# bullx cluster suite

## High Availability Guide

Extreme Computing

REFERENCE
86 A2 25FA 03

# Extreme Computing

# bullx cluster suite

## High Availability Guide

Software

April 2010

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

REFERENCE
86 A2 25FA 03

Printed in France

## Trademarks and Acknowledgements

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

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

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

# Table of Contents

# List of Figures



# List of Tables

# Preface

## Scope and Objectives

High Availability is used to ensure the continuous running of the cluster when there is a system component failure or a downtime event. This guide describes the configuration and management of the different forms of High Availability that apply to Bull Extreme Computing clusters.

**Refer to the Software Release Bulletin (SRB), delivered with your system, for details of the High Availability functions and features that apply to your delivery. Please pay particular attention to any restrictions which may be in place for your delivery.**

## Intended Readers

This guide is for administrators of bullx cluster suite system.

## Bibliography

Refer to the manuals included on the documentation CD delivered with your system OR download the latest manuals for your **bullx cluster suite** release, and for your cluster hardware, from: http://support.bull.com/

The *bullx cluster suite Documentation* CD-ROM (86 A2 12FB) includes the following manuals:

- *bullx cluster suite Installation and Configuration Guide* (86 A2 19FA)
- *bullx cluster suite Administrator's Guide* (86 A2 20FA)
- *bullx cluster suite Application Developer's Guide* (86 A2 22FA)
- *bullx cluster suite Maintenance Guide* (86 A2 24FA)
- *bullx cluster suite High Availability Guide* (86 A2 25FA)
- *InfiniBand Guide* (86 A2 42FD)
- *LDAP Authentication Guide* (86 A2 41FD)
- *SLURM Guide* (86 A2 45FD)
- *Lustre Guide* (86 A2 46FD)

The following document is delivered separately:

- The *Software Release Bulletin* (SRB) (86 A2 80EJ)

**The Software Release Bulletin contains the latest information for your delivery. This should be read first. Contact your support representative for more information.**

For **Bull System Manager**, refer to the *Bull System Manager* documentation suite.

For clusters that use the **PBS Professional** Batch Manager, the following manuals are available on the *PBS Professional CD-ROM*:

- *Bull PBS Professional Guide* (86 A2 16FE)
- *PBS Professional Administrator's Guide*
- *PBS Professional User's Guide* (on the *PBS Professional CD-ROM*)

For clusters that use **LSF**, the following manuals are available on the LSF CD-ROM:

- *Bull LSF Installation and Configuration Guide* (86 A2 39FB)
- *Installing Platform LSF on UNIX and Linux*

For clusters which include the **Bull Cool Cabinet**:

- *Site Preparation Guide* (86 A1 40FA)
- *R@ck'nRoll & R@ck-to-Build Installation and Service Guide* (86 A1 17FA)
- *Cool Cabinet Installation Guide* (86 A1 20EV)
- *Cool Cabinet Console User's Guide* (86 A1 41FA)
- *Cool Cabinet Service Guide* (86 A7 42FA)

### Highlighting

- Commands entered by the user are in a frame in 'Courier' font, as shown below:

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

- System messages displayed on the screen are in 'Courier New' font between 2 dotted lines, as shown below.

```
Enter the number for the path :
```

- Values to be entered in by the user are in 'Courier New', for example:

  COM1

- Commands, files, directories and other items whose names are predefined by the system are in '**Bold**', as shown below:

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

- The use of *Italics* identifies publications, chapters, sections, figures, and tables that are referenced.
- < > identifies parameters to be supplied by the user, for example:

  <node_name>

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

# Chapter 1. Cluster High Availability - an Introduction

**Important** Different types of High Availability are supported for different bullx cluster suite releases.
See the System Release Bulletin delivered with your cluster for details of the High Availability domains and features that apply to your system.

## 1.1 Definition

By High Availability, we mean a protocol, and its associated execution, which ensures operational continuity when there is a downtime event for a particular part of the cluster. High Availability ensures that a cluster is protected against any potential single points of failure (SPOF) that may exist. A continuously available cluster system is characterized as one with minimum downtime in any given year. The best High Availability rate is known as 99.999% availability.

Highly available clusters operate by having redundant nodes to take over a service, if and when a system component fails. Hardware and software faults are detected instantly, and the application is immediately switched to the redundant node with no lost of processing time, and with no direct intervention from the Administrator. This process is known as failover.

The implementation and configuration of High Availability focuses on system backup, failover processing, data storage, and access routines for the various components of the cluster. This manual describes the various mechanisms that are available for Bull Extreme Computing clusters. A prime requirement for these mechanisms is to ensure that there is no data corruption when failover occurs.

## 1.2 High Availability Domains

The architectural requirements, the software configuration operations, and High Availability management for the different domains of Bull Extreme Computing clusters to which High Availability may apply of are described in this manual.
The following domains are described:

### 1.2.1 High Availability for the Management Node

High Availability for the Management Node applies to the Management Services, and can also extend to the NFS services on the Management Node.

- See *Chapter 2* for a description of possible Management Node failures, High Availability architectural requirements, and the Management Node HA scripts that are provided.
- See *Chapter 3* for details on how to configure High Availability on the Management Node.
- See *Chapter 4* for details on managing High Availability on the Management Node.

### 1.2.2 Load Balancing for Login Nodes

See *Chapter 5* for details on how to implement Load Balancing on the Login Nodes using the **Heartbeat** and the **Linux Virtual Server** products.

### 1.2.3 High Availability for the File System

#### Lustre

**Lustre** uses an **active/active** node configuration. This means that all the I/O nodes are active and if a node fails the other nodes will pick up its load.

- See the *Lustre Guide* for a description of Lustre failover mechanisms, the hardware architecture required, an analysis of the different possible failures that are covered by High Availability for Lustre, and details on how to configure High Availability for Lustre.

#### NFSv3

**NFSv3** uses an **active/passive** node configuration for the NFS nodes.

- See *Chapter 6* for description of the hardware architecture required, the different possible failures that are covered by High Availability for NFSv3, and for details on how to configure and monitor High Availability for NFSv3.

### 1.2.4 High Availability for Resources and Jobs

- See the *PBS Professional Guide*, *LSF Guide*, and *SLURM Guide* for the description of High Availability for the different resource and job managers that exist for Bull Extreme Computing clusters.

# Chapter 2. Understanding Management Node High Availability

**Important** **Management Node High Availability is not supported for all system releases. See the Software Release Bulletin delivered with your system for details of any limitations which may apply to your release.**

This chapter describes the architecture required for the implementation of Management Node High Availability using **HA Cluster Suite**, and also describes some of the High Availability scripts included with the **bullx cluster suite** delivery. See the next chapter for details on how to configure High Availability for the Management Node.

## 2.1 Introduction

One of the most fragile points of the cluster is the Management Node. All management services for a cluster running on the Management Node will become unavailable if the Management Node goes down. The management of a compute cluster job runs directly from the Management Node. Therefore, if the Management Node goes down it will no longer be possible to submit any compute jobs, making all the nodes unusable.

In order to avoid such a lost of functionality, it is necessary to physically double up the Management Nodes, and to put a recovery mechanism into place for the cluster management services on a backup node. This node is also called the Secondary Node.

When a problem occurs on one of the Management Nodes, the management services running on that node rock over to the second one. High Availability for the cluster management is based on an architecture which includes two Management Nodes and shared disk space. The same architecture is also required to ensure High Availability for the Batch and Resource Manager in place.

**Note** The Resource/Batch Manager includes its own High Availability (failover) mechanism that may operate independently of the Management Node High Availability solution.

**Important** **If it is decided to change High Availability Management Nodes to standalone mode without High Availability, then the nodes must be completely re-installed from scratch.**

### 2.1.1 High Availability for the Management Node and the NFS File System

High Availability for the **NFS** services together with High Availability for the Management Node can also be implemented if you choose an all-in-one solution.

---

**Note** When the High Availability of the Management Node services and **NFS** services is implemented together on two Management Nodes, the Management Node services are on the first Management Node and the **NFS** services are on the second Management Node. In this situation the High Availability architecture is **active/active**.

---

High Availability for the **NFS** services can also be implemented on separate nodes in a large cluster installation in order to ensure better network performance, in this case see the Chapter 6 - *NFSv3 High Availability for I/O Nodes*.

### 2.1.2 Objectives for Management Node High Availability

The objectives for Management Node High Availability are:

- To keep critical services running
- To detect software and hardware failure and stop the faulty node by a software/hardware mechanism (fencing)
- To switch services to the Secondary Node smoothly if the Primary Node fails
- To ease the upgrade process for system services
- To alert the System Administrator so that the fault is fixed

The solution has to be easily upgradeable with a minimum of service downtime. This is only possible by using the same hardware/software configuration on both the Primary and Secondary Nodes. The upgrade of a system starts with the isolation and upgrade of the Secondary Node. The critical services are then migrated from the Primary to the Secondary node. This check validates the upgrade process. After this step the Primary Node can be upgraded without risk.

### 2.1.3 Basic Design Approach

For Management Nodes which are made Highly Available, the node which is actually running the critical services is designated as the **Primary Node.** The node which is ready to take the place of the Primary Node, should the Primary Node go down, is designated as the **Secondary Node.**

Both nodes send heartbeat signals through the network. If a node stops sending its heartbeat signal, it may mean there is a software or hardware breakdown. In this situation the High Availability software takes the actions necessary to maintain the availability of the critical services and the integrity of the data, whilst rocking over (failover) to the Secondary Node in a transparent way.

When the Secondary Node takes charge, the High Availability software cuts data, network and power access to the Primary Node by a fencing mechanism to ensure that there is no risk of it restarting suddenly. The Secondary Node takes charge of the critical services only if the Primary Node is fenced. This process ensures data integrity and is the only reliable solution against the **split brain** problem.

The High Availability management tool is **HA Cluster Suite.** This uses an advanced heartbeat mechanism to detect the state of the nodes and services. A fencing mechanism is mandatory to enable the High Availability feature.

Within a Bull Extreme Computing cluster the different constituent elements, for example, Compute Nodes, I/O nodes, network equipment, and services address a **virtual Management Node** using a virtual IP address. This virtual Management Node is in reality the active node. This virtualization of the Management Node is implemented by the mechanisms described within this section.

### 2.1.4 Fencing details

Fencing stops I/O operations for the shared storage, thus ensuring data integrity. The cluster infrastructure performs fencing using **CMAN/DLM** and the fence command provided by the **cman** package.

When HA Cluster Suite determines that a node has failed, it communicates this failure to other cluster-infrastructure components. The fencing program, when notified of the failure, fences the failed node. Other HA Cluster Suite components determine what actions to take — that is, they perform any recovery operations that need to be done. For example, **DLM** and **GFS**, when notified of a node failure, suspend all activity until they detect that the fencing program has completed fencing the failed node.

HA Cluster Suite can be configured to fence in two different ways. Firstly, when a node failure is detected the node is rebooted by the fencing method (power-off followed by power-on). Secondly, when a node failure is detected a dump is forced by the fencing method (with the **kdump** configuration, the dump is usually followed by a reboot).

The choice between these two options must be made when **HA Cluster Suite** is configured with the **-f <reboot | dump>** option for the **stordepha** command (**dump** is the default option).

**Important** **The dump option requires that the kdump option is configured on the Management Node. See Chapter 3, STEP 3 in the Installation and Configuration Guide for more details.**

Most nodes that fail will be rebooted (either automatically following a dump if the **dump** option is selected, or if the **reboot** option is selected for the fencing method for the other node that forces the node to power-off and then power-on). There is only one situation where the node remains powered off: if the **dump** option has been chosen and an I/O error is detected on a storage device. In this situation the node will then power itself off, and consequently when the order to dump sent by the other node is received, it will be ignored as the node has already been powered off. The node will remain powered off until it is powered on manually by the administrator.

**Note** When a node is powered on, **HA Cluster Suite** is not launched automatically when the system boots, it must be restarted manually by the Administrator in order that the node becomes the failover node for the other node in the HA pair.

## 2.1.5 Possible Management Node failures

### Management Node kernel panic/hang

When a node hangs or encounters a panic, it does not send its heartbeat messages within the authorized period. This silence is detected by the High Availability peer node which fences the silent node, and takes over the cluster services when the fencing is completed.

### Node Hardware Failure or Power Failure

The Primary Node does not send its heartbeat messages in the authorized period. This silence is detected by the High Availability peer node, which fences the silent node and takes over the cluster services when the fencing is completed.

**Note** The **BMCs** often provide an IPMI interface used for the fencing mechanism. **BMC** services are activated by default when the power cable is plugged in. Two power supply ports with two isolated power networks are needed to support High Availability for the hardware.

### Network Failure

The Primary Node does not send its heartbeat messages in the authorized period. The network is not available from the Primary Node. The Secondary Node tries to fence the Primary Node and if it succeeds, restarts the service. If it cannot fence the Primary Node, it will retry in an infinite loop.

### Service Failure

The Primary Node send its heartbeat messages in the authorized period but the service check associated with a Highly Available service fails. The service is restated automatically on the Primary Node, if an error occurs the service is migrated to the Secondary Node.

**Note** By default the authorized period for the heartbeat message is 21 seconds.

## 2.2 Architecture

The basic architecture for a medium sized cluster is shown in the diagram below.

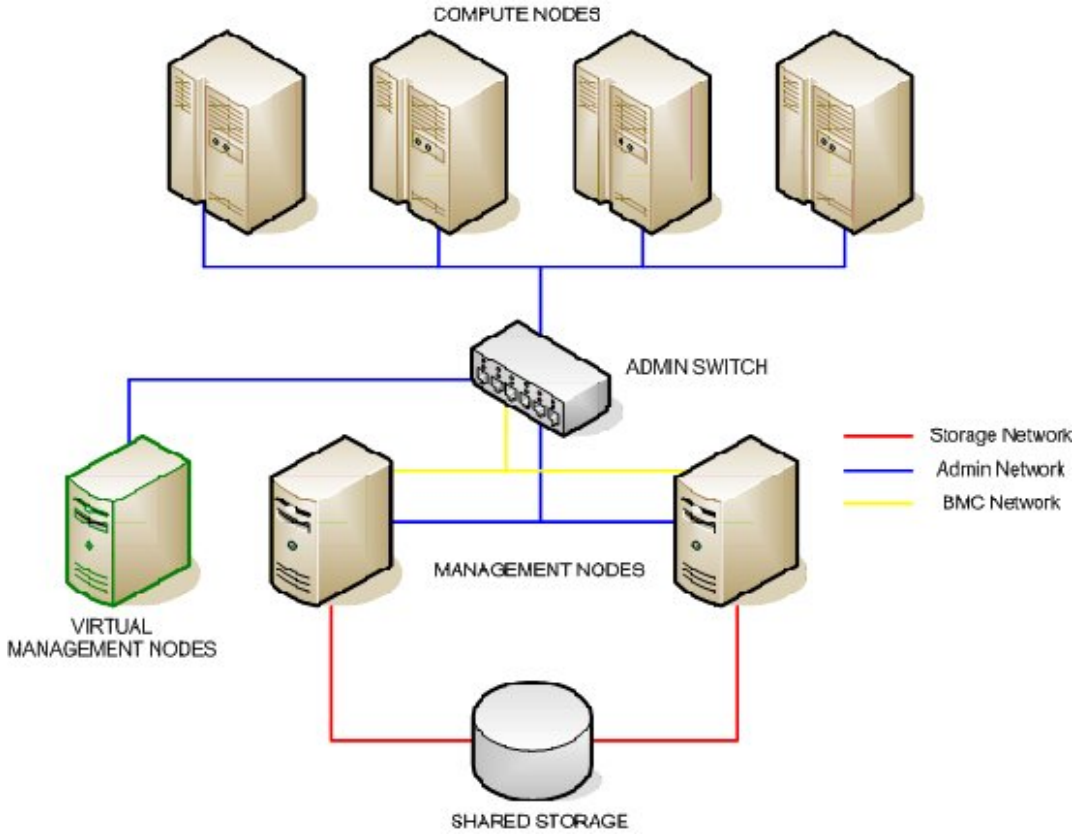


Figure 2-1. Medium Sized Cluster architecture

### 2.2.1 Hardware

Different architectures are used according to size of the cluster and the structure of the Service Nodes.

