Bull
HPC BAS3

Administrator's Guide

86 A2 31EM Rev. 02

Bull
HPC BAS3

Administrator's Guide

Subject: HPC Administrator’s tasks and tools.

Special Instructions: Refer to SRB.

Software Supported: HPC BAS3 V2

Software/Hardware required: Refer to SRB.

Date: November 2005

Bull S.A.
CEDOC
Atelier de reprographie
357, Avenue Patton BP 20845
49008 ANGERS Cedex 01
FRANCE

86 A2 31EM Rev02

Your suggestions and criticisms concerning the form, contents and presentation of this manual are invited.
A form is provided at the end of this manual for this purpose.

86 A2 31EM Rev02

Preface

Scope and Objectives

The purpose of this guide is to explain how to configure and administrate Bull High Performance Computing (HPC) cluster, 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 documentation of standard Linux distribution.

Intended Readers

This guide is for administrators.

Prerequisites

The installation of the all hardware and software components of the HPC must be completed.

Structure

This guide is organized as follows:

- Chapter 1 explains *General Concepts* for Bull Linux HPC.
- Chapter 2 describes the *Administration Tools* that can be used on a Bull HPC. These tools enable to manage system consoles, to run parallel commands, to install software, to control and monitor the Bull HPC.
- Chapter 3, *HPC Configuration*, enables the administrator to run the basic configuration tasks.
- Chapter 4 describes the *File Systems* that can operate on a Bull HPC (among which NFS and Lustre), and how to configure them.
- Chapter 5 deals with *Batch and Resource Management* tools (TORQUE, RMS).

Bibliography

- Bull HPC BAS3 Installation and Configuration Guide (86 A2 31EG)
- Bull HPC BAS3 User's Guide (86 A2 30EM)
- A Software Release Bulletin (SRB) provides release-specific information and installation instructions (86 A2 31EJ).

Syntax Notation

Commands entered by the user and messages displayed by the system to illustrate explanations are in a frame in "Courier" font. Example:

```
BIOS Intel
```

Commands, files, directories and other items whose names are predefined by the system are in "Bold". Example:

The **/etc/sysconfig/dump** file.

Table of Contents

1. General Concepts

2. Administration Tools

3. HPC Configuration

4. File Systems

5. Resource and Batch Management (TORQUE and RMS)

1. General Concepts

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 is composed of the following elements:

- **Compute nodes** for intensive calculation
- **Input/Output (I/O) nodes** to store and provide data in storage units
- **Management node** to manage the cluster
- **High speed interconnect switch** and **boards** to transfer data between compute nodes and I/O nodes
- **Ethernet** and **serial networks** are used for cluster management and maintenance.

Cluster systems are generally on a private network so that each system can trust the other systems in the cluster. This configuration enables to manage the nodes as a whole, instead of individually.

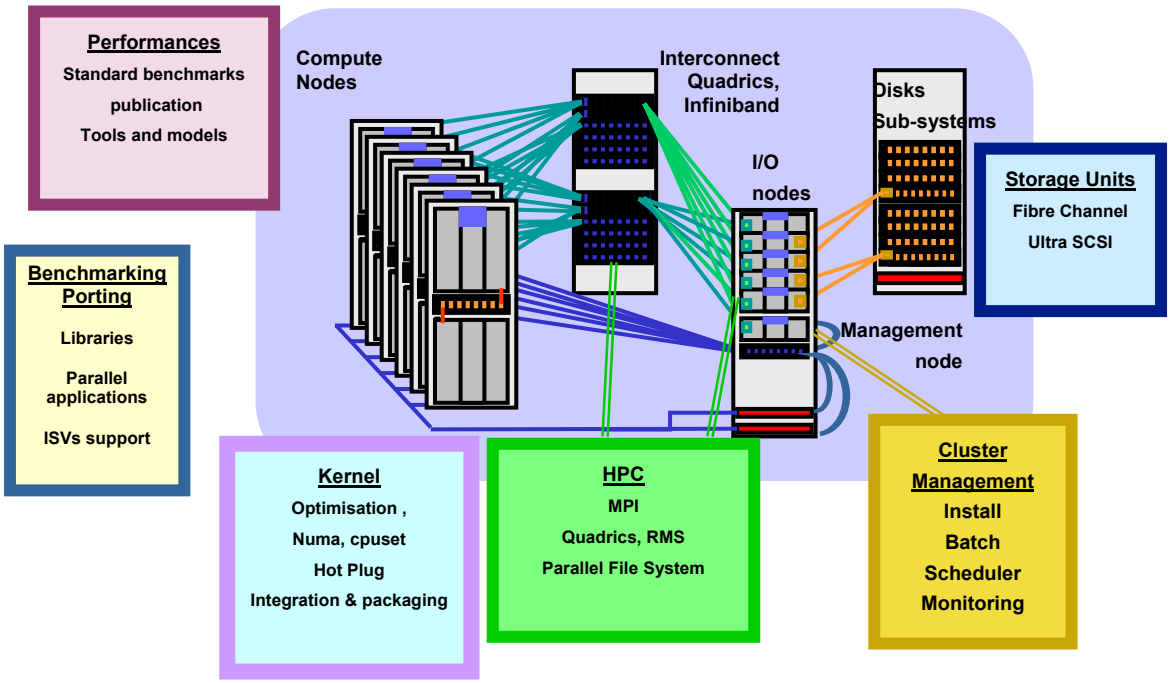
To benefit from the advantages of a cluster configuration, Bull provides a software environment, which makes the cluster very efficient. For example:

- A **Global File System** (**Lustre** from CFS) aggregates your distributed storage into a single group- or enterprise-wide file system.
- The **Message Passing Interface** (**MPI**) allows to run programs across all nodes.
- The **RMS** Resource Management System and **TORQUE** batch manager manage and control access to the distributed resources.

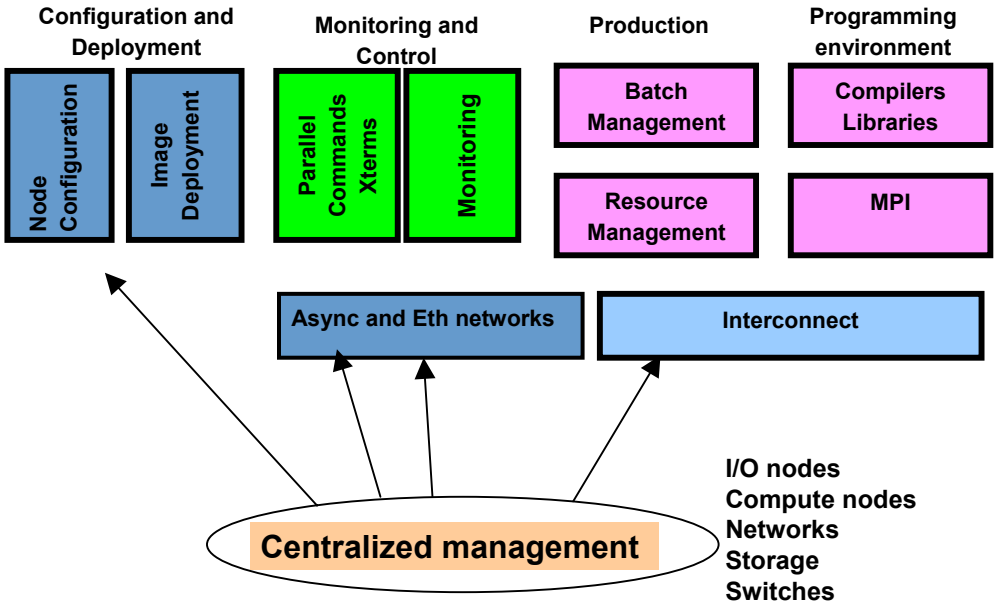
The Bull cluster administration offer is centralized on a node. All management products are running on this platform called the management node. All nodes are

controlled and monitored from this central point of management with the objective to minimize CPU activity and network traffic on compute and I/O nodes.

The management tools are mainly Open Source products with Bull added value. These products are configurable and adaptable to the management needs.



Developments have been done to adapt these products to the Bull platforms and BAS3 environment. All management functions are available thru a web interface or in remote command mode. The users can access a management function according to their profile.



1.2 Management Functions and Corresponding Products

Each management function is performed by a specific product, listed below.

Hardware management

- **ConMan** lets the administrator access the System consoles thru Async Network.

Configuration management

- **System Installation Suite** (**SIS**) and Ethernet Network enable the deployment of images.

System management

- **pdsh** is used to run parallel commands.
- **Nagios** monitors the cluster status.
- **Ganglia** manages cluster activity.
- **top**, **top_pfb** and **Ganglia** provide cluster performances and behavior.
- **syslog-ng** manages the System Logs.

- **mkCDrec** performs System Backup and Restore. This function is available for the management node.
- **LKCD** captures and analyzes crash dump.

Resource management

- **RMS** from Quadrics provides Cluster Resource management.
- **TORQUE** is responsible for Batch management.

2. Administration Tools

A set of administration tools is provided with a Bull HPC cluster. These tools let you centralize the cluster administration on a management node. All the nodes can thus be managed from a single spot. Bull HPC can manage hundreds of nodes within a cluster, as administration tools have been optimized in terms of CPU consumption and data exchange.

The administration tools are mainly Open Source software. The client part of the software is installed on the compute nodes and I/O nodes.

The administration functions are available most often through a WEB interface, or in remote command mode. Access requires specific user rights and is based on secured shells and connections. The administrator can access all monitoring and surveillance functions, while users can only view system information.

2.1 System Administration Tools

The system administration tools help the administrator to monitor the activity and behavior of the cluster, to maintain the cluster in good working order, to diagnose and, if possible, to resolve abnormal behavior.

A control and monitoring tool provides useful information; for example:

- one of the cluster nodes is not working
- the network is overloaded
- how the cluster is used…

This section describes **Nagios** and **Ganglia**, which are tools used in Bull HPC clusters.

2.1.1 Nagios

Nagios is a host and service monitor designed to inform the administrator of problems. The monitoring daemon runs intermittent checks on hosts and services you specify using external "plugins" which return status information to Nagios.

When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email for example). Current status information, historical logs, and reports can all be accessed via a web browser.

2.1.1.1 Surveying and Sending Alerts

For Nagios each element of the cluster is an object which can be surveyed. Every collected information is associated with an object and one of its functions. Collected information indicates the objects state and can provide diagnostic in case of incident. It is thus possible to survey all the cluster nodes.

Nagios provides also an alert mechanism. The administrator can configure the alerts that will activate warnings, and the action to trigger when they occur.

2.1.1.2 Displaying Cluster State

Nagios provides a graphical interface, in which the cluster objects are displayed in a hierarchical way on the screen, and grouped by type. Another part of the screen displays a view of the selected object.

The general view provides in real time the state of all the cluster elements (load level, resource usage …). The alerts are displayed and their origin can be quickly identified.

For more information, please refer to www.nagios.org .

2.1.2 Ganglia

Ganglia is a scalable distributed monitoring system for HPC systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state.

Ganglia enables to view monitoring information associated with a node, such as CPU load, memory consumption, or the network load (output, packets per second…). Ganglia also provides graphical representations of statistics about resource consumption, such as CPU average consumption for one or several days.

The ganglia system is comprised of two unique daemons, a PHP-based web frontend and a few other small utility programs.

2.1.2.1 Ganglia Monitoring Daemon (gmond)

gmond is a multi-threaded daemon which runs on each cluster node you want to monitor. **gmond** has four main responsibilities: monitor changes in host state, multicast relevant changes, listen to the state of all other ganglia nodes via a multicast channel and answer requests for an XML description of the cluster state.

