Extreme Computing

Lustre Guide

REFERENCE
86 A2 46FD 00

Extreme Computing

Lustre Guide

Software

November 2009

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

REFERENCE
86 A2 46FD 00

Printed in France

Trademarks and Acknowledgements

Table of Contents

List of figures

List of tables

Preface

Scope and Objectives

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

This manual presents Lustre File System concepts, describes how to configure Lustre (including the **High Availability** configuration), and helps the System Administrator to optimize and troubleshoot Lustre.

Intended Readers

This guide is for System Administrators of Bull Extreme Computing systems.

Prerequisites

Important

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

Structure

This manual is organized as follows:

Chapter 1, **Chapter 2** and **Chapter 3**
Describe concepts for Lustre, with or without the High Availability feature.

Chapter 4 and **Chapter 5**
Describe how to configure Lustre, with or without the High Availability feature.

Chapter 6 *Administrating Lustre*, details some useful Lustre functions for the System Administrator.

Chapter 7 *Monitoring Lustre*, with or without the High Availability feature, using **Bull System Manager - HPC Edition.**

Chapter 8 *Optimizing Lustre*

Chapter 9 *Troubleshooting Lustre*, with or without the High Availability feature.

Appendix A Describes the syntax of the Lustre configuration files and the commands that appear in this Guide.

Bibliography

- The HPC / Extreme Computing manuals that apply to your software delivery are listed in the *Software Release Bulletin*.
- Lustre documentation from http://manual.lustre.org

Note The Bull Support web site may be consulted for product information, documentation, downloads, updates and service offers:
http://support.bull.com

Chapter 1. Lustre Concepts

This chapter explains how parallel file systems work on a Bull HPC / Extreme Computing system.

The following topics are included:

1.1 Parallel File Systems Overview

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

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

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

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

1.2 Lustre Overview

Lustre - a parallel file system - uses object based disks for storage. Metadata servers are used for storing file system metadata. This design provides a substantially more efficient division of labor between computing and storage resources. Replicated, failover metadata Servers (**MDS**s) maintain a transactional record of high-level files and file system changes. Distributed Object Storage Targets (**OST**s) are responsible for actual file system I/O operations and for interfacing with storage devices. This division of labor, and of responsibility, leads to a truly scalable file system, and more reliable recoverability from failures, by combining the advantages of journaling and distributed file systems. **Lustre** supports strong file and metadata locking semantics to maintain the total coherency of the file systems, even when there is concurrent access. File locking is distributed across the storage targets (**OST**s) that constitute the file system, with each OST handling locks for the objects that it stores.

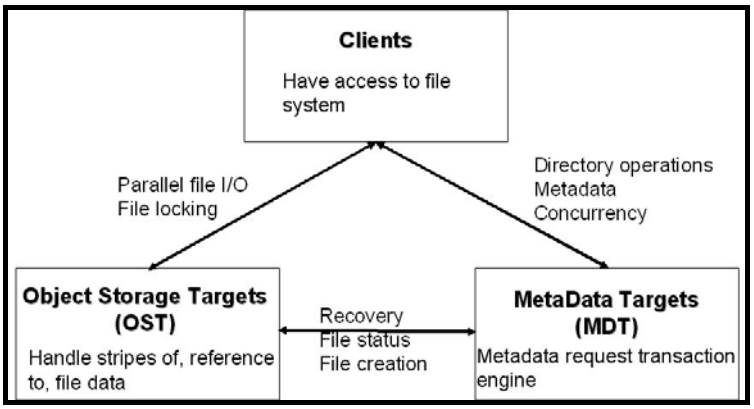

Figure 1-1. Lustre interactions

See Lustre: A Scalable, High-Performance File System Cluster File Systems, Inc. http://www.lustre.org/docs/whitepaper.pdf

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

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

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

MDS (MetaData Server)

MDS (MetaData Server) provides access to services called **MDTs** (MetaData Target).
A MDT provides a global NameSpace for the Lustre file system: it unifies the directory trees available from multiple file servers to provide a single global directory tree that can be mounted by the Lustre file system clients.

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

OSS (Object Storage Server)

OSS (Object Storage Server) provides access to services called **OST** (Object Storage Targets). An OST contains part of the file data (striping) for a given Lustre file system and very little metadata. Each OST has its own block device and backend file system where it stores stripes of files in local ext3-like files.

