used, the cluster management daemon will start in *slave* mode and will treat the specified host as its master. When no `Master` directive is present, the cluster management daemon will start in *master* 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 = "system"`

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 = "system"

The LDAPUser 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 LDAPUser 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.

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 `IpmiSessionTimeout` 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 = false`

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.

FrozenFile directive

Syntax: FrozenFile = { *filename1*, *filename2* }
Example: FrozenFile = {"/etc/dhcpd.conf","/etc/postfix/main.cf"}

The FrozenFile directive can be used to prevent files from being automatically generated. This can be 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 slave nodes is being configured, but also when the head node is initially installed. For slave nodes, the XML format also allows diskless operation.

D.1 Structure of Partitioning Definition

The global structure of a file that describes a partitioning set-up 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'?>

<!--
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 (/).
- 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.
- If diskless is enabled there should not be any devices.
-->

<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: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='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:group ref='filesystem' minOccurs='0' maxOccurs='1'/>
  </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 Slave Node Partitioning

The following example shows the default layout used for slave 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: Software RAID

This example shows a simple software RAID set-up. 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.5 Example: Logical Volume Manager

This example shows a simple LVM set-up. 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.6 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>
```

E

Example Finalise Script

An example finalise script is given below. The node-installer runs the script when provisioning is complete, but before switching to the local hard drive. The local hard drive is mounted under `/localdisk`. Node specific customizations can be made from a finalise script. Some extra environment variables can be used by the script. Note that while this script does not actually do anything useful, it does show how all the variables can be used. Examine the file in `/env` on the node after booting to discover the contents of all the variables.

```
#!/bin/bash

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

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

  echo "$interface type=$type" >> /localdisk/env
  echo "$interface ip=$ip" >> /localdisk/env
  echo "$interface netmask=$mask" >> /localdisk/env
done
```

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 slave nodes, set the base name for the slave nodes and the number of digits to append to the base name. Select the correct hardware manufacturer and click `Continue`.
8. Fill in the following settings for the network named `externalnet`:
   - Base Address
   - Netmask
   - Domain name
   - Gateway
   - Name Server 1

   The `externalnet` corresponds to the campus or corporate network that the cluster resides in. Note that assigning the cluster an IP address in this network will be done in one of the next screens.

9. To enable/disable Infiniband and IPMI in the cluster, enable/disable the `ibnet` and `ipminet` networks by checking/unchecking the checkboxes in the corresponding tabs. Click `Continue` to continue.
10. Assign an IP address for the head node on the `externalnet`. This is the IP address that will be used to access the cluster over the network.
11. If necessary, modify the slave node properties. Click `Continue` to continue.
12. 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.
13. Select the DVD drive containing the Bright Cluster Manager DVD and click `Continue`.
14. 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.
15. Optionally you may modify the disk layout for the master by selecting a pre-defined layout. The layout may be fine-tuned by editing the XML partitioning definition. Click `Continue` to continue.
16. Select a time-zone and optionally add NTP time-servers. Click `Continue` to continue.
17. Enter a hostname for the head node. Enter a password that will be used for system administration twice and click `Continue`.
18. Review overview screen, and click the `Start` button to start the installation.
19. 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 Slave 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 slave nodes to boot from the network, and boot the slave 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 6.6 contains more information on how to diagnose problems during the slave 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
5. The slave node will now be provisioned and will eventually boot. In case of problems, consult section 6.6
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