2.1.2.2 Ganglia Meta Daemon (gmetad)

Federation in Ganglia is achieved using a tree of point-to-point connections amongst representative cluster nodes to aggregate the state of multiple clusters. At each node in the tree, a Ganglia Meta Daemon (**gmetad**) periodically polls a collection of child data sources, parses the collected XML, saves all numeric, volatile metrics to round-robin databases and exports the aggregated XML over a TCP sockets to clients. Data sources may be either **gmond** daemons, representing specific clusters, or other **gmetad** daemons, representing sets of clusters. Data sources use source IP addresses for access control and can be specified using multiple IP addresses for failover. The latter capability is natural for aggregating data from clusters since each **gmond** daemon contains the entire state of its cluster.

2.1.2.3 Ganglia PHP Web Frontend

The Ganglia web frontend provides a view of the gathered information via real-time dynamic web pages. Most importantly, it displays Ganglia data in a meaningful way for system administrators and computer users. Although the web frontend to ganglia started as a simple HTML view of the XML tree, it has evolved into a system that keeps a colorful history of all collected data.

The Ganglia web frontend caters to system administrators and users. For example, one can view the CPU utilization over the past hour, day, week, month, or year. The web frontend shows similar graphs for Memory usage, disk usage, network statistics, number of running processes, and all other Ganglia metrics.

The web frontend depends on the existence of the **gmetad** which provides it with data from several Ganglia sources. Specifically, the web frontend will open the local port 8651 (by default) and expects to receive a Ganglia XML tree. The web pages themselves are highly dynamic; any change to the Ganglia data appears immediately on the site. This behavior leads to a very responsive site, but requires that the full XML tree be parsed on every page access. Therefore, the Ganglia web frontend should run on a fairly powerful, dedicated machine if it presents a large amount of data.

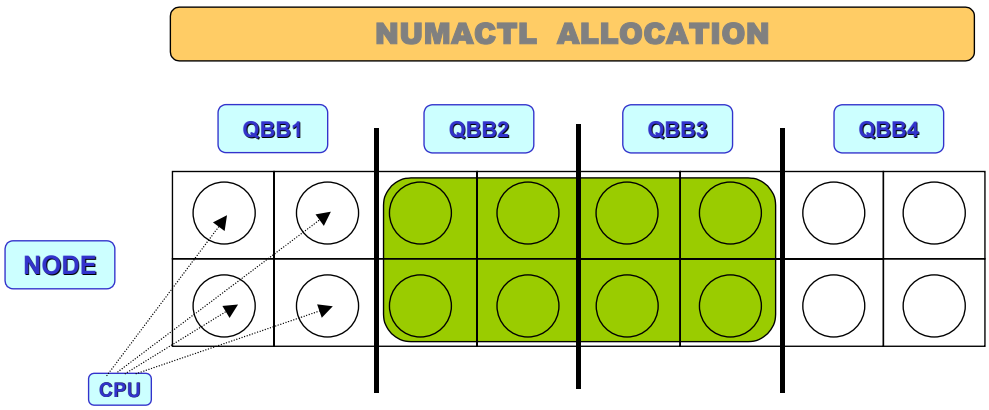
For more information, please refer to www.ganglia.sourceforge.net .

2.2 NUMACTL

2.2.1 Description

Numactl optimizes the allocation of CPU and memory for jobs on a single node NUMA SMP (Non Uniform Memory Access Symmetrical Multi Processor) architecture.

Numactl is dedicated to a **mono-numa** SMP system. The granularity level is the QBB (Quad Brick Board) in a node. Following is an example of a node with 16 CPUs:



❑

Numactl is able to define an execution area for an application: in this example QBB2 and QBB3 (2 * 4 CPUs) are allocated.

2.2.2 Using Libnuma, Numactl

IMPORTANT: *The scope of the Numactl command is a mono numa configuration, with 1 to 8 QBB that is to say, one node for a HPC cluster.*

In the paragraph concerning the numactl command the term "node" means a QBB in the numa configuration.

Libnuma is a library that offers a simple programming interface to the SMP NUMA policy supported by the Linux kernel. On a NUMA architecture, memory areas have different latency or bandwidth regarding which CPU they are accessed by. Available policies are page interleaving, preferred node allocation, local allocation, allocation only on specific nodes. It also allows to bind threads to

specific nodes. All policy exists per thread, but is inherited to children. For setting global policy per process it is easiest to run it using the numactl utility. For more finegrained policy inside an application this library can be used.

Numactl runs processes with a specific NUMA scheduling or memory placement policy. The policy is set for command and inherited by all of its children. In addition Numactl can set persistent policy for shared memory segments or files.

Most used policy settings are:

--interleave=nodes, -i nodes

Sets a memory interleave policy. Memory will be allocated using round robin on nodes. When memory cannot be allocated on the current interleave, target falls back to other nodes.

--membind=nodes, -m nodes

Allocates only memory from nodes. Allocation will fail when there is not enough memory available on these nodes.

--cpubind=nodes, -c nodes

Executes process only on the CPUs of nodes.

--localalloc, -l

Does always local allocation on the current node.

Example

To run a program which allocates memory with round robin allocation on the 4 nodes of a 16 CPU FAME, enter:

```
Numactl –i0,1,2,3 program_name
```

For more information refer to the installed **numa** man pages.

2.3 PTOOLS CPUSET

IMPORTANT: *PTOOLS CPUSET are integrated in TORQUE and MPI. If you need to use them for specific cases, do it carefully since a bad utilization could disturb TORQUE and MPI.*

2.3.1 Description

Ptools cpuset sets CPU on which a job can run (on a single node NUMA SMP architecture).

CPUSET is a feature of the Bull Linux kernel, which lets you define execution areas inside a multiprocessor system. It means that you can define, for each application, an execution area consisting of processors, to which the execution of the application will be limited. These execution areas are called *cpusets*.

The cpusets form a nested hierarchy. It means that cpusets can be created inside a cpuset.

The main purposes of the cpusets are the following:

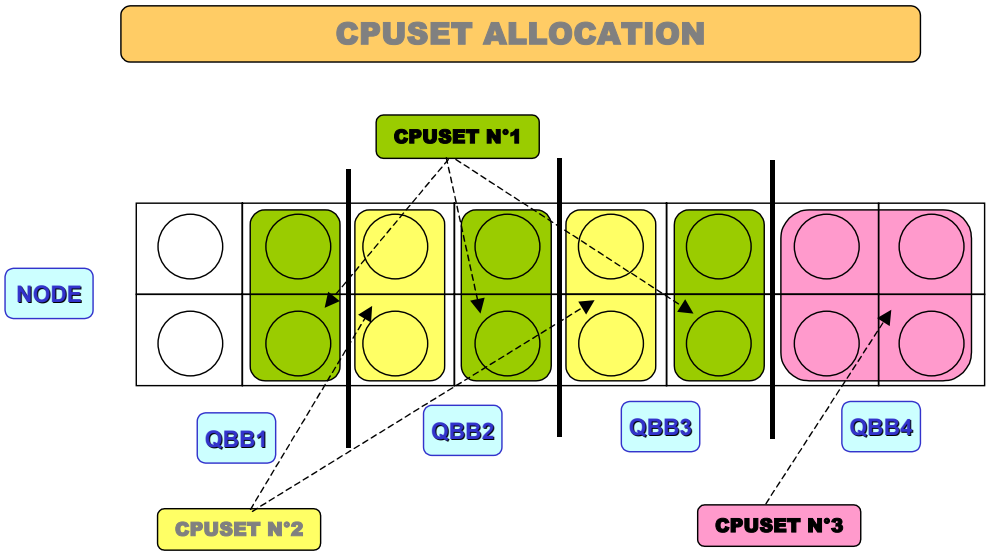
- To offer some kind of partitioning to multiprocessor systems
- To control the execution of an application for highest performance, especially on systems with a complex topology such as NUMA systems.

The cpusets also change the way you map processes on specific processors: when a task uses the **sched_setaffinity** system call, the list of processors given to this system call is translated to be understood *inside* the cpuset in which the application is running. For example, if an application running inside a cpuset with processors 4, 5 ,6 ,7, wants to bind one of its processes to the processor 0, the process will actually be bound to processor 4. This feature lets you run at the same time several applications for which you need to finely control on which processors their tasks are running.

Bull offers the **ptools** suite to control and use the cpusets.

> **Note:** The main difference between Numactl and Ptools is the granularity of the allocation: QBB for Numactl and CPU for CPUSET.

In the following example there are 3 cpusets:
cpuset N°1 with 2 CPUs on QBB1, 2 CPUs on QBB2 and 2 CPUs on QBB3,
cpuset N°2 with 2 CPUs on QBB2 and 2 CPUs on QBB3,
cpuset N°3 with 4 CPUs on QBB4.



2.3.2 Using Ptools

The **ptools** suite consists mainly of the following commands:

pcreate	create cpusets.
pexec	create a cpuset and run an application inside it. The cpuset is then destroyed when the application is completed.
passign	move a task inside a cpuset.
pdestroy	destroy a cpuset.
pls	list existing cpusets.

When a cpuset is created, a list of processors must be chosen.

Several flags can be set for each cpuset:

strict	also called **cpu_exclusive**. This cpuset will not share its processors with other cpusets that have the same parent cpuset.
autoclean	this cpuset will be automatically removed from the system, and its resources freed, when it becomes unused, (this means: when all the applications running inside this cpuset are completed).

Typical usage example

pexec –np <nb_cpus> --strict <my_app>

```
pexec -np 2 --strict ./myapp
Created /proc/cpusets/top_cpuset/cpuset1, id 7
Myapp running..
```

For more information refer to the **pexec**, **pcreate** and **passign** installed man pages.

2.4 Controlling Processes with the top Command

The **top** command provides different pieces of information, in real time, about the execution of processes and use of a node. The **top** command can:

- Provide a list of launched processes
- Show each process machine time from the most "greedy" down
- Determine which processes need a lot of resources
- Observe the system's life
- Indicate the amount of memory and percentage of swap used …

Refer to the *Bull HPC User's Guide* for more information about **top**.

2.5 Controlling Access to Nodes with the fping Command

fping is a tool to quickly ping N number of hosts to determine their reachability. By sending out pings in a round-robin fashion and checking on responses as they come in at random, a large number of hosts can be checked at once.

fping is a ping-like program which uses the Internet Control Message Protocol (ICMP) echo request to determine if a host is up. **fping** is different from **ping** in that you can specify any number of hosts on the command line, or specify a file containing the lists of hosts to ping. Instead of trying one host until it timeouts or replies, **fping** will send out a ping packet and move on to the next host in a round-robin fashion. If a host replies, it is noted and removed from the list of hosts to check. If a host does not respond within a certain time limit and/or retry limit it will be considered unreachable.

Unlike **ping**, **fping** is meant to be used in scripts and its output is easy to parse.

For details, see the **fping** man page.

2.6 Managing Consoles

The serial lines of the servers are the communication channel to the firmware (enabling access to EFI, Extensible Firmware Interface) and to the system low-level features of the system. This is why they play an important role for the system init surveillance or for taking control in case of crash or for debug.

To benefit from these functions, the serial lines are concentrated with Ethernet / Serial ports concentrators, making them available from the management node.

> **Note:** Storage Units may also offer console interfaces through serial ports, making possible configuration and diagnostics.

HPC uses **ConMan** as a console management tool. See *Using ConMan*, on page 2-10 for details.

2.6.1 Accessing the Serial Port of a Node

Use the following procedure to access the PortServer if you cannot use ConMan.

- Start a “konsole” window using the **/usr/bin/konsole** command.