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

Lustre Client

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

- A client node mounts the Lustre file system over the network and accesses the files using POSIX semantics.
- Each client communicates directly with the MDS and OSS.

1.3 The Role of the Lustre System Administrator

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

- Determine how the hardware infrastructure will be used (number of file systems, size, storage resources used, allocation of I/O nodes, accessibility of the various file systems by the Lustre clients, etc.).
- If necessary, modify the configuration of the storage devices and the configuration of the Quadrics or InfiniBand interconnects (network zoning, etc).
- Configure the Lustre service and activate the configured file systems.

During the lifetime of the file system, administrators will have to perform stop, start, or repair operations. They may decide to update or change the configuration loaded. They will also need to monitor the file system and the quality of the service.

1.4 Planning a Lustre System

1.4.1 Data Pipelines

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

1.4.2 OSS / OST Distribution

The number of clients has no real impact on the number of OSSs to be deployed. The number of the OSSs and the distribution of the OSTs, are determined as follows:

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

To increase efficiency, it is preferable to distribute OSTs evenly on OSSs, and to have fewer larger OSTs so that the space is used more efficiently.

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

1.4.3 MDS / MDT Distribution

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

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

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

Table 1-1. Inode Stripe Data

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

1.4.4 File Striping

Lustre stripes the file data across the OSTs in a round robin fashion. It is recommended to stripe over as few objects as is possible, to limit the network overhead and to reduce the risk of data loss, in case of OSS failure.
The stripe size must be a multiple of the page size. The smallest recommended stripe size is 512 KB because Lustre tries to batch I/O into 512 KB blocks on the network.

1.4.5 Lustre File System Limitations

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

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

Chapter 2. Lustre HA for the I/O Nodes - Concepts

Important

Lustre File System HA (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.

This chapter explains how to implement High Availability for I/O Nodes and **Lustre** file system and only applies to clusters which have installed the **Lustre** file system from the **XLustre** media.

2.1 Lustre Failover Mechanism

Lustre supports the notion of failover pairs. Two nodes which are connected to shared storage can function as a failover pair, in which one node is the active provider of the service (**OST** or **MDT**), and the second node is the passive secondary server.

The Lustre services are declared on both nodes with the same name. The **MDT** is configured with a list of servers (**OSS**s) for clients to pass through in order to connect to the **OST**s. The Lustre servers must have distinct network addresses.

The failover mechanism of the Lustre system is based on the capacity to enable client reconnection when the **OST**s and **MDT**s are moved to other nodes.