**Note** Chapter 1 in the *Installation and Configuration Guide* for details on the different architectures possible.

#### Small clusters

Either one single Service Node or two Highly Available Nodes are used for all the cluster services: Management, Login, and I/O (NFS only).

#### Medium-sized clusters

One Service Node runs the cluster management services and a separate Service Node runs the Login and the I/O (NFS only) services. High Availability for the management services and NFS services is supported for this configuration.

#### Large clusters

Services run on separate dedicated nodes (cluster management, login, I/O). Clusters that use the **Lustre** parallel file system need at least two separate services nodes dedicated to it. High Availability for the Management service and the NFS service is supported for this configuration.

### Management Node High Availability Architecture Requirements

For Management Node High Availability the architecture needs to include a minimum of:

- Two management service nodes connected to a shared data disk array
- Machines that include BMCs to provide a fencing service per node
- BMC network connected to the Administration network

The network switch SPOF on the administration network can be eliminated by using Ethernet bonding. This will be included according to specific customer requests. Each Management Node can handle one or more cluster services. Each service under HA Cluster Suite will run on only one Management Nodes at a time. The Resource and Batch managers will run on one of these nodes.

Medium and large clusters run the management and NFS services on two separate pairs of Service Nodes.

## 2.2.2 Software

High Availability is implemented using HA Cluster Suite. Therefore, HA Cluster Suite has to be configured, and the system adapted. The Red Hat Linux distribution has to be installed on all the Management Nodes, as well as all the applications needed for smooth operation of the cluster.

In addition, /etc/init.d scripts have been developed to implement High Availability of the management service on top of HA Cluster Suite. These scripts implement the start, stop, and status parameters required for the HA Cluster Suite software.

For each management service that is to be Highly Available it will be necessary to take an inventory of:

- The files and directories that have to be present on both the Primary and Secondary nodes (for example /etc/passwd, /etc/hosts, /etc/sshd, etc.) and that may occasionally be modified for configuration purposes.
- Which files and directories are dynamic and subject to change regularly, and so are difficult to synchronize. This data should be placed on the shared storage common to both Management Nodes (/var/lib/pgsql/data, etc.)
- Which files and directories are static (/usr/sbin/httpd, etc.) and are only modified for special administration tasks.

## 2.1.6 Network

The applications running on the Compute Nodes have to communicate with the management applications. Most of the time client applications communicate with a host which is identified by its IP address. However, when there is a failover process (the existing primary node stops being the active one and the secondary one becomes active) the new active node must be accessible. Therefore, the IP address has to follow the active node. The node which runs the active services uses a virtual IP address to maintain connectivity for the cluster nodes. Nevertheless, both Management Nodes must have a proper IP address so that they can be reachable separately.

Each Management Node has an IP address to communicate with the management network (for example, 10.0.0.1/24 for the Primary Node, and 10.0.0.2/24 for the Secondary Node). In effect, when the High Availability service is active on the Primary Node, the interface with the Management Node is the virtual IP address (for example, 10.0.0.250/24), virtual because it is not directly linked to a physical node but to the node which is active at any particular time. This is a functional resource not a physical one.

It is possible to define several virtual IP addresses for the services listed in the Cluster database. The ClusterDB database includes a **cluster_ipv** table to define the virtual IP addresses associated with each service.

HA Cluster Suite manages the virtual IP addresses, monitors them and configures network interfaces for them.

## 2.2.3 Network Configuration Example

### When the Active Node is the Primary Node

- The management interface with the primary node has two IP addresses (10.0.0.1/24 and 10.0.0.250/24).
- The management interface with the secondary node has one IP address (10.0.0.2/24).
- All cluster components can connect with the Primary Node using its real IP address (10.0.0.1/24) and its virtual IP address (10.0.0.250/24).
- The Secondary Node can only be accessed by using its real IP address (10.0.0.2/24).
- The active node is accessed using the virtual IP address (10.0.0.250/24).

### When the Active Node is the Secondary Node

- The management interface with the Primary Node has one IP address (10.0.0.1/24).
- The management interface with the Secondary Node has two IP addresses (10.0.0.2/24 and 10.0.0.250/24).
- The Primary Node can only be accessed by using its real IP address (10.0.0.1/24).
- All cluster components can connect with the Secondary Node using its real IP address (10.0.0.2/24) and its virtual IP address (10.0.0.250/24).
- The active node is accessed using the virtual IP address (10.0.0.250/24).

## 2.2.4 Shared data

To allow application failover, and to avoid data corruption when a split brain situation arises, the High Availability solution uses **GFS** (Global File System) for the shared data. The **GFS** file system allows concurrent read-write, and locking mechanism for the file system resources.

**HA Cluster Suite** manages the file system, monitors it and configures the **GFS** mount points.

## 2.3 Highly Available Management Services

A list of management services has been identified for the Bull Management Node High Availability solution. These are managed by **HA Cluster Suite** in two separate service lists, one for the management services and one for the NFS services:

**HA_MGMT**

This service list is composed of the management services, these are tightly coupled and interdependent. The **/usr/sbin/haservices** script manages High Availability for these services - see section *2.4.1* for more details.

**HA_NFS**

This manages High Availability for the **/etc/init.d/nfs** services.

For most other services, customers can create a custom **init** script using the standard template provided by the **Red Hat** Linux distribution.

### 2.1.7 HA_MGMT service list

**HA_MGMT** is the name of the group of management services controlled by **HA Cluster Suite** using the **haservices** script, and its custom **start**, **stop**, and **status** configuration files. The script and configuration files are in the **/etc/clustmngt/ha** directory, where they can be checked.

For each service included in the **HA_MGMT** group, Bull has identified those that use static data which must be synchronized for both the Primary and Secondary nodes, and those that use dynamic data and are shared for both the Primary and Secondary nodes. See the table below for information regarding each management service. Service configuration files use static data and must be synchronized.

| Service | Data which is shared | Shared mount point | Synchronized Configuration files |
|---|---|---|---|
| postgresql | DB Clusterdb | /var/lib/pgsql/data | |
| syslog-ng | client syslog | /var/log/HOSTS | /etc/syslog-ng/syslog-ng.conf |
| Nagios | X | X | /etc/nagios /usr/share/nagios |
| gmond | X | X | /etc/gmond |
| gmetad | Round Robin DB | /var/lib/ganglia/rrds | /etc/gmetad.conf |
| dhcpd | X | X | /etc/dhcpd.conf /etc/dhcpd-tpl.conf |
| lustre | Configuration file | /etc/lustre | |
| mgs | Lustre MGS | /home/lustre/run | |
| ldap | Lustre DB | /var/lib/ldap | |
| httpd | Server web | /var/www | |
| conman | Conman logs | /var/log/conman | /etc/conman.conf |
| systemimager | Ksis images | /var/lib/systemimager | /tftpboot |
| postfix | Mail nagios | /var/spool/postfix | /etc/postfix |

| | | | |
|---|---|---|---|
| Cyrus-imapd | MailBox | /var/lib/imap | /etc/cyrus.conf /etc/imapd.conf |
| crond | X | X | /etc/cron.d |
| logrotate | X | X | /etc/logrotate.d |

Table 2-1. HA_MGMT service list

### 2.1.8 HA_NFS service

**HA_NFS** is the name of the **NFS** service defined within **HA Cluster Suite**. For medium-sized clusters the **NFS** service will run on the Secondary Management Node, as this is an **active/active** architecture. For large clusters there are two servers dedicated to **NFS** High Availability. High Availability for the **NFS** service will work in the same way for both medium and large clusters.

| Service | Data which is shared | Mount point | Synchronized Configuration file |
|---|---|---|---|
| NFS | Data for network file system | /homenfs (for example) | /etc/exports |

Table 2-2. HA_NFS service

## 2.4 Understanding High Availability Scripts for the Management Node

The **/etc/clustmngt/ha** directory contains all the configuration files used to administer High Availability for the Management Node. These files need to be modified by the System Administrator to configure High Availability for the cluster Management Nodes.

When **HA Cluster Suite** is configured, the **/usr/sbin/haservices** script is defined as the main service for all the management functions. This script is the entry point for implementing High Availability for the cluster management services.

All the High Availability scripts used on the Management Node are in the **/usr/lib/clustmngt/ha/bin** directory. The list of scripts is as follows:

| | |
|---|---|
| **dbmconfig** | Updates files according to the Cluster Database information. |
| **haadminnodes** | List all the management nodes listed in the Cluster Database. |
| **hacron** | Enable/disable specific cron jobs, associated with the Management Node. This script modifies the **/etc/cron.d** directory. |
| **hapidcleaner** | Removes the **pid** files which may prevent some services from starting in a crashed environment |
| **hasyncadminnodes** | Manages the synchronization of files for the Primary and Secondary Management Nodes. |
| **haunwantedfiles** | Disables those services that could conflict with the administration of Management Node High Availability. |
| **haupdatedb** | Updates the database and defines the node which run this script as the active node. |

These scripts are configured by using the configuration files located in the **/etc/clustmngt/ha** directory.

### 2.4.1 /usr/sbin/haservices script

This script is used by **HA Cluster Suite**.

According to the parameter (**start**, **stop** or **status**), a range of commands, specified in the **/etc/clustmngt/ha/start**, **/etc/clustmngt/ha/stop** or **/etc/clustmngt/ha/status** data files will be executed sequentially.

Each file consists of a series of lines that include:

- Lines beginning with a hash symbol (#). These are comments.
- Empty lines. These are ignored.
- Lines that indicate a command and consist of the full path without any options or spaces.

## 2.4.2 /etc/clustmngt/ha/start file

This file contains the list of the commands to be carried out when the **haservices** script is launched with the **start** parameter. The commands are carried out in the order listed in the **start** file. When the first error occurs, **haservices** displays an error message and returns an error code.

### /etc/clustmngt/ha/start example file

Each line will be launched sequentially by the **haservices** script.

```
#each line will be launched sequentially, error are returned
/usr/lib/clustmngt/ha/bin/hapidcleaner
/etc/init.d/postgresql
/etc/init.d/bsm_nagios
/etc/init.d/syslog-ng

/etc/init.d/snmptrapd
#/etc/init.d/gmond
#/etc/init.d/gmetad
/etc/init.d/dhcpd
#/etc/init.d/ldap
/etc/init.d/httpd
/etc/init.d/conman
/etc/init.d/systemimager-server-rsyncd
# When using RMS Quadrics uncomment the next two lines
#/etc/init.d/msqld
#/usr/lib/clustmngt/bin/bin/harms

# At this step you should customize according to :
# When using LSF uncomment the next line
#/etc/init.d/lsf

# At this step you should customize according to :
# When using lustre uncomment the next line
#/etc/init.d/lustredbd.sh

# If you are using cyrus/imap uncomment the next two lines
#/etc/init.d/saslauthd
#/etc/init.d/cyrus-imapd

# If you are using dovecot uncomment the next line
#/etc/init.d/dovecot

# If you are using dovecot or cyrus uncomment the next line
#/etc/init.d/postfix

/usr/lib/clustmngt/ha/bin/hacron

#for management FDA storage
/usr/lib/clustmngt/ha/bin/hafda

#sleep 10 second before haupdatedb
/usr/lib/clustmngt/ha/bin/haupdatedb
```

## 2.4.3 /etc/clustmngt/ha/stop file

This file contains the list of the commands to be carried out when the **haservices** script is launched with the **stop** parameter. The commands are carried out in the order given in the **stop** file. When the first error occurs, **haservices** displays an error message.

The starting order of the commands is important. In the **stop** file the commands are usually executed in reverse order to that of the start file.

**/etc/clustmngt/ha/stop example file**

```
/usr/lib/clustmngt/ha/bin/hafda
/usr/lib/clustmngt/ha/bin/hacron

# If you are using cyrus/imap uncomment the next two lines
#/etc/init.d/saslauthd
#/etc/init.d/cyrus-imapd

# If you are using dovecot uncomment the next line
#/etc/init.d/dovecot

# If you are using dovecot or cyrus uncomment the next line
#/etc/init.d/postfix

# At this step you should customize according to :
# When using RMS Quadrics uncomment the next two lines
#/usr/lib/clustmngt/ha/bin/harms
#/etc/init.d/msqld

# At this step you should customize according to :
# When using LSF uncomment the next line
#/etc/init.d/lsf

# At this step you should customize according to :
# When using lustre uncomment the next line
#/etc/init.d/lustredbd.sh

#/etc/init.d/systemimager-server-rsyncd
/etc/init.d/conman
/etc/init.d/httpd
/etc/init.d/ldap
/etc/init.d/dhcpd
/etc/init.d/gmetad
/etc/init.d/gmond
/etc/init.d/bsm_nagios
/etc/init.d/snmptrapd
/etc/init.d/postgresql
```

## 2.4.4 /etc/clustmngt/ha/status file

The **haservices status** command will be executed periodically by **HA Cluster Suite**. An error on one or more services listed in this file will cause a **HA_MGMT** service failure.

**/etc/clustmngt/ha/status example file**

```
/etc/init.d/postgresql
/etc/init.d/syslog-ng
/etc/init.d/bsm_nagios
/etc/init.d/httpd
/etc/init.d/dhcpd
/etc/init.d/snmptrapd
#/etc/init.d/conman
#/etc/init.d/systemimager-server-rsyncd
```

### 2.4.5 hacron script

This script manages the crontabs on the Management Node. Those crontabs that need to run on the Primary Management Node have to be in the **/etc/cron.d** directory. These crontabs will need to be deactivated on the Secondary Management Node by adding them to the **/etc/cron.hasleep** directory.

When there is a call with the **start** parameter, the **hacron** script will move the files listed in the **/etc/clustmngt/ha/cron** file from the **/etc/cron.d** directory to the **/etc/cron.hasleep** directory in order to disable them.

When there is a call with the **stop** parameter, the **hacron** script will move the files listed in the **/etc/clustmngt/ha/cron** file from the **/etc/cron.hasleep** directory to the **/etc/cron.d** directory in order to re-activate them.