  Note: the **/usr/bin/konsole** command is delivered in the **kdebase** rpm of KDE (Kool Desktop Environment) graphical environment.

- Change the keyboard parameter for the “konsole” window:
  Click the **Settings** button, then **Keyboard**, and select **VT100 (historical)**.
- For a better definition setup the font as follows:

  In the "konsole" window click the **Settings** button, then **Font** and select **Large**. Then, click **Size** and select **80x24** (VT100).

  You get a 80 colons x 24 lines window for the EFI commands. If necessary, fit the window using the corner arrow.
- From the “konsole” window start a **telnet** session on the serial port of the PortServer that is connected to the serial port of the node.

If Linux is running

In this case, telnet will have opened a session on the client node via its serial port.

If KDB is valid (**kdb=on** in the **elilo.conf** file), it can be called using an appropriate key sequence (Control-a, Esc KDB, etc ...).

If the node is under EFI control

In this case, telnet will have displayed EFI traces and menus in the “konsole” window. The keyboard is operational for these menus (arrows keys).

- Enter the BIOS menus.

  Note: In VT100 emulation, use **Backspace** for the **DEL** character, and use **Control-Backspace** for the **Backspace** character

Attention: hit the Backspace key as soon as “`Press DEL to run Setup`” message is displayed, since the network generates a delay.

- Use the Editor under EFI to edit a file. The following commands are available:

  F1 Open File
  F2 Save File
  F3 Exit
  F7 Search
  F8 Search/replace
  F9 File Type

  Hit the following keys to get the corresponding characters:
  Esc 1, Esc 2, Esc 3, Esc 7, Esc 8, Esc 9.

2.6.2 Using ConMan

The **ConMan** command allows the administrator to manage all the consoles (including server consoles and storage subsystem consoles) on all the nodes.

ConMan is installed on the management node.

ConMan maintains a connection with all the lines that it administers. It provides access to the consoles using a logical name. It supports the key sequences that provide access to debuggers or to dump capture (Crash / Dump).

ConMan works with TS16 Digiboard PortServer using telnet. Once each portserver is configured, **ConMan** can access all NovaScale, PAP (Platform Administration Processor) and disk arrays via their serial port.

The advantages of ConMan on a simple telnet connection are the following ones:

- Symbolic names are mapped on a physical serial line
- There is a log file for each machine
- It is possible to join a console session, or to steal it
- There are three modes for accessing the console: monitor (read-only), interactive (read-write), broadcast (write only).

2.6.2.1 conman Syntax

conman <OPTIONS> <CONSOLES>

-b	Broadcast to multiple consoles (write-only).
-d HOST	Specify server destination. [127.0.0.1:7890]
-e CHAR	Specify escape character. [&]
-f	Force connection (console-stealing).
-F FILE	Read console names from file.
-h	Display this help.
-j	Join connection (console-sharing).
-l FILE	Log connection output to file.
-L	Display license information.
-m	Monitor connection (read-only).
-q	Query server about specified console(s).
-Q	Be quiet and suppress informational messages.
-r	Match console names via regex instead of globbing.
-v	Be verbose.
-V	Display version information.

Once a connection is established, enter "**&.**" to close the session, or "**&?**" to display a list of currently available escape sequences.

Run **man conman** for more information.

2.6.2.2 Examples

- To connect to the serial port of NovaScale `bull47`, run the command:

```
conman bull47
```

- To connect to the serial port of singlet 1 of ddn 3, run the command:

```
conman ddn3_s1
```

- To connect to serial port PAP 3, run the command:

```
conman pap3
```

2.6.2.3 Configuration File

The **/etc/conman.conf** file contains the consoles managed by conman.

Run **man conman.conf** for more information.

2.7 Running Parallel Commands with PDSH

A distributed shell is a tool that launches the same command on several nodes. Such a function is essential in a cluster environment when you want to carry out instructions on several nodes without having to do it manually on each node in turn. Several tools enable this functionality.

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 clusters usage.

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

Three utilities are included in pdsh:

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

2.7.1 Security and Data Integrity

pdsh utility relies on the security and authentication mechanisms provided by ssh and / or Kerberos V4 layers on which it is configured.

2.7.2 pdsh Description and Syntax

Following are the most used syntaxes:

pdsh –R <shell> **-w** <node list> **-l** <user> **-Options Command**

pdsh –R <shell> **-a -x** <list of excluded nodes > **-Options Command**

pdsh –R <shell> **-g** <genders attributes> **-Options Command**

–R <shell>	selects the remote shell service type, called *rcmd module*. This option can be replaced by setting the **pdsh_rcmd-type** environment variable.
-w <node list>	specifies the nodes to which the command applies. Example: `ns[1-5,10]`
-a	specifies that the command applies to all nodes defined in the **/etc/machines** file.

-x <list of excluded nodes > specifies the nodes to exclude from the pre-defined list.

-g <genders attributes> specifies the nodes that have the attributes defined by genders description files (see genders documentation).

-l <user> specifies the user rights for executing the remote command, if they are different from the local user.

-Options for a complete description of the options, please refer to the **pdsh** man page.

Examples

To execute the **pwd** command on all the nodes of the cluster using 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, then to display the date on all nodes or to exclude a node, enter:

```
export PDSH_RCMD_TYPE=ssh;

pdsh -a date

tiger1: Mon Dec 13 13:44:48 CET 2004
tiger0: Mon Dec 13 13:44:47 CET 2004
tiger2: Mon Dec 13 13:44:47 CET 2004
tiger3: Mon Dec 13 13:44:46 CET 2004


pdsh -a -x tiger0 date

tiger1: Mon Dec 13 13:44:48 CET 2004
tiger2: Mon Dec 13 13:44:47 CET 2004
tiger3: Mon Dec 13 13:44:46 CET 2004
```

To sort the results and to group the nodes with same output, enter:

```
pdsh -a -x tiger0 date | dshbak -c

----------------
tiger[2-3]
----------------
 Mon Dec 13 14:10:41 CET 2004
----------------
tiger[0-1]
---------------
 Mon Dec 13 14:10:42 CET 2004
```

2.7.3 pdcp Description and Syntax

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 enabling to define the nodes to be reached are similar to those of pdsh.

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

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

2.7.4 dshbak Description and Syntax

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 in a more convivial way. Note that the results must be directed in a buffer file before being processed by **dshbak**.

dshbak [-c] <buffer_file>

dshbak performs 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** displays only once the results for all nodes that have the same results. In this case 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, **pdsh** output is re-directed to **res_rpm_qsnetmpi** file. Then, the **dshbak** command formats and displays the result:

```
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
```

2.8 Deploying Software with System Installation Suite (SIS)

A deployment tool is a software used to install a distribution or packages on several machines at once. For large clusters, such a tool is essential, since it avoids to make the same installation a large number of times.

System Installation Suite (SIS) is the deployment tool used on Bull HPC.

SIS makes easy to propagate software distribution, content or data distribution changes, operating system update and software update, through your network of Linux machines.

SIS is used to ensure safe production deployments. By saving your current production image before updating your new production image you have a highly reliable contingency mechanism. If the new production environment is found to be flawed, simply roll-back to the last production image.

For more information you can refer to *Bull HPC Installation and Configuration Guide*, which provides a comprehensive example.

2.9 Managing System Logs with syslog-ng

2.9.1 Cluster System Logs Overview

For security and follow-up reasons and to decrease the administration work due to the size of the cluster, all the system logs are centralized on the management node. There are two ways to return system logs information to the management node:

- The logs are collected on each node, using standard mechanisms for archival and log files permutation. Some utilities ensure compression, transfer and archival of these log files on the management node in asynchronous mode. A centralized operation is performed on the management node, in order to extract and search events according to criterion (date, type, gravity, …).

  This asynchronous process intends to make curative actions on the incidents that have occurred on the cluster.

- Some events are immediately reported to the management node. Filters are used, which specify the type and gravity level of the events that must be immediately transferred.

  This synchronous process gives instantaneously to the administrator a global and synthetic view of the current system events.

syslog-ng (Syslog New Generation) is the powerful system log manager used on Bull HPC to manage the cluster system logs. Compared with standard managers, syslog-ng includes new features:

- Powerful configuration
- Ability to filter messages based on content using regular expressions
- Encoding and authentication of the network traffic
- Forwarding logs on TCP and UDP
- Compressing logs.

2.9.2 Configuring syslog-ng

syslog-ng is installed on the cluster with a default configuration. The scripts used to transfer log files are also installed. The administrators can modify the default configuration according to their needs.

The **/etc/syslog-ng/syslog-ng.conf** file contains the configuration parameters for syslog-ng. This file is divided in five sections:

- **options** - general options
- **source** - event sources
- **destination** - log destinations

- **filter** - filter definitions
- **log** - actions to be processed on messages

2.9.2.1 options Section

You can configure general parameters in the options section. Following is an example:

```
# Start of options area

options {

 sync (0);              # Number of events before writing in the logs
 time_reopen (10);      # Wait 10s before reconnecting if the connection
                        failed. Used when logs are centralized through network
 #time_reap (number);   # Closes a log file that is not accessed after
                        "number" seconds
 log_fifo_size (1000); # number of event lines stored, before writing them.
                        Enables to quickly take the events into account
                        and to free the process that has generated them.
 long_hostnames (off); # Usage of long names
 use_dns (no)           # Usage of DNS to find addresses
 use_fqdn (no);                # Usage of machine short name
 owner("root");         # logs owner
 group("root");         # logs group
 perm("644");           # logs rights mask
 keep_hostname (yes);   #
 create_dir (yes);      # Create directories for log storage
 use_time_recv(no);     # Local time will be used instead of the time written
                        in the logs
 #gc_idle_threshold(100);      # The garbage collector is started after 100
                               events if syslog-ng is inactive.
 #gc_busy_threshold(100);      # The garbage collector is started after 3000
                               events if syslog-ng is active.
};
```

2.9.2.2 source Section

The source section defines the log sources among the following ones: network, local files, peripheral, pipe, stream.

Syntax:
Source <identifier> {source-driver(params); source-driver(params); ...};

For example, the following lines are suitable for a Linux system. They enable to read the **/dev/log** stream, to receive syslog-ng internal messages and to take the kernel starting messages:

```
source src {
        unix-stream("/dev/log");
        internal();
        file("/proc/kmsg");
};
```

The possible sources are the following ones:

unix-stream(<filename>) stream pipes (used in Linux)

file(<filename>) files data (Linux kernel messages for example)

pipe(<filename>) named pipes (for interfacing Nagios for example)

tcp(<ip>,<port>) and **udp(<ip>,<port>)**
to listen on an address and a port.

internal() syslog-ng internal messages

2.9.2.3 destination Section

This section defines the destination of the logs

Syntax:
destination <identifier>
{ destination-driver(params); destination-driver(params);...};

The possible destinations are the following ones:

file(<filename>) to send in a file

tcp(<ip>,<port>) and **udp(<ip>,<port>)**
to send the logs on the network for another machine

unix-stream(<filename>) to send to stream pipes (used in Linux)

userttyr(<user>) to send to the <user > consoles, but only if this user is connected. You can use the "*" character to specify that the messages have to be send to all users.

program(<commandtorun>) to send towards a program.

You can specify several destination directives in a destination section, as in the following example:

```
destination debug {file("/var/log/debug.log"); };
destination messages {file("/var/log/messages.log"); };
destination console {usertty("root"); };
destination xconsole {pipe("/dev/xconsole"); };
destination mail2admin {program("/usr/bin/MailToAdmin"); };
destination full{
file("/dev/tty12");
file("/var/log/full.log" log_fifo_size(2000));
};
```

Note: You can add specific options such as `log_fifo_size(2000)` in the example above.

