HPC

HPC BAS4

Administrator's Guide

REFERENCE
86 A2 30ER 11

HPC

HPC BAS4

Administrator's Guide

Hardware and Software

December 2007

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 A2 30ER 11

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the rights of the proprietors of the trademarks mentioned in this manual.

All brand names and software and hardware product names are subject to trademark and/or patent protection.

Quoting of brand and product names is for information purposes only and does not represent trademark misuse.

Preface

Scope and Objectives

The purpose of this guide is to explain how to configure and manage Bull High Performance Computing (HPC) clusters, using the administration tools recommended by Bull.

It is not in the scope of this guide to describe in depth the Linux administration functions. For this information, please refer to the standard Linux distribution documentation.

Intended Readers

This guide is for HPC cluster system administrators.

Prerequisites

The installation of all hardware and software HPC components must have been completed.

Structure

This guide is organized as follows:

Chapter 1. Explains some *General Concepts* for Bull Linux HPC systems.

Chapter 2. *HPC Configuration*
Describes some basic configuration tasks including password management, security settings. It also describes how to run parallel commands, and kernel tuning parameters.

Chapter 3. *Cluster Database Management*
Describes the commands and the tools which enable the administrator to display and to change the Cluster Database.

Chapter 4. *Parallel File Systems*
Explains how these file systems operate on a Bull HPC system. It describes in detail how to install, configure and manage the Lustre file system.

Chapter 5. *Software Deployment*
Describes how to use KSIS to deploy, manage, modify and check software images.

Chapter 6. *Resource Management*
Explains how Quadrics RMS and SLURM Resource Managers work in order to have an optimal management of the resources.

Chapter 7. *Batch Management*
Explains how to optimize the execution of programs using TORQUE.

Chapter 8. *Monitoring*
Describes NovaScale Master - HPC Edition monitoring tool for Bull HPC systems.

Chapter 9. *Storage Devices Management*
Explains how to setup the management environment for storage devices, and how to use storage management services.

Chapter 10. *Kerberos – Network Authentication Protocol*
Describes how to set up and use Kerberos.

Chapter 11. *Cluster High Availability*
Explains the main concepts for implementing High Availability on a Bull HPC system.

Chapter 12. *Management Node High Availability*
Explains how to implement Management Node High Availability using Cluster Suite and specific HA scripts.

Chapter 13. *I/O Node and Lustre File System High Availability*
Explains how to implement High Availability for I/O Nodes and Lustre file system.

Bibliography

- Bull *HPC BAS4 Installation and Configuration Guide* (86 A2 28ER).
- Bull *HPC BAS4 User's Guide* (86 A2 29ER).
- Bull *HPC BAS4 Application Tuning Guide* (86 A2 19ER).
- Bull *HPC BAS4 Maintenance Guide* (86 A2 46ER).
- Bull *NovaScale Master Remote Hardware Management CLI Reference Manual* (86 A2 88EM).
- The *Software Release Bulletin* (SRB) provides release-specific information, and details of restrictions resulting from known problems.
- Bull *Voltaire Switches Documentation CD* (86 A2 79ET)

Highlighting

- Commands entered by the user are in a frame in "Courier" font. Example:

```
mkdir /var/lib/newdir
```

- Commands, files, directories and other items whose names are predefined by the system are in "Bold". Example:
  The **/etc/sysconfig/dump** file.
- Text and messages displayed by the system to illustrate explanations are in "Courier New" font. Example:
  `BIOS Intel`

- Text for values to be entered in by the user is in "`Courier New`". Example: `COM1`
- *Italics* Identifies referenced publications, chapters, sections, figures, and tables.
- < > identifies parameters to be supplied by the user. Example: `<node_name>`

Warning:

A Warning notice indicates an action that could cause damage to a program, device, system, or data.

Table of Contents

List of Figures

List of Tables

Chapter 1. General Concepts

This chapter describes the following topics:

- 1.1 *Cluster Architecture*
- 1.2 *Management Functions and Corresponding Products*

1.1 Cluster Architecture

A cluster is an aggregation of identical or very similar individual computer systems. Each system in the cluster is a **node**.

The cluster systems are tightly-coupled using dedicated network connections such as high-performance, low-latency interconnects. All systems in a cluster share common resources such as storage over dedicated cluster file systems.

A typical cluster infrastructure consists of **Compute nodes** for intensive calculation and **Service Nodes** for management, storage and software development services.

- **Compute Nodes** are optimized for code execution; limited daemons run on them, these nodes are not supposed to write to disk but transfer data to service nodes.
- **Service Node(s)** cover the following functionalities:
  - **Input/Output (I/O) Nodes** to store data in storage units.
  - A **Management Node** to administrate, manage and exploit the cluster.
  - **Login Nodes** to access the cluster and to submit jobs.
  - Other nodes can support services or act as servers for both parallel and distributed file-systems.

Different **networks** are used, each one dedicated to a function:

- High speed interconnect switches and boards to transfer data between compute nodes and I/O nodes.
- An administration network including Ethernet and serial networks, which are used for cluster management and maintenance.
- A backbone to link the HPC system and the external world.

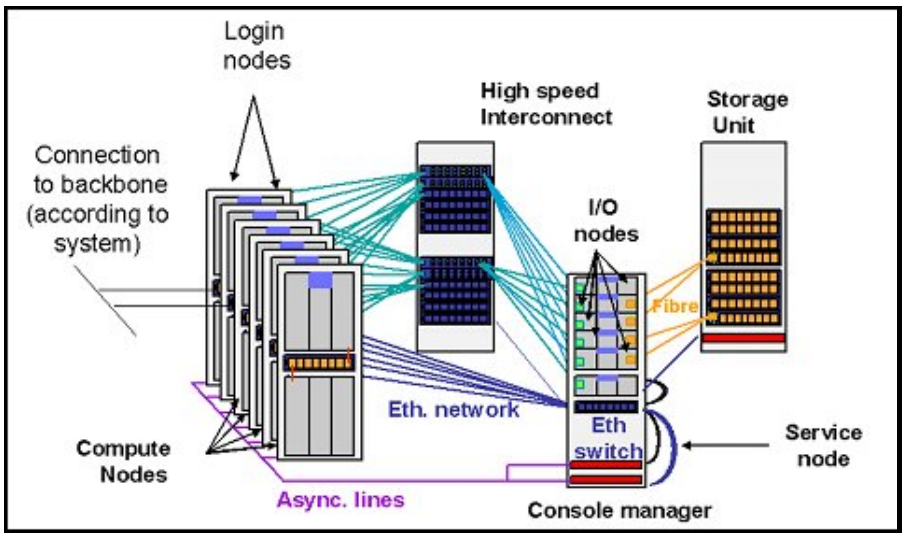


Figure 1-1. A typical HPC architecture

To benefit from the advantages of a cluster configuration, Bull provides a software environment, which makes the cluster very efficient. For example:

- A Parallel File System (**Lustre** from CFS) aggregates your distributed storage into a single group or an enterprise-wide file system.
- The Message Passing Interface (**MPI**) allows the programs to run across all nodes.
- A resource manager manages and controls access to distributed resources. According to the system this may either be Quadrics Resource Management System (**RMS**) or **SLURM** – an open-source resource manager.

The Bull cluster administration scheme is centralized on a node. All management products run on this platform called the Management Node. All nodes are controlled and monitored from this central point of management with the objective of ensuring that CPU activity and network traffic on the compute and I/O nodes runs as efficiently as is possible.

The management tools are mainly Open Source products. These products are configurable and adaptable to management needs and can be deactivated on demand if necessary.

These products have been developed and adapted to Bull platforms and their environments. All management functions are available through a browser interface or through a remote command mode. Users can access management functions according to their profile.

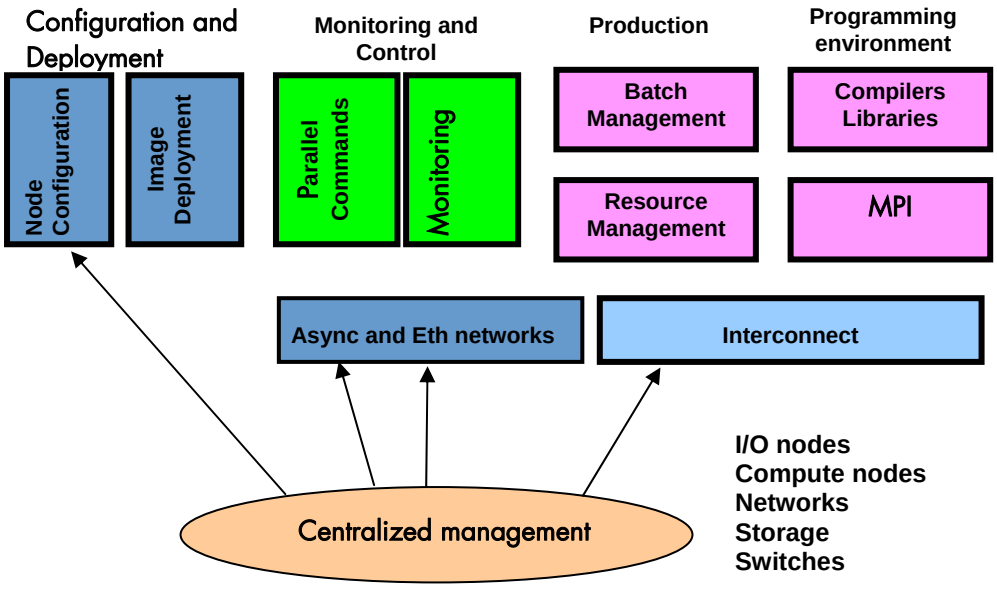


Figure 1-2. Management Functions

1.2 Management Functions and Corresponding Products

The management functions are performed by different products which are briefly presented below.

Configuration and Software Management

- **pdsh** is used to run parallel commands.
  See Chapter 2 –*HPC Configuration* for more information.
- **KSIS** and the Ethernet network enable the deployment of images.
  See Chapter 5 –*Software Deployment (KSIS)* for more information.
- The Cluster DataBase - **dbmConfig**, **dbmCluster**, **dbmNode** and other commands are available to manage the Cluster Database.
  See Chapter 3 –*Cluster DataBase Management* for more information.
- **Kerberos** is used to validate the identity of users, services and machines for a cluster.
  See Chapter 10 – *Kerberos – Network Authentication Protocol* for more information.

Resource and Batch Management

- **RMS** from **Quadrics** or **SLURM** (Simple Linux Utilities Resource Manager) is an open-source scalable resource manager.
  See Chapter 6 – *Resource Management* for more information.
- **BLBS** (Bull Load Balancing System) provides load-balancing services.
  See Chapter 6 – *Resource Management* for more information.
- **TORQUE** is used for Batch management.
  See Chapter 7 – *Batch Management (TORQUE)* for more information.

Monitoring

- **NovaScale Master - HPC Edition** monitors the cluster and activity and is included in the delivery for all Bull HPC Clusters.
  See Chapter 8 – *Monitoring with NovaScale Master – HPC Edition* for more information.

Console Management

- For **NovaScale 5xxx/6xxx** Series platforms **Conman** provides the administrator with access to the system consoles, and is used with the Platform Administration Processor **(PAP)** to manage the hardware, for example, powering on/off and monitoring the performance of the hardware components.
- For **NovaScale 4xxx** Series platforms **Conman** is used with NovaScale commands and Service Manager software which interface with Baseboard Management Controllers (**BMC**) maintenance coprocessors to manage and monitor the cluster.
- For **NovaScale 3xxx** Series platforms **ipmitools** commands are used for console management and work with **IPMI 1.5** and **2** and with NovaScale commands to monitor and manage the cluster.

See the Bull *HPC BAS4 Maintenance Guide* for more information.

Maintenance Tools

- **nsctrl** carries out various hardware and firmware tasks from the Management Node.
- **syslog-ng** manages the System Logs.
- **mkCDrec** performs system backups and restores. This function is available from the Service Node.
- **LKCD** captures and analyzes crash dumps.
- **Nodechecking** allows you to run tests on a node.
- **qsctrl** is used to check the Quadrics network status.
- **postbootchecker** verifies starting nodes.
- **NSdoctor** analyzes the reasons why a node has been excluded by RMS.

See the Bull *HPC BAS4 Maintenance Guide* for more information on the maintenance tools and cluster maintenance procedures.

Chapter 2. HPC Configuration

Most configuration tasks are performed at the time of installation. This chapter indicates how the Administrator can perform some additional basic configuration tasks. It also deals with the security policy for HPC systems.

The following topics are described:

- 2.1 *Configuring Services*
- 2.2 *Modifying Passwords and Creating Users*
- 2.3 *Managing Partitions*
- 2.4 *Creating Swap Partitions*
- 2.5 *Configuring Security*
- 2.6 *Running Parallel Commands with pdsh*
- 2.7 *Day to Day Maintenance Operations*

For more information, refer to the *Bull HPC Installation and Configuration Guide*, which describes the different steps for installing and configuring Bull HPC systems.

2.1 Configuring Services

- To run a particular functionality when Linux starts enter the command:

```
/sbin/chkconfig --level 235 name_of_service on
```

- To display Help information enter the command:

```
/sbin/chkconfig --help
```

- To display the list of services available, enter the command:

```
/sbin/chkconfig --list
```

Note:

Some utilities, such as **sendmail** and **nfs**, are not enabled by default. The administrator is responsible for their configuration.

2.2 Modifying Passwords and Creating Users

Two users are created when Linux is installed:

root administrator (password root)

linux ordinary user (password linux)

These passwords must be changed as soon as possible.

- To change the passwords use one of the following commands
  - **passwd user_name** command for root users
  - **passwd** command for ordinary users.
- To create new users enter the **/usr/sbin/useradd** command

```
useradd –g “group” –d “home login
```

2.3 Managing Partitions

This section explains how to add, delete or modify partitions.

Use the Linux **/sbin/parted** command to edit the GPT (GUID Partition Table) format of the disk. By default, the **parted** command loads the first disk **/dev/sda**.

To specify another disk (for example **/dev/sdb**), enter:

```
/sbin/parted /dev/sdb
```

- Run the **print** command to view the partitions table.
- Run the **help** command to view the commands.
- Run the **mkpartfs** command to create one or more partitions. For example:

```
mkpartfs primary ext2 6241.171 7184.955
```

Note:

The **ext3** fs-type is not implemented in this version of **parted**, but the **fs**-type can be modified using the Linux **mkfs** command as described below.

- Run the **resize** command to modify the size of a partition.
- Delete a partition using the rm <minor number> command corresponding to the partition to be deleted.
- Run q to validate the changes.

Use the Linux **/sbin/mkfs** command to modify the file system type.

- For ext3 file system run: /sbin/mkfs –j <device>. For example:

```
mkfs -j /dev/sdb8
```

- For other types, run: /sbin/mkfs –t <fs-type> <device>. For example:

```
mkfs -t ext2 /dev/sdc8
```

- Next, the mount points have to be defined in **/etc/fstab** file and these partitions mounted using the /bin/mount -a command so that the partitions will be mounted when the system is restarted.

2.4 Creating Swap Partitions

The **/sbin/mkswap** command lets you create swap partitions.

- Use the **/sbin/parted** command to edit the GPT format of the disk.
- Use the mkpartfs command to create one or more additional swap partitions. For example:

```
mkpartfs primary linux-swap 6241.171 7184.955
```

- Run the **mkswap** command.
- Run the **/sbin/swapon –a** command to take this swap into account.

For example, assuming your swap is on `/dev/sdc1`, do as follows to recreate it with a larger size:

```
#/sbin/swapoff -a
#/sbin/parted -s /dev/sdc rm 1
#/sbin/parted -s -- /dev/sdc mkpartfs primary linux-swap 0 <size of your disk in
          Mb, given by 'parted /dev/sdc p' 70000 for a 74 Gb disk for example>
#/sbin/mkswap -p 65536 -f -v1 -L SWAP-sdc1 /dev/sdc1
#/sbin/swapon -a
```

2.5 Configuring Security

This section provides the administrator with basic rules concerning cluster security. According to the cluster configuration you can set up different security policies.

The Management Node is the most sensitive element from a security point of view. This node will submit jobs in batch mode and it is a central point for management. This is the reason why security has to be enforced regarding access to this node. Very few people should be able to access this node and this access should be made using **OpenSSH** to eliminate eavesdropping, connection hijacking, and other network-level attacks effectively.

Compute node and I/O nodes should not have interactive logins. This means that no user except root should have access to these nodes. Management tools like Nagios will have access to both node types, while a batch manager like **TORQUE** will have access to compute nodes only.

If CPU and memory resources are shared among users, each user should not then have access to other partitions.

2.5.1 Setting up SSH

RMS can be configured to use **ssh** rather than **rsh** to execute administrative commands on the root partition. Create the **use-ssh** attribute and set its value to 1 to enable this:

```
rcontrol create attribute=use-ssh val=1
```

When **prun** runs with the **-r** option it executes the following command for each of the selected nodes:

```
/usr/bin/ssh -n ssh-args hostname
```

The optional ssh-args are taken from the **ssh-args** attribute.

SSH User Setup

Carry out the following steps to set up **SSH** for an admin user:

1. Create a public key:

```
ssh-keygen -tdsa -N ''
```

   This creates an **ssh** protocol 2 DSA certificate without passphrase in **~/.ssh/id_dsa.pub**.

2. Append this key to the list of authorized keys in **~/.ssh/authorized_keys2**.

3. Run **ssh** once by hand for each node responding yes at the prompt to add it to the list of known hosts:

```
atlas0: ssh atlas1 hostname
The authenticity of host 'atlas1 (192.168.84.2)' can't be
established.
```

```
RSA key fingerprint is
9c:d8:62:b9:14:0a:a0:18:ca:20:f6:0c:f6:10:68:2c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'atlas1,192.168.84.2' (RSA) to the
list of known hosts.
```

Note:

For the root user there is an authorized keys file for each node as ~root/.ssh/authorized_keys2 is local. The new key must be appended to each of these files.

Please refer to the chapter in this manual on Kerberos for more information on SSH and the use of keys.

2.6 Running Parallel Commands with pdsh

A distributed shell is a tool that allows the same command to be launched on several nodes. Such a function is essential in a cluster environment so that instructions can be carried out on several nodes instead of running the command manually on each node in turn. Different tools can be used to enable this possibility.

pdsh is a utility that runs commands in parallel on all the nodes or on a group of nodes of the cluster. It is a very flexible tool especially for large cluster usage.

pdsh is a multi-threaded client for remote shell commands. It can use different remote shell services, such as **rsh**, **ssh** and **kerberos**.

Three utilities are included in **pdsh**:

- **pdsh** is used to run commands in parallel.
- **pdcp** is used to copy files on a group of nodes in parallel.
- **dshbak** is used to format, sort and display the results of a command launched with **pdsh**.

The **pdsh** utility relies on the security and authentication mechanisms provided by **ssh** and / or **Kerberos** V4 layers on which it is configured. See the chapter in this manual on Kerberos.

2.6.1 Using pdsh

Syntax:

The following commands are the ones which are used most often:

pdsh –R <rcmd_module> -w <node_list> -l user -Options Command

pdsh –R <rcmd_module> -a -x <node_list> -Options Command

pdsh –R <rcmd_module> -g <group_attributes> -Options Command

The most important options are described below. For a complete description of the options, refer to the **pdsh** man page.

Standard Target Node List Options:

-w <node_list> Targets the specified list of nodes. Do not use the -w option with any other node selection option (-a, -g). The node list can be a comma-separated list (node 1, node2, etc.); no space is allowed in the list. If you specify only the '-' character, the target hosts will be read from stdin, one per line. The node list can also be an expression such as host[1-5,7]. For more information about node list expressions, see the HOSTLIST EXPRESSIONS in the pdsh man page.

-x <node_list> Excludes the specified nodes. The -x option can be used with other target node list options (-a, -g, -A). The node list can be a comma-separated list (node1, node2, etc.); no space is allowed in the list. The node list can also be an expression such as host[1-5,7]. For more information about the node list expressions, see the HOSTLIST EXPRESSIONS in the pdsh man page.

Standard pdsh Options:

-S Displays the largest value returned by the remote commands.

-h Displays commands usage and the list of the available rcmd modules and then quits.

-q Lists the option values and the target node list and exits without action.

-b Disables the Ctrl-C status feature so that a single Ctrl-C kills parallel jobs (Batch Mode).

-l <user> This option is used to run remote commands as another user, subject to authorization.

-t <cnx_timeout>Sets the connection timeout (in seconds). Default is 10 seconds.

-u <exec_time> Sets a limit on the amount of time (in seconds) a remote command is allowed to execute. Default is no limit.

-f <remote_cds_numb>
Sets the maximum number of simultaneous remote commands. Default is 32.

-R <rcmd_module>
Sets the rcmd module to use. The list of the available rcmd modules can be displayed using the -h, -V, or -L options. The default module is listed with -h or –V options.
Note: Instead of using this option, you can set the PDSH_RCMD_TYPE environment variable.

-L Lists information about all loaded **pdsh** modules and then quits.

-d Includes more complete thread status when SIGINT is received, and displays connection and command time statistics on stderr when done.

-V Displays pdsh version information, along with the list of currently loaded pdsh modules.

Group Attributes Options:

The following options use the cluster's group attributes as defined in the **/etc/genders** file.

-A Targets all nodes defined in the /etc/genders file.

-a Targets all nodes in the /etc/genders file except those with the pdsh_all_skip group attribute.

Note:

The **pdsh –a** command is equivalent to the **pdsh -A -X pdsh_all_skip** command. For example, you can set the **pdsh_all_skip** group attribute to the Service Nodes to exclude these specific nodes from cluster.

-g <gp-attr1[,gp_attr2,...]> Targets the nodes that have any of the specified group attributes. This option cannot be used with the -a and -w options.

-X <gp_attr1[,gp_attr2...]> Excludes the nodes that have any of the specified group attributes. This option may be combined with any other node selection options (-w, -g, -a, -A).

Examples:

- To execute the **pwd** command on all the nodes of the cluster using the **ssh** protocol, enter:

```
pdsh -R ssh -a pwd
```

- To list the system name of all nodes using **ssh** protocol, enter:

```
pdsh -R ssh -A uname -a
```

- To define **ssh** as default protocol, enter:

```
export PDSH_RCMD_TYPE=ssh;
```

- To display the date on all nodes, enter:

```
pdsh -A date
ns1: Mon Dec 13 13:44:48 CET 2004
ns0: Mon Dec 13 13:44:47 CET 2004
ns2: Mon Dec 13 13:44:47 CET 2004
ns3: Mon Dec 13 13:44:46 CET 2004
```

- To display the date on all nodes except on node `ns0`, enter:

```
pdsh -A -x ns0 date
ns1: Mon Dec 13 13:44:48 CET 2004
ns2: Mon Dec 13 13:44:47 CET 2004
ns3: Mon Dec 13 13:44:46 CET 2004
```

- To display the date of the IO group nodes and to merge the output of the nodes whose result is identical, enter:

```
pdsh -g IO -x ns0 date | dshbak -c
----------------
ns[2-3]
----------------
 Mon Dec 13 14:10:41 CET 2004
----------------
ns[1]
---------------
 Mon Dec 13 14:10:42 CET 2004
```

2.6.2 Using pdcp

pdcp is a variant of the **rcp** command. Its syntax is not in the form remote_user@node:path. All source files are on the local node. The options which enable the nodes to be reached to be defined are similar to those of **pdsh**.

Syntax:

pdcp –Options … <source [src2…]> <destination>

Examples:

```
pdcp -R ssh -w ns[1-5] /etc/hosts /etc/hosts
pdcp -R ssh -g Analyse /tmp/foo
```

In the first example one carries out a copy of **/etc/hosts** from the node where one **pdcp** executes to all the nodes specified using the – w option by copying across the same path for the command.

For a complete description of the options please refer to the **pdcp** man page.

2.6.3 Using dshbak

One of the problems linked to the execution of commands in parallel on a big cluster, is the exploitation of the results, especially if the command generates a long output. The results of a command executed with **pdsh** are displayed asynchronously and each line is stamped with the node name, as in the following example:

```
pdsh -w ns[0-2] pwd
          ns0 : /root
          ns2 : /root
          ns1 : /root
```

The **dshbak** utility formats the results of a **pdsh** command into a more user friendly form. Note that the results must be directed into a buffer file before being processed by **dshbak**.

Syntax:

dshbak [-c] <buffer_file>

dshbak can be used to create the following formatting:

- The node name, which was displayed on each line, is removed and replaced by a header containing this name.
- The generated list is sorted according to the node name if this name is suffixed by a number (ns0, ns1, ns2… ns500).
- If the **–c** option is present; **dshbak** will displays the identical results for several nodes once only. In this instance the header contains the node list.

Examples:

In the following example, the result of the pdsh command is not formatted:

```
pdsh -R ssh w ns[0-2] rpm -qa | grep qsnetmpipwd

ns1 : qsnetmpi-1.24-31
          ns2 : qsnetmpi-1.24-31
          ns0 : qsnetmpi-1.24-31
```

In the following example, the **pdsh** output is re-directed to **res_rpm_qsnetmpi** file, then the **dshbak** command formats and displays the results:

```
pdsh -R ssh w ns[0-2] rpm -qa | grep qsnetmpipwd > /var /res_pdsh/res_rpm_qsnetmpi

dshbak -c res_rpm_qsnetmpi
---------------
ns[0-2]
---------------
qsnetmpi-1.24-31
```

2.7 Day to Day Maintenance Operations

A set of maintenance tools is provided with a Bull HPC cluster. These tools are mainly Open Source software applications that have been optimized, in terms of CPU consumption and data exchange overhead, to increase their effectiveness on Bull HPC clusters which may include hundred of nodes.
See the Bull *HPC BAS4 Maintenance Guide* for details about these tools

Function	Tool	Purpose
Administration	ConMan ipmitool	Console Management
	nsclusterstop / nsclusterstart	Stopping/Starting the cluster
	nsctrl	Managing hardware and firmware (power on, power off, checking temperature, changing bios, etc)
	Remote Hardware Management CLI	
	syslog-ng	System log Management
	lptools (lputils, lpflash)	Emulex HBA (Host Bus Adapter) Management
	efibootmgr	EFI boot entry manager
Backup / Restore	Cloning system disk	Backing-up and restoring data
Monitoring	qsctrl	Monitoring Quadrics network status
	ibstatus, ibstat	Monitoring InfiniBand networks
	IBS tool	Providing information about and configuring InfiniBand switches
	lsiocfg	Getting information about storage devices
	pingcheck	Checking devices power status
Debugging	Nsdoctor	Analyzing the reasons why a node has been excluded by RMS
	ibdoctor / ibtracert	InfiniBand network problem diagnosis/ Tracing communication paths in InfiniBand networks
	In-Target Probe	Kernel problems
	KDB	
	LKCD, lcrash	Capturing and analyzing crash dumps
	crash	Runtime debugging
	/proc	
	dump	Tracing
	GDB	
Testing	Nodechecking	Running tests on a node
	postbootchecker	Making verifications on nodes as they start

Table 2-1. Maintenance Tools

Chapter 3. Cluster Database Management

This chapter describes the architecture of the Cluster Database, the commands and the tools which enable the administrator to display and to change this Cluster Database.

The following topics are described:

- 3.1 *Architecture of ClusterDB*
- 3.2 *ClusterDB Administrator*
- 3.3 *Using Commands*
- 3.4 *Managing the ClusterDB*
- 3.5 *ClusterDB Modeling*

3.1 Architecture of ClusterDB

The Cluster database (**ClusterDB**) of the Bull HPC delivery contains the data that is required for the cluster management tools (**NS Master – HPC Edition, KSIS, pdsh, syslog-ng, ConMan, NsDoctor, etc.**). Compared with sequential configuration files, the advantages of using a database are flexibility and the distribution of the data to all the tools which ensure a better integration whilst at the same time not duplicating common data. Cluster database management uses the highly-scalable, SQL compliant, Open Source object-relational **PostgreSQL**.

The following figure shows an architecture of **ClusterDB** and its relationship to the cluster management tools.

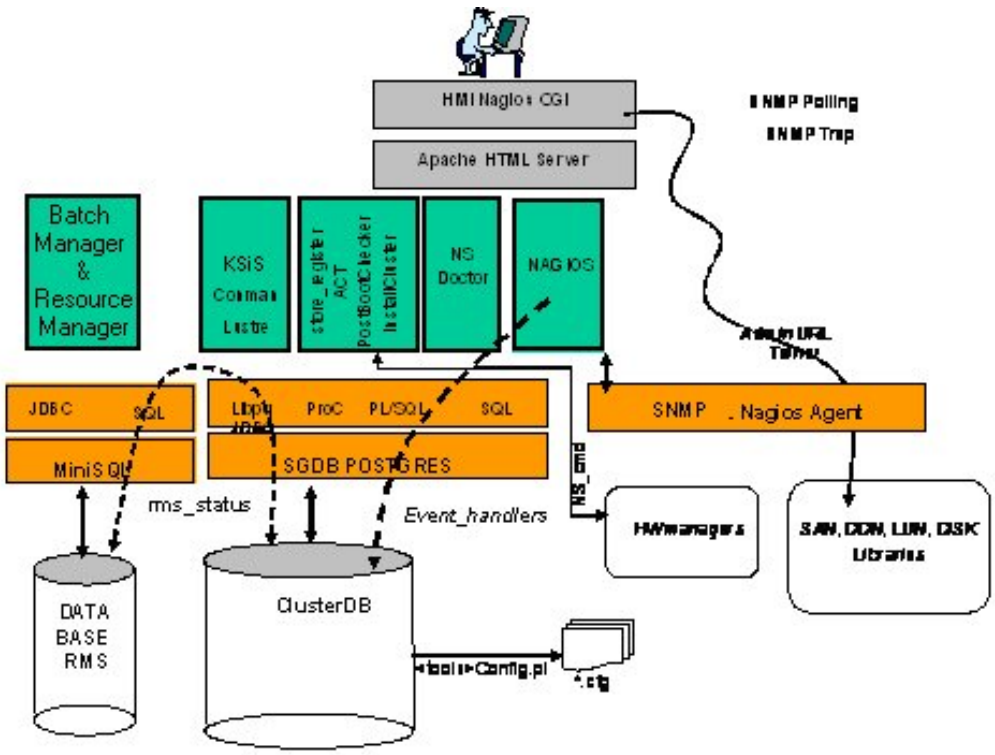


Figure 3-1. ClusterDB architecture

3.2 ClusterDB Administrator

ClusterDB is installed on the Management Node. The operations on **ClusterDB** must be performed from the Management Node.

The Database administrator is the **postgres** Linux user. This administrator is allowed to display and modify the **ClusterDB**, using the specific commands described in the next section. To manage the database (start, stop, save and restore), the administrator can use **PostgreSQL** tools (see 3.4 *Managing the ClusterDB*).

3.3 Using Commands

The administrators can consult or change the **ClusterDB** using the following commands:

- **changeOwnerProperties** changes the confidentiality parameters
- **dbmConfig** controls the consistency of the **ClusterDB** with the system. All database updates are marked to be a "candidate" for synchronization.
- **dbmCluster** operates on the whole cluster to get information, to check IP addresses and to check rack configuration.
- **dbmNode** displays information, or change attributes at the node level.
- **dbmHwManager** displays information, or change attributes at the Hwmanager level.
- **dbmGroup** manages the groups of nodes.
- **dbmEthernet** displays information, or change attributes for the Ethernet switches.
- **dbmIconnect** displays information, or change attributes for the interconnect switches.
- **dbmTalim** displays information, or change attributes for the remotely controlled power supply devices.
- **dbmSerial** displays information, or change attributes for the Portservers.
- **dbmFiberChannel** displays information about the Fiber Switches or changes the values of some attributes for a Fiber Switch or a subset of Fiber Switches
- **dbmServices** displays information about the Services or changes the values of some attributes for a Service
- **dbmDiskArray** displays information (for example **iproute**, **status**) and manages the disk array (status)

3.3.1 ChangeOwnerProperties

The cluster is handed over to the client with a name, a basename and IP address defined by Bull.

The IP address syntax used to identify equipment is of the form **A. V. U. H**.

V (the second byte) could be used for VLAN identification, **U** for Unit (Storage, Compute or Service) and **H** for Host (Host but also switch, disk subsystem or portserver).

The client may then want to change some of the attributes in keeping with their own security criteria.

These changes will in turn impact the **ClusterDB** Database, the network configuration for all the hosts, the configuration of storage bays and also the Lustre configuration (if installed).

Sometimes, the parameters which have been modified by the client may involve:

- Running **ECT** (Embedded Configuration Tool) for hosts with **Quadrics** Switches, **Ethernet** Switches, and **PAP** hosts.
- The network configuration of the nodes that will be done by **KSIS** at the time of the redeployment.
- Reconfiguring the DDN and FDA (Fibre Disk Array) subsystems to update them with the admin IP address and the gateway address.
- Manual operation of the **FDA**
- Running the **ddn_init** command on each **DDN** and for the reboot.
- Restarting the configuration of the Cluster Suite on I/O nodes, so that each node is aware of its peer node, using the correct names and IP addresses.
- The Lustre system is impacted if the node **basenames** are changed resulting in the obliteration of the file system followed by the creation of a new file system with the new data.
- If there is a change in the node **basenames** and of the admin IP address, the KSIS images are deleted from the database.

Consequently, when using this command, it is necessary to follow the process described below in order to reinitialize the system.

Syntax:

(This command is installed under /usr/lib/clustmngt/clusterdb/install)

changeOwnerProperties [--name <clustername>] [--basename <basename>]
[--adprivacy <bytes>] [--bkprivacy <bytes>]
[--bkgw <ip gateway>] [--bkdom <backbone domain>]
[--bkoffset <backbone Unit offset>]
[--dbname <database name] [--verbose]

Options:

--dbname	Specifies the name of the database to which the command applies. Default value: **clusterdb**. **Note:** This option must be used only by qualified people for debugging purposes.
--name	Specifies the name of the cluster. By default it is the basemane.
--basename	Specifies the basename of the node. (The node name is constituted of basename + netid). It is also the virtual node name.
--adprivacy	Privacy bytes. According to the admin netmask, one, two or three bytes can be changed. For example, if the admin netmask is 255.255.0.0, then **adprivacy** option can specify two bytes in the form **A.V**.
--bkprivacy	Privacy bytes. According to the backbone netmask, one, two or three bytes can be changed. For example, if the backbone netmask is 255.255.255.0, then **bkprivay** option can specify three bytes in the form **A.V.U**.
--bkgw	Specifies the backbone gateway
--bkdom	Specifies the backbone domain
--bkoffset	Specifies the backbone translation offset. It permits to translate the D.E.U.H backbone ip to D.E.(U + bkoffset).H

Examples:

- To change the basename and the byte A of the admin IP address enter:

```
changeOwnerProperties --basename nova --adprivacy 15
```

Process:

1. Following the change of parameters, the command **changeOwnerProperties** informs the Client that he has to run ECT on the Quadrics, Ethernet Switchs and the PAPs
2. Restart **dbmConfig**.
3. Manually configure on the FDA (if present).
4. Run **ddn_init** on each DDN and reboot (if DDN).
5. Cluster Suite: run **storedepha** (if HA).
6. Syslog: The DDN logs are archived with the base name on the IP address, rename and the log files updated (if DDN is present)
7. For a Lustre configuration if the basename is changed:
   a. Run **lustre_util stop**

b. Run **lustre_util remove**

c. Truncate the LUSTRE_OST, LUSTRE_MDT tables and use **storemodelctl generateost** and **storemodelctl generatemdt** to repopulate the tables with the new information.

d. Validate the recreated OSTs / MDTs: **lustre_investigate check**

e. Verify the Lustre models and regenerate the configuration file: **lustre_config**

f. Install new file systems: **lustre_util install**

3.3.2 dbmConfig

The **dbmConfig** command is used to maintain the consistency between the data in the **ClusterDB** and the different services and system files. The **dbmConfig** command shows the synchronization state or synchronizes different cluster services (**syshosts, sysdhcpd, conman, portserver, pdsh, nagios, snmptt, group, nsm**).

Syntax:

dbmConfig show [--service <name>] [--dbname <database name>] [--impact]

dbmConfig configure [--service <name> [--id <id> --timeout <timeout>] --restart --force --nodeps --impact] [--dbname <database name>]

dbmConfig help

Actions:

show	Displays the synchronization state of all the services or of a list of specified services.
configure	Runs the synchronization between the ClusterDB and all the services or a list of specified services. The configuration errors, if any, are listed on the console and in the **/var/log/synchro.log** file. It is necessary to check these messages. **Note:** The command reports an OK status to indicate that it has completed. This does not mean that no configuration error occurred.
help	Displays the syntax of the **dbmConfig** command.

Options:

--dbname	Specifies the name of the database to which the command applies. Default value: clusterdb. **Note:** This option must be used only by qualified people for debugging purposes.
--force	Reconfigures the service and restarts it.
--id	Reloads the configuration of the portserver identified by id. This option applies only to the portserver service (--service=portserver option).
--impact	Displays the configuration files and associated services impacted by the next dbmConfig configure command.
--nodeps	Forces the reconfiguration, despite the inter service dependencies.
--restart	Restarts the service instead of reloading it.

--service	Specifies the service from the following: syshosts, sysdhcpd, conman, portserver, pdsh, nagios, snmptt, group, nsm. For more information see Updated Configuration Files below.
--timeout	Specifies the timeout (in seconds) for restarting the portserver. This option applies only to the portserver service (--service=portserver option). Default value: 240.

Updated Configuration Files:

According to the specified service, the **dbmConfig configure --service** command updates a configuration file, as described below:

Service	Action
syshosts	Updates the **/etc/hosts** file with the data available in the administration base
sysdhcpd	Updates the **/etc/dhcpd.conf** file with the data available in the administration base.
conman	Updates the **/etc/conman.conf** file with the data available in the administration base.
portserver	Updates the portserver configuration file (/tftpboot/ps16*ConfigTS16 or /tftpboot/ps14*ConfigTS4), reloads the file on the appropriate portserver and reboots it.
pdsh	Updates the **/etc/genders** file with the data available in the administration base.
nagios	Updates several configuration files (**/etc/nagios/*.cfg**) with the data available in the administration base.
snmptt	Updates the **/etc/snmp/storage_hosts** file with the data available in the administration base.
group	Creates the predefined groups in the database. (No configuration file is updated.)
nsm	Updates the authentication file for the HW managers with the data available in the administration base.

If the administrator needs to modify these configuration files, for example, to add a machine that does not belong to the cluster, or to modify parameters, it is mandatory to use the template files created for this usage and to run the **dbmConfig** command again.

The templates files are identified by the **tpl** suffix. For example **/etc/hosts-tpl**, **/etc/dhcpd-tpl.conf**, **/etc/conman-tpl.conf**.

Examples:

- To configure the ConMan files, enter:

```
dbmConfig configure --service conman
```

- To list the synchronization state for Nagios, enter:

```
dbmConfig show --service nagios
```

3.3.3 dbmCluster

The **dbmCluster** command displays information about the whole cluster, or checks integrity and consistency of some elements of the ClusterDB.

Syntax:

dbmCluster show [- - dbname <database name>]

dbmCluster check ((- -ipaddr | - -rack) [- -verbose]) | - -unitCell [- -dbname <database name>]

dbmCluster --h | --help

Actions:

show	Displays the features of the cluster in terms of number of nodes and number of disks subsystems, as defined at the time of installation or update of the ClusterDB.
check	Checks integrity and consistency of some data of the ClusterDB: single IP addresses (**--ipaddr** option) or consistency of rack equipments (**--rack** option) or consistency of Unit Cell equipment (**--unitCell** option).
help	Displays the syntax of the **dbmCluster** command.

Options:

--dbname	Specifies the name of the database to which the command applies. Default: **clusterdb**. **Note:** this option must be used only by qualified people for debugging purposes.
--ipaddr	Checks that the IP addresses are distinct within the cluster.
--rack	Checks that the amount of equipment set for a rack in the database is not greater than the maximum. Checks also that there are not two sets of equipment on the same shelf.
--unitCell	Checks that the object Unit and Cell number are the same as the Ethernet switch connected to.

Examples:

- To check that each IP address is distinct, enter:

```
dbmCluster check --ipaddr
```

3.3.4 dbmNode

The **dbmNode** command displays information about the nodes (type, status, installed image etc.) or changes the values of some attributes for a node or a set of nodes (unit).

Syntax:

dbmNode show [--sysimage [--install_status={installed | not_installed | in_installed}]]

dbmNode show [--name <node name> --hwmanager | --cpu | --iproute | --serialroute]

dbmNode show [--unit <unit_num> --hwmanager | --cpu] [--dbname <database name>]

dbmNode set --name=<node name> --status={managed | not_managed} | --admin_macaddr <macaddr> | --backbone_macaddr <macaddr>

dbmNode set --unit <unit num> --status={managed | not_managed}

dbmNode set --nodelist=<node list> --status={managed | not_managed}

dbmNode set (--name=<node name> | --unit <unit num>) --cpu <total cpu chipset>

dbmNode set (--name=<node name> | --unit <unit num>) --hyperthreading={yes | no}

dbmNode set (--name=<node name> | --unit <unit num>) --cpu <total cpu chipset> --hyperthreading={yes | no} [--dbname <database name>]

dbmNode -h | --help

Actions:

show Displays type and status information for all the nodes or a set of nodes (--name option or --unit option). You can display the system images of nodes (using the --sysimage and --installed_status options), and the CPU or PAP features (using the --cpu and --hwmanager options).
The **Type** parameter specifies the node functions in the form `ACIMBNT`.
`A` means ADMIN
`C` means COMPUTE
`I` means I/O
`M` means META
`B` means INFINIBAND
`N` means NFS
`T` means TAPE
For example, the type for a compute node is displayed as "`-C-----`".

set Changes the value of some features for the specified node (--name option) or for all the nodes of the specified unit (--unit option) or for a set of nodes (--nodelist option).

Options:

--help Displays summary of options.

--admin_macaddr Specifies the MAC address of the eth0 interface connected to the administration network.

--**backbone_macaddr** Specifies the MAC address of the th1 interface connected to the backbone network.

--**cpu** Displays the CPU feature (model and number), or changes the number of CPUs.

--**install_status** Displays only the nodes that have the specified install status (installed, in_installed, not_installed).

--**name** Specifies the node name to which the action applies.

--**iproute** Displays the ethernet path (the localization and status of all Ethernet switches) between the node and the admin node

--**serialroute** Displays the serial path over portserver (the localization and status of all portservers) between the node and the admin node

--**hwmanager** Displays the name of the hwmanager that drives the node.

--**status** Changes the status (managed / not_managed). The "not_managed" status means that the node has not to be managed by the administration tools.

--**sysimage** Displays the nodes and the status of their system image.

--**unit** Specifies the unit to which the action applies.

--**hyperthreading** Changes the hyperthreading mode.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: this option must be used only by qualified people for debugging purposes.

Examples:

- To set the status of the `nova16` node to “up”, enter:

```
dbmNode set --name nova16 --status managed
```

- To change the MAC address of the `nova60` node, enter:

```
dbmNode set --name nova60 --admin_macaddr 00:91:E9:15:4D
```

- Below are various examples using the **dbmNode show** command:

```
dbmNode show
```

```
      Nodes names                                          Type      Status
-------------------------------------------------------------------------
      ns[0]                                                AC-M---   up
      ns[1-5,9-10]                                         -C-----   up
      ns[8]                                                -C-—B--   not_managed
      ns[6,11]                                             -CI----   down
      ns[7]                                                -CI----   up
      ns[12-13]                                            --I-B--   down
```

```
dbmNode show --sysimage
```

```
      Nodes names                        Type     Sys Image          Status
-------------------------------------------------------------------------------
      ns[4]                             -C-----   BAS4-16K           up
      ns[3]                             -C-----   BAS4-FAME          up
      ns[2,9]                           -C-----   ONEDISK            up
      ns[8]                             -C—-B--   ONEDISK            up
      ns[1,5,10]                        -C-----   NULL               up
      ns[6,11]                          -CI----   NULL               down
      ns[7]                             -CI----   NULL               up
      ns[12-13]                         --I-B--   NULL               down
```

```
dbmNode show --sysimage --install_status installed
```

```
      Nodes names                        Type      Sys Image          Status
-------------------------------------------------------------------------------
      ns[4]                             -C-----     BAS4-16K           up
      ns[3]                             -C-----     BAS4-FAME          up
      ns[2,9]                           -C-----     ONEDISK            up
      ns[8]                             -C-—B--     ONEDISK            up
```

```
dbmNode show --name ns0 --cpu
```

```
Name          Cpu model       Cpu total    Cpu available   Hyper threading
-------------------------------------------------------------------------
ns0        UNDEF           8            0               0
```

3.3.5 dbmHwManager

The **dbmHwManager** command displays information or change status at the level of the HW Manager.

Syntax:

dbmHwManager show	[--name <hwmanager name> --node | --status | --iproute]
dbmHwManager show	[--unit <unit num> --status] [--dbname <database name>]
dbmHwManager set	--name <hwmanager name> --status ={managed | not_managed}]
dbmHwManager set	--unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]
dbmHwManager	-h | --help

Actions:

show	Displays model, type and status information for all the hwmanagers or a subset of hwmanager (--unit option).
set	Changes the value of some features for the specified hwmanager (--name option) or for all the hwmanagers of the specified unit (--unit option).

Options:

--**help**	Displays summary of options.
--**name**	Specifies the hwmanager name to which the action applies.
--**iproute**	Displays the Ethernet path (the localization and status of all Ethernet switches) between the hwmanager and the admin node
--**node**	Displays the name of the nodes managed by the hwmanager.
--**status**	Changes the status (managed/not_managed). The "not_managed" status means that the hwmanager has not to be managed by the administration tools.
--**unit**	Specifies the unit to which the action applies.
--**dbname**	Specifies the name of the database on which the command is applied. Default: clusterdb. **Note:** This option must be used only by qualified people for debugging purposes.

Examples:

- To change the status of the PAP named `pap1` to “UP”, enter:

```
dbmHwManager set --name pap1 --status managed
```

3.3.6 dbmGroup

The **dbmGroup** command lets the administrator of the ClusterDB show or modify (add, delete, create) the organization of the groups of nodes.

Note:

The groups are using commands like pdsh, KSiS, to perform actions on a set of nodes.

Syntax:

dbmGroup

dbmGroup show	[--dbname <database name>]
dbmGroup add	--name <group name> --nodelist <node list> [--comment <description>] [--dbname <database name>]
dbmGroup del	--name <group name> | --all [--dbname <database name>]
dbmGroup modify	--name <group name> (--addnodelist <node list> | --delnodelist <node list>) [--dbname <database name>]
dbmGroup create	[--dbname <database name>]
dbmGroup	-h | --help

Actions:

show	Displays the group of nodes.
add	Adds a group to the existing ones.
del	Deletes one group or all groups.
modify	Adds or deletes a list of node in an existing group.
create	Recreates the predefined groups (criterion groups), in the case they have been deleted.

Options:

--**help**	Displays summary of options.
--**name**	Specifies the group name.
--**nodelist**	List of the netid for the nodes of the group, in the form [x,y-w].
--**comment**	Description of the group.
--**all**	Deletes all nodes.
--**addnodelist**	Adds a node list in an existing group.

--**delnodelist** Deletes a node list in an existing group.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: this option must be used only by qualified people for debugging purposes.

Predefined Groups:

Once the cluster is configured, some predefined groups are automatically created, depending on the node types defined in the ClusterDB.

The **dbmGroup show** command displays the groups and a short explanation for each one.

Note:
A group can be mono-type, or multi-type for the nodes which combine several functions. Seven mono-type groups can be defined: ADMIN, COMPUTE (or COMP), IO, META, IBA, NFS, TAPE. See below examples of mono-type and multi_type groups.

Example of Predefined Groups:

In the following example four sorts of groups are defined:

- One Group of all the nodes except the nodes whose type is `ADMIN`. This group is named `ALL`.
- The group nodes per type. For instance:

`ADMIN`	Group of all the nodes whose type is ADMIN (mono-type).
`ADMINCOMPMETA`	Group of all the nodes whose type is ADMIN, compute or IO (multi-type).
`COMPIBA`	Group of all the nodes whose type is compute and Infiniband (multi-type).
`COMPIO`	Group of all the nodes whose type is compute or IO (multi-type).
`COMPUTE`	Group of all the nodes whose type is compute (mono-type).
`IO`	Group of all the nodes whose type is IO (mono-type).
`IOIBA`	Group of all the nodes whose type is IO and Infiniband (multi-type).
`META`	Group of all the nodes whose type is METADATA (mono-type).

- The groups of COMPUTE nodes for each memory size. For instance:

`COMP48GB`	Group of all the nodes whose type is **compute** and with 48GBs of memory (mono-type).
`COMP128GB`	Group of all the nodes whose type is **compute** and with 128GB of memory (mono-type).

- The groups of nodes for each memory size. For instance:

NODES16GB	Group of all the nodes with 16GBs of memory.
NODES48GB	Group of all the nodes with 48GBs of memory.
NODES64GB	Group of all the nodes with 64GBs of memory.
NODES128GB	Group of all the nodes with 128GBs of memory.

Examples:

- To display all the groups defined in the ClusterDB, enter:

```
dbmGroup show
```

```
Group Name  Description                 Nodes Name
-------------------------------------------------------
ADMIN       Nodes by type:ADMIN         ns0
ALL         All nodes except node admin ns[4-5,8-10]
COMP        Nodes by type:COMP          ns[4,8]
COMP128GB   COMPUTE node with 128GB     ns8
COMP48GB    COMPUTE node with 48GB      ns4
IO          Nodes by type:IO            ns10
META        Nodes by type:META          ns[5,9]
NODES128GB  Nodes by memory size:128GB  ns8
NODES48GB   Nodes by memory size:48GB   ns[4,10]
NODES64GB   Nodes by memory size:64GB   ns[0,5,9]
```

- To add a new group, named GRAPH, which includes the nodes 1 and 4, 5, 6 (netid) into the database, enter:

```
dbmGroup add --name GRAPH --nodelist [1,4-6] --comment 'Graphic Group'
```

- To delete the GRAPH group from the database, enter:

```
dbmGroup del --name GRAPH
```

- To re-create the predefined groups if they have been deleted, enter:

```
dbmGroup create
```

```
=>
Create ALL                          [ OK ]
Create NODES4GB                     [ OK ]
Create NODES16GB                    [ OK ]
Create ADMIN                        [ OK ]
Create INFNFS                       [ OK ]
Create INFTAPE                      [ OK ]
Create IOINF                        [ OK ]
Create METAINF                      [ OK ]
```

3.3.7 dbmEthernet

The **dbmEthernet** command displays or change attributes for the Ethernet switches.

Syntax:

dbmEthernet show [--nw ={admin | backbone}]

dbmEthernet show [--name <switch name> [--status | --macaddr | --iproute | --linkhost]]

dbmEthernet show [--unit <unit num> [--status]] [--dbname <database name>]

dbmEthernet set --name <switch name> --status ={managed | not_managed} | --macaddr <macaddr>]

dbmEthernet set --unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]

dbmEthernet -h | --help

Actions:

show Displays name, networtk, ip address, Mac address and status information for all the switches or a subset of switches (--unit option).

set Changes the value of some features for a specified switch (--name option) or for all the switchs of the specified unit (--unit option).

Options:

--help Displays summary of options.

--name Specifies the switch name to which the action applies.

--nw Displays information about the given network type.

--iproute Displays the ethernet path (the localization and status of all ethernet switches) between the switch and the admin node.

--macaddr Changes the Macaddr of the Ethernet Switch.

--status Changes the status (managed / not_managed). The "not_managed" status means that the switch has not to be managed by the administration tools.

--unit Specifies the unit to which the action applies.

--dbname Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To display the features of the administration network, enter:

```
dbmEthernet show --nw admin
```

- To change the mac address of the Ethernet switch named `eswu1c2` to the value `00:91:E9:15:4D`, enter:

```
dbmEthernet set --name eswu1c2 --admin_macaddr 00:91:E9:15:4D
```

3.3.8 dbmIconnect

The **dbmIconnect** command displays or change attributes for the interconnect switches.

Syntax:

dbmIconnect show [--nw ={QsNet | InfiniBand}]

dbmIconnect show [--name <switch name> [--status | --iproute] |--linkhost]

dbmIconnect show [--unit <unit num> [--status]] [--dbname <database name>]

dbmIconnect set --name <switch name> --status ={managed | not_managed}

dbmIconnect set --unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]

dbmIconnect -h | --help

Actions:

show Displays name, networtk, admin and standby ip addresses, and status information for all the switches or a subset of switches (--unit option).

set Changes the value of some features for a specified switch (--name option) or for all the switches of the specified unit (--unit option).

Options:

--**help** Displays summary of options.

--**name** Specifies the switch name to which the action applies.

--**nw** Displays information about the given network type.

--**iproute** Displays the ethernet path (the localization and status of all ethernet switches) between the InterConnect switch and the admin node.

--**linkhost** Displays hosts plugged on a given interconnect switch.

--**status** Changes the status (managed / not_managed). The "not_managed" status means that the switch has not to be managed by the administration tools.

--**unit** Specifies the unit to which the action applies.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To display the features of the QsNet interconnect, enter:

```
dbmIconnect show --nw QsNet
```

- To change the status of the interconnect switch named `QR0N01` to the value `not_managed`, enter:

```
dbmIconnect set --name QR0N01 --status not_managed
```

3.3.9 dbmTalim

The **dbmTalim** command displays or change attributes for remotely controlled power supply devices.

Note:

Talim refers to remotely controlled power supply devices which are used to start and stop equipment.

Syntax:

dbmTalim show [--name <talim name> [--status | --macaddr | --iproute]]

dbmTalim show [--unit <unit num> [--status]] [--dbname <database name>]

dbmTalim set --name <talim name> --status ={managed | not_managed} | --macaddr <macaddr>

dbmTalim set --unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]

dbmTalim -h | --help

Actions:

show	Displays name, networtk, ip address, Mac address and status information for all the talim or a subset of talim (--unit option).
set	Changes the value of some features for a specified talim (--name option) or for all the talim of the specified unit (--unit option).

Options:

--**help**	Displays summary of options.
--**name**	Specifies the talim name to which the action applies.
--**iproute**	Displays the ethernet path (the localization and status of all ethernet switches) between the talim and the admin node
--**macaddr**	Displays the Macaddr or changes the Macaddr of the Talim.
--**status**	Displays the status or changes the status (managed / not_managed). The "not_managed" status means that the talim has not to be managed by the administration tools.
--**unit**	Specifies the unit to which the action applies.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To display the features of the talim named talim2, enter:

```
dbmTalim show --name talim2
```

- To change the mac address of the talim named `talim2` to the value `00:91:E9:15:4D`, enter:

```
dbmTalim set --name talim2 --macaddr 00:91:E9:15:4D
```

3.3.10 dbmSerial

Note:
The **dbmSerial** depends on the cluster's configuration and only applies to clusters which include a portserver.

The **dbmSerial** command displays or change attributes for the Portservers.

Syntax:

dbmSerial show [--nw ={node|pap|storage|mixed}]

dbmSerial show [--name <portserver name> [--status | --macaddr | --iproute | --linkhost]]

dbmSerial show [--unit <unit num> [--status]] [--dbname <database name>]

dbmSerial set --name <portserver name> --status ={managed | not_managed} | --macaddr <macaddr>]

dbmSerial set --unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]

dbmSerial -h | --help

Actions:

show	Displays name, networtk, ip address, Mac address and status information for all the Portserver or a subset of portserver (--unit option).
set	Changes the value of some features for a specified Portserver (--name option) or for all the Portserver of the specified unit (--unit option).

Options:

--**help**	Displays summary of options.
--**nw**	Displays information about the given network type.
--**name**	Specifies the Portserver name to which the action applies.
--**iproute**	Displays the Ethernet path (the localization and status of all ethernet switches) between the Portserver and the admin node.
--**status**	Displays the status or changes the status (managed / not_managed). The "not_managed" status means that the Portserver has not to be managed by the administration tools.
--**macaddr**	Display/changes the Mac address of the Portserver.
--**linkhost**	Displays hosts plugged on a given portserver.
--**unit**	Specifies the unit to which the action applies.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To display the features of all portservers, enter:

```
dbmSerial show
```

- To display the list of the hosts plugged on the portserver named ps16u1c0, enter:

```
dbmSerial show --name ps16u1c0 --linkhost
```

- To change the status of the portserver named ps16u1C0 , enter:

```
dbmSerial set --name ps16u1C0 --status managed
```

- To change the status of all portservers affiliated with unit 0, enter:

```
dbmSerial set --unit 0 --status not_managed
```

3.3.11 dbmFiberChannel

Displays the Database information about the Fiber Switches or changes the values of some attributes for a Fiber Switch or a subset of Fiber.

Syntax:

dbmFiberChannel show	[--nw]
dbmFiberChannel show	[--name <switch name> [--status | --iproute]]
dbmFiberChannel show	[--unit <unit num> [--status]] [--dbname <database name>]
dbmFiberChannel set	--name <switch name> --status ={managed | not_managed}
dbmFiberChannel set	--unit <unit_num> --status ={managed | not_managed} [--dbname <database name>]
dbmFiberChannel	-h | --help

Actions:

show	Displays name, networtk, admin ip address, and status information for all the switches or a subset of switches (--unit option).
set	Changes the value of some features for a specified switch (--name option) or for all the switches of the specified unit (--unit option).

Options:

--help	Displays summary of options.
--name	Specifies the switch name to which the action applies.
--nw	Displays information about all network type.
--iproute	Displays the ethernet path (the localization and status of all ethernet switches) between the Fiber switch and the admin node.
--status	Changes the status (managed / not_managed). The "not_managed" status means that the switch has not to be managed by the administration tools.
--unit	Specifies the unit to which the action applies.
--dbname	Specifies the name of the database on which the command is applied. Default: clusterdb. **Note:** This option must be used only by qualified people for debugging purposes.

Examples:

- To change the FC switch named `fcswu0c1` to up, enter:

```
dbmFiberChannel set --name fcswu0c1 --status managed
```

- To show the hierarchy iproute of the FC switch through Ethernet switches, enter:

```
dbmFiberChannel show --name fcswu0c1 --iproute
```

- To show information about FC switch, enter:

```
dbmFiberChannel show
```

3.3.12 dbmServices

Displays the Database information about the Services or changes the values of some attributes for a Service.

Syntax:

dbmServices show --objectlist

dbmServices show --object <object name> [--name <service name>]
[--dbname <database name>]

dbmServices set --object <object name> --name <service name> (--enable | --disable)
[--dbname <database name>]

dbmServices -h | --help

Actions:

show Displays the list of all the objects contained in Services table (--**objectlist** option). Or displays name, object type and if service is enabled or disabled (--**object** --**name** options).

set Changes the value of the **actif** field (enable or disable) for a specified service (**--object --name** options).

Options:

--**help** Displays summary of options.

--**objectlist** Displays the list of all the objects contained in Services table.

--**object** Specifies the object type of service to which the action applies.

--**name** Specifies the service name to which the action applies.

--**enable** Specifies that the service must be activated.

--**disable** Specifies that the service must be de-activated.

--**dbname** Specifies the name of the database on which the command is applied. Default: clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To print details on the service named “Ethernet interfaces” on object node, enter:

```
dbmServices show –-object node –-name “Ethernet interfaces”
```

- To change the service named “Ethernet interfaces” on object node to up, enter:

```
dbmServices set –-object node –-name “Ethernet interfaces” --enable
```

3.3.13 dbmDiskArray

dbmDiskArray displays information (for example **iproute**, **status**) and manages the disk array (status)

Syntax:

dbmDiskArray show [--name <diskarray name> --iproute | --serialroute] [--dbname <database name>]

dbmDiskArray set --name < diskarray name> --status={managed | not_managed} [--dbname <database name>]

dbmDiskArray -h | --help

Actions:

show Displays the type and status information for all the disk arrays or for a specified one (--name option).

set Changes the value of some of the features for a specified disk array (--name option).

Options:

--**help** Displays a summary of options.

--**name** Specifies the disk array name to which the action applies.

--iproute Displays the Ethernet path (including the location and status of all Ethernet switches) between the disk array and the Management Node.

--serialroute Displays the serial path which includes a portserver (the location and status of all portservers) between the disk array and the Management Node.

Note: The --serialroute option depends on the cluster’s configuration and only applies to clusters which include a portserver.

--status Changes the status (**managed**/ **not_managed**). The **not_managed** status means that the disk array will not be managed by the administration tools.

--**dbname** Specifies the name of the database to which the command is applied. Default = clusterdb.
Note: This option must be used only by qualified people for debugging purposes.

Examples:

- To print details of the disk array named **da0** using Ethernet switches, enter:

```
dbmDiskArray show --name da0 –iproute
```

- To change the status of the disk array named **da0** to up, enter:

```
dbmDiskArray set --name da0 –status managed
```

3.4 Managing the ClusterDB

The administrator of the **ClusterDB** must guarantee and maintain the consistency of the data. To view and administrate the database, the ClusterDB administrator can use the following PostgreSQL tools:

- The **PostgreSQL commands**.
  The **psql** command enables the PostgreSQL editor to run. You can run it as follows:

```
psql -U clusterdb clusterdb
```

- The **phpPgAdmin Web interface**.
  You can start it with an URL similar to the following one (`admin0` is the name of the Management Node):

```
http://admin0/phpPgAdmin/
```

Important:

These tools, which let the administrator update the **ClusterDB**, must be used carefully since incorrect usage could break the consistency of the **ClusterDB**.

For more information, refer to the **PostgreSQL** documentation delivered with the product.

3.4.1 Saving and Restoring the Database

The database administrator is responsible for saving and restoring the ClusterDB.

The administrator will use the **pg_dump** and **pg_restore** PostgreSQL commands to save and restore the database.

3.4.1.1 Saving the Database (pg_dump)

The **pg_dump** command has a lot of options. To display all the options, enter:

```
pg_dump --help
```

Note:

The **pg_dump** command can run while the system is running.

Saving the Metadata and the Data:

It is recommended that the following command is used:

```
pg_dump -Fc -C -f /var/lib/pgsql/backups/clusterdball.dmp clusterdb
```

Saving the Data only:

It is recommended that the following command is used:

```
pg_dump -Fc -a -f /var/lib/pgsql/backups/clusterdbdata.dmp clusterdb
```

Saving Data each Day

When the **clusterdb** rpm is installed, a **cron** is initialized to save the ClusterDB daily, at midnight. The data is saved in the **clusterdball[0-6].dmp** and **clusterdata[0-6].dmp** (**0-6** is the number of the day) in the **/var/lib/pgsql/backups** directory. This **cron** runs the **make_backup.sh** script, located in the directory **/usr/lib/clustmngt/clusterdb/install/**.

3.4.1.2 Restoring the Database (pg_restore)

The **pg_restore** command has a lot of options. To display all the options, enter:

```
pg_restore --help
```

Restoring the whole ClusterDB:

Requirement: ClusterDB does not exist anymore.

To list the existing databases, use the **oid2name** command:

```
oid2name
```

If you need to remove an inconsistent **ClusterDB**, enter:

```
dropdb clusterdb
```

When you are sure that the **ClusterDB** does not exist anymore, enter the following command to restore the whole database:

```
pg_restore -Fc --disable-triggers -C -d template1
/var/lib/pgsql/backups/clusterdball.dmp
```

Restoring the ClusterDB Data:

Requirement: ClusterDB must exist and be empty.

To create an empty ClusterDB, run these commands:

```
/usr/lib/clustmngt/clusterdb/install/create_clusterdb.sh -nouser
psql -U clusterdb clusterdb
clusterdb=> truncate config_candidate;
clusterdb=> truncate config_status;
clusterdb=> \q
```

To restore the data, enter:

```
pg_restore -Fc --disable-triggers -d clusterdb /var/lib/pgsql/backups/clusterdbdata.dmp
```

3.4.2 Starting and Stopping PostgreSQL

Starting and stopping **postgreSQL** is performed using the **service** Linux command. **postgreSQL** is configured to be launched at levels 3, 4 and 5 for each reboot.

Note:

Both **root** user and **postgres** user can start and stop PostgreSQL. However it is recommended to use always the **postgres** login.

To start **postgreSQL**, run the following script:

```
/sbin/service postgresql start
```

To stop **postgreSQL**, run the following script:

```
/sbin/service postgresql stop
```

3.4.3 Viewing the PostgreSQL Alert Log

The **postgreSQL** log file is **/var/log/postgres/pgsql**. This is read to view any errors, which may exist.

Note:

This file can increase in size very quickly. It is up to the database administrator to rotate this file when **postgreSQL** is stopped.

3.5 ClusterDB Modeling

3.5.1 Physical View of the Cluster Networks

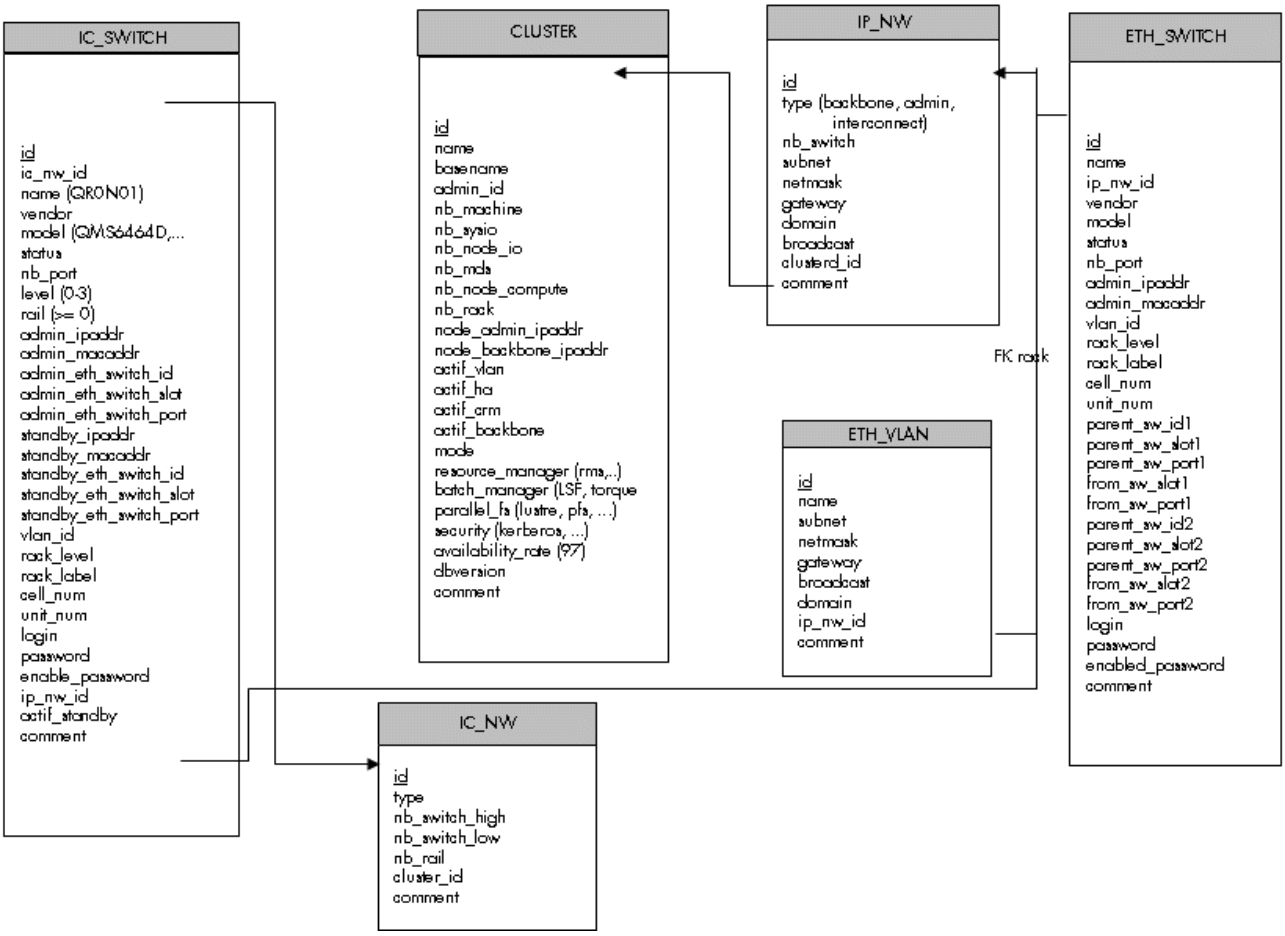


Figure 3-2. Cluster Network – diagram 1

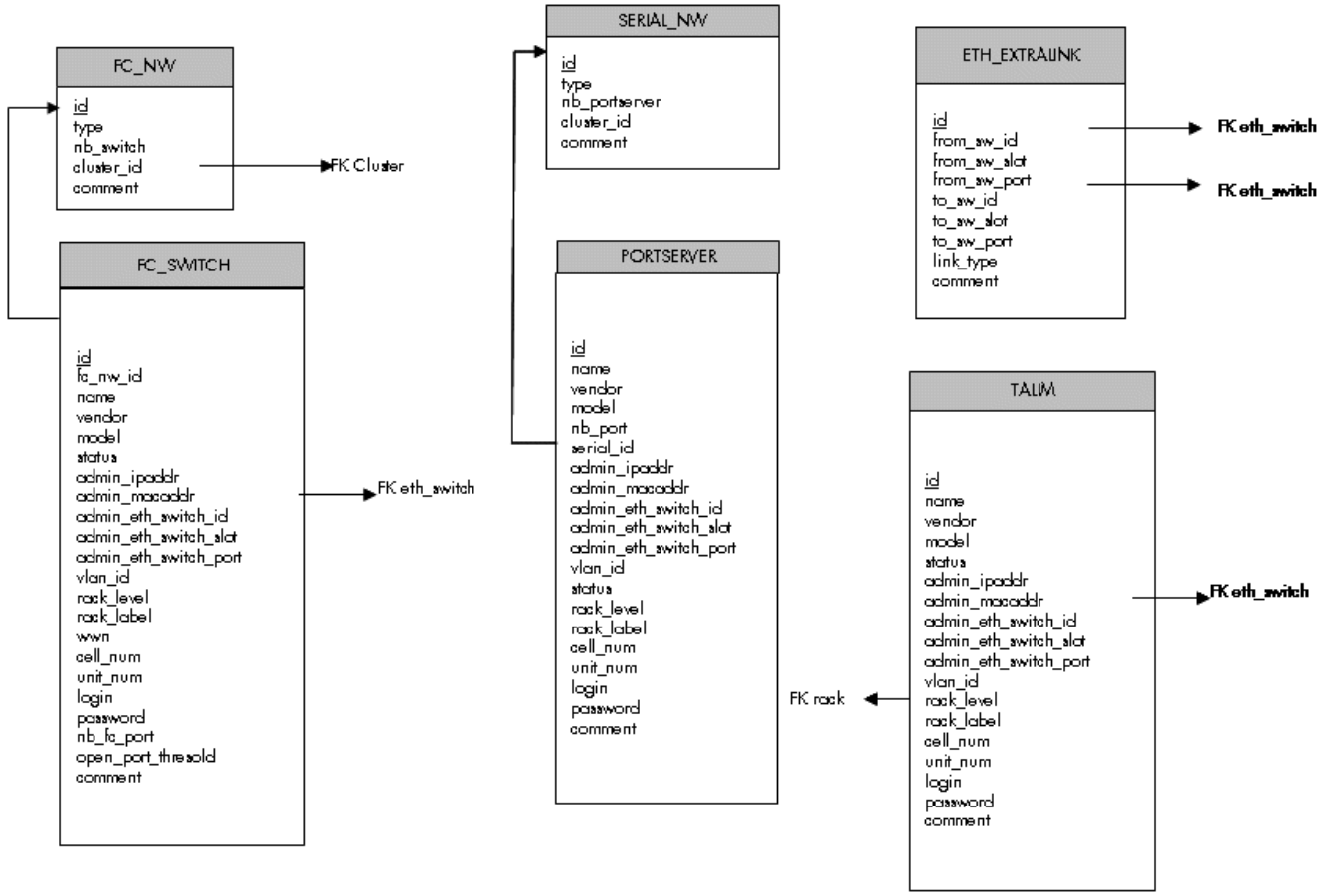


Figure 3-3. Cluster Network – diagram 2

3.5.1.1 CLUSTER Table

Column name	Description	Example	Fill in method
id	PK	540	preload - sequence
name	Name of the cluster	molecular	Preload & loadClusterdb
basename	Node basename	node	Preload & loadClusterdb
admin_id	FK table User		preload
nb_machine	Number of Nodes	601	preload – reconfigClusterdb
nb_sysio	Number of disk sub systems	56	preload – reconfigClusterdb
nb_node_io	Number of IO Nodes	54	preload – reconfigClusterdb
nb_mds	Number of MDS	2	preload – reconfigClusterdb
nb_node_compute	Number of Compute Nodes	544	preload – reconfigClusterdb
nb_rack	Number of rack	270	preload – reconfigClusterdb
node_admin_ipaddr	Virtual IP address of the Management node for the backbone network	10.1.0.65	preload
node_backbone_ipaddr	Virtual IP address of the Management node		preload
actif_vlan	Boolean on the VLAN configuration	true	preload
actif_ha	Boolean High Availability	true	Cluster Suite

Column name	Description	Example	Fill in method
actif_crm	CRM Boolean surveillance	true	preload
actif_backbone	Boolean, Use of a backbone	true	DV-true
mode	Mode 100%, 92% or 8%	100	preload – reconfigClusterdb
resource_manager	RMS or SLURM	rms	preload
batch_manager	LSF or TORQUE	torque	preload
parallel_fs	Lustre	lustre	prelaod
security	Kerberos	NULL	preload
availability_rate	Availability rate	20.3.3	preload
dbversion	Development model version for the database	16.2	Creation
comment	Free field		NULL

Table 3-1. Cluster Table

3.5.1.2 IP_NW Table

Column name	Description	Example	Fill in method
id	PK	4	preload – Sequence
type	backbone, admin	backbone	preload
nb_switch	Number of switches	10	preload
subnet	Sub-network	10.0.0.0	preload
netmask	Sub-network mask	255.255.0.0	preload
gateway	IP address of the gateway	10.0.255.254	preload
domain	Name of the domain	frec.bull.fr	preload
broadcast	IP address of the broadcast	NULL	NULL
cluster_id	FK on the CLUSTER		preload
comment	Free field		NULL

Table 3-2. IP_NW table

3.5.1.3 ETH_SWITCH Table

Column name	Description	Example	Fill in method
id	PK		preload-Sequence
name	Name of the switch		preload
ip_nw_id	FK on IP_NW		preload
vendor	Vendor	CISCO	preload
model	Modele of the SW	CISCO6509	preload
status	Nagios host_status	up	DV = up - Nagios
nb_port	Total number of port		preload
admin_ipaddr	Admin IP address of the Ethernet switch		preload
admin_macaddr	Mac Address of the Switch		ACT

Column name	Description	Example	Fill in method
vlan_id	FK on ETH_VLAN		preload
rack_level	Superposition level in the rack		preload
rack_label	Name of the rack		preload
cell_num	Name of the cell		preload
unit_num	Number of the Unit		preload
parent_sw_id1	Ethernet switch 1st parent		preload
parent_sw_slot1	Arrival slot number of the 1st parent switch	0	preload
parent_sw_port1	Connection port for the 1st switch	1	preload
from_sw_slot1	Starting slot number of the 1st switch	0	preload
from_sw_port1	Starting port number of the 1st switch	1	preload
parent_sw_id2	Ethernet switch 2nd parent		preload
from_sw_slot2	Starting slot number of the 2nd switch		preload
parent_sw_port2	Starting port number for the 2nd switch	2	preload
from_sw_slot2	Starting slot number of the 2nd switch		preload
from_sw_port2	Starting port number of the 2nd switch		preload
login	Administration login		cmdExpl
password	Administration password		cmdExpl
enabled_password	Cisco enabled password		ECT

Table 3-3. ETH_SWITCH Table

3.5.1.4 IC_NW Table

Column name	Description	Example	Fill in method
Id	PK		preload - Sequence
type	QSNet, Infiniband	QSNet	preload
nb_switch_high	Number of high switches	12	preload - reconfigClusterdb
nb_switch_low	Number of low switches	33	preload - reconfigClusterdb
nb_rail	Number of rails	3	preload
cluster_id	FK on the CLUSTER		preload
comment	Free field		

Table 3-4. IC_NW Table

3.5.1.5 IC_SWITCH Table

Column name	Description	Example	Fill in method
Id	PK		preload - Sequence
ic_nw_id	FK on the IC_NW		preload
name	Name of the Switch Interconnect	QR0N01	preload
vendor	Name of the Vendor	QUADRICS	preload
model	Model of the Switch	QMS6464D	preload

Column name	Description	Example	Fill in method
status	Nagios host_status	up	DV = up – Nagios
nb_port	Port number	64	preload
level	Level of the switch	1 – 2	preload
rail	Number of the rails	2	preload
admin_ipaddr	Administration IP address		preload
admin_macaddr	Mac Address of the switch	unused	NULL
admin_eth_switch_id	FK on ETH_SWITCH		preload
admin_eth_switch_slot	Arrival slot number on ETH_SW		preload
admin_eth_switch_port	Connection port on the ETH_SW	5	preload
standby_ipaddr	IP address of the standby controller		preload
standby_macaddr	Mac Address of the controller	unused	NULL
standby_eth_switch_id	FK on the ETH_SWITCH		preload
stanby_eth_switch_slot	Arrival slot number on ETH_SW		preload
standby_eth_switch_port	Connection port on the ETH_SW	6	preload
vlan_id	FK on the ETH_SWITCH		preload
rack_level	Level of superposition in the rack	G	preload
rack_label	Name of the rack	C0-A16	preload
cell_num	Number of the cell	1	preload
unit_num	Number of the Unit	0	preload
Login	Administration login	unused	preload or DV
password	Administration Password	unused	preload or DV
enable_password	Password enable		preload or DV
ip_nw_id	Foreign key on the IP_NW		preload
actif_standby	Configuration of a standby IP address		DV
comment	Free field		

Table 3-5. IC_Switch Table

3.5.1.6 SERIAL_NW Table

Column name	Description	Example	Fill in method
Id	PK	1	preload – sequence
rype	PAP Network, Node, Storage, Mixed	node	preload
nb_portserver	Number of PS	39	preload
cluster_id	FK on the CLUSTER		preload
comment	Free field		

Table 3-6. Serial_NW Table

3.5.1.7 PORTSERVER Table

Column name	Description	Example	Fill in method
id	Primary key		preload - sequence
name	Name of the portserver	ps16u1c0	preload
vendor	Name of vendor	DIGI	preload
model	Model of the PS	TS16	preload
nb_port	Total number of TTY/PS ports	16	preload
serial_id	FK on SERIAL_NW		preload
admin_ipaddr	Administration IP address		preload
admin_macaddr	Mac address of the PS		ACT
admin_eth_switch_id	FK on ETH_SWITCH		preload
admin_eth_switch_slot	Arrival slot number on ETH_SW		preload
admin_eth_switch_port	Connection port on the ETH_SW	10	preload
vlan_id	FK on the ETH_VLAN	40	preload
status	Nagios host_status	down	DV = up – Nagios
rack_level	Height of U in the rack		preload
rack_label	Name of the rack		preload
cell_num	Number of the cell		preload
unit_num	Number of the Unit		preload
login	Administration login		preload
password	Administration password		preload
comment	Free field		

Table 3-7. PORTSERVER Table

3.5.1.8 ETH_VLAN Table

Column name	Description	Example	Fill in method
id	PK	1	preload - sequence
name	Name of the VLAN	pad	preload
subnet	Sub-network IP address	10.4.0.0	preload
netmask	Netmask of the sub-network	255.255.0.0	preload
gateway	IP address of the gateway	10.4.255.254	preload
broadcast	IP address of the broadcast	10.4.255.255	preload
domain	Name of the domain	unused	preload – NULL
ip_nw_id	FW on the IP_NW		preload
comment	Free field		

Table 3-8. ETH_VLAN table

3.5.1.9 FC_NW Table

Note:
This table only applies to systems which include a Storage Area Network (SAN).