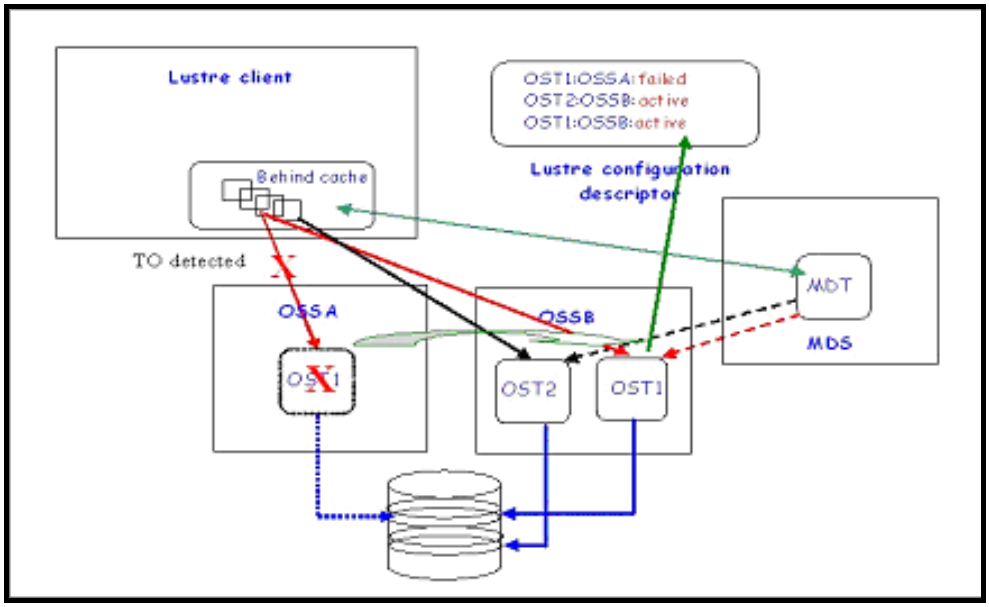

Figure 2-1. OST takeover and client recovery

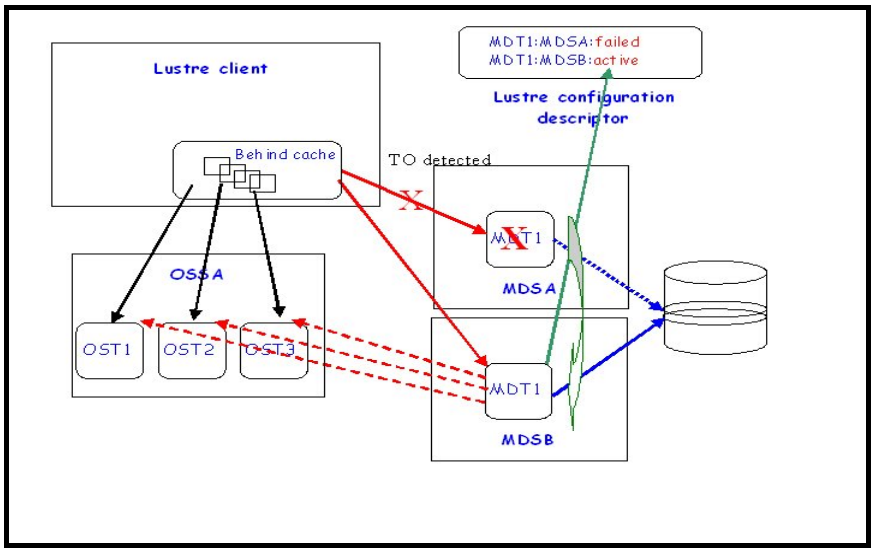

Figure 2-2. MDT takeover and client recovery

Lustre targets work in a synchronous mode: a client request is executed on the storage device and acknowledgement provided for the client only after it has been acknowledged by the device. In the case of a failure, the storage devices will ensure that committed data is preserved. Uncommitted data, meaning data not acknowledged, is kept by the clients in their "reserve cache" and can be resent when the system recovers.

When a request to a target (MDT or OST) is not acknowledged in time, the Lustre client suspects a failure and starts the recovery procedure as follows:

- Checks routing information in the Lustre configuration descriptor
- Reconnects to the migrated target
- Lost transactions are replayed
- Locks are re-acquired
- Partially completed multi-node operations are restarted.

On the server side, the target failover mechanism is similar to the High Availability service migration: the service is "stopped" on one node, and then restarted on the rescue node which has access to the same storage devices.

Transactions requested on metadata local to the targets (**ext3** log files) and those which are global for the Lustre system (MDT) are logged in log files. These files are replayed when there is a service migration.

Lustre does not prevent simultaneous access. This means that an external mechanism must ensure that the shared storage will only be accessed by one node at a time. This can be done by powering off the failed node.

This diagram below shows the sequence of events following a system failure leading to switch of nodes for the Lustre services. The switch is handled automatically by **HA Cluster Suite**.

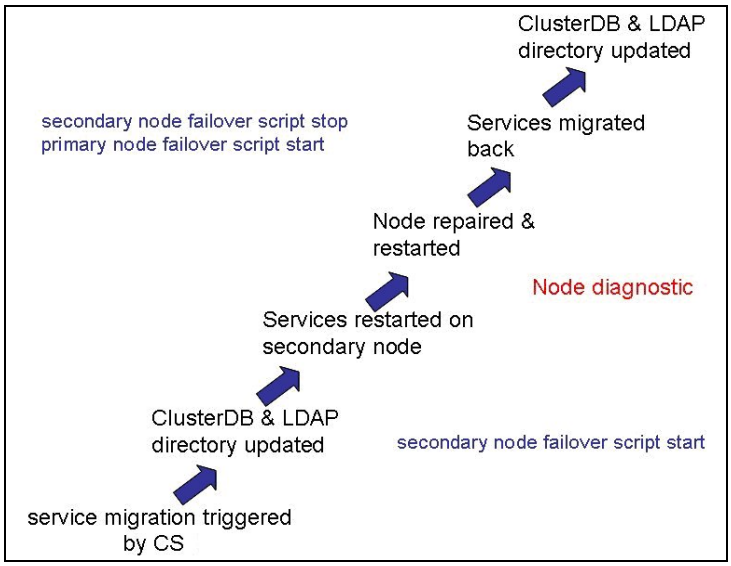

Figure 2-3. Service migration triggered by HA Cluster Suite

2.2 Hardware Architecture

Bull High-Availability management of the Lustre system relies on a specific hardware architecture.