In the following example, all logs will be sent to the management node, whose address is `192.168.0.100`:

```
destination central_log {tcp ("192.168.0.100" port(514); }
```

It can be interesting to use some macros to set coherent name to your destination files. Predefined macros exist, such as FACILITY, PRIORITY or LEVEL, DATE, FULLDATE, ISODATE, YEAR, MONTH, DAY, HOUR, MIN, SEC, FULLHOST, HOST. Following are some examples:

```
destination full {
file("/dev/tty12");
file("/var/log/full_$DAY-$MONTH-$YEAR.log"
owner("root")
group("adm")
perm(0640));
};
```

```
destination hosts {
file("/var/log/HOSTS/$HOSTS/$FACILITY/$YEAR/$MONTH/$DAY/$FACILITY$YEAR$MONTH
          $DAY"
owner("root")
group("adm")
perm(0600)
dir_perm(0700)
create_dirs(yes));
};
```

Note: do not forget to regularly remove or archive older files.

2.9.2.4 filter Section

This section describes the filtering mechanism for the events.

Syntax:
filter <identifier> {expression; };

The filters are defined by the following keywords:

facility(facility[,facility]) to filter by type

level(pri[,pri1, .. pri2 [,pri3]]) to filter on priorities or levels

program(regexp) to filter on the name of the program that has generated the message

host(regexp) to filter on a regular expression of the name of the host that has sent the message

match(regexp) to filter on a regular expression

filter(filtername) to use another filter.

All the keywords can be used several times. The expressions can contain AND, OR and NOT operators.

Examples:

```
filter f_iptables { match("IN=.*OUT=.*MAC=.*"); };

filter f_snort { match("snort: "); };

filter f_full { not filter(f_snort) AND NOT filter(f_iptables); };

filter f_messages { level(info..warn) AND NOT facility(auth, authpriv, mail,
          news); };
```

2.9.2.5 log Section

In this section you define how the messages will be processed, using the source, destination and filters defined in the previous sections.

Syntax:

log { source(s1); source(s2); ...
filter(f1); filter(f2); ...
destination(d1); destination(d2);
flags(flag1[, flag2...]; };

Examples:

```
log { source(src);
filter(f_news); filter(f_notice);
destination(newsnotice);
};

log { source(src);
destination(full);
};
```

2.10 Backing-up and Restoring System with mkCDrec

2.10.1 mkCDrec Overview

mkCDrec (make CD-ROM recovery) is an Open Source tool designed to make a bootable system image (including Linux system save), then to recover after a disaster happened, such as a disk crash or system intrusion.

The backups are generally on CD-ROM or DVD-ROM, or on an off-line disk (preferably in read-only mode), NFS mounted disk, or tape. Therefore, those backups are protected for non-authorized users.

The mkCDrec tool can be used for the following purposes:

- To restore software. Once booted from the **mkCDrec** CD-ROM or DVD-ROM, the **/etc/recovery/start-restore.sh** script can do the following:
  - restore the complete system after a disaster of some kind, (disk crash for example), or even after a successful system break-in
  - restore one disk (if there are more available disks in the system) from backup
  - restore backup of a disk onto a new (bigger) disk in the system.
- To make multiple copies, in order to increase reliability of the backups.
- As a rescue tool, for example to do **fsck** operations or to diagnose what's wrong with the system. See the mkCDrec Utilities to add more tools to your rescue CD-ROM or DVD-ROM.
- To “clone” a disk to another disk even when the target disk is smaller in size than the original disk, as long as there is room for the data. The **clone-dsk.sh** script will calculate the partition layout for you.
- It is possible to make multi-volume CD-ROMs so backups can be split up. It is also possible to backup all data required for boot on CD-ROM, in order to get a bootable CD-ROM, and to save other data on TAPE.
- To restore a single file system to an existing partition, using the **restore-fs.sh** command. The user can select the target file system type (to be formatted). The command has no arguments.
- To set-up (or migrate) to LVM, Software RAID, or another type of file system if your kernel permits it.
- To increase (or decrease) the partition size with the help of mkCDrec Utilities.

Currently, mkCDrec can backup and restore ext2, ext3, minix, ms-dos, xfs, jfs, fat, vfat and ReiserFS file systems. Logical Volume Manager (LVM), hard- and software RAID devices are also supported.

Note that mkCDrec is designed for system backup. It is not the objective of mkCDrec to backup all system data and it is recommended to regularly backup all your data with another method.

A typical example of usage is to run mkCDrec each night on a system and store the iso images on another system via NFS. In case of problem it will be possible to burn the saved image on a CD-ROM / DVD-ROM, then to restore the system.

Note: on Bull HPC, use mkCDrec on the management node only.

For more information, refer to http://mkcdrec.sourceforge.net

2.10.2 Configuring mkCDrec

The **/var/opt/mkcdred/Config.sh** file contains the configuration parameters for mkCDrec. All parameters have a default value, which is suitable for most of them. However, it is recommended to check the following parameters, either to verify that they fit your needs, or to define your own values in order to generate a coherent, but not too large, system backup.

BURNCDR	(Y or N) "**Y**" means that the CD-ROM/DVD-ROM will be burned directly from the machine. "**N**" means that iso images of the CD-ROM/DVD-ROM will be created.
ISOFS_DIR	Path of the temporary directory used before creating the iso images. Pay attention that this directory must be large enough to store the contents of a CD-ROM/DVD-ROM.
TMP_DIR	Path of the temporary directory used by mkCDrec.
DVD_DRIVE	(1 or 0) Set "**0**" to create CD-ROM backups or "**1**" to create DVD backups.
MAXCDSIZE	Maximum size of the created images (in ko). Example: 4200000 for DVD-ROM, 620000 for CD-ROM.
CDREC_ISO_DIR	Path of the directory used to store the iso backups. Pay attention that this directory must be large enough to store all the backups.
EXCLUDE_LIST	List of the directories and files to be saved in the backup. Take care to choose only what seems important to save, in order to obtain a backup of a reasonable size.
BOOTARCH	Defines the architecture of the system to backup (x86, ia64…). Check that the value fits your system.
RAMDISK_SIZE	It is advised to use 128

2.10.3 Creating a Backup

To create a backup, you must be logged as root user.

Go to the mkCDrec base directory (**/var/opt/mkcdrec** by default):

```
cd /var/opt/mkcdrec
```

Check that the system is operational for mkCDrec:

```
make test
```

mkCDrec displays messages if it has detected missing elements for backup. In this case, perform the appropriate corrections and restart **make test** until the test ends successfully.

Launch the backup operation:

```
make
```

When the operation is finished, iso images are created in the directory specified in the configuration file, ready to be burned.

2.10.4 Restoring a System

To restore a system, boot on the first CD-ROM/DVD-ROM, then run the command:

```
/etc/recovery/start-restore.sh
```

Follow the instructions displayed on the screen.

2.11 Capturing and Analyzing a Crash Dump with LKCD

2.11.1 LKCD Overview

The Linux Kernel Crash Dump (LKCD) product meets the needs of customers and system administrators who want a reliable method of detecting, saving and examining system crashes. LKCD enables to save the kernel memory image when the system dies due to a software failure (Oops, BUG, Panic, or DUMP button). This image can be stored on the disc (swap) or over the network (a dumpserver). This last solution is very useful in cluster configuration.

2.11.2 Configuring LKCD

LKCD configuration is described in the **/etc/sysconfig/dump** file. Although default values are sufficient for most systems, you can specify your own values for the following parameters:

DUMPDEV	Name of the dump device. It is typically a symbolic link to some block device in **/dev** (ex: DUMPDEV=/dev/sda3). It can be a network interface (ex: DUMPDEV=eth0).	
DUMPDIR	Location where the **lkcd save** command saves crash dumps in the case of a dump on disk. Default: DUMPDIR=/var/log/dump	
DUMP_LEVEL	Specifies the items to dump.	
	0 (DUMP_NONE)	Dump nothing, just return if called.
	1 (DUMP_HEADER)	Dump the dump header and first 128K bytes out.
	2 (DUMP_KERN)	Everything in DUMP_HEADER and kernel pages only.
	4 (DUMP_USED)	Everything except kernel free pages.
	8 (DUMP_ALL)	All memory.
DUMP_FLAGS	Flag parameters to use when configuring system dumps: 0x00000001 (DUMP_FLAGS_NONDISRUPT) Do not reboot after dumping. 0x80000000 (DUMP_FLAGS_DISKDUMP) Dump target is a local block device.	

	0X40000000 (DUMP_FLAGS_NETDUMP) Dump target is a network device.
DUMP_COMPRESS	Indicates which compression mechanism the kernel should attempt to use: 0 (DUMP_COMPRESS_NONE) Do not compress this dump. 1 (DUMP_COMPRESS_RLE) Use RLE compression. 2 (DUMP_COMPRESS_GZIP) Use GZIP compression. (not functional at 100%)

If the system is configured for the network, other parameters should be set :

TARGET_HOST=hostname	hostname/ip address of the target machine to which network dump needs to be sent.
TARGET_PORT=6688	port on which the netdump server in the TARGET_HOST machine is listening for network dumps. Default is 6688.
SOURCE_PORT=6666	port used on dumping machine to send dump data over the network. Default is 6666.
ETH_ADDRESS=ff:ff:ff:ff:ff:ff	Ethernet address used by the dumping machine to send out the data packets. This address must be specified in standard hex-digits-and-colons notation. If this variable is not set when lkcd is configured (**lkcd netconfig**), the MAC address will be set automatically. ATTENTION: if the network interface of the dumpserver is modified, you have to run **lkcd netconfig** on each system on the network. Note that a reboot launches **lkcd netconfig**.
TARGET_PORT_FILES=6665	port on which the dump-server is listening for files (**System.map** and **kerntpyes.o**).
NB_CRASHS_LOCAL=0	number of crashs the system has recorded. Automatically updated. This number is compared to the value on the server. Therefore, if a modification occurs on the server, this value must be updated manually.

If the default configuration is modified you have to set two more variables in the **/sbin/lkcd** file:

MAP=/boot/System.map KERNTYPES=/boot/Kerntypes	define the paths to **System.map** and **kerntypes.o** files, which are generated during the kernel compilation and useful for the dump analysis.

2.11.3 Operating LKCD

After any modification of a configuration file, you must run **lkcd config** or **lkcd netconfig**..

2.11.3.1 Recovering a dump

Once the system is configured for LKCD, recovering a dump can be made through different ways: LKCD is run on a BUG, Panic, or Oops message from the kernel. If you want to carry a dump out, use either **Sys Rq** or the dump button.

If you use **Sys Rq**, activate it with the following commands:

```
echo 1 > /proc/sys/kernel/sysrq
echo d > /proc/sysrq-trigger
```

If LKCD is configured for the network, the target machine needs to run a netdump-server.

2.11.3.2 Analyzing a dump with lcrash

lcrash is a system crash analysis tool that can greatly enhance an engineer's ability to analyze Linux system crash dumps. It contains many features in order to display information about the events leading up to a system crash in a clear, easy-to-read manner.

The **qlcrash** command runs a graphic interface of **lcrash**. The **qlcrash** syntax is identical to **lcrash**.

lcrash Syntax

lcrash <System.mapFile> <DumpFile> <KerntypesFile>

The analysis of the dump is made through interactive commands. These commands, invoked via an ASCII command-line user interface, provide access to a wide range of kernel internal data. Following are some of the provided commands:

stat	Displays pertinent system information and the contents of the **log_buf** array, which contains the latest messages printed via the kernel **printf** routines.
votp	Displays virtual to physical address mappings for both kernel and application virtual addresses.
ptype	Displays kernel data from the system dump and formats the data using a particular kernel data type.
symbol	Maps kernel symbols to virtual addresses.
dump	Dumps the contents of system memory in a variety of bases (hexadecimal, decimal, or octal) and data sizes (byte, short, int, or long).
task	Displays relevant information for selected tasks or all tasks running at the time of the crash.
trace	Displays a kernel stack backtrace for selected tasks, or for all tasks running on the system.
dis	Disassembles one or more machine instructions.