The **/etc/clustmngt/ha/cron** file ignores all empty lines; all commented lines starting with # are also ignored. All other lines are considered as the names of files included in **/etc/cron.d** directory.

### 2.4.6 hapidcleaner script

This script is used to clean the **pid** files which could interfere with the launching of services.

**Note** A **pid** file is a file containing in the first line an integer referring to a process.

The data file is **/etc/clustmngt/ha/pid2clean**. All blank lines, or those starting with #, are ignored. All other lines are considered as specifying the complete path for a **pid** file.

In the case of a call with the **start** argument, for each path specified in the **/etc/clustmngt/ha/pid2clean** file, the script checks if the corresponding process is in the course of being executed. If not, then the script deletes the file.

**/etc/HA/pid2clean example file**

```
# One filename per line. Files to remove at Cluster Suite startup
/var/lib/pgsql/data/postmaster.pid
```

### 2.4.7 hasyncadminnodes script

**Note** This script can only run on one of the Management Nodes at a time. Configure the following cron, **/etc/cron.d/hasyncadminnodes** to launch the script automatically on the Primary Management Node.

The corresponding data file is **/etc/clustmngt/ha/synchro**. All blank lines or those starting with # are ignored. All other lines are considered as specifying the complete path for a file.

In the case of a call with no arguments, the script connects to the passive node, checks each file specified in the **/etc/clustmngt/ha/synchro** file and returns a status different from zero when the first file is not synced with the Primary Node.

When the **-p** parameter is used, the script connects to the Secondary Node(s), and pushes each file that is not synced to the other node.

When the **-g** parameter is used, the script connects to the Secondary Node(s), and retrieves each file that is not synced.

The **–host <ipAddress>** parameter is used to specify a specific network host when the Cluster Database is not available.

The **-i** parameter is used to specify an interactive behaviour. You can choose either to push or to get a file.

The **–syncfile <syncfile>** parameter is used to specify an alternate synchronisation file, as and when required.

**/etc/clustmngt/ha/synchro file example:**

```
# Which files or to be synchronized ?
# One fullpath per line

/etc/clustmngt/ha/ip
/etc/clustmngt/ha/links
/etc/clustmngt/ha/mnttab
/etc/clustmngt/ha/pid2clean
/etc/clustmngt/ha/start
/etc/clustmngt/ha/status
/etc/clustmngt/ha/stop
/etc/clustmngt/ha/synchro

/etc/conman.conf
/etc/conman-tpl.conf
#/etc/dhcpd.conf
/etc/dhcpd-tpl.conf
/etc/exports
/etc/genders
/etc/gmetad.conf
/etc/gmond.conf
#/etc/hosts
/etc/hosts-tpl
/etc/resolv.conf
/etc/ssh/ssh_config
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/storageadmin/ddn_admin.conf
/etc/storageadmin/nec_admin.conf
/etc/storageadmin/storframework.conf
/etc/sudoers
/etc/syslog-ng/syslog-ng.conf
```

## 2.4.8 haunwantedfiles

This script deactivates those system services, at startup, that conflict with **HA Cluster Suite** and the administration of Management Node High Availability. This script must be run once when installing and each time there is an upgrade of the **Red Hat** or the **bullx cluster suite** system. For example, if it is decided to configure a Highly Available **postgresql** server then the dynamic data used by the **postgresql** database must be on the shared storage system. In order to avoid possible data corruption problems on the shared storage system, the **postgresql** server should only run on the active node, and the service should not be launched when there is a boot of a Management Node.

This script is based on the **noboot** configuration files in the **/etc/clustmngt/ha/** directory.

When this script is called, it reads **/etc/clustmngt/noboot** file and for each service listed runs the **chkconfig off** command.

**Note** Each service managed by **HA Cluster Suite** must be disabled at boot. Some **Lustre** services are managed by the **HA Cluster Suite** services.

## 2.5 Understanding Channel Bonding (Optional)

Implementing High Availability on a Management Node and the use or not of Channel Bonding are completely independent. It is the responsibility of the cluster administrator to decide whether or not to use Channel Bonding, assuming that the cluster equipment allows this.

**Note** If Channel Bonding is to be used it must be setup before installing High Availability on the Management Nodes.

In short, Channel Bonding makes it possible, when a machine is equipped with several Ethernet interfaces, to divide the work load or to implement High Availability on the different Ethernet interfaces.

As an example, consider a node equipped with two Ethernet interfaces (**eth0** and **eth1**) both of which are connected to the same local network. Either one or the other of these two interfaces can use the same IP address without disturbing the operation of the equipment connected to this local network.

Read the output of **modinfo bonding** on your node for more information.

**See** http://linux-ip.net/html/ether-bonding.html for more information about Channel bonding.

# Chapter 3. Configuring Management Node High Availability

**Important** Management Node High Availability is not supported for all system releases. See the Software Release Bulletin delivered with your system for details of any limitations which may apply to your release.

## 3.1 Installing the Primary and Secondary Management Nodes

**Important** The IP addresses used will depend on the address plan for the system. Those used in this section are examples only.

In this section, we assume the following:

- The Primary Management Node is called node1 and the Secondary Management Node is called node2.
- The `node1` IP address is `10.1.0.1`
- The `node2` IP address is `10.1.0.2`

**Important** This chapter describes the installation and configuration of Management Node High Availability for a system which is installed from scratch, as described in Chapter 3 in the *Installation and Configuration Guide*.

### 3.1.1 Prerequisites for Management Node High Availability

- Install the two Management Nodes as follows:

  **Primary Node**
  See the *Installation Process Overview* table on page 3-2 of the *Installation and Configuration Guide*. Install and configure **bullx cluster suite** as described in STEPs 1 (except the section *Alias Creation on eth0*), 2, and 3 in Chapter 3.

**Note** The external storage system referred to in STEP 1 point 5) must be fully configured, and shared LUNs with sufficient space must have been created. Refer to Table 3.1.

  **Secondary Node**
  See the *Installation Process Overview* table on page 3-2 of the *Installation and Configuration Guide*. Perform the installation and configuration operations as described in STEP 1 points 1), 2), 3), 4) (except the section *Alias Creation on eth0*), STEP 2 point 1, and STEP 3 only the section - *Configure the kdump kernel dump tool* in Chapter 3.

**Important** The Management, Backbone and Interconnect networks must be configured on the Secondary Node. Nothing else must be configured on the Secondary Node.

- Prevent **HA Cluster Suite** from being started automatically by running the command below on both the Primary and the Secondary Nodes.

```
chkconfig cman off
chkconfig rgmanager off
chkconfig qdiskd off
```

- For clusters which include the **PBS Professional** Batch Manager, stop this from being started automatically by running the command below on both the Primary and the Secondary Nodes.

```
chkconfig pbs off
```

- For clusters which include the **LSF** Batch Manager, stop this from being started automatically by running the command below on both the Primary and the Secondary Nodes.

```
chkconfig lsf off
```

- Stop the services included in the **HA_MGMT** group of services on both nodes.

**Important** **It is essential that the postgres service is stopped on both nodes with the command below.**

```
/usr/sbin/haservices stop
```

- Disable the boot time system services listed in the **haunwanted** script on both nodes.

```
/usr/lib/clustmngt/ha/bin/haunwantedfiles start
```

- Check that the **BMC** for both nodes is reachable using the **ping** and **ipmitool** commands.

**Note** Do not uses aliases for the IP addresses for any of the nodes. **HA Cluster Suite** will create aliases later.

## 3.1.2 Configuring Channel Bonding

The **Channel Bonding** mechanism makes it possible for a system to treat two physical interfaces in a logical bond with only one IP address. It allows the kernel to provide a single logical interface for two physical links connected to two distinct switches, with only one bond being used at the same time. The name of the interface is indicated by the user, it is usually similar to "`bond0`".

**Note** Channel Bonding is **NOT** a mandatory feature for High Availability.

**Important** **The following tasks must be performed on EACH Management Node.**

Before beginning the configuration operation, stop the interfaces that will be used, using the **ifdown** command (in the example below these interfaces are named `eth0` and `eth2`).

```
ifdown eth0
ifdown eth2
```

1. **Channel Bonding**
   Define the following files In the **/etc/sysconfig/network-scripts** directory for each Management Node:

   **ifcfg-bond0 file**

```
DEVICE=bond0
ONBOOT=yes
TYPE=Bonding
BOOTPROTO=none
NETMASK=255.255.0.0
IPADDR=10.1.0.1
USERCTL=no
PEERDNS=no
IPV6INIT=no
NOZEROCONF=yes
```

| | |
|---|---|
| `NOZEROCONF=yes` | To override the default address provided by **Red Hat** |
| `MACADDR` | Should not be used |
| `NETMASK` and `IPADDR` | Depends on the Management Node network configuration. |

   **ifcfg-eth0 file**

```
DEVICE=eth0
ONBOOT=yes
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
ISALIAS=no
BOOTPROTO=none
NOZEROCONF=yes
```

   **ifcfg-eth2 file**

```
DEVICE=eth2
ONBOOT=yes
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
ISALIAS=no
BOOTPROTO= none
NOZEROCONF=yes
```

   The **ifcfg-eth0:1** file must be deleted, if it exists, because the IP alias is managed by **HA Cluster Suite** software. This file will exist on the Primary Management Node only.

   `HWADDR` should not be used.

   `IPADDR` and `NETMASK` are not used here.

2. **Gateway and Route**
   Rename the **route-eth0** file in **route-bond0,** if it is present.
   You may then find something similar to the following in the **route-eth0** file.
   `10.0.0.0./8 via 10.1.255.254`

**Note** Replace `10.0.0.0` with the IP address for your network and the second address (`10.1.255.254`) with the address of the switch.

3. **modprobe.conf**

   In the **/etc/modprobe.conf** file define the '*bonding*' module on each node.

   Assuming that **eth0** is the first interface for channel bonding, add the following lines for the **eth0** interface:

```
alias bond0 bonding
options bonding mode=1 downdelay=1000 updelay=1000 miimon=1000
primary=eth0
```

   At boot you will now have a bonding device on your node. The bonding device will check the connectivity of each network interface each second, and switch to the active interface if the primary fails.

4. **Restart the Network**

   Run:

```
service network restart
```

Check your newly created network interface using the **ifconfig** and **ping** commands.

## 3.1.3 Configuring Services

### Configuring NTP

**NTP** is Highly Available by default and so does not need any special configuration beyond what is described below.

It is assumed that `10.1.0.1` is the IP address of the Primary Management Node and `10.1.0.2` is the IP address of the Secondary Management Node. If you are using a local clock, edit the **/etc/ntp.conf** file to configure NTP, as follows:

1. On the Primary Management Node:

```
restrict default nomodify notrap noquery
restrict 10.1.0.2
restrict 10.0.0.0 mask 255.0.0.0 notrap nomodify
restrict 127.0.0.1
peer 10.1.0.2
tos floor 1 orphan 7
server  127.127.1.0     # local clock
fudge   127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
tinker panic 0
tinker stepout 0
```

2. On the Secondary Management Node:

```
restrict default nomodify notrap noquery
restrict 10.1.0.1
restrict 10.0.0.0 mask 255.0.0.0 notrap nomodify
restrict 127.0.0.1
peer 10.1.0.1
tos floor 1 orphan 7
server  127.127.1.0     # local clock
```

```
fudge   127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
broadcastdelay  0.008
keys            /etc/ntp/keys
tinker panic 0
tinker stepout 0
```

3. Restart the service on both nodes:

```
service ntpd restart
```

And check the service with the **netstat** and **ps system** commands.

### Configuring syslog-ng

To avoid socket creation problems during machine boot, it is recommended that a listening socket is created with the 0.0.0.0 address, rather than using other virtual or real addresses. This special listening address allows the process to listen to all the network interfaces of the node. A virtual address should be chosen for the sockets that will be used for the **syslog** client.

On each Management Node edit the file **/etc/syslog-ng/syslog-ng.conf,** as follows.

1. Search for all the lines, which contain the `SUBSTITUTE` string.
2. For each line that is found, check and if necessary modify the lines which follow, as indicated below:
   - For each `source` block which contains an IP address set the value **0.0.0.0** instead of the IP alias.
   - For each `destination` block which contains an IP address set the value **127.0.0.1**. However, using an alias IP address may be useful if the Administrator would like the passive Management Node to send its logs to the active Management Node.

## 3.1.4 Change KSIS Post Configuration

The **NTP** service on the nodes has to include the IP addresses of both Management Nodes.

1. Disable the post deployment configuration of **NTP**. In the example which follows the Primary Management Node has an IP address of 10.0.0.1 and the Secondary Management Node has an IP address of 10.0.0.2. The command below disables the **ksis** post deployment configuration of the **NTP** service and has to be launched on the Primary Management Node:

```
ksis postconfig disable CONF_20_NTP
ksis postconfig buildconf
rsync -a --del /etc/systemimager/ root@10.0.0.2:/etc/systemimager
```

2. Deploy a node reference image to one node of each type e.g. **Login**, **I/O**, **COMPUTE(X)**.

```
ksis deploy < Node_image> node
```

3. Configure **NTP** on each deployed node type by adding two server lines for the Management Nodes in the **/etc/ntp.conf** file, as shown in the example below:

```
server 10.0.0.1
server 10.0.0.2
driftfile driftfile /var/lib/ntp/drift
```

4. Make an image of each node type:

```
ksis create <New_ Node_image> <New_ Node>
```

5. Deploy the new image to all the nodes of that type in the cluster.

```
ksis deploy <New_Node_image> node[1-x]
```

6. Run **postconfig.**

## 3.2 Implementing the Primary Node

The steps to implement High Availability on the primary node are as follows:

**Note** A Management Node, node1, is used in the description which follows.

### 3.2.1 Tuning lpfc parameters for NFS High Availability

**Note** This section only applies to Management Nodes which include the HA_NFS Service. It is mandatory for these nodes.

1. In the **/etc/modprobe.d/lpfc** file, add the **options lpfc lpfc_nodev_tmo=5** line before the lines below :

```
install lpfc modprobe -i lpfc; logger -p local7.info -t "IOCMDSTAT" "LOAD lpfc";
remove lpfc logger -p local7.info -t "IOCMDSTAT" "UNLOAD lpfc"; modprobe -ir lpfc;
```

2. Identify the kernel version installed on the node by running the command:

```
uname -r
```

3. Save the old **initrd** image using the kernel version, identified above:

```
mv /boot/initrd-<kernel_version>.img /boot/initrd-<kernel_version>.img-orig
```

4. Generate a new **initrd** image:

```
mkinitrd -v /boot/initrd-<kernel_version>.img <kernel_version>
```

### 3.2.2 Prepare the Partitions

1. **Create the partitions on the external storage devices.**

**Notes**

- It is recommended to create one LUN per application. It is possible to partition a LUN into multiple partitions (for example **/dev/sdb** in **/dev/sdb[1..5]**) with the **fdisk** command, if the external storage has already been configured. The actual device names depend on the external storage configuration.

- Each storage block device must be labeled with the proper **mkfs** (ext3, xfs, gfs2) system command, when they are prepared.