The I/O nodes operating in HA mode are grouped in I/O cells which brings together two I/O nodes that access one or more disk arrays. The I/O cell contains either **OST**s or **MDT**s, but both are exclusive.

Usually, nodes are 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 also being **SPOF**s (Single Point of Failure).

The LUNs within the storage systems are accessible for both nodes of the I/O cell, enabling OSSs and MDSs to retrieve their data when they are moved to the other node. But each LUN must be used by only one node at a time to avoid data corruption.

An I/O fencing mechanism is implemented so that the faulty node can not access the LUNs again after the OSSs or MDSs are restarted on the peer node of the I/O cell. In any case the node which fails is powered off.

The underlying mechanism which ensures OSS and MDS migration and I/O fencing is provided by **HA Cluster Suite**. The failover process relies on basic entities known as failover services. When a node fails, **HA Cluster Suite** determines how each service should be relocated.

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).

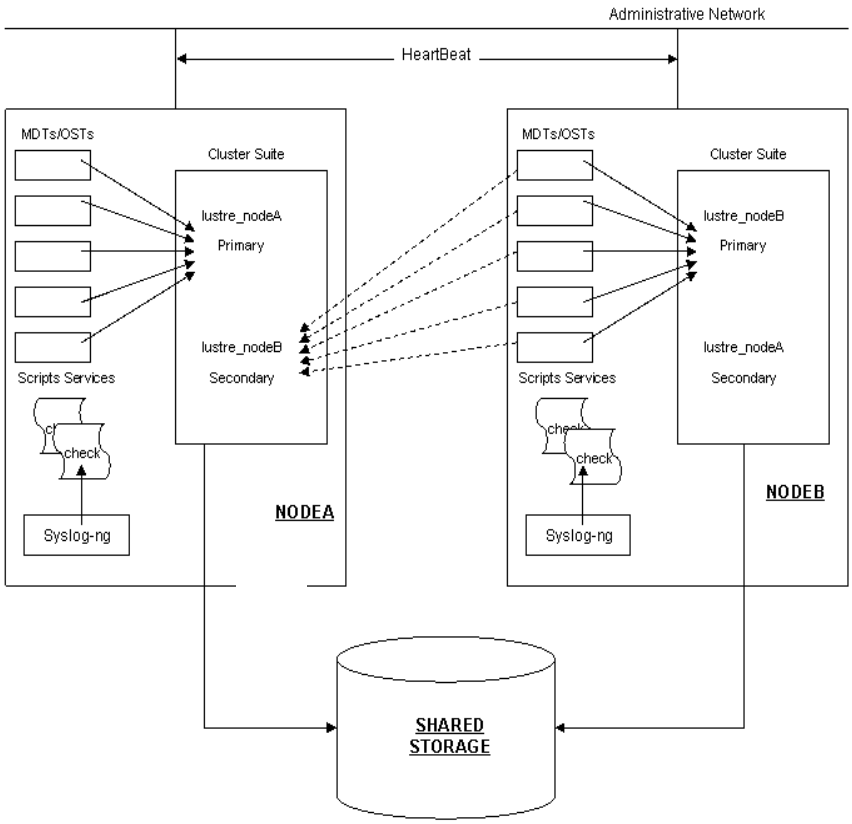

Figure 2-4. I/O Cell diagram

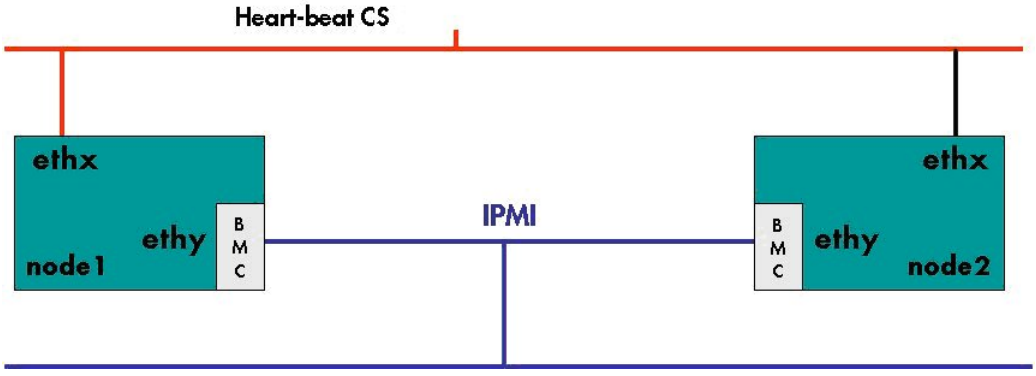

Figure 2-5. High Availability/HA Cluster Suite on NovaScale R440 and R460 IO/MDS nodes

In the case of multi-types I/O nodes (nodes which serve as both OSSs and MDSs), the Lustre file systems must be configured so that the same node does not support both MDT and OSTs services for the same file system. If not, a failure of this type of node constitutes a double failure (MDS + OSS) for the Lustre file system and its recovery is not guaranteed. The following figure illustrates how you can position OSTs and MDTs for two file systems FS1 and FS2.