Column name	Description	Example	Fill in method
id	PK	1	preload - sequence
type	Role of the network	SAN-META	preload
nb_switch	Number of switches	39	preload
cluster_id	FK on the CLUSTER		preload
comment	Free field		

Table 3-9. FC_NW table

3.5.1.10 FC_SWITCH Table

Note:
This table only applies to systems which include a Storage Area Network (SAN).

Column name	Description	Example	Fill in method
id	PK		preload-Sequence
name	Name of the switch		preload
fc_nw_id	FK on the FC_NW		preload
vendor	Name of the vendor	BROCADE	preload
model	SW model	Silkworm 200E	preload
status	Nagios host_status	up	DV = up - Nagios
admin_ipaddr	IP admin address on the fibre switch channel		preload
admin_macaddr	Mac Address of the Switch		ACT
admin_eth_switch_id	FK on the ETH_SWITCH		preload
admin_eth_switch_slot	Arrival slot number on ETH SW		preload
admin_eth_switch_port	Connection on the ETH SW	3	preload
vlan_id	FK on the ETH_VLAN		preload
rack_level	Superposition level in the rack		preload
rack_label	Name of the rack		preload
cell_num	Number of the cell		preload
unit_num	Number of the unit		preload
login	Administration login		preload

Column name	Description	Example	Fill in method
password	Administration Password		prelaod
nb_fc_port	Number of fibre channel ports		preload
open_port_threshold			preload
comment	Free field		

Table 3-10. FC_SWITCH table

3.5.1.11 TALIM Table

Column name	Description	Example	Fill-in method
id	PK		preload-Sequence
name	Name of the power switch		preload
vendor	Vendor name	Blackbox	preload
model	Model of the power switch		preload
status	Nagios host_status	up	DV = up - Nagios
admin_ipaddr	Admin IP address of the power switch		preload
admin_macaddr	Mac Address of the power switch		ACT
admin_eth_switch_id	FK on the ETH_SWITCH		preload
admin_eth_switch_slot	Arrival slot number on ETH SW		preload
admin_eth_switch_port	Connection port on the ETH SW	3	preload
vlan_id	FK on the ETH_VLAN		preload
rack_level	Superposition level in the rack		preload
rack_label	Name of the rack		preload
cell_num	Cell number		preload
unit_num	Unit number		preload
login	Administration login		preload
password	Administration password		prelaod
comment	Free field		

Table 3-11. TALIM table

3.5.1.12 ETH_EXTRALINK Table

This table is not active in this version.

3.5.2 Physical View of the Storage

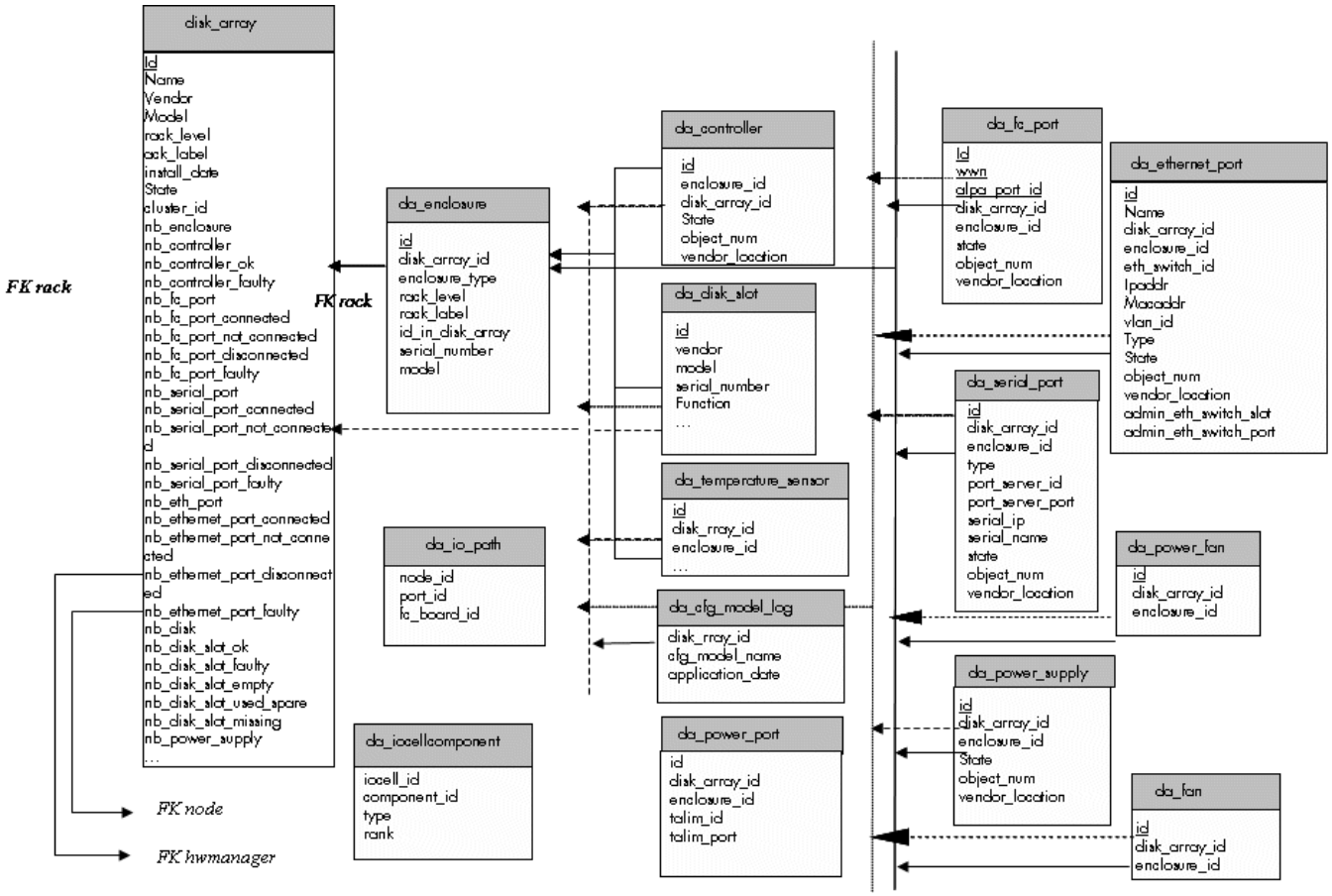


Figure 3-4. Storage physical view

3.5.2.1 disk_array Table

Field name	Field information	Fill in method
id	Unique identifier for the array in the database	preload - sequence
name	Name of the array (used for host in nagios)	preload
vendor	Vendor name: DDN, NEC, etc.	preload
model	Model name : S2A8500, FDA2300 …	preload
rack_level	Location in the rack	preload
rack_label	Label of the rack containing the disk array controller	preload
install_date	Date of bay installation	preload – current time
state	UNKNOWN, OK, WARNING, FAULTY, OFF_LINE, OUT_OF_CLUSTER	Preload: OUT_OF_CLUSTER Dynamic - NSM
cluster_id	Id of the cluster parent	preload
nb_enclosure	Number of disk enclosure	Dynamic (DV=0) - NSM

Field name	Field information	Fill in method
nb_controller	Number of controller	Dynamic (DV=0) - NSM
nb_controller_ok	Number of controller in OK state	Dynamic (DV=0) - NSM
nb_controller_faulty	Number of controller in FAULTY state	Dynamic (DV=0) - NSM
nb_fc_port	Number of FC ports	Dynamic (DV=0) - NSM
nb_fc_port_connected	Number of FC ports in CONNECTED state	Dynamic (DV=0) - NSM
nb_fc_port_not_connected	Number of FC ports in NOT_CONNECTED state	Dynamic (DV=0) - NSM
nb_fc_port_disconnected	Number of FC ports in DISCONNECTED state	Dynamic (DV=0) - NSM
nb_fc_port_faulty	Number of FC ports in FAULTY state	Dynamic (DV=0) - NSM
nb_serial_port	Number of serial ports	Dynamic (DV=0) - NSM
nb_serial_port_connected	Number of serial ports in CONNECTED state	Dynamic (DV=0) - NSM
nb_serial_port_not_connected	Number of serial ports in NOT_CONNECTED state	Dynamic (DV=0) - NSM
nb_serial_port_disconnected	Number of serial ports in DISCONNECTED state	Dynamic (DV=0) - NSM
nb_serial_port_faulty	Number of serial ports in FAULTY state	Dynamic (DV=0) - NSM
nb_eth_port	Number of Ethernet ports	Dynamic (DV=0) - NSM
nb_ethernet_port_connected	Number of ethernet ports in CONNECTED state	Dynamic (DV=0) - NSM
nb_ethernet_port_not_connected	Number of ethernet ports in NOT_CONNECTED state	Dynamic (DV=0) - NSM
nb_ethernet_port_disconnected	Number of ethernet ports in DISCONNECTED state	Dynamic (DV=0) - NSM
nb_ethernet_port_faulty	Number of ethernet ports in FAULTY state	Dynamic (DV=0) - NSM
nb_disk	Number of disks	Dynamic (DV=0) - NSM
nb_disk_slot_ok	Number of disks in OK state	Dynamic (DV=0) - NSM
nb_disk_slot_faulty	Number of disks in FAULTY state	Dynamic (DV=0) - NSM
nb_disk_slot_empty	Number of disks in EMPTY state	Dynamic (DV=0) - NSM
nb_disk_slot_used_spare	Number of disks slots in USED_SPARE state	Dynamic (DV=0) - NSM
nb_disk_slot_missing	Number of disks in MISSING state	Dynamic (DV=0) - NSM
nb_power_supply	Number of power supplies	Dynamic (DV=0) - NSM
nb_power_supply_ok	Number of power supplies in OK state	Dynamic (DV=0) - NSM
nb_power_supply_faulty	Number of power supplies in FAULTY state	Dynamic (DV=0) - NSM
nb_nb_fan	Number of fans	Dynamic (DV=0) - NSM
nb_fan_ok	Number of fans in OK state	Dynamic (DV=0) - NSM
nb_fan_faulty	Number of fans in FAULTY state	Dynamic (DV=0) - NSM
nb_nb_power_fan	Number of power_fan	Dynamic (DV=0) - NSM
nb_power_fan_ok	Number of power_fan in OK state	Dynamic (DV=0) - NSM
nb_ power_fan_faulty	Number of power_fan in FAULTY state	Dynamic (DV=0) - NSM
nb_nb_temperature_sensor	Number of temperature sensors	Dynamic (DV=0) - NSM

Field name	Field information	Fill in method
nb_ temperature_sensor _ok	Number of temperature sensors in OK state	Dynamic (DV=0) - NSM
nb_temperature_sensor _warning	Number of temperature sensors in WARNING state	Dynamic (DV=0) - NSM
nb_temperature_sensor _faulty	Number of temperature sensors in FAULTY state	Dynamic (DV=0) - NSM
nb_lun	Number of lun	Dynamic (DV=0) - NSM
nb_spare	Number of spare disk	Dynamic (DV=0) - NSM
serial_number	Serial number of the array	Dynamic - storegister
type	Type of the array: OSS, MDS, ADMIN. Coded like UNIX rights (OMA, or – instead of the letter when the role does not apply)	preload
cfg_model	Name of the last applied model	Automatic - storemodelctl
cfg_model_application_date	Date of the last model application	Automatic - storemodelctl
mgmt_station_id	FK on HWMANAGER	preload
mgmt_node_id	FK on NODE	preload
status	Nagios status	Dynamic – NSM (DV="up")
unit_num	Unit Number	preload
comment	Free field	

Table 3-12. Storage – disk_array table

3.5.2.2 da_enclosure Table

Field name	Field information	Fill in method
id	Unique identifier for the disk enclosure in the database	preload –sequence
disk_array_id	Id of the parent array for this enclosure	preload
enclosure_type	Type of the disk enclosure	preload
rack_level	Level in the rack	preload
rack_label	Label of the rack containing the enclosure	preload
id_in_disk_array	Id of the enclosure in the array	preload
serial_number	Serial number of the enclosure	automatic – storeregister
model	Model of the disk enclosure	preload

Table 3-13. Storage – da_enclosure table

3.5.2.3 da_disk_slot Table

Field name	Field information	Fill in method
id	Unique identifier for the disk_slot in the database	Automatic - sequence
vendor	Vendor name of disk	Automatic - storeregister
model	Model of disk	Automatic - storeregister

Field name	Field information	Fill in method
serial_number	Serial number of disk	Automatic – storeregister
function	Function of disk: EMPTY, DATA, SPARE (DATA_VAULT, DATA_FLARE, SPARE_VAULT, SPARE_FLARE, etc.)	Automatic – storeregister
capacity	Disk capacity in MBs	Automatic – storeregister
enclosure_id	Id of the parent enclosure	Automatic - storeregister
disk_array_id	Id of the parent array for this disk_slot	Automatic - storeregister
state	State of the disk slot : EMPTY, OK, WARNING, FAULTY, MISSING, USED_SPARE	Dynamic – NSM
disk_enclosure_id	Disk number in the enclosure	Automatic - storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic - storeregister

Table 3-14. Storage – da_disk_slot table

3.5.2.4 da_controller Table

Field name	Field information	Fill in method
id	Unique identifier for the controller in the database	Automatic – sequence
disk_array_id	Id of the parent array for this controller	Automatic – storeregister
enclosure_id	Id of the parent enclosure	Automatic - storeregister
State	State of the controller : OK , FAULTY, WARNING, OFF_LINE	Automatic – NSM
object_num	Controller identifier in the enclosure	Automatic – storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister

Table 3-15. Storage – da_controller table

3.5.2.5 da_fc_port Table

Field name	Field information	Fill in method
id	Unique identifier for the fc_port in the database	preload – sequence
wwn	World Wide Name of the host port.	Automatic – storeregister
alpa_port_id	Loop address of the port	Automatic – storeregister
disk_array_id	Id of the parent array for this fc_port	preload
enclosure_id	Id of the parent enclosure	preload
State	State of the host port : CONNECTED, NOT_CONNECTED, DISCONNECTED, FAULTY	Dynamic – NSM
object_num	fc_port identifier in the enclosure	preload
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister

Table 3-16. Storage – da_fc_port.table

3.5.2.6 da_serial_port Table

Field name	Field information	Fill in method
id	Unique identifier for the serial port in the database	preload – sequence
disk_array_id	Id of the parent array for this serial port	preload
enclosure_id	Id of the parent enclosure	preload
type	type of serial port	preload
port_server_id	Port_server linked to this serial connection	preload
port_server_port	Index of the port used on the portserver (start at 0)	preload
serial_ip	IP address used to access to this serial port	preload
serial_name	Name of the console for conman	preload
state	State of the serial port : CONNECTED, NOT CONNECTED, DISCONNECTED, FAULTY	Dynamic – NSM
object_num	Serial port identifier in the enclosure	Preload
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister

Table 3-17. Storage – da_serial_port table

3.5.2.7 da_ethernet_port Table

Field name	Field information	Fill in method
id	Unique identifier for the emperat port in the database	preload - sequence
name	Name attached to this IP address	preload
disk_array_id	Id of the parent array for this Ethernet port	preload
enclosure_id	Id of the parent enclosure for this Ethernet port	preload
eth_switch_id	Id of the parent emperat_switch or parent pap_node	preload
ipaddr	IP address of the Ethernet port	preload
macaddr	MAC address of the Ethernet port	Automatic – storeregister
vlan_id	Id of the VLAN containing this emperat port	preload
type	Type of the Ethernet port : PUBLIC, ADMIN	preload
state	State of the Ethernet port : CONNECTED, NOT CONNECTED, DISCONNECTED, FAULTY	Dynamic – NSM
object_num	Ethernet port identifier in the enclosure	preload – storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister
admin_eth_switch_slot	Arrival slot number on ETH SW	preload
admin_eth_switch_port	Connection port on the ETH SW	preload

Table 3-18. Storage – da_ethernet_port Table

3.5.2.8 da_power_supply Table

Field name	Field information	Fill in method
id	Unique identifier for the power supply in the database	Automatic – sequence
disk_array_id	Id of the parent array for this power supply	Automatic – storeregister
enclosure_id	Id of the parent enclosure for this power supply	Automatic – storeregister
state	State of the power supply : OK, FAULTY,MISSING, [WARNING]	Dynamic – NSM
object_num	Power supply identifier in the enclosure	Automatic – storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister

Table 3-19. Storage – da_power_supply table

3.5.2.9 da_fan Table

Field name	Field information	Fill in method
Id	Unique identifier for the fan in the database	Automatic – sequence
disk_array_id	Id of the parent array for this fan	Automatic – storeregister
enclosure_id	Id of the parent controller for this power supply	Automatic – storeregister
state	State of the power supply: OK, FAULTY, MISSING, [WARNING]	Dynamic – NSM
object_num	Fan identifier in the enclosure	Automatic – storegister
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storegister

Table 3-20. Storage – da_fan table

3.5.2.10 da_power_fan Table

Field name	Field information	Fill in method
Id	Unique identifier for the power_fan in the database	Automatic - - sequence
disk_array_id	Id of the parent array for this power_fan	Automatic- storeregister
enclosure_id	Id of the parent enclosure for this power_fan	Automatic- storeregister
State	State of the power_fan: OK, FAULTY, MISSING, [WARNING]	dynamic – NSM
object_num	Power_fan identifier in the enclosure	Automatic- storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic- storeregister

Table 3-21. Storage – da_power_fan table

3.5.2.11 da_temperature_sensor Table

Field name	Field information	Fill in method
Id	Unique identifier for the temperature sensor in the database	Automatic – sequence
disk_array_id	Id of the parent array for this power supply (if controller_id and enclosure_id are NULL)	Automatic – storeregister
enclosure_id	Id of the parent enclosure for this power supply (if controller_id and array_id are NULL)	Automatic – storeregister
sensor_name	Name of the temperature sensor	Automatic – storeregister
state	State of the □emperature sensor : OK, WARNING, FAULTY	Dynamic – NSM
object_num	Temperature sensor identifier in the enclosure	Automatic – storeregister
vendor_location	Location of the component expressed in the vendor terms.	Automatic – storeregister

Table 3-22. Storage – da_temperature_sensor table

3.5.2.12 da_io_path Table

Field name	Field information	Fill in method
node_id	Id of the node which access to this FC port	preload
port_id	Id of da_fc_port used by the node	preload
fc_board_id	Id of the HBA board	preload

Table 3-23. da_io_path table

3.5.2.13 da_iocell_component Table

Field name	Field information	Fill in method
Iocell_id	Id of the IO cell	Preload - sequence
component_id	Id of a node or of a disk array	Preload
Type	Type of the component ("disk_array" or "node")	Preload
Rank	Rank of the node in the IO cell, or rank of the disk array in the IO cell. Start at 0.	preload

Table 3-24. Storage – da_iocell_component table

3.5.2.14 da_cfg_model Table

Field name	Field information	Fill in method
disk_array_id	Id of a disk array	Dynamic - storemodelctl
cfg_model_name	Model of a model which has been applied to the disk array	Dynamic - storemodelctl

application date	Date where the model has been applied	Dynamic - storemodelctl

Table 3-25. Storage – da_cfg_model table

3.5.2.15 da_power_port Table

Field name	Field information	Fill in method
Id	Unique identifier for the power_port in the database	Preload sequence
disk_array_id	FK to disk array	preload
enclosure_id	FK to enclosure id	preload
talim_id	FK to T_ALIM	preload
talim_port	Plug to be powered on/off onT_ALIM	preload

Table 3-26. Storage – da_power_port table

3.5.3 Machine View

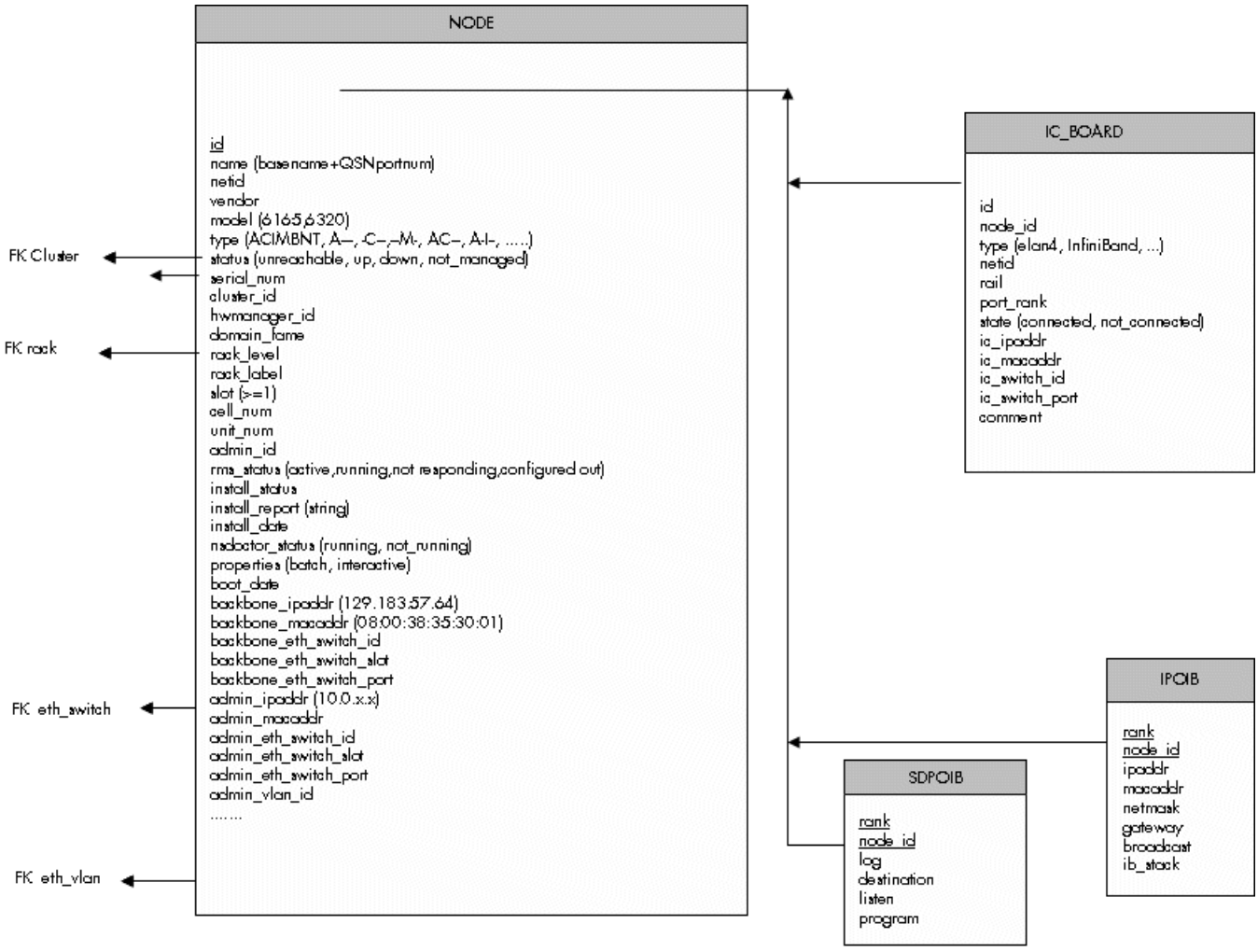


Figure 3-5. Cluster Database – Machine view 1

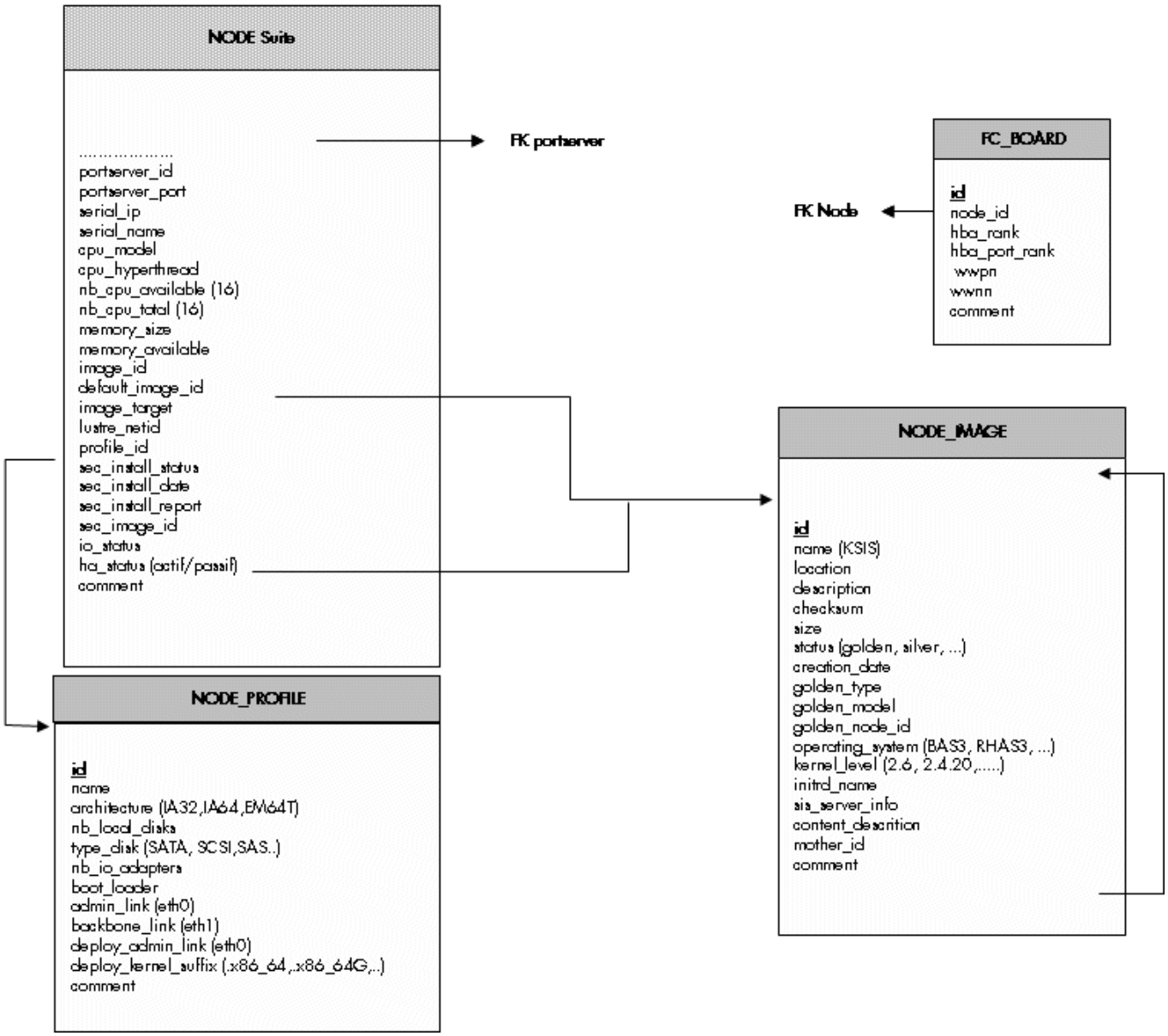


Figure 3-6. Cluster Database – Machine view 2

3.5.3.1 NODE Table

Column name	Description	Example	Fill-in method
id	primary key		preload– sequence
name	Node name	ns15	preload
netid	Node identifier number	1	preload
vendor	Name of vendor	Bull	preload
model	Node model	NS6165	preload
type	ACIMBNT node type, A------, -C-----, --I----, ---M---	A-IM---	preload

status	Nagios host_status up, down, unreachable	down	DV = up – Nagios
serial_num	Serial number		ACT
cluster_id	FK on the CLUSTER		preload
hwmanager_id	FK on the HWMANAGER		preload
domain_fame	Machine name PAP side		ACT
rack_level	Height in the rack	A	preload
rack_label	Rack name	CU0-A5	preload
slot	Number of slot for the node [1-14]	1	preload
cell_num	Cell number	1	preload
unit_num	Unit ID	3	preload
admin_id	FK towards ADMIN		admin
rms_status	RMS status	configure out	event handler
install_status	KsiS Status	not_installed	KsiS
install_report	message	Host not installed	KsiS
install_date	System installation date	13/12/04 10 :30 :10	KsiS
NsDoctor_status	running or not-running	not-running	NsDoctor + DV
properties	Torque properties	Batch	Torque + DV
boot_date	Date of the last boot		PostBootChecker
backbone_ipaddr	Backbone IP Address	129.183.57.64	Preload
backbone_eth_switch_id	FK on the ETH_SWITCH		Preload
backbone_eth_switch_slot	Arrival slot number on ETH SW		Preload
backbone_macaddr	mac adresse	08:00:38:35:30:01	ACT
backbone_eth_switch_port	Connection port for BK_ETH_SW	2	Preload
admin_ipaddr	Admin IP address	10.1.0.1	Preload
admin_eth_switch_id	FK on the ETH_SWITCH	1	Preload
admin_eth_switch_slot	Arrival slot number on ETH SW		Preload
admin_eth_switch_port	Connection port for AD_ETH_SW	5	Prelaod
admin_vlan_id	FK for ETH_VLAN		Preload
admin_macaddr			ACT
portserver_id	FK on the PORTSERVER		Preload
portserver_port	Port number for the PS		Preload
serial_ip	Serial line access IP address	129.183.75.10	Preload
serial_name	Name of the serial number	ns15s	Preload
cpu_model	CPU model	Montecito	Preload
cpu_hyperthread	Boolean	True	PostBootChecker
nb_cpu_available	Number of CPUs available	15	PostBootChecker
nb_cpu_total	Number of CPUs	16	Preload
memory_size	Memory size	64	Preload
memory_available	Size of memory available	64	PostBootChecker

image_id	FK on the NODE_IMAGE		KsiS
default_image_id	FK on the default image		KsiS
image_target	For future use	NULL	NULL
lustre_netid	For future use	NULL	NULL
profile_id	FKon the NODE_PROFILE		Preload
sec_install_status	Secondary image KSiS status		KSiS
sec_install_date	Secondary Image installation date		KSiS
sec_install_report	Secondary Image message		KSiS
sec_image_id	FK of the NODE_IMAGE		KSiS
io_status	I/O status of the node		storage
ha_status (active/passive)	HA status of the node		Cluster Suite
comment	Free field	NULL	

Table 3-27. Machine view – node table

3.5.3.2 NODE_IMAGE Table

Column name	Description	Example	Fill-in method
Id	PK		Sequence
name	Name of the image	try	KsiS
location	localisation	/path/name	KsiS
description	description		KsiS
checksum	checksum	12352425	KsiS
size	Image size		KsiS
status	image status	= golden, silver	KsiS
creation_date	date	=JJ/DD/YY HH :MI :SS	Trigger
golden_type	IO, HPC, MDS, ADMIN		KsiS
golden_model	6165,6320		KsiS
golden_node_id	id of node serving as the golden node		KsiS
operating_system	Distribution type	BAS4V2	KsiS
kernel_level	Kernel level	6.2	KsiS
initrd_name	Initrd name		KsiS
sis_server_info	name/version		KsiS
content_description	description of the image content		KsiS
mother_id	Link to original image		KsiS
comment	Free field		

Table 3-28. Machine view – Node_image table

3.5.3.3 NODE_PROFILE Table

Column name	Description	Example	Fill in method
Id	Primary Key	1	preload sequence
name	Name used to recognise the profile	MGMT	Preload
architecture	Type of architecture IA64, EM64T,etc.	IA64	preload
nb_local_disks	Number of internal disks	3	preload
type_disk	Type of disks (SATA, SCSI,SAS, etc)	SATA	preload
nb_io_adapters	Number of I/O cards	2	preload
boot_loader	elilo, grub	grub	KSiS
admin_link	admin interface (eth0)	eth0	DV
backbone_link	Interface backbone (eth1)	eth1	DV
deploy_admin_link	Deployment interface	eth0	DV
deploy_kernel_suffix	Kernel suffix (.x86_64, .x86_64G, etc.)	NULL	DV
comment	Free field		

Table 3-29. Machine view – Node_Profile table

3.5.3.4 IC_BOARD Table

This table describes Interconnect parameters (**Quadrics**, **Infiniband** or **GBEthernet**).

Column name	Description	Example	Fill in method
Id	Primary Key	1	preload sequence
node_id	FK on NODE	1	preload
type	type of card	elan4, infiniband	preload
netid	Node identifier number	3	preload
port_rank	Port number on the card	1	preload
ic_ipaddr	IP address of the IC Board	10.0.10.3	preload
Ic_macaddr	Mac address	unused	
rail	Number of the rail	2	preload
ic_switch_id	FK on IC_SWITCH		preload
ic_switch_port	Number of the IC_SWITCH port	64	preload
comment	Free field		

Table 3-30. Machine view – IC_BOARD table

3.5.3.5 IPOIB Table

This table describes Infiniband parameters for storage access.

Column name	Description	Example	Fill in method
rank	PK, Rank of the Infiniband adapter	0	updateIPOIB
node_id	PK, reference NODE	10	updateIPOIB
Ipaddr	ip address on Infiniband	172.193.1.1	updateIPOIB
macaddr	Mac address		updateIPOIB
gateway	ip address of the gateway		updateIPOIB
broadcast	ip address of the broadcast		updateIPOIB
ib_stack	type of stack IP, SDP, BOTH	SDP	updateIPOIB

Table 3-31. Machine view – IPOIB Table

3.5.3.6 SDPOIB Table

Column name	Description	Example	Fill in method
Rank	PK, Rank of the Infiniband adapter	0	updateSDPoIB
node_id	PK, reference NODE	10	updateSDPoIB
Log	Log in sdplib.conf		updateSDPoIB
Destination	Desination in sdplib.conf		updateSDPoIB
Listen	Listen in sdplib.conf		updateSDPoIB
Program	Program in sdplib.conf		updateSDPoIB

Table 3-32. Machine view – SDPOIB table

3.5.3.7 FC_BOARD table

Note:
This table only applies to systems which include a Storage Area Network (**SAN**).

Column name	Description	Example	Fill in method
Id	Primary key		storage
node_id	FK on the node	1	storage
hba_rank	Rank of the adapter		storage
hba_port_rank	Rank of the port		storage
wwpn	World Wide Port Name		storage
wwnn	World Wide Node Name		storage
comment	Free field		

Table 3-33. Machine view – FC_BOARD table

3.5.4 HWMANAGER View

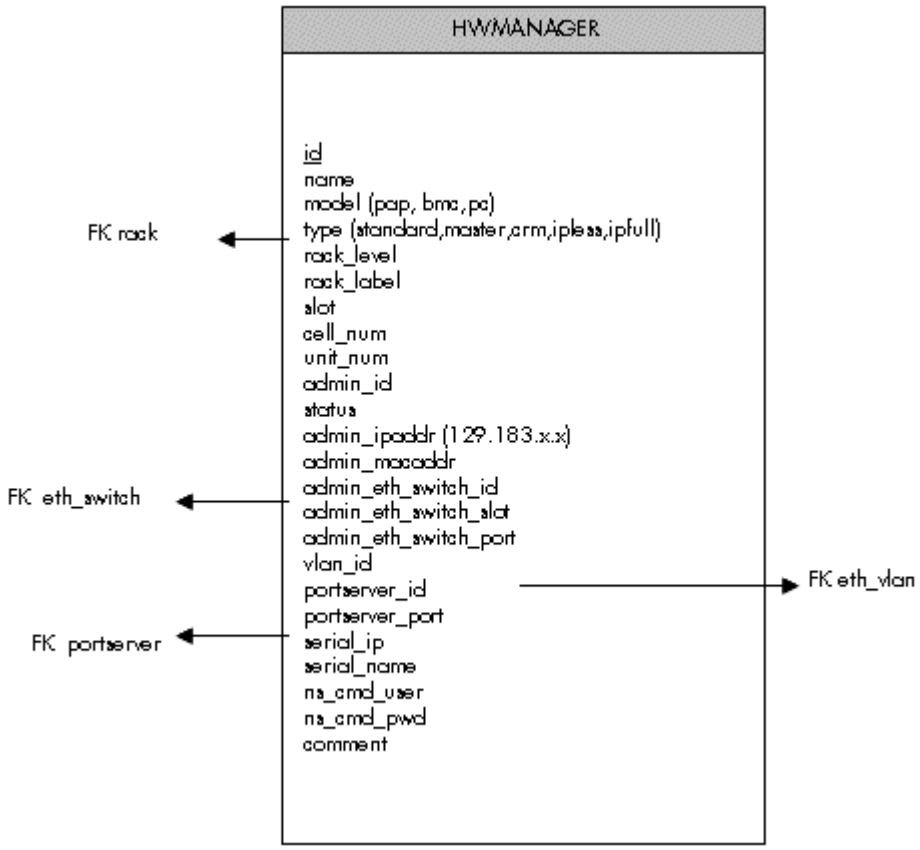


Figure 3-7. HWManager view

3.5.4.1 HWMANAGER Table

Column name	Description	Example	Fill in method
Id	Primary key		preload - Sequence
name	HWMANAGER IP name	papu1c2	preload
model	PAP or BAC	pap	preload
type	standard, master, crm, ipless, ipfull	standard	preload
rack_level	Height in the rack	E	preload
rack_label	Name of the rack	ISO0-H45	preload
cell_num	Number of the cell	3	preload
unit_num	Number of the unit	1	preload
admin_id	ADMIN id		admin
status	Nagios status	unreachable	DV=up – Nagios
admin_ipaddr	Admin IP address		preload
admin_macaddr	Mac address		ACT
admin_eth_switch_id	ETH_SWITCH id		preload
admin_eth_switch_slot	Arrival slot number on ETH SW		preload

Column name	Description	Example	Fill in method
admin_eth_switch_port	ETH_SWITCH connection port	2	preload
vlan_id	ETH_VLAN id		preload
portserver_id	PORTSERVER id		preload
portserver_port	Portserver port number		preload
serial_ip	Serial line access IP address	129.183.75.10	preload
serial_name	Serial line name	ns15s	preload
ns_cmd_user	User NC Commande	nsc	preload
ns_cmd_pwd	password	$nsc	preload
comment	Free field		

Table 3-34. HWManager Table

3.5.5 Complementary Tables

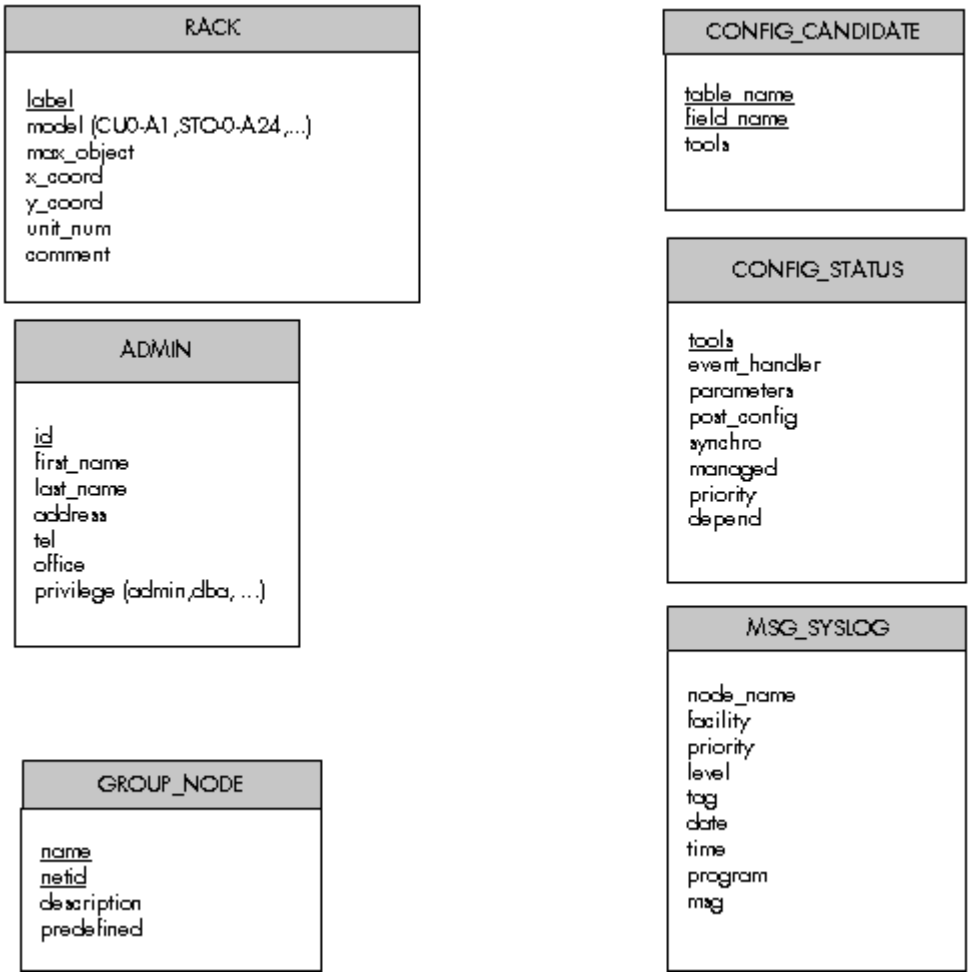


Figure 3-8. Cluster Database – Complementary tables

3.5.5.1 ADMIN Table

Column name	Description	Example	Fill in method
Id	PK		Sequence
first_name	First name	Stephane	admin
last_name	surname	Dupont	admin
address	address	…	admin
tel	Phone number		admin
office	office		admin
privilege	admin, dba, ….		admin

Table 3-35. Cluster Database – Admin table

3.5.5.2 RACK Table

Column name	Description	Example	Fill in method
Label	PK	RACK1	preload
Model	Type of rack	ARM3	preload
max_object	Maximum number of objects in the rack	3	preload
x_coord	Abscissa in the rows of racks		preload
y_coord	Ordinate in the length of racks		preload
unit_num	Number of theUnit	5	unused
comment	Free field		

Table 3-36. Cluster Database – Rack table

3.5.5.3 CONFIG_CANDIDATE Table

Column name	Description	Example	Fill in method
table_name	PK	node	creation
filed_name	PK	admin_ipaddr	creation
tools	list of the candidates tools	nagios, conman	creation

Table 3-37. Cluster Database – Config Candidate table

3.5.5.4 CONFIG_STATUS Table

Column name	Description	Example	Fill in method
tools	PK	nagios	creation
event_handler	generator of conf file	initNagiosCfg	creation
parameters	parameters of the event handlers	1,5,10	trigger
post_config	service to restart	nagios	creation
synchro	boolean, to be synchronized	True	trigger - dbmConfig
managed	Deactivation of the tool	True	creation
priority	Synchronisation order	1	creation
depend	List of the inter-dependency of the tool	group	creation

Table 3-38. Cluster database – Config_Status table

3.5.5.5 GROUP_NODE Table

Column name	Description	Example	Fill in method
name	PK	graphique	dbmGroup
netid	PK	10-20,25,30	dbmGroup
description	Comment about the group		dbmGroup
predefined	Predefined group	True	dbmGroup

Table 3-39. Cluster Database Group_Node table

3.5.5.6 MSG_SYSLOG Table

This table is not active in this version.

3.5.6 NsDoctor View

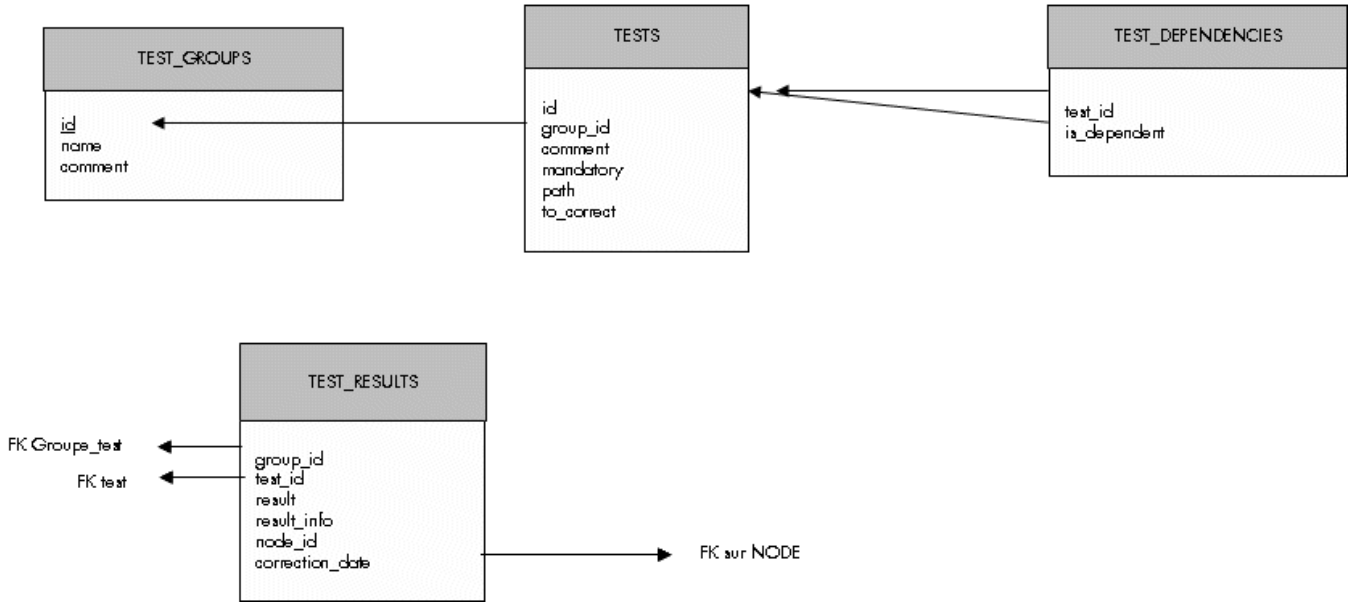


Figure 3-9. Cluster Database – NsDoctor view

3.5.6.1 TEST_GROUPS Table

Column name	Description	Example	Fill in method
id	Primary key		NsDoctor - Sequence
name	Name of the group of tests		preload
comment	Comment about the group		preload

Table 3-40. Cluster Database NsDoctor – Test_Groups table

3.5.6.2 TESTS Table

Column name	Description	Example	Fill in method
Id	Identifier of the test		NsDoctor -sequence
group_id	FK on GROUP_TEST_NSDOCTOR		preload
comment	Definition of the test		preload
mandatory	The test must be run if the group is selected		preload
path	Path Unix		preload
to_correct	Y/N/I		preload

Table 3-41. Cluster Database NsDoctor – Tests table

3.5.6.3 TEST_DEPENDENCIES Table

Column name	Description	Example	Fill in method
test_id	Identifier of the test		preload
is_dependent	Identifier of the test		preload

Table 3-42. Cluster Database NsDoctor – Test_Depencencies table

3.5.6.4 TEST_RESULTS Table

Column name	Description	Example	Fill in method
group_id	Identifier of the group		NsDoctor
test_id	Identifier of the test		NsDoctor
result	Result of the test (True ou False)		NsDoctor
node_id	FK on NODE		NsDoctor
correction_date	Date of the correction		NsDoctor

Table 3-43. Cluster Database NsDoctor – Test Results table

3.5.7 Nagios View

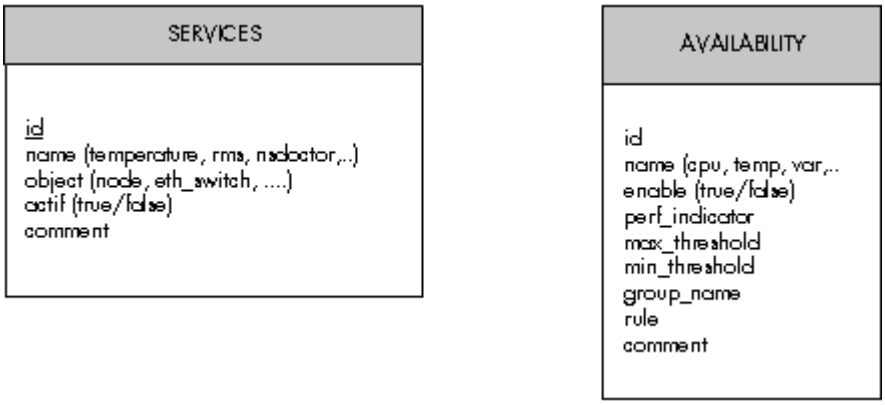


Figure 3-10. ClusterDB –Nagios View

3.5.7.1 Nagios SERVICES Table

Column name	Description	Example	Fill in method
id	Service id		dbmConfig
name	Service name	temperature	dbmConfig
object	Node , Eth_switch, portserver, etc.	node	dbmConfig
actif	Status of the service	true	Config & dbmServices
comment	comment	Temperature of the node	dbmConfig

Table 3-44. Nagios Services Table

3.5.7.2 Nagios AVAILABILITY Table

Column name	Description	Example	Fill in method
id	Service id		
name	CPU, temp, var	cpu	
enable	To check	true	
perf_indicator	Performance indicator	true	
max_thresold	Maximum threshold		
min_thresold	Minimum threshold		
group_name	Application group		
rule	Criterion rule		
comment	comment		

Table 3-45. Nagios Availability Table

3.5.8 Lustre View

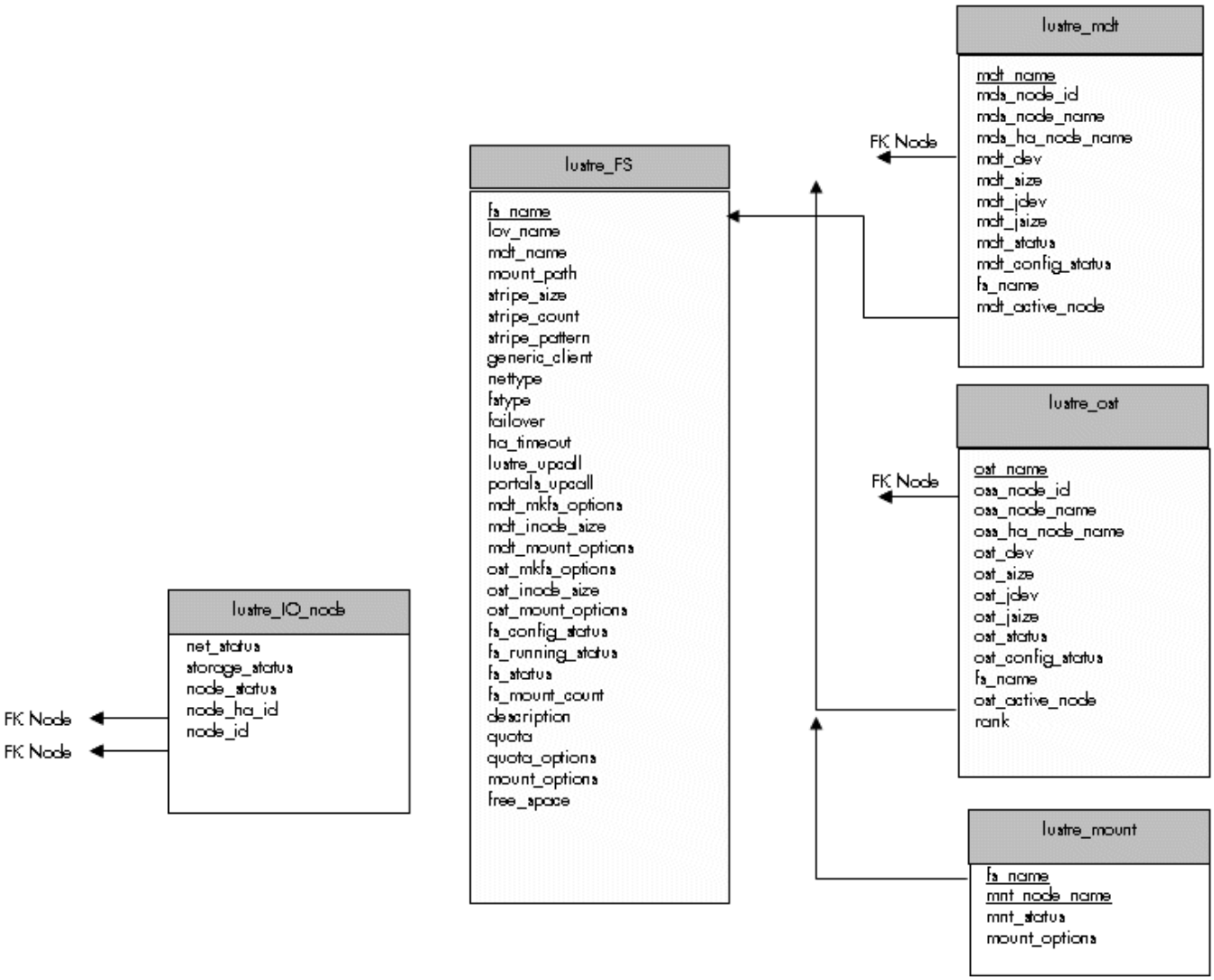


Figure 3-11. Cluster Database – Lustre view

3.5.8.1 lustre_fs Table

Each entry of the table describes a Lustre file system currently installed on the cluster.

Column name	Description	Example	Fill in method
fs_name	File system name	lustre_basic	lustre_config
mount_path	File system mount path	/mnt/lustre_basic	lustre_config
lov_name	LOV identification	lov_lustre_basic	lustre_config
mdt_name	MDT reference	mdt _ns44_1	lustre_config
stripe_size	Stripe size	4MB	lustre_config
stripe_count	Number of stripe per file	0 (all included OSTs)	lustre_config
stripe_pattern	Striping mode	0 (RAID0)	lustre_config
generic_client	Generic client profile	« client »	lustre_config
Nettype	Network type	elan	lustre_config

Fstype	Back-end file system type	ldiskfs	lustre_config
Failover	High-Availability indicator	« YES »	lustre_config
ha_timeout	High-Availability timeout for compute nodes	30	lustre_config
lustre_upcall	Lustre Exception processing script	/usr/bin/lustre_upcall	lustre_config
Portals_upcall	Portals layer exception processing script	/usr/bin/lustre_upcall	lustre_config
mdt_mkfs_options	MDT formatting options	mkfs command semantic	lustre_config
mdt_inode_size	Inode size for MDT back-end file system	1024	lustre_config
mdt_mount_options	MDT mount options	Mount command semantic	lustre_config
ost_mkfs_options	OSTs common formatting options	mkfs command semantic	lustre_config
ost_inode_size	Inode size for OSTs back-end file systems	1024	lustre_config
ost_mount_options	OSTs mount options	Mount command semantic	lustre_config
fs_config_status	File system configuration status		lustre_config
fs_running status	File system current running status		Lustre monitoring tools
fs_status	File system status		Lustre monitoring tools
fs_mount_count	File system mount counter	54	lustre_util
description	File system characteristics decription		lustre_config
Quota	User quotas management indicator	"YES"	lustre_config
quota_options	Quotas management tuning options		lustre_config
mount_options	Default mount options for the file system		lustre_config

Table 3-46. Cluster Database – Lustre View – Lustre_fs table

3.5.8.2 lustre_ost Table

Each entry of the table describes an OST available on the cluster.

Column name	Description	Example	Fill in method
ost_name	OST logical name	OST_ns32_1	lustre_investigate
oss_node_id	OSS ident in the node table	5	lustre_investigate
oss_node_name	Supporting OSS node name	ns32	lustre_investigate
oss_ha_node_name	Secondary OSS node name	ns33	lustre_investigate
ost_active_node	In case of HA management, current node name support	ns32	lustre_migrate
ost_dev	OST back-end device name	/dev/ldn.45.1	lustre_investigate
ost_size	OST back-end device size	140000000000	lustre_investigate

Column name	Description	Example	Fill in method
ost_jdev	External journal device name	/dev/ldn.45.2	lustre_investigate
ost_jsize	External journal device size	100000	lustre_investigate
ost_config_status	OST service configuration status		lustre_config
ost_status	OST service running status		Lustre management tools
fs_name	Proprietary file system name	lustre_basic	lustre_config

Table 3-47. Cluster Database – Lustre view – Lustre OST table

3.5.8.3 lustre_mdt Table

Each entry of the table describes an MDT available on the cluster.

Column name	Description	Example	Fill in method
mdt_name	MDT logical name	MDT_ns32_1	lustre_investigate
mds_node_id	MDS ident in the node table	5	lustre_investigate
mds_node_name	Supporting MDS node name	ns32	lustre_investigate
mds_ha_node_name	Secondary MDS node name	ns33	lustre_investigate
mdt_active_node	In case of HA management, current node name support	ns32	lustre_migrate
mdt_dev	MDT back-end device name	/dev/ldn.45.1	lustre_investigate
mdt_size	MDT back-end device size	140000000000	lustre_investigate
mdt_jdev	External journal device name	/dev/ldn.45.2	lustre_investigate
mdt_jsize	External journal device size	100000	lustre_investigate
mdt_config_status	MDT service configuration status		lustre_config
mdt_status	MDT service running status		Lustre management tools
fs_name	Proprietary file system name		lustre_config

Table 3-48. Cluster Database – Lustre View – Lustre_MDT Table

3.5.8.4 lustre_io_node Table

Each cluster node of I/O (I) or metadata (M) type has an entry in this table.

Column name	Description	Example	Fill in method
node_id	Ident of the node in the node table	ns32	preload
node_ha_id	Ident of the HA paired node in the node table	ns33	preload
net_status	Node network status	% available (0 – 33 – 66 – 100)	Lustre monitoring tools
storage_status	Node storage status	% available (0 – 12 – 25 - … - 100)	Lustre monitoring tools
node_Status	Node lustre status		Failover tools

Table 3-49. Cluster Database – Lustre View – Lustre_IO_node table

3.5.8.5 lustre_mount Table

Each entry of this table refers to a couple compute node / mounted Lustre file system.

Column name	Description	Example	Fill in method
mnt_node_name	Compute node name	ns87	lustre_util
nnt_status	Mount point status		lustre_util
fs_name	File system name		lustre_util
mount_options	Lustre file system current mount options for the compute node		lustre_util

Table 3-50. Cluster Database – Lustre view – Lustre_mount table

Chapter 4. Parallel File Systems

This chapter explains how these file systems operate on a Bull HPC system. It describes in detail how to install, configure and manage the Lustre file system.

The following topics are described:

- 4.1 *Parallel File Systems Overview*
- 4.2 *Lustre Overview*
- 4.3 *Lustre Administrator's Role*
- 4.4 *Planning a Lustre System*
- 4.5 *Lustre System Management*
- 4.6 *Installing and Managing Lustre File Systems*
- 4.7 *Monitoring Lustre System*

4.1 Parallel File Systems Overview

Parallel file systems are specifically designed to provide very high I/O rates when accessed by many processors at once.

A parallel file system provides network access to a "virtual" file system distributed across different disks on multiple independent servers or I/O nodes. Real files are split into several chunks of data or stripes, each stripe being written onto a different component in a cyclical distribution manner (striping).

For a parallel file system based on a client/server model, the servers are responsible for file system functionality and the clients provide access to the file system through a “mount” procedure. This mechanism provides a consistent namespace across the cluster and is accessible via the standard Linux I/O API.

I/O operations occur in parallel across multiple nodes in the cluster simultaneously. As all files are spread across multiple nodes (and even I/O buses and disks), I/O bottlenecks are reduced and the overall I/O performance is increased.

4.2 Lustre Overview

Lustre - a parallel file system - manages the data shared by several nodes, which is dispatched in a coherent way (cyclical distribution) on several disk systems. Lustre works in client / server mode. The server part supplies the functions of the file system, while the client part enables access to the file system through a mounting configuration.

Lustre relies on a set of Data and Meta Data servers which manage the following information related to the files:

- File attributes (name, access rights, hierarchy, etc.).
- File geometry, which means how a file is distributed across different servers.

When a node of the cluster needs access to the global file system, it will mount it locally via the client part. All the nodes can have access to the global file system.

MDS (MetaData Server)

MDS provides access to services called MDTs (MetaData Target).

A MDT provides a global NameSpace for a Lustre file system: it unifies the directory trees available from multiple file servers to provide a single global directory tree that can be mounted by the Lustre file system clients.

A MDT manages a backend ext3-like file system which contains all the metadata but none of the actual file data for an entire Lustre file system.

OSS (Object Storage Server)

OSS provides access to services called OST (Object Storage Targets).

An OST contains part of the file data (striping) for a given Lustre file system and very little metadata.
Each OST has its own block device and backend file system where it stores stripes of files in local ext3-like files.

One MDT and several OSTs make up a single Lustre file system and can be accessed through a Logical Object Volume (LOV). This set is managed as a group and can be compared to a NFS export or a LVM logical volume.
The LOV service is replicated on all the client nodes mounting the Lustre file system and distributes the I/O locking load among OSTs.

Lustre Client

A Lustre client results from the combination of an Object Storage Client (OSC) accessing the LOV.

A client node mounts the Lustre file system over the network and accesses the files with POSIX semantics.

Each client communicates directly with MDS and OSS.

4.3 Lustre Administrator's Role

Once the hardware has been setup and the software has been deployed, cluster administrators must perform the following tasks:

Determine how the hardware infrastructure will be used (number of file systems, size, storage resources used, allocation of I/O nodes, accessibility of the various file systems by the Lustre clients, etc.).

If necessary, modify the configuration of the storage devices and the configuration of the Quadrics interconnects (network zoning, etc).

Configure the Lustre service and activate the configured file systems.

During the file system lifetime, administrators may have to perform operations such as stop, start, or repair. They may decide to update a configuration or to change the one loaded. They also need to monitor the file system to check the current performance in case of degradation of service.

4.4 Planning a Lustre System

4.4.1 Data Pipelines

There are many data pipelines within the Lustre architecture, but there are two in particular which have a very direct performance impact: the network pipe between clients and OSSs, and the disk pipe between the OSS software and its backend storage. Balancing these two pipes maximizes performances.

4.4.2 OSS / OST Distribution

The number of clients has no real impact on the number of OSSs to be deployed. To determine the number of OSSs and how to distribute OSTs, two things have to be considered:

- The maximum bandwidth required gives the number of OSSs.
- The total storage capacity needed gives the number of OSTs.

To increase efficiency, it is preferable to distribute OSTs evenly on OSSs and to have fewer larger OSTs in order to use space more efficiently.

When calculating the size of the OSS server nodes, it is recommended that the CPUs are divided into thirds: one third for the storage backend, one third for the network stack and one third for Lustre.

4.4.3 MDS / MDT Distribution

The Lustre file system stores the file striping information in extended attributes (EAs) on the MDT. If the file system has large-inode support enabled (> 128bytes), then EA information will be stored inline (fast EAs) in the extra space available.

The table below shows how much stripe data can be stored inline for various inode sizes:

Inode Size	#of stripes stored inline
128	0(all EA is stored externally)
256	3
512	13
1024	35
2048	77
4096	163

Table 4-1. Inode Stripe Data

It is recommended that MDT file systems be formatted with the inode large enough to hold the default number of stripes per file to improve performance and storage efficiency.
One needs to keep enough free space in the MDS file system for directories and external blocks. This represents ~512 Bytes per inode.

4.4.4 File Striping

Lustre stripes the file data across the OSTs in a round robin fashion.

It is recommended to stripe over as few objects as is possible to limit network overhead and to reduce the risk of data loss, in case of OSS failure.

The stripe size must be a multiple of the page size. The smallest recommended stripe size is 512 KB because Lustre tries to batch I/O into 512 KB blocks on the network.

4.4.5 Lustre File System Limitations

On the device it manages, an OST reserves up to 400MB for an internal journal and 5% for the root user. This reduces the storage capacity available for the user's data. Like an OST, on the device it manages a MDT reserve.

The encapsulated modified ext3 file system used by MDTs and OSTs relies on the standard ext3 file system provided by the Linux system and optimizes performance and block allocation. It has the same limits in terms of maximum file size and maximum file system size.

4.5 Lustre System Management

Bull Lustre management tools provide services to manage large parallel file systems during their whole life cycle. Using these tools the cluster administrator will be able to:

- Configure and install **Lustre** file systems using the Lustre OST/MDT services provided by the storage management model deployment (refer to the Storage Devices Management chapter).
- Perform management operations such as start/stop, mount/umount file systems.

The administrator can monitor and get information about the Lustre system via a graphical user interface for performance and health monitoring.

Status targets of management tools for Lustre file systems and components current activity.

4.5.1 The Lustre Database

The Lustre management tools rely on the cluster database (ClusterDB) to store and get information about:

- I/O and Metadata nodes (lustre_io_node table),
- Lustre OST/MDT services (lustre_ost and lustre_mdt tables),
- File systems currently installed on the cluster (lustre_fs and lustre_mount tables).

Some of these tables information is loaded during the cluster deployment phase: those related to the I/O and Metadata nodes implementation and to the OST/MDT services repartition. The rest is maintained by the Lustre management tools.

Specific commands allow the administrator to edit and adjust information when necessary, for example in the case of node replacement due to hardware.

Note:

Updating the information stored in the Lustre database has direct consequences on the Lustre system behaviour. This must be done only by skilled administrators.

4.5.1.1 lustre_tables_dba

SYNOPSIS

lustre_ost_dba ACTION [options]
lustre_mdt_dba ACTION [options]
lustre_fs_dba ACTION [options]
lustre_io_node_dba ACTION [options]

DESCRIPTION

The **lustre_*tables*_dba** set of commands allows the administrator to display, parse and update the information of the Lustre tables in the ClusterDB.

lustre_ost_dba	Acts on the lustre_ost table, which describes the OST services
lustre_mdt_dba	Acts on the lustre_mdt table, which describes the MDT services
lustre_fs_dba	Acts on the lustre_fs table, which describes the Lustre file systems currently available on the cluster
lustre_io_node_dba	Acts on the lustre_io_node table, which gives the current status of the cluster I/O and metadata nodes.

These utilities are useful for checking the correctness of **ClusterDB** contents according to the last configuration updates. They allow the further adjustment of values in the instance of mistakes or failures of the Lustre management utilities thus avoiding a full repeat of the operation. They can also be used to force the Lustre services behaviour for some precise and controlled cases.

As these act on the global cluster configuration information, they must be used very carefully. The changes they allow may introduce fatal inconsistencies in the Lustre ClusterDB information.

ACTIONS

add	Adds an object description in the Lustre table.
update	Updates configuration items of an object description in the Lustre table.
del	Removes an object description from the Lustre table.
attach	Creates a link between Lustre tables objects (i.e. attaches an OST to a file system).
detach	Removes a link between Lustre tables objects (i.e. frees an OST).
list	Displays object information according to the selected criteria provided by the options.
set	Sets one or more status items of an object description.
-h(elp)	Displays this help and exits.
-v(ersion)	Displays the utility version and exits.

OPTIONS

The options list available for the actions depends on the kind of object they act on and on the action itself. Please, refer to the help of each command for option details.

4.5.2 /etc/lustre/storage.conf for Lustre Tools without ClusterDB

The **/etc/lustre/storage.conf** file stores information about the storage devices available on the cluster when ClusterDB is **NOT** present and it records which ones are OSTs and which ones are MDTs. It must be located on the management node. This file is composed of lines with the following syntax:

<ost|mdt>: name=<> node_name=<> dev=<> [ha_node_name=<>] [size=<kB>] [jdev=<> [jsize=<kB>]]

ost/mdt	This device is designated to be either an OST or a MDT.
name	The name given to the OST or MDT.
node_name	The hostname of the node containing the device.
dev	The device path (for example **/dev/sdd**).
ha_node_name	The hostname of the failover node. This has to be consistent with the content of **/var/lustre/status/lustre_io_nodes**.
size	Size of the device in kBs.
jdev	The name of the device where the **ext3** journal will be stored, if this is to be outside the main device. This parameter is optional. Loop devices cannot be used for this purpose.
jsize	The size of the journal device in kB.

Comments are lines beginning with # (sharp).

4.5.2.1 Filling /etc/lustre/storage.conf

This file is updated with the information obtained from the **/proc/partitions** or **/sys/block/** of the I/O nodes. For example, on a cluster where **ns13** is an I/O node:

```
>ssh ns13 -l root "cat /proc/partitions"
```

```
major minor  #blocks  name

   8     0   71687372 sda
   8     1     524288 sda1
   8     2   69115050 sda2
   8     3    2048000 sda3
   8    16   71687372 sdb
   8    32   17430528 sdc
   8    48   75497472 sdd
   8    64   17430528 sde
   8    80   75497472 sdf
   8    96   17430528 sdg
   8   112   75497472 sdh
```

sda and **sdb** are system disks of **ns13** so they must NOT be used as Lustre storage devices. Devices **sdd** to **sdh** are the devices which are available. 17430528 kB disks will be used as journal devices and 75497472 kB disks as the main devices.

This choice results in the following lines being included in **/etc/lustre/storage.conf** file for the management node:

```
mdt: name=ns13_sdd node_name=ns13 dev=/dev/sdd size=75497472
jdev=/dev/sdc jsize=17430528
ost: name=ns13_sdf node_name=ns13 dev=/dev/sdf size=75497472
jdev=/dev/sde jsize=17430528
ost: name=ns13_sdh node_name=ns13 dev=/dev/sdh size=75497472
jdev=/dev/sdg jsize=17430528
```

The decision as to which devices will be used as **MDTs** and which will be **OSTs** will be left to the administrator. This procedure fhas to be done for each I/O node and new lines appended to the **/etc/lustre/storage.conf** file of the management node. Bull provides a wizard to help the creation of the **storage.conf** file, this is **/usr/lib/lustre/lustre_storage_config.sh**.

4.5.2.2 Storage Inspection Wizard: /usr/lib/lustre/lustre_storage_config.sh

/usr/lib/lustre/lustre_storage_config.sh is a script that helps the administrator to complete the storage.conf file.

SYNOPSIS

```
lustre_storage_config.sh <node> <regexp_on_devices>
<upper_size_limit_of_journal_in_kb>
```

node This details the node to be inspected

regexp_on_devices A regular expression on the device name as given in the **/dev/ directory** of the I/O node. Do not forget this is a regular expression, not a globbing expression.

upper_size_limit_of_journal_in_kb
The difference between data devices and journal devices is made according to their size in kBs. If a device has a size greater than this parameter, it is assumed to be a data device. If a device has a size smaller than this parameter, it is assumed to be journal device.

The output produced by the **lustre_storage_config.sh** script is a template of lines to be used by **storage.conf**. These lines may require minor modifications. **lustre_storage_config.sh** looks for **/var/lustre/status/lustre_io_nodes** that can be used to fill in the **ha_node** field, therefore **lustre_io_nodes** should have been filled in before running the **lustre_storage_config.sh** script.

For example, for a cluster with two High Availability I/O nodes: **ns6** and **ns7**. The content for the **lustre_io_nodes** is as follows:

```
NODE_NAME=ns6
NODE_HA_NAME=ns7
LUSTRE_STATUS=OK

NODE_NAME=ns7
NODE_HA_NAME=ns6
LUSTRE_STATUS=OK
```

1. Link to Lustre devices is run using **stordiskname** and the output is as follows:

```
[root@ns6 ~]# ll /dev/ldn.nec.*
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.0 -> /dev/sdd
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.1 -> /dev/sde
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.2 -> /dev/sdn
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.3 -> /dev/sdo

[root@ns7 ~]# ll /dev/ldn.nec.*
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.0 -> /dev/sdd
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.1 -> /dev/sde
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.2 -> /dev/sdn
lrwxrwxrwx  1 root root 8 May 19 13:23 /dev/ldn.nec.3 -> /dev/sdo
```

   The same devices can be seen on both **ns6** and **ns7**.

2. All devices that start with **ldn** are to be used however it is not clear for the moment which are data devices and which are journal devices. From the management node the **lustre_storage_config.sh** script is run.

```
[root@ns2 /]# /usr/lib/lustre/lustre_storage_config.sh ns6 'ldn.*' 0
#call: ns6 ldn.* 0
```

   The resulting output is as follows:

```
ost: name=ost_ns6.nec.0 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.0 size=262144
ost: name=ost_ns6.nec.1 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.1 size=262144
ost: name=ost_ns6.nec.2 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.2 size=46137344
ost: name=ost_ns6.nec.3 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.3 size=46137344
```

   From the output above it can be seen that there are two sizes for the devices, the data devices (46137344 kBs) and the journal devices (262144 kBs).

3. The size of the journal device has been identified as 262144 kB, this means that the following command can be run:

```
[root@ns2
```

   The output is as follows:

```
/]# /usr/lib/lustre/lustre_storage_config.sh ns6   'ldn.*'
262144
#call: ns6 ldn.* 262144
ost: name=ost_ns6.nec.2 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns6.nec.3 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
```

4. The output is saved in the **storage.conf** file using the following command:

```
 [root@ns2 /]# /usr/lib/lustre/lustre_storage_config.sh ns6 'ldn.*' 262144
>>/etc/lustre/storage.conf
```

5. The same operation now has to be run on **ns7**, as below.

```
 [root@ns2 /]# /usr/lib/lustre/lustre_storage_config.sh ns7 'ldn.*' 262144
#call: ns6 ldn.* 262144
```

The output is as follows:

```
ost: name=ost_ns7.nec.2 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns7.nec.3 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
```

6. The output above is now saved in the **storage.conf** file using the following command:

```
 [root@ns2 /]# /usr/lib/lustre/lustre_storage_config.sh ns7 'ldn.*' 262144
>>/etc/lustre/storage.conf
```

At this point, the same devices will be stored twice in the **storage.conf** file as shown in the output below.

```
ost: name=ost_ns6.nec.2 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns6.nec.3 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
ost: name=ost_ns7.nec.2 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns7.nec.3 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
```

7. A decision has to be made at this point as to which devices will have **ns6** as the master node, and which devices will have **ns7** as the master node. An example is shown below:

```
ost: name=ost_ns6.nec.2 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns7.nec.3 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
```

8. The first device should be designated as an **mdt**. This is done by replacing **ost** by **mdt**, as shown below:

```
mdt: name=mdt_ns6.nec.2 node_name=ns6 ha_node_name=ns7
dev=/dev/ldn.nec.2 size=46137344 jdev=/dev/ldn.nec.0 jsize=262144
ost: name=ost_ns7.nec.3 node_name=ns7 ha_node_name=ns6
dev=/dev/ldn.nec.3 size=46137344 jdev=/dev/ldn.nec.1 jsize=262144
```

9. **storage.conf** is now ready. If you have more than one pair of High Availability nodes then the same operation will have to be repeated for each pair of nodes.

10. The consistency of the **/etc/lustre/storage.conf** files can be checked using the command below optimal:

```
lustre_util check_storage.
```

Important:

lustre_storage_config.sh associates the data devices and journal devices in alphabetical order. On some devices, for example DDN, this association is not necessarily optimal and special tuning may be required to improve the performance.

Note:

If it is planned to upgrade the cluster from one which does not have a database installed to one which includes a database then **lustre_util check_storage** should not report any errors.

4.5.2.3 Loading storage.conf into the Cluster Database using load_storage.sh

load_storage.sh is a script that is used to load **storage.conf** information into the **lustre_ost** and **lustre_mdt** tables of the cluster database. This may be useful:

- If a cluster database is added to your system.
- If there is a database, but no management tools are provided for the storage devices, for example for **NEC** devices.