**Important** **Each partition must be large enough for the size of the file system. See the recommended sizes in the table below. Large clusters are defined as having more than 100 nodes.**

| Service Group | Mount point | File System Type | | Label | Size Minimal | Recommended Size |
|---|---|---|---|---|---|---|
| | | Small Clusters | Large Clusters | | | |
| HA_MGMT | /var/lib/pgsql/data | gfs2 | ext3 | HA_MGMT:cdb | 2 GB | 2 GB |
| | /var/lib/ganglia/rrds | gfs2 | ext3 | HA_MGMT:ganglia | 20 GB | 20 GB |
| | /var/log/HOSTS | gfs2 | ext3 | HA_MGMT:syslog | 20 GB | 2 GB per node |
| | /var/lib/systemimager | ext3 | ext3 | images | 20 GB | |
| | /var/spool/postfix | gfs2 | ext3 | HA_MGMT:postfix | 2 GB | |
| | /var/lib/imap | gfs2 | ext3 | HA_MGMT:mail | 5 GB | |
| | /var/log/conman | gfs2 | ext3 | HA_MGMT:conman | 5 GB | |
| | /home/lustre/run | ext3 | ext3 | lustre_mgs | 2 GB | |
| | /etc/lustre | gfs2 | ext3 | HA_MGMT:lustre | 2 GB | |
| | /var/lib/ldap | gfs2 | ext3 | HA_MGMT:dblustre | | |
| HA_NFS | /homenfs | ext3 | ext3 | homenfs | 5 GB | 500 MB per user |
| LSF | /var/lib/lsf | gfs2 | gfs2 | HA_MGMT:lsf | 3 GB | 5 GB |
| PBS | /var/spool/PBS | gfs2 | gfs2 | HA_MGMT:PBS | 3 GB | 5 GB |
| SLURM | /gfs2shared/slurmdata | gfs2 | gfs2 | HA_MGMT:SLURM | 3 GB | 5 GB |

Table 3-1. Management Node partitions

## 3.2.3 Create the File System for HA_MGMT group of services

Prepare the file system on the shared storage device with the **mkfs <FStype>** command, where **FStype** refers to the **gfs2** or **ext3** file system type according to the service' and the size of the cluster, see *Table 3-1* .

For example, use the command below to prepare a **gfs2** file system for the **postgresql** service with the LUN matching the **/dev/sdc** block device.

```
mkfs.gfs2 -j 2 -t HA_MGMT:cdb /dev/sdc
```

For example, use the command below to prepare a **ext3** file system for the **postgresql** service with the LUN matching the **/dev/sdc** block device.

```
mkfs.ext3 -L HA_MGMT:cdb /dev/sdc
```

Other examples for other management services are shown below:

```
mkfs.gfs2 -j 2 -t HA_MGMT:ganglia /dev/sdd
mkfs.gfs2 -j 2 -t HA_MGMT:syslog /dev/sde
mkfs.ext3 -L images /dev/sdf
mkfs.gfs2 -j 2 -t HA_MGMT:mail /dev/sdg
mkfs.gfs2 -j 2 -t HA_MGMT:conman /dev/sdh
mkfs.gfs2 -j 2 -t HA_MGMT:pbs /dev/sdi
mkfs.gfs2 -j 2 -t HA_MGMT:lsf /dev/sdj
mkfs.gfs2 -j 2 -t HA_MGMT:SLURM /dev/sdk
```

**Note** You have to use the block device that corresponds to that which is shared by your storage system. Following the definition of the file system each storage device will be named using the **gfs2** label, for example **HA_MGMT:cdb**.

### Lustre clusters

If your cluster uses the **Lustre** file system, the following partitions have to be created:

```
mkfs.gfs2 -j 2 -t HA_MGMT:lustre /dev/sdl
mkfs.gfs2 -j 2 -t HA_MGMT:dblustre /dev/sdm
mkfs.gfs2 -L lustre_mgs /dev/sdn
```

**See** Chapter 4 - *Configuring Storage Management Services* in the *Installation and Configuration Guide* if you need to change the LUN configuration.

### stordepha command

Create the configuration files for the **stordepha** command now (The **stordepha** tool will be used to create the configuration file for **HA Cluster Suite**).

Edit the **/etc/storageadmin/ha/hafsmgmt.conf** file and add the file system types defined in *Table 3-1*, and uncomment each line as needed. The example below shows the file system types for a small cluster:

```
#LABEL=HA_MGMT:cdb  /var/lib/pgsql/data   gfs2
#LABEL=HA_MGMT:syslog     /var/log/HOSTS          gfs2
#LABEL=HA_MGMT:ganglia    /var/lib/ganglia/rrds   gfs2
#LABEL=HA_MGMT:conman     /var/log/conman         gfs2
#LABEL=images             /var/lib/systemimager   ext3
#LABEL=HA_MGMT:PBS        /var/spool/PBS          gfs2
#LABEL=HA_MGMT:dblustre /var/lib/ldap             gfs2
#LABEL=HA_MGMT:lustre     /etc/lustre             gfs2
#LABEL=HA_MGMT:postfix    /var/spool/postfix      gfs2
#LABEL=HA_MGMT:mail       /var/lib/imap           gfs2
#LABEL=lustre_mgs         /home/lustre/run        ext3
#LABEL=HA_MGMT:lsf        /var/lib/lsf/work       gfs2
#LABEL=HA_MGMT:SLURM      /gfs2shared/slurmdata   gfs2
```

### 3.2.4 Create the File System for the HA_NFS service

Prepare the file system on the shared storage device using the **mkfs** command. For example, use the command below to prepare a file system with the LUN matching the **/dev/sdn** block device.

```
mkfs.xfs  /dev/sdn
```

Create the configuration files for the **stordepha** command now. This is done by editing the **/etc/storageadmin/ha/hafsnfs.conf** file. In this file you set the labels or the devices, the mount points, the file system and the mount options, as shown in the examples below:

```
/dev/sdn /mount_point1 xfs
/dev/sdm /mount_point2 xfs rw,usrquota
```

```
LABEL=homenfs1 /mount_point1 xfs
LABEL=homenfs2 /mount_point2 xfs rw,usrquota
```

**Note** The **/mount_point1** and **/mountpoint2** directories must have been created previously on all the nodes. **rw,usrquota** are examples of mount options, these options will depend on the cluster.

### 3.2.5 Configure the Primary Node

**Important** **It is recommended to use the Heuristic function of stordepha (option -H) for High Availability node pairs.**

A configuration file has to be created for **HA Cluster Suite** using the **stordepha** tool, and then **HA Cluster Suite** launched on the **node1** Management Node using the **storioha** command.

1. Start the **postgresql** daemon:

```
service postgresql start
```

**Note** In the example that follows **ClusterDB** is loaded and the **postgresql** services is up and running.

2. Check which Management Nodes are listed by using the command:

```
/usr/lib/clustmngt/ha/bin/haadminnodes
```

In this example, the following will be displayed:

```
node1
node2
```

3. Change the status of the Secondary Management Node, as follows:

```
dbmNode set --name node2 --status managed
dbmConfig configure –-service syshosts
```

4. Run the command below in order to be able to use **phpPgAdmin:**

```
service httpd start
```

5. Use the **phpPgAdmin** web interface for the **ClusterDB** and change the value of the **CLUSTER.actif-ha** (TABLE.field) field to *true*.

6. Define the virtual IP address
   A virtual IP address for the backbone network for the **HA_MGMT** service must be defined and the value for the **CLUSTER.node_backbone_ipaddr** updated accordingly in the **ClusterDB** using the **phpPgAdmin** web interface.

7. For each service (**HA_MGMT** or **HA_NFS**), the alias IP addresses must be defined in the **ClusterDB** in the **cluster_ipv** table. Use the **phpPgAdmin** web interface to load the table:

   If the **HA_NFS** service is to be used then add all the IP address aliases to the **cluster_ipv** table, as shown below:

| Actions | | id | cluster_id | nw_type | service | ipaddr | vlan_id | comment |
|---|---|---|---|---|---|---|---|---|
| Éditer | Effacer | 1 | 1 | admin | HA_MGMT | 192.168.64.99 | 1 | *NULL* |
| Éditer | Effacer | 2 | 1 | interconnect | HA_MGMT | 172.20.64.99 | 1 | *NULL* |
| Éditer | Effacer | 3 | 1 | admin | HA_NFS | 192.168.64.88 | 1 | *NULL* |
| Éditer | Effacer | 4 | 1 | interconnect | HA_NFS | 172.20.64.88 | 1 | *NULL* |

4 ligne(s)

Figure 3-1. **cluster_ipv** table for the HA_NFS service

8. Create the **cluster.conf** file for the Primary Node. Verify that the Primary and Secondary Management Nodes are registered in an I/O cell by using the **storiocellctl** command.

   a. If the **HA_MGMT** and **HA_NFS** services are to be on the same Management Node then run the command below:

```
/usr/sbin/stordepha -c configure -i node1 -o admin+nfs [-H]
```

   b. If only the **HA_MGMT** service is to be included on the Management Node host with the **HA_NFS** service either not used OR on a dedicated I/O node, then run the command below:

```
/usr/sbin/stordepha -c configure -i node1 -o admin [-H]
```

9. Stop the **postgresql** daemon:

```
service postgresql stop
```

10. Launch **HA Cluster Suite** on the Primary Node (node1), with the command below.

```
storioha -c start
```

11. Check the status of **HA Cluster Suite** by running the **clustat** command:

```
clustat
```

This will return something like the following:

```
Cluster Status for HA_MGMT @ Mon Jan 12 18:29:50 2009
Member Status: Quorate

Member Name                              ID   Status
 ------ ----                              ---- ------
 node1                                     1 Online, Local, rgmanager
 node2                                     2 Offline

 Service Name                    Owner (Last)                    State
 ------- ----                    ----- ------                    -----
 service:HA_MGMT                 (none)                        disabled
 service:HA_NFS                  (none)                         disabled
```

### 3.2.6 Move the Dynamic Data for each service to the Shared File System

The dynamic data for the shared management services must now be moved to the shared storage system, as shown in the example below for **postgres:**

1. Create a temporary directory:

```
mkdir -p /mnt/tempshared
```

2. Mount the LUN of the shared storage to this directory:

```
mount LABEL=HA_MGMT:cdb /mnt/tempshared
```

3. Copy the data file - for **pgsql** in this example - to the mount point:

```
cp -prv /var/lib/pgsql/data/* /mnt/tempshared
```

4. **umount** the temporary directory:

```
umount /mnt/tempshared
```

5. Remove the local files:

```
rm -rf /var/lib/pgsql/data/*
```

6. Mount the data for the services:

```
mount LABEL=HA_MGMT:cdb /var/lib/pgsql/data
```

7. Start the service associated with the mounted data. Check the corresponding log file. Ensure that the correct owner and mode are set for the data. Stop the service.

**Note** Problems will result if the service has been incorrectly configured, or does not have the correct owner and mode set.

8. Repeat this step for all the services with shared data listed in *Table 3-1*.

**Important**

- When the **mount** command is used for the **postgres** service the owner and rights for the **/var/lib/pgsql/data** directory are changed and must be restored back manually to: owner=postgres, group=dba, rights =700

- When the mount command is used for the **ganglia** service the owner and rights for the **/var/lib/ganglia** and **/var/lib/ganglia/rrds** directories are changed and must be restored back manually to: owner=nobody, group=nobody, rights =*755*
- When the mount command is used for the mail service the owner and rights for the **/var/spool/mail** directory is changed and must be restored back manually to: owner=root, group=mail, rights =*775*
- When the mount command is used for the **ldap** service the owner and rights for the **/var/lib/ldap** directory is changed and must be restored back manually to: owner=ldap, group=ldap, rights =*700*
- Create the shared mount point directories listed in Table 2.1 if they do not exist already

## 3.2.7 Checking the HA Configuration for the Services

1. Check the configuration of the **HA_MGMT** services by running the commands below:

```
rg_test test /etc/cluster/cluster.conf stop service HA_MGMT
rg_test test /etc/cluster/cluster.conf start service HA_MGMT
```

2. Check that the output for the commands above is OK (No failure indicated in the output displayed).

**Note** The **rg_test** output is very verbose. It is only necessary to check the output to see if the services defined in the **/etc/clustmngt/ha/stop** and **/etc/clustmngt/ha/start** files have stopped and started correctly.

3. Stop the **HA_MGMT** services by running the command below:

```
rg_test test /etc/cluster/cluster.conf stop service HA_MGMT
```

**Note** This command is only used for checking the configuration and must not be used when the cluster is in production.

4. Stop **HA Cluster Suite:**

```
clusvcadm -d HA_MGMT
service gfs2 stop
storioha -c stop
```

## 3.2.8 Check the HA Configuration Files

The **/etc/clustmngt/ha** directory contains several configuration files that should be checked, to ensure that they are OK for the configuration of your Management Nodes.

**See** the previous chapter and the individual man pages for more information regarding these configuration files.

In particular, check the following files, which should be customized according to Lustre usage, Resource Manager and the mount points for the shared storage system:

/etc/clustmngt/ha/start
/etc/clustmngt/ha/stop
/etc/clustmngt/ha/status
/etc/clustmngt/ha/synchro
/etc/clustmngt/ha/cron

Run the command below on both Management Nodes to deactivate services and crons:

```
/usr/lib/clustmngt/ha/bin/haunwantedfiles start
/usr/lib/clustmngt/ha/bin/hacron stop
/usr/sbin/haservices stop
```

## 3.3 Implementing the Secondary Node

The Secondary Node must have the same packages installed on it as on the Primary Node. The different services must be configured individually, except those which are managed by High Availability.

The steps to implement High Availability on the Secondary Node are as follows:

### 3.3.1 Sync the ssh keys on both Management Nodes

1. Use the command below on the node1 Management Node:

```
ssh-copy-id root@node1
ssh-copy-id root@node2
```

2. Use the commands below on the node2 Management Node:

```
ssh-copy-id root@node1
ssh-copy-id root@node2
```

### 3.3.2 Retrieve the /etc/hosts from the Primary Management Node

Retrieve the /etc/hosts file from the primary Management Node and copy it on to the secondary Management Node.

```
scp /etc/hosts root@node2:/etc/hosts
```

### 3.3.3 Synchronize Data Files

From the Primary Management Node (node1), synchronize the data files for the services that cannot be shared by using the command:

```
/usr/lib/clustmngt/ha/bin/hasyncadminnode -p -host node2
```

### 3.3.4 Tuning lpfc parameters for NFS High Availability

**Note** This section only applies to Management Nodes which include the HA_NFS Service. It is mandatory for these nodes.