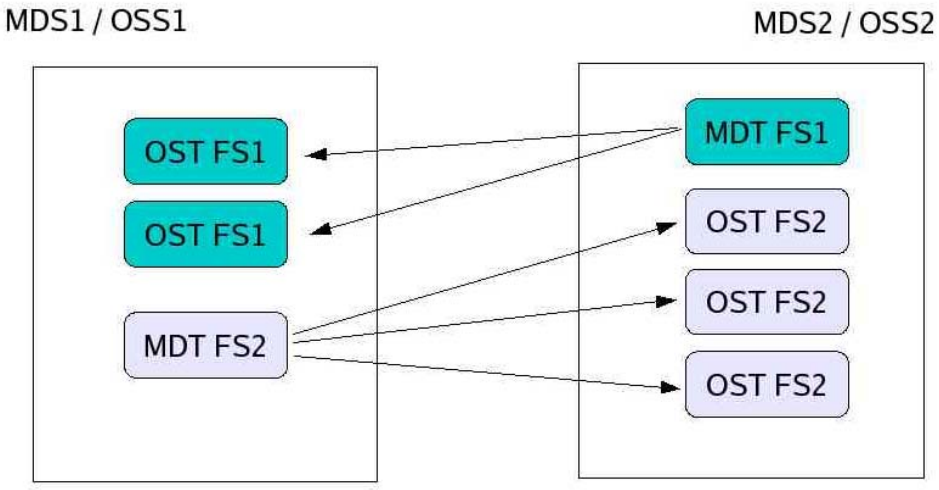

Figure 2-6. MDT/OST Dispatching on two nodes

2.3 High Availability Policy

In a Cluster Manager environment the customer can simply spread the application services across the clustered servers in any way that seems appropriate. All nodes will be actively running a share of the total load.

This is called **Active/Active clustering**, as all servers are active. If one server shuts down, the other servers will pick up the running load of its Lustre targets (MDTs or OSTs).

The High Availability mode to be applied in the Bull Extreme Computing context is ***mutual takeover*** for each node of a pair of nodes in the same I/O cell. In standard state, each I/O node supports its own Lustre targets, whereas in a failure state, one node manages its own Lustre targets plus all those from the failing node.

Firstly, all Lustre targets from the failing node will be migrated to the second node, even if the failure concerns only one Lustre target. It means that for each I/O node, one failover service is defined which includes all the currently installed Lustre targets.

HA Cluster Suite requires a service script for each failover service that will be managed. The script must be able to stop, start and report status for the service.

On each unitary High Availability cluster based on the I/O cells, two failover services are defined: one for the Lustre targets of each node.

In the I/O Cell above (Figure 2-4) we have:

lustre_nodeA with primary node NODEA and secondary node NODEB
lustre_nodeB with primary node NODEB and secondary node NODEA

A different failover script is associated with each of the two services provided they are not composed of the same Lustre targets (MDTs / OSTs).