For more information, refer to http://lkcd.sourceforge.net

3. HPC Configuration

Most configuration tasks are performed at the time of installation. In addition, this chapter lets the administrator perform some basic configuration tasks. It also deals with security policy for HPC. For more information, refer to the *Bull HPC Installation and Configuration Guide*, which describes the different steps for installing and configuring HPC.

3.1 Configuring Services

- To run functionalities when Linux starts up, 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, 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.

3.2 Modifying Passwords and Creating Users

Two users are created when Linux is installed:
- **Root**: administrator (password **root**)
- **Linux**: ordinary user (password **linux**)

Don't forget to change the passwords as soon as possible.

- To change the passwords, use one of the following commands:
  - **passwd user_name** if you are connected as root user
  - **passwd** if you are connected as ordinary user.
- To create new users enter the **useradd –g** ***group*** **–d** ***home login*** command.

3.3 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 you can modify the fs-type using the Linux **mkfs**command (see 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 your changes

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

- For **ext3** filesystem 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, you have to define the mount points in **/etc/fstab** file and to mount these partitions using the **/bin/mount -a** command so that the partitions will be mounted when the system is restarted.

3.4 Creating Swap Partitions

The Linux **/sbin/mkswap** command does not let you create swap partitions greater than 2GB.

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

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

- Edit the **/etc/fstab** file and add the corresponding partition(s) to the swap.
- Run the **/sbin/swapon –a** command to take this swap into account.

3.5 Configuring Security

3.5.1 Basic Rules

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 for what concerns access to this node. Very few people should be able to access this node and this access should be made using **OpenSSH** to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks.

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

If resources (CPUs and memory) are shared between users, each user should not have access to other partitions.

3.5.2 Setting up SSH

RMS can be set up 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 behavior:

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

When prun is run with the -r option it will execute the following command for each of the selected nodes

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

where the optional ssh-args are taken from the attribute ssh-args.

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 that 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.

4. File Systems

4.1 File System Overview

Large scale parallel applications (executing simultaneously on multiple processors) process enormous volumes of data and generate huge numbers of I/O requests. Consequently, the I/O system should be able to store several hundreds of mega bytes per second and many disks should be used in parallel in order to maximize throughput. Therefore, it is important to share the file system serving workload across many disks, buses and cluster nodes.

Linux clusters can use distributed file systems based on standard client/server models such as NFS (Network File System) to share data among nodes and provide a consistent namespace across all machines. These systems typically work well for home directories due to performance improvements such as client-side caching. This approach reduces the overall network load but parallel applications can produce incorrect results if the data presented to them is inconsistent. Using file locks can enforce data consistency. This introduces a bottleneck during data access due to the locks overhead and can lead to important performance degradation.

Two types of file systems exist: **local file systems** and **distributed file systems**. The former are used locally on a machine to give access to the files on the disk. The latter are required for accessing remote files via the network. They also may provide locking systems to maintain consistency between the various users. For example, if several people are working on the same file, this type of file system will manage this "simultaneous" work.

Above these two types of file systems, there are also **parallel file systems** that share the files over several disks, thus "parallelizing" access to different parts of the file.

4.1.1 Local File Systems

Most of today’s file systems use journalizing techniques borrowed from the world of databases in order to improve incident recovery. Transactions are written sequentially to a part of the disk called the "log" before being written to the disk in their definitive place in the file system.

If the machine crashes, consistency is maintained and checking can be rapidly performed.

XFS - eXtended File System

XFS, widely recognized as a very high performance 64 bit file system, ensures the system restarts rapidly after a crash and can handle extremely large file systems.

For more information, refer to http://oss.sgi.com/projects/xfs/

Ext3

The ext3 file system is a standard Linux File System, equivalent to ext2 with journalizing.

4.1.2 Distributed File Systems

Scientific computation often requires access to very large data files. The time it takes to access such files has a major influence on the speed of the program's execution. The ideal scenario would be to have a program running on a single machine with a high performance hard drive. However, the very principle of clusters is to share computation over all the nodes in a network. Access to remote file systems is therefore unavoidable. It is here that distributed file sharing systems come into play.

NFS - Network File System

NFS is a system working in client/server mode that authorizes access to a directory for several clients distributed over a network. It therefore allows data to be shared among the cluster nodes. The limits for the directory and file sizes are those of the server, and for the access speeds are those of the network.

4.1.3 Parallel File Systems

Parallel file systems are designed specifically 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 cyclic 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. Because 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.

For HPC Bull recommends the **Lustre** parallel file systems.

4.2 Lustre Parallel File Systems

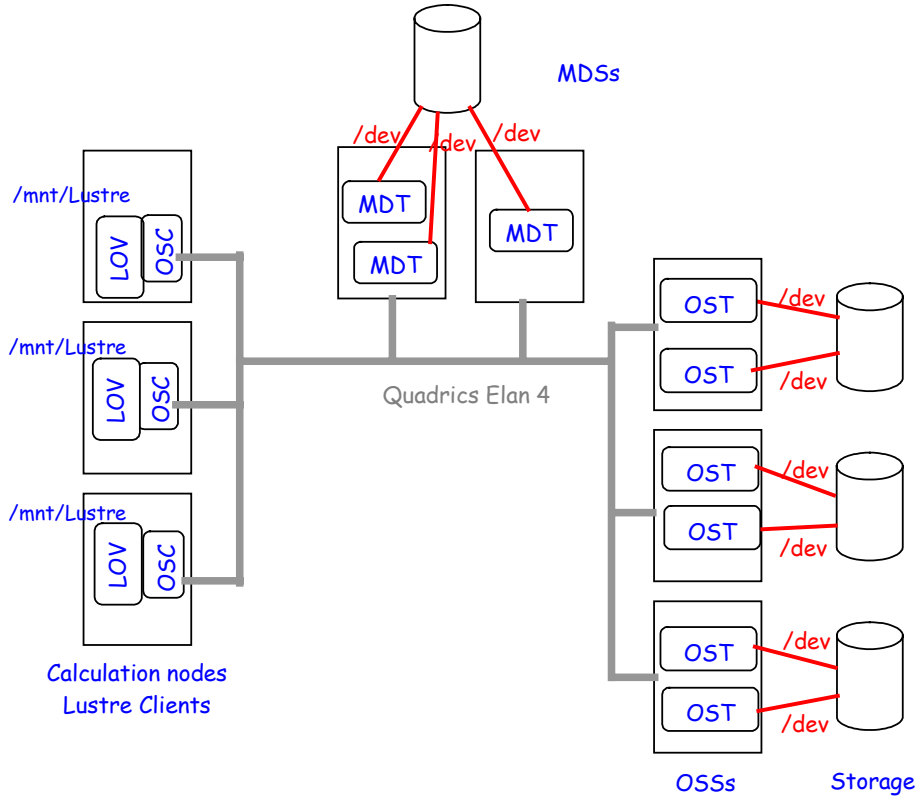
4.2.1 Lustre Overview

The **Lustre** global file system manages the data shared by several nodes, which are dispatched in a coherent way (cyclic distribution) on several disks 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 information related to the files:

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

When a node of the cluster needs an 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 systems.



MDS (MetaData Server)

MDS provides access to services called MDT (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 globally single directory tree that can be mounted by the Lustre file system clients.
- A MDT manages a backend ext3 file system which contains all of 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 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 of 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.

4.2.2 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, used storage resources, allocation of I/O nodes, accessibility of the various file systems by the Lustre clients, ...).
- If necessary, modify the configuration of the storage devices and the configuration of the Quadrics interconnect (network zoning, ...)
- Configure the Lustre service and activate the configured file systems.

During the file system lifetime, administrators may have to do operations such as stop, start, or repair. They may decide to update a configuration or to change the loaded one. They also need to monitor the file system to be aware of the current performance and of possible degradation of service.

4.2.3 Planning a Lustre File System

4.2.3.1 Data Pipelines

There are many data pipelines within the Lustre architecture, but there are two in particular which have 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.

4.2.3.2 OSS / OST Distribution

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

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

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

When sizing your OSS server nodes, we recommend that you divide your CPU into thirds: one third for the storage backend, one third for the network stack and one third for Lustre.

4.2.3.3 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 available space.

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

Inode size (Bytes)	**# of stripes stored inline**
128	0 (all EA would be stored in external block)
256	3
512	13
1024	35
2048	77
4096	163

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.

4.2.3.4 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 possible to limit network overhead and 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.

4.2.3.5 Lustre File System Limitations

- Lustre supports a maximum of 512 OSTs per Lustre file system (LOV).
- On the device it manages, an OST reserves up to 400MB for an internal journal and 5% for the root user. Like an OST, on the device it manages, a MDT reserve up to 400MB for an internal journal. This reduces the storage capacity available for the user's data.
- 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 performances and blocks allocation. It has the same limits in terms of maximum file size and maximum file system size.

4.2.4 Basic Lustre Management Tools

The Lustre package contains many tools; the most important are the following:

lmc	Lustre configuration maker which adds configuration data to the configuration file.
lconf	configure the node following directives in the configuration file. This command must be invoked from each node (I/O or client) involved in Lustre.
lfs	Lustre utility which checks the file system status and striping patterns.
lctl	interactive program which allows low level configuration.
mount.lustre	Lustre file system mount utility which allows a Lustre file system mount in a standard Linux way using the **client 0-config** way of configuration (replaces **lconf** for Lustre clients – can be used within **/etc/fstab**).

4.2.5 Bull Management Tools for Lustre

Bull has developed tools to create and manage Lustre file systems.

4.2.5.1 Lustre Configuration File: /etc/lustre/lustre.cfg

The Lustre administration tools use this file to get configuration information. On BAS3v2 this file only needs to be updated on the management node.

Syntax:

```
VARIABLE=VALUE
```

Lines beginning with # are comments.

LUSTRE_MODE=XML or **LDAP** or **HTTP**

XML	File system information is retrieved using the XML associated file which is copied in **/etc/lustre/conf** of OSS and MDS nodes when the file system is installed (lustre_util install). So this directory **does not need and must not be shared using NFS.**
HTTP	File system information is retrieved using the XML associated file which must be reachable from OSS and MDS nodes using http protocol. The URL of the HTTP server is set in the LUSTRE_HTTP_URL variable.
LDAP	File system information is retrieved from an ldap database. The URL of the LDAP server is set in the LUSTRE_LDAP_URL variable. File system information is loaded into the ldap database when the file system is installed (lustre_util install).
CLUSTERDB=no or **yes**	You must set this variable to **no**, because Lustre clusterdb tables are not enabled in BAS3v2 release.

LUSTRE_LDAP_URL=ldap://hostname

If LUSTRE_MODE=LDAP, the address of the ldap server is set in LUSTRE_LDAP_URL. For example if the ldap server is on a node called `ns2`, then `LUSTRE_LDAP_URL=ldap://ns2`. You can specify an absolute ldap url to lustre_util using this path: ldap://ns2/fs_name.