SYNOPSIS

```
/usr/lib/lustre/load_storage.sh < update|crush >  <storage.conf file>
```

update	Adds the new devices but does not erase the existing **lustre_ost** and **lustre_mdt** tables.
crush	Remove the **lustre_ost** and **lustre_mdt** tables, and then add new devices.
storage.conf file	The path to your **storage.conf** file (this usually **/etc/lustre/storage.conf**)

4.5.2.4 Practical Recommendation

If you use a High Availability MDS as the management node it will be possible to move the **/etc/lustre/storage.conf** file to **/var/lustre/status/**, using the command below and to then make a symbolic link to this file on the 2 MDS nodes:

```
ln -s /var/lustre/status/storage.conf /etc/lustre/storage.conf
```

The same thing can be done for the **/etc/lustre/models** directory. In this way, information does need to be replicated and is available on the node where **/var/lustre/status** is mounted.

4.5.3 Lustre Management Configuration File: /etc/lustre/lustre.cfg

Lustre management tools use this file to get configuration information. This file must reside on all OSS and MDS nodes. Refer to **lustre_util** man page to know how to distribute this file easily.

File Syntax:

```
VARIABLE=VALUE
```

Lines beginning with # are comments.

/etc/lustre/lustre.cfg contents:

LUSTRE_MODE=XML

XML: Information about filesystems is given to CFS Lustre tools using the XML format. These files are stored in the directory defined by LUSTRE_CONFIG_DIR on OSS, MDS and Management Node.

Default value is XML. This value is mandatory for failover configuration. **HTTP mode is no longer supported.**

CLUSTERDB=yes

When this variable is set to yes, storage, file systems and mount information is retrieved and stored from the clusterDB tables (**lustre_ost**, **lustre_mdt**, **lustre_mount** and **lustre_fs**.

LUSTRE_CONFIG_DIR=/etc/lustre/conf/

This variable contains the path of the directory where the XML/XMF files are created on the Management Node and where they have to be found on the OSSs and MDSs. The **lustre_util** command uses this path to store and read XML/XMF when required. This directory can be shared using NFS. If LUSTRE_MODE is set to XML, **lustre_util** creates this directory on all OSS and MDS nodes in order to copy the XML file associated with filesystems during the install process, as required by CFS Lustre tools (**lconf**).

Default value is **/etc/lustre/conf/**.

LUSTRE_NET=tcp or elan or o2ib

This variable specifies the kind of network used by Lustre on the whole cluster.

Default value is **tcp**. It is only used by the **lustre_check** monitoring tool.

LUSTRE_ADMIN=hostname

This variable contains the hostname of the I/O server used as central point of management for Lustre in case of cluster not monitored by a management station (CLUSTERDB="no"). The primary MDS node is to be chosen for that purpose.

No default value is defined.

LUSTRE_ADMIN2=hostname

LUSTRE_ADMIN2 is used only if the HA feature is enabled on the I/O nodes. It provides the hostname of the backup MDS used as alternative Lustre management point.

No default value is defined.

LUSTRE_LDAP_URL=ldap://hostname/

This variable contains the address of the ldap server used to store HA information. For example if the ldap server is on a node called ns2, then LUSTRE_LDAP_URL=ldap://ns2/.

No default value is defined.

LUSTRE_LDAP_URL2=ldap://hostname/

LUSTRE_LDAP_URL2 is used only when there is no management station supporting the full HA feature. In this case, it provides the LDAP URL of an alternative management station.

No default value is defined.

LUSTRE_DEBUG=yes or no

If this variable is set to "yes", Lustre management daemons are allowed to log trace information:

- in **/var/log/lustre** directory for failover
- in **/tmp/log/lustre** directory for database daemons

Default value is no.

I/O scheduler for block devices

LUSTRE_OST_DEV_IOSCHED = noop or anticipatory or deadline or cfq
(I/O scheduler for OST devices)

LUSTRE_OST_JNR_IOSCHED = noop or anticipatory or deadline or cfq
(I/O scheduler for OST ext3 journal devices)

LUSTRE_MDT_DEV_IOSCHED = noop or anticipatory or deadline or cfq
(I/O scheduler for MDT devices)

LUSTRE_MDT_JNR_IOSCHED = noop or anticipatory or deadline or cfq
(I/O scheduler for MDT ext3 journal devices)

These variables define the I/O scheduler for block devices. For details about I/O schedulers refer to the **/Documentation/block** directory of kernel sources.

Default and recommended values are:

- **deadline** for LUSTRE_MDT_DEV_IOSCHED,
- **noop** for LUSTRE_OST_DEV_IOSCHED, LUSTRE_OST_JNR_IOSCHED and LUSTRE_MDT_JNR_IOSCHED.

If OSTs/MDTs are disc partitions (not the whole device) the choice of the scheduler is left to the Administrator.

LUSTRE_SNMP=yes or no

If this variable is set to yes, the **snmpd** server will be enabled on the I/O nodes when **lustre_util set_cfg** is called (`chkconfig --level 345 snmpd on && service snmpd restart`). This allows the OSS and MDS to send snmp traps to the Management Node when errors occur. These traps force the nagios lustre service to run in order to check the health of the filesystems.

Default value is no.

DISABLE_LUSTRE_FS_NAGIOS=yes or no

Setting this to yes will disable the call of lustre_fs_nagios every 15 mn on management node.

Default value is no

LUSTRE_TUNING_FILE=/etc/lustre/tuning.conf

This is the path to the tuning file, the default value is **/etc/lustre/tuning.conf**.

4.5.4 Lustre Services Definition

The Lustre services MDT(s) and OST(s) rely on the devices created by the storage units configuration deployment. For this reason their distribution schema is tightly dependant of the storage configuration planning and vice versa.

A common model and deployment process is used for both storage units and Lustre services. The model describes the relationship between storage units, nodes, devices and Lustre services.

Refer to the "Storage Administration" chapter for more information.

Each Lustre service defined on the cluster I/O nodes is described by an entry in the **ClusterDB**. During the first cluster deployment phase, the model file is parsed for the storage elements which are created on the nodes and the information related to the Lustre services is stored in the Lustre tables of the **CLusterDB**, **lustre_mdt** for MDT services and **lustre_ost** for OST services.

This is theoretical information, which needs to be checked against the node reality using the **lustre_investigate check** utility. Inconsistencies may come from a model file error or elements configuration failure on nodes.

This check operation must be done after every cluster configuration or reconfiguration operation or every time the Lustre services configuration is modified.

4.5.4.1 lustre_investigate

SYNOPSIS

lustre_investigate check [-C <io_cell_list>|-n <nodes_list>|-f <file_system_name>]

lustre_investigate display [-C <io_cell_list>|-n <nodes_list>|-f <file_system_name>]

DESCRIPTION

lustre_investigate can be used only if the cluster configuration information is managed using the cluster management database, ClusterDB.
It allows the administrator to check the consistency between the information concerning the Lustre services and the real storage configuration available on I/O nodes.

Each Lustre service defined on the cluster I/O nodes is described by an entry in the ClusterDB. This entry provides information about the back-end device used by the service and the primary and the secondary node the service should run on.
Due to failures or cluster reconfiguration operations, this information may become obsolete. An availability status is maintained, which indicates if it is still correct or needs to be updated. This status is updated by running **lustre_investigate**.

lustre_investigate must be used from the management station. It issues check commands to each node of the selected range. The returned information is then evaluated against the one stored in the CLusterDB. It relies on the **pdsh** parallel shell to dispatch remote commands.

ACTIONS:

check	Parses the Lustre services entries in the ClusterDB according to the select criteria and checks their information consistency.
display	Displays the ClusterDB information about the Lustre services corresponding to the select criteria.

OPTIONS:

-h(elp)	Displays this help and exits.
-v(ersion)	Displays the current utility version and exits.
-C	Range of I/O cells (format [x,m-n]).
-n	Range of nodes (format <prefix>[x,m-n]).
-f	<file_system_name> is the name of the file system to work on.

If neither **-C**, **-n** nor **-f** are provided, all Lustre services declared in the cluster database management are processed.

4.5.5 Creating Lustre File Systems

4.5.5.1 Prerequisites

- **/etc/lustre/lustre.cfg** is assumed to be updated correctly as described in the section - Lustre Management Configuration File: /etc/lustre/lustre.cfg.
- If you are using a cluster database (CLUSTERDB=yes) **lustre_ost** and **lustre_mdt** tables are assumed to be updated correctly (use **lustre_investigate check** to verify).
- If you are not using a cluster database (CLUSTERDB=no), storage.conf must be correctly filled.
- Lustre tools use **ssh** to execute remote commands, so users must be allowed to log into nodes without being prompted for a password. This can be done by appending the right keys in **/root/.ssh/authorized_keys2**.

4.5.5.2 Lustre Model File (.lmf)

A Lustre model file describes one or several Lustre filesystems that can be used at the same time. This means they do not share OSTs or MDT. Such files are stored in the **/etc/lustre/models** directory.

File Syntax:

```
keyword: <value>
```

Lines beginning with # are comments.

Possible Keywords:

stripe_size	Specify the stripe size in bytes. Default is 1048576 (1M).
stripe_count	Specify the number of OSTs each file should be striped onto. The default value is 2.
stripe_pattern	Only pattern 0 (RAID 0) is supported currently.
nettype	Possible values are **tcp** or **elan** or **o2ib**. The default is elan.
generic_client	The name of the directory from which the filesystem is mounted on the mds. If the network is elan, the default is clientelan. If network is tcp, default is **clienttcp**.
fstype	File system type. Possible values are **ldiskfs** or **ext3**. Default (and recommended) is **ldiskfs**.
failover	Enable failover support on OST. Possible values are yes or no. The default is no.
ha_timeout	Timeout in seconds before going into recovery. Default is 30s.

lustre_upcall Location of the Lustre **upcall** script used by the client for recovery. No default script is defined.

portals_upcall Location of the Portals **upcall** script used by the client for recovery. No default script is defined.

mdt_mkfs_options Optional argument to **mkfs** for MDT. By default, no option is specified.

mdt_inode_size Specify new **inode** size for underlying MDT ext3 file system. The default is self-evaluated.

mdt_mount_options Optional argument to mount fs locally on the MDS. By default, no option is specified.

ost_mkfs_options Optional argument to mkfs for OST. By default, no option is specified.

ost_inode_size Specify new inode size for underlying OST ext3 file system. The default is self-evaluated.

ost_mount_options Optional argument to mount fs locally on OSS. By default, no option is specified.

cluster_id Specify the cluster ID (one filesystem uses a single cluster id)

mount_options Defines the default options to mount the filesystem on clients. Options are separated with ",". Available options are: ro, rw, user_xattr, nouser_xattr, acl, noacl. Default is no option and the filesystem will be mounted rw. For example, `mount_options: ro` means that, by default, this filesystem is mounted in read-only mode.

quota Enables quota support. Possible values are yes or no. The default is no.

quota_options If quota is set to yes, it describes options for quota support. The default is: quotaon=ug,iunit=5000,bunit=100,itune=50,btune=50. **Do not use other settings for the moment.**

description A ONE LINE free description of your filesystem (up to 512 chars). The default is empty string.

If previous keywords are used in the header of the file, before any filesystem definition (this means before any use of the **fs_name** keyword), they set the new default values which can be locally overloaded for a filesystem.

fs_name This keyword is the starting point of a filesystem definition. It is the name of the filesystem (name of the xml file or the entry in the ldap database).fsname must be defined for each filesystem.

mount_path The mount-point to use to mount Lustre filesystem. Same mount-point must not be used for another filesystem defined in the same model file. Default is /mnt/lustre_<fs_name>.

ost [name=<RegExp>] [node_name=<RegExp>] [dev=<RegExp>] [size=<RegExp>] [jdev=<RegExp>] [jsize=<RegExp>] [cfg_status=available|formated]

Specify OSTs to use with this filesystem, using regular expressions matching their name, node_name, device, size, journal device, journal size or status. At least one field must be specified. If several fields are specified, only OSTs matching all fields of the lines will be chosen. You can use as many OST lines as you need. **At least one OST line must be defined for each filesystem.**

mdt [name=<RegExp>] [node_name=<RegExp>] [dev=<RegExp>] [size=<RegExp>] [jdev=<RegExp>] [jsize=<RegExp>] [cfg_status=available|formated]

Specify MDT of this filesystem. It is the same syntax as for the OSTs. If several MDTs match, then the first will be used.

Note:

One and only one MDT line must be defined for each filesystem.

4.5.5.3 Extended Model Files (.xmf)

The purpose of the extended model files is to maintain a strict description of a filesystem. It is planned to replace xml files with this format. They have exactly the same syntax as previous model files, except that the OSTs/MDTs are strictly described and each OST/MDT line MUST point to one and only one OST/MDT of the **lustre_ost** and **lustre_mdt** tables. They can be used in place of lmf files. They are automatically generated in **LUSTRE_CONFIG_DIR** when you use **lustre_util install**, **update** or **rescue** commands.

4.5.5.4 Lustre Model Sample File

There follows a model file which describes two filesystems fs1 and fs2, on a cluster with nodes called ns<XX>. Information about OSTs and MDTs can be found using lustre_ost_dba list and lustre_mdt_dba list if a cluster database is present, or in **/etc/lustre/storage.conf** if no cluster database is present.

```
#####################################################
# Firstly, the new default values for the 2
# filesystems are defined

# To prevent failover
failover: no

# Set block-size to 4096 for mdt
mdt_mkfs_options: -b 4096
```

```
# Set block-size to 4096 for osts
ost_mkfs_options: -b 4096

# Network is elan
nettype: elan

# New mount options
ost_mount_options: extents,mballoc

######################################################
# First filesystem : fs1

# Filesystem name is fs1
fs_name: fs1

# mount-point of this filesystem will be /mnt/lustre1
# instead of the default /mnt/lustre_fs1
mount_path: /mnt/lustre1

# To specify osts hosted by nodes with names ending by odd numbers, with device
       names ending from 2 to 4
ost: node_name=ns.*[1,3,5,7,9] dev=.*[2-4]

# To specify the ost named ost_ns10.ddn1.6
ost: name=ost_ns10.ddn1.6

# The mdt will be the first hosted by ns12 with a name ending with a 3
mdt: node_name=ns12 name=.*3

######################################################
# Second filesystem : fs2

# Filesystem name is fs2
fs_name: fs2

# mount-point of this filesystem will be /mnt/lustre2
# instead of the default /mnt/lustre_fs2
mount_path: /mnt/lustre2

# To specify osts hosted by nodes with name ending with even numbers, with device
       names ending with 1,2,3 and 5
ost: node_name=ns.*[2,4,6,8,0] dev=.*[1-3,5]

# To specify the mdt named mdt_ns13.ddn12.31
mdt: name=mdt_ns13.ddn12.31

# To specify the generic_client to be fs2_client instead of
# clientelan
generic_client: fs2_client
```

4.6 Installing and Managing Lustre File Systems

lustre_util is the tool used to install, enable, disable, mount and unmount, one or more Lustre file systems from an administration node.

4.6.1 Installing Lustre File Systems using lustre_util

To install lustre file systems, the following tasks must be performed:

1 Use **lustre_util install** command to install the file system.

2 Use **lustre_util start** command to enable the file system.

3 Use **lustre_util mount** command to mount file systems on client nodes.

4.6.2 Removing Lustre File Systems using lustre_util

To uninstall lustre filesystems, the following tasks must be performed:

1 Use **lustre_util umount** command to unmount file systems on client nodes.

2 Use **lustre_util stop** command to disable the file systems.

3 Use **lustre_util remove** command to remove the file system.

4.6.3 lustre_util Actions and Options

SYNOPSIS

lustre_util set_cfg [-n <I/O nodes list > | -p <I/O nodes rms partition>]

lustre_util install -f < lmf or xmf path > [--kfeof] [--lconf <option>]

lustre_util update -f < lmf or xmf path > [--kfeof] [--lconf <option>]

lustre_util fsck -f < fs_name | all >

lustre_util chk_dev -f < lmf,xmf,xml files or fs_name | all >

lustre_util rescue -f < fs_name | all >

lustre_util start -f < fs_name | all > [--lconf <option>]

lustre_util tune_servers -f < fs_name | all >

lustre_util mount -f < fs_name | all > -n <nodes|recover|all>| -p <rms_partition> --mount <[+]opt1,opt2,...>

lustre_util umount -f < fs_name | all > -n <nodes|all> | -p <rms_partition>

lustre_util status [-f < fs_name | all >] [-n <nodes|all> | -p <rms_partition>]

lustre_util fs_status [-f < fs_name | all >]

lustre_util mnt_status [-f < fs_name | all >] [-n <nodes|all> | -p <rms_partition>]

lustre_util stop -f < fs_name | all > [--lconf <option>]

lustre_util remove -f < fs_name | all >

lustre_util info -f < lmf,xmf,xml files or fs_name | all >

lustre_util short_info -f < lmf,xmf,xml files or fs_name | all >

lustre_util lfsck -f < fs_name | all > -n <node> -d <shared_directory>

lustre_util build_mdt_db -f < fs_name | all > -n <node> -d <directory>

lustre_util build_ost_db -f < fs_name | all > -n <node> -d <directory>

lustre_util distribute_coherency -f < fs_name | all > -n <node> -d <directory>

lustre_util set_ost_rank -f < xml file >

lustre_util check_storage

lustre_util show_tuning

lustre_util show_cfg

lustre_util show_conf

lustre_util list

ACTIONS

set_cfg	Copies **/etc/lustre/lustre.cfg** to OSS and MDS nodes.
install	Checks if filesystems can be installed, and then install them.
update	Updates settings of an installed filesystem that do not require reformatting of **previously** formatted OSTs/MDT (New OSTs, different network type, etc).
fsck	Runs e2fsck on the OST/MDT. The filesystem must be offline.
chk_dev	Check the devices and their links on I/O nodes.
rescue	Makes a filesystem usable again by formatting OSTs that are assumed to be NOT correctly formatted.

tune_servers	Set the I/O schedulers of filesystems devices regarding **lustre.cfg** content. Apply the server related tunings of **tuning.conf** on OSS/MDS.
start	Makes installed filesystems available for mounting.
mount	Mounts filesystems on specified nodes.
umount	Unmounts filesystems on specified nodes.
fs_status	Updates and prints OSS and MDS status information.
mnt_status	Updates and prints information regarding client nodes.
status	Does **fs_status** AND **mnt_status**.
stop	Disables filesystems.
remove	Removes filesystems.
info	Prints information (mdt,ost,path,etc.) about filesystems.
short_info	Prints information (mdt,ost,path,etc.) about filesystems, but sort OSTs by nodes.
Lfsck	Builds mdt,osts lfsck database and distributes coherency checking of a Lustre filesystem.
build_mdt_db	Build mdt lfsck database (first step of lfsck).
build_ost_db	Build osts lfsck database (second step of lfsck).
distribute_coherency	Distributes coherency checking of a Lustre filesystem (third step of lfsck).
set_ost_rank	Set the rank of the OSTs in the cluster database regarding the rank they have in the XML.
check_storage	Check the consistency of the storage.conf or tables lustre_ost/lustre_mdt.
show_tuning	Display tuning parameter (from tuning.conf).
show_cfg	Display lustre.cfg variable.
show_conf	Display the lustre_util parameter.
list	Prints name of installed filesystems or those filesystems which failed to be installed.

OPTIONS

-f < filesystem path >

Filesystems to work with, this can be:
- For an install and update: Filesystem path MUST lead to an lmf file (lustre model file) or an xmf file (extended model file).
- For other operations that require the -f option, the filesystem path MUST be only the name of an installed (or attempted to be installed) filesystem.
- "all" stands for all installed filesystems.

-n < nodes >

-n <nodes_list>

Applies the command to the nodes list using pdsh syntax: name[x-y,z],...,namek.

-n all.

For mount, stands for "all clients which have mounted this fs as least one time". For umount, stands for "all clients which currently mount this fs".

-n recover .

For mount, stands for "all clients which are assumed to mount this fs". The main purpose of recover is to mount Lustre clients after a cluster emergency stop (main failure). Clients will be mounted with the same options as their previous mount.

--mount <[+]opt1,opt2,...>

This allows mount options to be specified. For example, if +bar is specified for a filesystem which has foo as a default mount option, mount will be run on the client with -o foo,bar options. If only bar is specified (without +), mount will be run with -o bar options.

--lconf <options>

This provides additional options to lconf for install, update, rescue, start and stop to be used.

-p <rms_partition>

Applies the command to the configured nodes of this running rms partition.

-F

Forces commands execution even though this may be dangerous (no user acknowledgement is asked for).

-t <time_in_second>

Sets the limit on the amount of time a command is allowed to execute. 0 means no timeout.

-u <user>

User name to use to log onto the nodes instead of root.

--fanout

Number of ssh connections allowed to run at the same time, default is 128.

--kfeof	Stands for "keep formatting even on failure". Without this option, lustre_util returns as soon as the first failure is detected while formatting. With this option, lustre_util returns only when all the formatting attempts return for all devices. This can be useful when formatting a large pool of devices. This way you can check the health of all the devices in one shot, and you do not have to reformat devices that succeed in being formatted in a previous pass (using lustre_util update).
-V	Verbose output.
-v	Print version and exits.
-h	Print help and exits.

set_cfg: Distributing /etc/lustre/lustre.cfg

This file must be copied on every OSS and MDS nodes. You can do it using the **set_cfg** command:

```
lustre_util set_cfg [ -n <I/O nodes list > | -p <I/O nodes rms partition> ]
```

If no node parameter is specified, this command copies **/etc/lustre/lustre.cfg** of the Management Node on the nodes that host OST and/or MDT. If nodes are specified, **lustre.cfg** will be only copied on those nodes. If LUSTRE_SNMP is set to "yes", and if the variable **disable_chkconfig_for_ldap = no**, snmp server will be enabled on (selected) I/O nodes. If LUSTRE_LDAP_URL is set to a server address, this server will be enabled.

info: Printing Information about Filesystem

```
lustre_util info -f < lmf,xmf,xml files or fs_name | all >
```

This command will print information about the file system descriptor you specify. If you specify only a file system name, this fs must be installed and information will be retrieved from the cluster database.

short_info: Printing Information about a Filesystem

```
lustre_util short_info -f < lmf,xmf,xml files or fs_name | all >
```

Same purpose as the previous command but displays OST sorted by nodes.

install: Installing a lustre Filesystem

```
lustre_util install -f <lmf or xmf path> -V [ --kfeof ] [ --lconf <options>]
```

This command formats the storage devices and performs operations required to install the filesystem such as loading filesystems information into **ldap** database and/or cluster database. If **-F** is used, no user acknowledge is required. If **-F** is not specified, user must enter "yes" to go on if a filesystem with the same name is already installed. xml files required by CFS Lustre tools (lconf) are automatically generated and copied where it is required. An xmf file is also automatically generated in LUSTRE_CONFIG_DIR for each filesystem.

Note:

This operation is quite long, **-V** (be verbose) option is recommended.

start: Enabling a Lustre Filesystem

```
lustre_util start -f fs_name -V [ --lconf <option>]
```

This command enables a filesystem and makes it available for mounting (online). Use of -V option (be verbose) is recommended.

mount: Mounting Lustre Filesystem

```
lustre_util mount -f fs_name -n <nodes|all|recover> | -p <rms_partition>
 [ --mount < [+]options> ]
```

This command will mount the filesystem on specified nodes using the mount-path defined in the model file. If this mount-path does not exist, it is automatically created. It is an error if this path is already used to mount another filesystem. If --mount is not specified, fs will be mounted with options defined in model file by mount_options. If you use --mount with a parameter which starts with +, fs will be mounted with default options AND with those you give to --mount. If the parameter does not start with +, fs will be mounted with only those you give to --mount.

umount: Unmounting Lustre Filesystem

```
lustre_util umount -f fs_name -n <nodes|all> | -p <rms_partition>
```

This command unmounts the filesystem on specified nodes. You can use the **-n all** option if you want to unmount the filesystem everywhere it is mounted. If umount fails because some processes have their working directories in the mount-path, use umount again with -F option, in order to kill such processes before the umount operation.

stop: Disabling a Lustre Filesystem

```
lustre_util stop -f fs_name [ --lconf <option>]
```

This command disables a filesystem. It will not be available for mounting any more (offline).

set_iosched: Set the I/O Schedulers of Filesystem Devices

```
lustre_util set_iosched -f < fs_name | all >
```

The main purpose of **set_iosched** is to be used as call-back when migration occurs and to set the I/O schedulers on the nodes where lustre services are restarted. You do not have to use it directly as **lustre_util start** sets the I/O schedulers automatically.

remove: Removing a Lustre Filesystem

```
lustre_util remove -f fs_name
```

This command totally removes the file system. All data will be lost. If **-F** is used, the action is done directly without any need of a user acknowledgement.
If **-F** is not used, the user is prompted and must answer explicitly "yes".

fs_status: Updating Filesystem Status and Printing Filesystem Information regarding OSS and MDS

```
lustre_util fs_status [ -f fs_name ]
```

This command updates the status of OSTs, MDTs, and filesystems. If no filesystem parameters are provided, all installed filesystems are checked. The output appears as follows:

```
FILESYSTEM STATUS

filesystem   Config status   Running status   Number of clts migration
tv2ost8      installed       offline         0            0 OSTs migrated
tv2fs1       installed       online          3            0 OSTs migrated
tv2fs2       installed       online          4            0 OSTs migrated
```

The **config status** can take one of the following values:

not installed — **fs** is not installed (it should never be visible).

loaded but not installed — **fs** information is in a database but **lustre_util install** failed.

Formatting — **lustre_util install** is running.

checking — **lustre_util fsck** is running.

installed: — fs is correctly installed.

not usable: **fs** is not correctly installed, because some devices failed to be formatted or **fsck** failed to repair some devices.

The **Running status** can take one of the following values:

offline **fs** is correctly stopped.

Starting **lustre_util** start is running.

online **fs** is started and OSTs/MDT are healthy.

not correctly started **fs** failed to start, some OSTs or MDT may be offline or unhealthy.

CRITICAL **fs** started, but for unknown reasons, some OSTs or MDT may be offline or unhealthy.

WARNING fs is started, but OSS or MDS may not be reachable and their states cannot be checked (elan can work).

stopping **lustre_util stop** is running.

not correctly stopped: **fs** failed to stop, some OSTs or MDT are still online or are in an unhealthy state.

mnt_status: Updating Clients Status and printing Filesystem Information regarding Clients

```
lustre_util mnt_status [ -f fs_name ] [-n <nodes|all> | -p <rms_partition> ]
```

This command checks if the filesystem is correctly mounted or unmounted on specified nodes. If no node is specified, **mnt_status** gives the status of all client nodes that work with this filesystem. If no filesystem parameter is provided, all installed file systems are checked. The output looks similar to the following:

```
CLIENT STATUS

filesystem    Correctly      should be mounted but   Correctly unmounted
              mounted        are not
tv2ost8       None                                       tv5
tv2fs1        tv[0-2]
tv2fs2        tv[0-2,5]      tv[3-4]
```

status: Updating Status of Servers and Clients, printing Filesystem Information regarding Servers and Clients

```
lustre_util status [ -f fs_name ] [ -n <nodes|all> | -p <rms_partition>  ]
```

This command performs a **fs_status** AND a **mnt_status** operation.

fsck: running e2fsck on OSTs and MDT

```
lustre_util fsck -f fs_name [ --lconf <option>]
```

This command runs e2fsck on OSTs and MDT. It reports if some devices have been repaired, if some nodes need to be rebooted, and also if some devices have unrecoverable errors. This command should be applied to offline file systems.

chk_dev: Check Devices and their links on I/O Nodes

```
lustre_util chk_dev  -f < lmf,xmf,xml files or fs_name | all >
```

This command checks devices information on filesystems I/O nodes:

- If the device exists.
- If the device is managed by **stormap**, it checks if device is up or down.
- If size in MBs is the expected size.

lfsck: Builds mdt,osts lfsck Database and distributes Coherency Checking of a Lustre Filesystem

```
lustre_util lfsck -f < fs_name | all > -n <node> -d <shared_directory>
```

<node> is a client which can mount the filesystem, but the **fs MUST NOT** be mounted when you start to use **lfsck.**

<shared_directory> is a shared directory where the **lfsck** database files will be placed. The I/O nodes and the client node must have read/write access to this directory using the same path.

Note:

The database **lfsck** files can be large, depending on the number of files in the filesystem (10GB or more for millions of files), so ensure there is enough space in the shared directory before using lfsck.

lfsck is to be used **ONLY** when unrecoverable errors have been found on OST devices or when OSTs have been reformatted. It attempts to correct problems such as:

- Inode exists but has missing objects = dangling inode. This normally happens if there was a problem with an OST.
- Inode is missing but OST has unreferenced objects = orphan object. This normally happens if there was a problem with the MDS
- Multiple inodes reference the same objects. This can happen if there was corruption on the MDS, or if the MDS storage is cached and loses some but not all of its writes.

After using **lustre_util lfsck**, you should check **lost+found** in the mountpoint of client.

Using **lfsck** is the same as using **build_mdt_db**, followed by **build_ost_db**, and then **distribute_coherency**.

build_mdt_db, build_ost_db, distribute_coherency : step by step lfsck

```
lustre_util build_mdt_db -f < fs_name | all > -n <node> -d <directory>
lustre_util build_ost_db -f < fs_name | all > -n <node> -d <directory>
lustre_util distribute_coherency  -f < fs_name | all > -n <node> -d <directory>
```

These options are to be used:

To restart an **lfsck** operation which has failed, avoiding the need to restart the process from the beginning. **Lustre_util** will provide information regarding which options should be used and when.

If the directory is not a shared directory and there is a need to copy database files, **lustre_util** will provide information regarding which files should be copied and where.

These operations should be done in done in the following order: **build_mdt_db**, then **build_ost_db**, and then **distribute_coherency**.

update: Update Filesystems already Installed

```
lustre_util update -f fs_name -V
```

This command allows you to update an **ALREADY INSTALLED** and offline filesystem with new settings (that do not require a reformatting of the **ALREADY FORMATED** devices):

- **stripe_count**
- **nettype**
- **generic_client**
- **failover**
- **mdt_mount_options**
- **ost_mount_options**
- **cluster_id**
- **mount_path**
- **quota**
- **quota_options**
- **description**
- **mount_options**
- **ost** (new OST can be added, previous OSTs must also be included and do not forget that their **cfg_status** should be currently "formatted". OSTs that currently have their **cfg_status** set to "format_failed" may be removed).

Update is done by updating the model file or the corresponding extended model file with the new settings. The following settings MUST be the same:

- **mdt**(mdt line of model file must lead to the same mdt, do not forget that the **cfg_status** of the mdt should be currently "formatted")
- **ost** that were previously part of the filesystem and that currently have their **cfg_status** set to "formatted".

- mdt_mkfs_options
- mdt_inode_size
- ost_mkfs_options
- ost_inode_size
- fs_name

Important:

An update operation should only be done on a file system which has been stopped correctly.

If High Availability is in use and if the OSTs are distributed on 2 OSSs that are mutually the failover node of each other then all OSTs must be on their primary location otherwise the update will take a long time.

Once the model file is updated, run:

```
lustre_util update -f <path to modified lmf/xmf>.
```

New OSTs will be formatted, new automatically generated xmf/xml will be copied to the right place, and mdt will be updated (write_conf). Only OSTs that have their **cfg_status** set to "format_failed" before the update may be removed.

Important:

Removing correctly formatted OSTs of a filesystem can cause data loss, Lustre_util will not allow this to be done.

Update can also be used after the installation of a new release of Lustre, if the underlying way of storing information on the **MDT** has changed.

rescue: Try to make the Installed Filesystem Work again

```
lustre_util rescue -f fs_name -V [ --lconf <option>]
```

This command can be used on installed file systems which have stopped:

- If the update failed (may be because new OSTs cannot be formatted)
- If fsck detects devices with unrecoverable errors
- If you lose the XML files
- Or for other reasons.

This command checks which OSTs have been successfully formatted and formats those that are assumed to be not correctly formatted. A new XML file is generated and copied in the right places and the MDT is updated (write_conf). Theoretically, the file system should be usable again, **but data may be lost**.

set_ost_rank : set OST rank in Cluster Database regarding the XML file

```
lustre_util set_ost_rank -f < xml file >
```

This option is only used if there has been an upgrade from a release of cluster database that does not have rank field in the **lustre_ost** table to a release that includes this field. The new field is updated with OST information found in the XML file.

check_storage : Checking Consistency of storage.conf or lustre_ost/lustre_mdt Tables

```
lustre_util check_storage
```

The main purpose of this option is to check if **storage.conf** has been correctly completed by the administrator. It should not be necessary to use this if a cluster database is used, however, this option can be available if required.

show_tuning: Display the Tuning Parameters

```
lustre_util show_tuning
```

Display the tuning parameters according to the content of **/etc/lustre/tuning.conf**.

show_cfg: Display lustre.cfg Variable

```
lustre_util show_cfg
```

Display lustre.cfg variable.

show_conf: Display lustre_util Configuration

```
lustre_util show_conf
```

Display lustre_util configuration, according to the content of **/etc/lustre/lustre_util.conf**.

list: Gives the List of Installed Filesystems

```
lustre_util list
```

This command prints the name of the filesystems which are installed, even if their installation is not yet complete.

Note:

An example of the complete process to create and install a Lustre file system is described in *Bull HPC BAS4 Installation and Configuration Guide.*

4.6.4 lustre_util Configuration File /etc/lustre/lustre_util.conf

This file contains some additional settings for **lustre_util.** The following values are set by default:

```
ssh_connect_timeout=20
```

This is the timeout in seconds given to the **connect_timeout** parameter of SSH.

```
install_timeout=0
```

This is the timeout in seconds for install, update and rescue operations and can be overwritten by the **-t** option.

```
start_timeout=0
```

Timeout in **s** for the start operation and can be overwritten by the **-t** option.

```
mount_timeout=60
```

Timeout in **s** for the mount operation and can be overwritten by the **-t** option.

```
umount_timeout=60
```

Timeout in **s** for the umount operation and can be overwritten by **-t** option.

```
stop_timeout=0
```

Timeout in **s** for the stop operation and can be overwritten by the **-t** option.

```
status_timeout=30
```

Timeout in **s** for **status, fs_status, mnt_status** operation and can be overwritten by the **-t** option.

```
set_ioscheds_timeout=60
```

Timeout in s for setting I/O schedulers on I/O nodes (in start and tune_servers operation), can be overloaded by -t option.

```
set_tuning_timeout=60
```

Timeout in s for applying tuning parameters on I/O nodes (in start,tune_servers and mount operation), can be overloaded by -t option.

```
 disable_nagios=no [yes]
```

yes will disable the update of the nagios pipe by **lustre_util.**

```
disable_chkconfig_for_ldap=yes [no]
```

yes will disable the **chkconfig** of **ldap** service in the **set_cfg** operation, **no** will allow this operation. It should be set to yes if administration node is an HA node.

```
use_stormap_for_chk_dev=yes [no]
```

If **yes**, **lustre_util** will check health of devices using **stormap -l**. It should only be set to no if stormap is not installed on I/O nodes. It is not a problem if devices you are using are not managed by **stormap.**

```
allow_loop_devices=no [yes]
```

Unless you explicitly want to use loop device, this should be set to no. This way, it prevents lconf to create huge loop devices in /dev/ directory when some LUNS disappear.

```
check_only_mounted_nodes_on_mnt_status=no [yes]
```

If set to yes, only nodes that are assumed to mount a filesystem will be checked on **status** and **mnt_status** operation.

```
default_fanout=128
```

Number of **ssh** connexions allowed to run at the same time. Can be overloaded using - -fanout option.

4.6.5 Lustre Tuning File /etc/lustre/tuning.conf

This file contains tuning parameters. The syntax is the following:

"<string>" <file> <target> [<delay>] [<filesystems>]

"<string>" String to write in file, it can contain spaces, MUST be between double-quotes

<file> Full path to the file where string will be written. Globbing is allowed. 2 macros can be used:

${mdt} stands for the name of the **mdt** of the filesystem.

${ost} stands for the name of ALL the **osts** (one line will be generated for each ost).

<target> A string composed of the **OSS, MDS**, or **CLT**, separated by semicolons. **OSS**, **MDS** and **CLT** can be followed by a nodes list (**pdsh syntax**) using colon.

<delay> A time in ms that we have to wait before continuing setting tuning parameters on a node. This is an optional argument, and the default is 0 ms.

<filesystem> A list of filesystem separated with semicolons. This is an optionnal argument, and the default is to allow this tuning for every filesystems.

For OSS and MDS, tuning parameters are set when a filesystem is started. For Clients, tuning parameters are set when the filesystem is mounted, for example:

- **"1" proc/sys/lnet/panic_on_lbug OSS;MDS;CLT**
  This line will enable panic on **lbug** on **ALL** types of node for all filesystems by running **echo "1" >proc/sys/lnet/panic_on_lbug** on all nodes.
- **"0" /proc/sys/lnet/panic_on_lbug OSS:ns[5-6];MDS:ns3 fs1;fs2**
  This line will disable panic on **lbug**:
  - on ns5 and ns6, if they are used as an OSS of **fs1** and/or **fs2,**
  - on ns3, if it is used as MDS of **fs1** and/or **fs2.**

String, file and target can be aliased using the following syntax:
alias <name>=<content>

alias can be declared anywhere in the file, but it also acts on the **WHOLE** file, not only on the lines that follow the declaration.

When you use **alias** on a string, the alias must also be in double quotes.

Example:

A **tuning.conf** example file is shown below:

```
####  ALIAS DECLARATION ###################################

alias health_check=/proc/fs/lustre/health_check
alias panic_on_lbug=/proc/sys/lnet/panic_on_lbug
alias ping_osc=/proc/fs/lustre/osc/*${ost}*/ping
alias debug=/proc/sys/lnet/debug

#### TUNING PARAMETER ###################################

"1"                    ping_osc        CLT

"0"                    panic_on_lbug   CLT

"0"                    panic_on_lbug   OSS;MDS

"524288"               debug           OSS;MDS;CLT
```

4.6.6 Lustre Filesystem Reconfiguration

This procedure allows you to change the distribution of the Lustre services which are defined on the I/O nodes, without having to re-deploy (which involves configuring the DDN storage systems and High Availability). The filesystems involved in the new distribution are stopped; the others continue to be operational.

The following example describes how to stop the `fs1` and `fs2` filesystems.

1. If needed save the data of the `fs1` and `fs2` filesystems.

2. Unmount the `fs1` and `fs2` filesystems:

```
lustre_util umount -f fs1 -n all [-F]
lustre_util umount -f fs2 -n all [-F]
```

3. Stop the `fs1` and `fs2` filesystems:

```
lustre_util stop -f fs1 [-F]
lustre_util stop -f fs2 [-F]
```

4. Remove the `fs1` and `fs2` filesystems:

```
lustre_util remove -f fs1
lustre_util remove -f fs2
```

5. Make the required modifications in the models associated with the filesystems. In our example `fs1` and `fs2` are grouped together in only one `fs3` filesystem.

6. Configure the new `fs3` filesystem (this operation erases the `fs1` and `fs2` filesystems data).

```
lustre_util install -f /etc/lustre/model/fs3.lmf
```

7. Start the new `fs3` filesystem:

```
lustre_util start -f fs3
```

8. Mount the new `fs3` filesystem:

```
lustre_util mount -f fs3 -p p2
```

9. If needed, restore the saved data.

4.6.7 Using Quotas with Lustre File Systems

4.6.7.1 Quota Settings in Model Files

Quotas are enabled by setting "quota" to "yes" in lmf file:

```
quota: yes
```

The default quota options are as follows:

```
quota_options: quotaon=ug,iunit=5000,bunit=100,itune=50,btune=50
```

quotaon=<u|g|ug> Enable quota for user|group|user and group.

iunit=<number of inodes> **iunit** is the granularity of inodes quotas. Inodes are acquired and released by a slice of inuit. iunit is a int type (>0), the default value in Lustre is 5000 inodes.

bunit=<size in MB> — **bunit** is the granularity of block quotas. Blocks are acquired and released by a slice of bunit MBs on each OSTs. bunit is expressed in MBs (>0), the default value in Lustre is 100 MBs.

itune=<percentage> — **itune** sets the threshold to release and acquire iunit inodes. For example, if a user/group owns n*iunit+m inodes, iunit inodes will be acquired for this user as soon as m goes above itune*iunit/100. If a user/group owns n*iunit-m inodes, iunit inodes will be released for this user/group as soon as m goes above itune*iunit/100. itune is a int type (100 > itune > 0), the default value in Lustre is 50.

btune=<percentage> — **btune** sets the threshold to release and acquire bunit block MBs for each OST. For instance, if a user/group owns n*bunit+m MB on one OST, bunit MBs will be acquired on this OST for this user/group as soon as m goes above btune*bunit/100. If a user/group owns n*bunit-m MBs on one OST, bunit MBs will be released on this OST for this user/group as soon as m goes above btune*bunit/100 MB. btune is a int type (100 > btune > 0), the default value in Lustre is 50.

4.6.7.2 Starting Quota: lfs Quotacheck

Once the filesystem is installed, started and mounted, run the following command on a client:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

This means that if quota_options are as follows:

`quotaon=ug,iunit=5000,bunit=100,itune=50,btune=50 and mountpoint is /mnt/lustre,`

it will be necessary to run:

```
lfs quotacheck -ug /mnt/lustre
```

The time taken by **quotacheck** depends on the size of the biggest device used by the filesystem as OST or MDT. On average, it takes 160s for a 1TB OST/MDT check.

4.6.7.3 Setting the Limits: lfs Setquota

lfs setquota sets limits on blocks and files.

```
lfs setquota [-u|-g] <name> <block-softlimit> <block-hardlimit> <inode-softlimit>
 <inode-hardlimit> <mount_point>
```

block-softlimit and **block-hardlimit** are expressed in kB.

Inode-softlimit and **inode-hardlimit** are expressed in number of inodes.

Limits on blocks/inodes MUST be greater than **bunit/iunit**. This means, for example, **bunit=100MB, block-softlimit** and **block-hardlimit** must be greater than 102400kB. If you have **iunit=5000**, **inode-softlimit** and **inode-hardlimit** must be greater than 5000.

Limits on blocks must be greater than the number of OST * bunit. This means, for example, if there are 9 OSTs and **bunit**=100 MBs, **block-softlimit** and **block-hardlimit** must be greater than 9*100*1024=921600 kBs.

For example:

```
lfs setquota -u bob 900000 1000000 5000 10000 /mnt/lustre
```

will set a **block-softlimit** to **900MB**, **block-hardlimit** to **1GB**, **inode-softlimit** to **5000**, **inode-hardlimit** to **10000** for user **testfs**, for a lustre filesystem mounted on /mnt/lustre.

```
lfs setquota -g dba 900000 1000000 5000 10000 /mnt/lustre
```

The command above will implement the same settings for all users of group **dba.**

Restrictions

- At present, soft limits are not supported in Lustre. So set **block-softlimit** and **inode-softlimit** to 0.
- It is strongly recommended to run **setquota** on a Lustre file system which is not busy. Otherwise an incorrect **block- hardlimit** value may be set.

4.6.7.4 Updating/Rescuing a Filesystem with Quota enabled

If a filesystem is rescued, quota will have to be enabled again using the command below.

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

If a filesystem is updated and new OSTs are not added the following command will have to be run again:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

If a filesystem is updated and **new OSTs are added** then the fs will have to be updated, started and mounted and then run the following command:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

For ***ALL*** groups and users, all the limits may be set to 0 with the following command:

```
lfs setquota -u <user> 0 0 0 0 <mount_point>
lfs setquota -g <group> 0 0 0 0 <mount_point>
```

For ***ALL*** groups and users, the limits may be set to their former values with the following command.

```
lfs setquota [-u|-g] <name> <block-softlimit> <block-hardlimit> <inode-
softlimit> <inode-hardlimit> <mount_point>
```

4.7 Monitoring Lustre System

Status information about the Lustre file system and I/O nodes is kept up to date in the ClusterDB by the Lustre management tools.

Using this information and that collected by performance daemons, the NS-Master HPC Edition supervision tool offers items specific to the Lustre system allowing the health and performance to be monitored from the management station – see the chapter on monitoring for more details.

4.7.1 Lustre System Health Supervision

4.7.1.1 The all status Map view

This includes global status indicators which provide the administrator with information about the global I/O system availability.

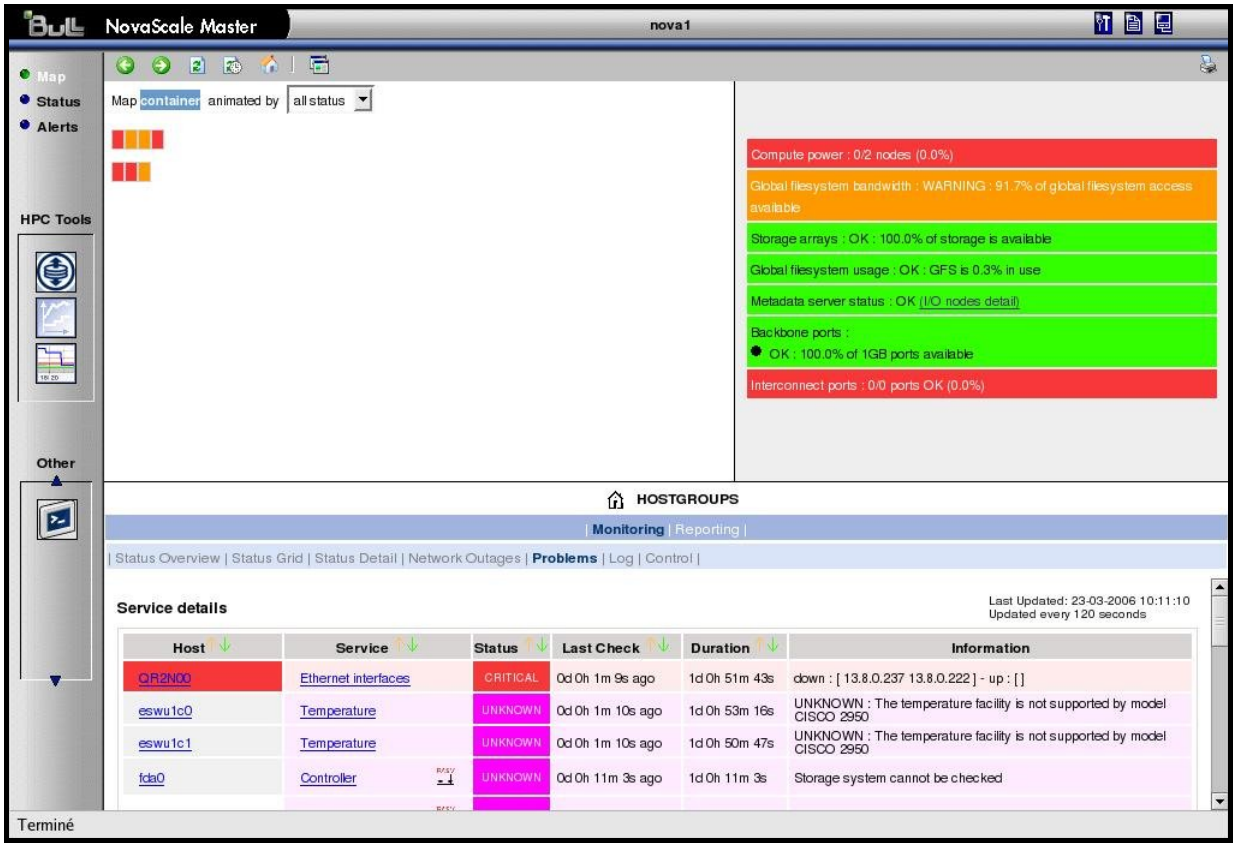


Figure 4-1. NovaScale Master Map view

System Availability Indicators are located at the right top of the topological view to give a status to the administrator at-a-glance. These include:

Available Global File System Bandwidth as a Percentage

This is indicated as a percentage of I/O nodes available. An I/O node is fully available if it has its three Quadrics rails and its height fibre links up and if its Lustre status is OK. If not, a degradation factor is applied as follows:

- **cancel** the node if Lustre is not OK
- **apply** a 30% factor of degradation per quadrics rail missing
- **apply** a 12% factor of degradation per fibre link missing

Available Storage Arrays as a Percentage

The ratio of running storage appliances (DDNs) against the total number is indicated.

Global File System Usage

This gives the current usage rate of the Lustre system for all the Lustre file systems together.

MDS Migration Alert

If High-Availability is configured, this alerts the administrator to a MDS failover migration. The Lustre system then no longer has the High-Availability status.

4.7.1.2 Filesystems Health Monitoring

This is done by the script **/usr/bin/lustre_fs_nagios**. It checks the state of each OSTs/MDTs, and sets the status of the filesystems into the ClusterDB according to whether they are online or offline. This script is called every 15 min on the Management Node using **/etc/cron.d/lustre_fs_nagios.cron**, which is automatically installed and enabled by lustre_utils RPM.

lustre_fs_nagios should not be used online by the administrator; however, it can be used to force a refresh of **nagios** lustre filesystem status entry.

4.7.1.3 The lustre_check Tool

The **lustre_check** tool keeps the I/O node availability information up to date in the **ClusterDB**. It runs on the management station, scheduled by a **cron** every 15 min.

When called, it checks the I/O nodes to collect the network and storage information. This information is stored for each node in the **lustre_io_node** table of the database where it is regularly scanned by the supervision tools.

The **lustre_check** tool is not likely to be used on line by the administrator; however, it can be used to force a refresh of the ClusterDB information and to get an instant status displayed node by node.

4.7.2 Lustre Filesystem Indicator

Within NovaScale Master the Nagios service plug-ins include a plug to monitor the health for the Lustre file system.

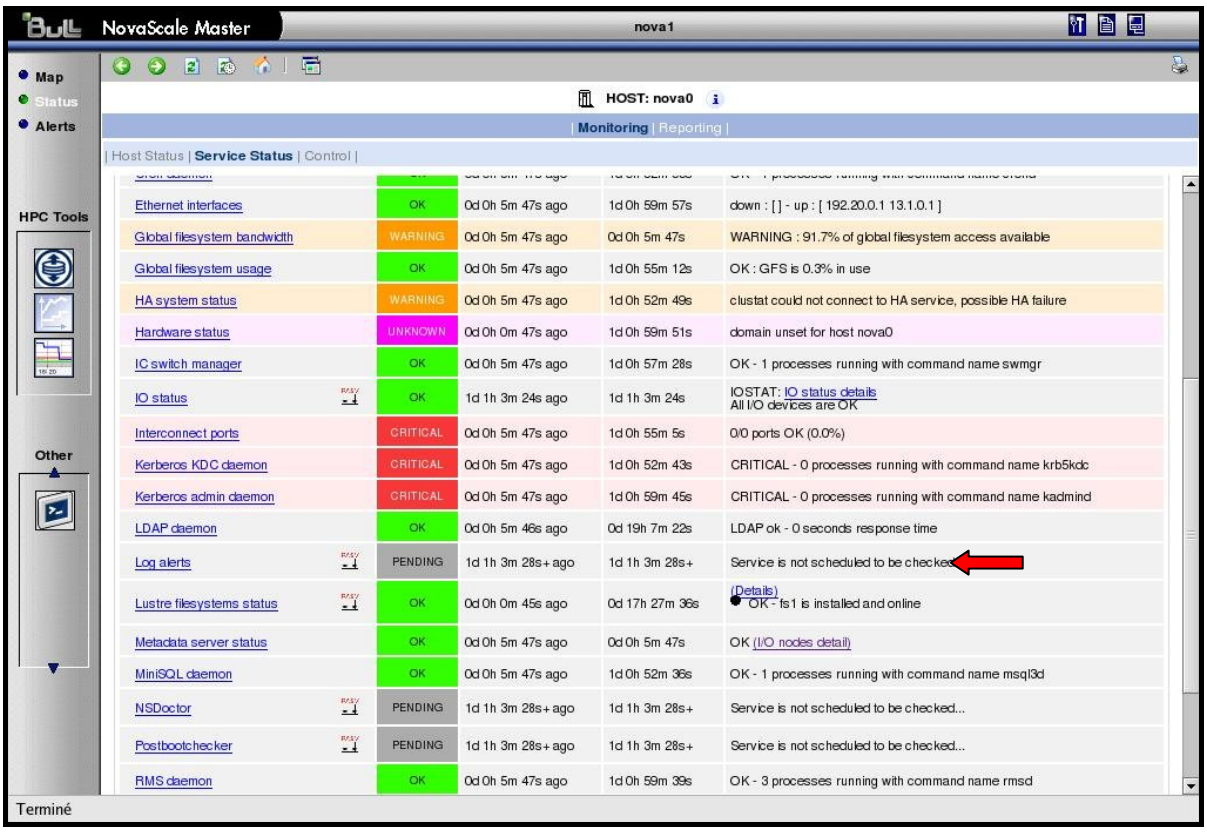


Figure 4-2. NovaScale Nagios file system indicator

The Lustre file system indicator relates to the Lustre file systems health as a whole. Clicking on the info link will displays a detailed status for each file system running.

Lustre Management Node Web Interface

With a web browser, you can easily check the Lustre filesystem status using the following URL: http://<mangement node>/lustre

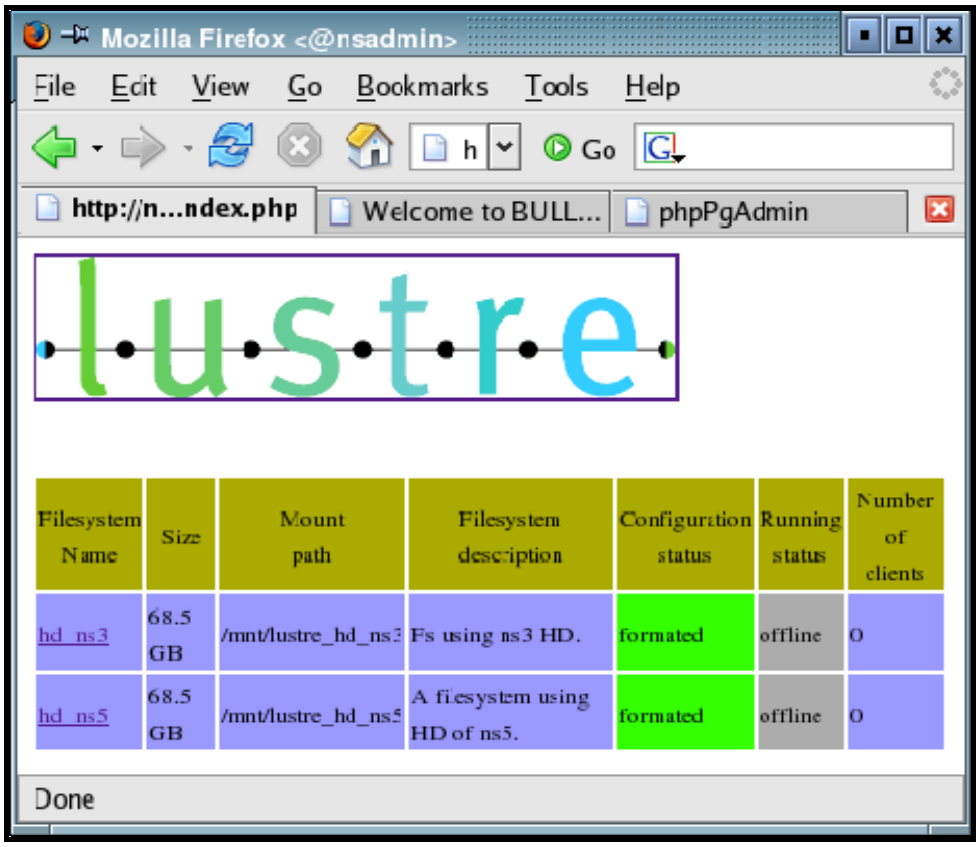


Figure 4-3. Lustre Management Node web interface

By clicking on the filesystem name, you can get details about the filesystem, using an interface that allows you to sort OSTs by name, active node, primary node, secondary node, device size, journal device, Config status, status or migration status.

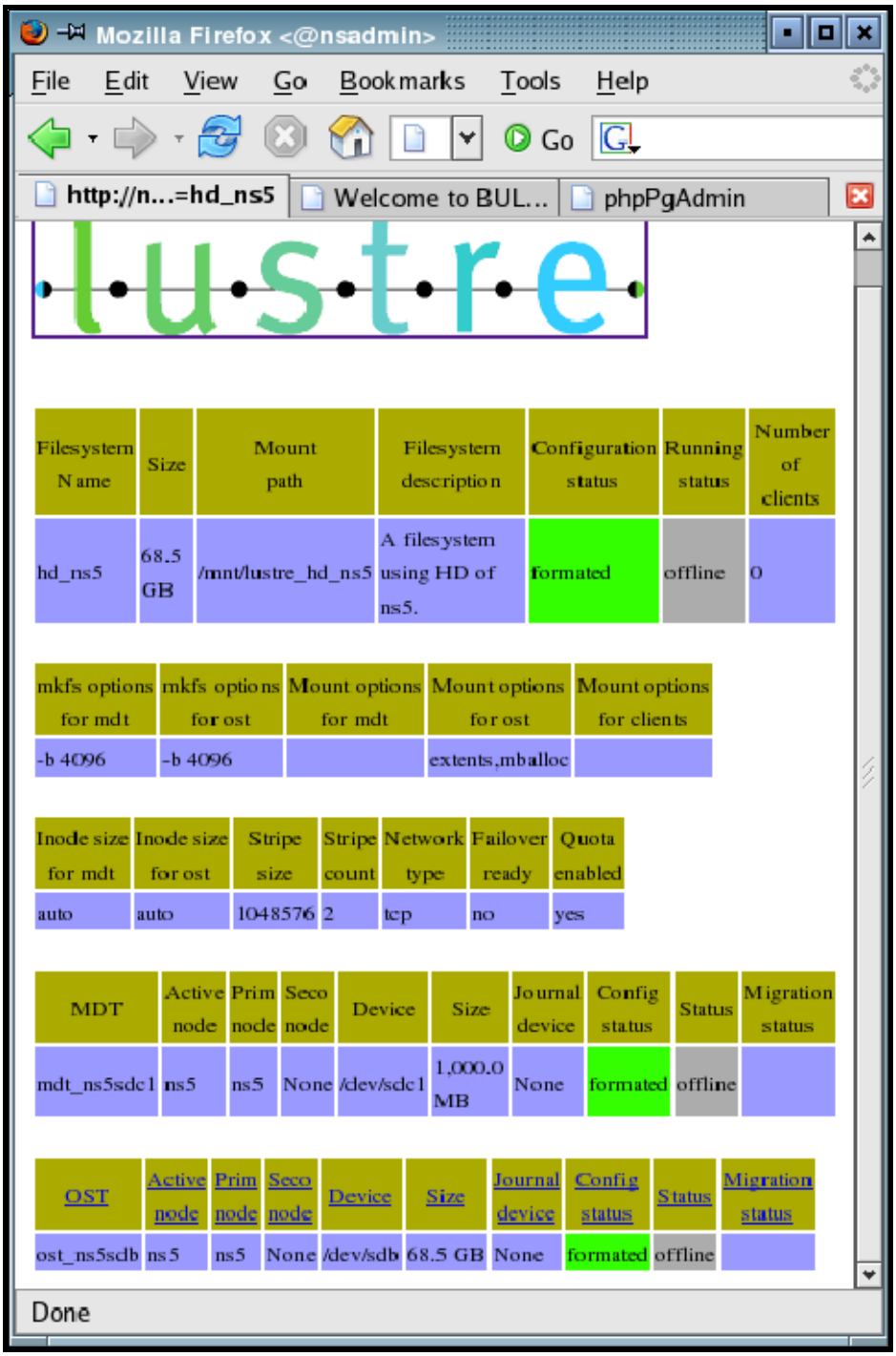


Figure 4-4. Detailed view of Lustre file systems

4.7.3 Lustre System Performance Supervision

4.7.3.1 Group Performance Views

By clicking on the Group performance button in the NovaScale Master console the administrator is provided with an at-a-glance view of the transfer rates of the Lustre system for the file systems all together. The information period can be specified.

Clicking on the compiled view will display a dispatched view giving the performance rates node by node for the same time period.

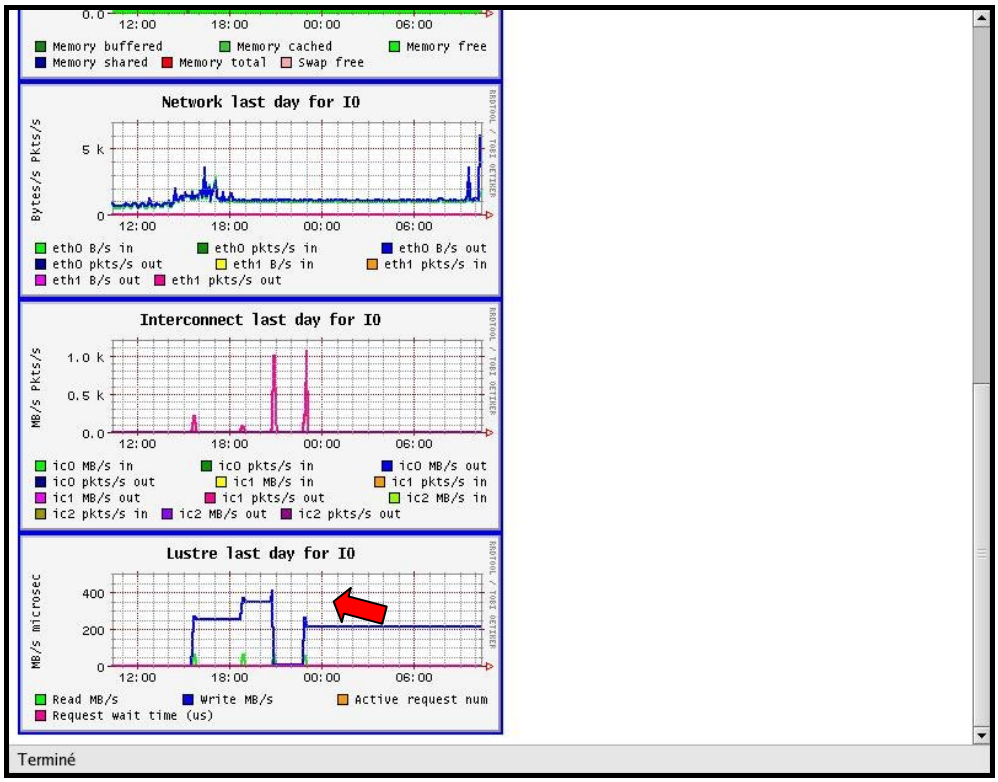


Figure 4-5. Group performance global view pop up window

Figure 4-6. Dispatched performance view pop up window

4.7.3.2 Node Performance Views

Views related to Lustre system local transfer and filling rates are available for each I/O node from the Global Performance view in the Nova Scale Master Console.

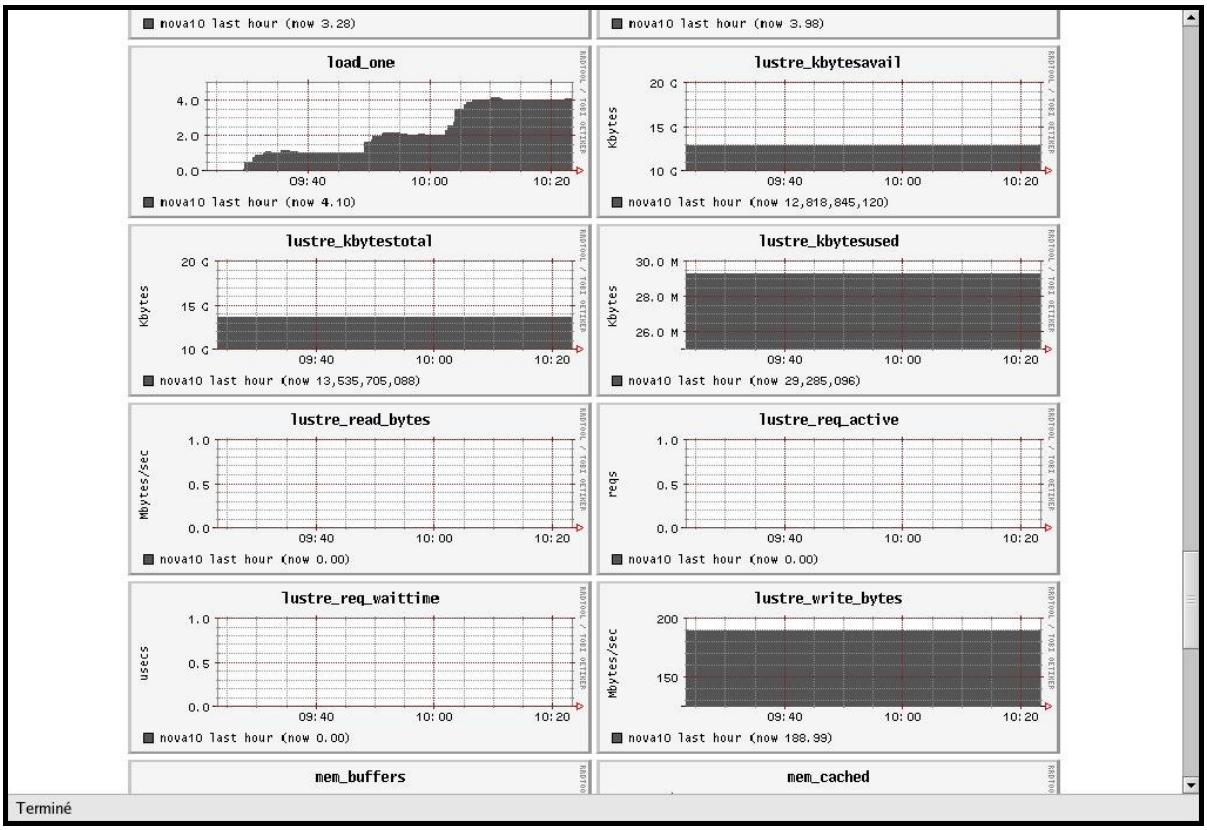


Figure 4-7. Global performance view pop up window

Chapter 5. Software Deployment (KSIS)

This chapter describes how to use KSIS to deploy, manage, modify and check software images.

The following topics are described:

5.1 Overview

A deployment tool is a piece of software used to install a distribution and packages on several machines at once. For large clusters, such a tool is essential, since it avoids doing the same installation a large number of times. **KSIS** is the deployment tool used on Bull HPC systems.

KSIS makes it easy to propagate software distributions, content or data distribution changes, operating system and software updates, for a network of Linux machines.

KSIS is used to ensure safe production deployments. By saving the current production image before updating it with the new production image, a highly reliable contingency mechanism is provided. If the new production environment is found to be flawed, simply roll-back to the last production image.

This chapter describes how to:

- Create an image for each type of node and save it on the image server. These images are called reference/golden images. The image server is on the Management Node and is operated by the KSIS server software.
- Deploy the node images.
- Manage the evolution of the images (**workon** images and patches).
- Check discrepancies between an image on a node and its reference on the image server.

Note:

The terms **reference node** and **golden node** are interchangeable. The same applies to the terms **reference image** and **golden image**.

The deployment is done using the administration network.

5.2 Configuring and Verifying a Reference Node

A reference node is a node which has had all the software installed on to it and from which the image is taken and stored on the image server. The reference image will be deployed onto the other nodes of the HPC system.

Installation and Configuration

Reference nodes have the BAS software installed on to them in the same way as ordinary compute or I/O nodes. A **KSIS client** is then installed onto these nodes using the **Cluster Management CD**. The operating system and applications must be installed and configured to make the node operational.

5.3 Main Steps for Deployment

Once the image server, reference nodes and client nodes are ready, the steps for the deployment are:

1. Create the image of the reference node to be saved on the Image Server:

```
ksis create <imageName> <ReferenceNodeName>
```

This command requests that a check level is chosen. Choose “basic”.

2. Deploy the image:

```
ksis deploy <imageName> node[1-5]
```

The following figure shows the creation and deployment of an image.

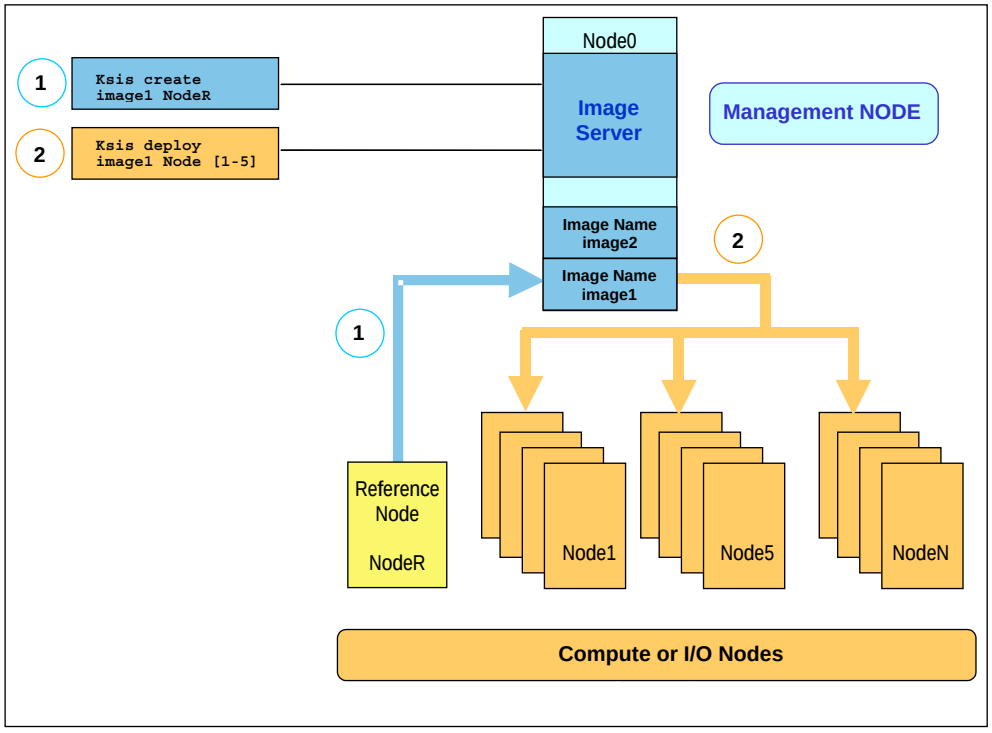


Figure 5-1. Main steps for deployment

5.4 Modifying Images and Managing their Release

5.4.1 Methods

There are two different methods to modify an image.

1. You can modify the image on the reference node and apply the process described in *5.3 Main Steps for Deployment* (creating and deploying an image).

Note:

This method is safer and should be used when the modifications are complex or when there are a great number of invoked files.

2. Alternatively use the **workon** mechanism, which consists in directly altering an image on the image server; for instance modifying a configuration file.
   The **ksis workon** command means that it is possible to "log" on the image and to modify it. This command opens a working environment where the image can be modified using the shell provided.

Note:

Environmental modifications are limited to file modifications which are seen when working on a filesystem and not on the nodes which are in use.

When the modifications are finished create a patch containing the modifications using the **ksis store** command.

Example:

```
ksis workon image1
ksis store image1.s1.0
```

It is possible to deploy and apply this patch on the specified nodes without having to deploy the whole image. The **ksis deploy** command deploys the patch.

Example:

```
ksis deploy image1.s1.0 nc[2-45]
```

The **ksis undeploy** command is used to remove the last patch from the nodes specified. This only works if the node has not been completely altered by the patch. For example, **ssh** must be available on the node.
Example:

```
ksis undeploy image1.s1.0 nc[2-6]
```

It is possible to create a new complete image of the node with the applied patch using the **ksis detach** command.

The **ksis list** command creates a list of images and patches available on the image server with their status.

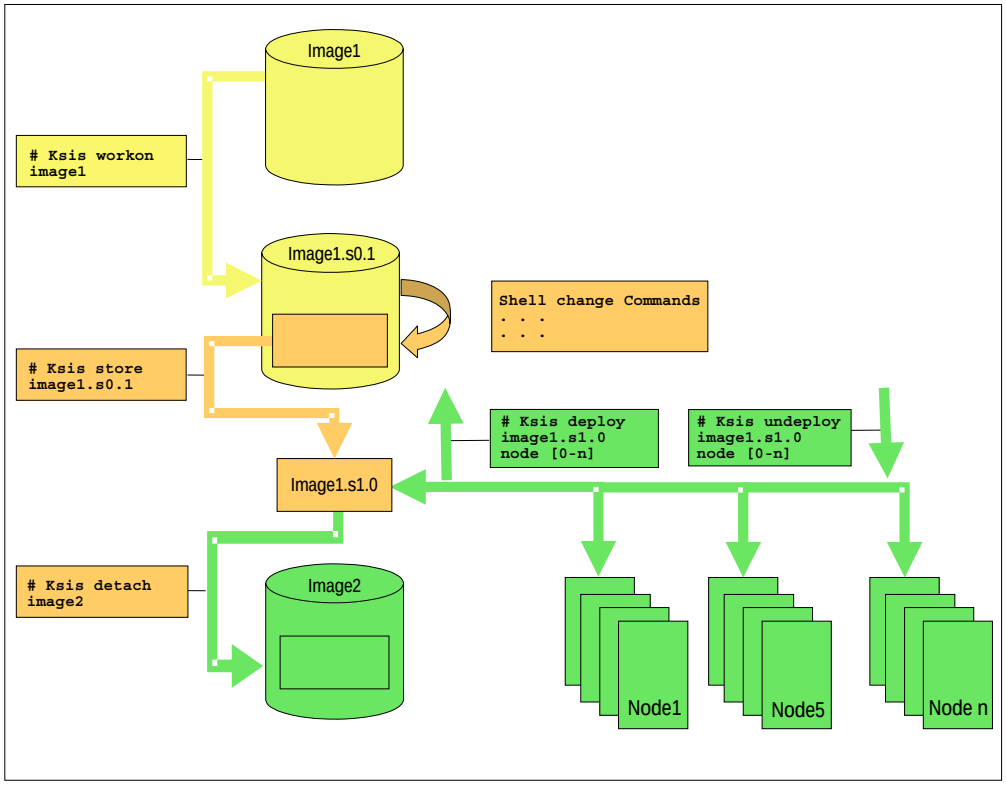


Figure 5-2. Image modification (workon, store, deploy, detach)

5.4.2 Naming Images or Patches with the Workon Mechanism

The image or patch derivation process follows these rules:

- Only one derivation is possible except at level 1.
- The user can define the name of the *golden* and *patched golden* images only at the time that they are created.
- The name of the patch and working patch image is the name of the mother image suffixed by **.sX.Y** where **X** refers to the patch branch number X, and **Y** refers to the patch number Y on this branch.

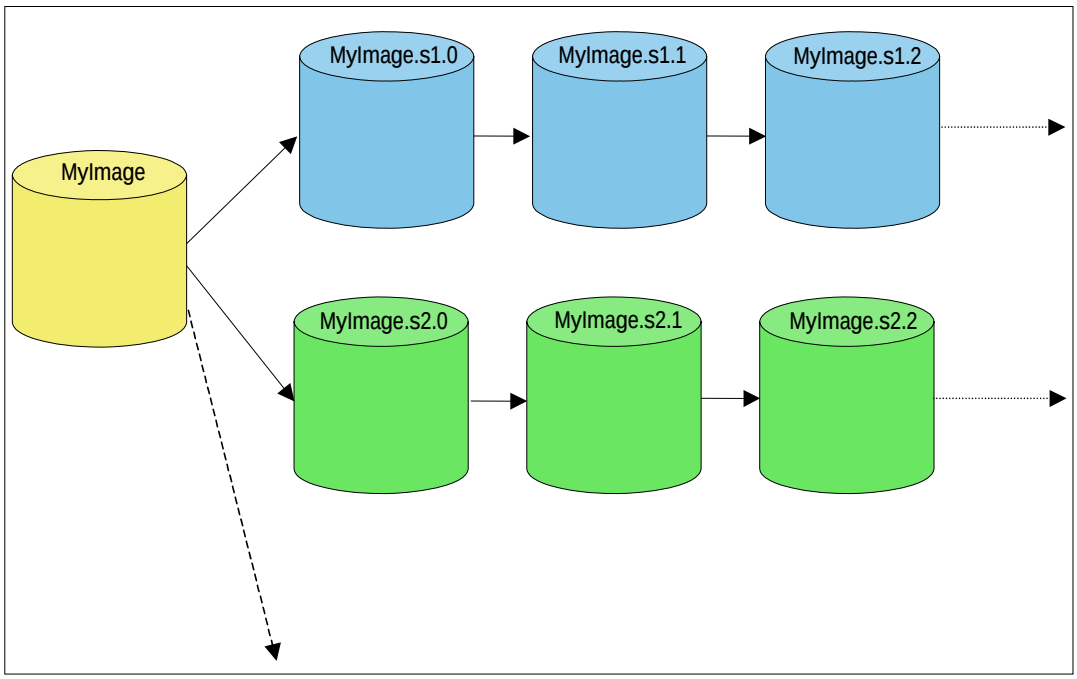


Figure 5-3. Names of derived images or patches

5.4.3 Image Types

There are four types of images (result of **ksis list** command):

Golden	Image obtained from a reference node.
Patched golden	Modified golden image (resulting from the **detach** command).
Working patch	Patch in progress on this image. Cannot be deployed. Waiting for **store** command.
Patch	Patch (result of a **store** command).

5.5 Checking Deployed Images

The **ksis check** command is used to compare the image deployed on a set of nodes with the corresponding reference image.
This is done by listing the discrepancies which exist for a series of tests performed on the nodes and the results of the same tests on the reference image.
Example:

```
ksis check nc[2-45]
```

Note: Nodes inside a node range are not always created from the same image.

5.5.1 Checking Principles

The descriptions of the image checks are stored in a database. When creating an image or a patch, the administrator specifies the required check level for this image or patch. Then **KSIS** copies the corresponding checking database to the image and executes the command associated with each check and stores the results as a *reference*. This *reference* is then included in the image.