1. In the **/etc/modprobe.d/lpfc** file, add the **options lpfc lpfc_nodev_tmo=5** line before the lines below :

```
install lpfc modprobe -i lpfc; logger -p local7.info -t "IOCMDSTAT" "LOAD lpfc";
remove lpfc logger -p local7.info -t "IOCMDSTAT" "UNLOAD lpfc"; modprobe -ir lpfc;
```

2. Identify the kernel version installed on the node by running the command:

```
uname -r
```

3. Save the old **initrd** image using the kernel version, identified above:

```
mv /boot/initrd-<kernel_version>.img /boot/initrd-<kernel_version>.img-orig
```

4. Generate a new **initrd** image:

```
mkinitrd -v /boot/initrd-<kernel_version>.img <kernel_version>
```

### 3.3.5 Prepare the Mount Points for the Shared Partitions

From the secondary Management Node (node2), prepare and clean the directory for mounting the shared storage:

```
mkdir -p /var/log/HOSTS
mkdir -p /var/lib/ganglia/rrds
mkdir -p /var/log/conman
mkdir -p /var/lib/systemimager
mkdir -p /var/spool/PBS
mkdir -p /var/lib/ldap
mkdir -p /etc/lustre
mkdir -p /var/spool/postfix
mkdir -p /var/lib/imap
mkdir -p /home/lustre/run
mkdir -p /var/lib/lsf/work
mkdir -p /gfs2shared/slurmdata
```

**Note** Each node needs to have symmetrical mount points.

You have to remove the **postgres** database from the local storage by using the command below.

```
rm -rf /var/lib/pgsql/data/*
```

**mportant** **The postgres database has to be on the shared storage bays on not on the local disks. Do not use the command above if the postgres database is already mounted.**

## 3.4 Starting Management Node High Availability

1. Start HA Cluster Suite on both nodes at the same time by using the **pdsh** command:

```
pdsh -w node[1,2] "storioha -c start"
```

2. On the Primary Node start the services by running the following command, where HA_MGMT is the service group name defined in the **cluster.conf** file:

```
clusvcadm –e HA_MGMT
```

3. Check that **HA Cluster Suite** is running correctly by running the command:

```
clustat
```

4. Check the **/var/log/ha.log** file.
   The output should show that **node1** and **node2** are defined as member names and that **node1** is the 'owner', which means that it is the active node, as in the example below.

```
Cluster Status for HA_MGMT @ Fri Jan 16 17:13:13 2009
Member Status: Quorate

 Member Name                              ID   Status
 ------ ----                              ---- ------
 node1                                     1 Online, Local, rgmanager
 node2                                     2 Online, rgmanager

 Service Name                  Owner (Last)                  State
 ------- ----                  ----- ------                  -----
 service:HA_MGMT               node1                        started
 service:HA_NFS                (none)                      disabled
```

5. Use the commands below to try and identify the cause of any problems. Also, refer to Chapters 2 and 4 in this manual.

```
cman_tool nodes
cman_tool status
fence_tool dump
group_tool
clustat
rg_test
```

6. On the **Primary Node** run the command below:

```
dbmConfig configure --restart --force
```

7. Launch **ntpd** by running the command below:

```
service ntpd start
```

8. Relocate the **HA_MGMT** services on the **Secondary Node** by running the command:

```
clusvcadm -r HA_MGMT -m node2
```

9. On the **Secondary Node** run the command below:

```
dbmConfig configure --restart --force
```

**Note** If there are any problems **HA Cluster Suite** must be stopped by disabling each element in the following order: **gfs2**, **rgmanage**r and **cman**. If there are problems stopping **HA Cluster Suite,** reboot both nodes and fix the configuration problem.

## 3.5 Automatic Synchronisation of Configuration Files

Set up the automatic synchronisation of the configuration files on both nodes by editing one of the cron files on each Management Node. On the Primary Management Node the cron script is located in the **/etc/cron.d/hasyncadminnodes** directory. On the Secondary Management Node the cron script is located in the **/etc/cron.hasleep/hasyncadminnodes** directory.

# Chapter 4. Administrating Management Node High Availability

**Important** Management Node High Availability is not supported for all system releases.
See the Software Release Bulletin delivered with your system for details of any limitations which may apply to your release.

**HA Cluster Suite** is the tool that is used to manage Management Node High Availability. The commands and services, described below, are used to configure, manage and monitor High Availability on the Management Node.

## 4.1 Configuration Tools

### 4.1.1 stordepha command

This command, below, configures High Availability on the Management Nodes automatically:

```
stordepha -c configure -o admin -i <node1_name>,<node2_name> [-H]
```

**Note** It is not recommended to use the **HA Cluster Suite** graphical interface.

The command requires:

1. Cluster Database access to the Primary Management Node.
2. A Secondary Management Node connected to the Administration network.

Assuming the Primary Management Node can access the **ClusterDB**, run the **stordepha** command, as shown above, to set the High Availability configuration.

## 4.2 Management Tools

| | |
|---|---|
| **storioha** | To **start/stop** the **HA Cluster Suite** daemons. |
| **clusvcadm** | To manage **HA Cluster Suite** using a command line interface. |

### 4.2.1 /usr/sbin/storioha

The **HA Cluster Suite** daemons are launched using the **storioha** command:

```
pdsh -w node1,node2 'storioha -c start'
```

The **HA Cluster Suite** daemons are stopped using this command:

```
pdsh -w node1,node2 'storioha -c stop'
```

### 4.2.2 clusvcadm

This command is used on the Management Nodes where **HA Cluster Suite** has been started. The most useful options are:

| | |
|---|---|
| **clusvcadm -d <service name>** | Disable a service |
| **clusvcadm -e <service name>** | Enable a service |
| **clusvcadm -e <service name>** | Enable a service on the member node where the command is executed |
| **clusvcadm -r <service name> -m <member>** | Relocate an active service to a specific member node |

Refer to the man page for this command for more information.

```
man clusvcadm
```

## 4.3 Monitoring Tools

| | |
|---|---|
| **clustat** | The **clustat** command displays the status of the cluster. It shows membership information and the state of all the user services configured. For more information about this tool, refer to the man page. |
| **/usr/sbin/haservices status** | This is used by **HA Cluster Suite** to monitor the **HA_MGMT** group of services. The service status can be checked using this command. |

### 4.3.1 clustat

This command is used on the Management Nodes where **HA Cluster Suite** has been started. It is used to display the status for **HA Cluster Suite**, see the example below:

```
[root@devha0 ~]# clustat
```

This will give output similar to that shown below:

```
Cluster Status for HA_MGMT @ Fri Feb 13 10:56:49 2009
Member Status: Quorate

 Member Name                              ID   Status
 ------ ----                              ---- ------
 node1                                    1 Online, Local, rgmanager
 node2                                    2 Online, rgmanager

 Service Name              Owner (Last)              State
 ------- ----              ----- ------              -----
 service:HA_MGMT           node1                     started
 service:HA_NFS            node2                     started
```

You can use the **clustat** command with the XML display mode (**–x** option):

```
clustat –x
```

This will give output similar to that shown below:

```
name="service:lustre_xena10" last-owner="xena10"
name="service:lustre_xena11" last-owner="xena11"
```

### 4.3.2 haservices status

This command is used to monitor the status of the cluster **HA_MGMT** group of services:

```
[root@devha0 ~]# /usr/sbin/haservices status
```

This gives output similar to that below:

```
/etc/init.d/postgresql OK
/etc/init.d/syslog-ng OK
/etc/init.d/bsm_nagios OK
/etc/init.d/httpd OK
/etc/init.d/dhcpd OK
/etc/init.d/snmptrapd OK
haservices is running...
```

### 4.3.3 Debug Command

The **rg_test** command allows you to start any **HA Cluster Suite** Service by hand. It is used to detect errors when the cluster services start.

**mportant** **The rg_test command should only be used for debugging and should not be used on a cluster in production.**

The command in the example below debugs the start-up of the HA_MGMT service.

```
rg_test test /etc/cluster/cluster.conf start service HA_MGMT
```

Correct any problems that are detected and restart the services by running the commands below:

```
rg_test test /etc/cluster/cluster.conf stop service HA_MGMT
rg_test test /etc/cluster/cluster.conf start service HA_MGMT
```

## 4.4 HA Cluster Suite Service Dependencies

Each **HA Cluster Suite** service is dependent on other services according to its position in the software stack, as shown in Figure 4-1.

### 4.4.1 HA Cluster Suite Software Stack

The **HA Cluster Suite** software stack includes five levels of services. At the top (level five) there are the Highly Available services and at the bottom (level one), the Cluster infrastructure services.

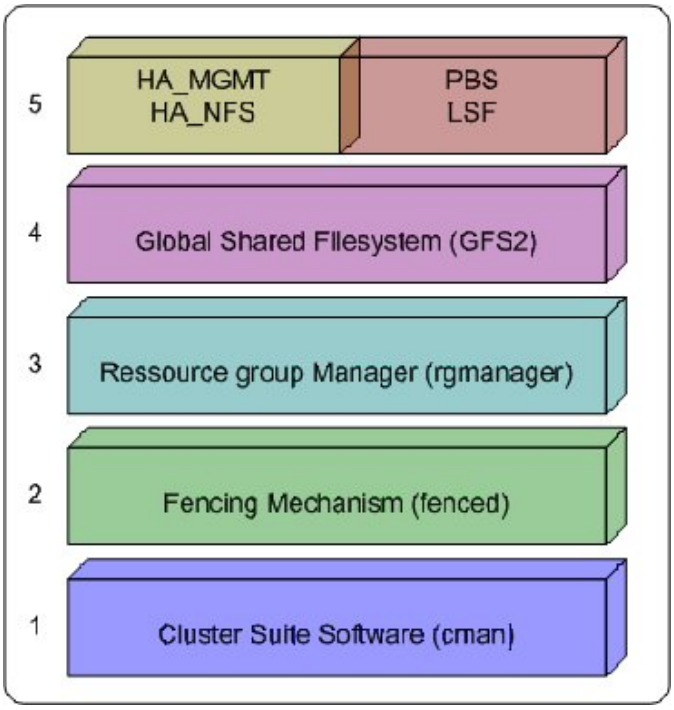


Figure 4-1. HA Cluster Suite Software Stack

| HA Cluster Suite Layer | Description |
|---|---|
| 5 | The service running on top of **HA Cluster Suite**: **PBS**, **LSF**, **HA_MGMT** and **HA_NFS** |
| 4 | The file system supporting the upper layer service, including the **gfs2** file system and the **dlm** kernel module. |
| 3 | HA Cluster Suite Resource group manager, used to assign services to nodes, checking service states, etc. |
| 2 | The fencing daemon and its mechanism needed to ensure the integrity of the cluster. |
| 1 | HA Cluster Suite daemon suite: **openais, ccsd, groupd** |

Table 4-1. Software Stack Layers

Each software stack layer element must be activated one after another. The **HA Cluster Suite** stack can be started manually by starting **cman (fenced** is launched by **cman)**, **rgmanager** and then **gfs2** file system afterwards. **HA Cluster Suite** is stopped by disabling each stack element in the reverse order. The **storioha** command allows you to start **cman** (including **fenced**) and **rgmanager** at the same time. It is recommended to start and stop the three first levels of **HA Cluster Suite** with the **storioha** command. Levels four and five have to be enabled and disabled by hand.

The commands listed below dialog with the **HA Cluster Suite** software stack.

**See** The man page for each command below for more information about it.

**Level 5 commands**

- system commands **clusvcadm, clustat, rg_test**
- service commands **etc/init.d/lsf, /etc/init.d/pbs /usr/sbin/haservices**

**Level 4 commands**

- system command **mount**
- service command **/etc/init.d/gfs2**
- specific daemon **gfs_controld, dlm_controld**

**Level 3 commands**

- system command **rg_test, clustat, storioha**
- service command **/etc/init.d/rgmanager**

**Level 2 commands**

- system command **storioha, ipmi_tool, fence_ipmilan**
- service command *no specific service command*
- specific daemon **fenced** (launched automatically by **cman** from Level 1)

**Level 1 commands**

- system command **stordepha, storioha, group_tool, cman_tool, fence_tool**
- service command **/etc/init.d/cman**
- specific daemon **ccsd, aisexec, groupd**

# Chapter 5. Load Balancing for Login Nodes

This chapter describes how to implement Load Balancing on the Login Nodes using **Linux Virtual Server** (LVS) and the **heart-beat** mechanism.

## 5.1 Introduction

The Login Node is the entry point used to access the cluster and its resources. The Login Node may be used by many different users at the same time, to either build their software application or to prepare jobs for submission via the batch manager.

For clusters with a single Login Node, the node may become overloaded with different user demands. Using multiple Login nodes with a load balancer helps to ensure the continuous availability of cluster resources for all the users. Load balancing, and High Availability services, require two nodes for a simple standard configuration. This configuration can be scaled according to the load of the cluster.

### 5.1.1 Definition of the Load Balancing Elements

The **Linux Virtual Server** - or **LVS** group - refer to the whole group of nodes involved in Load Balancing for the Login services. **LVS** is a Linux product to implement load balancing on the **IP** stack.

A resource in **LVS** terminology is a network connection (Layer 4) with a port, in our case **TCP port 22** corresponding to the **SSH** load balanced service.

**Heartbeat** is a GPL licensed portable cluster monitoring program that includes death-of-node detection, with a fencing mechanism to remove failed nodes from the cluster, and resource monitoring so that resources can be automatically restarted, or moved to other nodes when there is a failure. By default **Heartbeat** resources are assigned to a node. Heartbeat nodes send and receive messages periodically to their peer node in the network.

The **Active Load Balancer** node, also known as the **Active Director**, owns the **VIP** (Virtual IP address) and distributes packets, and balances the load, on the **Real Server Nodes. Heartbeat** gives the **VIP** address to the **Active Director.**

The **Passive Load Balancer**, node, also known as the **Passive Director**, is the back-up system in place in case of failure of the **Active Load Balancer**.

The **Director** which is active manages the resources for the **LVS** group. The **heartbeat** mechanism controls which Director is active at any one time.

The **Real Server Nodes**, refer to the physical nodes that are hosting the virtual Login services. Each client node connects to the Real Server nodes using the **VIP** for the **Active Director.**

### 5.1.2 Pre-Requisites

The minimal configuration contains only two nodes. The **Load Balancer**, **High Availability** software and the Login services run on two nodes at the same time.

The **Active** and **Passive Load Balancer** nodes MUST be symmetrical in terms of software and hardware. Any Login Node failure and change in the Active Load Balancer Node must

be transparent to the user, so that the environment and the tools available are always the same.

## 5.2 Login Node Architecture for Load Balancing High Availability

### 5.2.1 Basic Design Approach

The **Direct Routing** Load Balancer configuration, is used to ensure simplicity and performance. A client node creates a new connection to the Login service via a qualified domain name or **VIP** owned by the **Active Director** (Load Balancer). The **Active Director** receives incoming packets from all the Client Nodes, and for each packet checks for an **IPVS** server rule that corresponds to it. If, and when, a rule is found for the packet, it is then forwarded to a specific **Real Server.** The IP datagram is not modified, but the Ethernet frame is rewritten and pushed on the network.

For example, a packet entering the IP stack of the Active Director with a destination IP address of **x.y.z.a** and a **TCP** port of 22 is forwarded to a Login Node specified in the cluster. The Login Node receives the packet, and replies to the client directly (the packet is sent to the gateway of the Login Node).

The **Active Director** forwards the packet to a Login Node, included in the **Real Server Node** list, and maintains an internal state connection table for the client connections.

Each IP packet is forwarded without modification to a **Real Server Node** (a Login Node) that matches a connection table entry. If the entry does not exist in the table, it is created.

Multiple connections from the same client node IP address are forwarded to the same **Real Server Node.** The client node is a user node connected to the login service of the cluster infrastructure.