LUSTRE_HTTP_URL=http://path/to/xml

If LUSTRE_MODE=HTTP, the path to XML is set in LUSTRE_HTTP_URL. A good practice is to have the http server running on the management node and to make a symbolic link between **/etc/lustre/conf** and **/var/www/html/lustre**. So LUSTRE_HTTP_URL can be set this way: `LUSTRE_HTTP_URL=http://mgmt_node/lustre`. You can specify an absolute url to lustre_util using this path: http://mgmt_node/lustre/fs_name.xml.

4.2.5.2 Storage Configuration File: /etc/lustre/storage.conf

The **/etc/lustre/storage.conf** file describes the cluster storage configuration for Lustre tools. It stores information about the storage devices available on the cluster, describing which are OSTs and which are MDTs.

The **/etc/lustre/storage.conf** file must reside on the management node.

Syntax:

The **/etc/lustre/storage.conf** file is composed of lines with the following syntax:

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

Lines beginning with # (sharp) are comments.

ost|mdt	Specifies if this device is chosen to be an OST or MDT.
name	The name you want to give to the OST or MDT. For example, `/dev/sdd` on node `ns13` can be named `ns13_sdd`.
node_name	The hostname of the node where the device is.
dev	The device path (`/dev/sdd` for example).
size	Size of the device in kB, mandatory for loop devices, optional for other device.
jdev	The name of the device where the ext3 journal will be stored, if you want it 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. It is an optional parameter.

Filling /etc/lustre/storage.conf:

The **/etc/lustre/storage.conf** file is filled with information retrieved from the **/proc/partitions** of the I/O nodes.

For example, to display the content of **/proc/partitions** on a cluster where `ns13` is an I/O node, enter:

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

The following output is displayed:

```
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
```

We know that `sda` and `sdb` are system disks of `ns13`, so they must NOT be used as Lustre storage devices. Devices `sdd` to `sdh` are available devices. We will use 17430528 kB disks as journal devices and 75497472 kB disks as main devices. This choice gives the following lines in the **/etc/lustre/storage.conf** file of the management node:

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

The choice of which devices will be mdt or ost is left to the administrator.

Follow this procedure for each I/O node. The new lines must be appended in the **/etc/lustre/storage.conf** file of the management node.

4.2.5.3 lustre_config

Synopsis:

```
lustre_config create -s < path_to_filesystem_set_model >
lustre_config load -f < filesystem_name | path_to_xml >
lustre_config unload -f < filesystem_name | path_to_xml >
lustre_config list -f < filesystem_name | path_to_xml >
```

Description:

The **lustre_config** tool builds xml files describing Lustre file systems from a model file. It is also used to load and unload file systems information into the cluster database (if present), and to list the loaded file systems.

Actions:

create	Parses the model files (specified with **-s** option) and generates one xml descriptor for each file system in the **/etc/lustre/conf** directory.
load	Loads filesystems (specified with **-f** option) into the cluster database. Not usable in BAS3v2 since Lustre tables are not enabled in this release.
unload	Unloads file systems (specified with **-f** option) from the cluster database. Not usable in BAS3v2 since Lustre tables are not enabled in this release.
list	Prints the name of loaded file systems. Not usable in BAS3v2 since Lustre tables are not enabled in this release.

Options:

-s < path_to_filesystem_set_model >
Model files describing Lustre file systems (mostly in **/etc/lustre/models**). Wildcards are allowed.

-f < filesystem_name | path_to_xml >
File systems to work with. If you specify only a file system name, the xml descriptor will be searched in **/etc/lustre/conf**. Wildcards are allowed.

-v	Prints the version and exit.
-h	Prints help and exit.

4.2.5.4 Lustre Model File

A Lustre model file describes one or several Lustre file systems that can be used in the same time. It means they do not share OSTs or MDT. Such files are most often in **/etc/lustre/models**.

Syntax:

```
keyword:<value>
```

Possible keywords:

stripe_size	Specifies the stripe size in bytes. Default is 1048576 (1M).
stripe_count	Specifies the number of OSTs each file should be striped on. Default value is 0 and means to stripe on all available OSTs.
stripe_pattern	Only pattern 0 (RAID 0) is currently supported.
nettype	Possible values are **tcp** or **elan**. Default is **elan**.
generic_client	The name of the directory from which the file system is mounted on the mds. If network is **elan**, default is **clientelan**. If network is **tcp**, default is **clienttcp**.
fstype	File system type. Possible values are **ldiskfs** or **ext3**. Default (and recommended value) is **ldiskfs**.
failover	Enables failover support on OST and MDT. Possible values are yes or no. Default is no.
ha_timeout	Timeout in second for a client before going into high availability recovery. Default is 30s.
lustre_upcall	Location of the Lustre upcall script used by the client for high availability recovery. No default script is defined.

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

mdt_mkfs_options mkfs options for the MDT backend ext3 file system. By default, no option is specified.

mdt_inode_size Specifies the new inode size for underlying MDT ext3 file system. Defaults is self-evaluated.

mdt_mount_options mount options for the MDT backend ext3 file system. By default, no option is specified.

ost_mkfs_options mkfs options for OST backend ext3 file system. By default, no option is specified.

ost_inode_size Specifies the new inode size for underlying OST ext3 file system. Default is self-evaluated.

ost_mount_options mount options for the OST backend ext3 file system. By default, no option is specified.

cluster_id Specifies the cluster ID (one file system uses a single cluster id).

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

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

mount_path The mount-point to use to mount Lustre file system. The same mount-point must not be used for another file system 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>]

Specifies OSTs to use with this file system, using regular expressions matching their name, node_name, device, size, journal device, journal size. At least one field must be specified. If several fields are specified, only the OSTs matching every field 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 file system.

mdt: [name=<RegExp>] [node_name=<RegExp>] [dev=<RegExp>] [size=<RegExp>] [jdev=<RegExp>] [jsize=<RegExp>]

Specifies MDT of this file system. It is the same syntax than for OSTs. If several MDTs match, the first will be used. One and only one mdt line must be defined for each file system.

4.2.5.5 lustre_util

Synopsis:

```
lustre_util install -f < path_to_xml >

lustre_util start -f < fs_name | path_to_xml | ldap_url >

lustre_util mount -f < fs_name | path_to_xml | ldap_url > -n <nodes> |
          -p <rms_partition>

lustre_util umount -f < fs_name | path_to_xml | ldap_url > -n < nodes | '*'>
          | -p <rms_partition>

lustre_util status -f < fs_name | path_to_xml | ldap_url > -n < nodes | '*'>
          | -p <rms_partition>

lustre_util fs_status -f < fs_name | path_to_xml | ldap_url >

lustre_util mnt_status -f < fs_name | path_to_xml | ldap_url > -n < nodes  |
          '*'>| -p <rms_partition>

lustre_util stop -f < fs_name | path_to_xml | ldap_url >

lustre_util status -f < fs_name | path_to_xml | ldap_url >

lustre_util remove -f < fs_name | path_to_xml | ldap_url >

lustre_util info -f < fs_name | path_to_xml | ldap_url >

lustre_util set_cfg -n < clients list > | -p < rms partition >
```

Description:

lustre_util is a tool to install, enable, disable, mount and unmount, one or several Lustre filesystems from an administration node. You can use it when file systems xml descriptors are built (with lustre_config).

Actions:

install	Checks if the file system can be installed, and install it.
stop	Disables file systems.
start	Makes installed file systems available for mounting.

mount	Mounts file systems on specified nodes.
umount	Unmounts file systems on specified nodes.
fs_status	Prints OSS and MDS status information.
mnt_status	Prints information regarding client nodes.
status	Do fs_status AND mnt_status
remove	Removes file systems.
set_cfg	Creates **/etc/lustre/lustre.cfg** on all specified nodes.
info	Prints information (mdt, ost, path, ...) about file systems.

Options:

-f < fs_name | path_to_xml | ldap_url >
File systems to work with. This can be:
- A full path to xml file system descriptor.
- An ldap url : ldap://host/fs_name
- An http url : http://host/.../fs_name.xml
- A filesystem name. File system information will be retrieved regarding what is set for the variable LUSTRE_MODE in the **/etc/lustre/lustre.cfg** file.

-n <nodes_list>	Node list using pdsh syntax: `name[x-y,z],...,namek`.
-p <rms_partition>	Name of a running rms partition. The command will apply to the nodes of this partition.
-F	Forces commands execution even if dangerous (no user acknowledgement is asked for).
-t <time_in_second>	Sets the limit of time a command is allowed to execute. 0 means no timeout.
-u <user>	User name to use to log on the nodes instead of root.
--fanout	Number of ssh connections allowed to run in a same time, default is 256.
-V	Verbose output.
-v	Prints version and exit.
-h	Prints help and exit.

Installing a Lustre filesystem (install option)

```
lustre_util install -f /etc/lustre/conf/fs_name.xml -V
```

This command formats the storage devices and performs operations required to install the file system such as loading file systems information into ldap database and/or cluster database. If **-F** is used, no user acknowledge is required. If **-F** is not specified, the user must enter "yes" to go on if this file system is already installed. File system parameter MUST be a full path or an http url to its XML file.

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

Enabling a Lustre file system (start option)

```
lustre_util start -f fs_name -V
```

This command enables a file system and makes it available for mounting (online).

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

Mounting Lustre file system (mount option)

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

This command will mount the file system on the specified nodes using the mount-path defined in the XML. If this mount-path does not exist, it is automatically created. It is an error if this path is already used to mount another file system.

Unmounting Lustre file system (umount option)

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

This command will unmount the file system on the specified nodes. The mount-path is removed if the operation succeeds. You can use the **-n '*'** parameter if you want to unmount the file system everywhere.

Disabling a Lustre file system (stop option)

```
lustre_util stop -f fs_name
```

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

Removing a Lustre file system (remove option)

```
lustre_util remove -f fs_name
```

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

Printing file system information regarding OSS and MDS (fs_status option)

```
lustre_util fs_status -f fs_name
```

This command checks the status of OSTs and MDT, and tells if the filesystem can be mounted or not.

Printing file system information regarding clients (mnt_status option)

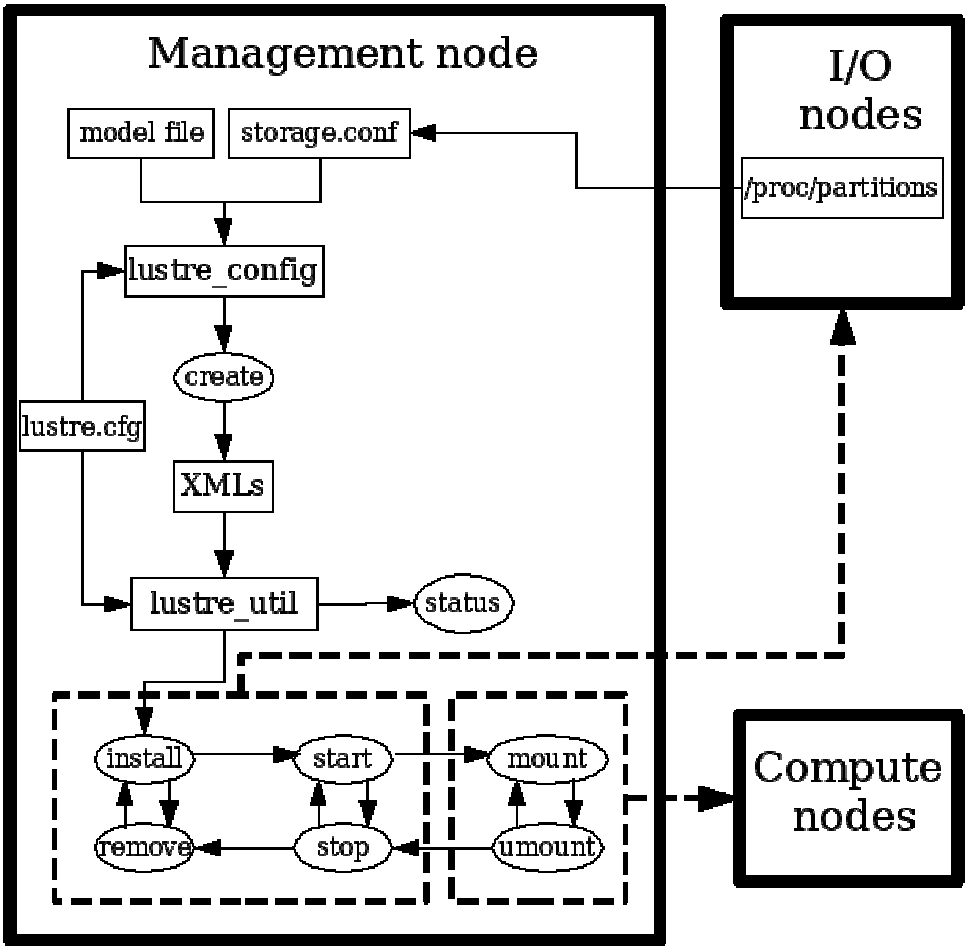
```
lustre_util mnt_status -f fs_name [ -n <nodes> ]
```

This command checks if the file system is correctly mounted or unmounted on the specified nodes. Without **-n** option, **mnt_status** gives the status of all clients that have to do with this filesystem.