Each time the **ksis check** command is used, KSIS executes the defined checks on each node and generates the results. If there is a discrepancy between the result and the *reference*, the check is set to KO, otherwise it is set to OK. The image server centralizes the results.
In this way, the load for control is spread over the nodes. It is also easy to modify and to add new checks.

5.5.2 Check Groups

According to the chosen level, checks for a given image or patch are extracted from the checks database (**/etc/systemimager/ksis_check_Repository/** on the Management Node) and are executed when the image is created. A check level is a particular check group.

Each check belongs to one or more groups defined in the **group** file inside the check directory. If the **-t** option is not specified all the checks are executed.

The checks belonging to the **skip** group are not run.

Name	Level	Group	OK	KO
CheckRpmList	Basic	rpm, fastrpm	List of RPMs installed on the node is the same on the reference image.	List of RPMs installed on the node is not the same on the reference image.
CheckRpmFiles	Sharp	rpm	None of the files delivered using the RPM seems to have been updated regarding contents and/or access rights.	One or more of the files delivered using RPM seems to have been updated regarding contents and/or access rights.

Name	Level	Group	OK	KO
CheckFastRpmFiles	Basic	rpm, fastrpm	None of the files delivered using RPM seems to have been updated regarding length, date, and/or access rights.	One or more of the files delivered using RPM seem to have been updated regarding length, date, and/or access rights.
CheckSRTDir	Basic	lsall	None of the files of the deployed image seems to have been updated regarding length, date, and/or access rights.	One or more of the files of the deployed image seem to have been updated regarding length, date, and/or access rights.
CheckMd5sumDir	Sharp	md5all	None of the files of the deployed image seems to have been updated regarding content (md5 on the content).	One or more of the files of the deployed image seem to have been updated regarding content (md5 on the content).
CheckMandatoryFiles	Basic	ksis	Ksis binaries are present on the node and have the same length as those on the Management Node.	Ksis binaries are not present on the node or have not the same length as those on the Management Node.
CheckUsedKernel	Basic	kernel	Kernel used by the node is the same as the one used on reference/golden node when the image has been created.	Kernel used by the node does not look the same as the one used on reference/golden node when the image has been created.
CheckEfiBootOrder	Basic	ksis	There is only one "network" entry in the EFI menu.	There is more than one "network" entry in the EFI menu, or it is not the first line.

Table 5-1. Standard checks delivered with Ksis

5.5.3 Modifying the Checks Database

It is possible to modify the database checks to adapt them to the way you use the image.

- To change check groups, edit the **group** file.
- To create a new check, add a new directory (**/etc/systemimager/ksis_check_Repository/<testName>.vid**) which includes at least the following:
  - **command** file, which contains the command to be run,
  - **group** file, which defines the group to which the command belongs.

This check will be included in the checks database and will be part of the checks performed on subsequent images.

5.5.4 Examining the Results

The checks result is a comparison between a command executed on the reference image and the same command executed on the nodes concerned. This comparison shows the evolution of the node against the reference and means that it is possible determine if it is necessary to deploy the node again.

5.5.5 Looking at the Discrepancies

If the discrepancies between a node and the reference image are not significant, it may still be useful to analyze their development. There are several ways to do this.

- The **ksis checkdiff** command displays the discrepancies between the reference image and the results for a given check.

Example:

```
ksis checkdiff CheckSRTDir node2
```

- You can also examine the results for the node:
  - **/etc/systemimager/ksis_check_Repository/** for an image,
  - **/usr/ksisClient/PATCH_<patchName>/ksis_check_Repository/** for a patch (name: <patchName>).

5.6 Importing and Exporting an Image

KSIS provides a function to export an image to another KSIS installation (on another administration node) or to import an image from another KSIS installation.

The **ksis export** command allows you to export a Reference image (not a Patch image). The image will be available as a tar file in the Ksis images directory: **/var/lib/systemimager/images/<imageName>.tar**

```
ksis export <imageName> [<option>]
```

Note:

The export operation does not automatically destroy the exported image.

The KSIS import command allows you to import a Reference image from a tar file in the KSIS images directory: **/var/lib/systemimager/images/<imageName>.tar.**

Once the import operation is completed, the image is available and may be listed by using the **ksis list** command.

The import / export feature can be used to archive images that are no longer used on nodes, but that the administrator wants to keep.

5.7 Working with Secondary Images

KSIS allows you to deploy a second bootable system onto the node, whilst keeping a primary system.

The first system is deployed on the first disk of the node and is known as **primary**. The second system is deployed on the second disk of the node and is known as **secondary**.

Note:

The secondary system cannot be patched or checked by KSIS.

5.7.1 Disk Partitioning Constraints

In order that an image can be developed on the system as a primary or secondary image, it must be built with the following constraints:

- All the system partitions must be on the first disk (**sda**).
- The second disk (**sdb**) must not be used, except for a swap partition.
- The third disk must reserved for the **/tmp** partition.

The following figure shows the partitions of a node deployed with a primary system:

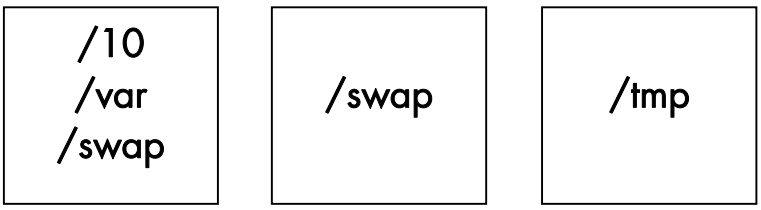


Figure 5-4. Primary system node partitions

The following figure shows the same node on which a secondary image has been deployed:

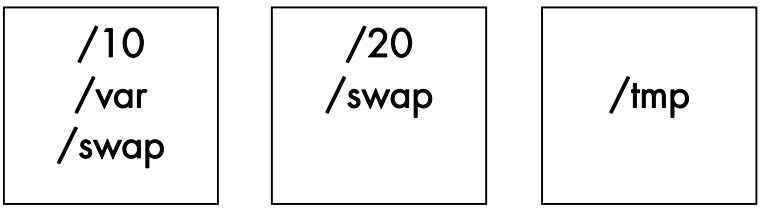


Figure 5-5. Secondary system node partitions

Note:

When a secondary system is deployed, it takes advantage of the swap created by the primary system (the reverse may also apply) and uses the **/tmp** partition for its own account. Consequently, the primary system loses the contents of its **/tmp partition.**

5.7.2 Managing Primary/Secondary Images

A set of KSIS commands allows you to manage both primary and secondary images on the nodes of a cluster.

5.7.2.1 Deploying Primary/Secondary Images

- By default, KSIS deploys the primary system, as in the following command:

```
ksis deploy <imageName> < nodeRangeOrGroupName> [<options>]
```

- The **–2** option allows you to indicate that the image must be deployed as secondary image:

```
ksis deploy <imageName> < nodeRangeOrGroupName> -2 [<options>]
```

- Note that deploying a new primary system will not affect a secondary system already deployed. To deploy an image again as primary, while erasing the secondary, enter:

```
ksis deploy <imageName> < nodeRangeOrGroupName> -m first [<options>]
```

5.7.2.2 Setting Primary or Secondary systems as the Boot Image

- After the deployment of an image in secondary mode, the affected nodes will boot on this secondary system. To allow the nodes to boot on the primary system at the next reboot, enter:

```
ksis setbootmode primary <nodeRangeOrGroupName>
```

- Conversely, to make nodes boot on the secondary system at the next reboot, enter:

```
ksis setbootmode secondary <nodeRangeOrGroupName>
```

Note:

The **ksis setbootmode** command only sets the mode for the next reboot, but it does not reboot the nodes.

- To ascertain the boot mode associated with each node, enter:

```
ksis nodelist
```

The output displays in the **Boot Mode** column either "**p**" (for the nodes configured to boot from their primary image) or "**s**" (for the nodes configured to boot from their secondary image). It also displays the name of the boot image.

5.7.2.3 Listing Primary/Secondary Images

- To ascertain the name of the images deployed as primary system images, enter:

```
ksis nodelist -1
```

- To ascertain the name of the images deployed as secondary system images, enter:

```
ksis nodelist -2
```

- To display the list of the nodes on which a secondary system has been deployed, enter:

```
ksis nodelist where 2inst
```

- To display the list of the nodes configured to boot from their secondary system images, enter:

```
ksis nodelist where 2boot
```

5.8 Ksis Commands

5.8.1 Syntax

```
ksis <action> <parameters> [<options>]
```

Options:

-S Step by step

-v Verbose

-g Debug mode

-G Detailed debug mode

Format for nodeRange or groupName parameter:

The nodes, to which the Ksis command applies, are specified either as a range of nodes (**nodeRange**) or as a group name (**groupName**).

- Several formats are possible for the **nodeRange** parameter, as shown in the following examples:
  - <nodeRange> = host[1]
  - <nodeRange> = host[1,2,3,9]
  - <nodeRange> = host[1-3]
  - <nodeRange> = host[1-3,9]
- The **groupName** is the name of a group of nodes defined in the ClusterDB. See the *Cluster Database Management* chapter for more information about these groups.

Getting Help:

For a complete description of the KSIS commands, enter:

```
ksis help
```

Or:

```
ksis help <action>
```

5.8.2 Advanced ksis create options

-d The **–d** option is used to define the individual disks of a node, which are to be included in the image.

```
ksis create <myImage> <myReferenceNode> -d <myDisks>
```

The disks to be included appear after the **–d** option in a comma-separated list, as shown in the example below. The node disks not listed will not be included in the image.

Example

```
ksis create MyImage MyGolden -d /dev/sda,/dev/sdb
```

In the command above only disks **sda** and **sdb** will be included in the image.

-dx The **-dx** option is used in the similar fashion to the **–d** option. The only difference is that this option is exclusive. In other words, unlike the **–d** option, all the references to the mounted disks which are not included in the image will be deleted and the **/etc/fstab** file which lists the mounts points will be updated.

When to use the –d and –dx options

The **–dx** option is used, for example, if for some reason it is decided that a particular disk bay (e.g. **/dev/sdj**) connected to the reference node, should not be included in an image when it is deployed.

If the **–d** option is used after deployment then the system will try to remount the **/dev/sdj** disk bay on all the deployed nodes. By using the **–dx** option with the **ksis create** command all references to the **/dev/sdj** bay are deleted, and it will not be remounted after deployment.

5.8.3 Creating the Image of the Reference Node

To create an image of the reference node use the **ksis create** command. This operation is done while you are logged onto the image server (Management Node).

```
ksis create <imagename> <reference_node_name> [options]
```

This command creates a copy of the image of the reference node on the image server (Management Node). The resulting status for this image is "golden".

When using this command the check level associated with this image is requested. Choose **basic** for a standard level (see *5.5 Checking Deployed Images* for other options).

5.8.4 Deleting an Image or a Patch

This command deletes the defined image or patch from the image server (Management Node).

```
ksis delete <imageNameOrPatchName>
```

5.8.5 Deploying an Image or a Patch

This step consists in the deployment of an image or a patch on the specified nodes.

```
ksis deploy <imageNameOrPatchName> <nodeRangeOrGroupName> [options]
```

When you deploy an image the command performs these steps on the nodes concerned:

- Checks the state of the node.
- Reboots the node in network mode.
- Loads the image from the image server using special algorithms to parallelize the loading and to minimize the loading time.
- Checks log files.
- Boots the node with the image loaded.

Note:
See Chapter 4 in the Bull *HPC BAS4 Maintenance Guide* for more information on the **Ksis** log files.

5.8.6 Removing a Patch

This action concerns only the images with the "patch" status. It consists in removing the last deployed patch from the nodes.

```
ksis undeploy <patchName> <nodeRangeOrGroupName> [options]
```

5.8.7 Getting Information about an Image or a Node

This command displays information for the specified image or node.

```
ksis show <imageNameOrNodeName>
```

5.8.8 Listing Images on the Image Server

This command gives the list and status of the images available on the image server. Their status is one of the following:

```
ksis list [<options>]
```

golden reference image (from a reference node also called golden node).

patch patch (result of a store command).

patched golden modified reference image (result of a detach command).

working patch modification in progress; cannot be deployed, waiting for store command.

Example:

```
ksis list
```

```
Image Name                Status          Creation Date

BAS3-v13u1u2              golden          2005-01-14 14:33:02
Compute_hpceth_u1u2       golden          2005-01-14 15:41:25
Compute_hpceth_u1u2.s1.0 patch            2005-01-20 13:49:27
Compute_hpceth_u1u2.s1.1 working patch    2005-01-22 14:41:03
```

5.8.9 Listing Images by Nodes

This command lists the current images available and their status on the nodes.

```
ksis nodelist [<options>]
```

Example:

```
ksis nodelist
```

```
nc1  unreach -
nc2  up Compute_hpceth_u1u2         2005-01-20 11:28:30
nc3  up Compute_hpceth_u1u2         2005-01-20 11:29:33
nc4  up Compute_hpceth_u1u2.s1.0    2005-01-21 12:03:01
nc5  down Compute_hpceth_u1u2.s1.0    2005-01-21 12:10:43
```

5.9 Modifying an Image

There are two means of creating a patch to be used to modify an image:

1. Using the **ksis workon** and **ksis store** commands to modify the image outside of the nodes on which it is deployed. These are reserved for minor modifications such as file changes.

2. **ksis buildpatch**, which is used for more complex image modifications.

Important:
ksis buildpatch and the used of patches should only be used for limited image changes. For fundamental image changes the best method remains the creation and the deployment of a new image.

5.9.1 Creating a Working Patch Image

```
ksis workon <ImageNameOrPatchName> [<options>]
```

This **workon** command allows an image to be modified without creating a new one.

The command duplicates the image and creates a workon environment (shell) from which all the modifications required can be performed.

The status for this new image is **working patch**.

5.9.2 Creating a Patch Image

```
ksis store <patchname> [<options>]
```

The **store** command is the mandatory step after running the **workon** command.

The command creates a new image containing the differences from the mother image. Using the **-R** option means that a reboot is necessary after installing the patch.

The status for this new image is **patch**.

When using this command the check level associated with this image is requested. Choose 'basic' for a standard level (see *5.5 Checking Deployed Images* for other options).

5.9.3 Creating a Patched Golden Image

```
ksis detach <imagename> [<options>]
```

The **detach** command allows you to modify an image without creating a new one. The command creates a new image resulting in the application of the patch on the source image. The status for this new image is **patched golden**.

5.9.4 Building a Patch

ksis buildpatch is used to create a patch from the differences between two images that can then be used to transform the software structure and content of a first node which has the first image deployed on it so that it matches a node which has the second image deployed on it.

Note:
ksis buildpatch can only be for two images which are derived from each other and not for images which are unrelated.

The command below would create a patch from the differences between the **<imageName1>** image and the **<imageName2>** image.

```
ksis buildpatch <imageName1> <imageName2>
```

Using ksis buildpatch

1. Make any changes required to the deployed version of the **<imageName1>** image. This is done by logging on to a node **n** which has **<imageName1>** on it and changing whatever needs to be changed. If necessary reboot on the node and check that everything is working OK.

2. Create an image of the node which has the **<imageName1>** image on it.

```
ksis create <imageName1> n
```

3. Create a patch of the differences between the **<imageName1>** and **<imageName2>** images. The patch will be automatically name e.g.**ImageName1.s1.0** for the first patch generated for **<ImageName1>** image.

```
ksis buildpatch <imageName1> <imageName2>
```

4. Deploy this patch on to the nodes which have **<imageName1>** on them.

```
ksis deploy <patch_name> <nodelist>
```

5. These nodes will now have a software content and structure which matches **<imageName2>.**

5.10 Checking Images

The **check** command checks the image deployed on a node set.

```
ksis check <nodeRangeOrGroupName>
```

The **checkdiff** command displays the discrepancies between a reference node and the results for a given check on a given node.

```
ksis checkdiff <testName> <node>
```

5.11 Importing and Exporting Images

This command exports an image from one cluster to another cluster.

```
ksis export <imageName>
```

This command imports an image previously exported from another cluster.

```
ksis import <imageName>
```

5.12 Setting Boot Mode

This command allows the nodes to boot on the primary system for the next reboot.

```
ksis setbootmode primary <nodeRangeOrGroupName>
```

This command allows the nodes to boot on the secondary system for the next reboot.

```
ksis setbootmode secondary <nodeRangeOrGroupName>
```

Chapter 6. Resource Management

Merely grouping together several machines on a network is not enough to constitute a real cluster. Resource Management software is required to optimize the throughput within the cluster according to specific scheduling policies.

A **resource manager** is used to allocate resources, to find out the status of resources, and to collect task execution information. From this information the scheduling policy can be applied. Bull HPC platforms use either **RMS**, a commercial product from **Quadrics**, or **SLURM** an open-source, scalable resource manager.

This chapter describes the following topics:

- 6.1 *Managing Resources with RMS*
- 6.2 *BLBS: Bull Load-Balancing System and RMS*
- 6.3 *Resource Management with SLURM*
- 6.4 *SLURM Configuration*
- 6.5 *Administrating Cluster Activity with SLURM*

6.1 Managing Resources with RMS

The key to achieving high-levels of performance for a large-scale parallel application is dedicating resources (CPUs, memory, network bandwidth and local I/O capability) to its execution. **RMS** allows a system administrator to manage these resources efficiently to ensure maximum performance or to dedicate resources to a particular type of application. Nodes can be configured into mutually exclusive sets of resources known as partitions each of which provide a specific system service. For example, the system could have an interactive partition for conventional UNIX processes and program development, a sequential batch partition, and a parallel partition running the **RMS** gang scheduler.

Free cycles on the interactive partition may be consumed by sequential batch jobs running from a low priority queue. Also, the system may be configured to allow certain users to run high-priority interactive jobs during working hours.

When **RMS** is installed **RMS** users, who are authorized to perform **RMS** management tasks, are then defined.

The **RMS** daemons continually monitor system availability and performance. The condition of every node in the system is stored in the **RMS** database together with a range of other vital statistics including CPU utilization, memory availability, I/O rates and error rates.

6.1.1 Managing Partitions

The administrator of an RMS system can control how the nodes are configured into partitions, who has access rights to each partition and the levels of resources that they can use. The administrator can suspend, resume or cancel jobs and alter their priority. All of this can be done direct from the command line or from the **Pandora GUI**. Scheduling decisions may be applied to all the processes in a parallel program at the same time.

To create a partition, the administrator uses the **rcontrol create partition** RMS command, with arguments specifying the node list, the required configuration type and the partition name. If the nodes that are to be included in a partition do not exist then the administrator has to create them using the **rcontrol create node** command.

Before modifying a partition, it is necessary to stop it, and any other partitions which include nodes that are involved in the modifications. To stop a partition the administrator uses the **rcontrol stop partition** command.

The administrator may define a policy for the allocation of resources for a set of partitions. A user may be authorized to specify the compute nodes (nodes contiguity), cyclic or block mode submission, etc.

If the management node has a passive backup node to ensure High Availability the administrator will need to decide which partitions will be automatically restarted following a failover. In order to do this the **autostart** field in the RMS database servers table will have to be modified for each partition concerned.

The command to do this is as follows:

```
# rmsquery "update servers set autostart='1' where partition = '<partition-name>'
```

It is possible to extract nodes from the production configuration without interrupting any computing being done on these nodes. To do this, all new allocations on these nodes are stopped.

There are two methods to remove nodes from a configuration:

- Use the **rcontrol configure out** command. This command requires that no tasks are running on the node.
- Or use the **rcontrol drain node** command, which enables the current tasks on the node to finish, but prevents the node from being selected for new tasks. When the current tasks finish, the node is automatically removed from the configuration.

6.1.2 Managing Limits

The system administrator can set **timelimits**, **timeslice** intervals, minimum job sizes and default memory limits on a per partition basis. The maximum job size (CPUs and memory) and the default priority can be controlled on a per-user or per-project basis as can the total number of CPUs in use at any point in time.

These limits are independent of the Linux system itself. They can be defined either for each task or they can correspond to the resources allocated to a partition.

The limits on core file size, file size, pipe size, and maximum number of processes are inherited from prun. The limit on the number of open files is inherited from prun unless a lower limit has been set for the partition. The CPU time limit is inherited from prun unless a lower limit applies to the resource request.

The RMS limits are inherited from the **/etc/security/limits.conf** file. This file contains the limits for the shell that the administrator has defined.
To display the limits values, enter:

```
# prun -p test -N 1 sh -c 'ulimit -a'
```

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals                 (-i) 32682
max locked memory       (kbytes, -l) 512
max memory size         (kbytes, -m) unlimited
open files                      (-n) 4096
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
stack size              (kbytes, -s) 2097152
cpu time               (seconds, -t) unlimited
max user processes              (-u) 32682
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Whether or not RMS inherits these limits depends on the value of the **inherit-user-limits** attribute:

```
# rcontrol create attribute inherit-user-limits val 1
# rcontrol set attribute inherit-user-limits val 0
```

For example, set the CPU time value to 180s:

```
# ulimit -t 180
```

Display the limits values:

```
# prun -p test -N 1 sh -c 'ulimit -a'
```

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals                 (-i) 32682
max locked memory       (kbytes, -l) 512
max memory size         (kbytes, -m) unlimited
open files                      (-n) 4096
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
stack size              (kbytes, -s) 2097152
cpu time               (seconds, -t) unlimited
max user processes              (-u) 32682
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Since the **inherit-user-limits val** is set to **0**, the **cpu time** limit has not been changed.

Now set **val** to **1**, to validate the limits.

```
# rcontrol set attribute inherit-user-limits val 1
```

Reload the `test` partition used for the job (mandatory):

```
# rcontrol reload partition test
```

Display the limits values:

```
# prun -p test -N 1 sh -c 'ulimit -a'
```

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals                 (-i) 32682
max locked memory       (kbytes, -l) 512
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) 180
max user processes              (-u) 32682
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

The **cpu time** limit is now effective.

The following example shows how the limits affect job execution.
Set a limit of 5 seconds for execution time (this is just an example; normally you should set a higher value for job execution):

```
# ulimit -t 5
```

Run the following command to get information about the nodes used for the `test` partition:

```
# rinfo
```

```
MACHINE     CONFIGURATION
nova        day

PARTITION        CPUS    STATUS        TIME   TIMELIMIT  NODES
root               48                                   nova[0,4,8]
test             0/32   running 12:21:59:51             nova[4,8]
```

Start a job, for example:

```
# prun -p test -N 2 tping 0 4M
```

```
  0:        0 bytes      2.57 uSec     0.00 MB/s
  0:        4 bytes      2.70 uSec     1.48 MB/s
  0:        8 bytes      2.71 uSec     2.96 MB/s
  0:       16 bytes      2.72 uSec     5.88 MB/s
  0:       32 bytes      2.72 uSec    11.78 MB/s
  0:       64 bytes      3.21 uSec    19.96 MB/s
  0:      128 bytes      3.64 uSec    35.19 MB/s
  0:      256 bytes      5.46 uSec    46.93 MB/s
  0:      512 bytes      5.75 uSec    88.99 MB/s
  0:     1024 bytes      6.40 uSec   160.11 MB/s
  0:     2048 bytes      7.57 uSec   270.47 MB/s
  0:     4096 bytes      9.71 uSec   421.81 MB/s
  0:     8192 bytes     14.89 uSec   550.33 MB/s
  0:    16384 bytes     23.95 uSec   683.99 MB/s
  0:    32768 bytes     42.19 uSec   776.60 MB/s
  0:    65536 bytes     78.57 uSec   834.10 MB/s
  0:   131072 bytes    151.24 uSec   866.65 MB/s
  0:   262144 bytes    296.83 uSec   883.15 MB/s
  0:   524288 bytes    587.03 uSec   893.12 MB/s
prun: /usr/bin/tping (host nova4 process 0 pid 25493) killed by
signal 9 (KILL)
prun: Warning: core files may be disabled by 'ulimit -c' setting
prun: looking for multi-threaded core file...
prun: no core file detected for analysis
```

Run the following command to get the Job Id:

```
# rinfo -J nova4
```

```
Job history for nova4
JOB      START TIME        END TIME          USERNAME   STATUS  CMD
2115     02/01/07 15:59:51 02/01/07 15:59:56  root       killed  tping 0 4M
```

Run the following command to check the status of the job:

```
# rinfo -j 2115
```

```
JOB          STATUS       COMMAND                  HOSTNAMES
2115         killed       tping 0 4M               nova[4,8]


RESOURCE     PARTITION    CPUS   PRI   USERNAME     PID     PROJECT
11714        test         2      50    root         19831   default

ETIME        ATIME        UTIME        STIME        PAGEFLTS MEM
00:06        00:12        00:09        00:00        0        58

ELAN OPS     BYTES TRANSFERRED
80964        4241483304

RAILS        SUBMITHOST
0            nova0
```

This output shows that the job cpu time was 6 seconds, so the job was killed, as expected.

6.1.3 Synchronizing Time

RMS will not operate correctly if the times on the nodes which make up the cluster vary. It is therefore necessary to have a time synchronization system in place. If the **rdate** command is installed on the machine at the time as the **RMS** RPMs are installed then the RMS RPM installation scripts will enable the command on the **rmshost** node and create a daily **cron** job on the compute nodes to use **rdate** to synchronize their time with the **rmshost** node.

6.1.4 Getting Topology Information with rinfo

When connecting to a machine of the cluster, it may be useful to have a global view of its topology. The **rinfo** command may be used for this purpose, alternatively run **/proc/cpuinfo**.

rinfo is a **RMS** command used on HPC platforms with **Quadrics** Interconnects and which provides a global overview of the partitions defined by **RMS** on a cluster including the number of CPUs and machines within it. **rinfo** will also indicate the number of CPUs used when an application is executed within a partition, and the state of affairs for the active applications.

This command can also be used to obtain further information about the topology of the cluster.

For example:

```
# rinfo
MACHINE  CONFIGURATION
nsad  day

PARTITION  CPUS STATUS  TIME TIMELIMIT  NODES
root   28          ns[13-15]
nsad
part1   0/8 running 1:00:07:02     ns[13-14]
part2   ??/0  down  --:--
```

In the example above the cluster consists of 28 processors and 3 nodes: `ns13`, `ns14` and `ns15`. The first RMS partition is shown as '`part1`' and consists of 2 nodes (`ns13` and `ns14`) and 8 CPUs and its status is 'running' which means that it can be used.

The second partition is '`part2`' and its status is 'down', with no nodes allocated, which means that it cannot be used.

6.1.5 Viewing processor information in /proc/cpuinfo

Information about the processors present on the machine is stored in **/proc/cpuinfo**. To view this information run the following command:

```
# cat /proc/cpuinfo
```

For example, a machine with 4 processors might display the following:

```
processor : 0
vendor  : GenuineIntel
arch  : IA-64
family  : Itanium 2
model  : 1
revision : 5
archrev : 0
features : branchlong
cpu number : 0
cpu regs : 4
cpu MHz : 1296.460997
itc MHz : 1296.460997
BogoMIPS : 1941.96

processor : 1
vendor  : GenuineIntel
arch  : IA-64
family  : Itanium 2
model  : 1
revision : 5
archrev : 0
features : branchlong
cpu number : 0
cpu regs : 4
cpu MHz : 1296.460997
itc MHz : 1296.460997
BogoMIPS : 1941.96

processor : 2
vendor  : GenuineIntel
arch  : IA-64
family  : Itanium 2
model  : 1
revision : 5
archrev : 0
features : branchlong
cpu number : 0
cpu regs : 4
cpu MHz : 1296.460997
itc MHz : 1296.460997
BogoMIPS : 1941.96

processor : 3
vendor  : GenuineIntel
arch  : IA-64
family  : Itanium 2
model  : 1
revision : 5
archrev : 0
features : branchlong
cpu number : 0
cpu regs : 4
cpu MHz : 1296.460997
itc MHz : 1296.460997
BogoMIPS : 1941.96
```

6.1.6 Day to Day RMS Operations

This section describes how to automate some routine or day-to-day RMS operations including archiving and backup procedures. The following routines are described:

- Periodic shift changes
- Summarizing accounting data
- Archiving data
- Backing up the database
- Restoring the database
- Increasing number of RMS contexts

6.1.6.1 Periodic Shift Changes

RMS supports periodic shift change through a **cron** job that changes the active configuration. For example, a system can be set up with two operating configurations: one called **day** and the other called **night**. During the day, the resources are split into two partitions: a login partition (called **login**) for program development and a parallel partition (called **parallel**) for execution of short parallel programs. At night, all of the nodes are assigned to a single partition (again called **parallel**) with a longer time limit for running parallel jobs.

Use the following commands to create this pair of configurations:

```
# rcontrol create partition=login configuration=day type=login
  nodes='nova[0-7]'
# rcontrol create partition=parallel configuration=day type=parallel \
timelimit=600 nodes='nova[8-63]'
# rcontrol create partition=parallel configuration=night type=parallel \
timelimit=3600 nodes='nova[0-63]'
```

To start the **day** configuration, enter the following:

```
# rcontrol start configuration=day
```

To switch to the **night** configuration, use this command:

```
# rcontrol start configuration=night
```

Note that, after the switch, any jobs running on the **parallel** partition will continue to run as the **parallel** partition in the **night** configuration which has more nodes. However, when changing back from **night** to **day**, it must be decided what will be done on any jobs that are running on nodes **'nova[0-7]'**. The options are to wait for them to finish or to kill them. To wait for them to finish, start the configuration with the **wait** option:

```
# rcontrol start configuration=day option=wait
```

To kill them, start the configuration with the **kill** option:

```
# rcontrol start configuration=day option=kill
```

Note:

In the current release, any requests that are suspended when a partition is stopped must be resumed manually if the partition is restarted.

When you are satisfied with the shift changes, create a **cron** job to perform them automatically.

6.1.6.2 Summarizing Accounting Data

Accounting records accumulate in the RMS database as each job is run. By default, a record is included in the **archive_tables** table to archive them (see *Archiving Data*) but you may prefer to delete them only after a report has been generated. Each site has its own requirements in this respect. A simple example script to produce a summary of resource usage is included in the release in **/usr/lib/rms/examples/scripts/accounting_summary**.

```
Accounting Summary of Machine nova at 16:01 Wed 21 Feb 2001
Usage by Project/User For Previous Day

Project User    CPU   User   Sys Number
Name  Name    Secs   Secs   Secs
Sessions
------------------------------------------------------------------------
default addy    596   533    6  8
   duncan    58   37    2  6
   johnt    540   227   51  15
   root    29272    2    8  37
   stephen    286   87   134  56
------------------------------------------------------------------------
Total default    30752   886   201 122
------------------------------------------------------------------------
Grand Total     30752   886   201 122
------------------------------------------------------------------------
```

When the accounts have been processed, the script can optionally delete the accounting records for resource requests that have completed.

This script (or one based on it) can be run nightly with a cron job, as shown in the following example.

```
0 0 * * * /usr/opt/rms/examples/scripts/accounting_summary
```

6.1.6.3 Archiving Data

A number of tables in the database grow over time or as jobs are submitted, for example, the **events** table and the **jobs** table. These tables can be kept to a reasonable size by periodically running **rmsarchive** . This command archives selected records and then deletes them from the database.

rmsarchive uses criteria specified in the **archive_tables** table to determine which records to archive. Each entry in the **archive_tables** has four fields:

Archives_tables Entry	Definition
Name	The name of a table to archive.
Lifetime	The maximum time in hours for the data to remain in the table
Timefield	The name of the field in name that records the data's lifetime.
Selectstr	A SQL select statement, based on fields in name, that determines which records to archive. If not null it is used in addition to the default selection based on lifetime and timefield.

Use **rmsquery** to create or update entries in the **archive_tables** table. For example, to add an entry that causes **rmsarchive** to delete all records in the transactions table that are more than three days old, enter:

```
# rmsquery "insert into archive_tables (name,lifetime,timefield) \
   values ('transactions',72,'mtime')"
```

The following example causes **rmsarchive** to delete all records from the transactions table that denote completed transactions:

```
# rmsquery "insert into archive_tables (name,lifetime,timefield,selectstr) \
   values ('transactions',72,'mtime','status = \'complete\'')"
```

Note:

Avoid the use of single quotation marks in **selectstr**.

rmsarchive archives the records as a sequence of SQL statements in a file called *machine_date*.sql, where *machine* is the name of the RMS machine and the *date* format is: **yyyy-mm-dd-HH:MM**. The archive file is compressed using **gzip** and stored in **/var/rms/archive**.

If you want to archive the database of a machine which is not the database of the host machine, give the name of the machine as a command line argument to **rmsarchive**. For example, to archive the database of a machine called `juno`:

```
# rmsarchive -m juno
```

When the archive process is complete, **rmsarchive** sends mail to any users listed by the **users-to-mail** attribute in the **attributes** table. This may be set by using **rcontrol**.

The mail message includes the name of the archive file, which tables were archived and how many records were deleted, as shown in the following excerpt:

```
rmsarchive: Nov 10 01:05:01

rmsarchive: '/usr/opt/rms/bin/rmstbladm -c -x 73 -v' started Nov 10
01:05:00
rmsarchive: finished Nov 10 01:05:01 total records 2918 file is
/var/rms/archive/nova_2001-11-10-01:05.sql

rmsarchive: rmstbladm output was ...
rmstbladm: archiving to /var/rms/archive/nova_2001-11-10-01:05.sql
rmstbladm: archiving table 'rms_nova:rms:acctstats': 4 records dumped
rmstbladm: archiving table 'rms_nova:rms:disk_stats': 0 records dumped
rmstbladm: archiving table 'rms_nova:rms:events': 4 records dumped
rmstbladm: archiving table 'rms_nova:rms:jobs': 4 records dumped
rmstbladm: archiving table 'rms_nova:rms:link_errors': 2902 records
dumped
rmstbladm: archiving table 'rms_nova:rms:resources': 4 records dumped
rmstbladm: gzipping '/var/rms/archive/nova_2001-11-10-01:05.sql'
```

rmsarchive also adds a record to the backups table to show the status of the archive procedure. For example:

```
name  status ctime    handle filename
archived
-------------------------------------------------------------------
archive complete 11/10/01 01:05:01 49  /var/rms/archive/nova_2001-11-10-
01:05.sql
2918
```

The archive procedure can be automated by adding an entry to the **crontab** file for the cron to execute. For example, to run **rmsarchive** daily at 1:05 a.m. using cron, add the following line to the root **crontab** file on the RMS host machine:

```
5 1 * * * /usr/opt/rms/bin/rmsarchive
```

rmsarchive invokes **rmstbladm** to manipulate the database. **rmstbladm** can be used directly to archive specific tables. The following example archives (without deleting) data from the **events** table:

```
# rmstbladm -d -t events > events.sql
```

Alternatively, you can execute a SQL query to extract (but not delete) data, as shown in the following examples.

```
rmsquery -v "select * from jobs where endTime <> 0 and \
 endTime < $old order by startTime" > jobs.dat

now=`rmsgettime`
old=`expr $now - 172800`
rmsquery -v "select * from backups where ctime <= $old \
 order by ctime" > link_errors.sql
```

Failure to clear old entries can cause problems. Note that the default **archive_tables** table contains an entry for the accounting statistics table. It may be prefered to remove it from the **archive_tables** table and use a script, as described in the *Summarising Accounting Data* section, to delete accounting statistics records only after a report has been generated.

6.1.6.4 Backing up the Database

To back up the database, use **rmsbackup**. **rmsbackup** backs up the tables as a set of SQL statements that can be restored using **rmstbladm**. The SQL statements are written to a file called ***machine_date*.sql**, where *machine* is the name of the RMS machine and the date format is: yyyy-mm-dd-HH:MM. The back-up file is compressed using **gzip** and stored in **/var/rms/backup**.

By default, **rmsbackup** archives the database first to remove any redundant entries. To prevent archiving, use the **-b** flag in the command line.

If you want to back up the database of a machine separate from the database of the host machine, give the name of the machine as a command line argument to **rmsbackup**. For example, to back up the database of a machine called **juno**:

```
# rmsbackup -m juno
```

When the backup is complete, **rmsbackup** sends a mail to any users listed in the **users-to-mail** attribute in the **attributes** table. This can be set using **rcontrol**.

The mail message includes the name of the backup file, which tables were backed up how many records each table contained, as shown in the example below:

```
rmsbackup: Nov 7 01:05:04

rmsbackup: '/usr/opt/rms/bin/rmstbladm -b -x 62 -v' started Nov 7 01:05:02
rmsbackup: finished Nov 7 01:05:04 total records 23548 file is
/var/rms/backup/nova_2001-11-07-01:05.sql

rmsbackup: rmstbladm output was ...
rmstbladm: backing up to /var/rms/backup/nova_2001-11-07-01:05.sql
rmstbladm: backing up table 'rms_nova:rms:access_controls': 1 records
dumped
rmstbladm: backing up table 'rms_nova:rms:acctstats': 322 records dumped
rmstbladm: backing up table 'rms_nova:rms:archive_tables': 7 records
dumped
rmstbladm: backing up table 'rms_nova:rms:attributes': 29 records dumped
rmstbladm: backing up table 'rms_nova:rms:backups': 57 records dumped
...
rmstbladm: backing up table 'rms_nova:rms:users': 10 records dumped
rmstbladm: gzipping '/var/rms/backup/nova_2001-11-07-01:05.sql'
```

rmsbackup also adds a record to the **backups** table to show the status of the back-up procedure. For example:

```
name  status ctime   handle filename
archived
-------------------------------------------------------------------------
backup complete 11/07/01 01:05:01 49  /var/rms/archive/nova_2001-11-07-
01:05.sql
23548
```

The back-up procedure can be automated by adding an entry to the **crontab** file for a **cron** to execute. For example, to run **rmsbackup** daily at 1:05 a.m. using a **cron**, add the following line to the **root crontab** file on the RMS host machine:

```
5 1 * * * /usr/opt/rms/bin/rmsbackup
```

6.1.6.5 Restoring the Database

Before restoring the database from a backup file, all jobs should be killed using **rcontrol** and **RMS** should be halted using **rmsctl**. Any updates made to the database since the backup was taken will be lost.

To restore the data, use **rmstbladm** as follows:

```
# rcontrol kill resource partition=parallel
# rmsctl stop
# rmstbladm -r machine_date.sql.gz
# rmsctl start
```

rmstbladm automatically unzips the file into the **/tmp** directory before restoring the data.

6.1.6.6 Increasing the number of RMS contexts

By default, RMS does not support more than 64 concurrent jobs per partition (assuming the cluster is made of 16 CPU nodes, for example Bull NovaScale 6160). This is due to the fact that only 1024 contexts are available by default in the RMS kernel module, each CPU requiring one context.

By default, the RMS contexts range from 1024 to 2047:

```
# cat /proc/qsnet/elan/ctx
```

```
DisplayCtxRange
System      [     0 -    31 ]
User        [    64 -  1023 ]
RMS         [  1024 -  2047 ]
Kcomms      [  2048 -  4095 ]
```

In order to increase this number, perform the following actions on ALL nodes:

1. Add the following line to **/etc/modprobe.conf**:
   `options elan4 elan4_ctxt_table_shift=14`

2. Re-compute the module dependencies as follows:

```
# depmod -a
```

3. Stop the RMS service:

```
# service rms stop
```

4. Stop Lustre (if running) and unload the corresponding kernel modules.

5. If eip is running, shut down the corresponding network (typically eip0):

```
# ifconfig eip0 down
```

6. Restart the qsnet service:

```
# service qsnet restart
```

7. Restart Lustre.

8. Restart the RMS service:

```
# service rms restart
```

9. After this procedure, check that the extended contexts are active (RMS ext):

```
# cat /proc/qsnet/elan/ctx
```

```
DisplayCtxRange
System     [     0 -    31 ]
User       [    64 -  1023 ]
RMS        [  1024 -  2047 ]
Kcomms     [  2048 -  4095 ]
RMS ext    [  4096 - 16383 ]
```

Note:
Make sure that all nodes in the cluster have the same extended context values.

6.1.7 More Information

For more detailed information about RMS, refer to the *RMS User's Guide* and the *RMS Reference Manual*. These manuals are available from the **Documentation** page on the www.quadrics.com web site.

For the latest information regarding your **Quadrics** delivery, please look at the **readme** file on the Quadrics CDs.

6.2 BLBS: Bull Load-Balancing System and RMS

6.2.1 Overview

Bull Load-Balancing System (**BLBS**) makes it possible to identify machines within an **RMS** partition which have low loads automatically. To do this the IP addresses of the low loaded machines are associated with the name of the relevant partition. This pairing (**partition_name**, **IP_address**) is configured by the **DNS** server of the Management Node so that access to this machine is available for the network.

The criterion used to calculate the load for each machine is the number of active users per machine in the partition.

6.2.2 Using BLBS

Every minute, a **cron** launches the **/usr/sbin/blbs** shell script on the management node. This script checks the **RMS** database for all the login type partitions in the cluster which are in a **running** state. For each of these partitions, the script then determines by means of the **rmsquery** command which machines have low loads.

For each partition analyzed, the **BLBS** script updates the local DNS server, using the **nsupdate** command. Each partition creates an entry in the **BLBS** domain on the Domain Name Server.

So, if there are three login partitions named `devel`, `valid` and `maint` three entries are created and updated on the **DNS** server with the names `devel.blbs`, `valid.blbs` and `maint.blbs`.Using these names the machines may then be accessed from outside the cluster.

6.2.3 Configuring BLBS

BLBS is delivered as a RPM on the **Cluster-Management CD** and is installed from this CD. The load-balancing feature is automatically activated after installation.

BLBS requires there to be defined **RMS** partitions reserved for the interactive connections managed by the load-balancing system. These **RMS** partitions must be of the *login* type. Therefore, the **type=login** option must be specified in the command that creates the partition, as follows:

```
rcontrol create partition=XXX type=login
```

It is desirable that the partition machines are login nodes so that the best level of service can be offered to users who connect to these nodes.

The whole **DNS** system of the cluster has to be properly configured to resolve the **BLBS** domain on the management node of the cluster, and in particular to start the **named** service on the management node.

During installation, a new zone is created in the **/etc/named.conf** file. This zone is managed by the **BLBS** rpm, so it should not be modified. However, it is still possible to add zones before or after this one is created, but avoid modifying it because when uninstalling or upgrading the RPM, the whole zone definition may be erased!

6.2.4 BLBS Administration

BLBS relies on the standard **named** daemon of the **bind** package, which is controlled by the **/etc/rc.d/init.d/named** script. It can be stopped and started again with the following commands:

```
service named start
service named stop
```

Because the updates of the load-balancing data are dynamic, it is only necessary to start the **named** daemon when booting up the management node.

By default, the **/usr/sbin/blbs** script runs in the background and is not visible to the system unless there is a problem in its execution. This means that useless mails for root are not created.

The script uses two parameters; the first one describes the method use to compute the load, and the second the criterion used to sort machines.

The parameters used by the script are given by the **cron** entry located in the **/etc/cron.d/dnsloadbalancing** file and are as follows:

- Use of the direct query of **RMS** database (**BULL** method).
- Search for the lowest number of users connected (**users** criterion).

It is possible to select **idlecpu** instead of **users** as a criterion to select machines with low CPU usage.

For analysis purposes, each time the **/usr/sbin/blbs** script updates the **DNS** server it creates a host named **time.blbs**, whose IP address is the current date and time of the machine. It is not possible to connect to this host, but the date of the last update may be ascertained by using, for example, the host command as follows:

host time.blbs returns **time.blbs** with the address 9.2.10.6 This means that the last update occurred on the 9th of February at 10:06.

If the host doesn't reflect the current time of the management node, check the following points:

- Is the **DNS** server properly configured and running correctly?
- Are there login partitions for the cluster? Are these running?
- Is **BLBS** running on the active management node when High Availability is functioning?
- Finally, remember that the **RMS** database is updated every 30 seconds; this delay must be taken into account because it directly impacts the load balancing system.

6.3 Resource Management with SLURM

6.3.1 SLURM Key Functions

As a cluster resource manager, SLURM has three key functions. Firstly, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Secondly, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates conflicting requests for resources by managing a queue of pending work.

Users interact with SLURM through four command line utilities:

- **SRUN** for submitting a job for execution and optionally controlling it interactively,
- **SCANCEL** for terminating a pending or running job,
- **SQUEUE** for monitoring job queues, and
- **SINFO** for monitoring partition and overall system state.

System administrators perform privileged operations through an additional command line utility, **SCONTROL**.

The central controller daemon, **SLURMCTLD**, maintains the global state and directs operations. Compute nodes simply run a **SLURMD** daemon (similar to a remote shell daemon) to export control to SLURM.

SLURM supports resource management across a single cluster.

SLURM is not a sophisticated batch system. In fact, it was expressly designed to provide high-performance parallel job management while leaving scheduling decisions to an external entity. Its default scheduler implements **First-In First-Out** (FIFO). A scheduler entity can establish a job's initial priority through a plug-in.

An external scheduler may also submit, signal, and terminate jobs as well as reorder the queue of pending jobs via the API.

6.3.2 SLURM Components

SLURM consists of two types of daemons and five command-line user utilities. The relationships between these components are illustrated in the following diagram:

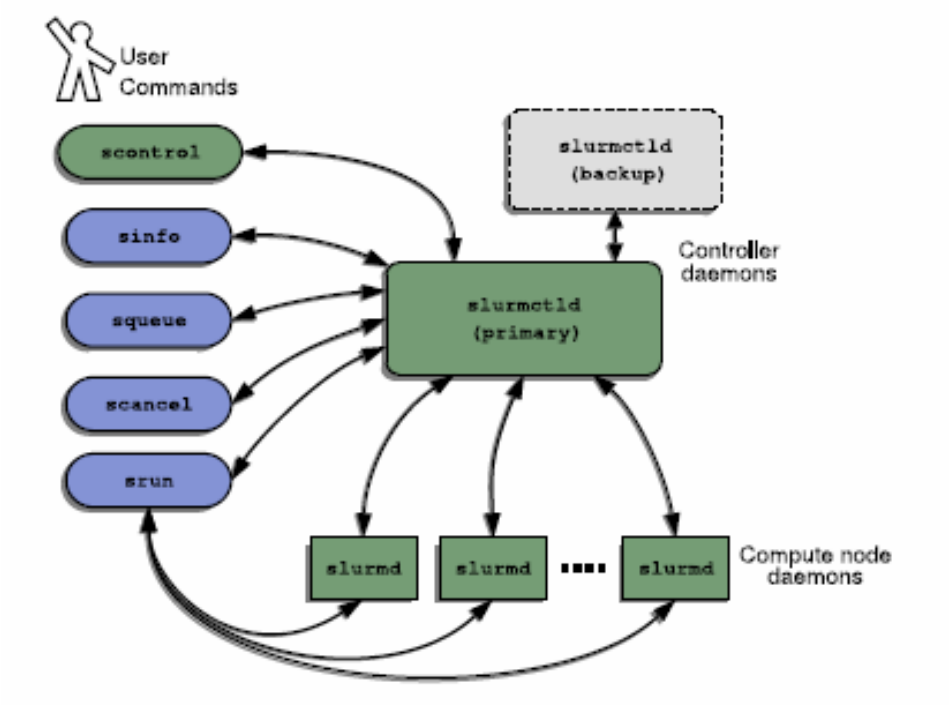


Figure 6-1. SLURM Simplified Architecture

6.3.3 SLURM Daemons

6.3.3.1 SLURMCTLD

The central control daemon for **SLURM** is called **SLURMCTLD**. SLURMCTLD is *multi*–threaded; thus, some threads can handle problems without delaying services to normal jobs that are also running and need attention. **SLURMCTLD** runs on a single management node (with a fail-over spare copy elsewhere for safety), reads the **SLURM** configuration file, and maintains state information on:

- Nodes (the basic compute resource)
- Partitions (sets of nodes)
- Jobs (or resource allocations to run jobs for a time period)
- Job steps (parallel tasks within a job).

The **SLURMCTLD** daemon in turn consists of three software subsystems, each with a specific role:

Software Subsystem	Role Description
Node Manager	Monitors the state and configuration of each node in the cluster. It receives state-change messages from each compute node's SLURMD daemon asynchronously, and it also actively polls these daemons periodically for status reports.
Partition Manager	Groups nodes into disjoint sets (partitions) and assigns job limits and access controls to each partition. The partition manager also allocates nodes to jobs (at the request of the Job Manager) based on job and partition properties. SCONTROL is the (privileged) user utility that can alter partition properties.
Job Manager	Accepts job requests (from SRUN or a metabatch system), places them in a priority-ordered queue, and reviews that queue periodically or when any state change might allow a new job to start. Resources are allocated to qualifying jobs and that information transfers to (SLURMD on) the relevant nodes so the job can execute. When all nodes assigned to a job report that their work is done, the Job Manager revises its records and reviews the pending-job queue again.

Table 6-1. Role Descriptions for SLURMCTLD Software Subsystems

The following figure illustrates these roles of the SLURM Software Subsystems.

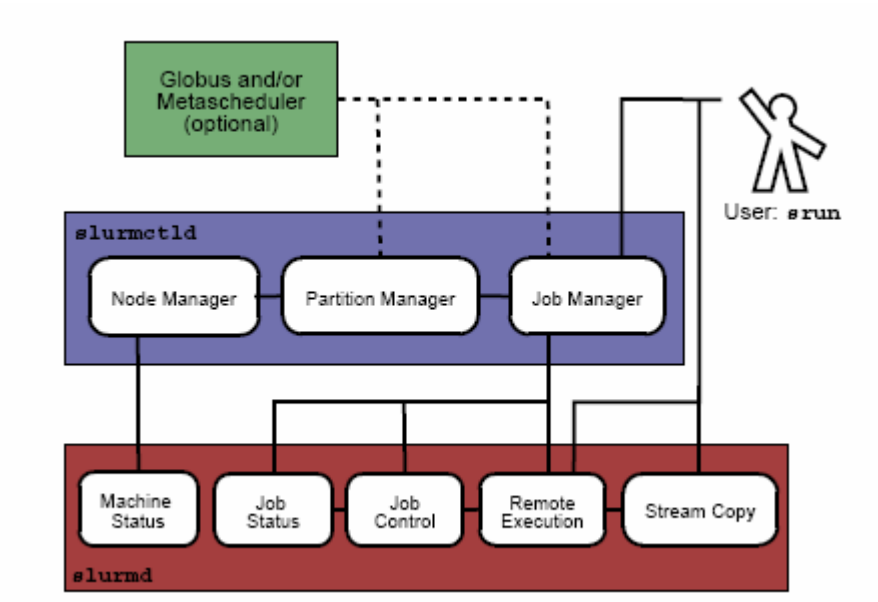


Figure 6-2. SLURM Architecture - Subsystems

6.3.3.2 SLURMD

The **SLURMD** daemon runs on all the compute nodes of each cluster that **SLURM** manages and performs the lowest level work of resource management. Like **SLURMCTLD** (previous subsection), **SLURMD** is multi-threaded for efficiency; but, unlike SLURMCTLD, it runs with root privileges (so it can initiate jobs on behalf of other users).

SLURMD carries out five key tasks and has five corresponding subsystems. These subsystems are described in the following table.

SLURMD Subsystem	Description of Key Tasks
Machine Status	Responds to **SLURMCTLD** requests for machine state information and sends asynchronous reports of state changes to help with queue control.
Job Status	Responds to **SLURMCTLD** requests for job state information and sends asynchronous reports of state changes to help with queue control.
Remote Execution	Starts, monitors, and cleans up after a set of processes (usually shared by a parallel job), as decided by **SLURMCTLD** (or by direct user intervention). This can often involve many changes to process-limit, environment-variable, working-directory, and user-id.
Stream Copy Service	Handles all **STDERR**, **STDIN**, and **STDOUT** for remote tasks. This may involve redirection, and it always involves locally buffering job output to avoid blocking local tasks.
Job Control	Propagates signals and job-termination requests to any SLURM-managed processes (often interacting with the Remote Execution subsystem).

Table 6-2. SLURMD Subsystems and Key Tasks

6.3.4 Scheduler Types

The system administrator for each machine can configure **SLURM** to invoke one of several alternative local job schedulers. To determine which scheduler SLURM is currently invoking on any machine, execute the following command:

```
scontrol show config |grep SchedulerType
```

where the returned string will have one of the values described in the following table.

Returned String Value	Description
builtin	A first-in-first-out scheduler. SLURM executes jobs strictly in the order in which they were submitted (for each resource partition), unless those jobs have different priorities. Even if resources become available to start a specific job, SLURM will wait until there is no previously-submitted job pending (which sometimes confuses impatient job submitters). This is the default.

backfill	Modifies strict FIFO scheduling to take advantage of resource islands that may appear as earlier jobs complete. SLURM will start later-submitted jobs out of order if resources become available, *and* if doing so does not delay the expected execution time of any earlier-submitted job. To increase the job's chances of benefiting from such backfill scheduling: (1) specify reasonable time limits (the default is the same time limit for all jobs in the partition, which may be too large), and (2) avoid requiring or excluding specific nodes by name.
wiki	Uses the Maui Scheduler, with a sophisticated set of internal scheduling algorithms. This choice can be configured in several ways to optimize job throughput. Details are posted on a support web site at the following URL: http://supercluster.org/maui

Table 6-3. SLURM Scheduler Types

6.4 SLURM Configuration

The SLURM configuration file, **slurm.conf**, is an ASCII file that describes the following:

- General SLURM configuration information
- The nodes to be managed
- Information about how those nodes are grouped into partitions
- Various scheduling parameters associated with those partitions.

The SLURM configuration file includes a wide variety of parameters. This configuration file must be available on each node of the cluster. A full description of the parameters is included in the **slurm.conf** man page. The **slurm.conf** file should define at least the configuration parameters defined in the samples provided and likely additional ones. Any text following a "**#**" is considered a comment. The keywords in the file are not case sensitive, although the argument typically is (e.g., "**SlurmUser=slurm**" might be specified as "**slurmuser=slurm**"). Port numbers to be used for communications are specified as well as various timer values.

A description of the nodes and their grouping into partitions is required. A simple node range expression may optionally be used to specify ranges of nodes to avoid building a configuration file with large numbers of entries. The node range expression can contain one pair of square brackets with a sequence of comma separated numbers and/or ranges of numbers separated by a "-" (e.g. "linux[0-64,128]", or "lx[15,18,32-33]").

Node names can have up to three name specifications: **NodeName** is the name used by all **SLURM** tools when referring to the node, **NodeAddr** is the name or IP address SLURM uses to communicate with the node, and **NodeHostname** is the name returned by the command */bin/hostname* -s. Only **NodeName** is required (the others default to the same name), although supporting all three parameters provides complete control over naming and addressing the nodes. See the **slurm.conf** man page for details on all configuration parameters.

Nodes can be in more than one partition and each partition can have different constraints (permitted users, time limits, job size limits, etc.). Each partition can thus be considered a separate queue. Partition and node specifications use node range expressions to identify nodes in a concise fashion. The Example #2 configuration file (see Section 6.4.2 slurm.conf Example Files) defines a 1154-node cluster for **SLURM**, but it might be used for a much larger cluster by just changing a few node range expressions. Specify the minimum processor count (**Procs**), real memory space (**RealMemory,** megabytes), and temporary disk space (**TmpDisk**, megabytes) that a node should have to be considered as available for use. Any node lacking these minimum configuration values will be considered DOWN and not scheduled. An annotated sample configuration file for SLURM is provided with this distribution as **/etc/slurm/slurm.conf.example**. Edit this configuration file to suit the needs of the user site and cluster, and then copy it to **/etc/slurm/slurm.conf**.

6.4.1 Configuration Parameters

Three types of SLURM configuration parameters are described in this section.

- General configuration parameters
- Node configuration parameters
- Partition configuration parameters

6.4.1.1 General Configuration Parameters

This section describes the overall configuration parameters available for SLURM.

AuthType

Define the authentication method for communications between SLURM components. Acceptable values at present include **auth/none**, **auth/authd** and **auth/munge**. The default value is **auth/none**, which means the UID included in communication messages is not verified. This may be fine for testing purposes, but do not use **auth/none** if any security is needed.

- **auth/authd** indicates that Brett Chun's **authd** is to be used (see http://www.theether.org/authd/ for more information).
- **auth/munge** indicates that Chris Dunlap's munge is to be used (this is the best supported authentication mechanism for **SLURM.** The munge application will need to be installed in order to use this functionality (- see http://www.llnl.gov/linux/munge/ for more information).

All SLURM daemons and commands must have been terminated prior to changing the value of **AuthType** and later restarted (SLURM jobs can be preserved).

BackupAddr

Name to use when referring to the **BackupController** for establishing a communications path. This name will be used as an argument to the **gethostbyname()** function for identification.
For example, `elx0000` might be used to designate the Ethernet address for node `lx0000`. By default the **BackupAddr** will be identical in value to BackupController.

BackupController

The name of the machine where SLURM control functions are to be executed in the event that **ControlMachine** fails. This node may also be used as a compute server if so desired. It will come into service as a controller only upon the failure of **ControlMachine** and will revert to a "*standby*" mode when the **ControlMachine** becomes available once again. This should be a node name without the full domain name (e.g. `lx0002`). While not essential, it is recommended that a backup controller be specified.

CacheGroups

If set to 1, the **SLURMD** daemon will cache **/etc/groups** entries. This can improve performance for highly parallel jobs if NIS servers are used and unable to respond very quickly. The default value is 0 to disable caching group data.

CheckpointType

Define the system-initiated checkpoint method to be used for user jobs. The **SLURMCTLD** daemon must be restarted for a change in **CheckpointType** to take effect. Acceptable values at present include "`checkpoint/none`" and "`checkpoint/aix`" (only on AIX systems). The default value is "`checkpoint/none`".

ControlAddr

Name to use when referring to the **ControlMachine** for establishing a communications path. This name will be used as an argument to the **gethostbyname()** function for identification. For example, "`elx0000`" might be used to designate the Ethernet address for node "`lx0000`". By default the **ControlAddr** will be identical in value to **ControlMachine**.

ControlMachine

The name of the machine where SLURM control functions are executed. This should be a node name without the full domain name (e.g. "`lx0001`"). This value must be specified.

Epilog

Fully-qualified pathname of a script to execute as user root on all nodes when a user's job completes (e.g. **/usr/local/slurm/epilog**). This may be used to purge files, disable user login, etc. By default there is no epilog.

FastSchedule

If set to 1 (the default), then the configuration of each node will be considered to be that which is specified in the configuration file. If set to 0, then scheduling decisions will be based upon the actual configuration of each individual node. If the number of node configuration entries in the configuration file is significantly lower than the number of nodes, setting **FastSchedule** to 1 will permit much faster scheduling decisions to be made. (The scheduler can check only the values in a few configuration records instead of possibly thousands of node records. If a job cannot be initiated immediately, the scheduler may execute these tests repeatedly.) Note that on systems with hyper-threading, the processor count reported by the node will be twice the actual processor count. Review these values for scheduling purposes.

FirstJobId

The job id to be used for the first job submitted to **SLURM** without a specific requested value. Job id values generated will be incremented by 1 for each subsequent job. This may be used to provide a meta-scheduler with a job id space, which is disjoint from the interactive jobs. The default value is 1.

HeartbeatInterval

Obsolete parameter. Interval of heartbeat for **SLURMD** daemon is half of **SlurmdTimeout**. Interval of heartbeat for SLURMCTLD daemon is half of SlurmctldTimeout.

InactiveLimit

The interval, in seconds, a job or job step is permitted to be inactive before it is terminated. A job or job step is considered inactive if the associated **SRUN** command is not responding to **SLURM** daemons. This could be due to the termination of the SRUN command or the program being in a stopped state. A batch job is considered inactive if it has no active job steps (e.g. periods of pre- and post-processing). This limit permits defunct jobs to be purged in a timely fashion without waiting for their time limit to be reached. This value should reflect the possibility that the **SRUN** command may be stopped by a debugger or considerable time could be required for batch job pre- and post-processing. This limit is ignored for jobs running in partitions with the **RootOnly** flag set (the scheduler running as root will be responsible for the job). The default value is unlimited (zero). May not exceed 65533.

JobAcctType

Define the job accounting mechanism type. Acceptable values at present include **jobacct/aix** (for AIX operating system), **jobacct/linux** (for Linux operating system) and j**obacct/none** (no accounting data collected). The default value is **jobacct/none**. In order to use the **SACCT** tool, **jobacct/aix** or **jobacct/linux** must be configured.

JobAcctLogFile

Define the location where job accounting logs are to be written. For **jobacct/**none this parameter is ignored. For **jobacct/linux** this is the fully-qualified file name for the data file.

JobAcctFrequency

Define the polling frequencies to pass to the job accounting plug-in. For jobacct/none this parameter is ignored. For j**obacct/linux** the parameter is a number of seconds between polls.

JobCompLoc

The interpretation of this value depends upon the logging mechanism specified by the **JobCompType** parameter.

JobCompType

Define the job completion logging mechanism type. Acceptable values at present include j**obcomp/none**, **jobcomp/filetxt**, and **jobcomp/script**. The default value is **jobcomp/none**, which means that upon job completion the record of the job is purged from the system. The value **jobcomp/filetxt** indicates that a record of the job should be written to a text file specified by the **JobCompLoc** parameter. The value **jobcomp/script** indicates that a script specified by the **JobCompLoc** parameter is to be executed with environment variables indicating the job information.

JobCredentialPrivateKey

Fully qualified pathname of a file containing a private key used for authentication by **SLURM** daemons.

JobCredentialPublicCertificate

Fully qualified pathname of a file containing a public key used for authentication by SLURM daemons.

KillTree

This option is mapped to **ProctrackType=proctrack/linuxproc**. It will be removed from future releases.

KillWait

The interval, in seconds, given to a job's processes between the **SIGTERM** and **SIGKILL** signals upon reaching their time limit. If the job fails to terminate gracefully in the interval specified, it will be forcibly terminated. The default value is 30 seconds. May not exceed 65533.

MaxJobCount

The maximum number of jobs **SLURM** can have in its active database at one time. Set the values of **MaxJobCount** and **MinJobAge** to insure the **SLURMCTLD** daemon does not exhaust its memory or other resources. Once this limit is reached, requests to submit additional jobs will fail. The default value is 2000 jobs. This value may not be reset via **scontrol reconfig**. It only takes effect upon restart of the **SLURMCTLD** daemon. May not exceed 65533.

MinJobAge

The minimum age of a completed job before its record is purged from SLURM's active database. Set the values of **MaxJobCount** and **MinJobAge** to insure the SLURMCTLD daemon does not exhaust its memory or other resources. The default value is 300 seconds. A value of zero prevents any job record purging. May not exceed 65533.

MpiDefault

Identifies the default type of **MPI** to be used. **SRUN** may override this configuration parameter in any case. Currently supported versions include: **lam** (which supports **LAM MPI** and **Open MPI**, but specify "none" instead and let LAM MPI and Open MPI select the plug-in using an option of the **SRUN** command), **mpich-gm**, **mvapich**, and none (default, which works for many other versions of MPI).

PluginDir

Identifies the places in which to look for SLURM plug-ins. This is a colon-separated list of directories, like the PATH environment variable. The default value is **/usr/local/lib/slurm**.

PlugStackConfig

Location of the configuration file for **SLURM** stackable plug-ins that use the Stackable Plug-in Architecture for Node job (K)control (**SPANK**). This provides support for a highly configurable set of plug-ins to be called before and/or after execution of each task spawned as part of a user's job step. Default location is **plugstack.conf** in the same directory as the system **slurm.conf**. For more information on SPANK plug-ins, see the spank(8) manual.

ProctrackType

Identifies the plug-in to be used for process tracking. The **SLURMD** daemon uses this mechanism to identify all processes that are children of processes it spawns for a user job. Acceptable values at present include **proctrack/aix** (which uses an AIX kernel extension and is the default for AIX systems), **proctrack/linuxproc** (which uses Linux process tree), **proctrack/rms** (which uses Quadrics kernel patch and is the default if **SwitchType=switch/elan**) and **proctrack/pgid** (which is the default for all other systems). The SLURMD daemon must be restarted for a change in **ProctrackType** to take effect.

Note:

proctrack/linuxproc is not compatible with **switch/elan**.

Prolog

Fully-qualified pathname of a script to execute as user root on every node when a user's job begins execution (e.g. **/usr/local/slurm/prolog**). This may be used to purge files, enable user login, etc. By default there is no prolog.

PropagatePrioProcess

Setting **PropagatePrioProcess** to "1", will cause a users job to run with the same priority (aka nice value) as the users process which launched the job on the submit node. If set to "0", or left unset, the users job will inherit the scheduling priority from the **SLURM** daemon.

PropagateResourceLimits

A list of comma-separated resource limit names. The **SLURMD** daemon uses these names to obtain the associated (soft) limit values from the users process environment on the submit node. These limits are then propagated and applied to the jobs that will run on the compute nodes. This parameter can be useful when system limits vary among nodes. Any resource limits that do not appear in the list are not propagated. However, the user can override this by specifying which resource limits to propagate with the **SRUN** commands "**--propagate**" option. If neither of the propagate resource limit's parameters are specified, then the default action is to propagate all limits. Only one of the parameters, either **PropagateResourceLimits** or **PropagateResourceLimitsExcept**, may be specified.

PropagateResourceLimitsExcept

A list of comma-separated resource limit names. By default, all resource limits will be propagated, (as described by the **PropagateResourceLimits** parameter), except for the limits appearing in this list. The user can override this by specifying which resource limits to propagate with the **SRUN** commands "**--propagate**" option.

ReturnToService

If set to 1, then a DOWN node will become available for use upon registration. The default value is 0, which means that a node will remain in the DOWN state until the system administrator explicitly changes its state (even if the SLURMD daemon registers and resumes communications).

SchedulerAuth

An authentication token, which must be used in a scheduler communication protocol. The interpretation of this value depends upon the value of **SchedulerType**. In the Wiki scheduler plug-in, this value must correspond to the checksum seed with which Maui was compiled.

SchedulerPort

The port number on which **SLURMCTLD** should listen for connection requests. This value is only used by the Maui Scheduler (see SchedulerType).

SchedulerRootFilter

Identifies whether or not **RootOnly** partitions should be filtered from any external scheduling activities. If set to 0, then **RootOnly** partitions are treated like any other partition. If set to 1, then **RootOnly** partitions are exempt from any external scheduling activities. The default value is 1. Currently only used by the built-in backfill scheduling module **sched/backfill** (see SchedulerType).

SchedulerType

Identifies the type of scheduler to be used. Acceptable values include **sched/builtin** for the built-in FIFO scheduler, **sched/backfill** for a backfill scheduling module to augment the default FIFO scheduling, **sched/hold** to hold all newly arriving jobs if a file **/etc/slurm.hold** exists otherwise use the built-in FIFO scheduler, and **sched/wiki** for the Wiki interface to the Maui Scheduler. The default value is **sched/builtin**. Backfill scheduling will initiate lower-priority jobs if doing so does not delay the expected initiation time of any higher priority job. Note that this backfill scheduler implementation is relatively simple. It does not support partitions configured to share resources (run multiple jobs on the same nodes) or support jobs requesting specific nodes. When initially setting the value to **sched/wiki**, any pending jobs must have their priority set to zero (held). When changing the value from sched/wiki, all pending jobs should have their priority change from zero to some large number. The **SCONTROL** command can be used to change job priorities. The **SLURMCTLD** daemon must be restarted for a change in scheduler type to become effective.

SelectType

Identifies the type of resource selection algorithm to be used. Acceptable values include **select/linear** for allocation of entire nodes assuming a one-dimensional array of nodes in which sequentially ordered nodes are preferable, **select/cons_res** for allocation of individual processors within the available nodes. The default value is **select/linear**.

SlurmUser

The name of the user under which the **SLURMCTLD** daemon executes. For security purposes, a user other than "**root**" is recommended. The default value is "**root**".

SlurmctldDebug

The level of detail to provide **SLURMCTLD** daemon's logs. Values from 0 to 7 are legal, with "0" being "**quiet**" operation, and "7" being extremely verbose. The default value is 3.

SlurmctldLogFile

Fully-qualified pathname of a file into which the **SLURMCTLD** daemon's logs are written. The default value is none (performs logging via **syslog**).

SlurmctldPidFile

Fully-qualified pathname of a file into which the SLURMCTLD daemon may write its process id. This may be used for automated signal processing. The default value is "/var/run/slurmctld.pid".

SlurmctldPort

The port number that the **SLURM** controller, **SLURMCTLD**, listens to for work. The default value is **SLURMCTLD_PORT** as established at system build time.

Note:

Either **SLURMCTLD** and **SLURMD** daemons must not execute on the same nodes or the values of **SlurmctldPort** and **SlurmdPort** must be different.

SlurmctldTimeout

The interval, in seconds, that the backup controller waits for the primary controller to respond before assuming control. The default value is 120 seconds. May not exceed 65533.

SlurmdDebug

The level of detail to provide **SLURMD** daemon's logs. Values from 0 to 7 are legal, with “0” being "quiet" operation, and “7” being extremely verbose. The default value is 3.

SlurmdLogFile

Fully-qualified pathname of a file into which the SLURMD daemon's logs are written. The default value is none (performs logging via syslog). Any "%h" within the name is replaced with the hostname on which the SLURMD is running.

SlurmdPidFile

Fully-qualified pathname of a file into which the **SLURMD** daemon may write its process id. This may be used for automated signal processing. The default value is **/var/run/slurmd.pid**.

SlurmdPort

The port number that the SLURM compute node daemon, **SLURMD**, listens to for work. The default value is **SLURMD_PORT** as established at system build time.

SlurmdSpoolDir

Fully-qualified pathname of a directory into which the SLURMD daemon's state information and batch job script information is written. This must be a common pathname for all nodes, but should represent a directory that is local to each node (reference a local file system). The default value is **/var/spool/slurmd**.

Note:

This directory is also used to store SLURMD's shared memory lockfile, and should not be changed unless the system is being cleanly restarted. If the location of **SlurmdSpoolDir** is changed and **SLURMD** is restarted, the new daemon will attach to a different shared memory region and lose track of any running jobs.

SlurmdTimeout

The interval, in seconds, that the SLURM controller waits for **SLURMD** to respond before configuring that node's state to DOWN. The default value is 300 seconds. A value of zero indicates the node will not be tested by **SLURMCTLD** to confirm the state of **SLURMD**, the node will not be automatically set to a DOWN state indicating a non-responsive **SLURMD**, and some other tool will take responsibility for monitoring the state of each compute node and its **SLURMD** daemon. The value may not exceed 65533.

StateSaveLocation

Fully-qualified pathname of a directory into which the SLURM controller, SLURMCTLD, saves its state (e.g. **/usr/local/slurm/checkpoint**). **SLURM** state will be saved here to recover from system failures. **SlurmUser** must be able to create files in this directory. If a **BackupController** is configured, this location should be readable and writable by both systems. The default value is "**/tmp**". If any SLURM daemons terminate abnormally, their core files will also be written into this directory.

SrunEpilog

Fully-qualified pathname of an executable to be run by SRUN following the completion of a job step. The command line arguments for the executable will be the command and arguments of the job step. This configuration parameter may be overridden by srun's **--epilog** parameter.

SrunProlog

Fully-qualified pathname of an executable to be run by **SRUN** prior to the launch of a job step. The command line arguments for the executable will be the command and arguments of the job step. This configuration parameter may be overridden by the **SRUN --prolog** parameter.

SwitchType

Identifies the type of switch or interconnect used for application communications. Acceptable values include "**switch/none**" for switches not requiring special processing for job launch or termination (**Myrinet**, **Ethernet**, and **InfiniBand**), **switch/elan** for Quadrics Elan 3 or Elan 4 interconnects. The default value is **switch/none**. All SLURM daemons, commands and running jobs must be restarted for a change in **SwitchType** to take effect. If running jobs exist at the time **SLURMCTLD** is restarted with a new value of **SwitchType**, records of all jobs in any state may be lost.

TaskEpilog

Fully qualified pathname of a program to be executed as the SLURM job's owner after termination of each task. See **TaskPlugin** for execution order details.

TaskPlugin

Identifies the type of task launch plug-in, typically used to provide resource management within a node (e.g. pinning tasks to specific processors). Acceptable values include **task/none** for systems requiring no special handling or **tasks/affinity** to enable the **--cpu_bind** and/or **--mem_bind** affinity **SRUN** options. The default value is **task/none**. The order of task prolog/epilog execution is as follows:

1. pre_launch(): function in **TaskPlugin**
2. **TaskProlog**: system-wide per task program defined in **slurm.conf**
3. user prolog: job step specific task program defined using the SRUN --task-prolog option or **SLURM_TASK_PROLOG** environment variable
4. Execute the job step's task user epilog: job step specific task program defined using the **SRUN --task-epilog** option or **SLURM_TASK_EPILOG** environment variable
5. **TaskEpilog**: system-wide per task program defined in **slurm.conf**
6. **post_term()**: function in **TaskPlugin**

TaskProlog

Fully-qualified pathname of a program to be executed as the **SLURM** job's owner prior to the initiation of each task. Aside from the normal environment variables, this has **SLURM_TASK_PID** available to identify the process ID of the task being started. Standard output from this program of the form **export NAME=value** will be used to set environment variables for the task being spawned.

See: TaskPlugin for execution order details.

TmpFS

Fully-qualified pathname of the file system available to user jobs for temporary storage. This parameter is used in establishing a node's TmpDisk space. The default value is **/tmp**.

TreeWidth

SLURMD daemons use a virtual tree network for communications. **TreeWidth** specifies the width of the tree (i.e. the fanout). The default value is 50.

UseCPUSETS

When set to 1, this determines whether CPUSETS (Multiprocessor partitioning for Linux) will be used.

CPUSETs are lightweight objects in the Linux kernel that enable users to partition their multiprocessor servers by creating execution areas. The ultimate objective is to contain processes to a certain number of well-identified processors.

CPUSETs:

- Allow the creation of sets of CPUs on the system, and bind applications to them.
- Provide a way to create sets of CPUs inside a specific CPU set: hence a system administrator can partition a system among users, and users can then further partition their partition among their applications.
- Memory used by a CPUSET may be restricted to some of the nodes of a NUMA system.

Note:

This parameter is only effective if the TaskPlugin parameter (see above) is set to **task/affinity**. The use of this parameter also requires that the **libcpuset.so** library be installed on the compute nodes.

Example

```
UseCPUSETS=1 or UseCPUSETS=Yes   # default is not to use CPUSETS
```

UsePAM

If set to 1, **PAM (Pluggable Authentication Modules for Linux)** will be enabled. **PAM** is used to establish the upper bounds for resource limits. With PAM support enabled, local system administrators can dynamically configure system resource limits. Changing the upper bound of a resource limit will not alter the limits of running jobs, only jobs started after a change has been made will pick up the new limits. The default value is 0 (not to enable **PAM** support).

Remember that **PAM** also needs to be configured to support **SLURM** as a service. For sites using PAM's directory based configuration option, a configuration file named SLURM should be created. The module-type, control-flags, and module-path names that should be included in the file are: **auth required pam_localuser.so**, **auth required pam_shells.so, account required pam_unix.so**, **account required pam_access.so** and **session required pam_unix.so**. For sites configuring PAM with a general configuration file, the appropriate lines (see above), where SLURM is the service-name, should be added.

WaitTime

Specifies how many seconds the **SRUN** command should by default wait after the first task terminates before terminating all remaining tasks. The **--wait** option on the SRUN command line overrides this value. If set to 0, this feature is disabled. May not exceed 65533.

6.4.1.2 Node Configuration Parameters

The configuration of nodes (or machines) that will be managed by **SLURM** is also specified in **/etc/slurm/slurm.conf**. Only the **NodeName** must be supplied in the configuration file. All other node configuration information is optional. It is advisable to establish baseline node configurations, especially if the cluster is heterogeneous. Nodes that register to the system with less than the configured resources (e.g. too little memory) will be placed in the "DOWN" state to avoid having any jobs scheduled on them. Establishing baseline configurations will also speed SLURM's scheduling process by permitting it to compare job requirements against these (relatively few) configuration parameters and possibly avoid having to check job requirements against every individual node's configuration.

The resources checked at node registration time are: **Procs**, **RealMemory** and **TmpDisk**. While baseline values for each of these can be established in the configuration file, the actual values upon node registration are recorded and these actual values may be used for scheduling purposes (depending upon the value of **FastSchedule** in the configuration file).

Default values can be specified with a record in which **NodeName** is DEFAULT. The default entry values will apply only to lines that follow it in the configuration file; the default values can be reset multiple times in the configuration file with multiple entries where "**NodeName=DEFAULT**". The "**NodeName=**" specification must be placed on every line describing the configuration of nodes. In fact, it is possible and desirable to define the configurations of all nodes in only a few lines. This convention permits significant optimization in the scheduling of larger clusters. In order to support the concept of jobs requiring consecutive nodes on the some architecture, node specifications should be placed in this file in consecutive order. No single node name may be listed more than once in the configuration file.

Use "**DownNodes=**" to record the state of nodes which are temporarily in a DOWN or DRAINED state without altering permanent configuration information. A job step's tasks are allocated to nodes in the order in which the nodes appear in the configuration file. There is presently no capability within SLURM to arbitrarily order a job step's tasks.

A simple node range expression may optionally be used to specify ranges of nodes to avoid building a configuration file with a large numbers of entries. The node range expression can contain one pair of square brackets with a sequence of comma-separated numbers and/or ranges of numbers separated by a "-" (e.g. "**linux[0-64,128]**", or "**lx[15,18,32-33]**"). Presently the numeric range must be the last characters in the node name (e.g. "**unit[0-31]rack1**" is invalid).

The node configuration specifies the following information:

NodeName

The Name that SLURM uses to refer to a node. Typically this would be the string that **/bin/hostname –s** returns; however, it may be an arbitrary string if **NodeHostname** is specified. If the **NodeName** is "DEFAULT", the values specified with that record will apply to subsequent node specifications unless explicitly set to other values in that node record or replaced with a different set of default values. For architectures in which the node order is significant, nodes will be considered consecutive in the order defined. For example, if the configuration for **NodeName=charlie** immediately follows the configuration for **NodeName=baker** they will be considered adjacent in the computer.