The **Least Connection Algorithm** loads, each Real Server Login Node with an approximately equal number of packet connections.

The Load Balancing architecture described, below, is simple and reduces the cost of having Login Nodes waiting indefinitely without work. Two **Director Nodes** and the **Real Server Node** are merged together. It is easy to add more **Real Server Nodes** (Login Nodes) - to this architecture.

### Load Balancing Job 1

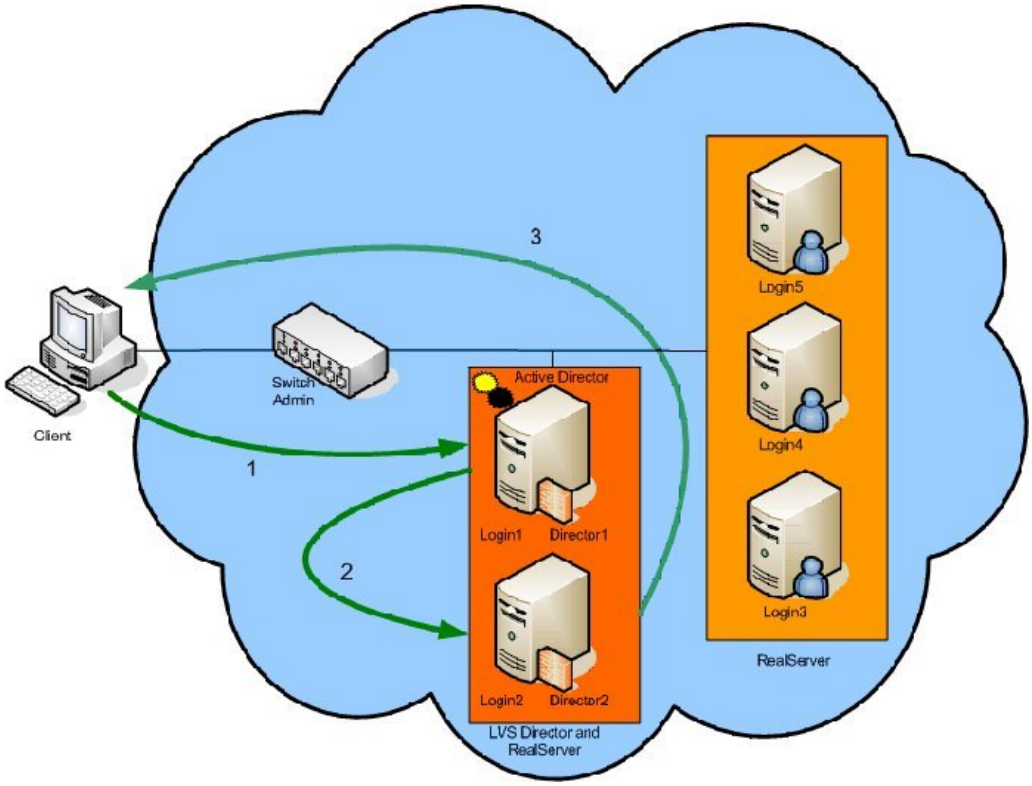


Figure 5-1. Load Balancing Job 1

In the diagram above, a client connects to the Login Services:

1. A user, on a client node, connects to the Login services using the **VIP** address owned by the **Active Director**.
2. The **Active Director** forwards it to a **Real Server Node** included in the cluster. In this example it is the **Passive Director** (Login2), but it could be any cluster Login Node, including the **Active Director**.
3. The **Real Server Node** (Passive Director/Login2) replies to the client, directly bypassing the **Active Director**.

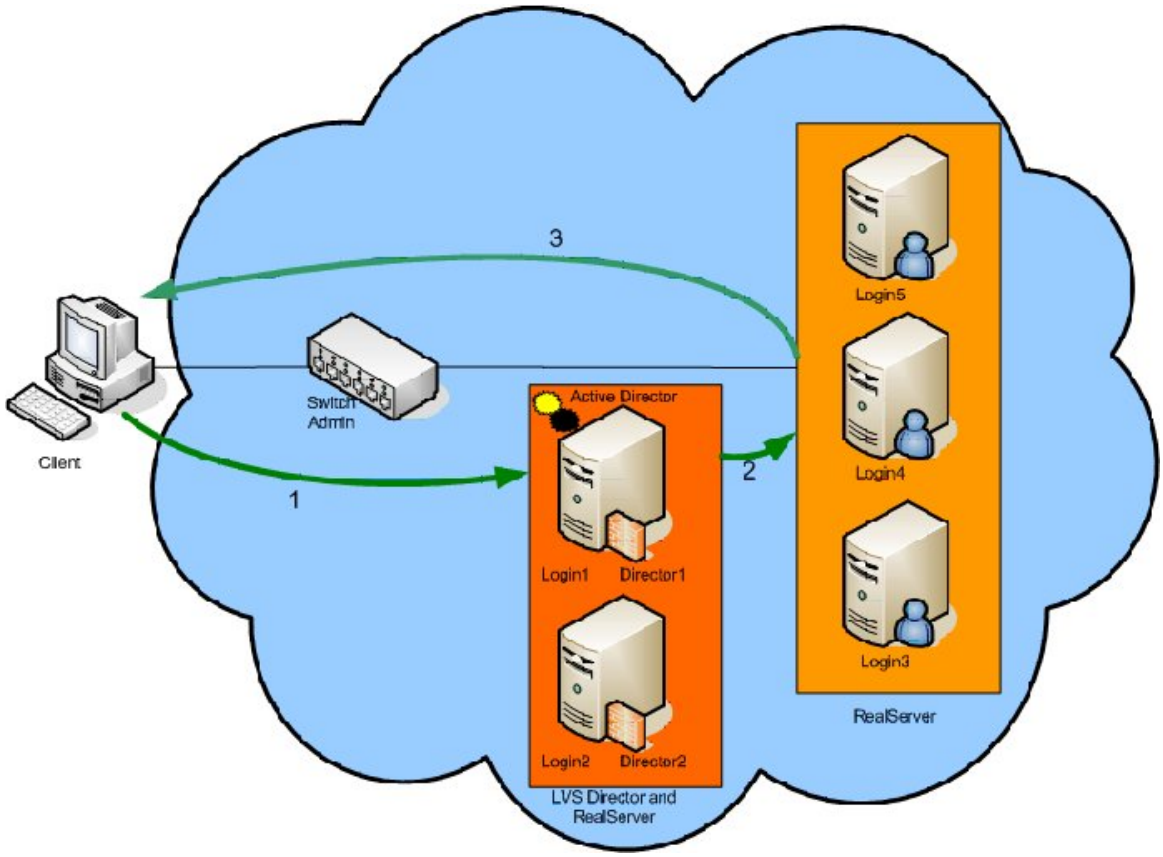


Figure 5-2. Load Balancing Job 2

In this diagram above, another client connects to the Login Services:

1. A user, on a client node, connects to the Login services using the VIP address owned by the **Active Director**.
2. The **Active Director** forwards it to a **Real Server Node** included in the LVS cluster. In this example it is a **Real Server Node** (Login4), but it could be any cluster Login Node, including the **Active Director**.
3. The **Real Server Node** (Login4) replies to the client, directly bypassing the **Active Director**.

## 5.2.2 Network Configuration for Linux Virtual Server

For the **Direct Routing** configuration, both the **Active Director** and **Real Server Nodes** need to declare a **VIP** address on their network interfaces. The **Active Director** adds this **VIP** to a physical network interface, for example **eth2**. The **Active Director** has to reply to the **ARP** request. **Real Server Nodes** need to add **VIP** addresses to their loopback interface. In our example, the **Active Director** is a **Real Server**, it needs to add this address to both the physical and the loopback interfaces.

**Important** **All the Director Nodes and the Real Server Nodes have to be on the same Ethernet network segment for the Direct Routing mechanism to work. IP packet forwarding from the Director Node to a Real Server Node is done at the Ethernet level.**

The **Keepalived** daemon, monitors the status of the **Real Server Node**s within a **LVS** cluster. **Keepalived** is configured to remove **Real Server Nodes** from the cluster group if they stop responding to a TCP ping, and to send a notification email to the System Administrator.

**Note** **Heartbeat** needs two network interfaces for message signaling. Standard network cards can be used, one on the Admin network, and the other on the Backbone network. The traffic generated by **Heartbeat** for these two interfaces is low and will not interfere with other network messages.

## 5.2 Implementing Load Balancing for the Login Nodes

In this example, we will set up two Login Nodes with an Active/Passive Director. The Active/Passive mechanism will be controlled by the **Heartbeat** mechanism.

The Network configuration is:

- Director1, Login1 132.167.130.211
- Director2, Login2 132.167.130.212
- Login3 132.167.130.214
- Login4 132.167.130.215
- Login5 132.167.130.216
- VIP 132.167.130.210
- Admin switch 132.167.130.1

The Heartbeat mechanism selects the **Active Director** from its configuration file (Director1 in this example). The **Active Director** manages the **VIP** address. It forwards each **IP** packet whose destination IP address equals the **VIP** address, and with TCP port 22 specified, to a Login node from the **Real Server Node** group.

The Heartbeat mechanism configures the **VIP** address as an alias for a physical network interface. There is no need to save a network configuration to a system file as the network configuration is set dynamically when **Heartbeat** starts. Each Login Node has to have a **VIP** address set up manually on the loopback interface for its **Real Server** equivalent.

### 5.2.3 Configure the Login Nodes

A Login Node has to accept packets with the **VIP** address as the destination. This **VIP** address is owned by the **Active Director**. Multiple network cards with the same IP address would confuse network communications. As loopback interfaces cannot send **Ethernet** communications, and **Address Resolution Protocol** (ARP) requests, all the Login Nodes should be configured to use the same **IP** address for the loopback interface.

This is done by configuring aliases for the loopback interface. Add this line to the /etc/sysconfig/network-scripts/ifcfg-lo:0 file for each Login Node:

```
DEVICE=lo:0
IPADDR=132.167.130.210
NETMASK=255.255.255.255
ONBOOT=yes
```

Then restart and check the network interface, using the command below.

```
/etc/init.d/network restart
ifconfig -a
```

### 5.2.4 Install the Load Balancing RPMs on the Director Nodes

Install the HeartBeat and KeepAlived RPMS, available in the BONUS directory on the XHPC media on to each Director node. Some of these RPMs are dependent on the PERL libraries found in the common directory on the XHPC media.

### 5.2.5 Configuring Load Balancing on the Director Nodes

Load Balancing on the Director Nodes is carried out using the Keepalived software and the parameters specified in the /etc/keepalived/keepalived.conf configuration file, example below.

```
         global_defs {
                lvs_id LVS_MAIN
        }

        virtual_server 132.167.130.210 22 {
                # 5 second timeout
                delay_loop 5

                # Least Connection Algorithm
                lb_algo lc

                # Direct Routing
                lb_kind DR

                protocol TCP

   #Real Server declaration
   #2 second timeout before suppress it from the spool

                real_server 132.167.130.211 22 {
                        weight 1
                        TCP_CHECK {
                                connect_timeout 2
                        }
                }
                real_server 132.167.130.212 22 {
                        weight 1
                        TCP_CHECK {
                                connect_timeout 2
                        }
                }
                real_server 132.167.130.213 22 {
                        weight 1
                        TCP_CHECK {
                                connect_timeout 2
                        }
                }
```

```
                real_server 132.167.130.214 22 {
                        weight 1
                        TCP_CHECK {
                                connect_timeout 2
                        }
                }
                real_server 132.167.130.215 22 {
                        weight 1
                        TCP_CHECK {
                                connect_timeout 2
                        }
                }
        }
```

Modify the **real_server** lines in the **keepalived.conf** file using your Login Node parameter values.

**Heartbeat** will start, stop and manage the **keepalived** daemon. Therefore the **keepalived** daemon needs to be disabled at startup by using the command below:

```
chkconfig --level 3  keepalived off
```

### 5.2.6 Configuring High Availability on the Director Nodes

Plug a cross-over network cable between the two Director Nodes, or use two different networks and configure a different IP address on each of them. This second interface will be a backup network interface for **Heartbeat** messaging. In the example, below, the network interface **eth3** is used with a cross-over cable.

1. On the first **Director Node**, add the IP address, below, to the **/etc/sysconfig/network-scripts/ifcfg-eth3** file:

```
DEVICE="eth3"
ONBOOT="yes"
BOOTPROTO="static"
NETMASK="255.255.255.0"
IPADDR="10.42.0.6"
```

2. On the second **Director Node**, add the IP address, below, to the **/etc/sysconfig/network-scripts/ifcfg-eth3** file:

```
DEVICE="eth3"
ONBOOT="yes"
BOOTPROTO="static"
NETMASK="255.255.255.0"
IPADDR="10.42.0.7"
```

3. Restart the network and check that the connection is OK for each Director Node.

```
/etc/init.d/network restart
ping 10.42.0.6
ping 10.42.0.7
```

4. Configure **Heartbeat** on each **Director Node**. The **Heartbeat** configuration file is in the **/etc/ha.d/** directory, and needs to be edited as follows:

```
logfile /var/log/ha-log
# on beat per second
keepalive 1
# 10 second before peer is seen as dead
deadtime 10
```

```
        warntime 5
      # 60 second start time
        initdead 60

      # backup heartbeat interface
        bcast   eth3

     # address from the peer LoadBalancer node on the admin network
        ucast eth0 132.167.130.212

      # don't rock over the service to the primary LoadBalancer if it
sudenly woke up
        auto_failback off

        # list the LoadBalancer node
        node    director1
        node    director2

        # heuristic to check the network ip connectivity before beeing
the Active LoadBalancer
# put your admin switch/router here
        ping 132.167.130.1
        respawn hacluster /usr/lib/heartbeat/ipfail
```

5. Configure the **VIP** on each network interface. A script is provided, below, to help do this. The first argument is the network interface, the second is the VIP address, and the third is **start stop** status. The **arping** command is mandatory; it updates the **ARP** entry for the **Active Director** on the Ethernet segment. Manually check that the equipment has updated the **ARP** entry with the **ping** and the **arp** commands.

   Modify the **NetMask** and **IP** address of the network switch according to the configuration details for your network. This script must be set up correctly on the two **Director Nodes** as it is used by the **haresources** file - the **Hearbeat** startup script. This script must be copied into the **/etc/ha.d/resource.d/IPalias** directory on both the Director Nodes.

```
        #!/bin/sh

        case $3 in
            start)
#Setup the network inteface according to haressource file
                /sbin/ifconfig lo:0 $2 netmask 255.255.255.255
                /sbin/ifconfig $1 $2 netmask 255.255.255.0

#Send arp request, arp reply to the router
                /sbin/arping -U -s $2 -c 5 -I $1 132.167.130.1
                /sbin/arping -U A -s $2 -c 5 -I $1 132.167.130.1
                ;;
            stop)
                /sbin/ifconfig $1 down
                ;;
            status)
                /sbin/ifconfig $1
                ;;
            *)
                echo "Called with unknown argument."
                exit 1
        esac
```

6. Set the **read** and **execute** bit for this script:

```
chmod 755 /etc/ha.d/resource.d/IPalias
```

7. Configure the network authentication for the heartbeat on the **Director Nodes**. This is done by editing the **/etc/ha.d/authkeys** file.

   a. Generate a random key by running the command:

```
cat /dev/urandom | dd count=1 2>/dev/null | sha1sum |cut -b 1-8
868c65f8
```

   b. Modify the**/etc/ha.d/authkeys** file and add the **auth key** values generated:

```
        auth 1
        1 md5 868c65f8
```

   c. Then copy the **/etc/ha.d/haresources** file on to each Director Node - service startup configuration file.

   d. Set **Heartbeat** to bring up the **IPV** alias and the **keepalived** daemon on the **director1** node using the **IPalias** script created before, using the command below.

```
        director1 IPalias::eth2:0::132.167.130.210 keepalived
```

**Note** In the command above the **VIP** address has been defined on the **eth2** network interface, as an example. Use the interface which is specific to your **Real Server** Ethernet network segment.