At any moment, the Lustre failover service on an I/O node is composed of all the Lustre targets (MDTs / OSTs) associated with the **active** file systems. Its composition is subject to change according to the Lustre file systems activation.

On an I/O node, the Lustre targets (MDTs / OSTs) are not started when the system boots but only by Lustre administrative tools by means of file system start. This to ensure consistency of the Lustre file system targets start on all the nodes it relies on.

After a reboot of a failed MDS or I/O node, the Lustre failover services are not automatically relocated. This may be done only by the administrator using a Lustre management tool. This mechanism is chosen to avoid inopportune Lustre failover services migration when there is a partial repair of the primary node.

Important

A simultaneous migration of the management station and of the metadata server is considered as a double failure and is not supported. Likewise the simultaneous migration of MDS and OSS is not supported.

2.4 Managing Lustre High Availability

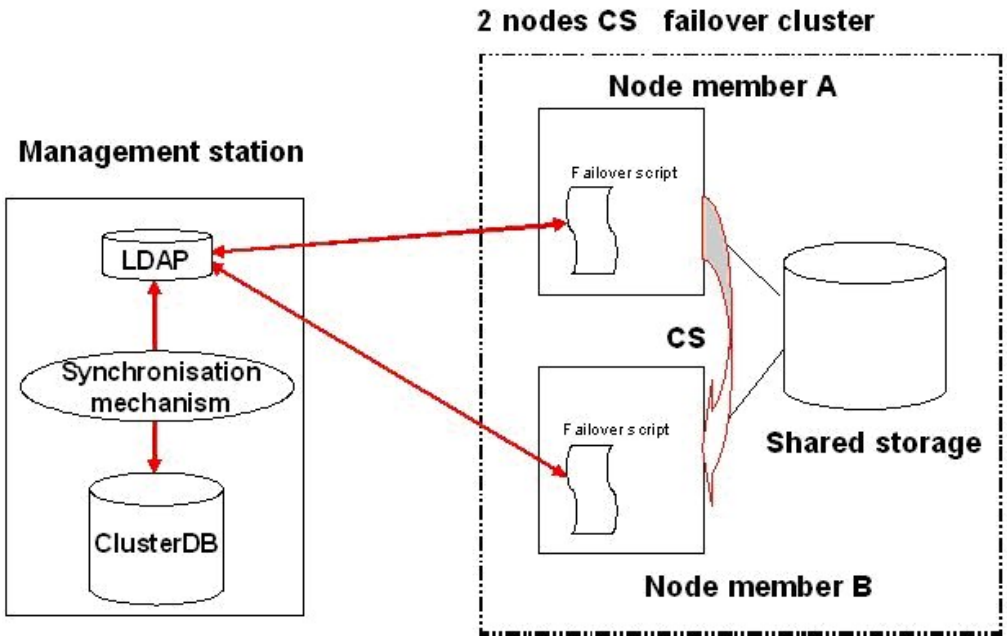

Figure 2-7. Lustre High-Availability Management architecture

When targets (MDT/OST) failover is configured, not only the information about paired targets (a cell) is stored in the Lustre configuration information of the ClusterDB, but also which instance of that target is the currently active one. It is the High Availability application (HA Cluster Suite script) that has to dynamically update the active instance information of the Lustre tables into the ClusterDB.

LDAP Directory

The LDAP directory is the pivot of the Lustre High Availability architecture. It has been proved to be the more flexible and more efficient way to share Lustre configuration information on a large cluster. It is the repository of the actual file systems distribution on the cluster:

- Which file systems are active,
- Lustre services (OST/MDT) migrations.

The information status it contains is updated by the Lustre failover script every time a High Availability event occurs.

This information status has to be synchronized with the ClusterDB one in order to ensure file systems status and Lustre services migrations monitoring by the Lustre administration tools.

Synchronization with the ClusterDB is carried out by an HTTP server named **lustredbd** which runs on the same node as the **LDAP** daemon.

Using such a mechanism allows Lustre file systems relying on some migrated nodes, to be stopped and restarted respecting the actual Lustre services node distribution.

Failover Scripts

The failover scripts for a HA pair of nodes will include different Lustre services. These change according which Lustre file systems are activated/de-activated.