NodeHostname

The string that **/bin/hostname -s** returns. A node range expression can be used to specify a set of nodes. If an expression is used, the number of nodes identified by **NodeHostname** on a line in the configuration file must be identical to the number of nodes identified by **NodeName**. By default, the **NodeHostname** will be identical in value to **NodeName**.

NodeAddr

Name by which a node should be referred when establishing a communications path. This name will be used as an argument to the **gethostbyname()** function for identification. If a node range expression is used to designate multiple nodes, they must exactly match the entries in the **NodeName** (e.g. "**NodeName=lx[0-7] NodeAddr="elx[0-7]"**). **NodeAddr** may also contain IP addresses. By default, the **NodeAddr** will be identical in value to **NodeName**.

DownNodes

Any node name, or list of node names, from the **NodeName=** specifications. The **DownNodes=** configuration permits marking certain nodes as being in a DOWN or DRAINED state without altering the permanent configuration information listed under a **NodeName=** specification.

Feature

A comma-delimited list of arbitrary strings indicative of some characteristic associated with the node. There is no value associated with a feature at this time, a node either has a feature or it does not. If desired, a feature may contain a numeric component indicating, for example, processor speed. By default a node has no features.

Procs

Number of processors on the node (e.g. "**2**"). The default value is **1**.

RealMemory

Size of real memory on the node in MegaBytes (e.g. "**2048**"). The default value is 1.

Reason

Identifies the reason for a node being in state **DOWN** or **DRAINED** or **DRAINING**. Use quotes to enclose a reason having more than one word.

State

State of the node with respect to the initiation of user jobs. Acceptable values are **BUSY**, **DOWN**, **DRAINED**, **DRAINING**, **IDLE**, and **UNKNOWN**.

- **BUSY** indicates the node has been allocated work and should not be used in the configuration file.
- **DOWN** indicates the node failed and is unavailable to be allocated work.
- **DRAINED** indicates the node was configured unavailable to be allocated work and is presently not performing any work.
- **DRAINING** indicates the node is unavailable to be allocated new work, but is completing the processing of a job.
- **IDLE** indicates the node available to be allocated work, but has none at present
- **UNKNOWN** indicates the node's state is undefined, but will be established when the SLURMD daemon on that node registers.

The default value is UNKNOWN.

TmpDisk

Total size of temporary disk storage in **TmpFS** in MegaBytes (e.g. "16384"). **TmpFS** (for "Temporary File System") identifies the location that jobs should use for temporary storage. Note that this does not indicate the amount of free space available to the user

on the node, only the total file system size. The system administration should ensure that this file system is purged as needed, thus allowing user jobs to have access to most of this space. The **Prolog** and/or **Epilog** programs (specified in the configuration file) might be used to ensure that the file system is kept clean. The default value is 1.

Weight

The priority of the node for scheduling purposes. All things being equal, jobs will be allocated the nodes with the lowest weight that satisfies their requirements. For example, a heterogeneous collection of nodes might be placed into a single partition for greater system utilization, responsiveness and capability. It would be preferable to allocate smaller memory nodes rather than larger memory nodes if either will satisfy a job's requirements. The units of weight are arbitrary, but larger weights should be assigned to nodes with more processors, memory, disk space, higher processor speed, etc. Weight is an integer value with a default value of 1.

6.4.1.3 Partition Configuration Parameters

The partition configuration permits different job limits or access controls to be established for various groups (or partitions) of nodes. Nodes may be in more than one partition, making partitions serve as general-purpose queues. For example, the same set of nodes may be put into two different partitions, each with different constraints (time limit, job sizes, groups allowed to use the partition, etc.). Jobs are allocated resources within a single partition.

The partition configuration file contains the following information:

AllowGroups

Comma-separated list of group IDs that may execute jobs in the partition. If at least one group associated with the user attempting to execute the job is in AllowGroups, he will be permitted to use this partition. Jobs executed as user root can use any partition without regard to the value of AllowGroups. If user root attempts to execute a job as another user (e.g. using SRUN'S **--uid** option), this other user must be in one of the groups identified by **AllowGroups** for the job to successfully execute. The default value is "ALL".

Default

If this keyword is set, jobs submitted without a partition specification will utilize this partition. Possible values are "YES" and "NO". The default value is "NO".

Hidden

Specifies if the partition and its jobs are to be hidden by default. Hidden partitions will by default not be reported by the SLURM APIs or commands. Possible values are "YES" and "NO". The default value is "NO".

RootOnly

Specifies if only user ID zero (i.e. user root) may allocate resources in this partition. User root may allocate resources for any other user, but the request must be initiated by user root. This option can be useful for a partition which is to be managed by some external entity (e.g. a higher-level job manager) and prevents users from directly using those resources. Possible values are "YES" and "NO". The default value is "NO".

MaxNodes

Maximum count of nodes that may be allocated to any single job. The default value is "UNLIMITED", which is represented internally as -1. This limit does not apply to jobs executed by **SlurmUser** or user root.

MaxTime

Maximum wall-time limit for any job in minutes. The default value is "UNLIMITED", which is represented internally as -1. This limit does not apply to jobs executed by **SlurmUser** or user root.

MinNodes

Minimum count of nodes that may be allocated to any single job. The default value is 1. This limit does not apply to jobs executed by **SlurmUser** or user root.

Nodes

Comma-separated list of nodes that are associated with this partition. Node names may be specified using the node range expression syntax described above. A blank list of nodes (i.e. "Nodes= ") can be used if one wants a partition to exist, but have no resources (possibly on a temporary basis).

PartitionName

Name by which the partition may be referenced (e.g. "Interactive"). Users can specify this name when submitting jobs.

Shared

Ability of the partition to execute more than one job at a time on each node. Shared nodes will offer unpredictable performance for application programs, but can provide higher system utilization and responsiveness than otherwise possible. Possible values are **FORCE**, **YES**, **and NO**.

- **FORCE** makes all nodes in the partition available for sharing without user means of disabling it.
- **YES** makes nodes in the partition available for sharing if and only if the individual jobs permit sharing (see the SRUN "**--shared**" option).
- **NO** makes nodes unavailable for sharing under all circumstances. The default value is NO

State

State of partition or availability for use. Possible values are **UP** or **DOWN**. The default value is **UP**.

6.4.2 slurm.conf Example Files

This section provides two examples of **slurm.conf** files.

Example #1

```
ControlMachine=linux0
ControlAddr=linux0
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdLogFile=/var/log/slurm/slurmd.log.%h
```

```
StateSaveLocation=/var/log/slurm/log_slurmctld
SlurmdSpoolDir=/var/log/slurm/log_slurmd/
SlurmUser=slurm
SlurmctldDebug=3    # default is 3
SlurmdDebug=3       # default is 3
SlurmctldPort=6817
SlurmdPort=6818
AuthType=auth/none
SelectType=select/linear
SchedulerType=sched/builtin   # default is sched/builtin
#JobCompType=jobcomp/filetxt  # default is jobcomp/none
#JobCompLoc=/var/log/slurm/slurm.job.log
SwitchType=switch/none
ProctrackType=proctrack/pgid
#JobAcctType=jobacct/linux   # default is jobacct/none
#JobAcctLogFile=/var/log/slurm/slurm_acct.log

FastSchedule=1      # default is `1'
FirstJobid=1000     # default is `1'
ReturnToService=0   # default is `0'
MpiDefault=none     # default is "none"

JobCredentialPrivateKey=/etc/slurm/slurm.key
JobCredentialPublicCertificate=/etc/slurm/slurm.cert

# NODE CONFIGURATION
NodeName=linux[10-37] Procs=8 State=UNKNOWN

# PARTITION CONFIGURATION
PartitionName=global Nodes=linux[10-37] State=UP Default=YES
PartitionName=test Nodes=linux[10-20] State=UP
PartitionName=debug Nodes=linux[21-30] State=UP
```

Example #2

```
#
# Sample slurm.conf for mcr.llnl.gov
#
ControlMachine=mcri ControlAddr=emcri
BackupMachine=mcrj BackupAddr=emcrj
#
AuthType=auth/munge
Epilog=/usr/local/slurm/etc/epilog
FastSchedule=1
JobCompLoc=/var/tmp/jette/slurm.job.log
JobCompType=jobcomp/filetxt
JobCredentialPrivateKey=/usr/local/etc/slurm.key
JobCredentialPublicCertificate=/usr/local/etc/slurm.cert
PluginDir=/usr/local/slurm/lib/slurm
Prolog=/usr/local/slurm/etc/prolog
SchedulerType=sched/backfill
SelectType=select/linear
SlurmUser=slurm
SlurmctldPort=7002
SlurmctldTimeout=300
SlurmdPort=7003
```

```
SlurmdSpoolDir=/var/tmp/slurmd.spool
SlurmdTimeout=300
StateSaveLocation=/tmp/slurm.state
SwitchType=switch/elan
TreeWidth=50
#
# Node Configurations
#
NodeName=DEFAULT Procs=2 RealMemory=2000 TmpDisk=64000
State=UNKNOWN
NodeName=mcr[0-1151] NodeAddr=emcr[0-1151]
#
# Partition Configurations
#
PartitionName=DEFAULT State=UP
PartitionName=pdebug Nodes=mcr[0-191] MaxTime=30 MaxNodes=32
Default=YES
PartitionName=pbatch Nodes=mcr[192-1151]
```

6.4.3 SCONTROL – Managing the SLURM Configuration

SCONTROL manages available nodes (for example, by "draining" jobs from a node or partition to prepare it for servicing). It is also used to manage the **SLURM** configuration and the properties assigned to nodes, node partitions and other SLURM-controlled system features.

Note:
Most **SCONTROL** options and commands can only be used by System Administrators. Some **SCONTROL** commands *report* useful configuration information or manage job *checkpoints*, and any user can benefit from invoking them appropriately.

NAME

SCONTROL - Used to view and modify SLURM configuration and state.

SYNOPSIS

```
SCONTROL   [OPTIONS...] [COMMAND...]
```

DESCRIPTION

SCONTROL is used to view or modify the SLURM configuration including: job, job step, node, partition, and overall system configuration. Most of the commands can only be executed by user root. If an attempt to view or modify configuration information is made by an unauthorized user, an error message will be printed and the requested action will not occur. If no command is entered on the execute line, SCONTROL will operate in an interactive mode and prompt for input. It will continue prompting for input and executing commands until explicitly terminated. If a command is entered on the execute line, SCONTROL will execute that command and terminate. All commands and options are case-insensitive, although node names and partition names are case-sensitive (node names

"LX" and "lx" are distinct). Commands can be abbreviated to the extent that the specification is unique.

6.4.3.1 OPTIONS

-a, --all

When the show command is used, then it displays all partitions, their jobs and jobs steps. This causes information to be displayed about partitions that are configured as hidden and partitions that are unavailable to user's group.

-h, --help

Print a help message describing the usage of SCONTROL.

--hide

Do not display information about hidden partitions, their jobs and job steps. By default, neither partitions that are configured as hidden nor those partitions unavailable to a user's group will be displayed (i.e. this is the default behavior).

-o, --oneliner

Print information one line per record.

-q, --quiet

Print no warning or informational messages, only fatal error messages.

-v, --verbose

Print detailed event logging. This includes time-stamps on data structures, record counts, etc.

-V , --version

Print version information and exit.

6.4.3.2 Commands

all

Show all partitions, their jobs and jobs steps. This causes information to be displayed about partitions that are configured as hidden and partitions that are unavailable to a user's group.

abort

Instruct the SLURM controller to terminate immediately and generate a core file.

checkpoint CKPT_OP ID

Perform a checkpoint activity on the job step(s) with the specified identification. CKPT_OP may take one of the following values: "disable" (disable future checkpoints), "enable" (enable future checkpoints), "able" (test if presently not disabled, report start time if checkpoint in progress), "create" (create a checkpoint and continue the job step), "vacate" (create a checkpoint and terminate the job step), "error" (report the result for the last checkpoint request, error code and message), or "restart" (restart execution of the previously checkpointed job steps). ID can be used to identify a

specific job (e.g. "<job_id>", which applies to all of its existing steps) or a specific job step (e.g. "<job_id>.<step_id>").

completing

Display all jobs in a COMPLETING state along with associated nodes in either a COMPLETING or DOWN state.

delete SPECIFICATION

Delete the entry with the specified SPECIFICATION. The only supported SPECIFICATION presently is of the form PartitionName=<name>.

exit

Terminate the execution of SCONTROL.

help

Display a description of SCONTROL options and commands.

hide

Do not display partition, job, or job-step information for partitions that are configured as hidden or partitions that are unavailable to the user's group. This is the default behavior.

oneliner

Print information one line per record.

pidinfo PROC_ID

Print the SLURM job id and scheduled termination time corresponding to the supplied process id, PROC_ID, on the current node. This will only work for processes that SLURM spawns and their descendants.

ping

Ping the primary and secondary SLURMCTLD daemon and report if they are responding.

quiet

Print no warning or informational messages, only fatal error messages.

quit

Terminate the execution of SCONTROL.

reconfigure

Instruct all SLURM daemons to re-read the configuration file. This command does not restart the daemons. This mechanism would be used to modify configuration parameters (**Epilog**, **Prolog**, **SlurmctldLogFile**, **SlurmdLogFile**, etc.), register the physical addition or removal of nodes from the cluster or recognize the change of a node's configuration, such as the addition of memory or processors. The SLURM controller (**SLURMCTLD**) forwards the request to all other daemons (**SLURMD** daemon on each compute node). Running jobs continue execution. Most configuration parameters can be changed by just running this command, however, SLURM daemons should be shutdown and restarted if any of the following parameters are to be changed:

AuthType, BackupAddr, BackupController, ControlAddr, ControlMach, PluginDir, StateSaveLocation, SlurmctldPort or SlurmdPort.

resume job_id

Resume a previously suspended job.

show ENTITY ID

Display the state of the specified entity with the specified identification. ENTITY may be config, daemons, job, node, partition or step. ID can be used to identify a specific element of the identified entity: the configuration parameter name, job ID, node name, partition name, or job step ID for entities config, job, node, partition, and step respectively. Multiple node names may be specified using simple node range expressions (e.g. "lx[10-20]"). All other ID values must identify a single element. The job step ID is of the form "job_id.step_id", (e.g. "1234.1"). By default, all elements of the entity type specified are printed.

shutdown

Instruct all SLURM daemons to save current state and terminate. The SLURM controller (SLURMCTLD) forwards the request all other daemons (SLURMD daemon on each compute node).

suspend job_id

Suspend a running job. Use the resume command to resume its execution. User processes must stop on receipt of SIGSTOP signal and resume upon receipt of SIGCONT for this operation to be effective. Not all architectures and configurations support job suspension.

update SPECIFICATION

Update job, node or partition configuration per the supplied specification. SPECIFICATION is in the same format as the SLURM configuration file and the output of the show command described above. It may be desirable to execute the show command (described above) on the specific entity that is to be updated, use cut-and-paste tools to enter updated configuration values to the update. Note that while most configuration values can be changed using this command, not all can be changed using this mechanism. In particular, the hardware configuration of a node or the physical addition or removal of nodes from the cluster may only be accomplished through editing the SLURM configuration file and executing the reconfigure command (described above).

verbose

Print detailed event logging. This includes time-stamps on data structures, record counts, etc.

version

Display the version number of SCONTROL being executed.

!!

Repeat the last command executed.

6.4.3.3 Specifications for the Update Command - Jobs

Account=<account>

Account name to be changed for this job's resource use. Value may be cleared with blank data value, "Account=".

Contiguous=<yes|no>

Set the job's requirement for contiguous (consecutive) nodes to be allocated. Possible values are "YES" and "NO".

Dependency=<job_id>

Defer job's initiation until specified job_id completes. Cancel dependency with job_id value of "0", "Depedency=0".

Features=<features>

Set specified features to the job's required features on nodes. Multiple values may be comma separated if all features are required (AND operation) or separated by "|" if any of the specified features are required (OR operation). Value may be cleared with blank data value, "Features=".

JobId=<id>

Identify the job to be updated. This specification is required.

MinMemory=<megabytes>

Set the job's minimum real memory required per node to the specified value.

MinProcs=<count>

Set the job's minimum number of processors per node to the specified value.

MinTmpDisk=<megabytes>

Set the job's minimum temporary disk space required per node to the specified value.

Name=<name>

Set the job's name to the specified value.

Partition=<name>

Set the job's partition to the specified value.

Priority=<number>

Set the job's priority to the specified value. Note that a job priority of zero prevents the job from ever being scheduled. By setting a job's priority to zero, it is held. Set the priority to a non-zero value to permit it to run.

Nice[=delta]

Adjust job's priority to the specified value. Default value is 100.

ReqNodeList=<nodes>

Set the job's list of required nodes. Multiple node names may be specified using simple node range expressions (e.g. "lx[10-20]"). Value may be cleared with blank data value, "ReqNodeList=".

ReqNodes=<count>

Set the job's count of required nodes to the specified value.

ReqProcs=<count>

Set the job's count of required processors to the specified value.

Shared=<yes|no>

Set the job's ability to share nodes with other jobs. Possible values are "YES" and "NO".

StartTime=<time_spec>

Set the job's earliest initiation time. It accepts times of the form HH:MM:SS to run a job at a specific time of day (seconds are optional). (If that time is already past, the next day is assumed.) You may also specify midnight, noon, or teatime (4pm) and you can have a time-of-day suffixed with AM or PM for running in the morning or the evening. It is also possible to specify the day on which the job will be run, by giving a date in the form MMDDYY or MM/DD/YY or MM.DD.YY. It is also possible to give times, such as now + count time-units, where the time-units can be minutes, hours, days, or weeks and SLURM can be told to run the job today with the keyword today and to run the job tomorrow with the keyword tomorrow.

Notes for date/time specifications:

- Although the 'seconds' field of the HH:MM:SS time specification is allowed by the code, the poll time of the SLURM scheduler is not precise enough to guarantee dispatch of the job on the exact second. The job will be eligible to start on the next poll following the specified time. The exact poll interval depends on the SLURM scheduler (e.g. 60 seconds with the default **sched/builtin**).
- If no time (HH:MM:SS) is specified, the default is (00:00:00).
- If a date is specified without a year (e.g. MM/DD) then the current year is assumed, unless the combination of MM/DD and HH:MM:SS has already passed for that year, in which case the next year is used.

TimeLimit=<minutes>

Set the job's time limit to the specified value.

Connection=<type>

Reset the node connection type.

Geometry=<geo>

Reset the required job geometry.

Rotate=<yes|no>

Permit the job's geometry to be rotated. Possible values are "YES" and "NO".

6.4.3.4 Specifications for the Update Command - Nodes

NodeName=<name>

Identify the node(s) to be updated. Multiple node names may be specified using simple node range expressions (e.g. "lx[10-20]"). This specification is required.

Reason=<reason>

Identify the reason why the node is in a "DOWN" or "DRAINED" or "DRAINING" state. Use quotes to enclose a reason having more than one word.

State=<state>

Identify the state to be assigned to the node. Possible values are "NoResp", "DRAIN" "RESUME", "DOWN", "IDLE", "ALLOC", and "ALLOCATED". "RESUME is not an actual node state, but it will return a DRAINED, DRAINING, or DOWN node to service, either IDLE or ALLOCATED state as appropriate. The "NoResp" state will only set the "NoResp" flag for a node without changing its underlying state.

6.4.3.5 Specifications for Update and Delete Commands - Partitions

AllowGroups=<name>

Identify the user groups that may use this partition. Multiple groups may be specified in a comma-separated list. To permit all groups to use the partition specify "AllowGroups=ALL".

Default=<yes|no>

Specify if this partition is to be used by jobs that do not explicitly identify a partition to use. Possible values are "YES" and "NO".

Hidden=<yes|no>

Specify if the partition and its jobs should be hidden from view. Hidden partitions will by default not be reported by SLURM APIs or commands. Possible values are "YES" and "NO".

Nodes=<name>

Identify the node(s) to be associated with this partition. Multiple node names may be specified using simple node range expressions (e.g. "lx[10-20]"). Note that jobs may only be associated with one partition at any time. Specify a blank data value to remove all nodes from a partition: "Nodes=".

PartitionName=<name>

Identify the partition to be updated. This specification is required.

RootOnly=<yes|no>

Specify if only allocation requests initiated by user root will be satisfied. This can be used to restrict control of the partition to some meta-scheduler. Possible values are "YES" and "NO".

Shared=<yes|no|force>

Specify if nodes in this partition can be shared by multiple jobs. Possible values are "YES", "NO", and "FORCE".

State=<up|down>

Specify if jobs can be allocated nodes in this partition. Possible values are "UP" and "DOWN". If a partition allocated nodes to running jobs, those jobs will continue execution even after the partition's state is set to "DOWN". The jobs must be explicitly canceled to force their termination.

MaxNodes=<count>

Set the maximum number of nodes that will be allocated to any single job in the partition. Specify a number or "INFINITE".

MinNodes=<count>

Set the minimum number of nodes that will be allocated to any single job in the partition.

6.4.3.6 ENVIRONMENT VARIABLES

Some SCONTROL options may be set via environment variables. These environment variables, along with their corresponding options, are listed below. (Note: Command-line options will always override these settings.)

SLURM_CONF The location of the SLURM configuration file.

SCONTROL_ALL `-a, --all`

6.4.3.7 SCONTROL EXAMPLE

```
# scontrol
scontrol: show part class
PartitionName=class TotalNodes=10 TotalCPUs=20 RootOnly=NO
   Default=NO Shared=NO State=UP MaxTime=30 Hidden=NO
   MinNodes=1 MaxNodes=2 AllowGroups=students
   Nodes=lx[0031-0040] NodeIndices=31,40,-1
scontrol: update PartitionName=class MaxTime=99 MaxNodes=4
scontrol: show job 65539
JobId=65539 UserId=1500 JobState=PENDING TimeLimit=100
   Priority=100 Partition=batch Name=job01 NodeList=(null)
   StartTime=0 EndTime=0 Shared=0 ReqProcs=1000
   ReqNodes=400 Contiguous=1 MinProcs=4 MinMemory=1024
   MinTmpDisk=2034ReqNodeList=lx[3000-3003]
   Features=(null) JobScript=/bin/hostname
scontrol: update JobId=65539 TimeLimit=200 Priority=500
scontrol: quit
```

6.4.4 Pam_Slurm Module Configuration

This section describes how to use the **pam_slurm** module. This module restricts access to compute nodes in a cluster where Simple Linux Utility for Resource Management (**SLURM**) is in use. Access is granted to root, any user with a SLURM-launched job currently running on the node, or any user who has allocated resources on the node according to the **SLURM** database.

Use of this module is recommended on any compute node where it is desirable to limit access to just those users who are currently scheduled to run jobs.

For systems using **PAM** (Pluggable Authentication Modules), the modules are typically located under **/lib** although under some systems (often **x86_64** based), modules can be located under **/lib64** instead.

Check where the **<pam_slurm.so>** is located on your system and add the following line to **/etc/pam.d/system-auth**:

```
account    required     /<lib|lib64>/security/pam_slurm.so
```

If it is necessary to allow access, all the time, for an administrative group (for example, wheel), stack the **pam_access** module ahead of **pam_slurm**:

```
account    sufficient   /lib/security/pam_access.so
account    required     /<lib|lib64>/security/pam_slurm.so
```

Then edit the **pam_access** configuration file (**/etc/security/access.conf**):

```
+:wheel:ALL
-:ALL:ALL
```

When access is denied because the user does not have an active job running on the node, an error message is returned to the application:

```
Access denied: user foo (uid=1313) has no active jobs.
```

This message can be suppressed by specifying the "no_warn" argument in the PAM configuration file.

6.5 Administrating Cluster Activity with SLURM

SLURM consists of two types of daemons.

- **SLURMCTLD** is sometimes called the "controller" daemon. It orchestrates **SLURM** activities, including queuing of job, monitoring node states, and allocating resources (nodes) to jobs. There is an optional backup controller that automatically assumes control in the event that the primary controller fails. The primary controller resumes control when it is restored to service. The controller saves its state to disk whenever there is a change. This state can be recovered by the controller at startup time. State changes are saved so that jobs and other states can be preserved when the controller moves (to or from a backup controller) or is restarted.

  Note that files and directories used by **SLURMCTLD** must be readable or writable by the user **SlurmUser** (the SLURM configuration files must be readable; the log file directory and state save directory must be writable).

- The **SLURMD** daemon executes on all Compute nodes. It resembles a remote shell daemon which exports control to SLURM. Because SLURMD initiates and manages user jobs, it must execute as the user **root**.

6.5.1 Starting the Daemons

The SLURM daemons are initiated at node startup time, provided by the **/etc/init.d/slurm** script. If needed, the **/etc/init.d/slurm** script can be used to check the status of the daemon, start, startclean or stop the daemon on the node.

Once a valid configuration has been set up and installed, the **SLURM** controller, **SLURMCTLD**, should be started on the primary and backup control machines, and the **SLURM** compute node daemon, **SLURMD**, should be started on each compute server. The **SLURMD** daemons need to run as root for production use, but may be run as a user for testing purposes (obviously no jobs should be running as any other user in the configuration). The SLURM controller, **SLURMCTLD**, must be run as the configured **SlurmUser** (see the configuration file).

For testing purposes it may be prudent to start by just running **SLURMCTLD** and **SLURMD** on one node. By default, they execute in the background. Use the **-D** option for each daemon to execute them in the foreground and logging will be done to the terminal. The **-v** option will log events in more detail with more **v**'s increasing the level of detail (e.g. -vvvvvv). One window can be used to execute **slurmctld -D -vvvvvv**, whilst **slurmd -D –vvvvv** is executed in a second window. Errors such as "*Connection refused*" or "*Node X not responding*" may be seen when one daemon is operative and the other is being started. However, the daemons can be started in any order and proper communications will be established once both daemons complete initialization. A third window can be used to execute commands such as, ***srun -N1 /bin/hostname***, to confirm functionality.

Another important option for the daemons is **-c** to clear the previous state information. Without the **-c** option, the daemons will restore any previously saved state information: node state, job state, etc. With the **-c** option all previously running jobs will be purged and the node state will be restored to the values specified in the configuration file. This means that a node configured down manually using the **SCONTROL** command will be returned to

service unless also noted as being down in the configuration file. In practice, **SLURM** restarts with preservation consistently.

6.5.2 Starting and Stopping the SLURM Daemons with slurm.sh

The **slurm.sh** script at **/etc/slurm/slurm.sh** may optionally be used to start and stop **SLURM** and to display the **SLURM** daemons on all of the nodes. It must be run as root.

The script must be edited and the following parameters in the script must be configured before using it.

1. **SLURM_NODES**
   These are the Compute Nodes in the cluster. The forms of syntax supported for this parameter are:
   Form #1: "node1 node2 node3"
   Form #2: node[1,2,3]
   Form #3: node[1-3]
   Form #4: node[6,15-30,36]
   For example:
   SLURM_NODES="bali[6,15-30,36]"
   If the Management Node is also going to be used as a Compute Node (that is, both **slurmd** and **slurmctld** daemons will run on the Management Node), then the SLURM_NODES list of nodes must include the Management Node.

2. **SLURM_OPTIONS_CONTROLLER** and **SLURM_OPTIONS_DAEMONS**
   These options determine how the **SLURM** daemons are started.
   To start SLURM without preserving its state, use:
   SLURM_OPTIONS_CONTROLLER="-c"
   SLURM_OPTIONS_DAEMONS="-c"
   To start SLURM and preserve its state, use:
   SLURM_OPTIONS_CONTROLLER=""
   SLURM_OPTIONS_DAEMONS=""

3. **SLURM_USER**
   This is the **SLURM** user as defined in **slurm.conf**, for example:
   SLURM_USER=slurm

4. **SLURM_HOME** and **SLURM_CONFIG**
   These options are already set by default:
   SLURM_HOME=/usr
   SLURM_CONFIG=/etc/slurm

After the **/etc/slurm/slurm.sh** parameters have been setup, it can be used to:

1. Start **SLURM** on all of the nodes:

```
/etc/slurm/slurm.sh start
```

2. Shutdown **SLURM**:

```
/etc/slurm/slurm.sh stop
```

3. Display the **SLURM** daemons running on the cluster:

```
/etc/slurm/slurm.sh
```

4. The kill option can be used to kill the **SLURMD** daemons if the **SLURMCTLD** daemon is not running:

```
/etc/slurm/slurm.sh kill
```

6.5.3 SLURMCTLD (Controller Daemon)

NAME

SLURMCTLD - The central management daemon of SLURM.

SYNOPSIS

slurmctld [OPTIONS...]

DESCRIPTION

SLURMCTLD is the central management daemon of SLURM. It monitors all other SLURM daemons and resources, accepts work (jobs), and allocates resources to those jobs. Given the critical functionality of SLURMCTLD, there may be a backup server to assume these functions in the event that the primary server fails.

OPTIONS

-c

Clear all previous SLURMCTLD states from its last checkpoint. If not specified, previously running jobs will be preserved along with the state of **DOWN**, **DRAINED** and **DRAINING** nodes and the associated reason field for those nodes.

-D

Debug mode. Execute SLURMCTLD in the foreground with logging to stdout.

-f <file>

Read configuration from the specified file. See NOTE under ENVIRONMENT VARIABLES below.

-h

Help; print a brief summary of command options.

-L <file>

Write log messages to the specified file.

-v

Verbose operation. Using more than one v (e.g., -vv, -vvv, -vvvv, etc.) increases verbosity.

-V

Print version information and exit.

ENVIRONMENT VARIABLES

The following environment variables can be used to override settings compiled into **SLURMCTLD**.

SLURM_CONF

The location of the SLURM configuration file. This is overridden by explicitly naming a configuration file in the command line.

Note:

It may be useful to experiment with different **SLURMCTLD**-specific configuration parameters using a distinct configuration file (e.g. timeouts). However, this special configuration file will not be used by the **SLURMD** daemon or the **SLURM** programs, unless each of them is specifically told to use it. To modify communication ports, the location of the temporary file system, or other parameters used by other **SLURM** components, change the common configuration file, **slurm.conf**.

6.5.4 SLURMD (Compute Node Daemon)

NAME

SLURMD - The compute node daemon for SLURM.

SYNOPSIS

slurmd [OPTIONS...]

DESCRIPTION

SLURMD is the compute node daemon of SLURM. It monitors all tasks running on the compute node, accepts work (tasks), launches tasks, and kills running tasks upon request.

OPTIONS

-c

Clear system locks as needed. This may be required if SLURMD terminated abnormally.

-D

Run **SLURMD** in the foreground. Error and debug messages will be copied to **stderr**.

-M

Lock **SLURMD** pages into system memory using **mlockall** to disable paging of the SLURMD process. This may help in cases where nodes are marked DOWN during periods of heavy swap activity. If the **mlockall** system call is not available, an error will be printed to the log and **SLURMD** will continue as normal.

-h

Help; print a brief summary of command options.

-f <file>

Read configuration from the specified file. See NOTES below.

-L <file>

Write log messages to the specified file.

-v

Verbose operation. Using more than one v (e.g., -vv, -vvv, -vvvv, etc.) increases verbosity.

-V

Print version information and exit.

ENVIRONMENT VARIABLES

The following environment variables can be used to override settings compiled into **SLURMD**.

SLURM_CONF

The location of the SLURM configuration file. This is overridden by explicitly naming a configuration file on the command line.

Note:

It may be useful to experiment with different **SLURMD**-specific configuration parameters using a distinct configuration file (e.g. timeouts). However, this special configuration file will not be used by the **SLURMD** daemon or the **SLURM** programs, unless each of them is specifically told to use it. To modify communication ports, the location of the temporary file system, or other parameters used by other SLURM components, change the common configuration file, **slurm.conf**.

6.5.5 Scheduler Support

The scheduler used by SLURM is controlled by the **SchedType** configuration parameter. This is meant to control the relative importance of pending jobs. SLURM's default scheduler is **FIFO** (First-In First-Out). A backfill scheduler plug-in is also available. Backfill scheduling will initiate a lower-priority job if doing so does not delay the expected initiation time of higher priority jobs; essentially using smaller jobs to fill holes in the resource allocation plan. SLURM also supports a plug-in for use of the **Maui** Scheduler, which offers sophisticated scheduling algorithms. Motivated users can even develop their own scheduler plug-in if so desired.

6.5.6 Node Selection

The node selection mechanism used by SLURM is controlled by the **SelectType** configuration parameter. If you want to execute multiple jobs per node, but apportion the processors, memory and other resources, the ***cons_res*** (consumable resources) plug-in is recommended. If you tend to dedicate entire nodes to jobs, the ***linear*** plug-in is recommended.

6.5.7 Logging

SLURM uses the **syslog** function to record events. It uses a range of importance levels for these messages. Be certain that your system's **syslog** functionality is operational.

6.5.8 Corefile Format

SLURM is designed to support generating a variety of core file formats for application codes that fail (see the ***--core*** option of the ***srun*** command).

6.5.9 Security

Unique job credential keys for each site should be created using the **openssl** program **openssl must be used (not ssh-genkey) to construct these keys.** An example of how to do this is shown below.

Specify file names that match the values of **JobCredentialPrivateKey** and **JobCredentialPublicCertificate** in the configuration file. The **JobCredentialPrivateKey** file must be readable only by **SlurmUser**. The **JobCredentialPublicCertificate** file must be readable by all users. Both files must be available on all nodes in the cluster. These keys are used by **slurmctld** to construct a job credential, which is sent to **srun** and then forwarded to **slurmd** to initiate job steps.

```
> openssl genrsa -out /path/to/private/key 1024
> openssl rsa -in /path/to/private/key -pubout -out /path/to/public/key
```

6.5.10 SLURM Cluster Administration Examples

SCONTROL may be used to print all system information and modify most of it.

Only a few examples are shown below. Please see the **SCONTROL** man page for full details. The commands and options are all case insensitive.

- Print detailed state of all jobs in the system.

```
adev0: scontrol
scontrol: show job
JobId=475 UserId=bob(6885) Name=sleep JobState=COMPLETED
  Priority=4294901286 Partition=batch BatchFlag=0
  AllocNode:Sid=adevi:21432 TimeLimit=UNLIMITED
  StartTime=03/19-12:53:41 EndTime=03/19-12:53:59
  NodeList=adev8 NodeListIndecies=-1
```

```
  ReqProcs=0 MinNodes=0 Shared=0 Contiguous=0
  MinProcs=0 MinMemory=0 Features=(null) MinTmpDisk=0
  ReqNodeList=(null) ReqNodeListIndecies=-1

JobId=476 UserId=bob(6885) Name=sleep JobState=RUNNING
  Priority=4294901285 Partition=batch BatchFlag=0
  AllocNode:Sid=adevi:21432 TimeLimit=UNLIMITED
  StartTime=03/19-12:54:01 EndTime=NONE
  NodeList=adev8 NodeListIndecies=8,8,-1
  ReqProcs=0 MinNodes=0 Shared=0 Contiguous=0
  MinProcs=0 MinMemory=0 Features=(null) MinTmpDisk=0
  ReqNodeList=(null) ReqNodeListIndecies=-1
```

- Print the detailed state of job *477* and change its priority to zero. A priority of zero prevents a job from being initiated (it is held in "pending" state).

```
adev0: scontrol
scontrol: show job 477
JobId=477 UserId=bob(6885) Name=sleep JobState=PENDING
  Priority=4294901286 Partition=batch BatchFlag=0
  more data removed....
scontrol: update JobId=477 Priority=0
```

- Print the state of node adev13 and drain it. To drain a node, specify a new state of **DRAIN**, **DRAINED**, or **DRAINING**. **SLURM** will automatically set it to the appropriate value of either **DRAINING** or **DRAINED** depending on whether the node is allocated or not. Return it to service later.

```
adev0: scontrol
scontrol: show node adev13
NodeName=adev13 State=ALLOCATED CPUs=2 RealMemory=3448
TmpDisk=32000
  Weight=16 Partition=debug Features=(null)
scontrol: update NodeName=adev13 State=DRAIN
scontrol: show node adev13
NodeName=adev13 State=DRAINING CPUs=2 RealMemory=3448
TmpDisk=32000
  Weight=16 Partition=debug Features=(null)
scontrol: quit
Later
adev0: scontrol
scontrol: show node adev13
NodeName=adev13 State=DRAINED CPUs=2 RealMemory=3448
TmpDisk=32000
  Weight=16 Partition=debug Features=(null)
scontrol: update NodeName=adev13 State=IDLE
```

- Reconfigure all SLURM daemons on all nodes. This should be done after changing the SLURM configuration file.

```
adev0: scontrol reconfig
```

- Print the current SLURM configuration. This also reports if the primary and secondary controllers (slurmctld daemons) are responding. To just see the state of the controllers, use the command ping.

```
adev0: scontrol show config
Configuration data as of 03/19-13:04:12
AuthType     = auth/munge
BackupAddr    = eadevj
```

```
BackupController = adevj
ControlAddr    = eadevi
ControlMachine  = adevi
Epilog      = (null)
FastSchedule   = 1
FirstJobId    = 1
InactiveLimit   = 0
JobCompLoc    = /var/tmp/jette/slurm.job.log
JobCompType    = jobcomp/filetxt
JobCredPrivateKey = /etc/slurm/slurm.key
JobCredPublicKey = /etc/slurm/slurm.cert
KillWait     = 30
MaxJobCnt     = 2000
MinJobAge     = 300
PluginDir     = /usr/lib/slurm
Prolog      = (null)
ReturnToService  = 1
SchedulerAuth   = (null)
SchedulerPort   = 65534
SchedulerType   = sched/backfill
SlurmUser     = slurm(97)
SlurmctldDebug  = 4
SlurmctldLogFile = /tmp/slurmctld.log
SlurmctldPidFile = /tmp/slurmctld.pid
SlurmctldPort   = 7002
SlurmctldTimeout = 300
SlurmdDebug    = 65534
SlurmdLogFile   = /tmp/slurmd.log
SlurmdPidFile   = /tmp/slurmd.pid
SlurmdPort    = 7003
SlurmdSpoolDir  = /tmp/slurmd
SlurmdTimeout   = 300
TreeWidth     = 50
JobAcctLogFile  = /tmp/jobacct.log
JobAcctFrequncy  = 5
JobAcctType    = jobacct/linux
SLURM_CONFIG_FILE = /etc/slurm/slurm.conf
StateSaveLocation = /usr/local/tmp/slurm/adev
SwitchType    = switch/elan
TmpFS       = /tmp
WaitTime     = 0

Slurmctld(primary/backup) at adevi/adevj are UP/UP
```

- Shutdown all SLURM daemons on all nodes.

```
adev0: scontrol shutdown
```

Chapter 7. Batch Management (TORQUE)

This chapter explains the benefits for cluster resources when the execution of the programs is optimized. It describes **TORQUE**, which allocates resources and manages queues on a HPC Cluster with one or several nodes.

The following topics are described:

- *7.1 TORQUE Features*
- *7.2 TORQUE Architecture*
- *7.3 Configuration Files*
- *7.4 Configuring Passwordless Access for TORQUE*
- *7.5 Configuring TORQUE*
- *7.6 Configuring TORQUE and RMS*
- *7.7 Common Commands*

7.1 TORQUE Features

TORQUE is a resource manager providing control over batch jobs and distributed compute nodes. **TORQUE** uses a queue mechanism for job execution, which works according to preconfigured priority criteria.

A job is a shell script that includes one or more parallel applications to be processed. Prologue and epilogue mechanisms are attached.

The standard output files for batch processing (**stderr**, **stdout**) may be spooled.

With the user command interface and an API library, TORQUE allows you to:

- Submit a job.
- Display the state and characteristics of a job.
- Cancel a job.
- Change the characteristics of a waiting or running job (Note that for a running job, only the limits and the output files can be changed).
- Stop or resume a job.
- Manage more than 5000 active or waiting jobs.

In addition, when the cluster (or a part of the cluster) is used for time sharing, a mechanism interfaces with the batch manager to verify whether or not a job can be executed. The batch manager allocates jobs to the nodes according to different criterion, such as the following ones:

- Computing power of the nodes
- How the memory is used on each node
- Number of running jobs on the nodes
- Available space in the local temporary directory of the nodes
- Available space in the swap
- Job dates (submission, start, completion).

The batch manager lets the administrator define thresholds for each resource. If a node resource exceeds a threshold, no new job will be assigned to this node. Furthermore, the jobs running on this node will be suspended, re-started or stopped.

For more information about TORQUE refer to the following Web site:
http://www.clusterresources.com/products/torque/

7.2 TORQUE Architecture

The following figure shows the architecture of the TORQUE resource manager, in a cluster consisting of a Management Node (the server) and 3 compute nodes. It demonstrates how the different components interact to determine the reservation of resources.

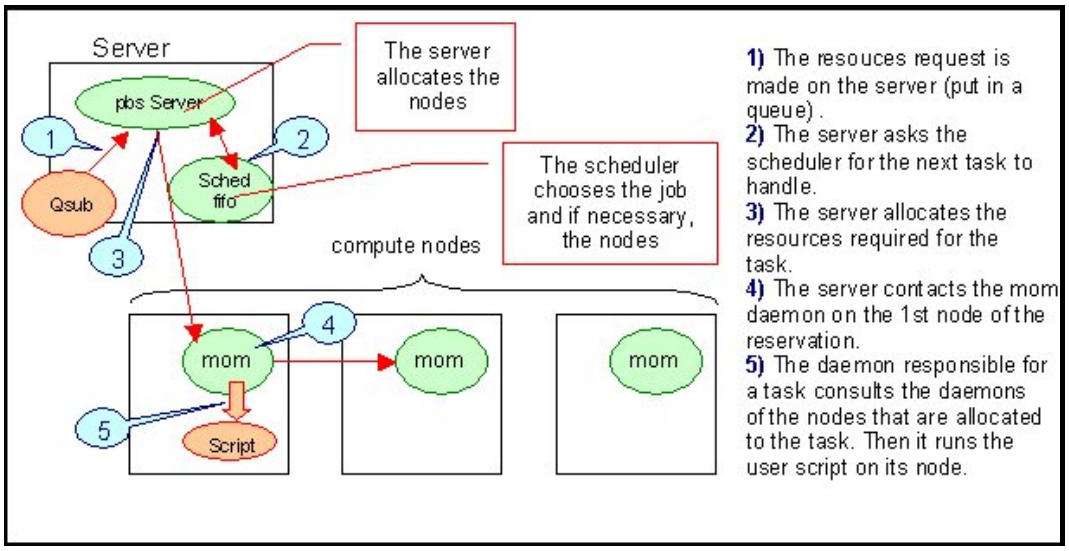


Figure 7-1. TORQUE Resource Manager Architecture

TORQUE includes the following 3 components:

- The TORQUE server, which is responsible for the management of the resources and of queues. It provides the basic resource manager services, such as allocate/de-allocate resources, suspend/resume tasks.
- The **pbs_mom** daemons, which are responsible for resources monitoring, and for managing the execution of user batch scripts. These run on all the compute nodes.
- The scheduler, this is the intelligent part of the resource manager. It selects which tasks will be executed among the waiting tasks. It can also choose the resources (compute nodes) for the task.

A set of user commands allows a user to submit a job, view the job status, delete jobs, etc. Some commands offer administration services, such as adding or removing nodes, or configuring the server.

7.3 Configuration Files

The configuration files are listed below. All of them are in the **/var/torque** directory.

mom-priv/config	configuration of the MOM
server-priv/nodes	resources declaration
server_name	name of the server

The contents of these files must be set according to the configuration, as described in the following sections.

7.4 Configuring Passwordless Access for TORQUE

ssh keys have to be configured to create public/private keys for an ordinary user of a cluster so that passwordless access is enabled for the whole of the cluster/partition on which the application and **TORQUE** is running. Otherwise **TORQUE** will not work correctly.

This is done by using the **ssh-keygen** command, as shown below:

```
ssh-keygen -trsa
```

Append this key to the list of authorized keys.

Note:
See Chapters 2 and 10 in this manual for more information on configuring **ssh**

7.5 Configuring TORQUE

In cluster mode, the Management Nodes (may be one or more for a HA configuration) and the compute nodes are not configured in the same way.

7.5.1 On the Compute Nodes

- Set the configuration files as follows:

mom_priv/config $clienthost <name of the machine where the TORQUE server runs>

server_name <name of the machine where the TORQUE server runs>

- Restart the pbs_mom service as follows:

```
service pbs_mom restart
```

or:

```
/etc/init.d/pbs_mom restart
```

7.5.2 On the Server

- Set the configuration files as follows:

server_name <name of the machine where the TORQUE server runs>

server_priv/nodes List all the compute nodes in the following format:

```
nodename1 np=X [property …]
nodename2 np=X [property …]
nodename3 np=X [property …]
...
```

X is the number of CPUs for the node, and property is a string defining a property of the node. Any string can be used; for example **bigdisk**, **bigmemory**, **smalljob**, and so on.

Note:

The **nodes** file contains the description of the compute nodes managed by TORQUE. This information must be coherent with the RMS configuration.

Restart the **pbs_server** service as follows:

```
service pbs_server restart
```

or:

```
/etc/init.d/pbs_server restart
```

If any jobs are running, then it is recommended to restart the server as follows:

```
qterm -t quick
service pbs_server start
```

or:

```
qterm -t quick
/etc/init.d/pbs_server start
```

- Check that the system is operational by running the following command:

```
pbsnodes -a
```

This command displays the state of the nodes managed by **TORQUE**.

Note:

Immediately after the TORQUE server is restarted (**pbs_server** service), it is possible that some nodes will appear in the Unknown state. This is normal as it may take 2 or 3 minutes for the nodes to inform the server of their respective states (using the **pbs_mom** service).

7.6 Configuring TORQUE and RMS

7.6.1 Architecture

The **TORQUE** batch manager and the **RMS** resource manager have similar functions, but each one offers some different features. For example **TORQUE** provides an efficient mechanism to manage several queues of jobs and a high-performance scheduler for these jobs and queues. On the other hand, **RMS** provides a very fine accounting mechanism and is able to configure the Quadrics network for the best performance.

When **TORQUE** and **RMS** work together, the configuration of **RMS** must be coherent with the configuration of **TORQUE**.

When **TORQUE** and **RMS** are used together, **TORQUE** will manage the job queue and scheduling whilst **RMS** analyzes the **PBS_NODEFILES** file and then executes the programs on the nodes allocated by **TORQUE**.

The following figure shows how the resources are allocated in a coherent way by **TORQUE** (on the server) and **RMS** (using the **pbs_mom** daemon which is responsible for this task).

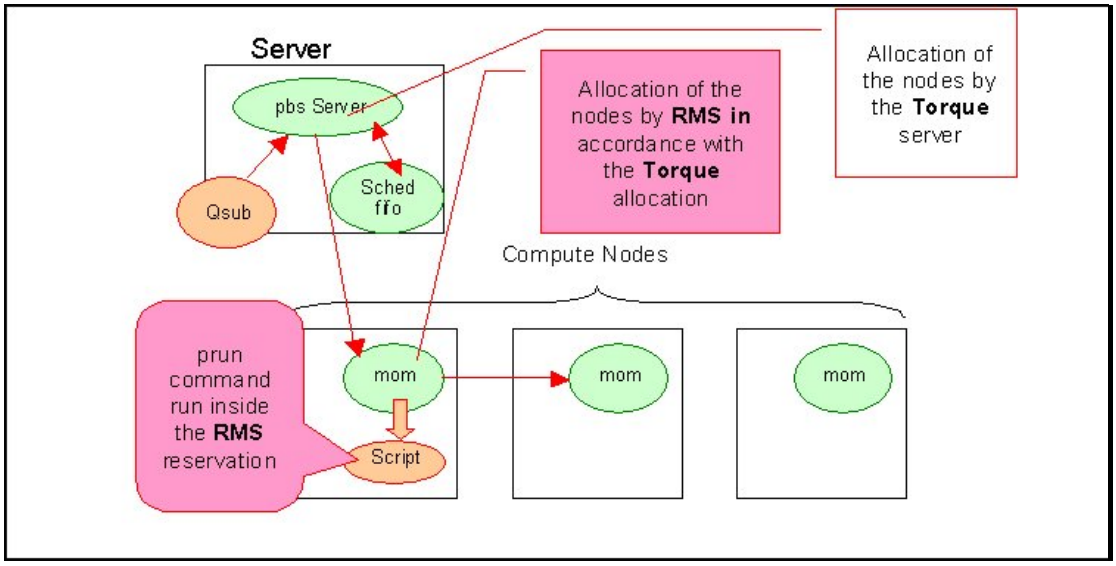


Figure 7-2. TORQUE resource allocation

The next section describes the configuration tasks that the administrator must perform after the installation of the package to make the **TORQUE** / **RMS** system operational.

7.6.2 Configuration

When **TORQUE** and **RMS** are used together, the **RMS** configuration must be coherent with the **TORQUE** configuration; this means:

- The node names must be identical.
- Configure TORQUE as described in the section 7.5
- All the compute nodes managed by **TORQUE** are also managed by **RMS**.

- The root user must be able to run the following command from all the compute nodes and apply it to all the compute nodes, without having to enter a password.

```
prun –t -r –H node_list uname
```

For detailed information about the RMS configuration, refer to www.quadrics.com and see the *RMS User's Guide* and the *RMS Reference Manual* on the Documentation page.

7.7 Common Commands

This section lists the most common commands. Since **TORQUE** and **RMS** work together in a transparent way, the user runs only **TORQUE** submission commands. The **prun** program is launched inside the **TORQUE** node reservation policy.

7.7.1 TORQUE Commands for Job Management

- **qsub** is used to submit a job. You must be logged on as a ordinary user; root is not allowed to submit a job.
  Examples:
  - To submit **myscript.sh** on 4 nodes, enter this command on the Management Node:

```
qsub -l nodes=4 myscript.sh
```

    The job is queued, and then started when the resource is available (4 nodes).
  - To submit **myscript.sh** on 4 nodes with 2 processors on each node, enter this command on the Management Node:

```
qsub -l nodes=4 :ppn=2 myscript.sh
```

    The job is queued, and then started when the resource is available (8 CPUs).
- **qdel** to delete a job.
- **qhold / qrls** to hold / release a job waiting in a queue.

7.7.2 Commands to start a Program inside a Job

- **prun** to start a parallel program.

7.7.3 Commands to Display Status

- **qstat** (**TORQUE** command) to display the status of queues and jobs.
- **pbsnodes** (**TORQUE** command) to display the status of nodes and jobs running on the nodes.
- **rinfo** (**RMS** command) to display the status of queues, nodes and jobs.
- **rmsquery** (**RMS** command) to submit **SQL** queries. This should be used carefully.

7.7.4 Administrator's Commands and Configuration Files

The administrator is in charge of the coherence of the **TORQUE** and **RMS** configurations. The following commands can be used:

- **rcontrol** (**RMS** command) to control the use of the system resources.
- **qmgr** (**TORQUE** command) to interface with the batch system.

Chapter 8. Monitoring with NovaScale Master - HPC Edition

NovaScale Master - HPC Edition provides the monitoring functions for Bull HPC systems. It relies on Nagios and Ganglia open source software. Nagios is used to monitor the operating status for the different components of the cluster. Ganglia collects performance statistics for each cluster node and displays them graphically on a cluster scale. The status of a large number of elements can be displayed.

This chapter covers the following topics:

8.1 Launching NovaScale Master - HPC Edition

Note:

The cluster database (**ClusterDB**) must be running before starting monitoring. See the *Cluster Data Base Management* Chapter.

1. If necessary restart the **gmond** and **gmetad** services:

```
service gmond restart
service gmetad restart
```

2. Start the monitoring service:

```
service nagios start
```

3. Start Mozilla and go to the following URL:

   http://<ManagementNode>/NSMaster/

Note:

Mozilla is the mandatory navigator for **NovaScale Master – HPC Edition**

8.2 Access Rights

8.2.1 Administrator Access Rights

By default the Administrator will use the following login and password:

login: **nagios**
password: **nagios**

The graphical interface for monitoring opens, see Figure 8-1.
The Administrator will be able to enter host and service commands via the interface, whereas an ordinary user will only be able to consult the interface.

8.2.2 Standard User Access Rights

By default the ordinary user will use the following login and password:

login: **guest**
password: **guest**

8.2.3 Adding Users and Changing Passwords

The **htpasswd** command is used to create new user names and passwords.

To create additional users for the graphical interface, do as follows:

1. Enter the following command:

```
htpasswd /etc/nagios/htpasswd.users <login>
```

   This command will prompt you for a password for each new user, and will then ask you to confirm the password.

2. You must also:
   a. Define the user in the **/opt/NSMaster/core/share/etc/rbm/missions.xml** file (either as an Administrator or as an ordinary user).
   b. Add the password to the **/opt/NSMaster/core/etc/htpasswd.users** file.

To change the password for an existing user, do as follows:

1. Enter the following command:

```
htpasswd /etc/nagios/htpasswd.users <login>
```

   Enter and confirm the new password when prompted.

2. Change the password listed for the user in the **/opt/NSMaster/core/etc/htpasswd.users** file.

Note: Some of these steps have to be done as the root user.

See the NS Master documentation for more information on adding users and on account management.

8.3 Hosts, Services and Contacts for Nagios

Nagios defines two entities: **hosts** and **services.**

A **host** is any physical server, workstation, device etc. that resides on a network.

A **host group** definition is used to group one or more hosts together for display purposes in the CGIs.

A **service** definition is used to identify a "service" that runs on a host. The term "service" is used very loosely. It can mean an actual service that runs on the host (POP, SMTP, HTTP, etc.) or some other type of metric associated with the host (response to a ping, number of logged in users, free disk space, etc.).

Note:
NovaScale Master – HPC Edition will display the services specific to each host when the host is selected within the different parts of the NovaScale Master – HPC Edition interface.

A **contact** definition is used to identify someone who should be contacted in the event of a problem on your network.

A **contact group** definition is used to group one or more contacts together for the purpose of sending out alert/recovery notifications. When a host or service has a problem or recovers, Nagios will find the appropriate contact groups to send notifications to, and notify all contacts in these contact groups. This may sound complex, but for most people it doesn't have to be. It does, however, allow for flexibility in determining who gets notified for particular events.

For more information on these definitions and the arguments and the directives which may be used to format the definitions see:
http://nagios.sourceforge.net/docs/2_0/

Alternatively, select the **Documentation** link on the Nova Scale Master opening screen or by selecting the Documentation button in the title bar.

8.4 Using NovaScale Master - HPC Edition

The graphical interface of NovaScale Master - HPC Edition is shown inside a Web browser.

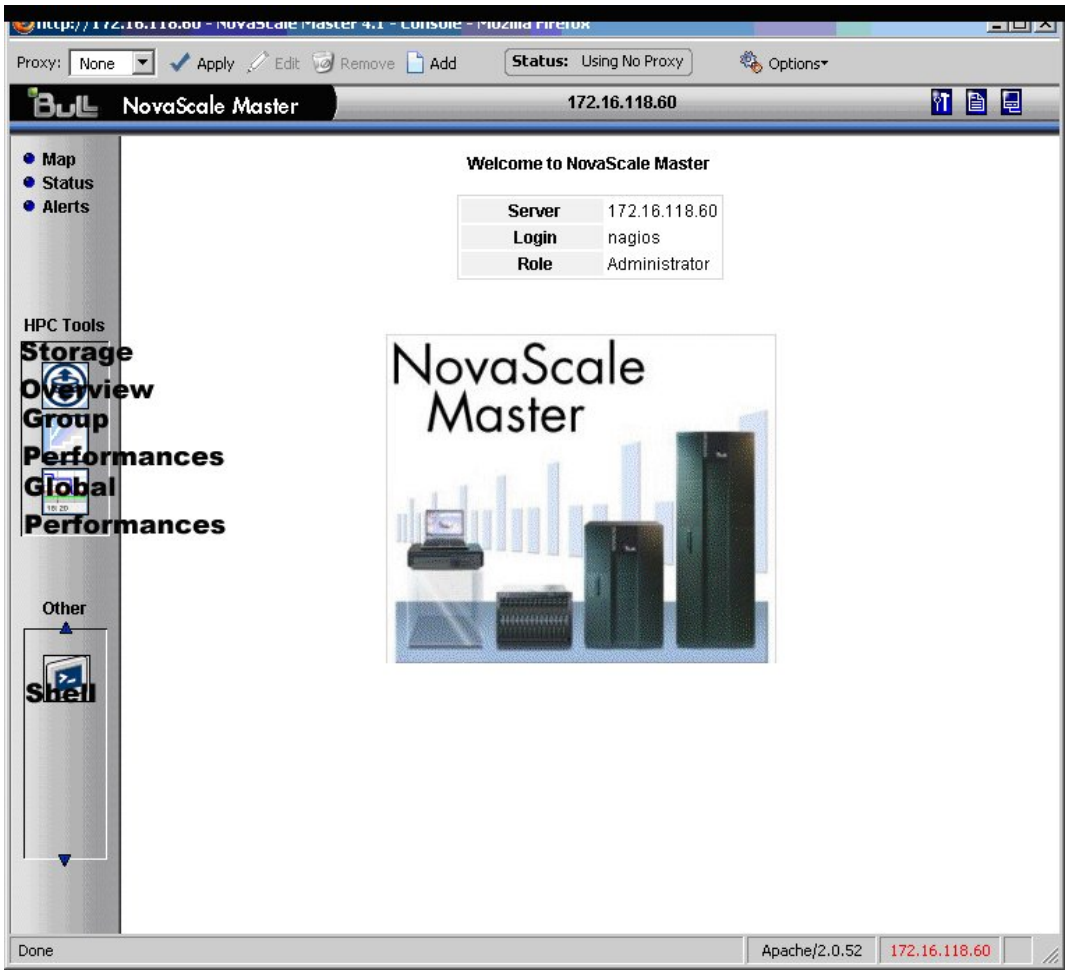


Figure 8-1. NovaScale Master - HPC Edition opening view

8.4.1 NovaScale Master - HPC Edition – View Levels

Initially the console will open and the administrator can then choose to view different types of monitoring information with a range of granularity levels either by clicking on the icons in the left hand vertical tool bar and then by clicking on the links in the different windows displayed. Using the links it is possible to descend to a deeper level for more detailed information for a particular host or service. For example, the Cabinet Rack map view in Figure 8-2 leads to the Rack View in Figure 8-3, which in turns leads to a more detailed Services view for the host selected in Figure 8-4.

8.5 Map Button

The Map Button is displayed at the top right hand side of the opening, when it is selected the drop down menu provides two options inside the main window. The map container can either be animated by **all status** or by **ping** views.

8.5.1 All Status Map View

The **all status** map view presents a chart of the cluster representing the various server rack cabinets in the room.

The color of each cabinet is determined by the component within it which is in the worst state.

By default, in addition to the view of the rack cabinets in the room, the Monitoring Problems window will appear at the bottom of the screen with a status for all hosts and services and the Availability Indicators view window will appear on the top right hand side – see Figure 8-2.

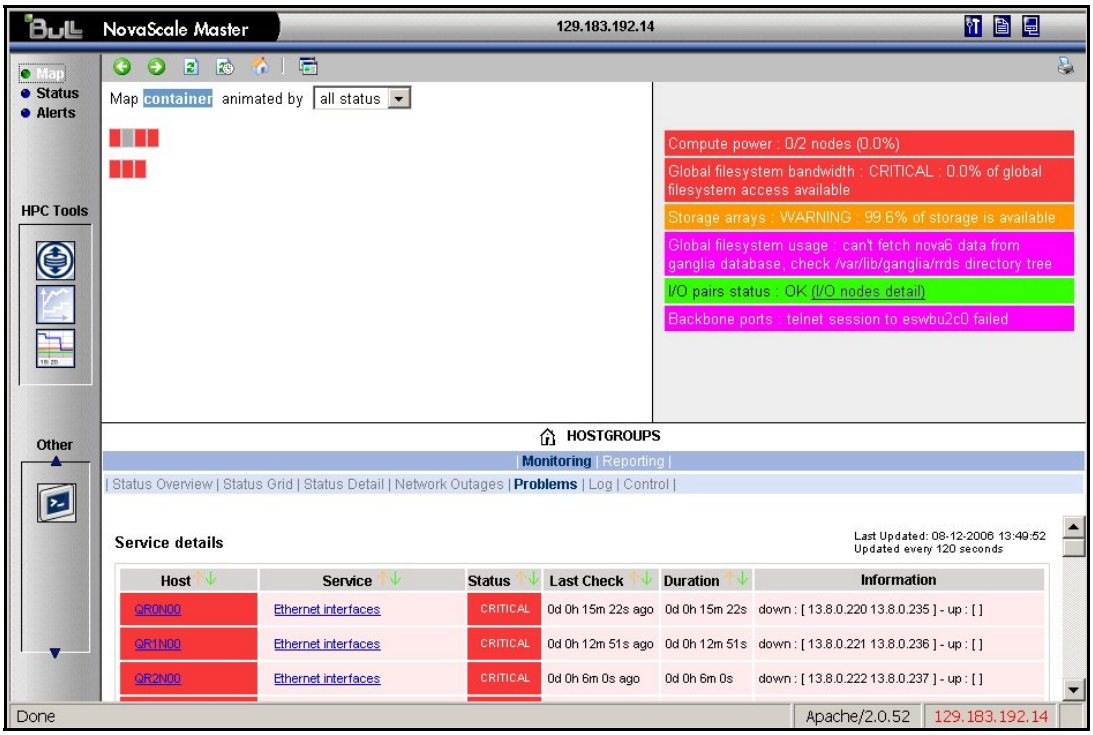


Figure 8-2. Map button all status opening view

When the cursor passes over a rack, information (its label, its type and the elements contained in the rack) about the rack is displayed. When the user clicks on a cabinet, a detailed view of the cabinet is displayed – see Rack view in Figure 8-3. This displays additional information, including its physical position and the services which are in a non-OK state.

8.5.2 Rack View

The Rack view details the contents of the rack: the nodes, their position inside the rack, their state, with links to its Alert history, etc. The list of the problems encountered is displayed at the bottom of the view – see Figure 8-3.

Clicking on a component displays a detailed view for it.

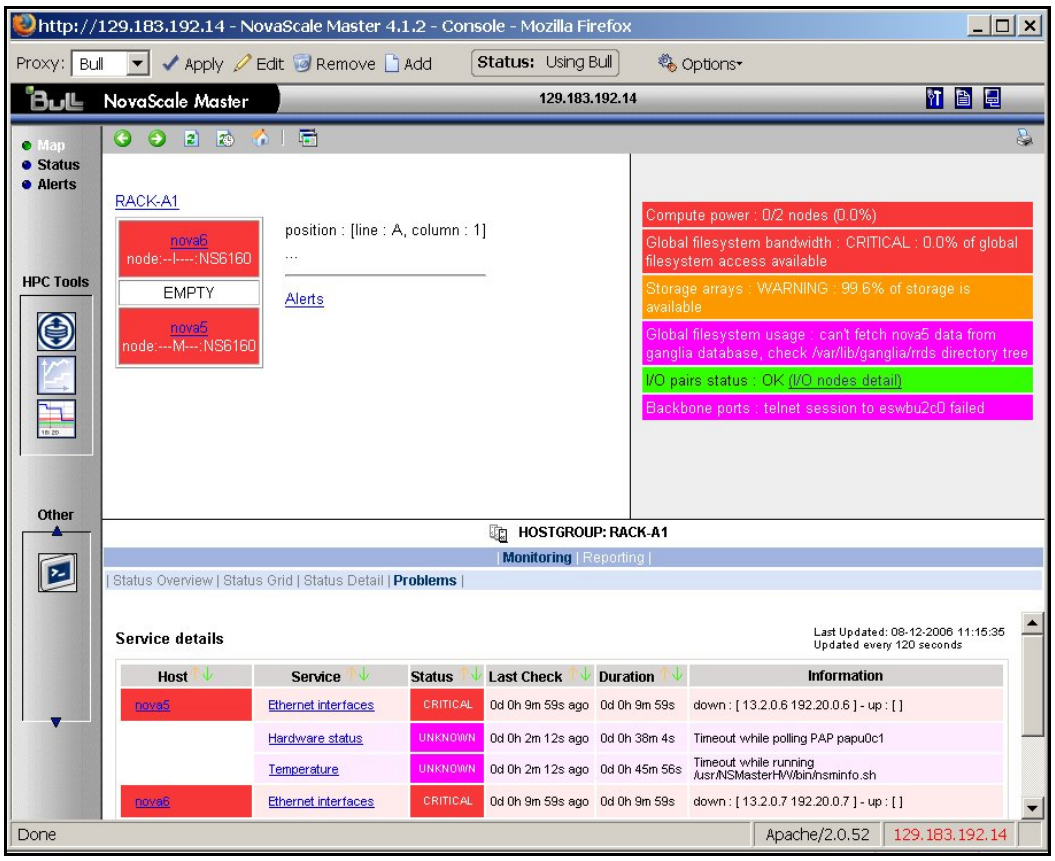


Figure 8-3. Rack view with the problems window at the bottom

More detailed information regarding the hardware components and services associated with a host appear, when the host in the rack view is clicked on, in the top right pane of the Rack view. This leads to another pop up window which includes further information for the host and its services – see Figure 8-4.

8.5.3 Host Services detailed View

Clicking the **Host Status** or a **Service Status** link in this window displays more specific information for the component or service.

The control button in the middle of screen will provide the details for the Service monitoring information and the Service commands for the hardware component.

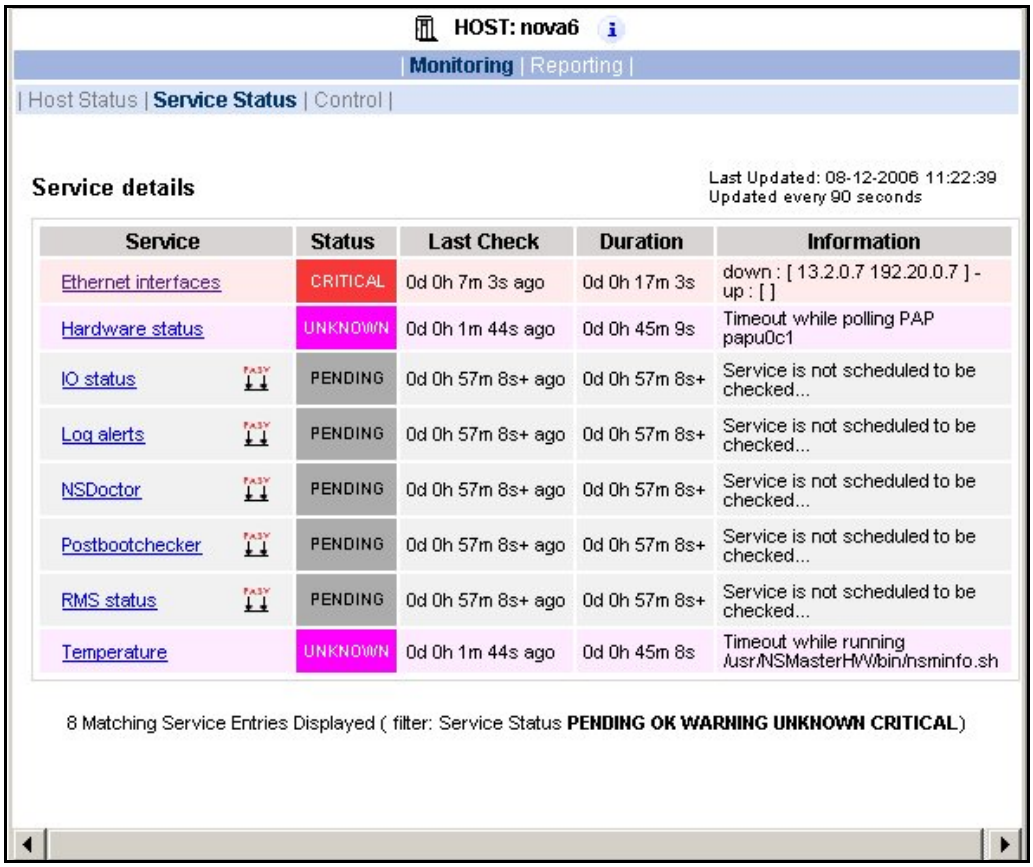


Figure 8-4. Host Service details

By clicking on the links in the windows even more detailed information is provided for the services.

8.5.4 Ping Map View

The **ping** map view is similar to the **all status** map view except that it only shows the state of the pings sent to the different components in the cabinets. The state of the services associated with the nodes is not taken into account.

By default the Monitoring Problems window will appear at the bottom of the screen.

8.6 Status Button

By clicking on the Status button a screen appears which lists all the hosts and the services running on each one of them as shown in Figure 8-5. More detailed information may be seen for each Host Group by selecting either the individual Host Group or by selecting the links in the Host Status Totals or Service Status Totals columns.

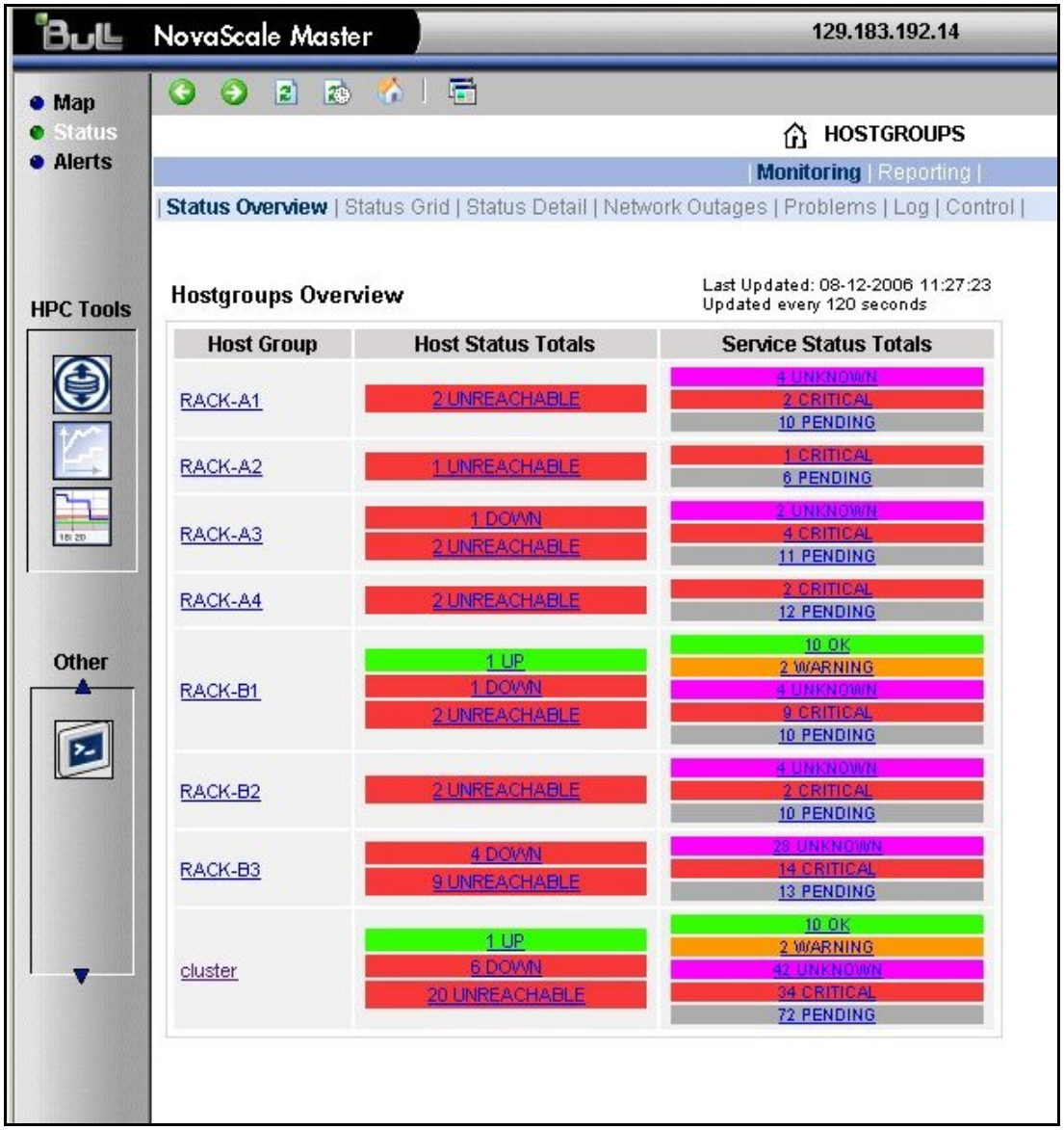


Figure 8-5. Status overview screen

8.7 Alerts Button

The **Nova Scale Master Alert Viewer** application displays monitoring alerts (also called events) concerning a set of **hostgroups**, **hosts** and **services**. The application provides filter functions in order to display alerts on all monitored resources or on only a subset of these resources.

Alerts are visible following the selection of the Alert Button followed by the Reporting link tab, and then by the Alert Viewer – see Figure 8-6.

Whenever a service or host status change takes place, the monitoring server generates an alert, even when status passes from **CRITICAL** to **RECOVERY** and then to **OK**. Alerts are stored in the current monitoring log and are then archived.

The NovaScale Master - HPC Edition Alert Viewer application scans the current monitoring log and archives according to filter **report period** settings.

Alert Types

The alerts can be filtered according to the following alert types:

- Hosts and Services
- Hosts
- Services.

Note: By default, Hosts and Services is selected.

Alert Level

Alerts can be filtered according to the following alert levels:

- **All** – Displays all alerts.
- **Major and Minor problems** - Displays host alerts with DOWN or UNREACHABLE status levels or displays service alerts with WARNING, UNKNOWN or CRITICAL status levels.
- **Major problems** -Displays host alerts with DOWN or UNREACHABLE status levels displays service alerts with UNKNOWN or CRITICAL status levels.
- **Current problems** -Display alerts with a current non-OK status level. When this alert level is selected, the Time Period is automatically set to 'This Year' and cannot be modified.

Note: By default, **All** is selected.

Report Period
This setting can be changed using the drop down menu provided.

8.7.1 Active Checks

Active monitoring consists in running at regular intervals a plug-in program which will carry out checks and send the results back to Nagios. The plug-in will send various codes which correspond to the Alert alarm level.

These are 0 for OK/UP (Green background), 1 for WARNING (Orange background), 2 for CRITICAL/DOWN (Red background), 3 for UNKNOWN (Violet background). The plug-in will also display an explanatory text for the alarm level.

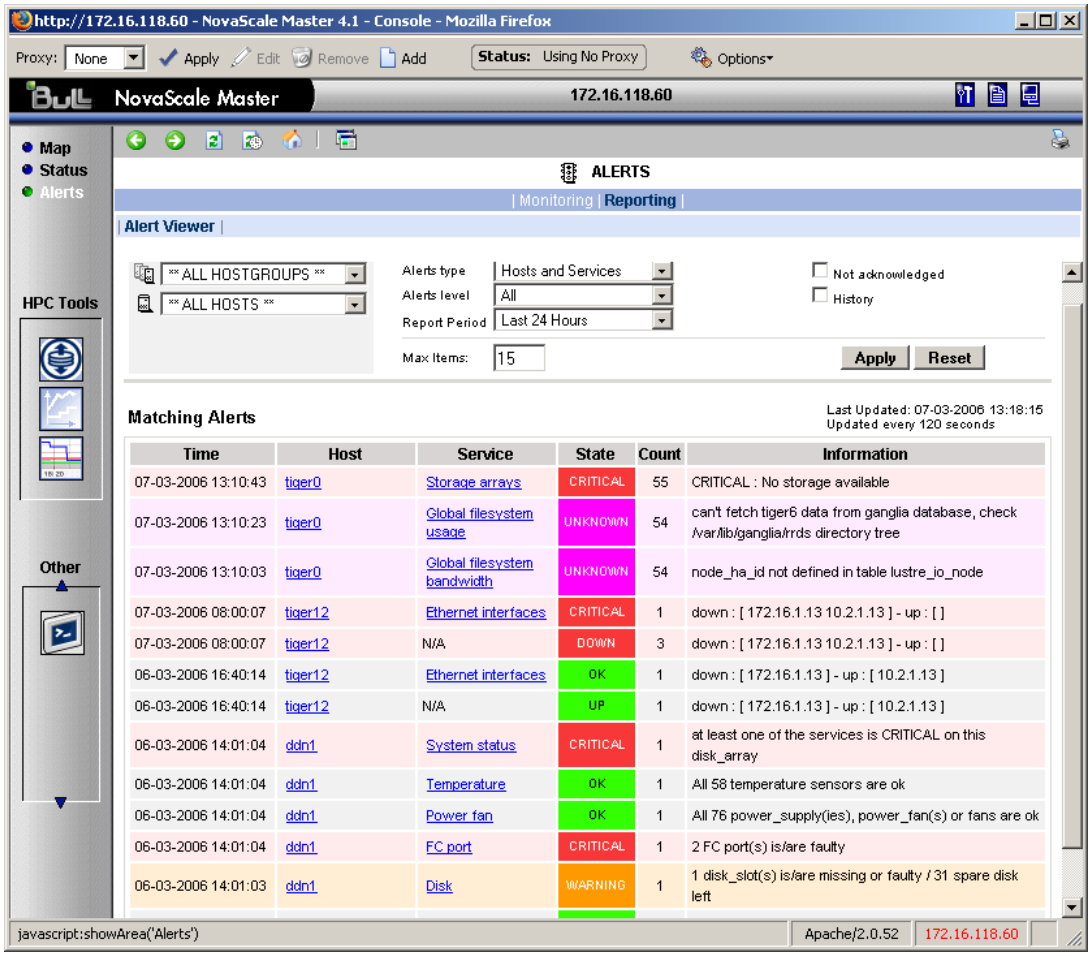


Figure 8-6. Alert Window showing the different alert states

8.7.2 Passive Checks

With this form of monitoring a separate third-party program or plug-in will keep Nagios informed via its external command file **(/var/spool/nagios/nagios.cmd)**. It submits the result in the form of a character string which includes a timestamp, the name of the host and service concerned as well as the return code and the explanatory text.