4.2.6 Installing, Creating and Managing Lustre File systems

Note: An example of the complete process to create and install Lustre file system is available in *Bull HPC BAS3 Installation and configuration Guide*.

The following figure describes how the **lustre_config** and **lustre_util** tools operate, using the **lustre.cfg** and **storage.conf** files.



4.2.6.1 Creating Lustre File systems Using lustre_config

Prerequisites

- **/etc/lustre/lustre.cfg** is assumed to be correctly updated on the management node.
- **/etc/lustre/storage.conf** file must be correctly updated on the management node.

Lustre model sample file

Following is a model file describing two file systems (`fs1` and `fs2`), on a cluster whose nodes are called `ns<XX>`.

```
####################################################
# First, we define new default values for the 2 file systems

# We do not want 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 file system : fs1

# File system name is fs1
fs_name: fs1

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

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

# We want also 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 by 3
mdt: node_name=ns12 name=.*3
####################################################
# Second file system : fs2

# File system name is fs2
fs_name: fs2

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

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

# We want the mdt named mdt_ns13.ddn12.31
mdt: name=mdt_ns13.ddn12.31

# We want generic_client to be fs2_client instead of clientelan
generic_client: fs2_client
```

Creating XML descriptors of fs1 and fs2

Assuming previous data are saved in a file called **/etc/lustre/models/demo.lmf**, the `fs1.xml` and `fs2.xml` files are created by running the command:

```
lustre_config create -s /etc/lustre/models/demo.lmf
```

The `fs1.xml` and `fs2.xml` files will be generated and saved in **/etc/lustre/conf**.

Displaying content of XML descriptors

To check that OSTs and MDT are actually those you want to use with `fs1` or `fs2`, run the following commands:

```
lustre_util info -f /etc/lustre/conf/fs1.xml
lustre_util info -f /etc/lustre/conf/fs2.xml
```

4.2.6.2 Installing Lustre File Systems Using lustre_util

lustre_util uses ssh to execute remote commands, so the user must be allowed to log on the nodes without prompting for a password. This can be done by appending the right keys in the **/root/.ssh/authorized_keys2** file.

To install Lustre file systems, perform the following tasks:

1. Use **lustre_util install** command to install the file system.
2. Use **lustre_util start** command to enable the file systems.
3. Use **lustre_util mount** command to mount file systems on client nodes.

4.2.6.3 Removing Lustre File Systems Using lustre_util

To uninstall Lustre file systems, perform the following tasks:

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 systems.

5. Resource and Batch Management (TORQUE and RMS)

This chapter explains how you can benefit for the cluster resources to optimize the execution of the programs. It describes **TORQUE**, which allocates resources and manages queues on a HPC Cluster with one or several nodes.

5.1 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 configured priority criteria.

A job is a shell script that includes one or more parallel applications to be processed. A prologue and epilogue mechanism is attached.

The standard output files for batch processing (stderr, stdout) can be spooled.

Thanks to a 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 in time sharing, a mechanism interfaces the batch manager to define whether a job can be executed. The batch manager allocates the 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

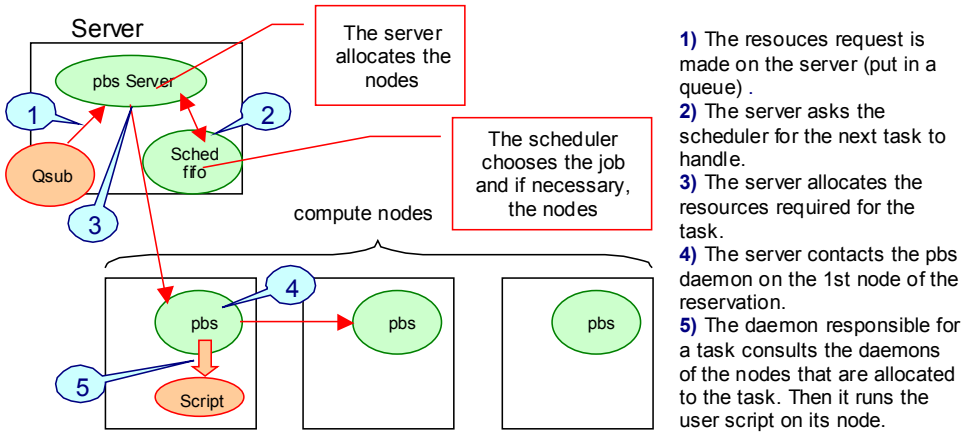
- Available space in the swap
- Job dates (submission, starting, completion)

The batch manager lets the administrator define thresholds for each resource. If a node resource exceeds one threshold, no new job will be assigned to this node. Furthermore, the jobs running on this node will be hanged, re-started or stopped.

For more information about TORQUE refer to the following Web site: http://www.clusterresources.com/products/torque/ .

5.2 TORQUE Architecture

The following figure shows the architecture of the TORQUE resource manager, in a cluster made of a management node (the server) and 3 compute nodes. It explains how the different components interact during the reservation of resources.



TORQUE includes the 3 following components:

- The **TORQUE server**, which is responsible for the management of the resources and for the queues. It provides the basic services of a resource manager, such as allocate/de-allocate resources, suspend/resume tasks.
- The **pbs_mom daemons**, which are responsible for the resources monitoring, and for managing the execution of user batch scripts. They run on all the compute nodes.
- The **scheduler**, which is the intelligent part of the resource manager. It selects which task 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… Some commands offer administration services, such as adding or removing nodes, or configuring the system.

5.3 Configuration Files

Following are the configuration files. All of them are in the **/etc/pbs** directory.

pbs.conf	configuration type.
pbs_mom.conf	configuration of the MOM (Node manager)
pbs_nodes	resources declaration.
server_name	name of the server.

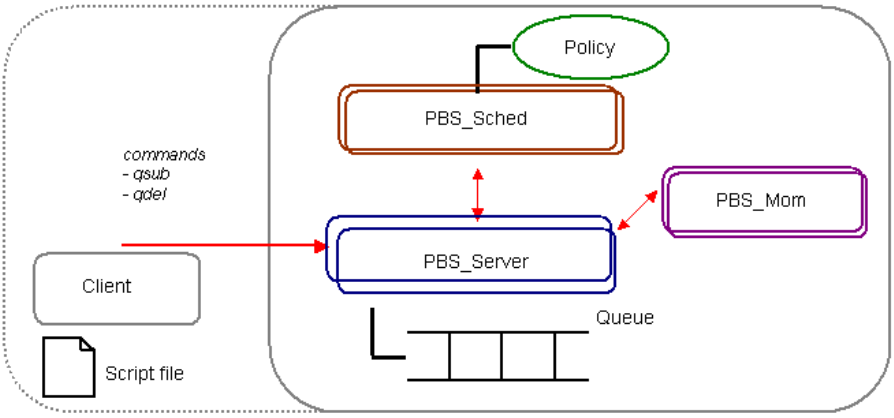
You must set the contents of these files according to your configuration, as described in the following sections.

5.4 Configuring TORQUE in Stand-alone Mode

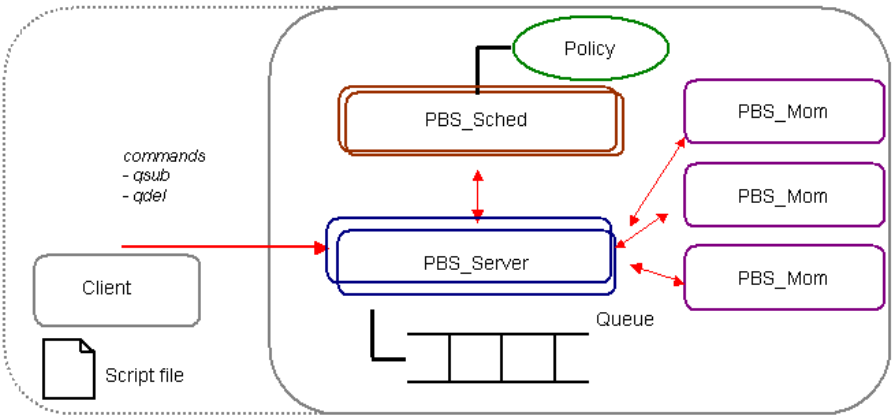
5.4.1 Architecture

In stand-alone mode all the components of the resource manager are present. Two configurations are possible:

- A **basic configuration**, for which a single node is declared to the resource manager. In this configuration, the users have only to specify the number of requested CPUs when they submit tasks (using the **qsub** command).



- A **pseudo-cluster configuration**, for which several nodes are declared to the resource manager. Each pseudo-node is a QBB (a set of CPUs created using cpuset). In this configuration, the users can specify both the number of requested nodes (QBBs) and the number of CPUs assigned to each node, when they submit tasks (using the **qsub** command).