A generic failover script **/usr/sbin/lustre_failover** is implemented on the I/O nodes. The set of Lustre services to be managed is determined dynamically by checking the **LDAP** directory for the active file system and its group of services on a node.
To ensure consistency of services, the failover script associated with a failover service is a symbolic link to the generic script whose name includes the primary node name:

```
/usr/sbin/lustre_failover_<primary_node_name>
```

The group of Lustre services to be checked is determined by parsing the script name. The symbolic link is configured when the cluster is deployed.
A **naming convention** is established which is used when the **Lustre** file systems and **HA Cluster Suite** are configured:

```
failover service name = lustre_<primary_node_name>
```

2.5 Error Detection and Prevention Mechanisms

Lustre Single Point of Failure (SPOF) tracking

Tracked Lustre services SPOF include:

1. Service crash (no longer running on the node).
2. Lustre services in an unavailable state (hanging, starting, etc.).
3. Repetitive abnormal comportment (systematic client eviction, etc.).
4. I/O errors on the back-end device.

The Lustre services failures (points 1 to 3) detection relies on the intrinsic Lustre health monitoring system. This internal failures management mechanism maintains diagnose items in the local **/proc/sys/lustre** and **/proc/fs/lustre** directories of each I/O or metadata node.

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

This check will process the diagnose items maintained by **Lustre**.

The detection of I/O errors on the back-end device (point 4) detection relies on the storage management monitoring daemon **storfilter**. When it detects a problem, this daemon warns the Lustre failover script using a dedicated target.

When one of these SPOFs is detected, the node is powered off followed by a Lustre services migration.

2.6 Lustre Failures covered by High Availability

2.6.1 I/O Node and Metadata 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.

Lustre Software Failure

Two scenarios can take place leading to the same action:

- The health monitoring mechanism of the Lustre system detects the failure and update the health information in the **/proc** directory. The next time the status target of the Lustre failover script is activated by the **HA Cluster Suite**, it will detect the problem and power off the failing node.
- The status parameter of the **Lustre** failover script detects that some **Lustre** services are missing, or not in the correct state. It will firstly try to restart the service, and if there is no response it will reboot the node where the service is running.

Both scenarios trigger the **I/O node power down** failure treatment.

2.6.2 Storage Failures

Fibre Channel Adapter Errors

Fibre channel adapters are used to access to external storage systems, shared by the two nodes of the HA I/O cell. They store the OSS and MDS data.
Fibre channel adapter errors are ignored. 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 off, be fenced by the Secondary Node, and **HA Cluster Suite** will switch 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 off, be fenced by the Secondary Node, and **HA Cluster Suite** will switch the service to the Secondary node.

Note **Lustre** verifies that the device generating the I/O error is being used by Lustre. If it is not then no corrective action will be taken.

Local Disk errors

When a local disk fails **HA Cluster Suite** will switch to the Secondary Node if this disk supports a system file system, e.g. /root.

2.6.3 Ethernet Network Failures

A failure of the heartbeat network will stop heartbeat exchanges leading each node to initiate to service take over. A fencing race starts between both nodes.

To avoid this fencing race, it is strongly advised to use the **Heuristic** function when deploying the High Availability configuration with the **stordepha** command (option **-H**). This function enables a node to check for problems on its side on the Ethernet network, using a periodic ping to a hardware component outside of the High Availability node pairs (Administration **IP** address of the **Ethernet** switch linked to the node by the heartbeat network).

Note Only the heartbeat network is monitored by **HA Cluster Suite**. A failure on another Ethernet network other than the one used for heartbeat will not lead to service takeover; however the failure will be displayed on the Management Node via **Bull System Manager - HPC Edition**.

Ethernet Network Access Failure (NIC or link Failure)

The management network is also used to send fence requests.
The node which is unable to use the management network cannot fence its peer node. Thus, the peer node wins the fencing race, and takes over the cluster services.

There is no risk of split brain (i.e. both nodes of the I/O cell running simultaneously the same Lustre service).

Management Network Failure

If the management network is unavailable for both nodes of the HA I/O cell, none will be able to fence its peer node. **HA Cluster Suite** does not initiate any failover action.
If one the node of the HA pair is able to fence the peer node, it wins the fencing race and takes over the services.

Chapter 3. Lustre and Management Node HA - Concepts

Lustre File System HA (High Availability) together with Management Node HA 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 section describes the **Lustre** features which have to be configured so that Management Node High Availability is supported.