Passive checks appear with a grey background in the list of alerts.

8.7.3 Notifications

Notifications are sent out if a change or a problem occurs. The Notification may be one of 3 types- e-mail, SNMP Trap or using a user Script. Host and service notifications occur in the following instances:

When a hard state change occurs

When a host or service remains in a hard non-OK state and the time specified by the **<notification_interval>** option in the host or service definition has passed since the last notification was sent out (for that specified host or service). In order to prevent recurring notifications, set the **<notification_interval>** value to 0 - this stops notifications from being sent out more than once for any given problem.

The Monitoring Control window see Figure 8-8 provides the facility to enable or disable notifications.

The Notification level is set in the Maps ➞Hostgroups ➞ Reporting ➞Notifications window. The different notification levels are as indicated in Figure 8-7.

Figure 8-7. Hostgroups Reporting Notifications Window showing the Notification Levels

8.7.4 Acknowledgments

As the **Administrator**, you may acknowledge alerts and decide whether they should be displayed or not.

8.7.5 Comments

Users of a particular host can post comments using the Monitoring Control window as shown in Figure 8-8.

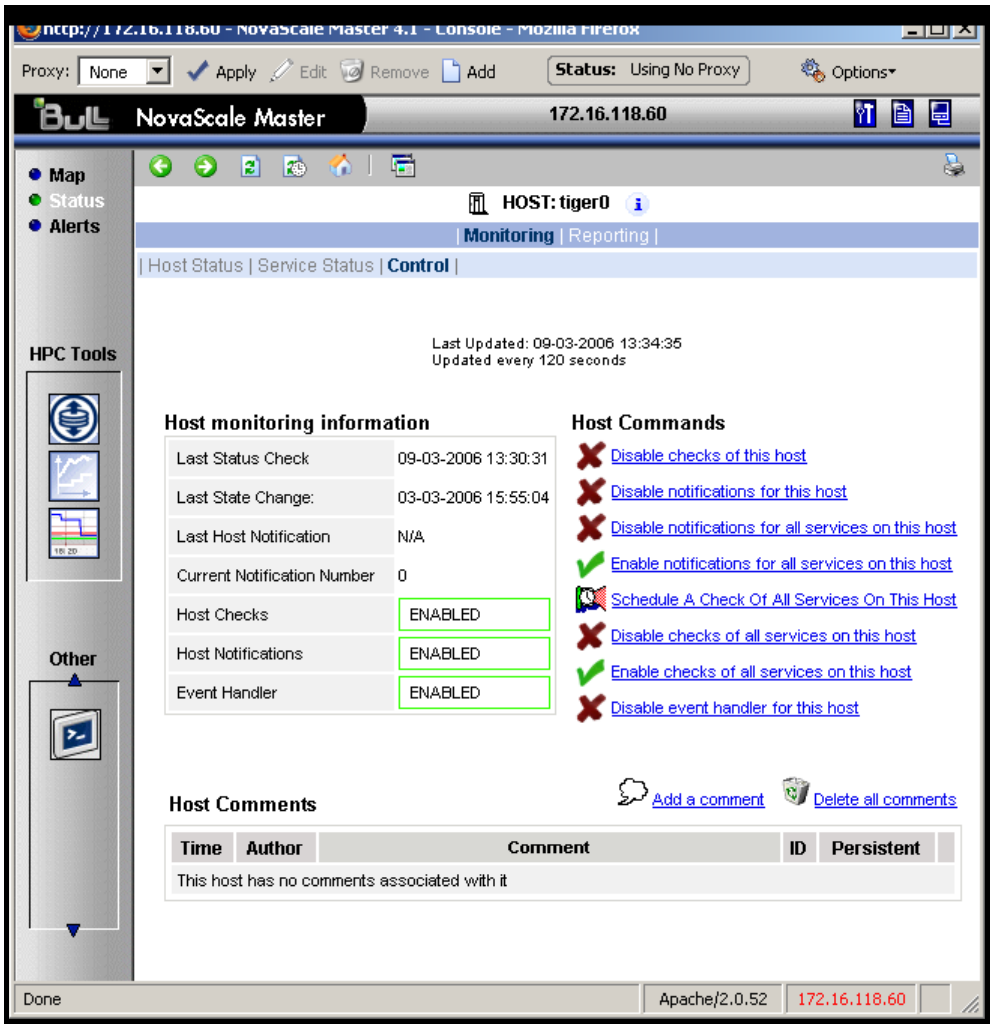


Figure 8-8. Status Monitoring Control window showing the links to add and delete comments

8.7.6 Logs

The current Nagios log file is **/var/log/nagios/nagios.log**. The log archives for the preceding weeks is saved **/var/log/nagios/archives**. The Service Log Alert window may be displayed by selecting it in the Service Status window as shown in Figure 8-9.

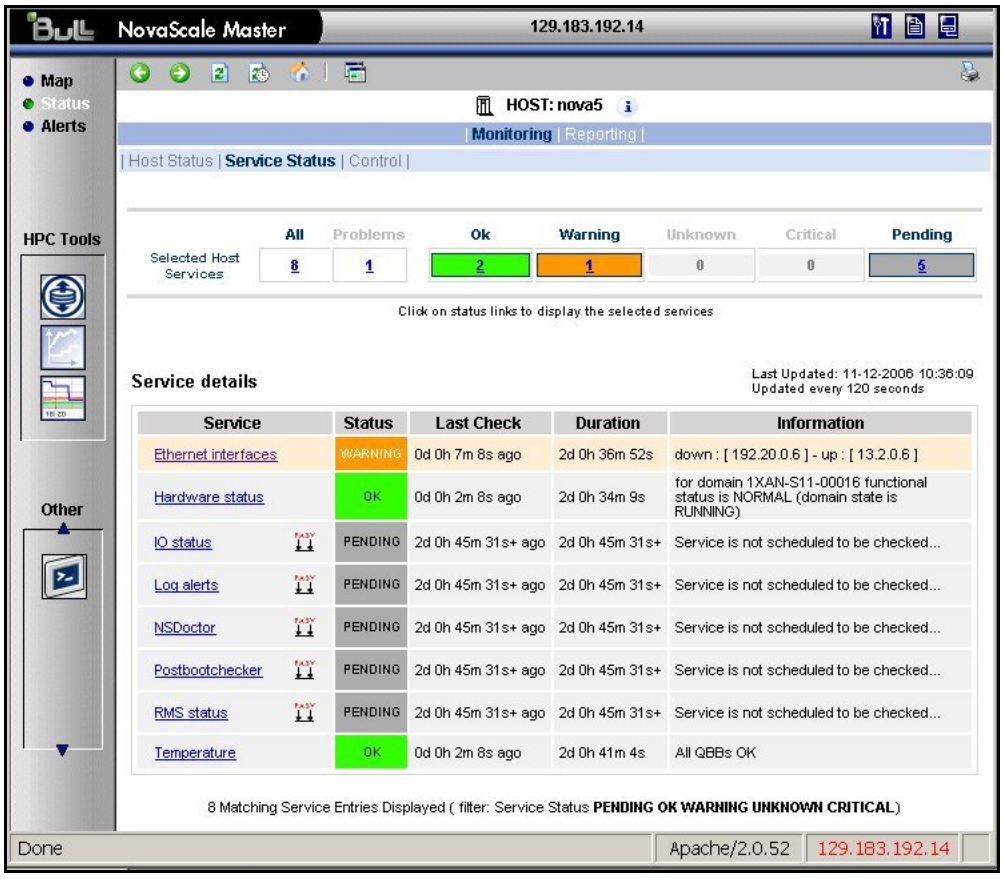


Figure 8-9. Monitoring Service Status window for a host. More details for the Log alerts are available by selecting the Log alerts link in the middle of the screen.

8.7.7 Alert Definition

The different parameters which may be used for an alert are as follows:

$HOSTNAME$: The name of the host from which the alert is returned.

$HOSTALIAS$: The content of the comma separated field ':'

For a node this is: **node:<type>:<model>**
with **<type>** = for example A—, -C—, AC-M-
with **<model>** = for example NS6160.

For a PAP this is: **pap:<type>**
with **<type>** = master, standard.

For an Ethernet switch: **eth_switch:<model>**

with **<model>** = for example. CISCO 3750G24TS.

For an interconnect switch : **ic_switch:<model>**

with **<model>** = for example the type of material (**node, pap**, **eth_switch, ic_switch**).

8.7.8 Running a Script

NovaScale Master - HPC Edition can be configured to run a script when a state changes or an alert occurs.

Below is an example of script, which is run when **RMS** sends a **configure-out** event on a node.

```
#!/usr/bin/perl -w

# Arguments : $SERVICESTATE$ $STATETYPE$ $HOSTNAME$ $HOSTSTATE$ $OUTPUT$

$service_state = shift;
$state_type = shift;
$host_name = shift;
$host_state = shift;
$output = join(" ", @ARGV);

# Sanity checks
if ($state_type !~ "HARD") { exit 0; }
if ($service_state !~ "WARNING" && $service_state !~ "CRITICAL") {
  exit 0;
}

# Launch NSDoctor if needed
if ($host_state =~ "UP" &&
    $output =~ /automatically configured out|no response/) {
  system("/usr/sbin/nsdoctor.pl $host_name");
}
exit 0;
```

User scripts which define events or physical changes to trigger Nagios alerts may also be used.

More information on scripts or third party plugins is available in the documentation from http://www.nagios.org/docs/

In order that e-mail alerts are sent whenever there is a problem, a SMTP server such as sendmail or postfix has to be in running on the Management node.

By default, the e-mail alerts are sent to nagios@localhost on the Management Node.

Note:
All the mails are redirected to the root user automatically using an alias from the **nagios** user to the root user.

Normally, by default, only the cluster administrators will receive the alerts for each change for all hosts and services. To send the alert e-mails to other addresses, it is necessary to create new contacts and to add them to the contact groups. The files to modify are **/etc/nagios/contacts.cfg** and **/etc/nagios/contactgroups.cfg**.

8.7.9 Generating SNMP Alerts

When **NovaScale Master - HPC Edition** receives an alert (service in a WARNING or CRITICAL state, host in DOWN or UNREACHABLE state), the event handler associated with the service or host sends an SNMP trap, using the **snmptrap** command.

The Management Information Base (**MIB**) is available in the file **/usr/share/snmp/mibs/NSMASTERTRAPMIB.txt**. This describes the different types of traps and the information that they contain.

In order that an SNMP trap is sent the following actions should be performed:

1. Add the IP address of the host(s) that will receive the traps in the **/etc/nagios/snmptargets.cfg** file (one address per line).

2. Add the contact that will receive the traps to a contact group. To do this, edit the **/etc/nagios/contactgroups.cfg** file and change the line:

   ```
   members  nagios
   ```
   in:
   ```
   members  nagios,snmptl
   ```

3. Restart nagios:

```
service nagios reload
```

8.7.10 Resetting an Alert Back to OK

To reset an alert back to zero click on the service or the host concerned, then on the menu **Submit passive check result for this service**. Set the **Check Result** to OK, if it is not already the case, fill in the field **Check Output** with a short explanation then click on the button **Commit.** The return to the OK state will be visible after the next command reading by the Nagios demon.

8.7.11 nsmhpc.conf configuration file

The **/etc/nsmhpc/nsmhpc.conf** file contains several configuration parameters. Most of them have default values, but for some services the administrator may have to specify parameter values. A message will inform the administrator if a value is missing.

8.8 Storage Overview

By selecting the Storage overview button in the vertical toolbar on the left hand side a window containing the information similar to that shown below is displayed.

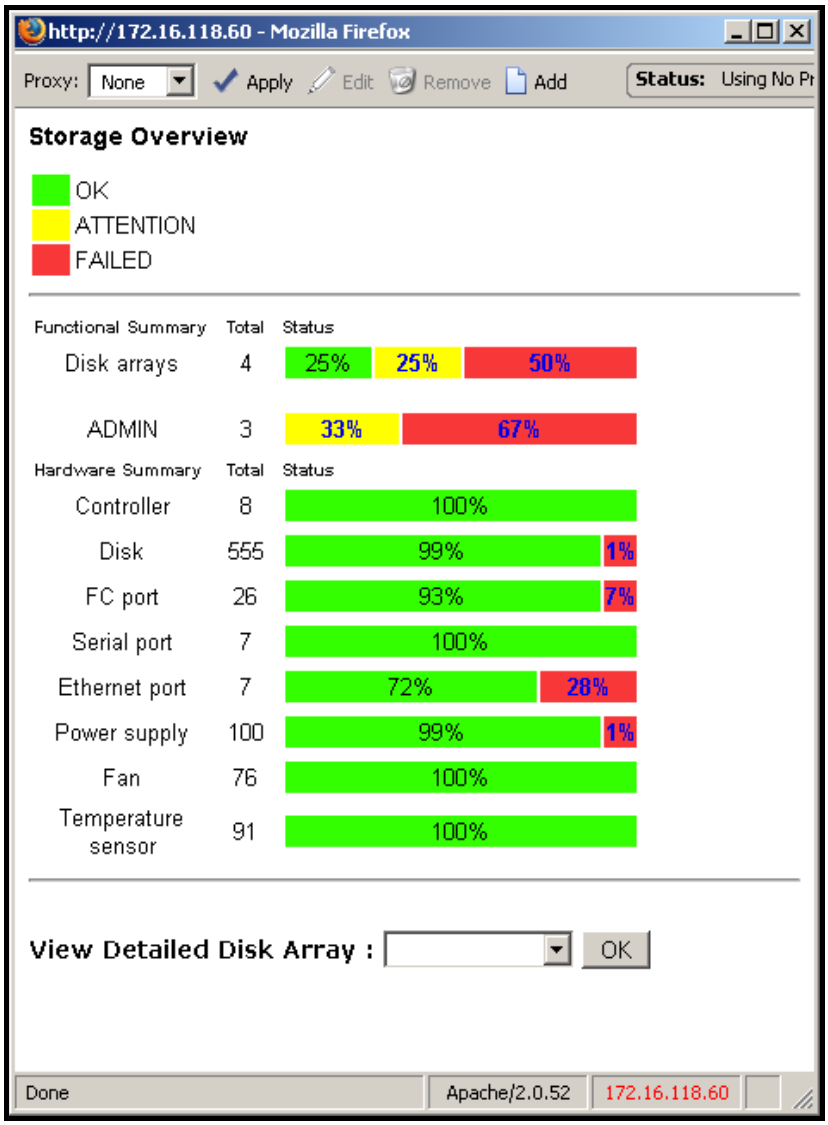


Figure 8-10. Storage overview window

More detailed information is provided by clicking on the ATTENTION and FAILED sections of the component summary status bars.

See *Chapter 9 – Storage Device Management* for information on **NovaScale Master - HPC Edition** and storage views.

8.9 Shell

The Shell button can be used to open a shell and enter commands on the administration node.

8.10 Monitoring the Performance - Ganglia Statistics

NovaScale Master - HPC Edition provides the means to visualize the performance for the cluster by selecting the icons in the vertical left hand tool bar – see Figure 8-1. This can be done either for a Global Performance View, which displays data either for a complete cluster or on a node by node basis, or in a Group Performance View. These views enable the statistical examination of a predefined group of nodes in the database.

The parameters which enable the calculation of the performance of the cluster are collected on all the nodes by Ganglia and are displayed graphically. One can also choose the observation period and display the measurement details for a particular node using the Ganglia interface.

8.11 Group Performance View

This view displays the group performance for 6 different metric types for the complete cluster as shown in Figure 8-11. Using this view it is possible to see view the nodes in groups and then to zoom on a particular node.

Figure 8-11. Group Performance view

8.12 Global Performance View

The global performance view gives access to the native interface for Ganglia and provides an overall view of the cluster. It is also possible to view the performance data for individual nodes.

Five categories of data collected. These are:

- Load for CPUS and running processes
- Memory details
- Processor activity
- Network traffic in both bytes and packets
- Storage.

Each diagram shows changes for the performance metrics over a user defined period of time.

Figure 8-12. Global overview for a host (top screen)

More detailed views are shown by scrolling the window down – see Figure 8-13.

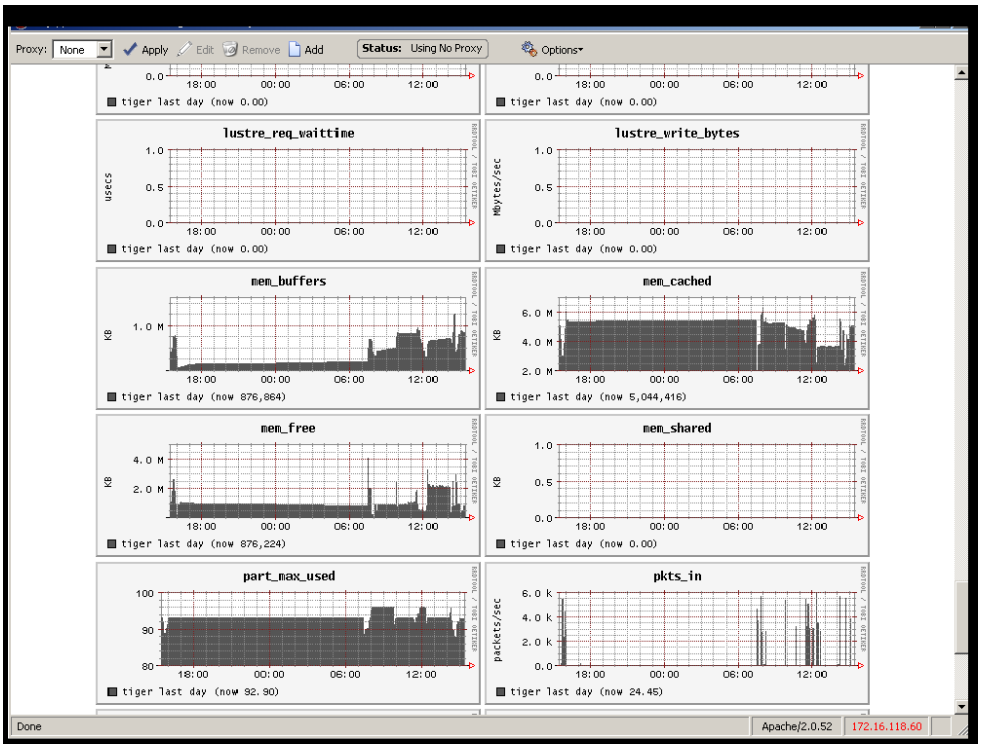


Figure 8-13. Detailed monitoring view for a host (bottom half of screen displayed in Figure 8-12)

8.12.1 Modifying the Performance Graphics Views

The format of the graphs displayed in the performance views can be modified by editing the file **/usr/share/nagios/conf.inc**. The section which follows the line Metrics enumeration defines the different graphs; each graph is created by a call to the producer of the Graph class. To create a new graph, it is necessary to add the line:

```
$myGraph = new Graph("<graphname>")
```

<graphname> is the name given to graph.

To specify a metric to the graph, the following command must be edited as many times as there are metrics to be added or changed:

```
$myGraph->addMetric(new Metric("<metricname>", "<legende>", "<fonction>",
"<couleur>", "<trait>"))
```

<metricname> the name given by Ganglia for the metric.

<legende> text displayed on the graph to describe the metric.

<fonction> aggregating function used to calculate the metric value for a group of nodes, currently the functions **sum** and **avg** are supported.

<couleur> HTML code color.

<trait> style for feature displayed (**LINE1, LINE2, AREA, STACK**), See the man page for **rrdgraph** for more details.

Use the command below to add the graph to those which are displayed:

```
graphs:$graphSet->addGraph($myGraph)
```

8.12.2 Refresh Period for the Performance View Web Pages

By default the refresh period is 90 seconds. This can be modified by changing the value for the parameter **refresh_rate** in the file **/etc/nagios/cgi.cfg**.

8.13 Configuring and Modifying Nagios Services

8.13.1 Configuring Using the Database

The command to be used to regenerate the Nagios services database configuration files is:

```
/usr/sbin/dbmConfig configure --service nagios --restart
```

This command will also restart Nagios after the files have been regenerated.

Use the following command to test the configuration:

```
service nagios configtest
```

8.13.2 Modifying Nagios Services

The list and configuration of Nagios services is generated from the database and from the file **/etc/nagios/services-tpl.cfg**. This file is a template used to generate the complete files.

All template modifications necessitate the Nagios configuration file to be regenerated using the following command:

```
dbmConfig configure --service nagios
```

Note:
To check that all services have been taken into account, you can use the **dbmServices** command (this command is described in the *Cluster Database Management* chapter in the present guide). If it is not the case, enter the following commands:

```
/usr/lib/clustmngt/clusterdb/bin/nagiosConfig.pl --init
dbmConfig configure –-service nagios
```

Refer to http://nagios.sourceforge.net/docs/2_0/checkscheduling.html for more information on configuring the services.

8.13.2.1 Clients without Customer Relationship Management software

If a CRM product is not installed then the Nagios configuration files will have to be changed to prevent the system from being overloaded with error messages. This is done as follows:

1. Edit the **/etc/nagios/contactgroups** file and change the line which reads **members nagios,crmwarn,crmcrit** so that it reads **members nagios**

2. In the **/etc/nagios/nagios.cfg** file change the status of the line **process_performance_data=1** so that it is commented.

8.13.3 Changing the Verification Frequency

Usually the application will require that the frequencies of the Nagios service checks are changed. By default the checks are carried out once every ten minutes, except on certain services. To change this frequency, the **normal_check_interval** parameter has to be added to the body of the definition of the service and then modified accordingly.

8.14 General Nagios Services

Nagios includes a wide range of plug-ins, each of which provides a specific monitoring service which is displayed inside the graphical interface. In addition Bull has developed additional monitoring plug-ins which are included within NovaScale Master – HPC edition. The plug-ins and corresponding monitoring services are listed below.

The services listed in this section apply to all node types. The Ethernet Interfaces service also applies to all forms of material/devices.

8.14.1 Ethernet Interfaces

The Ethernet interfaces service indicates the state of the Ethernet interfaces for a node. The plug-in associated with this service is **check_fping** which runs the **fping** command for all the Ethernet interfaces of the node. If all the interfaces respond to the ping, the service posts OK. If **N** indicates the total number of Ethernet interfaces and at least **1** or at most **N-1** interfaces do not answer, then the service will display **WARNING**.

8.14.2 RMS Status

The **RMS** status of each node is monitored by the **RMS** event handler which communicates it to Nagios in a written form using the FIFO protocol.

In addition, a review of the **RMS** database is carried out regularly by the plug-in **check_node_rms.pl** to check if the status has changed.

8.14.3 Hardware Status

The material status of each node is posted to the passive Hardware status service, corresponding to the **check_node_hw.pl** plug-in which interfaces with the **PAP** associated with the node. If the state is different from OK, the plug-in creates a link making it possible to open the associated PAP window in order that more details are provided concerning the cause of the problem.

This service is configured so as to be checked every five minutes in order not to overload the **PAP**.

8.14.4 Temperature

The temperature service checks the temperature of the node. Currently, only the temperatures of four QBB boards are monitored by means of the **nsminfo** command. This service returns an alarm equivalent to the level of the worst **QBB** board state and posts for each state the number or boards which are in this state. The different possible states are **NORMAL, WARNING, CRITICAL, FATAL, UNKNOWN**.

8.14.5 Alert Log

The passive service **Log alerts** displays the last alarm raised by system log for the machine – see Figure 8-9. A mapping is made between the **syslog** severity levels and the Nagios alarm levels: **OK** gathers info, debug and notice alarms; **WARNING** gathers warn and err alarms; **CRITICAL** gathers **emerg**, **crit**, **alert**, **panic** alerts.

8.14.6 I/O Status

The I/O status reports the global status of HBA, disks and LUNs on a cluster node.

Refer to Chapter 9, section *Monitoring Node I/O Status* for more information.

8.14.7 Postbootchecker

The **postbootchecke**r tool carries out various analyses after a node is rebooted. It communicates the results of its analyses to the corresponding passive service.

8.15 Management Node Nagios Services

These services are available on the management node only.

8.15.1 MiniSQL Daemon

This active service uses the **check_proc** plug-in to verify that the **msql3d** process is functioning correctly. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.2 RMS Daemon

This active service uses the **check_proc** plug-in to verify that the **RMSD** process is functioning correctly. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.3 Quadrics Switch Manager

This active service uses the **check_proc** plug-in to verify that the **swmgr** process is functioning correctly. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.4 ClusterDB

This active service uses the plug-in **check_clusterdb.pl** to check that connection to the database is being made correctly. It remains at the **OK** alert level whilst the connection is possible but switches to **CRITICAL** if the connection becomes impossible.

8.15.5 Cron Daemon

This active service uses the **check_proc** plug-in to verify that the **cron** daemon is running on the system. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.6 Compute Power Available

A Bull plug-in checks the computer power available and the alert level associated with it and then displays the results in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.7 Global Filesystems bandwidth available

A Bull plug-in checks the bandwidth for the global filesystem and the alert level associated with it and then displays the results in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.8 Storage Arrays available

A Bull plug-in checks how much space is available for the storage arrays and the alert level associated with it and then displays the results in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.9 Global Filesystem Usage

A Bull plug-in checks the global filesystem usage and the alert level associated with it and then displays the results in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.10 I/O pairs Migration Alert

A Bull plug-in checks the I/O pairs status and the alert level associated with them and then displays the results in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.11 Backbone Ports Available

This service calculates the percentage of ports which are usable on the backbone switches. All the ports which are not usable have to be in the state *administratively down*.

The results are displayed in the Availability Indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

8.15.12 Quadrics Ports Available

This service calculates the number of ports (external and internal links) which are available for the Quadrics switches. These are displayed as a percentage of the total number of **Quadrics** ports for the cluster.

Note:
When internal links are included the total number of ports shown will be greater than the number of physical ports possible for a **Quadrics** switch. For example, a switch with 32 ports will display 96 ports within **NovaScale Master – HPC Edition.**

8.15.13 HA System Status

This service is based on the output of the **clustat** command. It displays the state of the management nodes which are running with High Availability. As soon as one or more management nodes rocks to the 'offline' state the service displays a list of all the nodes in the 'offline' state and returns an alert level of **CRITICAL**. If all the management nodes are 'online' then the service returns **OK**.

8.15.14 Kerberos KDC Daemon

This active service uses the plug-in **check_proc** to check if the daemon **krb5kdc** is running on the system. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.15 Kerberos Admin Daemon

This active service uses the plug-in **check_proc** to check if the **kadmind** daemon is running on the system. It remains at the **OK** alert level whilst the daemon is running but switches to **CRITICAL** if the daemon is stopped.

8.15.16 LDAP Daemon

This active service checks if the **check_ldap** plug-in which the Lightweight Directory Access Protocol **(LDAP)** uses with **Lustre** is working correctly. This plug-in makes a connection to **LDAP** using **fs=lustre** as root for the naming hierarchy.

8.15.17 Lustre filesystems access

This is a passive service which is run every 10 minutes by a cron. The cron connects to a client node taken from a specified group at random, for example a Compute Node, and attempts to create and write (stripe) a file on all the **Lustre** file system directories that are listed in the Cluster DB and that are mounted on the node. The file is deleted at the end of the test. If the operation is successful an **OK** code is sent to Nagios with the message *'All Lustre filesystems writable'*. If not, a **CRITICAL** code is returned with the message *'Lustre problem detected'*.

The service uses the **lustreAccess.group** parameter defined in the **/etc/nsmhpc/nsmhpc.conf** file to specify the group containing the nodes that can be used for the test (default: COMP).

8.15.18 NFS filesystems access

This is a passive service which is run every 10 minutes by a cron. The cron connects to a client node taken from a specified group at random, for example a Compute Node, and looks for all the NFS filesystems mounted on this node. It then tries to create and write a file in a specified sub-directory, on all NFS filesystems. The file is deleted at the end of the test. If the operation is successful an **OK** code is sent to Nagios. If not, a **CRITICAL** code is returned with detailed information.

The service uses three parameters, defined in the **/etc/nsmhpc/nsmhpc.conf** file:

- **nfsAccess.group**, which specifies the group containing the nodes that can be used for the test (default: COMP).
- **nfsAccess.directory**, which specifies an existing sub-directory in the filesystem where the file test will be created.
- **nfsAccess.user**, which specifies a user authorized to write in the sub-directory defined in the **nfsAccess.directory** parameter.

8.15.19 InfiniBand Links available

This service calculates the percentage of links that are usable for the **InfiniBand** switches.

The results are displayed in the Availability indicators view pane on the top right hand side of the opening window for the Map button as shown in Figure 8-2.

The administrator must specify two parameters in the **/etc/nsmhpc/nsmhpc.conf** file:

- **indicator.ib.numUpLinks**, which specifies the number of installed up links (top switches <-> bottom switches)
- **indicator.ib.numDownLinks**, which specifies the number of installed down links (bottom-switches <-> nodes)

According to these values and the values returned by the ibs tool, the service will be able to define the availability of the InfiniBand interconnect.

See Chapter 3 in the BAS4 *Maintenance Guide* for more information on the IBS tool.

8.16 Ethernet Switch Services

The Ethernet switches which are not used should be set to *disabled* so that Ethernet switch monitoring works correctly. This is usually done when the switch is first configured. The services for each Ethernet switch are displayed when the switch is selected in either the cluster host group or the host window.

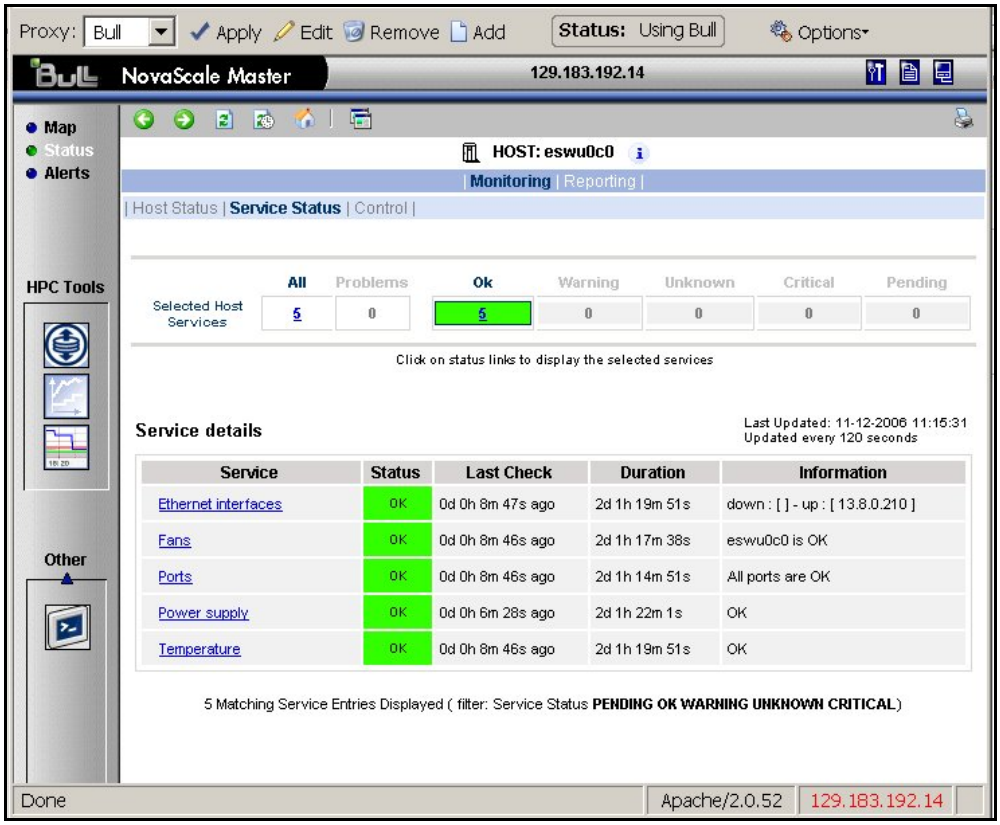


Figure 8-14. Ethernet Switch services

8.16.1 Ethernet Interface

The **Ethernet interfaces** service checks that the Ethernet switch is responding by using a ping to its IP address.

8.16.2 Power supply

The **Power supply** service checks the power supply is functioning properly by using the **check_esw_power.pl** plug-in.

8.16.3 Temperature

The **Temperature** service monitors the temperatures of the Ethernet switches by using the **check_esw_temperature.pl** plug-in.

8.16.4 Fans

The **Fans** service monitors the fans for the Ethernet switches using the **check_esw_fans.pl** plug-in.

8.16.5 Ports

The **Ports** service monitors the ports for the switches. If one or more ports are detected as being in a *notconnect* state, this service will display the WARNING state and indicate which ports are unavailable.

8.17 Portserver Services

The Portserver has to be configured to send traps. The **Current threshold exceeded** and **Temperature threshold exceeded** traps have to be activated and the destination address for the traps has to be the IP address of the management node (or the IP alias if High Availability is in place).

The configuration settings are:

```
set snmp trap_dest=<@IP noeud d'admin>
set snmp curr_thresh_exc_trap=on
set snmp temp_thresh_exc_trap=on
set snmp login_trap=off
set snmp auth_trap=off
set snmp cold_start_trap=off
set snmp link_up_trap=off
```

8.17.1 Temperature Threshold Service

This passive service indicates when the temperature threshold has been crossed by the portserver. The alert generates a SNMP trap which moves up the temperature monitoring service.

8.17.2 Current Threshold Service

This passive service indicates when the current threshold tolerated for a port has been exceeded by the portserver. The alert generates a SNMP trap which moves up the current monitoring service.

8.18 More Nagios Information

See the Nagios documentation for more information, in particular regarding the configuration. Look at the following web site for more information
http://nagios.sourceforge.net/docs/2_0/

In addition look at the NovaScale Master - HPC Edition documentation suite, this includes an *Installation Guide*, a *User's Guide*, an *Administrator's Guide* and a *Remote Hardware Management CLI Reference Manual*.

Chapter 9. Storage Device Management

Bull cluster management tools provide services to manage a large amount of storage systems and storage resources. This chapter explains how to setup the management environment, and how to use storage management services.

The following topics are described:

9.1 Overview of Storage Device Management for Bull HPC Clusters

Bull HPC clusters may contain various kinds of storage devices. Thus, storage device management may quickly become a complex task, due to the variety and the number of management interfaces.

Using Bull storage management services the cluster administrator will be able to:

- Monitor the status of storage devices
- Monitor the status of brocade switches
- Monitor storage within cluster nodes
- Get information about faulty components
- Get synthetic reports for the storage resources
- Automate the deployment of storage device configurations
- Ensure consistency between storage systems, I/O nodes and Lustre configurations
- Configure individual storage devices using a command line interface from the cluster management station
- Obtain access to the management tools for each storage device, regardless of its user interface.

Bull HPC clusters are deployed with both a specific hardware infrastructure and with software packages to simplify and unify these management tasks.

The hardware infrastructure enables the management of all the storage devices from the cluster Management Nodes:

- Built-in LAN management ports of the storage devices are connected to the cluster management network.
- Built-in serial ports of the storage devices are connected to the cluster management network, using terminal servers.
- Management stations or proxy servers (for example Windows stations) hosting device management tools are connected to the cluster management network, or are reachable from the Management Nodes.

The software packages installed on the cluster Management Node and on other cluster nodes provide various device management services:

- **Device monitoring**. A device inventory is performed and detailed descriptions of all the storage devices are stored in the cluster data base. The storage devices are monitored by the cluster Management Node, using standardized protocols such as **SNMP**, **syslog**, or proprietary interfaces. The Management Node waits for event notification from the devices. To prevent silent failures, forced updates are scheduled by the Management Node. All the events are automatically analyzed and the cluster DB is updated to reflect status changes. Storage device status can be monitored using **NovaScaleMaster – HPC Edition** and by querying the cluster DB with the **storstat** command. These services enable the browsing of a global view covering all the storage devices to a more detailed view focusing on a single storage device.

- **Advanced device management**. Administrators trained to manage the storage devices and familiar with the terminology and operations applicable to each kind of storage device can use the command line interfaces available on the cluster Management Node. These commands are specific to a family of storage system (for example **ddn_admin**, **nec_admin**, etc.). They enable the reading of configuration and status information and also configuration tasks to be performed. The syntax and output are as close as is possible to the information provided by the device management tools provided with the storage system. The most useful information and operations are available through these commands. Nevertheless, they do not offer all the management services for each device. Their advantage is that they provide a command line interface on the cluster Management Node They can also be used to build custom tasks, by parsing command outputs or creating batches of commands.

Warning:

Changing the configuration of a storage device may affect all the cluster nodes using this device.

- **Access to management tools**. The storage administrator trained to manage storage devices can also access to the management tool for each storage device. The serial ports can be used with conman (or telnet). The Ethernet ports can be connected with telnet or a web browser. Management software on proxy UNIX servers can be used with **ssh** (command mode) or X11 (graphical applications). Similarly, an ssh service and a VNC server are provided for Windows, in order to enable access to the management software on proxy Windows servers, either in command mode or in graphical mode.
- **Storage device configuration deployment.** For small clusters, the administrator can use either the device specific commands installed on the cluster Management Node, or the tools for each storage device. For medium to large clusters, there are often lots of storage systems with the same hardware and logical configuration. For these kinds of complex environments, configuration deployment services are provided. The administrator can prepare storage device configuration models, and automatically deploy these models on multiple storage devices. The storage deployment service also configures the I/O nodes accordingly, and transmits the configuration to Lustre management tools, to ensure consistent configurations between storage systems, nodes, and Lustre.
  These services are available only in command mode.

Warning:

System Administrators must be trained to manage the storage devices, and be familiar with the terminology and operations applicable to each kind of storage device. They must be aware of the impact of updating a storage device configuration

The next sections explain how to setup and use this environment.

9.2 Monitoring Node I/O Status

Each node is monitored and many I/O errors reported in **syslog** are tracked. A global I/O status is computed locally on each node and is reported to the management station using dedicated **syslog** messages.

The current I/O status of each node can be verified by displaying the "I/O status" service of the node using NovaScale Master for HPC.

The semantic of the service is as follows:

OK	No problem detected
WARNING	An I/O component in WARNING state is in an unstable state but the resource is still available. It may also indicate that the current number of I/O components is different from its expected reference number.
CRITICAL	Degraded service. Maintenance operation mandatory Hereafter is a list of possible critical errors: A fatal disk error has been reported by the linux I/O stack in **syslog** A fatal HBA error has been reported by a device driver in syslog A link down transition has been notified by a device driver A LUN resource cannot be acceded by a multipath pseudo-device. A device referenced by the persistent binding mechanism (alias) is missing.
UNKNOWN	Can't access the status
PENDING	Not yet initialized

The I/O status transmitted by each node to the management station results of the synthesis of multiple controls. This detailed information is available on each node using the **lsiodev** command:

```
lsiodev -l
```

The I/O status monitoring service builds a reference during its initial startup, usually at the first boot of the node.

The reference contains the expected number of various classes of devices (named "I/O counters").

Two reference counters (**nb_io_adapters** and **nb_local_disks**) are stored on the management station in cluster DB in the table node_profile. The other reference counters are stored on the local node.

At boot time the **nb_io_adapters** and **nb_local_disks** counters are automatically adjusted from the cluster DB node I/O profile.

You can view details of I/O status reference counter values of each node by the link **I/O status details** of the **I/O status** service on the node using NovaScale Master for HPC.

I/O Status Details of node : nova5

- The number of I/O resources is different from expected
- === Global I/O Status is WARNING ===

I/O Counters of node : nova5

Status	Counter	Value	Definition	OK State Counter	Value
WARNING	nb_io_adapters	2 / 5	I/O adapters and internal chips	nb_io_adapters_configured	2 / 5
WARNING	nb_local_disks	3 / 4	Physical disks	nb_local_disks_ok	3 / 4
OK	nb_io_ports	1 / 1	I/O ports	nb_io_ports_connected	1 / 1
OK	nb_fixed_luns	3 / 3	Fixed LUNs (/dev/sd*) directly mapped to local disks	nb_fixed_luns_ok	3 / 3
WARNING	nb_reconf_luns	10 / 8	Reconfigurable LUNs (/dev/sd*) from external storage or RAID adapter	nb_reconf_luns_ok	10 / 8
OK	nb_pseudos	0 / 0	Multipath pseudo-devices (/dev/dm-*, /dev/emcpower*)	nb_pseudos_ok	0 / 0
OK	nb_iopaths	0 / 0	Multipath I/O paths (under pseudo-devices)	nb_iopaths_ok	0 / 0
OK	nb_aliases	8 / 8	Device aliases (/dev/ldn*) for LUNs or pseudo-devices	nb_aliases_ok	8 / 8

(Counter Value = Current / Expected)

I/O Resources of node : nova5

Adapter 03:01 LSI LSI LSI53C1030 Driver: mptspi CONFIGURED	**host0**	**sdb** (0:0:10:0) **OK** (Fixed LUN)	Physical Disk **sdb** **OK** SEAGATE SPI 286102MB
		sdc (0:0:11:0) **OK** (Fixed LUN)	Physical Disk **sdc** **OK** SEAGATE SPI 286102MB
		sda (0:0:9:0) **OK** (Fixed LUN)	Physical Disk **sda** **OK** SEAGATE SPI 286102MB
	host1		
Adapter 2d:01 Emulex LP11000 Driver: lpfc CONFIGURED	**host2** (Port) WWN: 10:00:00:00:c9:4b:c0:9a CONNECTED	**sdd** (2:0:0:0) **OK** DDN 1000MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.24**
		sde (2:0:0:1) **OK** DDN 1048576MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.25**
		sdl (2:0:0:12) **OK** DDN 49896MB (Reconfigurable LUN, FC)	
		sdm (2:0:0:13) **OK** DDN 49896MB (Reconfigurable LUN, FC)	
		sdf (2:0:0:2) **OK** DDN 1000MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.26**
		sdg (2:0:0:3) **OK** DDN 1048576MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.27**
		sdh (2:0:0:4) **OK** DDN 1000MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.28**
		sdi (2:0:0:5) **OK** DDN 1048576MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.29**
		sdj (2:0:0:6) **OK** DDN 1000MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.30**
		sdk (2:0:0:7) **OK** DDN 1048576MB (Reconfigurable LUN, FC)	← Alias **ldn.ddn0.31**
ldn.ddn0.24 Alias **OK** linked to **sdd**			
ldn.ddn0.25 Alias **OK** linked to **sde**			
ldn.ddn0.26 Alias **OK** linked to **sdf**			
ldn.ddn0.27 Alias **OK** linked to **sdg**			
ldn.ddn0.28 Alias **OK** linked to **sdh**			
ldn.ddn0.29 Alias **OK** linked to **sdi**			
ldn.ddn0.30 Alias **OK** linked to **sdj**			
ldn.ddn0.31 Alias **OK** linked to **sdk**			

Figure 9-1. I/O Status Details NovaScale Master HPC Edition example screens

The **iorefmgmt** command is used to manage I/O device monitoring reference counters.

To get the list of the reference counter enter:

```
iorefmgmt -g
```

Use the help or the man page to obtain a description of the counters used, alternatively see the definitions in the section below.

If the reference is wrong, it can be updated as follows:

```
iorefmgmt -s -n <counter_name> -v <value>
```

You can adjust reference counters to the current discovery value by the command:

```
iorefmgmt –c adjust
```

The counters **nb_io_adapters** and **nb_local_disks**" cannot be adjusted on a node.

You can manage these counters in the cluster DB node profile table on the administration station by using the command:

```
iorefmgmt -c dbset|dbget|dbdel
```

For more information use the **iorefmgmt** man page or help.

All these operations can be done from the management station, using **ssh** or **pdsh**.

9.2.1 I/O Counters Definitions

- **nb_io_adapters** is the expected number of I/O adapters on the node (a multi-port adapter is counted as 1, an internal I/O chip is also counted as one adapter).
- **nb_io_adapters_configured** is the number of I/O adapters expected to be configured (driver loaded).
- **nb_local_disks** is the expected number of physical disks on a node.
  A physical disk may be:
  - an internal disk which is directly attached,
  - a physical disk in a SCSI JBOD,
  - a physical disk behind a RAID controller.
- **nb_local_disks_ok** is the number of physical disks expected to be healthy.
- **nb_io_ports** is the expected number of Fibre Channel ports.
- **nb_io_ports_connected** is the number of Fibre Channel ports expected to be connected.
- **nb_fixed_luns** is the expected number of LUNs which are not reconfigurable.
  A LUN which is not reconfigurable is directly mapped to a physical disk.
- **nb_fixed_luns_ok** is the number of LUNs which are not reconfigurable that are expected to be accessible.
- **nb_reconf_luns** is the expected number of reconfigurable LUNs.
- **nb_reconf_luns_ok** is the number of reconfigurable LUNs expected to be accessible.
  A "reconfigurable LUN" is typically a LUN in an external storage system (usually a RAID system) or a LUN presented by a RAID HBA, on top of RAIDed local disks.
- **nb_iopaths** is the expected number of paths involved in multi-path to reach LUNs which are reconfigurable.
- **nb_iopaths_ok** is the number of paths involved in multipath expected to be alive.
- **nb_aliases** is the expected number of aliases on Fibre Channel block devices.
  Aliases are used to obtain a persistent device naming scheme, regardless of the order that the FC devices are detected.
- **nb_aliases_ok** is the number of aliases on Fibre Channel block devices expected to be correctly mapped.
- **nb_pseudos** is the expected number of multipath pseudo-devices on a node.
- **nb_pseudos_ok** is the number of multipath pseudo-devices expected to be usable.

9.3 Monitoring Storage Devices

This section explains how the administrator can monitor and get information about all the managed storage systems of the cluster, using a unified interface. The two following interfaces are available for the administrator:

- Graphical User Interface (NovaScale Master – HPC Edition):
  - Hosts and service monitoring for storage devices.
  - Storage views, providing detailed information regarding the storage systems.
- Command line interface:
  - **storstat** command, to query the ClusterDB for storage information.
  - Archiving of **syslog** messages.

Note:

The monitoring relies on information stored in the **ClusterDB**. This information is updated periodically and also when failures or repairs are notified by storage devices. The monitoring information is therefore not updated in real-time when silent state changes occur, such as repairs.

The administrators can force a refresh of the Data Base information using the **storcheck** command:

```
storcheck -c <cluster_name>
```

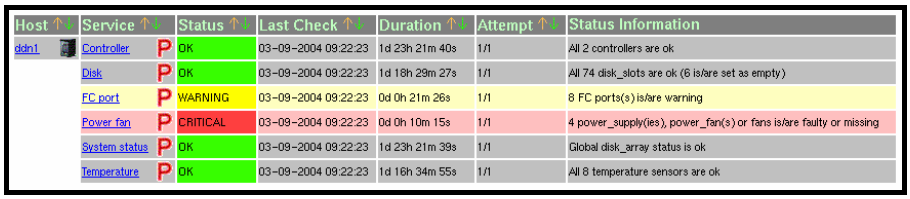
This command will check all the storage systems of the cluster. It is possible to reduce the scope to a single storage system:

```
storcheck -c <cluster_name> -n <disk_array_name>
```

9.3.1 NovaScale Master - HPC Edition: Host and Service Monitoring for Storage Devices

Storage device monitoring is integrated in the global monitoring of the cluster. Each storage system is identified by a host and associated service, regardless of the number of controllers and Ethernet ports.

NovaScale Master - HPC Edition continuously updates the host status and service status values, without any administrator intervention. All **NovaScale Master - HPC Edition** features and services apply to storage devices. Nevertheless, the administrator using **NovaScale Master - HPC Edition** should be aware of the following points.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
ddn1	Controller P	OK	03-09-2004 09:22:23	1d 23h 21m 40s	1/1	All 2 controllers are ok
	Disk P	OK	03-09-2004 09:22:23	1d 18h 29m 27s	1/1	All 74 disk_slots are ok (6 is/are set as empty)
	FC port P	WARNING	03-09-2004 09:22:23	0d 0h 21m 26s	1/1	8 FC ports(s) is/are warning
	Power fan P	CRITICAL	03-09-2004 09:22:23	0d 0h 10m 15s	1/1	4 power_supply(ies), power_fan(s) or fans is/are faulty or missing
	System status P	OK	03-09-2004 09:22:23	1d 23h 21m 39s	1/1	Global disk_array status is ok
	Temperature P	OK	03-09-2004 09:22:23	1d 16h 34m 55s	1/1	All 8 temperature sensors are ok

Figure 9-2. Detailed service status for a storage host

The host and service monitoring offers uniform monitoring for all the cluster components, with history and statistical capabilities. It provides for each storage system a general view of the major functional domains.

However, this monitoring does not allow the easy identification of the storage devices among other cluster components nor individual faulty hardware components to be identified. These limitations are compensated by the use of Storage Views (see 9.3.2 *NovaScale Master - HPC Edition: Storage &* I/O Information).

9.3.1.1 Host Semantic

The host name is a logical name, which uniquely identifies a storage system. But caution, it is not bound to an IP address; it is not possible to ping using this parameter.

The host status indicates whether the storage system is manageable or not:

UP	The storage system responds through the management interfaces
UNREACHABLE	Some network problems prevent the management interface from being reached.
DOWN	The management interfaces of the storage system do not answer to requests. But note that from a storage point of view, the storage system may process I/O requests from attached hosts.

9.3.1.2 Service Semantic

Several generic services are defined for storage systems. They reflect the global status of a class of components in the selected storage system:

- Disk
- Power-Fan
- Temperature
- Controller
- FC ports
- System status.

Disk Service

This service describes the global status for the **HDD**s. It monitors both disk failures and if any disks have been removed (for example for maintenance purpose).

OK	**No problem** Criteria: No disk errors All referenced disks are present
WARNING	**Maintenance operation must be scheduled** Criteria: Some disk failures, and / or removed referenced disks Does not meet the criteria for critical status.
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: The number of faulty / missing disks is higher than the number of spare disks.
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Note:

The cluster database has been initialized with a detailed status including all populated and empty disk slots. The administrator, who decides to permanently remove some HDDs, must manually update the database reference configuration (using the **storstat –u** command). Otherwise, empty slots due to a permanent removal will lead to a permanent WARNING status.

Power-Fan Service

Describes the global status for the power supply and for the fan modules. These two kinds of hardware parts are grouped and monitored using a single service.

OK	**No problem** Criteria: • All power supplies and fans are ok • All reference power supplies and fans are present
WARNING	**Maintenance operation must be scheduled** Criteria: • Some power supplies and/or fans are in the warning or critical state • Does not meet the criteria for critical status.

CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • The percentage of faulty/missing power supplies or fans objects has reached the threshold defined in **/etc/storageadmin/storframework.conf** (**service_power_fan_critical_threshold** parameter).
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Temperature Service

Describes the global status for temperature sensors.

OK	**No problem** Criteria • All temperature sensors are OK
WARNING	**Maintenance operation must be scheduled** Criteria: • Some temperature sensors are not in the OK state • Critical criteria not met
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • Some temperature sensors are in the critical state.
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Controller Service

This service shows the controller status. The controller refers to the storage system elements in charge of host connection and I/O processing.

OK	**No problem** Criteria: • All controllers are OK
WARNING	**Maintenance operation must be scheduled** Criteria: • Some controllers have a warning state and none are faulty (or missing).
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • One controller or more is faulty (or missing).
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Fibre Channel Port Service

This service shows the host connectivity status:

OK	**No problem** Criteria: • All FC ports are OK.
WARNING	**Maintenance operation must be scheduled** Criteria: • Not in critical status • Some ports have a warning status
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • One or more ports are in a critical status.
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Note:

If the FC link is connected to a switch, and the link is broken 'after' the switch and not between the controller and the switch, the failure is not detected by the disk array and therefore will not be displayed by the FC port service.

System Status Service

This service is a collector and gathers all problems of the storage system. If one of the services described above is warning or critical, the system status service will be critical. This service also reflects the other problems which may arise but are not classified in one of the previously defined services. For example, all the other services may be OK, while the system status is warning or critical.

OK	**No problem** Criteria: • Disk array semantic.
WARNING	**Maintenance operation must be scheduled** Criteria: • Some of the other services are warning (but none critical). • The storage system has detected a warning which is not reported by one of the other services.

CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • One of the other services is critical. • The storage system has detected a critical error which is not reported by one of the other services.
UNKNOWN	**Can't access the status**
PENDING	**No yet initialized**

9.3.2 NovaScale Master - HPC Edition: Storage & I/O Information

NovaScale Master – HPC Edition contains specific views, which focus on the monitoring of storage devices and I/O systems for the nodes connected to these devices. It enables administrators to pinpoint faulty hardware components, and provides detailed reporting and configuration information for storage systems.

The Storage and I/O information view is selected by clicking on the **Storage overview** icon on left hand side of the **NovaScale Master – HPC Edition** console – see Figure 9-3. A pop-up window appears containing a view pre-selected on the summary view of the storage systems and hardware component status – see Figure 9-4.

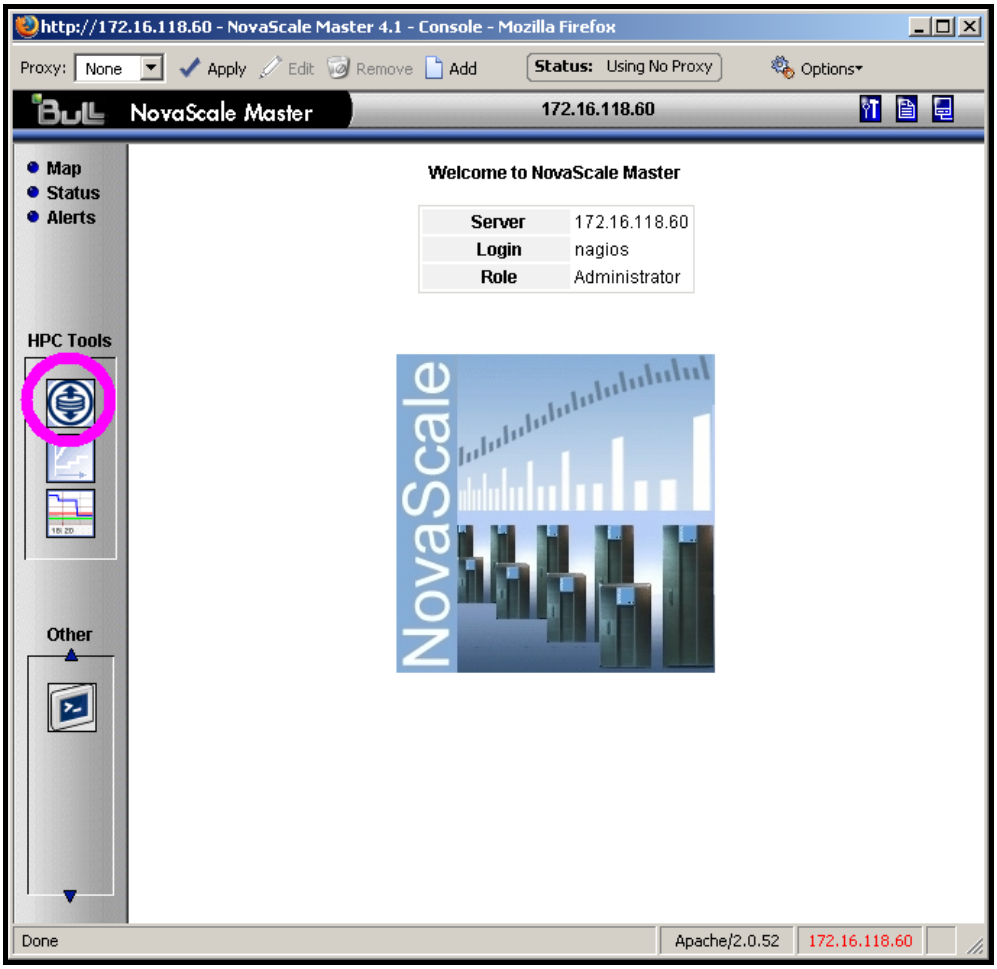


Figure 9-3. Nova Scale Master opening console window with the Storage overview icon circled

9.3.2.1 Storage Views

The storage views provide information about:

- **Disk arrays**. Their status refers to the last known operational status of the storage system under review. It is similar to the 'system status' service in NovaScale Master host and service views. For example a storage system that does not answer to management requests is considered as faulty.
- **Individual hardware components** (Disk, FC port, RAID controller, and so on). There is no equivalent in the host and service monitoring services, that provide a single service for all the disks of a storage system.

Note:

The disk array status is a superset of the individual hardware components status. It is usually managed by the disk array and is not limited to the hardware components managed by storage views. Therefore the disk array status may be more severe than the worst status of the individual hardware components.

The status used in the storage views are the following ones:

OK	No problem
ATTENTION	Maintenance operation must be scheduled, but the system still delivers the expected service.
FAILED	Degraded service. Maintenance operation mandatory.

9.3.2.2 Storage Overview

This view offers a synthesis of the Storage devices monitored in the cluster.

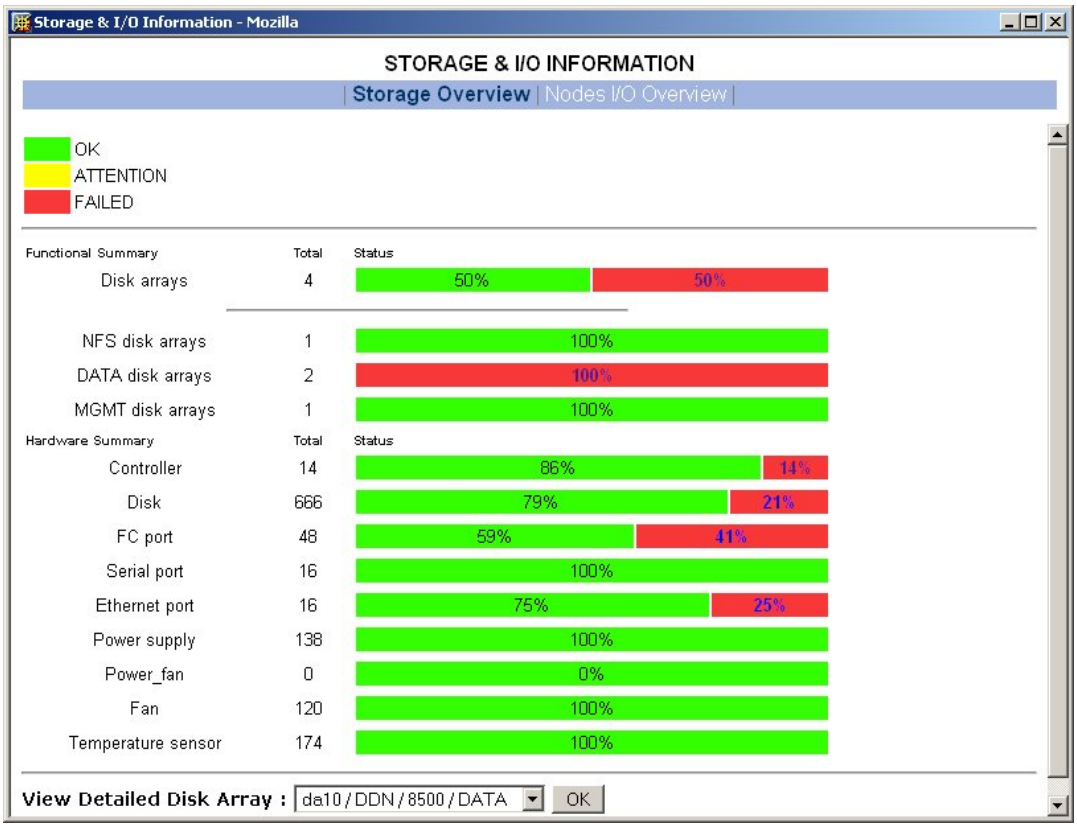


Figure 9-4. Storage overview

Functional Summary

This diagram refers to storage systems. It sorts the storage systems according to their operational status and to their respective roles (using Lustre OSS/MDS terminology).

Hardware Summary

This diagram provides statistics on low level hardware components such as HDDs, Fibre Channel ports, RAID controllers, power supplies, FANs, etc. The diagram is displayed by family of components, sorted by state.

The administrator clicks the ATTENTION and FAILED percentages links in the Storage overview pop-up window to get an inventory list of storage systems or hardware components in the selected state – see Figure 9-6.

9.3.2.3 Inventory View of Storage Systems and Components requiring attention

This view - Figure 9-6 - displays the list of faulty components that should be either examined or replaced. The components are grouped by storage system. For each component, the view displays:

- The description of the component

- Its status
- Location information of the component within the device and within the cluster, its rack level and label.

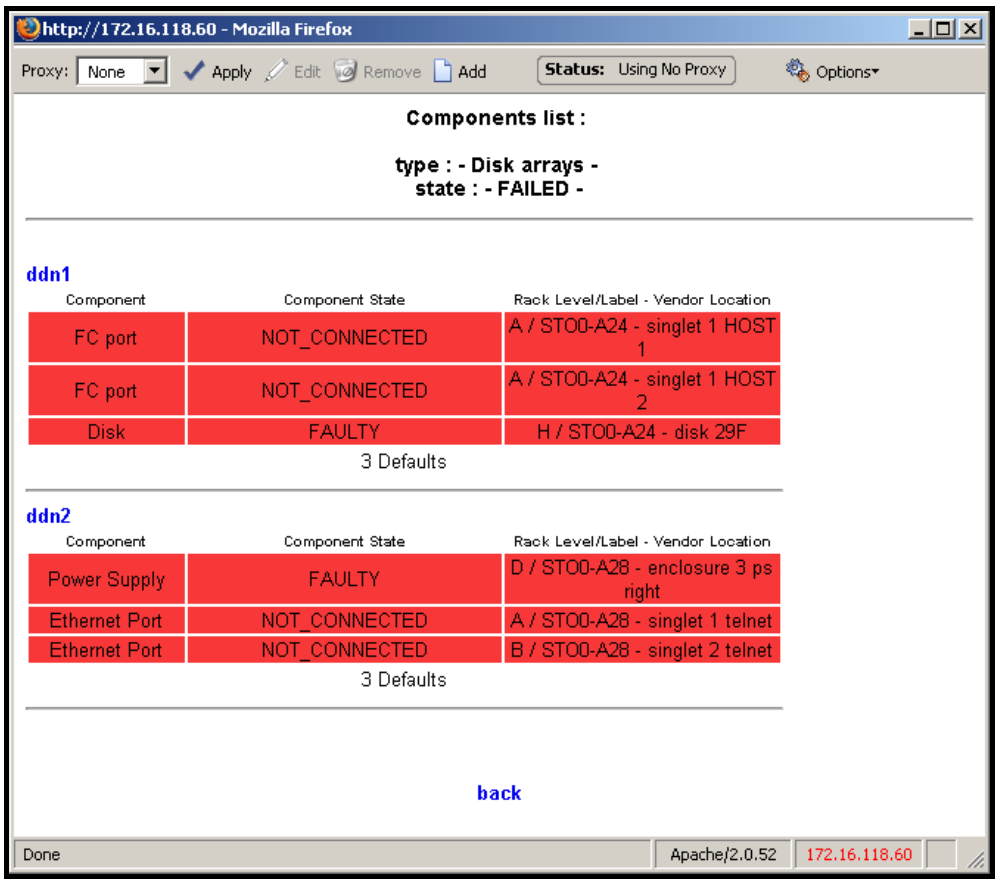


Figure 9-5. Inventory view of faulty storage systems and components

Note:

The hardware components whose status is OK are not listed

This view is useful for planning maintenance operations for the components that are to be examined or replaced.

9.3.2.4 Detailed View of a Storage System

The Storage detailed view - Figure 9-6 - can be displayed by selecting a storage system in the Storage Summary Overview (see Figure 9-4).

This view provides detailed information for the selected storage system:

- Technical information (disk array status, firmware version, addressing information for management purposes, etc.).
- Front and rear diagram view, where the status of all the hardware components is represented by a color code.
- I/O cell and I/O path information:

- An I/O cell is a set of nodes and storage systems functionally tied together. For example, the two nodes of an HA pair, and their storage systems form an I/O cell.
- An I/O path is a logical path between a node and the host port of a storage system. When a point-to-point connection is used, the I/O path is physically represented by a cable. In SAN environment, the I/O path represents both the I/O initiator (the node) and I/O target (the host port of the storage system).

- "Error List" hyperlink (list of faulty components).
- "Lun / Tier / Zoning List" hyperlink (information about the logical configuration of the storage system).

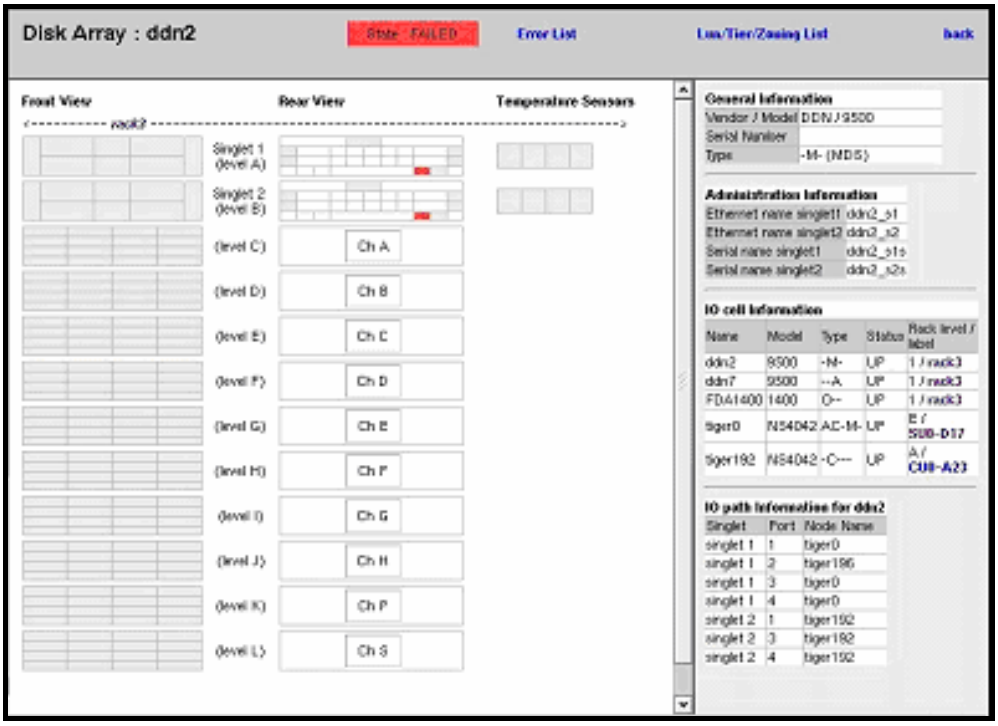


Figure 9-6. Storage detailed view

In the Storage Detailed view the item's description is shown through the use of mouse Tool tips.

9.3.2.5 Nodes I/O Overview

This view – Figure 9-7 – offers a synthesis of the I/O information about the nodes of the cluster.

It shows I/O status statistics and allows the list of nodes to be filtered on a selected I/O status value.

Clicking on the I/O status value of a node allows detailed information about the I/O resources of the node and the associated I/O counters to be displayed.

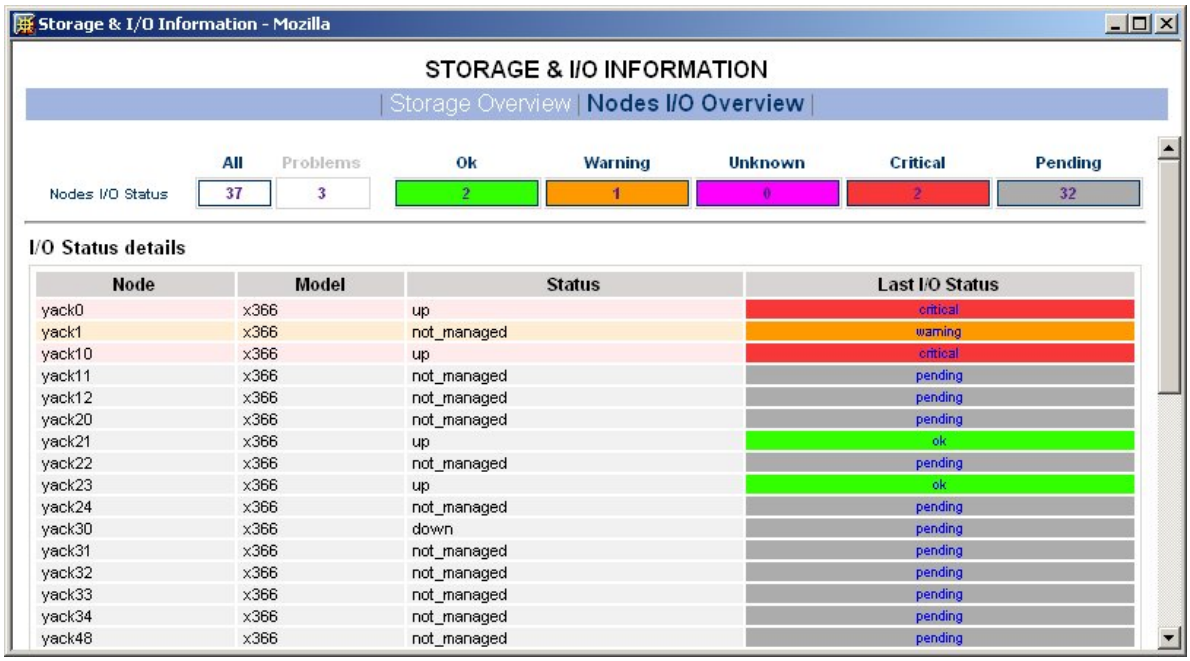


Figure 9-7. Nodes I/O Overview

9.3.3 Querying the Cluster Management Data Base

The **storstat** command obtains status information from the **ClusterDB** and formats the results for storage administrators.

Please refer to the help page for this command for more information:

```
storstat -h
```

The following paragraphs describe the most useful options.

9.3.3.1 Checking Storage System Status

To display all the registered storage systems with their status and location in the cluster use the command below. The location is based on rack label and position in the rack:

```
storstat -a
```

To display a list of faulty storage systems:

```
storstat –a -f
```

To check the status of a storage system using the name identifying the storage system:

```
storstat -a -n <disk_array_name> -H
```

9.3.3.2 Checking Status of Hardware Elements

To display a list of faulty components for all registered storage systems:

```
storstat -d -f -H
```

For each element, the following information is displayed:

- Disk array name
- Enclosure of the disk array housing the component
- Type of the component
- Status of the component
- Location of the component within the enclosure or disk array. This location uses vendor specific terminology
- Location of the enclosure (or disk array) in the cluster.

The **–n <disk_array_name>** flag can be used to restrict the list to a single storage system.

To display a list of all the components for a storage system:

```
storstat -d -n <disk_array_name>
```

Note:

If the **–n** flag is omitted, the list is extended to all the registered storage systems.

To check the number of available or faulty elements in the cluster (or in a selected storage system):

```
storstat -c
```

or

```
storstat -c -n <disk_array_name>
```

9.3.4 Checking the Logs for Storage Devices

The logs of the DDN S2A storage systems may be stored on the Management Node. The default location is **/var/log/DDN/**. Each singlet has a dedicated log file, identified either by its IP address or by an IP name.

9.4 Monitoring Brocade Switch Status

Each Brocade Fibre Channel switch is monitored by **NovaScale Master - HPC Edition**.

The same check period as for Ethernet switches will be used (10 minutes, possibly configurable). No specific configuration is required on the FC switches in order to be able to use the telnet interface.

Several generic services are defined for brocade switch. They reflect the global status of a class of components of the selected switch.
A mapping between SNMP MIB (Management Information Base) values available and returned from the switch and NS MASTER HPC status give the following set of states for each managed services:

Ethernet interface Service

OK	**No problem** Criteria: • The Fping of the Ethernet interface is OK
CRITICAL	Criteria: • The Fping of the Ethernet interface is KO

FC port

OK	**No problem** Criteria: • All FC ports are OK.
WARNING	**Maintenance operation must be scheduled** Criteria: • Not in critical status • Some ports have a warning status • Number of operating port higher than expected in the DB (fc_switch.oper_port_threshold)
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • One or more ports are in a critical status. • Number of operating ports lower than expected (fc_switch.oper_port_threshold)
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Fans

OK	**No problem** Criteria: • All fans are present and OK
WARNING	**Maintenance operation must be scheduled** Criteria: • Some fans are in the warning state • Does not meet the criteria for critical status.
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • At least one of the fan is in a critical state
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Power Supply

OK	**No problem** Criteria: • All power supplies are present and ok • No Power Supply is detected on the switch.
WARNING	**Maintenance operation must be scheduled** Criteria: • Some power supplies are in the warning state • Does not meet the criteria for critical status.
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • At least one of the power supplies is in a critical state
UNKNOWN	**Can't access the status**
PENDING	**Not yet initialized**

Temperature Sensor

OK	**No problem** Criteria: • All Temperature sensor are present and OK
WARNING	**Maintenance operation must be scheduled** Criteria: • Some Temperature sensor are in the warning state • Does not meet the criteria for critical status.
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • At least one of the Temperature Sensor is in a critical state

UNKNOWN	Can't access the status
PENDING	Not yet initialized

Global Status

OK	**No problem** Criteria: • Global brocade switch status is ok.
WARNING	**Maintenance operation must be scheduled** Criteria: • Some of the other services are warning (but none critical). • Switch name (switchX) different as expected (fcswX)
CRITICAL	**Degraded service. Maintenance operation mandatory** Criteria: • One of the other services is critical. • The storage system has detected a critical error which is not reported by one of the other services.
UNKNOWN	**Can't access the status**
PENDING	**No yet initialized**

The different services managed by NovaScale Master HPC for the brocade switch are shown below:

Host	Service	Status	Last Check	Duration	Attempt	Status Information
fcswu0c0	Ethernet interfaces	OK	28-02-2006 11:01:15	0d 3h 4m 35s	1/1	down : [ ] - up : [10.0.0.90]
	FC ports	CRITICAL	28-02-2006 11:01:15	0d 3h 4m 35s	1/1	8 FC ports - OK [0 1 3 4 5 6 7] - WARNING [2]. Number of operating ports (2) lower than expected (4)
	Fans	OK	28-02-2006 11:01:15	0d 3h 4m 35s	1/1	All 3 Fans are OK
	Power supply	OK	28-02-2006 10:56:50	0d 18h 5m 23s	1/1	All 1 Power Supplies are OK
	Status	CRITICAL	28-02-2006 11:01:15	0d 0h 13m 20s	1/1	Global switch status is CRITICAL
	Temperature	OK	28-02-2006 11:01:15	0d 3h 4m 35s	1/1	All 4 Temperature Sensors are OK

Figure 9-8. Detailed Service status of a brocade switch

9.5 Managing Storage Devices with Bull CLI

This section introduces the commands provided for each family of devices.

These commands offer the most useful subset of management features, implemented in each storage system.

For Storage systems not listed in paragraphs 9.5.1 and 9.5.2, the administration will be done via the tools delivered with the Storage System.

9.5.1 Bull FDA Storage Systems

The administrator must be familiar with the **FDA** terminology and management tasks.

Note:

See the Bull FDA documentation for the **StoreWay FDA** model for more information on the options, parameters and possible values.

The **nec_admin** command usually requires at least two input parameters:

- The IP address (or host name) of the Windows system which hosts the FDA Storage Manager for the target FDA system.
- The name of the target FDA system.

The following services are provided by the command:

- **rankbind**
- **ldbind**
- **addldset**
- **addldsetld**
- **sparebind**
- **sparerebuild**
- **delldset**
- **ldunbind**
- **rankunbind**
- **spareunbind**
- **unconfig**
- **getstatus**
- **direct**

All the FDA arrays are supposed to be manageable using a single login / password. The **nec_admin** command enforces the parameters defined in the **/etc/storageadmin/nec_admin.conf** file as follows:

```
# NEC CLI Command path

# On Linux iSMpath="/opt/iSMSMC/bin/iSMcmd"
# On Windows iSMpath="/cygdrive/c/Program
\Files/FDA/iSMSM_CMD/bin/iSMcmd"
iSMpath = /opt/iSMSMC/bin/iSMcmd
#iSMpath="/cygdrive/c/Program\ Files/FDA/iSMSM_CMD/bin/iSMcmd"
```

```
# NEC iStorage Manager host Administrator
hostadm = administrator
# NEC iStorage Manager administrator login
necadmin = admin
# NEC iStorage Manager administrator password
necpasswd = adminpassword
```

For more information, read the man page or check the command's help.

9.5.2 DataDirect Networks Systems - DDN Commands

The administrator must be familiar with the DDN S2A terminology and management tasks. If necessary the administrator must refer to the documentation provided with S2A storage systems in order to understand the options, parameters and possible values.

The DDN specific commands usually require at least one input parameter:

- The IP address (or host name) of the target singlet for the command.

9.5.2.1 ddn_admin

This command allows you to get information from a singlet, or to configure the singlet.
The following services are provided by the **ddn_admin** command:

- **deletelun**
- **formatlun**
- **getinfo**
- **getfmtstatus**
- **getstatus**
- **setlun**
- **setzoning**
- **shutdown**
- **showall**
- **setcache**

9.5.2.2 ddn_stat

This command is used to collect statistical information.
The following services are provided by the **ddn_stat** command:

- **getbasic**
- **getlength**
- **repeatIO**
- **repeatMB**

For more information, read the man page or check the command's help.