5.4.2 Configuration Steps for a Basic Configuration

- For a basic configuration, set the TORQUE configuration files as follows:

pbs.conf `configuration = cluster`

pbs_mom.conf `$clienthost localhost`

pbs_nodes `localhost np = <number of cpu>`

server_name `localhost`

- Restart the services, with the following commands:

```
/etc/init.d/pbs_sched restart
/etc/init.d/pbs_server restart
/etc/init.d/pbs_mom restart
```

5.4.3 Configuration Steps for a Pseudo-cluster Configuration

- For a pseudo-cluster configuration the pseudo-nodes must be defined in the **/etc/hosts** file, with their IP address, as in the following example:

```
127.1.0.1    pbsserver
127.1.0.2    qbb0
127.1.0.3    qbb1
127.1.0.4    …
```

- Set the configuration files as follows:

pbs.conf `configuration = mono`

pbs_mom.conf `$clienthost pbsserver`

pbs_nodes
```
qbb0 np = <number of cpu>
qbb1 np = <number of cpu>
...
```

server_name `pbsserver`

Restart the services with the following commands:

```
/etc/init.d/pbs_sched restart
/etc/init.d/pbs_server restart
/etc/init.d/pbs_mom_qbb0  restart
/etc/init.d/pbs_mom_qbb1  restart
```

5.5 Configuring TORQUE and RMS in Cluster Mode

5.5.1 Architecture

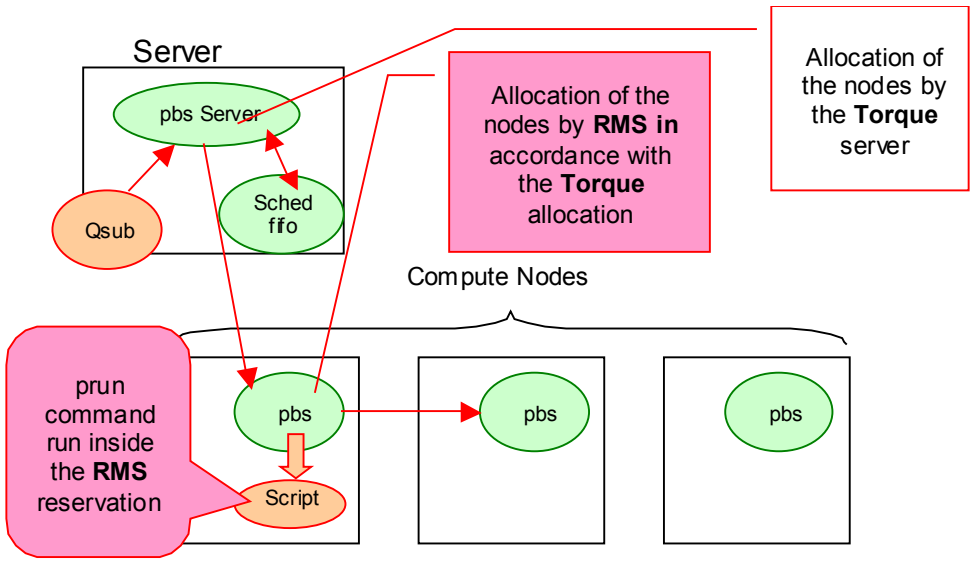
The **TORQUE** batch manager and the **RMS** resource manager have similar functions, but each one offers some distinguish features. For example TORQUE provides an efficient mechanism to manage several queues of jobs and a high-performance scheduler for these jobs and queues. On the other hand, RMS provides a very fine accounting mechanism and is able to configure the Quadrics network for the best performances.

The Bull HPC uses both TORQUE and RMS capabilities, in a transparent way for the user, to manage the more efficiently the job queues, priorities, scheduling, and resource allocation. When TORQUE and RMS work together (the **torque-rms-bull** package is installed), the configuration of RMS must be coherent with the configuration of TORQUE.

This section describes the configuration tasks that the administrator must perform after installation of the package, to make the TORQUE / RMS system operational.

Note: If you don't use RMS, just ignore the RMS configuration.

The following figure shows how the resources are allocated in a coherent way by TORQUE (on the server) and RMS (on the **pbs_mom** daemon that is responsible for the task).



5.5.2 Configuring TORQUE

In cluster mode, the management nodes (one or more in HA configuration) and the compute nodes are not configured in the same way.

5.5.2.1 On the Compute Nodes

- Set the configuration files as follows:

pbs.conf `configuration = cluster`

pbs_mom.conf `$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
```

5.5.2.2 On the Server

- Set the configuration files as follows:

server_name `<name of the machine where the TORQUE server runs>`

pbs_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 of the node, and `property` is a string defining a property of the node. Any string can be used; for example `bigdisk`, `bigmemory`, `smalljob` …

Note: The **pbs_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 jobs are running, 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, 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 may happen that the nodes appear in the `Unknown` state. This is normal, since it can take some minutes (2 or 3), for the nodes to inform the server of their respective state (using the **pbs_mom** service).

5.5.3 Configuring RMS

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.
- The compute nodes managed by TORQUE must belong to the same RMS partition.
- This partition must be either *batch* or *general*. Attention, in batch mode the compute nodes can be accessed only through TORQUE.
- All the compute nodes managed by TORQUE are also managed by RMS (except the nodes in *timeshared* mode in TORQUE).
- The root user must be able to run the following command from all the compute nodes and to all the compute nodes, without having to enter a password.

```
prun -t partition -H node_list uname
```

For detailed information about RMS configuration, refer to www.quadrics.com and see *RMS User Guide* and *RMS Reference Manual* in the Documentation page.

5.6 Bull Added Features

This section describes the new features added by Bull, enabling TORQUE and RMS to work together.

5.6.1 TORQUE and RMS Consistency

The nodes are allocated respectively by TORQUE and RMS. This feature is transparent for the users.

The association between a TORQUE job and an RMS resource uses two parameters:

- the **alt_id** parameter of a TORQUE job, which contains the identifier of the corresponding RMS resource,
- the **batchid** parameter of an RMS resource, which contains the identifier of the TORQUE job.

The state of the nodes (*configure in* / *configure out*) is automatically detected by TORQUE.

5.6.2 Suspending and Resuming a Job

The **suspend** and **Resume** TORQUE functions have been modified to release the resources, so that these resources can be allocated to another job. .

Note: the different processes of the job are stopped (SIGSTOP), but remain in the memory of the nodes used by these processes.

Attention: The released resources of a suspended job can be allocated to another job. So, if the suspended job is resumed, the resources must be available, else the operation is not allowed.

The commands to suspend and resume are the following ones:

```
qsig -s suspend JobId
qsig -s resume JobId
```

5.6.3 TORQUE and CPUSET

The user can assign cpusets to a TORQUE job. Using cpusets guarantees that CPUs are reserved to a job all along its execution.

The management of the cpusets corresponding to a TORQUE job requires the **torque-rms-bull-cpuset** package. It must be installed on all the compute nodes. This software uses the prologue / epilogue mechanism supplied by TORQUE to create (or delete) the cpusets corresponding to the TORQUE jobs.

Attention: The installation of the **torque-rms-bull-cpuset** package removes the existing prologue / epilogue scripts.

All information related to the cpusets of a job is stored in the **/var/PBS/cpuset/<JobId>.CPUSET** file, on each compute node affected by this job. This file has a lifespan equal to that of the cpuset.

Since TORQUE doesn't guarantee the execution of the epilogue script, a *garbage collector* program is run during the epilogue of each job to delete any zombie cpuset (whose job doesn't exist anymore in TORQUE).

TORQUE runs only either an interactive session or a script on the first node of the reserved CPUs, using **qsub –I** or **qsub script.sh**. Only this session or this script can be put inside the appropriate cpuset.

To launch an application on several nodes, it is mandatory to use **bullprun.sh** (and not **prun**), to put all the created processes in the cpusets (created by the prologue script) on all the nodes that belong to this job.

5.6.4 Affinities Between Queues and Nodes

The TORQUE server can be configured to use several queues. See the *OpenPBS Administrator's Guide* for detailed information about the queues configuration.

In addition to these standard features, you can set a property on a queue. In this way, only the nodes that have this property can be candidates for running the jobs present in the queue.

Note: See *Configuring TORQUE*, on page 5-7 to define the node properties.

To set a required property on a queue, use the **qmgr > set queue** command.

For example, to specify that `my_property` is a required property for a node to run a job waiting in `my_queue`, enter:

```
qmgr
 > set queue my_queue required_property=my_property
```

5.7 Usual Commands

This section lists the most usual commands. Since TORQUE and RMS work together in a transparent way, the user runs only TORQUE submission commands. The **prun** or **bullprun.sh** programs are naturally launched inside the TORQUE reservation.

5.7.1 TORQUE Commands for Job Management

- **qsub** to submit a job. You must be logged as a simple 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 lnodes=4 myscript.sh
```

    The job is queued, then started when the resource is available (4 CPUs).

  - To submit `myscript.sh` on 4 nodes and 2 processors on each node, enter this command on the management node:

```
qsub -l lnodes=4 :ppn=2 myscript.sh
```

    The job is queued, then started when the resource is available (8 CPUs).

- **qdel** to delete a waiting or running job.
- **qhold** / **qrls** to hold / release a job waiting in its queue.
- **qsig –s suspend** / **resume** to suspend / resume a running job.

5.7.2 Commands to Start a Program Inside a Job

- **prun** to start a parallel program. To be used if cpuset is not installed.
- **bullprun.sh** to start a parallel program. To be used if cpuset is installed. Same syntax than **prun**.

5.7.3 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. To be used carefully.

5.7.4 Administrator's Commands and Configuration File

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 the batch system.

The following file defines the compute nodes and their properties:

- **/etc/pbs/pbs_nodes** file (TORQUE)

5.8 Checking the TORQUE / RMS Configuration

This section gives an example of procedure enabling to simply check that the TORQUE / RMS system is correctly configured.

Prerequisite: the user accounts must exist on all the compute nodes.

1. As a simple user submit a job:

```
qsub -I -lnodes=2
```

   If the user is in its reservation, an interactive shell opens.

2. Check that the environment variables (PBS*) are correctly set.
3. Enter:

```
prun uname -a
```

   This command should execute **uname** on the two allocated nodes.

4. List the active cpusets:

```
pls
```

   The shell must be in the cpuset specified in the **/usr/spool/PBS/<JobId>.CPUSET** file.

5. Verify that the **alt_id** field of the TORQUE job corresponds to the identifier of the RMS resource.

   To display the **alt_id**, enter:

```
qstat –f JobId
```

To display the identifier of the RMS resource, enter:

```
echo $RMS_RESOURCEID
```

6. Check the coherency of the TORQUE and RMS reservations:

```
rinfo
```

7. Close the interactive session:

```
exit
```

8. Check that all resources are released:

```
rinfo
pbsnodes –a
```

Run **pls** on all the nodes allocated to the job:

```
pls
```

Check that the **/usr/spool/PBS/<JobId>.CPUSET** file doesn't exist anymore.

Glossary

ACL
Access Control List.

ConMan
A management tool, based on telnet, enabling access to all the consoles of the cluster.

EFI
Extensible Firmware Interface.

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.

HPC
High Performance Computing.

KDE
Kool Desktop Environment.

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.

MDS
MetaData Server.

MDT
MetaData Target.

MkCDrec
Make CD-ROM Recovery. A tool making bootable system images.

MPI
Message Passing interface.

Nagios
A powerful monitoring tool, used to monitor the services and resources of Bull HPC clusters.

NFS
Network File System.

NTP
Network Time Protocol.

OpenSSH
Open Source implementation of the SSH protocol.

OSC
Object Storage Client.

OSS
Object Storage Server.

OST
Object Storage Targets.

PAP
Platform Administration Processor (Bull NovaScale platforms).

PDSH
A parallel distributed shell.

RMS
Resource Management System. Manages the cluster resources.

SIS
System Installation Suite.

SSH
Secure Shell. A protocol for creating a secure connection between two systems.

Syslog-ng
Syslog New Generation, a powerful system log manager.

TORQUE
Tera-scale Open-source Resource and QUEue manager. A batch manager controlling and distributing the batch jobs on compute nodes.

UID
User ID.

XFS
eXtended File System.

Index

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

X

Vos remarques sur ce document / Technical publications remarks form

Titre / Title :	**Bull HPC BAS3 Administrator's Guide**

N° Référence / Reference No. :	86 A2 31EM Rev02

Date / Dated :	November 2005

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront attentivement examinées. Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.
Your comments will be promptly investigated by qualified personnel and action will be taken as required. If you require a written reply, furnish your complete mailing address below.

NOM / NAME : ..... DATE : .....

SOCIETE / COMPANY : .....

ADRESSE / ADDRESS : .....

.....

Remettez cet imprimé à un responsable Bull S.A. ou envoyez-le directement à :
Please give this technical publications remarks form to your Bull S.A. representative or mail to:

Bull S.A.
CEDOC
Atelier de reprographie
357, Avenue Patton BP 20845
49008 ANGERS Cedex 01
FRANCE

BULL CEDOC
357 AVENUE PATTON
BP 20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 31EM Rev02