8. Start the **Heartbeat** mechanism and configure it to boot at startup:

```
chkconfig  --level 3 heartbeat on
/etc/init.d/heartbeat start
```

## 5.3 Checking the Load Balancing Setup

- Check the **syslog** file to verify that all the configuration details are correct.
- Check that the alias interfaces have been set by Heartbeat, for example, using eth2 on both Director Nodes. The **Ip addr** and **ifconfig** commands can be used to check this.
- Run the **ipvsadm** command to show the Ethernet routing rules for the **Active Director**.
- All the Login Nodes have to have a loopback interfaces configured on the IPV alias interface with a **netmask** of 32.
- Check the connection for all the Login Nodes by running the command below on each node individually:

```
ssh VIPAddress hostname
```

  If everything is correctly configured the hostname displayed will be different for each **Login Node.** This command must be executed from two different nodes.

Reboot the **Active Director** by connecting to its real **IP** address and not its virtual **IP** address. And check the other Director which will become active following the reboot of the first Director. Double-check that all the configuration details are correct. If everything appears OK reboot the second Director and run the tests on the first Director which will became active.

## 5.4 Configure the Login Node Services Symmetrically for all Login Nodes

All the cluster **Login Node**s need to be configured so that the Login services are synchronized on them in order to ensure consistency.

1. Synchronize all the Login Nodes with the remote NTP server on the Management Node.

**Note** Chapter 3 - *Configuring High Availability for the Management Node* for information on how to do this.

2. Run the command below to check that the clocks are synchronized on both nodes:

```
pdsh -w <Login_Node[1_to_x]> date
```

3. Synchronize the **sshd** configuration file located in the **/etc/ssh** directory on each **Login Node.**

   Run the command below to check this.

```
pdsh -w <Login_Node[1_to_x]> sha1sum /etc/ssh/*
```

4. Mount the **user** home directory from each **Login Node** on to the Management Node.

```
ssh <Login_Node> mount -t nfs <nfsshareip>:/homenfs /homenfs
```

Check that the same **RPM**s are installed on each Login Node.

# Chapter 6. NFSv3 High Availability for I/O Nodes

**Important** NFS High Availability for I/O nodes is not supported for all system releases.

See the Software Release Bulletin delivered with your system for details of any limitations which may apply to your release.

The High Availability solution described in this chapter is supported for NFS V3 ONLY.

## 6.1 Hardware Architecture

The management of a Bull High Availability **NFS** system requires a specific hardware architecture.

The I/O nodes operating in High Availability mode are grouped in an I/O cell, bringing together two I/O nodes that access the same disk array. The two I/O nodes are connected to separate switches, one for the BMC network, and one for the heartbeat Ethernet network, as shown in *Figure 9.1*.

Usually, nodes are connected directly to the storage systems ports, without intermediate switches or HUBs. This point to point linking avoids additional active components which increase the risk of system failure by being SPOFs (Single Point of Failure).

The LUNs within the storage systems are accessible for both nodes of the I/O cell. But each LUN must be used by only one node at a time, to avoid data corruption.

A fencing mechanism is implemented so that the faulty node cannot access the LUNs following a failure, and switch of active node, within an I/O cell. In addition a node which fails is powered off or forced to dump depending on the option specified with the **stordepha** command.

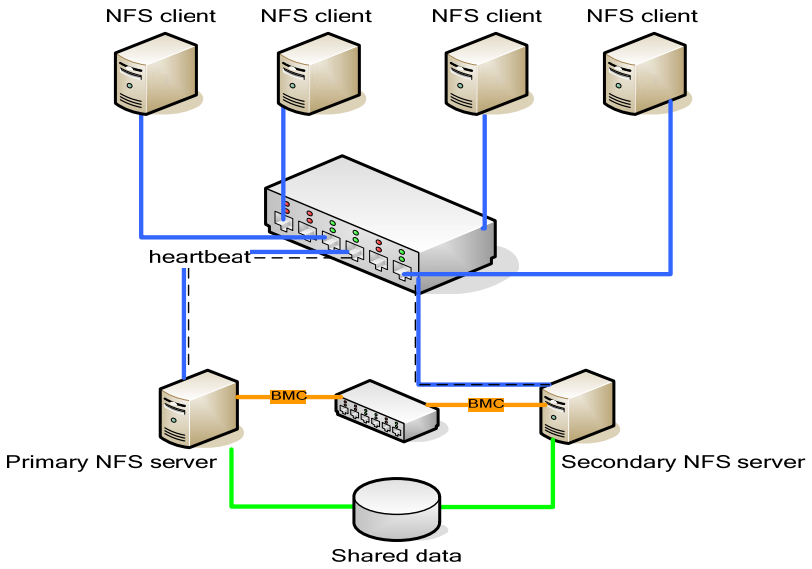


Figure 6-1. I/O Cell

An **active/passive** High Availability configuration is implemented for **NFS** services. This means that only the Primary Node is used for the NFS services, the virtual IP address, and for mounting the file systems. If the Primary Node/Server fails, the Secondary Node/Server takes over automatically. The transition will be seamless and the client will not be aware of the failover.

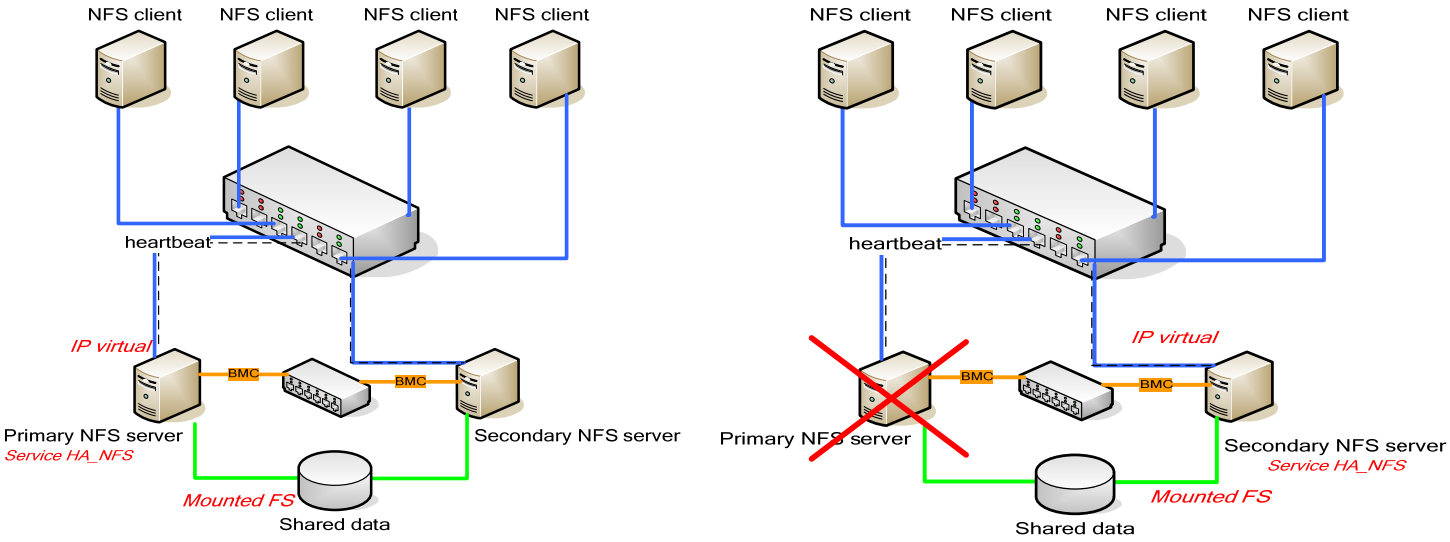


Figure 6-2. Active/Passive architecture for NFS services

HA Cluster Suite requires a service script to **stop**, **start** and report the **status** of each failover service that it manages. The failover script for the NFS service is the standard **/etc/init.d/nfs**.

## 6.2 Error Detection and Prevention Mechanisms

### NFS Single Point of Failure (SPOF) tracking

Tracked NFS service SPOFs include:

1. A service crash when **NFS** stops running on the node.
2. Mounted points no longer available
3. Unconfigured virtual IP addresses
4. I/O errors on the back-end device

A regular check of **NFS** services is scheduled for HA Cluster Suite by the failover script, and the results are returned to the status indicator.

HA Cluster Suite manages and monitors the mount points and the virtual IP address associated with the **HA_NFS** service.

The detection of I/O errors on the back-end device relies on the **storfilter** storage management monitoring daemon. When it detects a problem, this daemon powers off the node.

## 6.3 NFSv3 Failures covered by High Availability

### 6.3.1 I/O Node Failures

#### I/O Node Panic/Hang

When a Primary Node hangs or panics, the heartbeat message will not be sent within the authorized period. This silence is detected by the Secondary HA Node, which fences the silent node and then takes over the cluster services.

#### I/O Node Power down

The Primary Node does not send its heartbeat messages within the authorized period. This silence is detected by the Secondary HA node which fences the silent node, and then takes over the cluster services.

#### NFS Software Failure

HA Cluster Suite checks the **/etc/init.d/nfs** service periodically. This active check sets the current state of the **HA_NFS** service. If the **NFS** service is missing or in an incorrect state, HA Cluster Suite will try to restart it, and if the service is not in a correct state, HA Cluster Suite will migrate the service to the secondary node.

### 6.3.2 Storage Failures

#### Fibre Channel Adapter Errors

Fibre channel adapters are used to access the external storage systems, shared by the two nodes in the HA I/O cell. Fibre channel adapter errors are ignored. Fibre Channel Link errors are usually transient and do not lead to a node failover. If the adapter error is significant, it leads to a Linux disk error which triggers an alert. In this situation the Primary Node will power itself off, and be fenced by the Secondary Node and HA Cluster Suite will migrate the service to the Secondary node.

#### Disk Subsystem Controller Failure

The node detects a disk error when a disk controller fails. Disk errors are monitored by the **I/O status** service which alerts the I/O nodes. In this situation the Primary Node will power itself off, and be fenced by the Secondary Node and HA Cluster Suite will migrate the service to the Secondary node.

**Note** If the second disk controller fails it will not be possible to recover the system.

#### Local Disk errors

When a local disk fails HA Cluster Suite will switch to the Secondary Node.

### 6.3.3 Ethernet Network Failures

**Note** Only the heartbeat network is monitored by HA Cluster Suite. Any other Ethernet network failures will not lead to service takeover; however these failures will be displayed on the Management Node via **Bull System Manager - HPC Edition**.

#### BMC Ethernet Network Access Failure for the switch or link

The **BMC** management network is also used to send fence requests. The node which is unable to use the **BMC** management network cannot fence the other node in the High Availability pair. Thus, the second node which can access the Management network wins the fencing race, and takes over the cluster services. There is no risk of split brain (i.e. both nodes within the I/O cell simultaneously mounting the same device).

#### Ethernet Heartbeat Network (management or backbone) failure for the switch or link

A failure of the **Ethernet** heartbeat network stops heartbeat exchanges, and results in each node in the I/O cell attempting to take over the service. The **Heuristic** functionality (option **-H** of **stordepha**) is used to prevent the nodes from initiating a fencing race.

## 6.4 Configuring High Availability for NFSv3

Large clusters may contain multiple I/O cells, and HA Cluster Suite must be configured for each I/O cell. This process is fully automated by Bull Cluster Management tools. All the information necessary is taken from the Cluster Database and is used to generate the HA Cluster Suite configuration files. These files are pushed to all nodes that are managed by HA Cluster Suite.

**mportant** **HA Cluster Suite commands not described in this section must not be used, as they may lead to fatal inconsistencies for NFS. The GUI must not be used as well. The HA Cluster Suite setup is predefined to enable the failover process to work smoothly, and prevent the risk of split brain (i.e. both nodes of the I/O cell simultaneously running the same NFS service). The Administrator must not attempt to modify HA Cluster Suite's configuration for the I/O cells.**

The management tasks for HA Cluster Suite are:

1. Distributing the configuration file
2. Starting HA Cluster Suite.

By default, HA Cluster Suite is not started automatically at boot time, and it is recommended that this is NOT enabled.

For NFS High Availability, HA Cluster Suite manages the virtual IP addresses and the mount points. But the Administrator must configure the **/etc/exports** file to export file systems using shared storage, as HA Cluster Suite will not do this.

Two node NFS High Availability is designed around an **active/passive** architecture. The passive node is in place to take the load if the active node fails.

### 6.4.1 Pre-requisites

- The following package must be installed on the I/O Node : **storageadmin-node**
- The **I/O cell** for the I/O nodes must be defined in the Cluster Database. If this is not the case, use the **storiocellctl** command to define the I/O Cell.

**mportant** **Highly Available I/O NFS nodes must be deployed using a dedicated KSIS I/O node image that strictly excludes Lustre.**

## 6.4.2 Configuring Disks

To ensure the persistency of the storage devices following the reboot of different nodes, we encourage you to use a label instead of a block device in your configuration file.

Create one file system per block device, if this has not already been done. You will do this using the appropriate system command, for example, **mkfs.ext3**, or **mkfs.xfs**.

- This command creates an **ext3** file system with the *homenfs1* label on the **/dev/sdc** block device.

```
mkfs.ext3 -L homenfs1 /dev/sdc
```

- This command will do the same for a **XFS** file system.

```
mkfs.xfs -L homenfs2 /dev/sdc
```

### Setting labels for Storage Devices

If the storage devices have already been configured, it will be possible to obtain and set a label for a mounted file system.

1. For an **ext2**, **ext3** file system use the command, below, to visualize the property of the block device, and set the appropriate label:

```
tune2fs -l /dev/sdc           #show the current label on sdc
tune2fs -L homenfs1 /dev/sdc  #set the label homenfs1 on sdc
```

2. Use the command below to obtain the label of a **XFS** block device:

```
echo "label" | xfs_admin  /dev/sdc
```

3. Use the command below to rename the label **nfsuser** to **homenfs1**:

```
echo "label homenfs1" | xfs_admin  /dev/sdc
```

The **ext3** or **xfs** file systems must have been created using the appropriate commands (**parted**, **mkfs**) on the Primary Node.