9.5.2.3 ddn_init

This command is used for the initial setup of a singlet or a couplet. It must be used very carefully as it usually restarts the singlet(s).

The command uses the information preloaded in the ClusterDB. Some parameters may be overwritten using the command line.

ddn_init connects to each singlet through the serial port, using conman. Thus, it may be necessary to provide the name of the conman console.

A login / password is required to modify the singlet configuration. **ddn_init** attempts to connect with factory defaults login / password, using a command line supplied login / password, and with the login / password defined in **/etc/storageadmin/ddn_admin.conf**. The **ddn_admin** command then enforces the login / password defined in **ddn_admin.conf.**

9.5.2.4 ddn_conchk

This command checks the connections to a DDN system, and compares them with the connections predefined in the **ClusterDB**.

Conman, the serial network and the LAN must be ready for use in order to check the Serial/Ethernet consistency.

Attached nodes must be up, running, and reachable from the management station to check the fibre channel consistency.

9.5.2.5 ddn_set_up_date_time

This command is used to update the date and time of DDN subsystems with the UTC date and time of the management station. The administrator can specify a list of DDN systems to be synchronized.

A recommended practice, which is the installation default, is to periodically synchronize all DDN systems using a daily **cron**.

9.5.2.6 ddn_check_format

This command allows you to check the formatting status for a list of DDN systems.

9.5.2.7 ddn_firmup

This command automatically upgrades the firmware of the singlets of a DDN system. The Management Node can be used as TFTP server.

9.6 Using Management Tools

Please refer to the documentation of the storage systems to understand which management tools are available. Then determine how they can be accessed from Bull cluster Management Node using Linux utilities (**conman**, **telnet**, **web browser**, **X11**, **rdesktop client**, **ssh client**, etc.).

9.7 Configuring Storage Devices

9.7.1 Planning Tasks

Storage system configuration requires several planning operations. At least two steps are required.

STEP 1 – DEFINE THE DEVICE CONFIGURATION

The storage administrator must define the storage configuration which is required for the cluster. It is especially important for RAID storage systems, which enable the creation of logical disks (LUNs) with full flexibility in terms of number and size.

Usually, the storage configuration is a compromise of several parameters:

- The available storage resources and configuration options for the storage systems.
- The performance requirements (which may drive the choice of RAID types, LUN numbers, LUN size, stripping parameters, memory cache tuning, etc.).
- The file systems and applications requirements. It is thus necessary to identify which cluster nodes will use the storage resources, the applications and/or services running on these nodes, and the system requirements for each one.

At the end of this planning phase, the administrator must be able to define for each storage system:

- The grouping of hardware disks (HDD) and the **RAID** modes to use.
- The **LUN**s to be created on each RAID volume, with their size and, if necessary, tuning parameters.
- The **LUN** access control rules. This means how the storage system should be configured to ensure that a LUN can be accessed only by the cluster node which is supposed to use this storage resource. Depending on the way the nodes are connected to a storage system, two methods of LUN access control can be used:
    1. Port-mode **LUN** access control: describes the visibility of the LUNs on each port of the storage system
    2. **WWN**-mode LUN access control: describes the visibility of the LUNs according to the initiator's worldwide name (WWN of the host fibre channel adapter). This method requires the collection of WWN information on nodes before applying the configuration on the storage systems.
- Miscellaneous tuning parameters.

STEP 2 – DEPLOY THE STORAGE CONFIGURATION

Changing the configuration of a storage system may not be a transparent operation for the cluster nodes using storage resources which have been configured previously.

Thus the storage administrator is advised to respect the following process when deploying a storage configuration:

- Stop all the applications accessing data on the selected storage systems.

- Unmount the file systems accessing data on the selected storage systems and, if possible, shutdown the nodes.
- Modify the storage system configuration.
- Restart the attached nodes, or force them to re-discover the modified storage resources.
- Update the node's configuration.
- Mount file systems, restart applications.

9.7.2 Deployment Service for Storage Systems

Note:
This service is currently supported for DDN and FDA storage systems.

Medium and large clusters are usually built with multiple storage systems with the same hardware configuration. The purpose of the deployment service is to simplify the configuration tasks by:

- Automatically deploying the same logical configuration on multiple storage systems.
- Forcing I/O nodes to discover the storage resources and to setup a deterministic disk naming to simplify resource discovery on I/O nodes. This mechanism also ensures a persistent device naming, as well as an identical naming for the LUNs visible on each node of an HA pair.
- Transmitting storage configuration information to Lustre, to facilitate OST/MDT discovery.

This deployment service is well suited for storage systems and nodes dedicated to a single function, such as the I/O system of the cluster. It is hazardous to use it on storage systems or nodes which have multiple functions, such as nodes which are simultaneously Management Nodes and I/O nodes. Read the explanation and warnings of the next paragraphs carefully, to determine if this powerful and automated process is suitable for your cluster.

9.7.3 Understanding the Configuration Deployment Service

The configuration deployment service relies on modeling the storage system configuration. The model defines all the configuration parameters (see 9.7.1 Planning Tasks, Step 1), and establishes a relationship to the Lustre configuration. The model contains the list of the target storage systems to be configured.

The recommended process to modify the storage configuration in a large cluster, using the storage configuration deployment service, follows.

Warning:

The administrators must follow the 5 step process described in the following paragraphs. Otherwise, there is a high risk of inconsistency between storage systems, nodes, and Lustre file system, leading to a non operational file system

STEP 1 – DEFINE THE STORAGE CONFIGURATION

The administrator must either create a model to specify the storage configuration to deploy, or use an existing model.

The administrators can define multiple models. They are responsible for managing versions and for remembering the purpose of each model.

STEP 2 – DISABLE THE GLOBAL FILE SYSTEM

If necessary, backup all the data that must be preserved.

Release the storage resources used on the I/O nodes. Typically, unmount and stop the global file system.

Please refer to Lustre administration information for details.

STEP 3 – CONFIGURE THE STORAGE SYSTEMS USED BY THE GLOBAL FILE SYSTEM

The model contains all the directives to configure the storage systems. When multiple storage systems must be configured with the same configuration, the configuration operations are performed in parallel.

Warning:

The application of a model on a storage system is destructive. The configuration deployment is not an incremental process that modifies only the differences between the current configuration and the model. The first step erases the existing configuration, and then the new configuration is built using a known reference. All data will be lost.

The application of the model stops when all the commands have been acknowledged by the target storage systems. A synthetic report is provided to quickly identify which storage devices have been successfully configured and which ones have failed.

Usually, the configuration does not complete, and tasks such as disk formatting continue to run. Another command is used to check that these tasks complete.

STEP 4 – UPDATE THE CONFIGURATION OF THE I/O NODES

The model is also applied to all the I/O nodes attached to the storage devices that have been reconfigured. The I/O nodes rescan their I/O environment to detect the new I/O configuration, and check that all the resources described in the model have been detected.

Links are created to facilitate the identification of disks and simplify the deployment of Lustre.

Warning:

The rescanning of the I/O environment destroys all the disk devices detected by Fibre Channel HBAs, even if these **HBA**s are not dedicated by the global file system.

STEP 5 – TRANSMIT THE NEW STORAGE CONFIGURATION TO LUSTRE CONFIGURATION TOOLS

This last step automatically populates the ClusterDB with all the created OST and MDT devices.

The configuration deployment process is shown in the figure below.

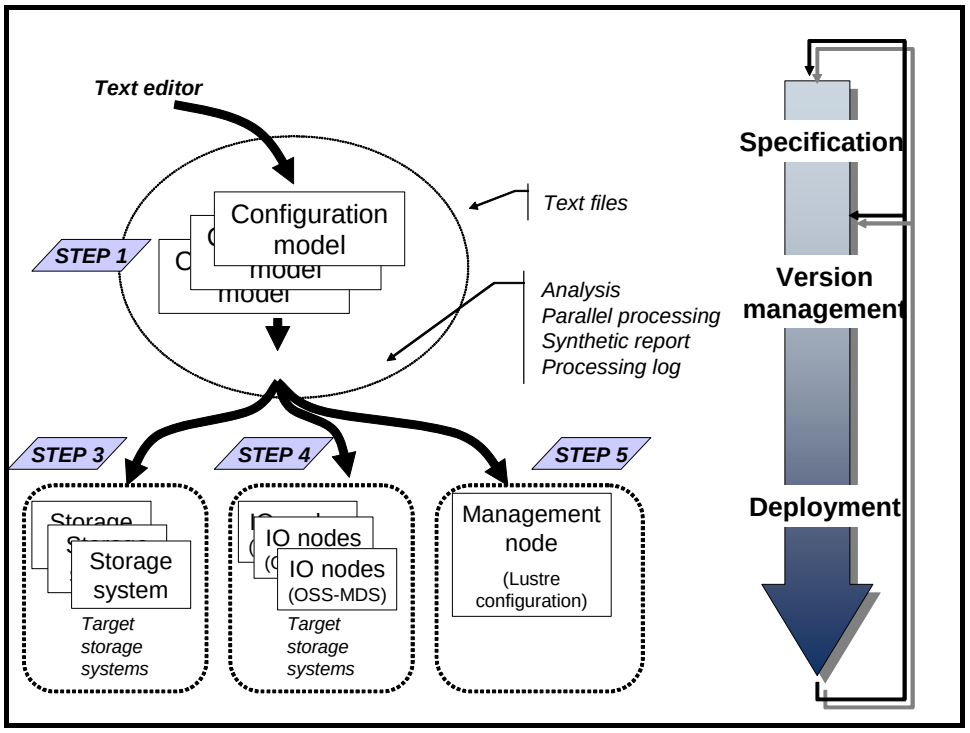


Figure 9-9. Deployment service for storage configurations

9.7.3.1 STEP 1 - Preparing and Managing Configuration Models

The configuration model is a text file. It uses an XML syntax style. To obtain details about the syntax, the administrator can refer to the **template.model** file, delivered with the rpm in **/usr/share/doc/storageadmin-framework-<version>**.

Another way to obtain a model template is to use the following command:

```
stormodelctl -c showtemplate
```

This template describes one LUN model for each supported storage system vendor (some parameters are vendor-specific).

A model is identified by its file name. The **.model** suffix is mandatory and a recommended practice is to store all the models in the same directory. The ClusterDB contains a history of the models applied to the storage systems. Thus the administrators should not change the contents of a model without changing its name.

A global model is made up of a list of LUN models.

A LUN model is a description of the configuration to be applied to a storage system; it includes:

- A description of LUNs using an associated label.
- LUN Access control rules describing the LUNs visibility for host ports.
- Lustre related directives (OST-MDT role in failover pairs).
- Storage system tuning parameters.
- A list of the storage systems to configure using the LUN model.

9.7.3.2 STEP 2 – Disabling the Global File System

Before changing the configuration of storage systems, it is mandatory to stop I/O activity, stop the global file system and unmount the local file systems on the nodes attached to the storage systems.

Refer to the *Parallel File Systems* Chapter for more information about how to disable Lustre.

9.7.3.3 STEP 3 - Applying a Model to Storage Systems

Note:

It is possible to skip the storage system configuration phase and to use only the I/O Node and Lustre configuration phases. In this case the administrator must manually configure the storage system, in accordance with the configuration defined in the model. This way of operating is also useful when the administrator does not want to erase the existing configuration (for example to safeguard existing data), or for the storage systems that do not support the automatic configuration.

The application of a configuration model to storage systems is performed in two phases:

1. The configuration of storage resources and tuning of parameters
2. The application of LUN access control directives

If the LUN access control method used is the **WWN**-mode (use of <NodePort> directives in the model file, see the model template for detailed description), it is necessary to update the cluster database with information about the Fibre Channel adapters of the cluster nodes before applying the configuration model. This is done using the following command:

```
ioregister -a
```

If the LUN access control method used is the Port-mode (use of <StoragePort> directives only in the model file), there is no need to use this command.

A model contains a list of storage systems to configure. The **stormodelctl** command checks the state of the storage systems in the **ClusterDB** before attempting to configure them.

```
stormodelctl -c applymodel -m <model>
```

Warning:

This operation destroys all the data stored in the storage systems listed in the model.

Important:

It may be necessary to wait several minutes for the completion of the command. No message will be displayed for the intermediate steps.

The administrator can exclude storage systems from the list (**-e** flag), or add storage systems (**-i** flag).

The **stormodelctl** command returns a configuration message in the following format:

```
<disk array name> : <message>
```

The output may be processed by the **dshbak** command to reformat the results.

The administrator must check the output of the command. If errors are reported for some disk arrays, detailed analysis is required to diagnose and resolve the problem. The **stormodelctl** command can then be used to apply selectively the model on the disk arrays that have not been configured, using the **–i** flag.

The **–v** flag provides a verbose output, giving better control of the operations performed on the storage system.

The command only transmits the configuration information to the target storage systems. LUN formatting is a background task. To control the formatting process, use the **checkformat** sub-command:

```
stormodelctl -c checkformat -m <model>
```

Important:

Wait for the command to complete before running the next step.

Please refer to the help of the **stormodelctl** command for additional options.

9.7.3.4 STEP 4 - Updating the Configuration of the I/O Nodes

The storage configuration model can also be used to configure the I/O nodes in order to simplify Lustre deployment. This service can be used even if the model has not been used to configure the storage systems.

The configuration of the I/O nodes is performed by the **stordepmap** command.

```
stordepmap -m <model> -l root
```

Important:

The command scans the disk on each node but no message is displayed for the intermediate steps. The command may take a long time to complete.

Warning:

ClusterDB is used to retrieve the I/O nodes attached to the storage systems referenced in the model. The state of the I/O nodes is also checked before the command attempts to configure the node.

Do not run **stordepmap** if Lustre is running.

The **stordepmap** command performs the following operations:

- Transmits configuration information to the nodes, using **ssh.**
- Forces the node to rescan the attached storage resources.
- Checks that the expected devices have been detected.
- Recreates symbolic links to easily identify the disks.

Some of these actions require root access rights on the target I/O nodes, while administrators usually are not root users on the Management Node. The operations on the target node are performed through **ssh**.

Note:

ssh must be configured to run without the need for passwords.

Errors are reported in the output of the command. If errors are reported for some nodes, detailed analysis will then be required to diagnose and resolve the problem. The **stordepmap** command can then be used to apply the model selectively on the nodes that have not been configured, using the **–i** flag.

Note:

When a node is re-installed (for example with **KSIS**), the I/O configuration must be rebuilt after the installation using the following command:

```
stordepmap -m <model> -i <reinstalled_node>
```

Please refer to the help of the **stordepmap** command for additional options.

9.7.3.5 STEP 5 - Transmitting OST/MDT Information to Lustre

The purpose of this phase is to provide Lustre with the inventory of the **OST**s and **MDT**s which are available for Lustre usage on each I/O node. This information is transmitted to Lustre by the **stormodelctl** command:

```
stormodelctl -c generateost -m <model>

stormodelctl -c generatemdt -m <model>
```

Note:

The storage model may not contain all the OSTs or MDTs. For example, **MDT** may use the internal **SCSI** disks of a node. The **stormodelctl** command transmits information only for the defined OSTs and MDTs. Refer to the Lustre configuration documentation to understand how to declare and configure other nodes used by Lustre.

Refer to the *Parallel File Systems* chapter to understand how to clean OSTs and MDTs which are no longer used, and how to update the Lustre file system configuration and restart the Lustre file system(s).

9.7.4 Alternate Storage Configuration Process for Lustre I/O Nodes

The process can apply to any kind of storage system. Compared to the deployment service for storage systems, it is less deterministic, uses fewer controls, but is more flexible and easier to use when the cluster has a limited number of storage systems.

Nevertheless, this process ensures the storage management prerequisites for Lustre:

- Persistent naming of storage devices.
- Same device name for a LUN on each node of an HA pair.
- Simplify device inventory for Lustre.

Furthermore, this process can be done locally on I/O nodes, and does not require a management station.

9.7.4.1 STEP 1 – Configuring the Storage Systems

The administrator can use either the device specific commands installed on the cluster Management Node, such as **nec_admin** or **ddn_admin**, or the tools of each storage device.

9.7.4.2 STEP 2 – Updating the I/O Node's Configuration

The I/O node configuration is modified using the **stordiskname** and **stormap** commands.

The **stordiskname** will assure the unique and persistent symbolic naming of devices for a "single" node. Moreover for a pair of HA nodes, it will also guarantee that symbolic naming will be the same for both nodes of each device that is reachable from both sides. If a device is restricted to one node, meaning that it is reachable only from one node in the High Availability pair, then there is no symbolic naming created for this device. Information about symbolic naming is stored in a specific file.

The **stormap –c** command will simply create the symbolic link device naming for each device with regard to information given by the previous **stordiskname** command in the specific file. This link is used at each boot of the node so that the same naming is used.

Warning:

Do not run **stormap** if Lustre is running.

A configuration where there are several paths for the same LUN but no multipath solution installed is not supported. In this case, the **stordiskname** command returns the following error :

```
ERROR: -= This tool does not manage the configuration where a given UID
appears more than once on the node. A multipath solution is required =-
```

Note:

In the case of newly created devices on the Storage Subsystem, the **stordiskname -u** command can be used so that existing devices keep the same symbolic naming, and so that new device symbolic names are added to the existing ones.

The **-u** option must be used in exactly the same way as the **–c** option with regard to the "**single node**", "**HA pair**", "**management station**"... configurations. See the different cases above with the **-c** option.
The **-u** option must be also used if any maintenance operation leads to changes for access to the devices (such as changing controller on the Storage Subsystem, or changing connections). Option **-u** will update the new World Wide Port Names for access to the devices, keeping all devices symbolic names as they were before the maintenance operation.
Please refer to the **stordiskname** and **stormap** commands help files for additional information on other options.

The **stordiskname** and **stormap** commands must be used as follows.

If the node is a "single node", meaning the node is not in a HA pair:

- Either from the management station (or "centralized node if no management station"):

```
stordiskname -c -r <node_name>
```

Once the operation has finished:

```
ssh root@<node_name> "stormap -c"
```

where **<node_name>** is the IP name of the target node.

- Or locally on the node:

```
stordiskname -c
```

Once the operation has finished:

```
stormap -c
```

If the node is in a HA pair:

- Either from the management station (or "centralized node if no management station"):

```
stordiskname -c -r <node1_name>,<node2_name>
```

Once the operation has finished, run either:

```
ssh root@<node1_name> "stormap -c"
ssh root@<node2_name> "stormap -c"
```

where **<node1_name>** is the name of one node in the HA pair, and **<node2_name>** is the name of the other node in the HA pair.

Or:

```
pdsh -w <node1_name>,<node2_name> "stormap -c" | dshbak -c
```

- OR locally on one of the nodes in the HA pair:

Note:

All I/O nodes must be able to connect to each other using **ssh,** this means that the **RSA** keys must have been installed on all the nodes.

```
stordiskname -c -n <node_name>
```

Once the operation has finished:

```
stormap -c

ssh root@<node_name> "stormap -c"
```

where **<node_name>** is the name of the adjacent node in the HA pair.

Important

The **stordiskname** command builds a **/etc/storageadmin/disknaming.conf** file which contains information, including symbolic link names, the LUN UIDs and the LUN's WWPN access. Only the **stordiskname** command can create or modify this file regarding node specific information.

This file will be erased when redeploying the **ksis** reference image, or when the system is restored for a node.

Therefore, **stordiskname**, if used with the **–r** option (remote) from the Management Node, will enable backups and restorations of the **/etc/storageadmin/disknaming.conf** file to be managed automatically. It is highly recommended that this is done. If not used with the **–r** option, the administrator has to manage the backup of the **/etc/storageadmin/disknaming.conf** file himself.

When used remotely (**-r**), immediately following a **ksis** image re-deployment, or a node system restoration, the following commands must be used in order that the **LUN**s are addressed by the same symbolic link names as previously used before to avoid the need to reconfigure **Lustre**.

The **stordiskname** command should be executed from the Management Node using the **–u** (update) option as shown below:

For a node which is not in a High Availability pair

```
stordiskname -u -r <node_name>
```

For a node which is in a High Availability pair

```
stordiskname -u -r <node1_name>, <node2_name>
```

Note:
Include the **–m** mode option, if this was specified when the **stordiskname** was previously executed. This applies to both High Availability and non High Availability nodes.

Once the **disknaming.conf** file has been copied over, the symbolic links must be recreated using the information that it contains. Therefore, run the **stormap** command as described previously.

Note
If a node has been rebooted after the **disknaming.conf** file was copied over the symbolic links will have been created automatically at boot time, therefore there is no need to run **stormap** again.

9.7.4.3 STEP 3 – Use of I/O Node Configuration Information by Lustre

Please refer to the *Parallel File Systems* chapter in this manual. Lustre provides management commands to detect all the LUN devices, and identify them using their persistent names.

9.8 User Rights and Security Levels for the Storage Commands

9.8.1 Management Node

Situation 1: superuser (= root) user

All the storage commands are available but it is not recommended to launch any of them as the root user for obvious security reasons.

Situation 2: non root user

Nagios user: The storage views, like all the NSMASTER- HPC web pages, are only accessible for the Nagios user who is automatically created during the installation/configuration of the cluster – see Chapter 3 *Cluster Database Management* for more details.

Any specific security rules/access rights will have been applied to the storage commands. Therefore, the non root users, for example, admin, must be part of the **dba** group, and the Nagios supplementary group, in order to be able to launch storage commands.

For example:

```
useradd -g dba -G nagios <username>
```

Some of these **dba** restricted access commands must be used with the **sudo** command in order to have root privileges. The reason why this privilege is restricted is that these commands may access other nodes, in addition to the MANAGEMENT node, by using **ssh**, to get or set information.

The following commands must be launched with **sudo**:

- **iorefmgmt**
- **ioregister**
- **lsiodev**
- **lsiocfg**
- **stordepha**
- **storioha**
- **stordepmap**
- **stormap**
- **stormodelctl**

Notes:

- **sudo** is a standard linux command. It allows a permitted user/group to execute a command as the superuser or as another user, as specified in the **/etc/sudoers** file which is managed by the superuser only. This file contains a list of groups/commands which have these root privileges. Refer to the **sudo** man pages for more information. To use a command with **sudo**, the command has to be prefixed by the word 'sudo' as in the following example:
  `<prompt>: sudo /usr/sbin/iorefmgmt`

- The PATH of the **dba** "username" must be completed in order to access these root commands without the absolute PATH in the **sudo** command:
  `export PATH=$PATH:/usr/sbin` in the **$HOME/.bashrc** of login "username"
  The sudo command is:
  `<prompt>: sudo iorefmgmt`

9.8.2 Other Node Types

All the available storage commands can only be launched as the root user, without exception.

9.8.3 Configuration Files

All 3 configuration files, which an administrative user of the **dba** group can modify manually, are located in the **/etc/storageadmin/** directory of the management node.

The 3 files are:

ddn_admin.conf	Specific to a DDN configuration
nec_admin.conf	Specific to a NEC configuration
storframework.conf	General configuration file for storage management

Chapter 10. Kerberos - Network Authentication Protocol

Kerberos is a security suite product which can be used to validate the identity of users, services and machines for a whole network.

The use of this product is optional, nevertheless all the packages required for its use are installed by default.

The purpose of this chapter is to describe how to implement Kerberos on a HPC cluster.

10.1 Environment

10.1.1 Kerberos Infrastructure

The Network Authentication Protocol Kerberos is included in the Bull BAS delivery and distribution.

There exist 3 types of machine within the Kerberos infrastructure:

- The Kerberos server with the Key Distribution Centre (**KDC**) keys server and administration server housed on a server called **secu0** (by default, for a HPC cluster, this will be part of the Management Node).
- The servers containing one or more applications which are protected by Kerberos; these servers are named **secui**. A Kerberos configuration file is shared with the Kerberos server.
- The Kerberos client machines. These are not used until Kerberos authenticates the user's rights to access the applications on **secui** and to **secu0**.

10.1.2 Validating the Installation

The settings made by Kerberos will be validated by the telnet access authentication. The server **secu1** hosts the remote telnet service. The telnet connection to **secu1** will be made by **secu0**.

10.1.3 Authentication of the SSH V2 Connections

The remote service SSH will be installed on **secu1** with a connection to a SSH client from **secu0**.

10.2 KERBEROS Infrastructure Configuration

10.2.1 secu0 Server including KDC Server and Administration Server

Verify the installation of the latest version (1.3.4-9) of the Kerberos **RPM** on **secu0.**

Important:

For security reasons, the Kerberos package is compiled with the option **-without-krb4** to prohibit compatibility with Kerberos 4.

10.2.2 Configuration Files

/etc/krb5.conf

This file containing the details of the KDC addresses and the administration server will be recopied on all the servers containing kerberized applications as well as on all the client machines.

```
...
[libdefaults]
default_realm = DOMAIN.COM
...

[realms]
DOMAIN.COM ={  kdc=secu0:88
               admin_server = secu0:749
               default.domain=domain.com
               }

[domain.realm] .domain.com=DOMAIN.COM
               domain.com=DOMAIN.COM
...
```

/var/kerberos/krb5kdc/kdc.conf

This file, containing amongst other things the information necessary to produce the tickets, is specific to the Kerberos server.

```
...
[realms]
DOMAIN.COM={
preauth=yes
max_life= 2d
max_renewable_life= 10d
...
}
```

10.2.3 Creating the Kerberos Database

Use the following command the Kerberos database.

```
/usr/kerberos/sbin/kdb5_util create -s
enter KDC database master key : XXXX
```

10.2.4 Creating the Kerberos Administrator

The KDC server may be administered from any network machine using the command **kadmin** as long as the user's identity is authenticated.

As the Kerberos administrator node does not exist initially it is possible to connect the first time using root with the command **kadmin.local** on the KDC server. It is not possible to authenticate oneself with this command as one is logged onto the KDC server.

```
/usr/kerberos/sbin/kadmin.local
kadmin.local : addprinc user/admin
enter PW   : YYYY
```

Now one will be able to authenticate oneself as «user» from any Kerberos client machine (the account Unix «user» must have been created as above) in order to connect to the administrator server, and to manage Kerberos assuming the admin demon has been launched. See below for more details.

Important:

The Kerberos administrators which have been created – «user» in the example above - must belong to the root group in order to be able to reach and modify Kerberos files.

10.2.5 Starting the KDC Server

Use the following command to start the KDC server.

```
/sbin/service krb5kdc start
```

Verifying the local connection to Kerberos on the KDC server using «user»'s administrator access rights:

```
/usr/kerberos/bin/kinit  user/admin
          kinit(V5) : Cannot resolve network address for KDC in requested realm
          while getting initial credentials
```

The problem in the above message is one of confirming «user»'s credentials and should be resolved by replacing **secu0** by its IP address in the **krb5.conf** file.

```
/usr/kerberos/bin/kinit  user/admin
PW : YYYY
```

If there is no error message then everything is OK and the administrator «user» will obtain a Ticket-Granting Ticket (**TGT**).

10.2.6 Adding Access Control List (ACL) Rights for the Kerberos Administrator Created

In the file **/var/kerberos/krb5kdc/kadm5.acl**, add the line by:

```
user/admin  @DOMAIN.COM *
```

10.2.7 Starting the Administration Daemon

Use the following command to start the administration daemon.

```
/sbin/service  kadmin  start
```

It should now be possible to connect to the system and to administer the KDC server with a view to specifying the principals. A principal is an entity in the Kerberos realm – every user, instance and service in the Kerberos realm has a designated principal. The creation of principals can be done from any network machine using – in the example above - the administrator's access rights for user/admin.

10.2.8 Creating Principals Associated with Users

The Kerberos Administrator will create on the KDC server the principals associated with users. These users must have associated UNIX accounts existing on the client machines.

The Kerberos Administrator can create the principals either:

- Locally on the KDC (using the command **kadmin.local**) without needing to authenticate himself.

or

- From another network machine (for example a client machine) using the command **kadmin** as long as he has authenticated himself as a principal with administration rights and the administration demon has been launched. For example, for user `Durand`:

```
kadmin.local
PW : YYYY
kadmin : addprinc durand
PW : ZZZZ  (add the user password on the client machines)
Principal " durand@DOMAIN.COM " created
```

For a principal user the secret key shared between the KDC and the client machine is derived from the user's password.

The process has to be repeated for all other users.

10.2.9 Creating Principals Associated with Remote Kerberized Services

The principals associated with services have to be created. However, MIT provides some basic services which have already been kerberized in their Kerberos distribution. For the **ftp, telnet,** and **rsh** services (included as part of the default installation using the package krb5-workstation), the principal associated with them generic and is called **host principal**.

This **host principal,** whose name is derived from the name of the machine, can be used for Kerberos Authentication of the basic kerberized services - rlogin, telnet, etc. residing on this host.

Creation of the host principal for the server **secu1**:

Connect to **secu0** or to **secu1** with the **kadmin** command.

```
kadmin.local
addprinc  -randkey   host/secu1.domain.com
```

Important:

The hostname has to correspond with its first occurrence in the line associated with the machine in the file **/etc/hosts**.

10.3 Configuring the secu1 Machine Hosting the Remote Service 'host principal'

Verify the installation of the latest version (1.3.4-9) of the Kerberos RPMs on **secu1**.

Copy the configuration file **/etc/krb5.conf** from **secu0** to **secu1,** and to any other machines which may be on the system.

10.3.1 Generating the Key Associated with the Remote Service 'host principal'

This secret key is shared between the KDC server **secu0** and the server housing the remote service **secu1**. This is necessary so that **secu1** can decipher the Kerberos tickets which are transmitted to it. The key can be created on any of these 2 servers but must then be copied from one to the other.

Important:

The file for the keys defined in the configuration file **kdc.conf** is as follows:

```
/var/kerberos/krb5kdc/kadm5.keytab
```

The file for the keys used by the command **kadmin** is as follows:

```
/etc/krb5.keytab
```

Therefore the name of the file has to be modified in the **kdc.conf** file or a link has to be made between the files as follows:

```
ln -s /var/kerberos/krb5kdc/kadm5.keytab  /etc/krb5.keytab
```

Connect as a Kerberos administrator («user») to **secu0:**

```
kadmin
ktadd  host/secu1.domain.com
```

Then recopy the key to **secu1**.

Important:

When working it is recommended to have a keytab file for each service and to store on each server only the keys associated with the remote services hosted on the server and not the keys whose services are hosted by other servers. However, the KDC server must of course have its own specific keytab file for all the keys of the remote services

10.4 Validating Kerberos Authentication for the Telnet Service

Command to create the **TGT** ticket for a user connected as `Durand` on **secu0**:

```
kinit
PW : xxxx  (password user durand)
klist
....
```

Launching a telnet server

telnetd has to be launched on **secu1**.

For the command:

```
/etc/xinetd.d/krb5-telnet
```

Add the arguments **server_args = -a user** to force Kerberos Authentication when the client connects using telnet. Set **disable = no** to launch the **krb5-telnet** demon when next starting **xinetd.**

It is also possible to set **disable = yes** in the file **/etc/xinetd.d/krb5-telnet** to prevent telnet from starting.

Restart **xinetd** using the command:

```
/etc/rc.d/init.d/xinetd restart
```

Using the command **chkconfig --list** verify that the daemon **krb5-telnet** is being used and not the usual demon **krb5**.

To test the telnet connection from the client machine client **secu0** to the server **secu1** hosting the telnet service for the user Durand on **secu0**, use the following command:

```
telnet -a secu1
```

The connection has to be confirmed – without a password being provided – by the following message:

```
Kerberos accepts you as  " durand@DOMAIN.COM "
```

If a password is used then a **Keberos** Authentication is not made and the password will be available across the network.

10.5 Kerberos Authentication and SSH

The remote service SSH is installed on **secu1** with a SSH client connection from **secu0**.

10.5.1 Configuring the Server SSH on the Machine secu1

A typical **sshd_config** configuration file will contain the following:

```
#         $OpenBSD: sshd_config,v 1.69 2004/05/23 23:59:53 dtucker Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.

Port 22
Protocol 2
ListenAddress xxx.xxx.xxx.xxx
#ListenAddress ::

# HostKey for protocol version 1
#HostKey /usr/local/etc/ssh_host_key
# HostKeys for protocol version 2
#HostKey /usr/local/etc/ssh_host_rsa_key
#HostKey /usr/local/etc/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 1h
#ServerKeyBits 768

# Logging
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
#LogLevel VERBOSE

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6

RSAAuthentication no
PubkeyAuthentication no
```

```
#AuthorizedKeysFile  .ssh/authorized_keys

# For this to work you will also need host keys in /usr/local/etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
PermitEmptyPasswords no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes

# Kerberos options
KerberosAuthentication yes
# If the Kerberos authentication is denied, an Authentication password is not
# provided for the user :
KerberosOrLocalPasswd no
KerberosTicketCleanup yes

# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication mechanism.
# Depending on your PAM configuration, this may bypass the setting of
# PasswordAuthentication, PermitEmptyPasswords, and
# "PermitRootLogin without-password". If you just want the PAM account and
# session checks to run without PAM authentication, then enable this but set
# ChallengeResponseAuthentication=no
UsePAM yes

#AllowTcpForwarding yes
#GatewayPorts no
#X11Forwarding no
#X11DisplayOffset 10
#X11UseLocalhost yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
```

```
#UsePrivilegeSeparation yes
#PermitUserEnvironment no
#Compression yes
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS yes
#PidFile /var/run/sshd.pid
#MaxStartups 10

# no default banner path
#Banner /some/path

# override default of no subsystems
Subsystem  sftp /usr/local/libexec/sftp-server
```

Important:

The following pre-requisites apply:

- The file **/etc/hosts** of the remote machine with which the ssh connection is being made has to have its hostname in the form:
  `x.x.x.x  secu1.domain.com secu1`
- The hostname of the remote machine may be of the form:
  `secu1.domain.com` or `secu1`.
- The principal service associated with this machine has to be the same as its Fully Qualified Domain Name **FQDN**:
  `secu1.domain.com`.

10.5.2 SSH Client

On the machine **secu0** or another machine a typical **ssh_config** file appears as follows:

```
#         $OpenBSD: ssh_config,v 1.19 2003/08/13 08:46:31 markus Exp $

# This is the ssh client system-wide configuration file.  See
# ssh_config(5) for more information.  This file provides defaults for
# users, and the values can be changed in per-user configuration files
# or on the command line.

# Configuration data is parsed as follows:
#  1. command line options
#  2. user-specific file
#  3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.
```

```
# Site-wide defaults for various options

# Host *
#   ForwardAgent no
#   ForwardX11 no
RhostsRSAAuthentication no
RSAAuthentication no
PasswordAuthentication no
HostbasedAuthentication no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-
        cbc,aes256-cbc
#   EscapeChar ~
Port 22
Protocol 2
GSSAPIAuthentication yes

# Pour le forwarding des tickets :
GSSAPIDelegateCredentials yes
```

Note:

The forwarding of TGT tickets by **ssh** is activated by the parameter **GSSAPIDelegateCredentials yes** for the ssh client file.

10.6 Troubleshooting Errors

```
Error : " Permission denied (gssapi-with-mic,keyboard-interactive) "
```

There are several possible causes for this error:

- The target machine must have its **full name** in its **/etc/hosts** file as shown below:

```
@IP        secu1.domain.com secu1
```

If several names are associated with the same IP address the name with which one is connecting has to be at the top of **/etc/hosts** as shown below:

```
          @IP  parallel.domain.com  parallel
          @IP  secu1.domain.com  secu1
```

- Verify that the file **/etc/krb5.conf** is identical on the KDC server and the SSH servers and clients.
- Verify that the keys in the file **/etc/krb5.keytab** are identical on the KDC server and the SSH server.
- Verify that the user has a valid TGT ticket.

10.7 Using Kerberos and RMS

RMS does not carry out a Kerberos Authentication but can ensure the forwarding of tickets, and if necessary their renewal. For this to happen, it is necessary to configure on all cluster machines the RMS Authentication library to be used with Kerberos.

This library is identified as follows; **/usr/lib/librmsauth.so**

By default there is a symbolic link which points to a library stub as follows: **librmsauth_null.so**

The activation of the Kerberos Authentication consists simply of making a link with the library **librmsauth_krb5.so**

RMS has then to be restarted.

To check that everything is working correctly, simply launch a RMS job. This will post the tickets available as shown in the example below – with the addresses highlighted:

```
prun -t -k -n2 klist
0 klist: You have no tickets cached
0 Ticket cache:
0 Default principal: qx@DOMAIN.COM
0
0 Valid starting     Expires            Service principal
0 10/06/05 10:54:17  10/07/05 10:31:25  krbtgt/DOMAIN.COM@DOMAIN.COM
0       renew until 10/06/05 10:31:25
0
0
0 Kerberos 4 ticket cache: /tmp/tkt557
1 klist: You have no tickets cached
1 Ticket cache:
1 Default principal: qx@DOMAIN.COM
1
1 Valid starting     Expires            Service principal
1 10/06/05 10:54:17  10/07/05 10:31:25  krbtgt/DOMAIN.COM@DOMAIN.COM
1       renew until 10/06/05 10:31:25
1
1
1 Kerberos 4 ticket cache: /tmp/tkt557
```

To turn off Kerberos Authentication, it is necessary to restore the links to the stub library and to relaunch **RMS**.

10.8 Generating Associated Keys for Nodes of a Cluster

The Perl program, below, generates on the Kerberos server, hosted on the management node, the Kerberos key (keytab) associated with each node and then transfers it to the node using Secure Copy (**SCP**) security (confidentiality and Authentication ensured by private key / public key).

The pre-requisite here is the preliminary installation of a private key / private key infrastructure between the management node and each compute node thus ensuring the secure transfer of the Kerberos keys.

The file **/etc/ssh/ssh_config_public** is the configuration file for the SSH client which uses Authentication by the public key/private key protocol.

```
#!/usr/bin/perl -w

print "Lower limit of cluster nodes: ";
$inf = <STDIN>;
chomp ($inf);
print "Upper limit of cluster nodes: ";
$sup = <STDIN>;
chomp ($sup);

my $serv="secu";
my $serv0="secu";
my $keytab="_keytab";
my $krb5_keytab=":/etc/krb5.keytab";

# Key creation for each node of the cluster
# Each key is generated on the management node and is stored in a temporary
# file (and also in the KDC base) ; this file will then be recopied on the
# associated node;
#  The remote recopy by SCP will be secured by public key/ private key.

for ($i=$inf; $i <=$sup; $i++) {
        $serv="$serv0$i";
        print("Création keytab pour $serv\n");
        system ("rm -f /tmp/$serv$keytab");
        system ("kadmin.local -q 'ktadd -k /tmp/$serv$keytab
          host/$serv.domain.com'");
        system ("scp -rp /tmp/$serv$keytab $serv$krb5_keytab");
        system ("rm -f /tmp/$serv$keytab");
}

print("\n-----> The new keys for the nodes secu$inf to secu$sup have been
          generated \n\n");
```

10.9 Modifying the Lifespan and Renewal Period for TGT Tickets

The default the duration for a Ticket-Granting Ticket (**TGT**) ticket is 10 hours and it cannot be renewed while it is still active. In other words its duration must be at 0 before it is renewed.

These two time periods can be modified by a user.

For example the command below is used to change the duration of a ticket to 2 days and its renewal period to 5 days.

```
kinit  -l 2d  -r 5d
```

The ticket obtained with this command will be valid for 2 days and may be renewed at any time during these 2 days in order to obtain a new ticket which is also valid for 2 days up until the 5 day limit is reached.

However, the values specified by the user have to be inside the maximum values defined by the Kerberos configuration.

To modify these values in the Kerberos configuration file **/var/kerberos/krb5kdc/kdc.conf** do as follows:
In the block `[realms]`, add:

```
max_life = 2d
max_renewable_life = 10d
```

Then relaunch the **krb5kdc** and **kadmin** daemons.

- To authorize the creative entity principal for **krbtgt** tickets to deliver tickets with the values above, use the command below:

```
kadmin
modprinc -maxlife "2 days"  krbtgt/DOMAIN.COM
modprinc -maxrenewlife "10 days"  krbtgt/DOMAIN.COM
```

- To authorize the values for the user concerned use the command below:

```
kadmin
modprinc -maxlife "2 days"  durand
modprinc -maxrenewlife "10 days"  durand
```

10.10 Including Addresses with Tickets

By default tickets do not include addresses.

Use the following command in order that tickets generated include the addresses of the local machine.

```
add noaddresses=no in the paragraph [libdefaults] for the file
/etc/krb5.conf
```

10.11 Using Kerberos in High Availability Mode

To use Kerberos on a administration node with HA functioning the files **/etc/HA/start**, **/etc/HA/stop** and **/etc/HA/status** have to be edited, removing the comment status for the following lines:

```
#/etc/init.d/krb5kdc
#/etc/init.d/kadmin
```

After this the HA services have to be relaunched to take into account these modifications.

Note:

Before editing these three scripts, it is important Kerberos is activated so that the services can exploit Kerberos. Also, Kerberos should be deactivated after all the kerberized services have completed.

Chapter 11. Cluster High Availability

By High Availability we mean a protocol, and its associated execution, which ensures cluster-system operational continuity when there is a downtime event. A continuously available cluster system is characterized as having practically no downtime in any given year. The most desirable High Availability rate is known as 99.999% availability, a good deal of planning for High Availability centers around backup, failover processing, data storage and access.

This chapter explains the main concepts for implementing High Availability on a Bull HPC system.

11.1 Planning for High Availability

The key requirements of large-scale production Linux clusters are the need to provide a high level of reliability, the need to continue to operate if there is a failure, and the need to be able to make repairs to the system while it is in production.

As a part of ensuring cluster High Availability, different redundancy and recovery mechanisms are implemented for different domains:

- Management node High Availability – see Chapter 12
- I/O nodes and Lustre File-system High Availability – see Chapter 13.

The system must firstly detect the errors, then identify the specific component involved, and then correct them automatically. If the fault requires a reduction in the system's operating configuration whilst it is being fixed, then this should be done quickly and automatically. Normally, the following failure modes may be protected against:

Type of failure	Specific failure	Recovery Mechanism
Management Node Failures	• Management Node kernel panic/hang • Node hardware failure or power down	**Cluster Suite**
I/O Node and Metadata Failures	• I/O node panic/hang • I/O node power down • Lustre request to MDT/OST not acknowledged in time	
Storage Failures	• Fibre adapter errors • Disk subsystem failure • SCSI adapter error or hang	
Ethernet Network Failures	• Ethernet network access failure (NIC or link failure) • Management network failure	

Table 11-1. Possible Cluster component failures and HA Recovery Mechanism

Note:
The **QsNetII** product, from **Quadrics**, provides High Availability features for clusters that use this type of architecture. See the *Bull HPC BAS4 Maintenance Guide* for details.

Chapter 12. Management Node High Availability

This chapter explains how to implement Management Node High Availability using Cluster Suite and specific HA scripts. The following topics are described:

12.1 Introduction

One of the most fragile points of the cluster is the Management Node. All management services for a cluster running on the management node will become unavailable if the Management Node goes down. For a compute cluster job management, for example using **Quadrics RMS**, will run directly from the Management Node. Therefore, if the Management Node goes down it will no longer be possible to submit any compute jobs, making all the nodes unusable.

In order to avoid such a lost of functionality, it is necessary to physically double up the Management Nodes, and to put a recovery mechanism into place for the cluster management services on the backup node.

When a problem occurs on one of the Management nodes, the supported management applications running on that node rock over to the second one. High Availability of the cluster management is based on an architecture which includes two Management Nodes and shared disk space.

Important

When there is a switch of Management Node, the High Availability of the application program running on the cluster is not guaranteed. Only if the application program includes a means of continuing a task(s) which was/were in progress when the sudden interruption, and subsequent switch, occurred can High Availability for the application program be ensured.

For the management nodes which are made Highly Available, the node which is actually running the critical services is designated as the **Primary Node**. The node which is ready to take the place of the Primary Node should the Primary Node go down is designated as the **Back-up Node**.

The Primary node sends signals (**heart-beats**) through the network. If a software or hardware breakdown occurs, the High Availability software takes the actions necessary to maintain the availability of the services, and the integrity of the data, whilst rocking (failover) over to the Back-up node in a transparent way.

When the backup Back-up node takes charge, the High Availability software ensures that the first Primary node "does not awake" suddenly, forcing the backup Back-up node to shutdown as a result of a power switch. When the backup Back-up node is certain that the failing Primary node is down, it will take charge of the services from this node.

It is possible that the node which has failed may retake charge of its functions after a reboot, or an intervention of an operator. This High Availability feature makes it possible for an operator to rock the services to the backup Back-up node (service of relocating) in order to intervene at the time of the maintenance actions.

The High Availability management tool is **Cluster Suite.** A heart-beat mechanism detects if the Management Nodes are running. If the Primary node is found to be unattainable then a **fencing** mechanism is activated, this powers off the failing node. Then the **Back-up node** becomes the **Primary node**.

Within an HPC Cluster the different constituent elements, for example Compute Nodes, I/O nodes, network equipment, and so on, have to address a **virtual management node.** This virtual management node is in reality the Primary node. This virtualization of the management node is implemented by the mechanisms described within this section.

The elements which allow the administrator to ensure that **High Availability** is in place on the Management Nodes for a cluster are described in this chapter.

Note:

The concept of High Availability applies to many different functionalities of the Management Node. This chapter is only concerned with protecting critical services using High Availability.

12.1.1 Possible Management Node failures

Management Node kernel panic/hang

When a node hangs or encounters a panic, it does not send its heartbeat messages within the authorized period. This silence is detected by the High Availability peer node which fences the silent node, and takes over the cluster services when the fencing is completed.

Node Hardware Failure or Power Down

The node does not send its heartbeat messages in the authorized period. This silence is detected by the High Availability peer node, which fences the silent node and takes over the cluster services when the fencing is completed.

12.2 Configuration

12.2.1 Hardware

Management Node services must be rapidly made operational again, when a Management Node goes down. For this to happen, the hardware elements must be doubled up. In order to avoid data synchronization problems between the 2 Management Nodes, a rack of disks is shared between them. This rack is designed to be tolerant of breakdowns and ruptures.

At the Ethernet connection level there are 2 network interfaces which are seen as single interface. The electric power supply of the two Management Nodes may be controlled remotely using the Ethernet network.

Typical hardware structures are shown below.

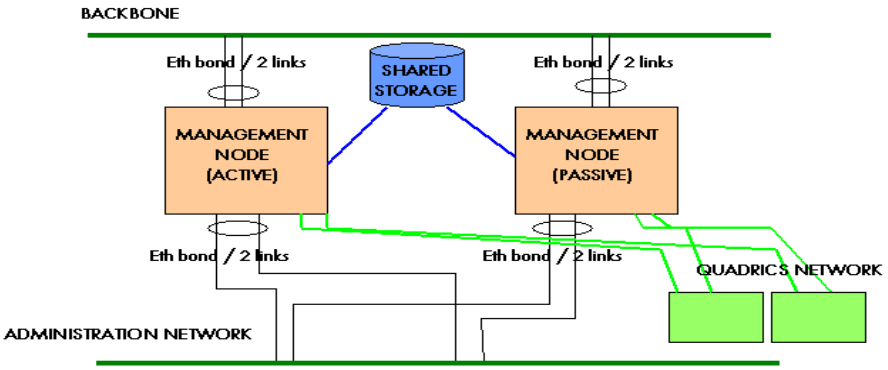


Figure 12-1. High Availability Management Node configuration

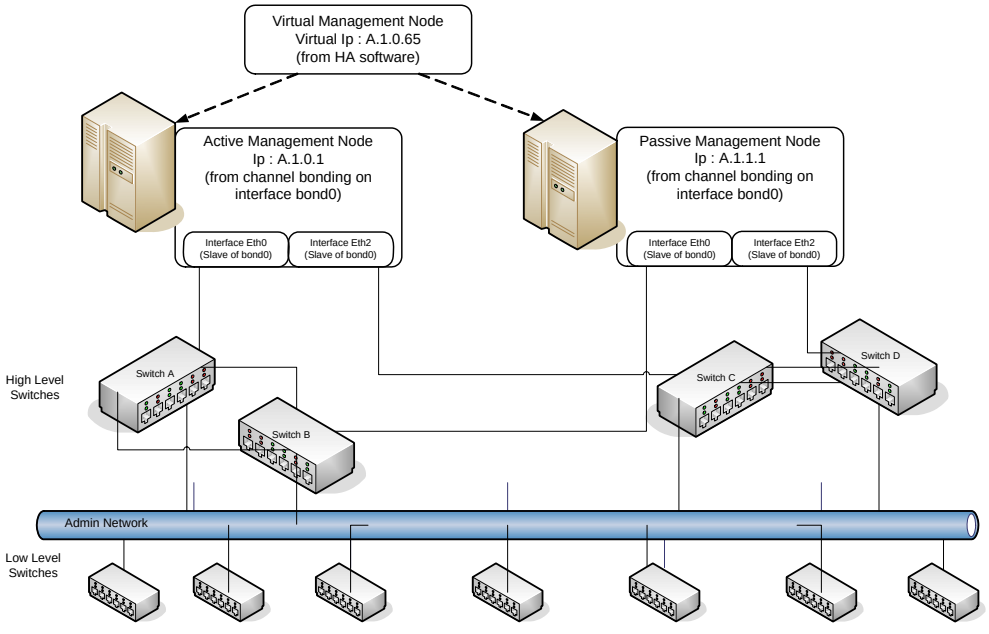


Figure 12-2. High Availability management network configuration

Each service will run on only one of the Management Nodes, which are contained within the High Availability system, at a time.

In order to distinguish the Management Nodes one is called the Primary node and the other is called the Secondary node. The one on which the services are running will be in active mode, and the other one (this may also be known as the backup node) will be in passive mode, ready to step in if the active node breaks down.

The primary node is the node which appears first alphabetically.

The primary node may also be defined as follows:

- The only management node available on the cluster that makes it possible to meet all the management functional needs.
- The node on which modifications are made, these are then propagated or replicated on the other management nodes.

12.2.2 Software

High Availability is implemented using **Cluster Suite**. Therefore Cluster Suite has to be configured, and the system adapted, so that it runs correctly.

A Linux distribution has to be installed on all the management nodes, as well as all the applications needed for smooth operation of the cluster.

In addition, scripts have been developed to enable **Cluster Suite** to implement High Availability on the management nodes. These scripts manage the High Availability of some services, by for example starting, stopping or changing the state of a service.

Important:
Some services include the High Availability functionality directly, for example SLURM, others allow a synchronization mechanism to run. These services are not covered by this section of the manual.

For each service which is to be made highly available it will be necessary to take an inventory of:

- Which files and directories have to be present on both the active and passive nodes (**/etc/hosts, /etc/sshd,** etc.) and that can be modified for configuration purpose, although, this is seldom necessary.
- Which files and directories may be subject to change and so difficult to synchronize. This data should be placed on the shared discs which are common to both management nodes (**/etc/httpd**, etc.)
- Which files and directories do not change (**/usr/sbin/httpd**, etc.) and so do not need any special administration.

Example of how to configure High Availability for a HTTP server

Prerequisites:

- 2 nodes with separate IP addresses.
- Both nodes are exactly the same in functional terms, both from a software and system point of view. There may be differences for the MAC addresses and other hardware references.
- Each node has a ***power on/power off*** mechanism.
- A virtual IP address exists.
- All the HTTP server files are in the **/var/www** directory.
- The HTTP service is managed by **/etc/init.d/httpd**.
- The **/dev/sdb** device (visible from both nodes) corresponds to a storage rack shared between both nodes, and the partition **/dev/sdb1** is free and large enough to contain **/var/www**.

Implementing High Availability for the HTTP service consists of:

- Stopping the http service on both nodes and removing it from the boot start up.
- Initializing the **/dev/sdb1** partition, and then recopying the contents of **/var/www** to the **/dev/sdb1** partition.

To implement High Availability on the management nodes of a cluster, a list of all the services to be made highly available should be drawn up. As will be seen later, it has been decided that **Cluster Suite** will manage only one service, **haservices** which manages all the High Availability elements and services.

See:
The **Cluster Suite** documentation for detailed information.

12.2.3 Network

The applications running on the compute nodes have to communicate with the management applications. Most of the time client applications communicate with a host which is identified by its IP address. However, when there is a failover (the existing active node stops being the active one and the passive one becomes active) the new active node must be as accessible as the previous one. Therefore the IP address has to follow the active node. Nevertheless, both management nodes must have a proper IP address so that they can be reachable separately.

Accordingly, each management node has an IP address to interface with the management network (for example, 10.0.0.1/24 for the primary node, and 10.0.0.2/24 for the secondary node). In effect, when the High Availability service is active on the primary node the interface with the management node is a virtual IP address (for example, 10.0.0.250/24), virtual in the sense that it does not identify the primary node but the node which is active. This is more a functional resource than a physical one.

When the Active Node is the Primary Node

- The management interface with the primary node has two IP addresses (10.0.0.1/24 and 10.0.0.250/24).
- The management interface with the secondary node has one IP address (10.0.0.2/24).
- All cluster components can connect with the primary node using its real IP address (10.0.0.1/24) and its virtual IP address (10.0.0.250/24).
- The secondary node can only be accessed by using its real IP address (10.0.0.2/24).
- The active node is accessed using the virtual IP address (10.0.0.250/24).

When the Active Node is the Secondary Node

- The management interface with the primary node has one IP address (10.0.0.1/24).
- The management interface with the secondary node has two IP addresses (10.0.0.2/24 and 10.0.0.250/24).
- The primary node can only be accessed by using its real IP address (10.0.0.1/24).
- All cluster components can connect with the secondary node using its real IP address (10.0.0.2/24) and its virtual IP address (10.0.0.250/24).
- The active node is accessed using the virtual IP address (10.0.0.250/24).

12.3 Understanding High Availability Scripts for the Management Node

The **/etc/HA** directory contains all the tools which enable High Availability to be implemented and which are also used to manage the critical services which have been made highly available. It also contains the configuration files for these High Availability tools.

When Cluster Suite is configured, the **/etc/HA/bin/haservices** script is defined as the main service. This script is the entrance point for implementing High Availability for the cluster management services.

All the High Availability scripts used on the management node are in the **/etc/HA/bin** directory.

The list of scripts is as follows:

cs	Starts all the Cluster Suite services.
dbmConfig	Updates files according to the ClusterDB status.
haadminnodes	Lists the management nodes contained in the database.
hacisco	Removes the ARP tables for the switches included in the database.
hacron	De-activates/activates the crons specified in the **/etc/cron.d** file.
hadown	Simulates a script which always returns the same result.
hafake	Simulates a service.
haip	Handles (Manages) the virtual IP addresses associated with the active node.
halinks	Manages the dynamic links for the directories which have to be on a shared storage system.
halsf	Encapsulates the LSF scripts.
hamount	Manages the mount points of the shared storage system
hapidcleaner	Removes the **.pid** file which may prevent some services from starting
hapostfix	Manages the postfix mailing service
harms	Manages all the services associated with **Quadrics RMS**
harpms	Checks that the list of installed RPMs is identical on each management node
haservices	Main script which controls all the HA services

hasynchro	Manages the synchronization of files between the management nodes
hasyslogng	Manages **syslog-ng**
hatstfs	Tests the writing capability for different systems of shared files
haunwantedfiles	Manages the services that must not be launched when a node is started
haupdatedb	Updates the database according to the active node

Some of these scripts are configurable using the files in the **/etc/HA** folder.

12.3.1 /etc/HA/bin/haservices script

This script has to be used within in Cluster Suite.

According to the parameter (**start**, **status** or **stop**), a range of commands, specified in the **/etc/HA/start**, **/etc/HA/stop** or **/etc/HA/status** data files will be executed.

Note:

Three different data files are necessary because the commands to start, stop or report the status are not the same, or are not executed in the same order.

Each file consists of a series of lines which include:

- Lines beginning with a hash symbol (#). These are comments.
- Empty lines. These are ignored.
- Lines which indicate a command and consist of the full path without any options or spaces.

Note:
Some of the example data files delivered in the BAS4 rpm will be different from the examples shown in the next three sections.

12.3.2 /etc/HA/start file

This file contains the list of the commands to be carried out when the **haservices** script is launched is run with the **start** parameter. The commands are carried out in the order listed in the **start** file. When the first error occurs, **haservices** displays an error message.

/etc/HA/start example file:

```
# What to do at starting time and in which order ?
# Programs to call with 'start' parameter

/etc/HA/bin/haunwantedfiles
/etc/HA/bin/haip
/etc/HA/bin/hamount
/etc/HA/bin/halinks
```

```
/etc/HA/bin/hapidcleaner

/etc/HA/bin/hasyslogng

/etc/init.d/postgresql
/etc/HA/bin/haupdatedb
/etc/HA/bin/dbmconfig

#/etc/HA/bin/hacisco

#/etc/init.d/krb5kdc
#/etc/init.d/kadmin

#/etc/HA/bin/hasynchro
/etc/init.d/snmptrapd
/etc/init.d/nagios
/etc/init.d/gmond
/etc/init.d/gmetad
/etc/init.d/dhcpd
/etc/init.d/ldap
/etc/init.d/httpd
/etc/init.d/conman
/etc/init.d/nfs
/etc/init.d/systemimager


/etc/init.d/qsnet
/etc/init.d/msqld
/etc/HA/bin/harms

/etc/init.d/saslauthd
/etc/init.d/cyrus-imapd
#/etc/HA/bin/hapostfix
```

12.3.3 /etc/HA/status file

This file contains the list of the commands to be carried out when the **haservices** script is launched with the **status** parameter. The commands are carried out in the order listed in the **status** file. When the first error occurs, **haservices** displays an error message.

/etc/HA/status example file:

```
# Which facility has to be monitored ?
# Programs to call with 'status' parameter

/etc/HA/bin/haip
/etc/HA/bin/hamount
/etc/HA/bin/hasyslogng
/etc/HA/bin/hatstfs

/etc/init.d/postgresql

#/etc/init.d/krb5kdc
#/etc/init.d/kadmin

/etc/init.d/snmptrapd
/etc/init.d/nagios
/etc/init.d/gmond
/etc/init.d/gmetad
/etc/init.d/dhcpd
/etc/init.d/ldap
/etc/init.d/httpd
```

```
/etc/init.d/conman

/etc/init.d/nfs
/etc/init.d/qsnet
/etc/init.d/msqld
/etc/HA/bin/harms

/etc/init.d/saslauthd
/etc/init.d/cyrus-imapd
#/etc/HA/bin/hapostfix

/etc/HA/bin/halsf
```

12.3.4 /etc/HA/stop file

This file contains the list of the commands to be carried out when the **haservices** script is launched with the **stop** parameter. The commands are carried out in the order given in the **stop** file. When the first error occurs, **haservices** displays an error message.

The starting order of the commands is important. In the **stop** file the commands are usually executed in reverse order to that of the start file.

/etc/HA/stop example file:

```
# What to do at stoping time and in which order ?
# Programs to call with 'stop' parameter

/etc/HA/bin/halsf

/etc/HA/bin/harms
/etc/init.d/msqld
/etc/init.d/qsnet

#/etc/HA/bin/hapostfix
/etc/init.d/saslauthd
/etc/init.d/cyrus-imapd

/etc/init.d/systemimager
/etc/init.d/nfs
/etc/init.d/conman
/etc/init.d/httpd
/etc/init.d/ldap
/etc/init.d/dhcpd
/etc/init.d/gmetad
/etc/init.d/gmond
/etc/init.d/nagios
/etc/init.d/snmptrapd

#/etc/init.d/krb5kdc
#/etc/init.d/kadmin

/etc/init.d/postgresql

/etc/HA/bin/hasyslogng
/etc/HA/bin/hamount
/etc/HA/bin/haip
```

12.3.5 hacron script

In the case of a call with the **start** option, for each file listed in the **/etc/HA/cron** directory, this script will move the corresponding file from the **/etc/cron.d** directory to the **/etc/cron.d/SLEEP** directory in order to deactivate the cron.

In the case of a call with the **stop** option, for each file listed in **/etc/HA/cron**, this script will move the corresponding file from the **/etc/cron.d/SLEE**P directory to the **/etc/cron.d** directory in order to re-activate this cron.

The **/etc/HA/cron** file ignores all empty lines; all comment lines starting with # are ignored. All other lines are considered as the names of files which are usually included in **/etc/cron.d**.

12.3.6 haip script

This script manages the virtual IP addresses.

In the case of a call with the **start** argument, for each relevant line in the **/etc/HA/ip** data file, the script will add the IP address specified to the interface indicated. Also, an **ARP** update is sent out in order to notify all hardware components linked to this local network that the virtual IP address is now associated with the network interface specified.

In the case of a call with the **stop** argument, for each relevant line in the **/etc/HA/ip** directory, the script withdraws the specified virtual IP address from the network interface indicated.

In the case of a call with the **status** argument, for each relevant line in the **/etc/HA/ip** directory, the script checks that the specified virtual IP address is associated with the network interface indicated. When the first error occurs, an integer different from zero is returned.

The **/etc/HA/ip** file can handle empty lines, all comment lines starting with # are ignored. All other lines are considered as specifying a network interface and a virtual IP address in the following format:

- tabulation
- name of the interface, for example eth0, eth2, bond0
- tabulation
- IP address:
  - 4 integers separated with a dot, for example : 10.0.0.50
  - a slash
  - an integer (netmask).

Example of 2 virtual addresses for 2 network interfaces

```
bond0  10.0.0.50/8
eth1 192.168.255.1/16
```

12.3.7 halinks script

This script makes it possible to manage the symbolic links for the directories which have to be included in the shared storage system. The corresponding data file is **/etc/HA/links**. Any blank line or one starting with # is ignored. Any other line is considered as specifying the complete path of a directory.

In the case of a call with the **start** or **status** argument, the script checks that each path specified in the **/etc/HA/links** file is a symbolic link. For the first error met, an error code different from zero is returned.

In the case of a call with the **repair** argument, the script checks that each path specified in the **/etc/HA/links** file is a symbolic link. If it is not the case, it renames the directory by suffixing the name with **.orig-<date>**. Then, the link is created pointing towards what should be on the shared storage system. For example, for a path starting with **var**, the link points towards a similar path in which **var** is replaced with **varha** (for example **/var/lib/pgsql** replaced with **/varha/lib/pgsql**). If the shared filesystem is mounted and the shared directory indicated by the symbolic link does not exist, then the script copies the original directory into the directory indicated by the symbolic link.

Important:

On the primary management node of the cluster, it is advisable to launch this script after the shared filesystems have been mounted with the **/etc/HA/bin/hamount start** command. To avoid any corruption problems, do not perform this operation while Cluster Suite is running.

On the secondary management node, it is advisable not to have mounted the shared filesystems and not to have Cluster Suite running.

In the case of a call with the **undo** argument, the script checks that each path specified in the **/etc/HA/links** file corresponds to a symbolic link. If it does, it deletes the link, and then renames it as the most recent directory **<link_name>.orig-<date>** in **<link_name>**. **To avoid any corruption problems, do not perform this operation while Cluster Suite is running.**

/etc/HA/links file example:

```
# directory placed on shared disk

/etc/httpd
/etc/nagios
/etc/openldap
/etc/snmp
/etc/systemimager

/usr/lib/clustmngt/rms/etc
/var/log/postgres

/usr/lib/nagios
/usr/share/nagios
/usr/lib/rms/msql
/var/lib/ganglia
/var/lib/ldap
/var/lib/pgsql
```

```
/var/lib/systemimager
/var/lib/imap
/var/log/conman
/var/log/httpd

/var/log/systemimager
/var/spool/nagios
/var/spool/imap
/var/www
/var/rms

/usr/share/lsf
```

12.3.8 hamount script

This script manages the mount/un-mount operations for filesystems, usually, those which are included in shared storage systems.

When Cluster Suite is running, do not run this script manually on any of the management nodes of the cluster in order to avoid data corruption.

The data file of this script is **/etc/HA/mnttab**. A valid line contains the following:

- A device name, beginning with **/dev/**: `/dev/sda, /dev/sda1, /dev/hdb, /dev/hdb3, /dev/sdm,` etc.
- One or more spaces.
- A directory path indicating the mount point for the specified device.

Note:

Do not use space or tab characters in the device and path names. The following regular expressions are valid -**^(/dev/.*?)\s+(.*?)\s*$** -:

Examples of valid lines:

```
/dev/sdj1              /etcha
/dev/sdj2              /usrha
/dev/sdj3              /tftpboot
/dev/sdj4              /homeha
/dev/sdp1              /varha
```

In the case of a call with the **start** argument, for each specified mount point, the script unmounts and mounts the filesystem.

In the case of a call with the **stop** argument, for each specified mount point, the script unmounts the filesystem.

In the case of a call with the **status** argument, for each mount point, the script checks that the filesystem is correctly mounted.

In order that the administrator properly understands the consequences of a call to **hamount** with the **stop** argument, the filesystem unmounting operations are indicated below:

- If the filesystem is easily unmounted, then the operation is successful.

- If not, the script searches for all the processes which are included in the tree structure, and sends a SIGSTOP to them.
- If necessary the above operation will be repeated several times.
- If this does not work then a SIGKILL is sent to all processes included in the tree structure.
- If necessary the above operation will be repeated several times.
- Finally, as a last measure, a forced unmount is run (lazy umount).

Note:

A forced unmount may not be stable.

12.3.9 hapidcleaner script

This script is used to clean the **pid** files which could disturb the launching of services.

Note:

A **pid** file is a file containing in the first line an integer referring to a process.

The data file is **/etc/HA/pid2clean**. All blank lines or those starting with # are ignored. All other lines are considered as specifying the complete path for a pid file.

In the case of a call with the **start** argument, for each path specified in the **/etc/HA/pid2clean** file, the script checks if the corresponding process is in the course of being executed. If not, then the script deletes the file.

/etc/HA/pid2clean example file:

```
# One filename per line. Files to remove at CS startup
/var/lib/pgsql/data/postmaster.pid
```

12.3.10 harms script

haservices manages the **Quadrics RMS** services via the **harms** script.

RMS services have to be deactivated manually using the **chkconfig** command to avoid problems when a node in a **Quadrics** cluster rocks from passive to active.

12.3.11 hasynchro script

Note:

This script can only be run from the active node and has to be launched manually as it will not start automatically.

This script enables the management of files which can only be synchronized manually. These files are rarely modified.

The corresponding data file is **/etc/HA/synchro**. All blank lines or those starting with # are ignored. All other lines are considered as specifying the complete path for a file.

In the case of a call with the **start** or **status** arguments, the script connects to the passive node checks each file specified in the **/etc/HA/synchro** file and returns a status different from zero when the first different file is encountered.

In the case of a call with the **repair** argument, the script connects to the passive nodes, and for each file different from that of the active node, it replaces the passive node file with the active node version.

/etc/HA/synchro file example:

```
# Which files or to be synchronized ?
# One fullpath per line

/etc/HA/ip
/etc/HA/links
/etc/HA/mnttab
/etc/HA/pid2clean
/etc/HA/start
/etc/HA/status
/etc/HA/stop
/etc/HA/synchro


/etc/conman.conf
/etc/conman-tpl.conf
#/etc/dhcpd.conf
/etc/dhcpd-tpl.conf
/etc/exports
/etc/genders
/etc/gmetad.conf
/etc/gmond.conf
#/etc/hosts
/etc/hosts-tpl
/etc/resolv.conf
/etc/ssh/ssh_config
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/storageadmin/ddn_admin.conf
/etc/storageadmin/nec_admin.conf
/etc/storageadmin/storframework.conf
/etc/sudoers
/etc/syslog-ng/syslog-ng.conf
```

12.3.12 hasyslogng script

The logs generated by the **hasyslogng** script running on the active node are saved in the **/varha/log** folder.

It should be noted that when the Management Node starts and is not in active mode **syslog-ng** saves its logs in the **/var/log** folder. This means that the Administrator has to be aware of the existence of these different folders (**/varha/log** and **/var/log**) and also of the Management Node mode in order to identify the location of the logs.

Further, some of the logs may be saved on the common storage sub systems using the symbolic link mechanism which is put into place using the **halinks** command. The decision to save the logs on the storage sub system is made on a case by case basis. Again the Administrator has to keep in mind that the **syslog-ng** logs may be stored in different places according to settings made on the Management Node and its mode (active or passive).

12.3.13 haunwantedfiles script

This script manages any problems for the services which are executed when the management node starts, and which are intended to be managed by Cluster Suite. For example, if it is decided that the **httpd** server should be highly available, then its data files must be on the shared storage system. In order to avoid data corruption problems on the shared storage system the httpd server should only run on the active node, and the service should not be launched when the management node starts.

This script is based on the configuration files of **/etc/HA/bin/haservices** (**/etc/HA/start**, **/etc/HA/stop** and **/etc/HA/status**).

In the case of a call with the **start** or **status** parameters, the **haunwanted** script checks that no existing links in the system start directories points towards a service managed by **haservices**.

In the case of a call with the **report** parameter, for each service managed by **haservices,** the script will check in each start directory **(/etc/rc*.d)** for the presence of a link pointing to the service. If a link exists, then the script creates the **HA_UNWANTED** directory in the **/etc/rc*.d** directory where the link may be found and places the link in this directory, and then runs the service with the **stop** argument.

In the case of a call with the **undo** parameter, all the links in the **/etc/rc*.d/HA_UNWANTED** directory are moved up one level in the tree structure.

Note:
The **haunwantedfiles** script works in verbose mode. In the event of a call with the **stop** parameter, several messages similar to "`service xx stop FAILED`" are issued. This is normal. An OK status should be returned when the execution is finished. If the script is run for a second time, the output should be "OK" immediately.

12.4 Understanding Channel Bonding (Optional)

Implementing High Availability on a management node and the use or not of Channel Bonding are completely independent. It is the responsibility of the cluster administrator to decide whether or not to use Channel Bonding, assuming that the cluster equipment allows this.

Note:
If Channel Bonding is to be used it must be setup before installing High Availability on the management nodes. See *HPC BAS4 Installation and Configuration Guide* for details.

In short, Channel Bonding makes it possible, when a machine is equipped with several Ethernet interfaces, to divide up the work load or to implement High Availability on the different Ethernet interfaces.

As an example, consider a node equipped with two Ethernet interfaces (**eth0** and **eth1**) both of which are connected to the same local network. Either one or the other of these two interfaces can use the same IP address without disturbing the operation of the equipment connected to this local network.

For more information about Channel bonding, see:
http://linux-ip.net/html/ether-bonding.html

12.5 Installing the Primary and Secondary Management Nodes

Important:

The IP addresses used will depend on the address plan for the system. Those used in this section are examples only.

In this section, we assume what follows:

- The Primary Management Node is called `ns0` and the Secondary Management Node is called `ns13`.
- The `ns0` IP address is `10.1.0.1`
- The `ns13` IP address is `10.1.0.13`
- The alias IP address is `10.1.0.65`

12.5.1 Prerequisites

- Check that the following HA specific RPMs are installed:
  - **HighAvailability-x.x-x.noarch.rpm** (installed from the Cluster Management CD).

  These RPMs install configuration files in the **/etc/HA** folder.
- The same RPMs must be installed on both `ns0` and `ns13`. To check the installed RPMs on both nodes, run the following command on ns0:

```
/etc/HA/bin/harpms fullstatus
```

- To check the list of the management nodes, enter:

```
/etc/HA/bin/haadminnodes
```

According to our example, this command will display:

```
ns0
ns13
```

- Verify the BMC configuration of the Management Node. See *section 2.3.3* in the HPC BAS4 *Installation and Configuration Guide* for more information.
- The ClusterDB has been configured on `ns0` as described before in this chapter.
- The network must be configured on both nodes. Do not define an alias on `ns13`.
- The database (**/var/lib/pgsql**) must **NOT** be a mount point, neither on `ns0` nor `ns13`. It must be present locally on `ns0`.
- Using the **phpPgAdmin** web interface for the ClusterDB change the value of **CLUSTER.actif-ha** field to *true* and the **NODE.status** field of the 2 Management Nodes to *up*. A virtual IP address for the backbone must be defined and the value for the **CLUSTER.node_backbone_ipaddr** updated in the **ClusterDB** with **phpPgAdmin**.

- The shared storage system for the management nodes must be fully configured and shared LUNs which include sufficient space must have been created. The shared storage systems must not be used, particularly not by the shared LUNs.
- All the directories that will be shared should not be mount points for those which will be managed by **halinks**. All other directories, for example **/tftpboot,** which are mount points, will be managed manually.

12.5.2 Configuring Channel Bonding

The Channel bonding mechanism makes it possible for a system to treat two physical interfaces in a logical bond with only one IP address. It allows the kernel to provide a single logical interface for two physical links connected to two distinct switches, with only one bond being used at the same time. The name of the interface can be indicated by the user, it is usually similar to "`bond0`".

Note: Channel Bonding is not a mandatory feature for High Availability.

Important:

The following tasks must be performed on each management node.

Before beginning the configuration operation stop the interfaces (In the example below these are named eth0 and eth2) which will be used using the **ifdown** command.

```
ifdown eth0
ifdown eth2
```

1. **Channel Bonding**

   In the **/etc/sysconfig/network-scripts** directory of each management node, define the following files:

   ifcfg-bond0 file

```
DEVICE=bond0
ONBOOT=yes
TYPE=Bonding
BOOTPROTO=none
NETMASK=255.255.0.0
IPADDR= IP address of the node on eth0 interface
(10.1.0.1 for ns0 and 10.1.0.13 for ns13)
USERCTL=no
PEERDNS=no
IPV6INIT=no
NOZEROCONF=yes     #To overide the default address provided
                    by Red Hat
```

   `MACADDR` is optional, however it should not be used.

   `NETMASK` and `IPADDR` depend on the network configuration of the management node.

ifcfg-eth0 file

```
DEVICE=eth0
ONBOOT=yes
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
ISALIAS=no
BOOTPROTO=none
NOZEROCONF=yes     #To overide the default address provided
                    by Red Hat
```

ifcfg-eth2 file

```
DEVICE=eth2
ONBOOT=yes
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
ISALIAS=no
BOOTPROTO= none
NOZEROCONF=yes     #To overide the default address provided
                    by Red Hat
```

The **ifcfg-eth0:1** file must be deleted because the IP alias is managed by the HA software. This file exists only on the Primary Management Node.

HWADDR is optional, however it should not be used.

IPADDR and NETMASK are not used here.

1. Gateway and Route

   Rename the **route-eth0** file in **route-bond0** if it is present.

   You may then find something similar to the following in the **route-eth0** file.

   ```
   10.0.0.0./8 via 10.1.255.254
   ```

Note:

Replace 10.0.0.0 with the IP address for your network and the second address (10.1.255.254) with the address of the switch

2. modprobe.conf

   In the **/etc/modprobe.conf** file define the 'bonding' module.

   Assuming that eth0 is the first interface set as "slave" of the channel bonding, add the following lines for the eth0 interface:

   ```
   alias bond0 bonding
   options bonding mode=1 downdelay=1000 updelay=1000 miimon=1000
   primary=eth0
   ```

3. Restart the Network

   Run:

```
service network restart
```

12.5.3 Configuring Services

Configuring NTP

As a result of its configuration capabilities NTP is a service which does not require High Availability management with **Cluster Suite**.

Assuming that `10.1.0.1` is the IP address of the primary management node and `10.1.0.13` is the IP address of secondary management node, edit the **/etc/ntp.conf** file to configure NTP as follows, if you are using a local clock:

1. On the primary management node:

```
restrict default nomodify notrap noquery
restrict 10.1.0.13
restrict 10.0.0.0 mask 255.0.0.0 notrap nomodify
restrict 127.0.0.1
peer 10.1.0.13
server  127.127.1.0     # local clock
fudge   127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
tinker panic 0
tinker stepout 0
```

2. On the secondary management node:

```
restrict default nomodify notrap noquery
restrict 10.1.0.1
restrict 10.0.0.0 mask 255.0.0.0 notrap nomodify
restrict 127.0.0.1
peer 10.1.0.1
server  127.127.1.0     # local clock
fudge   127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
tinker panic 0
tinker stepout 0
```

3. Restart the service on both nodes:

```
service ntpd restart
```

Configuring syslog-ng

To avoid socket creation problems during machine boot, it is recommended that a listening socket is created with the address 0.0.0.0, rather than using other virtual or real addresses. A virtual address should be chosen for the sockets which will be used to send logs.

On each management node edit the file **/etc/syslog-ng/syslog-ng.conf**.

1. Search for all the lines which contain the `SUBSTITUTE` string.

2. For each line that is found, check and if necessary modify the lines which follow as indicated below:
   - For each `source` block which contains an IP address set the value **0.0.0.0** instead of the IP alias.
   - For each `destination` block which contains an IP address set the value **127.0.0.1**. However, using an alias IP address may be useful if the administrator would like all passive management nodes to send their logs to the active management node.

Configuring RMS

Usually the RMS partitions are not in **auto-start** mode. When a node rocks to another node, there is no existing High Availability mechanism to automatically restart the partitions. To ensure that the job submitted continues to be treated, it is recommended to activate the **auto-start** mode on the RMS partitions that the administrator wants to be used for the job.

Please refer to the RMS documentation for more information about using **auto-start** mode.

Configuring OCFS2 (useful for SLURM and LSF/SLURM)

A Cluster File System allows all nodes in a cluster to access a device concurrently via the standard file system interface. This allows the easy management of applications that need to run across a cluster, which is essential to assure the functionality of High Availability.

There are two tools packages. **ocfs2-tools** includes the command line tools and **ocfs2console** includes the GUI front -end for the tools.

To configure a **OCFS2** Cluster File System see:

http://oss.oracle.com/projects/ocfs2/documentation/

Configuring SLURM

To use the native failover mechanism, we must define a **SLURM** backup controller.

The **SLURM** controller, **slurmctld**, will periodically save the status of the job into a directory so that the said status may be recovered after a fatal system error. When using a backup controller, the filesystem in which this directory resides should be shared between the Primary Node (ControlMachine) and Back-up Node (BackupController). The location where job accounting logs are to be written must also be defined in the shared directory.

Below is a **slurm.conf** configuration file (slurm v1.15) for example:

```
ControlMachine=bali0
ControlAddr=10.0.0.1
BackupController=bali1
BackupAddr=10.0.0.2
StateSaveLocation=/ocfs2shared/slurmdata
JobAcctType=jobacct/log
JobAcctLoc=/ocfs2shared/slurmdata/slurm_accounting.log
```

Change the owner properties for the **/ocfs2shared/slurmdata** filsystem with the command:

```
chown –R slurm:root /ocfs2shared/slurmdata
```

Configuring LSF

To recover from server failures, host reboots, or service restarts, a mechanism of **workdir-replication** is used. **LSF** is configured to maintain copies of the logs, to be used for backups. In this case, we define two **LSF** servers corresponding to the Management Nodes. In essence, the Primary Node writes to two physical discs, its own local disc and to a mounted disc on the Back-up Node.

For more information, see the documentation available from:

http://lsfadmin.home.cern.ch/lsfadmin/docs/lsf6.1_admin/H_logs.html

To enable duplicate logging, set the **LSB_LOCALDIR** in the **lsf.conf** to a directory on the Primary Node, not on the Back-up Node.

Below is an example of an **lsf.conf** file :

On the Primary Node:

```
LSB_LOCALDIR=/usr/share/lsf/work
LSB_SHAREDIR=/ocfs2shared/lsf/work
LSF_SERVER_HOSTS="bali0 bali1"
LSF_MASTER_LIST="bali0 bali1"
LSF_SLURM_TMPDIR=/hptc_cluster/lsf/tmp
LSB_RLA_WORKDIR=/ocfs2shared/lsf/work/bali/rla_workdir
```

On the Back-up Node:

```
LSB_SHAREDIR=/ocfs2shared/lsf/work
LSF_SLURM_TMPDIR=/hptc_cluster/lsf/tmp
LSF_SERVER_HOSTS="bali0 bali1"
LSF_MASTER_LIST="bali0 bali1"
LSB_RLA_WORKDIR=/ocfs2shared/lsf/work/bali/rla_workdir
```

The tree structure of the **/usr/share/lsf/work** must be copied to **/ocfs2shared/lsf/work** with the same assess rights:

```
mkdir /ocfs2shared/lsf
mkdir /ocfs2shared/lsf/work
mkdir /ocfs2shared/lsf/work/bali
mkdir /ocfs2shared/lsf/work/bali/logdir/
mkdir /ocfs2shared/lsf/work/bali/lsf_cmddir/
mkdir /ocfs2shared/lsf/work/bali/lsf_indir/
mkdir /ocfs2shared/lsf/work/bali/rla_workdir/
chown -R lsfadmin:root /ocfs2shared/lsf
```

12.5.4 Implementing the Primary Node

The steps to implement High Availability on the primary node are as follows:

12.5.4.1 Prepare the Partitions

1. **Create the partitions on the external storage.**

Note:

The actual disk names may be different than shown in the example below, depending on the number of disks in the system.

For example on the **/dev/sde**, **/dev/sdf**, **/dev/sdg**, **/dev/sdh**, **/dev/sdi** devices, run the **mkfs.ext3** command as shown below:

```
mkfs.ext3 /dev/sdf
```

2. **Declare the file systems. One external storage LUN per file system.**

Mount point	File System (example)	Size		
		Mini	Maxi	Recommended
/etcha	/dev/sdf	500 MB	1 GB	1 GB
/usrha	/dev/sdg	5 GB	15 GB	15 GB
/varha	/dev/sdh	20 GB	50% of the remaining = 112 to 121 GB	70 GB
/homeha	/dev/sdi	10 GB	50% of the remaining = 112 to 121 GB	50 GB
/tftpboot	/dev/sde	2 GB	10 GB	10 GB

Edit the **/etc/HA/mnttab file** and add the following lines, according to the above table:

```
/dev/sdf          /etcha
/dev/sdg          /usrha
/dev/sdh          /varha
/dev/sde          /tftpboot
/dev/sdi          /homeha
```

Important:

The size of the file systems must fit the space available. See the recommended sizes in the table.

3. **Mount these new file systems.**

   a. Save the current **tftpboot**:

```
mv /tftpboot /tftpboot.orig
```

b. Create the mount directories:

```
mkdir /etcha /usrha /varha /homeha /tftpboot
```

c. Mount the file systems:

```
/etc/HA/bin/hamount start
```

d. Retrieve the tftpboot data.

```
cp -af  /tftpboot.orig/*  /tftpboot/.
```

e. To check that the mount operations have been correctly performed run:

```
mount
```

or:

```
df -h
```

and check that the **/etcha /usrha /varha /tftpboot /homeha** partitions are mounted.

12.5.4.2 Check Configuration Files

The **/etc/HA** directory contains several configuration files that should be checked to ensure they fit the configuration of your management nodes. You should particularly check The following files should be customized according to the Resource Manager and the mount points of the shared storage system:

/etc/HA/start
/etc/HA/stop
/etc/HA/status
/etc/HA/synchro
/etc/HA/mnttab
/etc/HA/ip
/etc/HA/links

/etc/HA/cron

Run the command below to deactivate the crons:

```
/etc/HA/bin/hacron stop
```

Define the alias IP address

Edit the **/etc/HA/ip** file and add the following line (example):

```
bond0       10.1.0.65/16
bond1      176.18.118.65/24
```

or if bonding is not used

```
eth0       10.1.0.65/16
```

```
eth1      176.18.118.65/24
```

12.5.4.3 Configuring Cluster Suite

Run the command below to generate the **/etc/cluster/cluster.conf** file. See section 12.6.2 for more information.

```
cs-config-admin
```

12.5.4.4 Control the Services Started at Boot Time

Cluster Suite software stops and restarts some system services. It is mandatory that these services are not started at boot time. The **haunwantedfiles** script ensures this task: it stops the daemons and prevents them from starting at next boot. Run:

```
/etc/HA/bin/haunwantedfiles repair
```

12.5.4.5 Set-up the Links to Shared Disks

To set up the links to shared disks, run:

```
/etc/HA/bin/halinks repair
```

12.5.5 Start HA on the Primary Node

1. Start Cluster Suite by running this command:

```
cs -c start
```

2. Then start the services, running the following command, where HA_NSM is the name defined in the **cluster.conf** file.:

```
clusvcadm –e HA_NSM
```

3. Check that the launch is successful, by running the command:

```
clustat
```

and by checking the **/var/log/ha.log** file.

12.5.6 Implementing the Secondary Node

The secondary node(s) must have the same packages installed on them as on the primary node. The different services must be configured except those which are managed by High Availability.

The steps to implement High Availability on the secondary node are as follows:

12.5.6.1 Retrieve /etc/hosts

Retrieve the **/etc/hosts** file from the primary management node and copy it to the secondary management node.

12.5.6.2 Synchronize Files

From the primary management node, synchronize the files that cannot be put on the shared storage system:

```
/etc/HA/bin/hasynchro repair
```

This command has to be run twice. The first time the command synchronizes the files. The second time the command should return an OK status.

12.5.6.3 Inhibit Daemons for the Launch

From the secondary management node, inhibit the daemons controlled by the Cluster Suite by removing the links of **/etc/rc.d** and by stopping the daemons if they are running.

```
/etc/HA/bin/haunwantedfiles repair
```

12.5.6.4 Set-up Dynamic Links

From the secondary management node, set-up the links for the shared directories:

```
/etc/HA/bin/halinks repair
```

Important:

Run this command a second time and check that the output is "OK".

12.5.6.5 Prepare the Mount Points for the Shared Partitions

From the secondary management node, create directories as follows:

```
mkdir /etcha /homeha /usrha /varha
```

12.5.6.6 Start Cluster Suite

Run:

```
cs -c start
```

Check that Cluster Suite has started correctly and that the primary management node is the active node which is running:

```
clustat
```

The output should show that ns0 and ns13 are defined as 'member name' and that ns0 is the 'owner' which means that it is the active node as in the example below.

```
Member Status: Quorate, Group Member

  Member Name                              State      ID
  ------ ----                              -----      --
  ns0                                    Online     0x0000000000000001
  ns13                                   Online     0x0000000000000002

  Service Name         Owner (Last)                  State
  ------- ----         ----- ------                  -----
  HA_NSM               ns0                         started
```

12.6 Managing Cluster Suite

The following commands and service are often used to manage Cluster Suite:

cs	To start / stop the Cluster Suite daemons (see 12.6.1).
cs-config-admin	A command for the automatic configuration of High Availability on the Management Nodes (see 12.6.2).
clustat	To obtain online information regarding the Cluster Suite status (see 12.6.3).
clusvcadm	To administer Cluster Suite using a command line interface (see 12.6.4).
'HA System Status' service	This Nagios service displays the state of the management nodes which are running with High Availability. It is used within NovaScale Master – HPC Edition (see Chapter 8).

12.6.1 /etc/HA/bin/cs

The Cluster Suite daemons are launched using the **cs** command:

```
cs -c start
```

The Cluster Suite daemons may be stopped by using the command:

```
cs -c stop
```

Important:

Using this command may mean that it is not possible to relaunch the Cluster Suite daemons without rebooting the node.

12.6.2 cs -config-admin

The command, below, is to configure High Availability on the Management Nodes automatically:

```
/usr/sbin/cs-config-admin
```

There is no need to use the **cs -config-admin** graphical interface.

The command requires:

1. ClusterDB access to the Primary Management Node.
2. A Secondary Management Node must be connected to the Administration network.

Assuming the Primary Management Node can access the ClusterDB, run the **cs -config-admin** command without including any additional parameters.

If the command returns the error below;

```
No da_iocell_component preloaded for management nodes, give HA
node name as parameter please.
```

Then either update the ClusterDB with details of the management iocell, or run the command again using the Secondary node name as a parameter, as shown below;

```
cs-config-admin <node_ha_name>
```

The command must have been run successfully before the High Availability mechanism is launched.

12.6.3 clustat

This command is only run on the management nodes on which Cluster Suite has been started. It is used to display the status for Cluster Suite, see the example below:

```
[root@nova0 ~]# clustat
```

This will give output similar to that shown below:

```
Member Status: Quorate, Group Member

  Member Name                        State     ID
  ------ ----                        -----     --
  tiger0                             Online    0x0000000000000001
  tiger7                             Online    0x0000000000000002

  Service Name         Owner (Last)               State
  ------- ----         ----- ------               -----
  HA_NSM               tiger0                     started
```

12.6.4 clusvcadm

This command is only run on the management nodes on which Cluster Suite has been started. The most useful options are:

clusvcadm -d <service name>	Disable a service
clusvcadm -e <service name>	Enable a service
clusvcadm -e <service name> -m <member>	Enable a service on a member node
clusvcadm -r <service name> -m <member>	Relocate an active service to a specific member node

Refer to the man page for this command for more information.

```
man clusvcadm
```

12.7 Patching a Node / Updating an Application

12.7.1 Introduction

Important:

It is not possible to give a generic procedure to apply a patch or to de-install/install an application or a package on management nodes, while High Availability is active and operational. It is the responsibility of the administrator to find the best solution. The administrator needs to understand the High Availability mechanism as well as any particular concerns for the application or for the data which needs to be updated.

Below is a list of difficulties which may arise when applying a patch or de-installing/installing a package or an application:

- The impossibility to de-install due to files or directories which cannot be accessed, following the High Availability operation to link them with data on the shared storage system.
- The operations of removing links and replacement by directories can lead to the situation where data remains on the shared storage system. Therefore as the **halinks repair** command does not remove data, the data on the shared storage system will not be updated and this will result in incompatibilities at reboot, and even data corruption.
- When the patch is run on the active node, a **migration** of data on the shared storage system may result. When another node becomes the active node then this new active node does not include the patch and it will not be able to start some services or will corrupt some data. Furthermore, if a passive node is forcefully patched by manually mounting the file systems, the patch operation may fail and/or corrupt data.
- If the patch is applied to the local data, but not distributed to the data of the shared storage system, then there is a risk of data corruption and/or impossibility of starting the application.

Although there is no procedure for updating applications on a High Availability management node, some examples follow. Only the principles are described. It will be necessary to include additional manipulations at the time of a real intervention.

12.7.2 Example of Updating RMS

In this example, RMS is updated with a patch, using the command:

```
rpm -Uvh <package pathname>
```

The RMS directories which are shared are the following ones:

/usr/lib/rms/msql

/var/rms

Notes:

- Updating RMS packages locks the whole cluster, since all the management services will be stopped
- The update must be performed on all management nodes, but using different procedures for the primary and secondary nodes described below.
- Free space on the local disks and on the shared storage system is used to perform the update.

Updating RMS on the Primary Node

The steps to update RMS on the primary node are as follows:

1. Stop the service managed by Cluster Suite. For example, to stop the HA_NSM service, run:

```
clusvcadm -d HA_NSM
```

2. Return the symbolic links to their original state:

```
/etc/HA/bin/halinks undo
```

3. Remove the shared directory:

```
rm -rf /usr/lib/rms/msql
rm -rf /var/rms
```

4. Restore the starting scripts to their original state:

```
etc/HA/bin/haunwantedfiles undo
```

5. Mount the shared storage system:

```
/etc/HA/bin/hamount start
```

6. Copy the shared directories from the shared storage system to the local disks:

```
cp -af /usrha/lib/rms/msql /usr/lib/rms/msql
cp -af /varha/rms /var/rms
```

7. Remove the directories from the shared storage system:

```
rm -rf /usrha/lib/rms/msql
rm -rf /varha/rms
```

8. Apply the patch by running the command:

```
rpm -Uvh <package pathname>
```

9. Restore the symbolic links by running the command:

```
/etc/HA/bin/halinks repair
```

10. De-activate the start of the services managed by Cluster Suite by running the command:

```
/etc/HA/bin/haunwatedfiles repair
```

11. It is better to stop Cluster Suite on the secondary nodes if these are not patched before restarting the service on the primary node. On each secondary node run the command:

```
cs -c stop
```

12. Restart the service managed by Cluster Suite:

```
clusvcadm -e HA_NSM
```

Updating RMS on the Secondary Node

The steps to update RMS on the secondary node are as follows:

1. Stop Cluster Suite:

```
cs -c stop
```

2. Return the symbolic links to their original state:

```
/etc/HA/bin/halinks undo
```

3. Restore the starting scripts to their original state by running the command:

```
/etc/HA/bin/haunwantedfiles undo
```

4. Apply the patch:

```
rpm -Uvh <package pathname>
```

5. Restore the symbolic links:

```
/etc/HA/bin/halinks repair
```

6. De-activate the launch of the services managed by Cluster Suite:

```
/etc/HA/bin/haunwatedfiles repair
```

7. Restart Cluster Suite:

```
cs -c start
```

If Cluster Suite has not correctly started, reboot the node, then restart Cluster Suite once the node is booted and reachable.

12.7.3 Example of Updating Lustre

Updating **Lustre** is similar to that of **RMS** when the command **rpm – U** is used . The only difference lies in the shared directories. For example, the **/var/lib/ldap/** directory exists for **Lustre**, whereas for **RMS** the **/var/lib/ldap** directory is present in the **/etc/HA/links** file. So, it will be necessary to use this directory for copy and delete operations and not the **/var/lib/ldap/** directory.

Chapter 13. I/O Node and Lustre File System High Availability

This chapter explains how to implement High Availability for I/O Nodes and Lustre file system.

13.1 Introduction to Lustre File System

Lustre uses object based disks for storage. Metadata servers are used for storing file system metadata. This design provides a substantially more efficient division of labor between computing and storage resources. Replicated, failover metadata Servers (**MDS**s) maintain a transactional record of high-level files and file system changes. Distributed Object Storage Targets (**OST**s) are responsible for actual file system I/O operations and for interfacing with storage devices. This division of labor, and of responsibility, leads to a truly scalable file system and more reliable recoverability from failures by providing a combination of the advantages of journaling and distributed file systems. **Lustre** supports strong file and metadata locking semantics to maintain total coherency of the file systems even when there is concurrent access. File locking is distributed across the storage targets (**OST**s) that constitute the file system, with each OST handling locks for the objects that it stores.

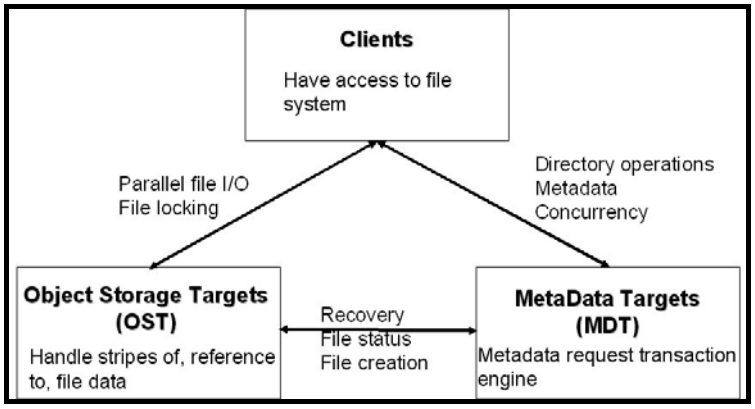


Figure 13-1. Lustre interactions

For more information, see:
Lustre: A Scalable, High-Performance File System Cluster File Systems, Inc.
http://www.lustre.org/docs/whitepaper.pdf

13.2 Lustre Failover Mechanism

Lustre supports the notion of failover pairs. Two nodes which are connected to shared storage can function as a failover pair, in which one node is the active provider of the service (OST or MDT), and the second node is the passive secondary server.

The Lustre services are declared on both nodes with the same name. The MDT is configured with a list of servers (OSSs) for clients to pass through in order to connect to the OSTs. The Lustre servers must have distinct network addresses.

The failover mechanism of the Lustre system is based on the capacity to enable client reconnection when the OSTs and MDTs are moved to other nodes.

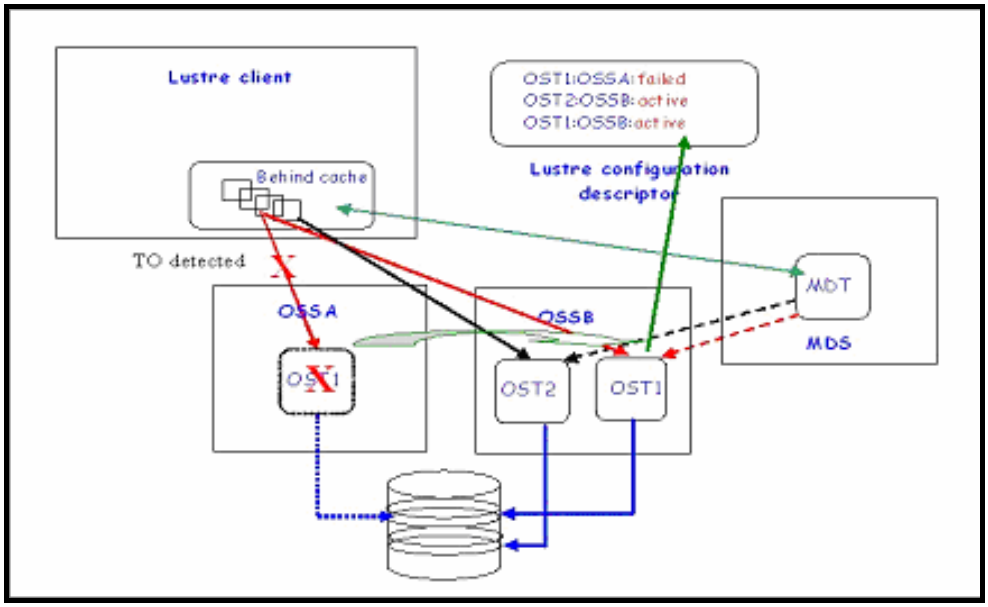


Figure 13-2. OST takeover and client recovery

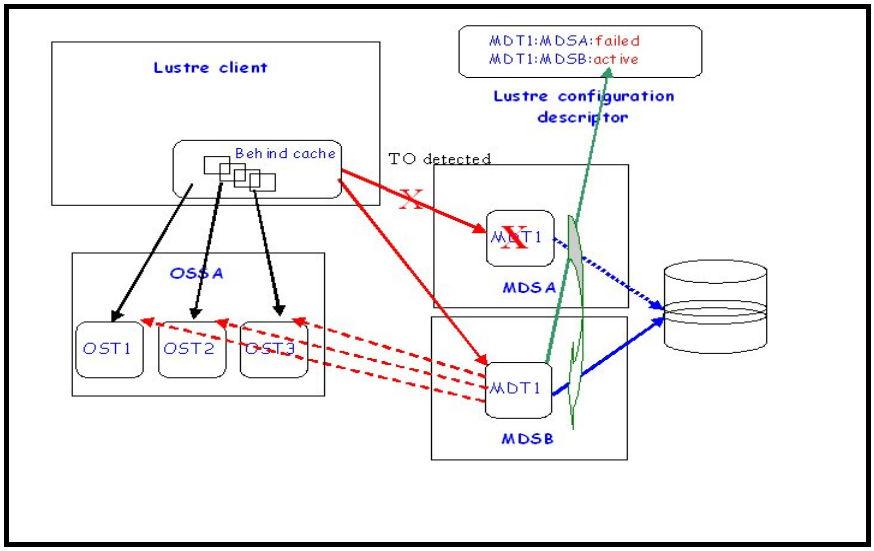


Figure 13-3. MDT takeover and client recovery

Lustre targets work in a synchronous mode: a client request is executed on the storage device and acknowledgement provided for the client only after it has been acknowledged by the device. In the case of a failure, the storage devices will ensure that committed data is preserved. Uncommitted data, meaning data not acknowledged, is kept by the clients in their "reserve cache" and can be resent when the system recovers.

When a request to a target (MDT or OST) is not acknowledged in time, the Lustre client suspects a failure and starts the recovery procedure as follows:

- Checks routing information in the Lustre configuration descriptor
- Reconnects to the migrated target
- Lost transactions are replayed
- Locks are re-acquired
- Partially completed multi-node operations are restarted.

On the server side, the target failover mechanism is similar to the High Availability service migration: the service is "stopped" on one node, and then restarted on the rescue node which has access to the same storage devices.

Transactions requested on metadata local to the targets (**ext3** log files) and those which are global for the Lustre system (MDT) are logged in log files. These files are replayed when there is a service migration.

Lustre does not prevent simultaneous access. This means that an external mechanism must ensure that the shared storage will only be accessed by one node at a time. This can be done by powering off the failed node.

13.3 Hardware Architecture

Bull High-Availability management of the Lustre system relies on a specific hardware architecture.

The I/O nodes operating in HA mode are grouped in I/O cells which brings together two I/O nodes that access one or more disk arrays. The I/O cell contains either **OST**s or **MDT**s, but both are exclusive.

Usually, nodes are directly connected to the storage systems ports without intermediate switches or HUBs. This "point to point link" avoids having additional active components which may become other SPOFs.

The LUNs within the storage systems are accessible for both nodes of the I/O cell, enabling OSSs and MDSs to retrieve their data when they are moved to the other node. But each LUN must be used by only one node to avoid data corruption.

An I/O fencing mechanism is implemented so that the faulty node can not access the LUNs again after the OSSs or MDSs are restarted on the peer node of the I/O cell. In any case the failing node is powered off.

The underlying mechanism which ensures OSS and MDS migration and I/O fencing is provided by **Cluster Suite**. The failover process relies on basic entities known as failover services. When a node fails, **Cluster Suite** determines how each service should be relocated.

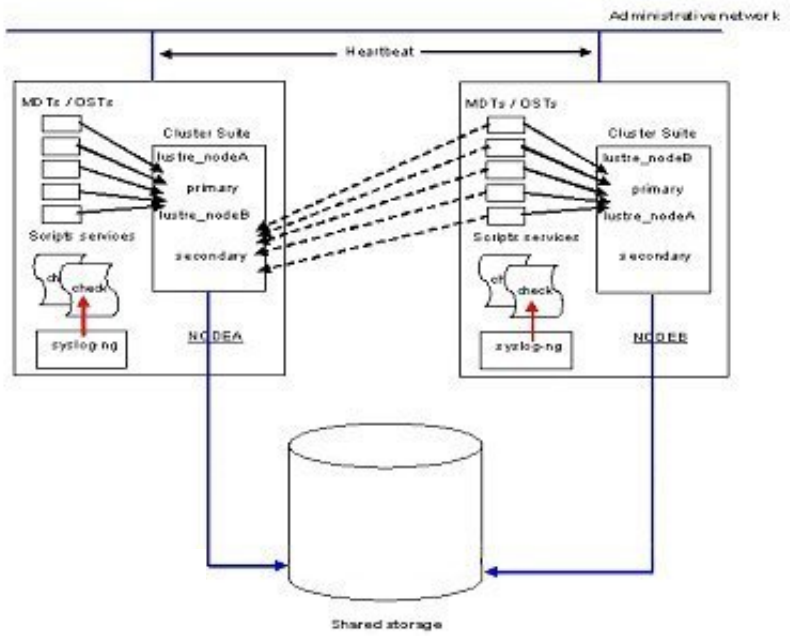


Figure 13-4. I/O Cell diagram

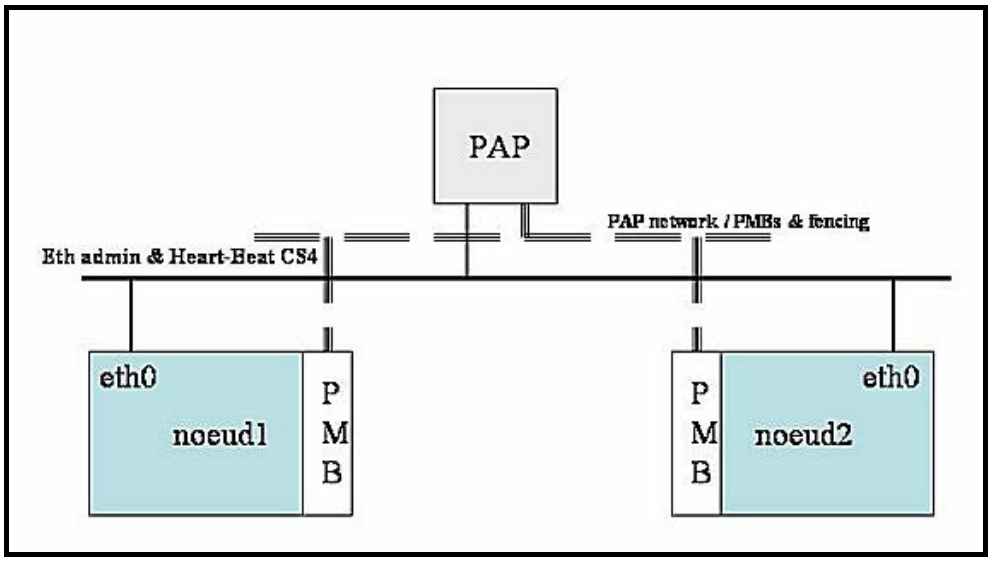


Figure 13-5. High Availability/Cluster Suite on Novascale 50xx nodes

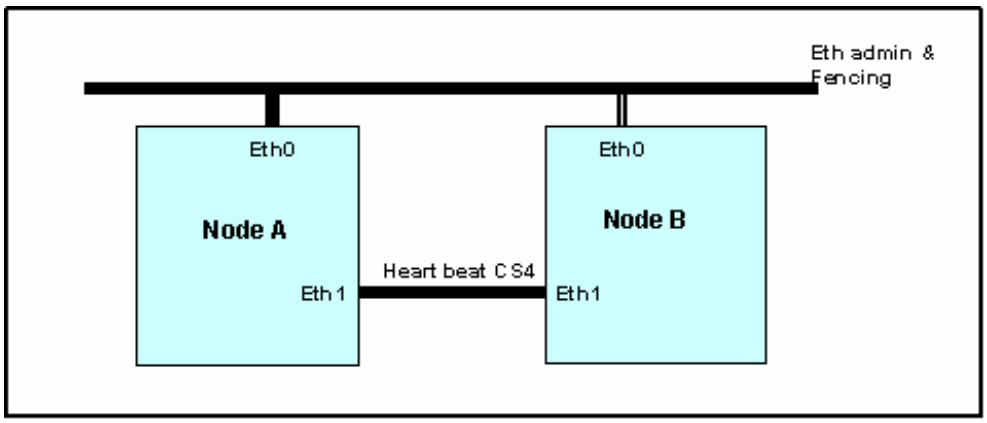


Figure 13-6. High Availability/Cluster Suite on Novascale 40xx nodes

In the case of multi-types I/O nodes (nodes which serve as both OSSs and MDSs), the Lustre file systems must be configured so that the same node does not support both MDT and OSTs services for the same file system. If not, a failure of this type of node constitutes a double failure (MDS + OSS) for the Lustre file system and its recovery is not guaranteed. The following figure illustrates how you can position OSTs and MDTs for two file systems **FS1** and **FS2**.

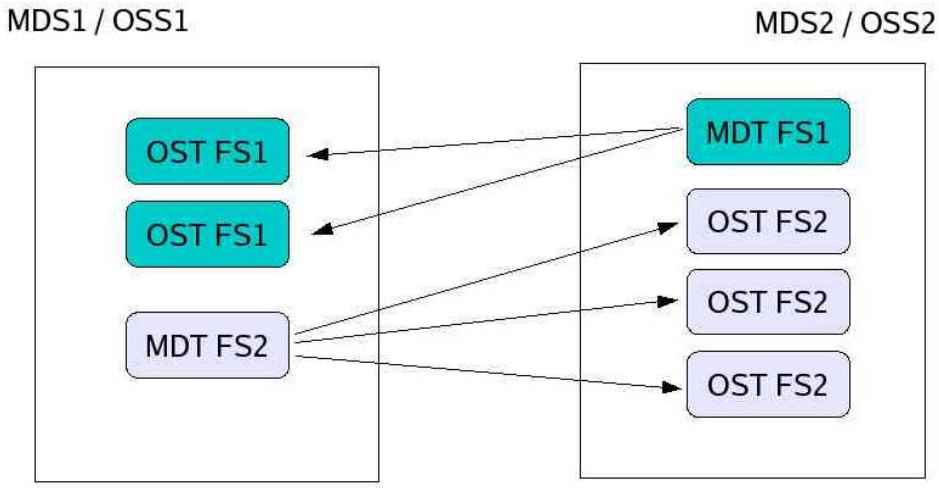


Figure 13-7. MDT/OST Dispatching on two nodes

13.4 High Availability Policy

In a Cluster Manager environment the customer can simply spread the application services across the clustered servers in any way that seems appropriate. All nodes will be actively running a share of the total load.

This is called **Active/Active clustering**, as all servers are active. If one server shuts down, the other servers will pick up the running load of its services.

The High Availability mode to be applied in the Bull HPC context is ***mutual takeover*** for each node of a pair of nodes in the same I/O cell. In standard state, each I/O node supports its own Lustre services (MDTs or OSTs), whereas in a failure state, one node manages its own services plus all the services from the failing node.

Firstly, all Lustre services from the failing node will be migrated to the second node, even if the failure concerns only one Lustre service. It means that for each I/O node, one failover service is defined which includes all the currently installed Lustre services.

Cluster Suite requires a service script for each failover service that will be managed. The script must be able to stop, start and report status for the service.

On each unitary High Availability cluster based on the I/O cells, two failover services are defined: one for the Lustre services of each node.

In the I/O Cell above we have:

lustre_nodeA with primary node NODEA and secondary node NODEB
lustre_nodeB with primary node NODEB and secondary node NODEA

A different failover script is associated with each of the two services provided they are not composed of the same Lustre components (MDTs / OSTs).

At any moment, the Lustre failover service on an I/O node is composed of all the Lustre services (MDTs / OSTs) associated with the **active** file systems. Its composition is subject to change according to the Lustre file systems activation.

On an I/O node, the Lustre services (MDTs / OSTs) are not started for the boot but only by Lustre administrative tools by means of file system start. This to ensure consistency of the Lustre file system services start on all the nodes it relies on.

For a reboot of a failed MDS or I/O node, the Lustre services are not automatically relocated. This is may be done only by the administrator using a Lustre management tool. This mechanism is chosen to avoid inopportune Lustre services migration when there is a partial repair of the primary node.

Important:

A simultaneous migration of the management station and of the metadata server is considered as a double failure and is not supported.

13.5 High Availability Management

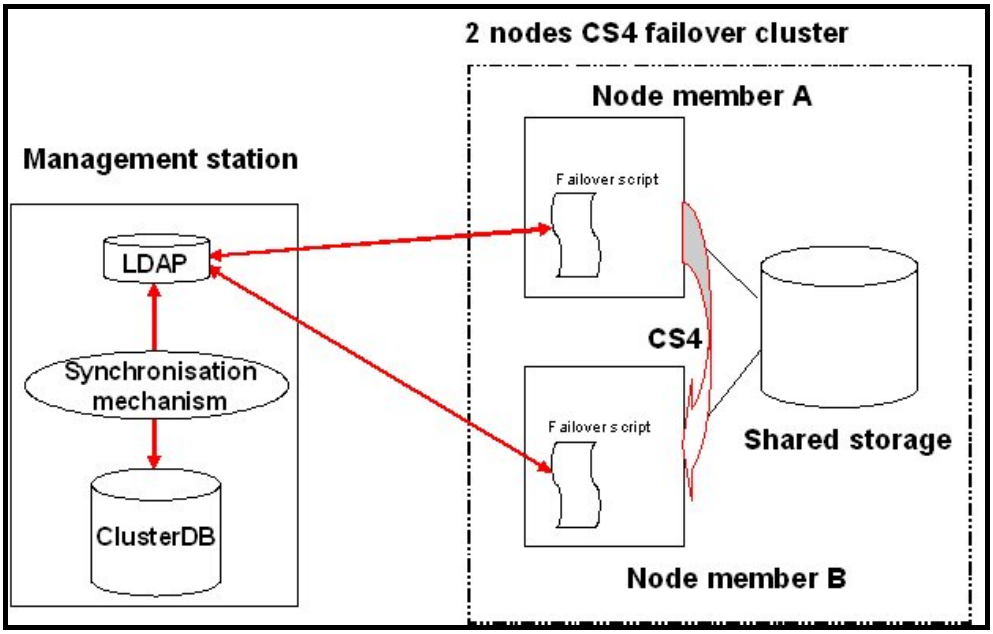


Figure 13-8 Lustre High-Availability Management architecture

When targets (MDT/OST) failover is configured, not only the information about paired targets (a cell) is stored in the Lustre configuration information of the ClusterDB, but also which instance of that target is the currently active one. It is the High Availability application (Cluster Suite script) that has to dynamically update the active instance information of the Lustre tables into the ClusterDB.

LDAP Directory

The LDAP directory is the pivot of the Lustre High Availability architecture. It has been proved to be the more flexible and more efficient way to share Lustre configuration information on a large cluster. It is the repository of the actual file systems distribution on the cluster:

- which file systems are active,
- Lustre services (OST/MDT) migrations.

The information status it contains is updated by the Lustre failover script every time a High Availability event occurs.

This information status has to be synchronized with the clusterDB one in order to ensure file systems status and Lustre services migrations monitoring by the Lustre administration tools.

Using such a mechanism allows Lustre file systems relying on some migrated nodes, to be stopped and restarted respecting the actual Lustre services node distribution.

Clients /Servers Reconnections

An 'epoch' number on every storage controller, an incarnation number on the metadata server/cluster, and a generation number associated with connections between clients and other systems form the infrastructure for Lustre recovery, enabling clients and servers to detect restarts and select appropriate and equivalent servers.

Failover Scripts

The sole difference between both failover scripts of a HA unitary cluster is the set of Lustre services it includes. The specificity of this set is to evolve according to the Lustre file systems activation / de-activation.

A generic failover script **/usr/sbin/failover/lustre_failover** is implemented on the I/O nodes. The set of Lustre services to manage is dynamically determined by requesting the LDAP directory for active file systems having a group of services on this node.

The group to check for is always the same for a node but different for each one. To deal with that, the failover script associated with a failover service is a symbolic link to this generic script which name includes the primary node name:

```
/usr/sbin/lustre_failover_<primary_node_name>
```

The group of Lustre services to check for is determined by parsing the script calling name. The symbolic link is configured once at cluster deployment time.

A **naming convention** is established which is to be taken in account by Lustre file systems configuration and the Cluster Suite configuration:

```
failover service name = lustre_<primary_node_name>
```

13.6 Error Detection and Prevention Mechanisms

Lustre Single Point Of Failure (SPOF) tracking

Tracked Lustre services SPOF include:

1. Service crash (no longer running on the node).

2. Lustre services in an unavailable state (hanging, starting, etc.).

3. Repetitive abnormal comportment (systematic client eviction, etc.).

4. I/O errors on the back-end device.

The Lustre services failures (points 1 to 3) detection relies on the intrinsic Lustre health monitoring system. This internal failures management mechanism maintains diagnose items in the local **/proc/sys/lustre** and **/proc/fs/lustre** directories of each I/O or metadata node.

A regular check of Lustre services sanity is scheduled by the Cluster Suite through the status target of the failover script. This check will process the diagnose items maintained by Lustre.

I/O errors on the back-end device (point 4) detection relies on the storage management monitoring daemon **storfilter**. When it detects a problem, this daemon warns the Lustre failover script using a dedicated target.

When one these SPOF is detected, a Lustre services migration is triggered followed by a node power off.

13.7 Analysis of Failure Modes

13.7.1 I/O Node and Metadata Failures

I/O Node Panic/Hang

When a node hangs or encounters a panic, it does not send it's heartbeat messages in the authorized period. This silence is detected by the peer HA node which fences the silent node and takes over the cluster services when the fencing is completed.

I/O Node Power down

The node does not send its heartbeat messages in the authorized period. This silence is detected by the peer HA node which fences the silent node and takeover the cluster services when the fencing is completed.

Lustre Software Failure

Two scenarios can take place leading to the same action:

- The health monitoring mechanism of the Lustre system detects the failure and update the health information in the **/proc** directory. The next time the status target of the Lustre failover script is activated by the Cluster Suite, it will detect the problem and power off the failing node.
- The status target of the Lustre failover script detects that some Lustre services are missing or not in the correct state. It first tries to restart them. In case of failure it will power off the failing node.

Both scenarios trigger the **I/O node power down** failure treatment.

13.7.2 Storage Failures

Fibre Channel Adapter Errors

Fibre channel adapters are used to access to external storage systems, shared by the two nodes of the HA I/O cell. They store the OSS and MDS data.

Fibre channel adapter errors are ignored. Typically, link events may be transient on fibre channel links, and should not lead to a node failover. If the adapter error is real, it will lead to a linux disk error, which is monitored.

Disk Subsystem Failure

When a disk subsystem, despite its internal redundancy, encounters errors which prevent the processing of node's I/O requests, the node will detect a disk error.

Disk errors are monitored by the I/O status service; they are notified to the Management Node, and to Lustre management locally on the node. Lustre will execute the necessary actions, and then stops the node. The node being silent, the Cluster Suite will take over the service on the peer HA node.

Note:

Lustre verifies that the device generating the I/O error is being used by Lustre. If it is not then no corrective action will be taken.

SCSI Adapter Error or Hang

SCSI adapters are usually used to store systems data, binaries, swap, and temporary files. A SCSI adapter failure leads to a kernel hang or panic, or to a lustre service failure. These two types of failure have already been described.

13.7.3 Ethernet Network Failures

A failure of the heartbeat network will stop heartbeat exchanges leading each node to initiate to service take over. A fencing race starts between both nodes.

Note:

Only the heartbeat network is monitored by Cluster Suite. A failure on another Ethernet network than the one used for heartbeat will not lead to service takeover; however the failure will be displayed on the management node via NovaScale Master - HPC Edition.

Ethernet Network Access Failure (NIC or link Failure)

The management network is also used to send fence requests to the appropriate PAP, this is the only way with the FAME architecture to stop a node remotely.
The node which is unable to use the management network cannot fence its peer node.
Thus, the peer node wins the fencing race, and takes over the cluster services.
There is no risk of split brain (i.e. both nodes of the I/O cell running simultaneously the same Lustre service).

Management Network Failure

If the management network is unavailable for both nodes of the HA I/O cell, none will be able to fence its peer node. The Cluster Suite does not initiate any failover action.
If one the node of the HA pair is able to fence the peer node, it wins the fencing race and takes over the services.

13.8 Using Cluster Suite (Cluster with Management Node)

Large clusters may contain multiple I/O cells, and within each I/O cell, the Cluster Suite must be configured. This process is fully automated by Bull cluster management tools. All the necessary information are extracted from the cluster DB, and read to use Cluster Suite configuration files are pushed to each node which must be controlled by the Cluster Suite.

Important:

Cluster Suite commands not described in the present paragraph must not be used, as they may lead to fatal inconsistency for the Lustre file system. The GUI must not be used as well. All the Cluster Suite setup is predefined to enable the failover process expected by the Lustre file system, and prevent any risk of split brain (i.e. both nodes of the I/O cell running simultaneously the same Lustre service). Administrator must not attempt to modify the Cluster Suite's configuration within I/O cells.

The management tasks for Cluster Suite are:

- Distributing the configuration file
- Starting the Cluster Suite.

By default, there is not automatic start at boot time, and it is not recommended to enable this.

13.8.1 Distributing the cluster.conf file on the I/O Node

The **/etc/cluster/cluster.conf** is generated using node HA pair defined in the cluster DB. The following options are selected, and must not be changed:

- Name of the services to be managed.
- Manual start of services (to avoid split brain if a node can not join its peer node).
- List of nodes.
- For NovaScale 5xxx/6xxx:
  - Heartbeat through the management network
  - Use of fence based on Bull PAP (parameters extracted from the ClusterDB)
- For NovaScale 40xx :
  - Heartbeat through a dedicated Ethernet network; this network must be configured on all nodes; it must not be the eth0 Ethenet network (where IMPI is configured), and IP name must be likewise: <node_name>_hb (in **/etc/hosts**).

The Cluster Suite configuration files are automatically generated and deployed on each node by the **stordepha** command.

```
stordepha -c configure -a
```

The **–a** flag means all nodes. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the man page of the command for more information.

Note:

The **cluster.conf** file is not preserved when a node is reinstalled by KSIS. It must also not be integrated in a node image. After each node's deployment, the **storedepha** command must be used to restore the node's configuration.

13.8.2 Starting / Stopping Cluster Suite's Daemons

The Cluster Suite's daemons can be configured from the Management Node on all or a subset of the HA I/O nodes, or locally on each node. In both case, all the required daemons are started and stopped consistently, in the right order.
Starting the Cluster Suite starts the Cluster Suite daemons. But the Cluster Suite services do not start, because the automatic start is disabled.
Stopping the Cluster Suite on a node causes its services to fail on the peer node.

- From the management station:

```
stordepha -c start|stop -a
```

The **–a** flag means all nodes. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the man page of the command for more information.

- Locally on a node:

```
storioha -c start|stop
```

13.8.3 Checking the Cluster Suite Status

The Cluster Suite's status can be verified from the Management Node on all or a subset of the HA I/O nodes, or locally on each node.

- From the management station:

```
stordepha -c status -a
```

The **–a** flag means all nodes. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the man page of the command for more information.

- Locally on a node:

```
storioha -c status
```

Alternatively, it is also possible to use the Cluster Suite's **clustat** command:

```
clustat
```

or:

```
clustat -i <refresh period>
```

13.9 Managing Lustre High Availability (Cluster with Management Node)

Lustre High-Availability management is included in the Lustre management framework under the form of add-ons and specific tools. It is operated from the management station.

The management tasks for Lustre failover are:

- Setting up the hardware and software configurations information.
- Enabling file systems for failover support.
- Dealing with the nodes migrations and Lustre services take over.

13.9.1 ClusterDB Information

Two kinds of information are included into the Lustre tables of the ClusterDB to allow Lustre services failover management and monitoring:

- Static information linked with the cabling schema of the paired node.
- Dynamic information about the actual nodes migrations and Lustre services distribution.

lustre_io_node table for each node of M(etadata) and I(/O) type gives

- Its paired node identity, pre-loaded at cluster install
- Its current migration status, maintained up to date by the failover management tools.

lustre_ost and **lustre_mdt** tables for each Lustre service give

- Its primary and secondary nodes loaded by the storage/Lustre deployment process
- Its currently supporting node (active) dynamically set by the failover management tools.

This information can be accessed and if necessary very carefully updated using the standard **lustre_tables_dba** tools.

Note:

The **mds_ha_node** and **oss_ha_node** are initialized by the contents of the **lustre_io_node** tables.

13.9.2 LDAP Directory – the lustre_ldap Utility

The Lustre LDAP directory contains the description of each Lustre file system currently installed on the cluster I/O nodes with its current services distribution on the cluster I/O nodes. It is located on the management station.

When a file system is started, it is noted as active in the LDAP directory enabling the management of the takeover of its services by the Lustre failover scripts of the I/O nodes. This is done automatically by the **lustre_util** utility.

The Lustre failover scripts check it each time a High-Availability event occurs on the nodes to get the Lustre services list they are supposed to act on. They update it with the result of the migration operations. A synchronization mechanism ensures the transfer of this information to the **ClusterDB**.

Configure and start the Lustre LDAP directory.

1. Create the **/var/lib/ldap/lustre** directory:

```
mkdir -p /var/lib/ldap/lustre
chown ldap.ldap /var/lib/ldap/lustre
```

2. Enable and start the LDAP service:

```
chkconfig --level 345 ldap
service ldap start
```

3. In the **/etc/sudoers** configuration file, verify that the ldap user has access to the **lustre_tables_dba** commands:

```
Cmnd_Alias LUSTRE_DB=/usr/sbin/lustre_ost_dba *, /usr/sbin/lustre_mdt_dba *,
/usr/sbin/lustre_io_node_dba *
ldap  ALL = NOPASSWD: LUSTRE_DB
```

   If not, use the **visudo** tool to update the **/etc/sudoers** file.

The management of the LDAP directory is performed using the **lustre_ldap** utility:

- Callback for **lustre_util** and the LDAP server for ClusterDB synchronization
- Online interface for the administrator to display the LDAP directory contents.

```
lustre_ldap show [-f <file_system_name>]
```

Display the current status of the file system as seen by the High-Availability system:

- **active** for a started file system,
- **unactive** for a stopped file system.

Without any parameters, the command will show the status of all the file systems loaded in the LDAP directory.

```
lustre_ldap list [-f <file_system_name>]
```

List the LDAP descriptor of the file system in LDIF format. ithout any parameters, the command will list all the file systems names loaded in the LDAP directory. Regarding the LDIF format of the display, it is mainly useful for maintenance process.

Important:

lustre_ldap can also be used to update the LDAP directory contents for punctual corrections. This has to be done very carefully provided the failover information consistency could be broken.

13.9.3 Failover Tools Configuration – the /etc/lustre/lustre.cfg File

Edit the configuration file of the management tools **/etc/lustre/lustre.cfg** , to include the modifications below.

To use failover file systems, set **LUSTRE_LDAP_URL** according to the name of the Management Node (ldap://<mgmt node>/).

To enable the failover tools trace feature, set **LUSTRE_DEBUG** to "yes".

Verify that **LUSTRE_NET** is set according to the cluster type. By default this will be set to **tcp** and it may be necessary to change it to **elan** or **02ib**.

On each I/O node, the Lustre failover scripts will log events in the **/var/log/lustre** directory. On the management station, the **lustre_ldap** daemon will log events in the **/tmp/log/lustre** directory.

13.9.4 Managing Lustre Failover Services on I/O and Metadata Nodes – the lustre_migrate Tool

Lustre failover services are used by the Cluster Suite to control the Lustre OST/MDT services migration.

Warning:

The failover services have to be started before the Lustre file systems are started. They can be stopped only when all Lustre file systems are stopped

The **lustre_migrate** command allows the failover Lustre services on the cluster to be managed.

Without any parameters, the command acts on all the I/O and metadata nodes.

Failover Services start/stop/status

```
lustre_migrate hastat [-n <node_list>]
```

Display the status of the Lustre failover services on the I/O and metadata nodes of the list. Without any parameters, it acts on all the I/O and metadata nodes.

```
lustre_migrate hastart [-n <node_list>]
```

Start the Lustre failover services on the I/O and metadata nodes of the list. Without any parameters, it acts on all the I/O and metadata nodes.

```
lustre_migrate hastop [-n <node_list>]
```

Stop the Lustre failover services on the I/O and metadata nodes of the list. Without any parameters, it acts on all the I/O and metadata nodes.

Failover Services Migration Control

For maintenance purposes, it may be useful to migrate Lustre services of a node to its HA paired, so that the node can be stopped without disturbing the Lustre system.

```
lustre_migrate export -n <node_name>
```

Stop the **lustre_<node_name>** failover service for which the node <node_name> is primary, and restart it on its secondary node. The secondary node information is taken from the ClusterDB. If the **lustre_<node_name>** failover service was already running on its secondary node, the command has no effect.

To relocate a failover service on its primary node once it is repaired:

```
lustre_migrate relocate -n <node_name>
```

Stop the **lustre_<node_name>** failover service for which the node **<node_name>** is primary, on its secondary node and restart it on its primary node. The secondary node information is taken from the **ClusterDB**. If the **lustre_<node_name>** failover service was already running on its primary node, the command has no effect.

13.9.5 Configuring File Systems for Failover

Configuring file systems for failover means configuring two paired OSS/MDS as possible support for each OST/MDT, one being the primary node, the other being the secondary node.

The failover feature is declared in the **/etc/lustre/models/<file_system_name>.lmf** model file. In the file system model update the following parameters:

- **failover=yes** enables failover configuration generation,
- **timeout=<xx>** sets the recovery time-out value. This time-out is used by Lustre to manage its recovery process. Recommended value = 60.

A file system is then described as usual, composed of one MDT and several OSTs taken from the ClusterDB.

The secondary node declared in the ClusterDB will be taken in account for a second OST/MDT declaration.

The file system is then managed using the **lustre_util** utility in a standard way. The failover specificity (LDAP directory interaction, status display, alternative mount , etc.) being automatically supported by the Lustre management tools.

13.10 High Availability Management without a Management Node

This mode of management is restricted to "small" clusters for which, as a result of their size, a management station is not necessary and an alternative solution, less automatic but lighter, is acceptable.

A "small" cluster means a cluster with a reduced number (1 or 2) pairs of I/O mixed **MDS** and **OSS** nodes.

- This cluster has no dedicated administrative network. The backbone network or the interconnect, if it is of Ethernet type, may be used for administrative purposes.
- The **MDS** pair of nodes are configured and used as a "*central point*" of management for Lustre.
- The cluster and **LDAP** databases with their associated tools are not available. They are replaced by a zone on the MDS nodes shared storage partition which manages Lustre's configuration and status information. This information is saved regularly in ext3 files.
- The only remaining Lustre management tools are **lustre_util** for Lustre file systems management and **lustre_migrate** for Lustre services High Availability management.
- Errors are reported in the local **syslog-ng** log files.

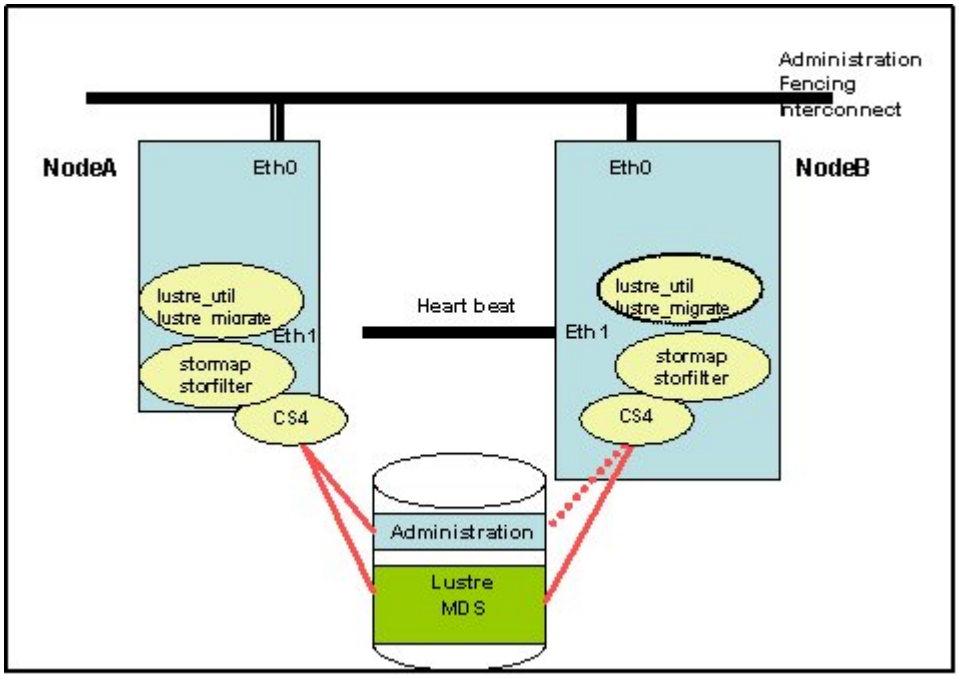


Figure 13-9 HA for a small cluster

All I/O nodes must be able to connect to each other using **SSH**. This means that the **RSA** keys must be installed on all the nodes.

13.10.1 I/O Nodes Pair Configuration

The automatic mode is not available for I/O nodes pair configuration, so each pair of I/O nodes has to be manually configured for HA. The Cluster Suite configuration file, **/etc/cluster/cluster.conf** must be customized and installed for each node. Both I/O nodes of the same pair use the same **cluster.conf** file.

For each pair of nodes:

- Select the appropriate **cluster.conf** template file – see note below - from the **/etc/storageadmin** directory.

Note:

cluster.conf.tpl files are to be used for NovaScale 5xxx and NovaScale 6xxx machines. **cluster.conf.tpl2** files are to be used for NovaScale 40xx and IPMI machines.

- Uncomment the **/var/lustre/status** file system declaration.
- Set a different cluster name for replacing the **HA_CLUSTER_NAME** keyword.
- Replace each keyword by the appropriate value (keywords are self-explanatory).
- Copy the **/etc/cluster/cluster.conf** on both nodes of the HA pair.

Note:

If the pair of I/O nodes is of the **Novascale 40xx** type, additional configuration is necessary for the supplementary Cluster Suite heart-beat Ethernet interface, commonly connected to the eth1 interface

On both paired nodes, configure and start the eth1 interface using the following IP addresses in accord with what is configured in the **/etc/cluster/cluster.conf** file:

<primary node> IP@10.0.0.1
<secondary node> IP@10.0.0.2

Declare the additional interfaces in the **/etc/hosts** table to enable Cluster Suite to find them. Add the lines:

```
# For Cluster Suite Heart_Beat
     10.0.0.1 <primary_node_hostname>_hb
     10.0.0.2 <secondary_node_hostname>_hb
```

The **_hb** suffix is used to differentiate easily between the Ethernet interfaces on the nodes.

The **eth0** interconnect should be used when connecting 2 Ethernet interfaces which include Lustre services. Similarly, if the interconnect is accessed using eth0 then it has to be configured using Lustre as this will use 2 interfaces by default.

On each I/O node, update the **/etc/lustre_modprobe.conf** configuration file:

```
options lnet networks=elan0,tcp0(eth0)
```

Lustre failover scripts are self-customized from the generic **/usr/sbin/lustre_failover** file by the way of symbolic links. On both I/O nodes create the following links:

```
ln –s /usr/sbin/lustre_failover
/usr/sbin/lustre_failover_<primary_node_hostname>

ln –s /usr/sbin/lustre_failover
/usr/sbin/lustre_failover_<secondary_node_hostname>
```

13.10.2 Central Point Configuration: MDS Pair

Prepare an **ext3** file system on a partition of the shared storage:

- Create the **/var/lustre/status** directory and add the following line to the **/etc/fstab** file on both **MDS** nodes:
  `/dev/sd<xx> /var/lustre/status ext3 sync,noauto,data=journal 0 0`
- Mount the **/var/lustre/status** device on the primary **MDS** node.

The I/O nodes connection configuration and migration status is defined by the **lustre_io_nodes** template in the **/etc/lustre** directory. This file has to be manually initialized by the administrator with the information concerning the paired connections. It is updated by the Lustre failover scripts when node migrations occur. It must contain one descriptor per node. Both paired nodes have their own descriptor:

```
NODE_NAME=<node_hostname>
NODE_HA_NAME=<paired_node_hostname>
LUSTRE_STATUS=OK or MIGRATED – must be initialized to OK
```

To initialize the **lustre_io_nodes** template, the administrator will:

- Copy the template sample from **/etc/lustre** directory to the **/var/lustre/status** directory mounted on the **MDS** primary node
- Initialize the node descriptors using his favourite editor

The Lustre management configuration file **/etc/lustre/lustre.cfg** provides the environment variables for nodes to connect to the "**central point**" for Lustre management. It has to be spread over all the I/O nodes.

- Update the file **/etc/lustre/lustre.cfg** contents so that it is possible to run Lustre without a management station:
  - **CLUSTERDB=no**
    sets the mode **DBless** for the Lustre management tools
  - **LUSTRE_ADMIN_NODE=<primary_MDS_hostname>**
    gives the central management point address
  - **LUSTRE_ADMIN_NODE2=<secondary_MDS_hostname>**
    gives the secondary management address
  - **LUSTRE_DEBUG=yes**
    turns on the logging mode for Lustre failover scripts
- Sends the file to all the I/O nodes using the **pdcp** utility.

13.10.3 High Availability Operations

These operations are conducted from the "*central point*" of management, identified by the **/var/lustre/status** mountpoint.

The **/var/lustre/status** mountpoint is placed under the control of the Cluster Suite in association with the primary MDS service. This means that it is automatically mounted on the secondary node for node migration. When migrated, the backup **MDS** becomes the central active management point.

Cluster Suite activation is performed on each node using the **storioha –c start** command. It can be parallelized from the central management point using the **pdsh** utility.

The Lustre failover services are started using the **lustre_migrate hastart** command.

From this point on, Lustre file systems are managed in a standard way using the **lustre_util** and **lustre_migrate** utilites, only.

13.11 Lustre High Availability Operations

13.11.1 Service Migration triggered by Cluster Suite

This process is conducted on node failure detection by the High-Availability system.

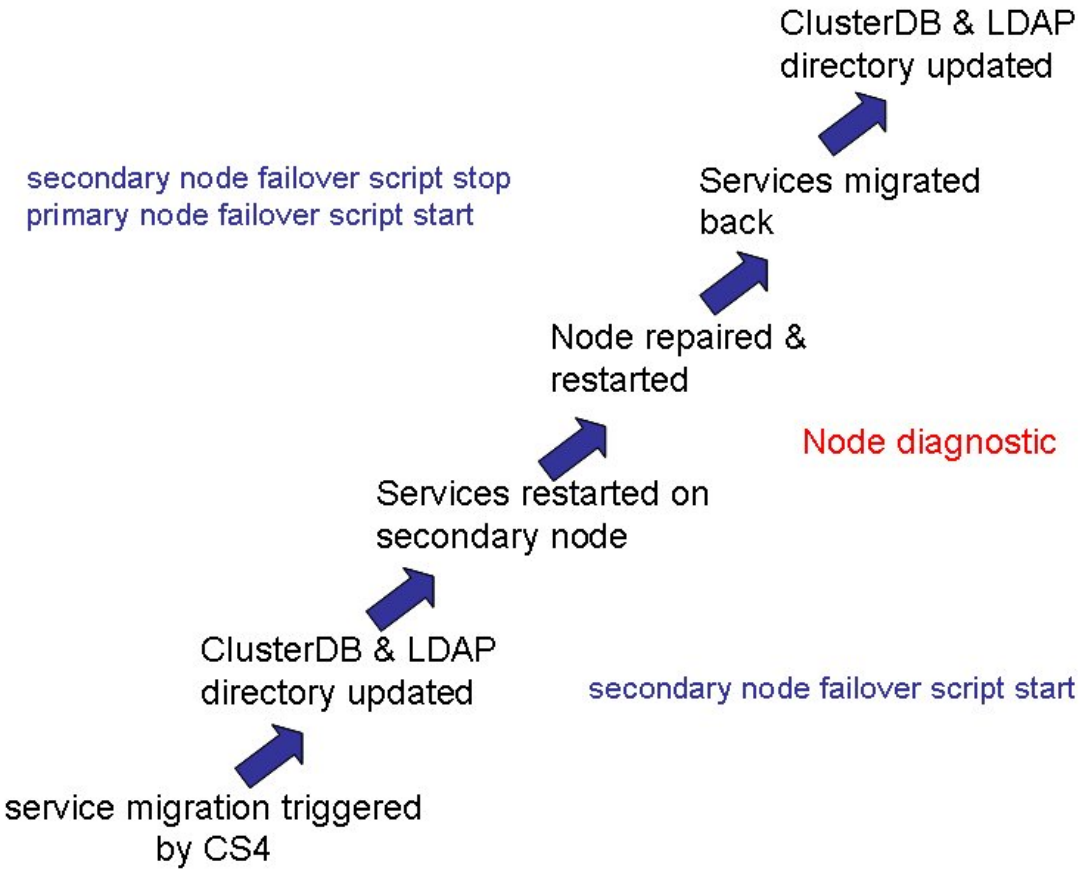


Figure 13-10 Service migration triggered by Cluster Suite

13.11.2 Service Migration triggered by Administrator

This process is conducted when the administrator needs to insulate a node without stopping the Lustre system.

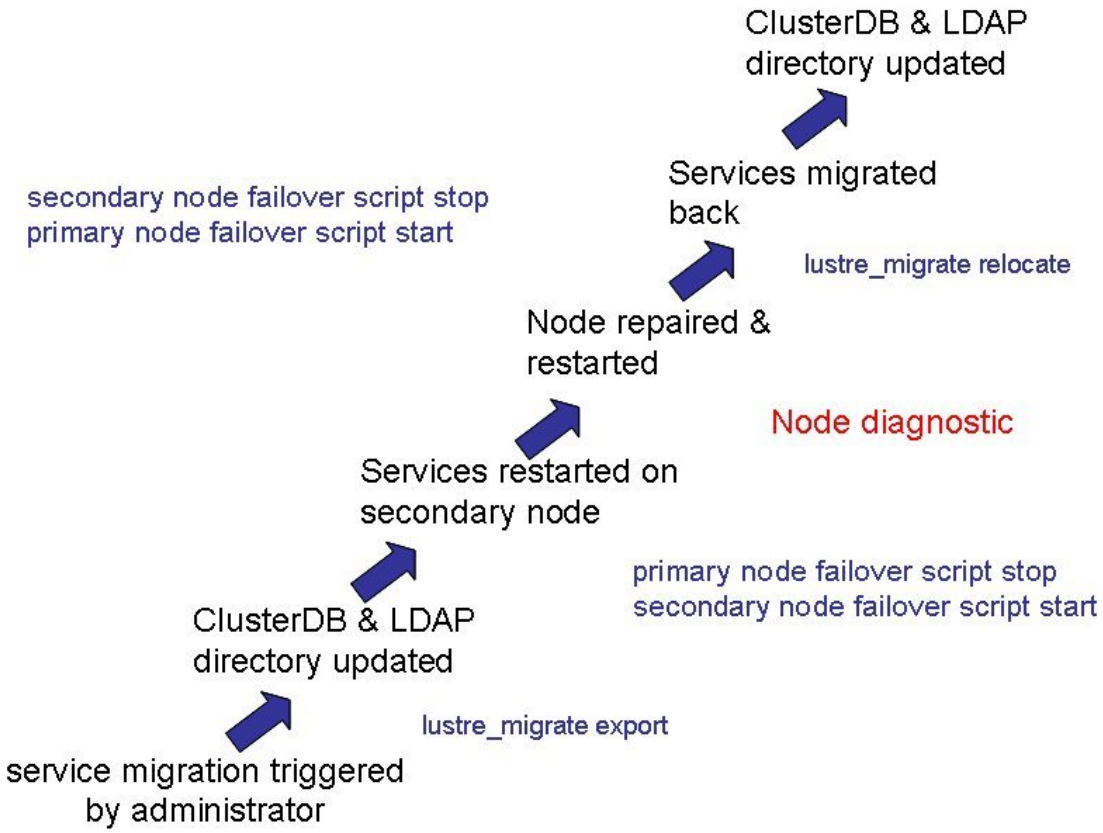


Figure 13-11 Service migration triggered by the Administrator

13.12 Monitoring Lustre High Availability

Two approaches are available from the monitoring tools: nodes migrations and the resulting OST/MDT file systems actual distribution.

On line commands allow the administrator to get an instant status of the Lustre High-Availability system.

If the cluster has a management node, important global health indicators are available via **NovaScale Master - HPC Edition** main view. They constitute a warning system for the administrator.

A trace system can be activated for debug and problem resolution purposes.

13.12.1 Command Line Monitoring

The following command displays the current failover paired nodes status under the form of an array with one line for each pair of nodes, as follows:

```
lustre_migrate nodestat
```

```
node name    node status    node HA name    node HA status
ns6          OK             ns7             MIGRATED
```

For each node, the status is that of the Lustre failover service it is primary for:

KO the Lustre failover service is UP

WARNING the Lustre failover service is UP some Lustre services are missing. A node migration may be in progress

MIGRATED the Lustre failover service has successfully migrated to the paired node and is now running on it

CRITICAL the Lustre failover service is no longer operating. The node migration has failed

The following command displays the current Lustre failover services distribution and status as seen by the Cluster Suite.

```
lustre_migrate hastat
```

For each Lustre file system installed on the cluster, the following command displays the detailed distribution of the MDTs and OSTs.

```
lustre_util info -f <File system name>
```

13.12.2 Graphic Monitoring

The Graphic Monitoring feature is available only if the cluster has a management node.

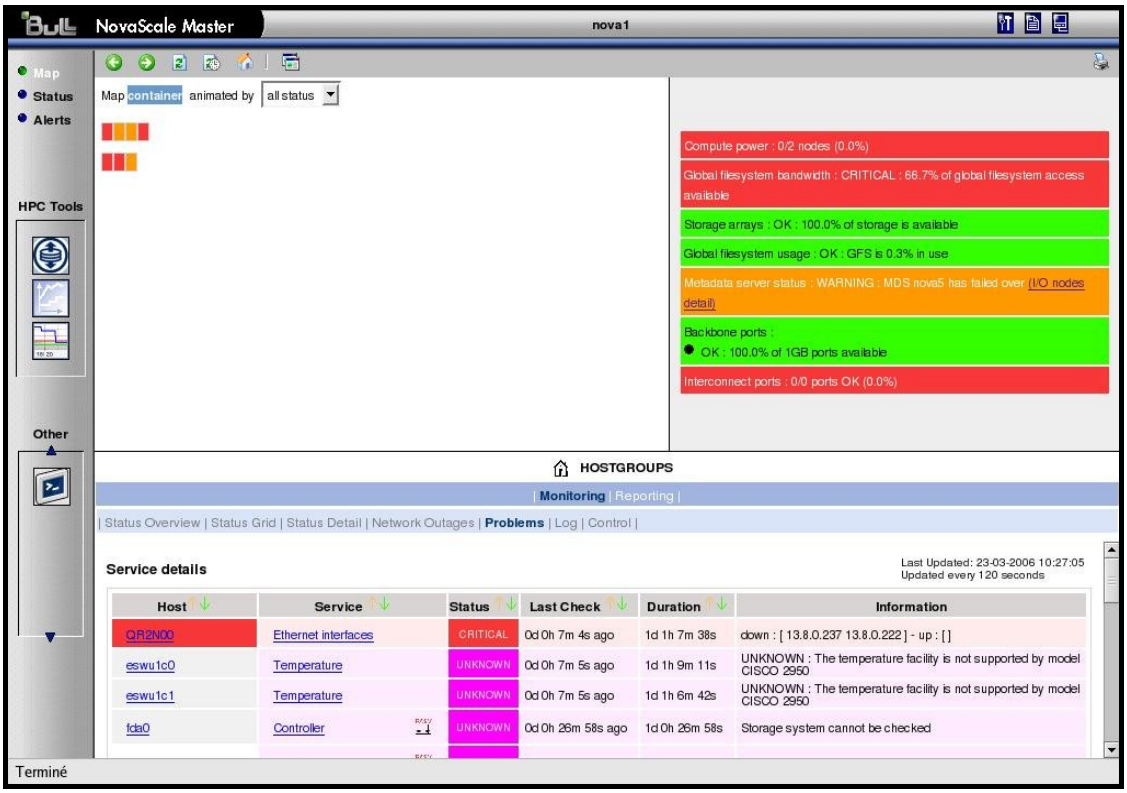


Figure 13-12 NovaScale Master Map all status screen

The I/O pairs status alert indicates if a migration of the metadata server has occurred. In this case, the Lustre system is no longer Highly -Available and an intervention is highly urgent.

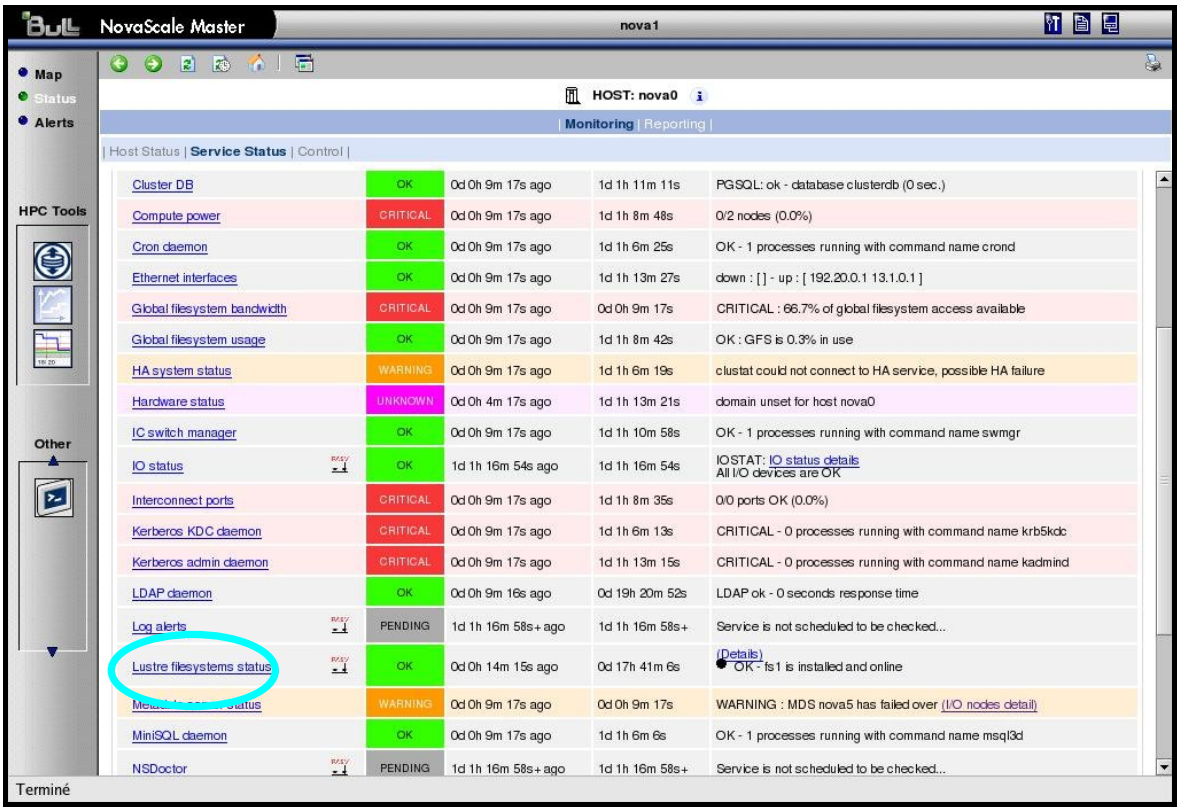


Figure 13-13 Lustre filesystem status indicator in the Host service status window

The Lustre file system indicator warns about failures. Clicking on the info link will display MDTs/OSTs detailed status.

13.12.3 Traces and Debug

Failover Tools Traces

These enabled by setting the LUSTRE_DEBUG parameter of the **/etc/lustre/lustre.cfg** file to yes.

On the management station, a daily log file, for example **//tmp/log/lustre/LDAP-<dd mm>.log**, is recorded under the **/tmp/log/lustre** directory by the **lustre_ldap** daemon. It gives information about migration events transmitted to the LDAP directory.

On the I/O and metadata nodes, a daily log file is recorded under the **/var/log/lustre** directory by the **lustre_failover** scripts. It gives information about failover events and their management.

System Log Files

On each I/O and metadata node, the Cluster Suite and the failover scripts log events in the **/var/log/messages** and the **/var/log/syslog** files. These files are centralized on the management station by the **syslog-ng** system.

Glossary and Acronyms

A

ACL

Access Control List.

B

Bisectional Bandwidth

The bandwidth flowing through a fabric while half the nodes send and receive a full duplex stream of data to the other half of the nodes.

BLBS

Bull Load-Balancing System makes it possible to identify machines within an **RMS** partition which have low loads automatically.

C

Cell

The set of nodes linked to the same PortServer and Ethernet switch. There are 4 cells in 1 unit. (See also *Unit*).

CGI

Common Gateway Interface.

ConMan

A management tool, based on telnet, enabling access to all the consoles of the cluster.

Cron

A UNIX command for scheduling jobs to be executed sometime in the future. A cron is normally used to schedule a job that is executed periodically - for example, to send out a notice every morning. It is also a daemon process, meaning that it runs continuously, waiting for specific events to occur.

Cygwin

A Linux-like environment for Windows. The Bull cluster management tools use Cygwin to provide ssh support on a Windows system, enabling access in command mode from the Cluster management system.

D

DDN S2A

DataDirect Networks S2A

DNS

Domain Name Server. A server that retains the addresses and routing information for TCP/IP LAN users.

E

EFI

Extensible Firmware Interface.

G

Ganglia

A distributed monitoring tool used to view information associated with a node, such as CPU load, memory consumption, network load.

GID

Group ID.

GPT

GUID Partition Table.

H

HBA

Host Bus Adapter.

Hyper-Threading

Hyper-Threading technology is an innovative design from Intel that enables multi-threaded software applications to process threads in parallel within each processor resulting in increased utilization of processor execution resources. To make it short, it is

to place two logical processors into a single CPU die.

HPC
High Performance Computing.

K

KDC
Key Distribution Centre.

KDE
Kool Desktop Environment.

KSIS
Utility for image building and development.

L

LDAP
Lightweight Directory Access Protocol.

LKCD
Linux Kernel Crash Dump. A tool capturing and analyzing crash dumps.

LOV
Logical Object Volume.

Lustre
Parallel file system managing the data shared by several nodes.

LVM
Logical Volume Manager.

M

MIB
Management Information Base.

MDS
MetaData Server.

MDT
MetaData Target.

MkCDrec
Make CD-ROM Recovery. A tool making bootable system images.

MPI
Message Passing interface.

MTBF
Mean Time Between Failures.

N

Nagios
A powerful monitoring tool, used to monitor the services and resources of Bull HPC clusters.

NFS
Network File System.

NIC
Network Interface Card.

NTP
Network Time Protocol.

O

OpenSSH
Open Source implementation of the SSH protocol.

OSC
Object Storage Client.

OSS
Object Storage Server.

OST
Object Storage Targets.

P

PAM

Platform Administration & Maintenance.

PAP

Platform Administration Processor (Bull NovaScale platforms).

PDSH

A parallel distributed shell.

Q

QBB

Quad Brick Board – The QBB is the heart of the **NovaScale 5xxx/6xxx Series** platforms, housing 4 Itanium™ 2 processors and 16 DIMMs.

R

RMS

Resource Management System. Manages the cluster resources.

S

SAN

Storage Area Network.

SIS

System Installation Suite.

SLURM

Simple Linux Utility for Resource Management.

SSH

Secure Shell. A protocol for creating a secure connection between two systems.

Syslog-ng

Syslog New Generation, a powerful system log manager.

T

TGT

Ticket-Granting Ticket.

TORQUE

Tera-scale Open-source Resource and QUEue manager. A batch manager controlling and distributing the batch jobs on compute nodes.

U

Unit

Generally it is the set of nodes linked to the same Quadrics switch. One unit contains 4 cells. (See also Cell).

UID

User ID

V

VNC

Virtual Network Computing. It is used to enable access to Windows systems and Windows applications from the Bull NovaScale cluster management system.

W

WWPN

World Wide Port Name- a unique identifier in a Fibre Channel SAN.

X

XFS

eXtended File System.

Index

/

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

Technical publication remarks form

Title:	HPC BAS4 Administrator's Guide

Reference:	86 A2 30ER 11

Date:	December 2007

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please include your complete mailing address below.

NAME: _____ DATE: _____

COMPANY: _____

ADDRESS: _____

_____

Please give this technical publication remarks form to your BULL representative or mail to:

Bull - Documentation Dept.
1 Rue de Provence
BP 208
38432 ECHIROLLES CEDEX
FRANCE
info@frec.bull.fr

Technical publications ordering form

To order additional publications, please fill in a copy of this form and send it via mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone: +33 (0) 2 41 73 72 66
FAX: +33 (0) 2 41 73 70 66
E-Mail: srv.Duplicopy@bull.net

Reference	Designation	Qty
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
[_ _] : The latest revision will be provided if no revision number is given.		

NAME: ______________________________ DATE: ____________

COMPANY: ______________________________

ADDRESS: ______________________________

____

PHONE: ____________________ FAX: ____________________

E-MAIL: ______________________________

For Bull Subsidiaries:
Identification: ______________________________

For Bull Affiliated Customers:
Customer Code: ______________________________

For Bull Internal Customers:
Budgetary Section: ______________________________

For Others: Please ask your Bull representative.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 A2 30ER 11