These file systems will be managed by HA Cluster Suite, add the details about the file system to the **hafsnfs.conf** file. The file system details must NOT be included in the **/etc/fstab** file. Alternatively, use the **noauto** option to disable the **/etc/fstab** file, see **man fstab** for more information. Both nodes within the I/O cell will export the **ext3** and **xfs** file systems using HA Cluster Suite.

## 6.4.3 Configuring HA Cluster Suite

1. Configure NFS High Availability on both the NFS I/O nodes by editing the **/etc/storageadmin/ha/hafsnfs.conf** file. In this file you set the labels or the devices, the mount points, the file system and the mount options, as shown in the examples below:

```
/dev/sdn /mount_point1 xfs
/dev/sdm /mount_point2 xfs rw,usrquota
```

```
LABEL=homenfs1 /mount_point1 xfs
LABEL=homenfs2 /mount_point2 xfs rw,usrquota
```

**Note** The `/mount_point1` and `/mountpoint2` directories must have been created previously on all the nodes. **rw,usrquota** are examples of mount options, these options will depend on the cluster.

2. Choose the virtual **IP** address to be managed by HA Cluster Suite. The interfaces include the management interface, and possibly the backbone, and/or the **InfiniBand** interface. These IP address aliases are used to ensure that any server breakdown is transparent on the **NFS** client side.

   On the Management Node edit the **/etc/hosts-tpl** file, this is a template file used as a source for the **/etc/hosts** file. Add the IP address aliases, for example:

```
xxx.xxx.1.76       node1nfs           #node1nfs on admin network
xxx.xxx.xxx.135    node1nfs_BK        #same node on Backbone
xxx.xxx.1.76       node1nfs_ic0       #same node on the interconnect
```

3. Run the following command to generate the new **/etc/hosts** file:

```
dbmConfig configure --service syshosts --force
```

4. Deploy the **/etc/hosts** file on to the **I/O** and **NFS** client nodes:

```
pdcp -w node1,node2 /etc/hosts /etc/hosts
```

5. Run the **stordepha** command using the correct parameters, for example:

```
stordepha -c configure -o nfs -i node1,node2 [–H]
```

**Note** The **-i** option, above, refers to NFS **node1** and **node2** nodes. The **-H** option is recommended when configuring with the **stordepha** command.

**Important** **The netmask address must be the same as the primary IP address defined for the interface.**

6. It is possible to define the heartbeat network on the backbone network, as opposed to the administration network. In this case, and assuming the I/O nodes are connected to the backbone network, **stordepha** must be used with the **-hb** option, as shown below:

```
stordepha -c configure -o nfs -i node1,node2 –hb backbone [–H]
```

**See** The **stordepha – help** file for a description of the different parameters for the command.

7. The configuration file for the HA Cluster Suite (**/etc/cluster/cluster.conf**) is automatically generated and distributed on both nodes within an I/O cell.

### Restoring a node

After restoring the system on a node, the **cluster.conf** file has to be rebuilt using the **stordepha** command, as described above.

### 6.4.4 Starting HA Cluster Suite

1. To start HA Cluster Suite on the I/O nodes, run the command below from the Management Node remotely:

```
stordepha -c start -i node1,node2
```

   Or, locally on the node, run the command below:

```
storioha -c start
```

2. To check HA Cluster Suite's status, run the command below from the Management Node remotely:

```
stordepha –c status –i node1,node2
```

3. Activate the **HA_NFS** service with the command below on **ONLY ONE** of the I/O nodes:

```
ssh <node_name> "clusvcadm -e HA_NFS"
```

**Important** If an I/O node is fenced, **HA Cluster Suite** must be restarted after the reboot manually with the **stordepha** command.

### 6.4.5 Configuring NFS Servers

HA Cluster Suite manages the IP address aliases, the mount points, and the **NFS** service. The service is activated on only one node, as this is an **active/passive** configuration.

Therefore, on both I/O target nodes, you have to disable the **NFS** service from being activated during the boot sequence.

```
chkconfig --del nfs
```

For a **NFS** HA configuration, it is necessary to define the mount points to be exported to each I/O node, for example, on each NFS server:

1. Edit the **/etc/exports** file:

```
/mount_point1 *(rw,no_root_squash,sync,fsid=7)
/mount_point2 *(rw,no_root_squash,sync,fsid=8)
```

**Note** The **fsid** number varies according to the file system, it is mandatory to set a **fsid** on each file system for an **High Availability NFS** system.

2. Run the command below so that **NFS** High Availability is configured for the I/O cell:

```
exportfs -va
```

### 6.4.6 Configuring NFS Clients

**NFS** clients must mount the **NFS** file system using an administration IP address alias, and without any special options.

For example, add the mount point for the **/etc/fstab** file:

```
node1nfs:/mount_point1 /mount_point1 nfs defaults 0 0
```

**Note** **/mount_point1** must have been created previously on the target node. Then you can mount it with the **mount -a** option.

## 6.5 Monitoring NFS High Availability

### 6.5.1 System Log Files

HA Cluster Suite log events in the **/var/log/messages** and **/var/log/syslog** files on each I/O **NFS** node. The **syslog-ng** service centralizes these files on the Management Node.

### 6.5.2 Checking HA Cluster Suite settings

Once HA Cluster Suite activates the **NFS** service the settings can be checked using the commands below:

1. To check the mount points:

```
df
```

3. To check the alias IP addresses:

```
ip addr show
```

4. To check the status/localization of the NFS service:

```
clustat
```

# Glossary and Acronyms

## A

**ABI**
Application Binary Interface

**ACL**
Access Control List

**ACT**
Administration Configuration Tool

**ANL**
Argonne National Laboratory (MPICH2)

**API**
Application Programmer Interface

**ARP**
Address Resolution Protocol

**ASIC**
Application Specific Integrated Circuit

## B

**BAS**
Bull Advanced Server

**BIOS**
Basic Input Output System

**Blade**
Thin server that is inserted in a blade chassis

**BLACS**
Basic Linear Algebra Communication Subprograms

**BLAS**
Basic Linear Algebra Subprograms

**BMC**
Baseboard Management Controller

**BSBR**
Bull System Backup Restore

**BSM**
Bull System Manager

## C

**CGI**
Common Gateway Interface

**CLI**
Command Line Interface

**ClusterDB**
Cluster Database

**CLM**
Cluster Management

**CMC**
Chassis Management Controller

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

**Cron**
A UNIX command for scheduling jobs to be executed sometime in the future. A cron is normally used to schedule a job that is executed periodically - for example, to send out a notice every morning. It is also a daemon process, meaning that it runs continuously, waiting for specific events to occur.

**CUBLAS**
CUDA™ BLAS

**CUDA™**
Compute Unified Device Architecture

**CUFFT**
CUDA™ Fast Fourier Transform

**CVS**
Concurrent Versions System

**Cygwin**
A Linux-like environment for Windows. Bull cluster management tools use Cygwin to provide SSH support on a Windows system, enabling command mode access.

## D

**DDN**
Data Direct Networks

**DDR**
Double Data Rate

**DHCP**
Dynamic Host Configuration Protocol

**DLID**
Destination Local Indentifier

**DNS**
Domain Name Server:
A server that retains the addresses and routing information for TCP/IP LAN users.

**DSO**
Dynamic Shared Object

## E

**EBP**
End Bad Packet Delimiter

**ECT**
Embedded Configuration Tool

**EIP**
Encapsulated IP

**EPM**
Errors per Million

**EULA**
End User License Agreement (Microsoft)

## F

**FDA**
Fibre Disk Array

**FFT**
Fast Fourier Transform

**FFTW**
Fastest Fourier Transform in the West

**FRU**
Field Replaceable Unit

**FTP**
File Transfer Protocol

## G

**Ganglia**
A distributed monitoring tool used to view information associated with a node, such as CPU load, memory consumption, and network load.

**GCC**
GNU C Compiler

**GDB**
Gnu Debugger

**GFS**
Global File System

**GMP**
GNU Multiprecision Library

**GID**
Group ID

**GNU**
GNU's Not Unix

**GPL**
General Public License

**GPT**
GUID Partition Table

**Gratuitous ARP**
A gratuitous ARP request is an Address Resolution Protocol request packet where the source and destination IP are both set to the IP of the machine issuing the packet and the destination MAC is the broadcast address xx:xx:xx:xx:xx:xx. Ordinarily, no reply packet will occur. Gratuitous ARP reply is a reply to which no request has been made.

**GSL**
GNU Scientific Library

**GT/s**
Giga transfers per second

**GUI**
Graphical User Interface

**GUID**
Globally Unique Identifier

## H

**HBA**
Host Bus Adapter

**HCA**
Host Channel Adapter

**HDD**
Hard Disk Drive

**HoQ**
Head of Queue

**HPC**
High Performance Computing

**Hyper-Threading**
A technology that enables multi-threaded software applications to process threads in parallel, within each processor, resulting in increased utilization of processor resources.

## I

**IB**
InfiniBand

**IBTA**
InfiniBand Trade Association

**ICC**
Intel C Compiler

**IDE**
Integrated Device Electronics

**IFORT**
Intel® Fortran Compiler

**IMB**
Intel MPI Benchmarks

**INCA**
Integrated Cluster Architecture:
Bull Blade platform

**IOC**
Input/Output Board Compact with 6 PCI Slots

**IPMI**
Intelligent Platform Management Interface

**IPO**
Interprocedural Optimization

**IPoIB**
Internet Protocol over InfiniBand

**IPR**
IP Router

**iSM**
Storage Manager (FDA storage systems)

**ISV**
Independent Software Vendor

## K

**KDC**
Key Distribution Centre

**KSIS**
Utility for Image Building and Deployment

**KVM**
Keyboard Video Mouse (allows the keyboard, video monitor and mouse to be connected to the node)

## L

**LAN**
Local Area Network

**LAPACK**
Linear Algebra PACKage

**LDAP**
Lightweight Directory Access Protocol

**LDIF**
LDAP Data Interchange Format:
A plain text data interchange format to represent LDAP directory contents and update requests. LDIF conveys directory content as a set of records, one record for each object (or entry). It represents update requests, such as Add, Modify, Delete, and Rename, as a set of records, one record for each update request.

**LKCD**
Linux Kernel Crash Dump:
A tool used to capture and analyze crash dumps.

**LOV**
Logical Object Volume

**LSF**
Load Sharing Facility

**LUN**
Logical Unit Number

**LVM**
Logical Volume Manager

**LVS**
Linux Virtual Server

## M

**MAC**
Media Access Control (a unique identifier address attached to most forms of networking equipment).

**MAD**
Management Datagram

**Managed Switch**
A switch with no management interface and/or configuration options.

**MDS**
MetaData Server

**MDT**
MetaData Target

**MFT**
Mellanox Firmware Tools

**MIB**
Management Information Base

**MKL**
Maths Kernel Library

**MPD**
MPI Process Daemons

**MPFR**
C library for multiple-precision, floating-point computations

**MPI**
Message Passing Interface

**MTBF**
Mean Time Between Failures

**MTU**
Maximum Transmission Unit

## N

**Nagios**
A tool used to monitor the services and resources of Bull HPC clusters.

**NETCDF**
Network Common Data Form

**NFS**
Network File System

**NIC**
Network Interface Card

**NIS**
Network Information Service

**NS**
NovaScale

**NTP**
Network Time Protocol

**NUMA**
Non Uniform Memory Access

**NVRAM**
Non Volatile Random Access Memory

## O

**OFA**
Open Fabrics Alliance

**OFED**
Open Fabrics Enterprise Distribution

**OPMA**
Open Platform Management Architecture

**OpenSM**
Open Subnet Manager

**OpenIB**
Open InfiniBand

**OpenSSH**
Open Source implementation of the SSH protocol

**OSC**
Object Storage Client

**OSS**
Object Storage Server

**OST**
Object Storage Target

## P

**PAM**
Platform Administration and Maintenance Software

**PAPI**
Performance Application Programming Interface

**PBLAS**
Parallel Basic Linear Algebra Subprograms

**PBS**
Portable Batch System

**PCI**
Peripheral Component Interconnect (Intel)

**PDSH**
Parallel Distributed Shell

**PDU**
Power Distribution Unit

**PETSc**
Portable, Extensible Toolkit for Scientific Computation

**PGAPACK**
Parallel Genetic Algorithm Package

**PM**
Performance Manager

Platform Management

**PMI**
Process Management Interface

**PMU**
Performance Monitoring Unit

**pNETCDF**
Parallel NetCDF (Network Common Data Form)

**PVFS**
Parallel Virtual File System

## Q

**QDR**
Quad Data Rate

**QoS**
Quality of Service:
A set of rules which guarantee a defined level of quality in terms of transmission rates, error rates, and other characteristics for a network.

## R

**RAID**
Redundant Array of Independent Disks

**RDMA**
Remote Direct Memory Access

**ROM**
Read Only Memory

**RPC**
Remote Procedure Call

**RPM**
RPM Package Manager

**RSA**
Rivest, Shamir and Adleman, the developers of the RSA public key cryptosystem

## S

**SA**
Subnet Agent

**SAFTE**
SCSI Accessible Fault Tolerant Enclosures

**SAN**
Storage Area Network

**SCALAPACK**
SCALable Linear Algebra PACKage

**SCSI**
Small Computer System Interface

**SCIPORT**
Portable implementation of CRAY SCILIB

**SDP**
Socket Direct Protocol

**SDPOIB**
Sockets Direct Protocol over Infiniband

**SDR**
Sensor Data Record

Single Data Rate

**SFP**
Small Form-factor Pluggable transceiver - extractable optical or electrical transmitter/receiver module.

**SEL**
System Event Log

**SIOH**
Server Input/Output Hub

**SIS**
System Installation Suite

**SL**
Service Level

**SL2VL**
Service Level to Virtual Lane

**SLURM**
Simple Linux Utility for Resource Management – an open source, highly scalable cluster management and job scheduling system.

**SM**
Subnet Manager

**SMP**
Symmetric Multi Processing:
The processing of programs by multiple processors that share a common operating system and memory.

**SNMP**
Simple Network Management Protocol

**SOL**
Serial Over LAN

**SPOF**
Single Point of Failure

**SSH**
Secure Shell

**Syslog-ng**
System Log New Generation

## T

**TCL**
Tool Command Language

**TCP**
Transmission Control Protocol

**TFTP**
Trivial File Transfer Protocol

**TGT**
Ticket-Granting Ticket

## U

**UDP**
User Datagram Protocol

**UID**
User ID

**ULP**
Upper Layer Protocol

**USB**
Universal Serial Bus

**UTC**
Coordinated Universal Time

## V

**VCRC**
Variant Cyclic Redundancy Check

**VDM**
Voltaire Device Manager

**VFM**
Voltaire Fabric Manager

**VGA**
Video Graphic Adapter

**VL**
Virtual Lane

**VLAN**
Virtual Local Area Network

**VNC**
Virtual Network Computing:
Used to enable access to Windows systems and Windows applications from the Bull NovaScale cluster management system.

## W

**WWPN**
World–Wide Port Name

## X

**XFS**
eXtended File System

**XHPC**
Xeon High Performance Computing

**XIB**
Xeon InfiniBand

**XRC**
Extended Reliable Connection:
Included in Mellanox ConnectX HCAs for memory scalability

# Index

## S

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

REFERENCE
86 A2 25FA 03