See The *High Availability Guide* for more information on the Installation and Configuration of Management Node High Availability

3.1 Lustre and the Management Node

In a standard configuration three specific **Lustre** components run on the Management Node:

Lustre MGS service

This component is used by **Lustre** to store the global Lustre configuration and includes information such as the location of the different Lustre server nodes. Lustre clients refer to it for all requests. Lustre I/O nodes call it to enable their connections. It is not a problem if **MGS** is missing, as long as a Lustre component is not re/started. All running Lustre components include a copy of **MGS** in their cache. **MGS** has to be available, so that it can be updated, when a new Lustre component is started.

Lustredbd (Lustre DataBase daemon) service

The **lustredbd** service keeps the coherence between the **ClusterDB** and **ldap** up to date This service is required when there is a HA failover at start time for a service, so that the service can be restarted remotely. The **lustre_migrate hastart** command refers to **lustredbd** when there is a software, or hardware failover, on any Lustre I/O High Availability node.

Ldap daemon

This has the same use as **lustredbd**. It is referred to by all Lustre I/O nodes when there is a status request. Lustre I/O nodes have a local copy of the **ldap** content, and are capable of responding to a status request without contacting the **ldap** server. **Ldap** is mandatory when a Lustre service starts.

3.2 Lustre Services Managed by the Management Node

High Availability is provided by the High Availability feature, some data is shared and some services are managed by the High Availability software, as shown in Figure 3-1.

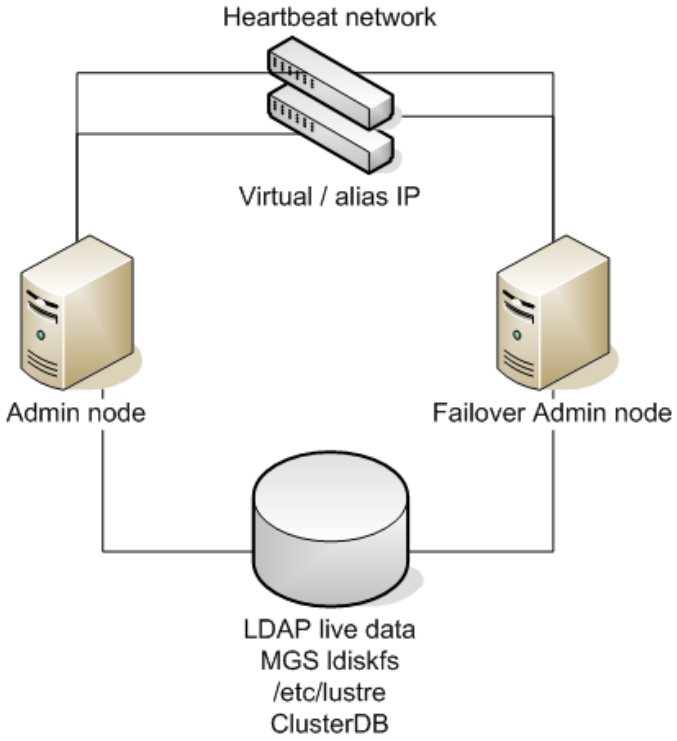

Figure 3-1. Management of Lustre components for systems with Management Node High Availability

Shared data

To support High Availability on the Management Node, the location of all the files and folders required when Lustre is running should be looked at carefully. All dynamic data has to be shared by both the primary and secondary Management Nodes using a shared storage device. HA Cluster Suite's management tools will mount the shared data on the active Management Node only for an active/passive Highly Available Management Node.
The dynamic **Lustre** data is stored in the **/etc/lustre** folder on the Management Node.

WARNING
To avoid the shared data being mounted twice the High Availability software should manage the mounting of the shared storage device.

Required services

The **Lustre** services that are managed by the HA Management Node software should be listed in the HA scripts, for example **lustre_failover** which lists the **start**, **stop** and **status** targets. These services include **MGS**, **LDAP** and **lustredbd**. Lustre also needs access to the ClusterDB, through **lustredbd** for failover cases or directly through the use of the **lustre_util** tool.

3.3 MGS Service Configuration Flow Chart

See the diagram, below, for details of how the **MGS** Service should be configured on Highly Available Management Nodes.

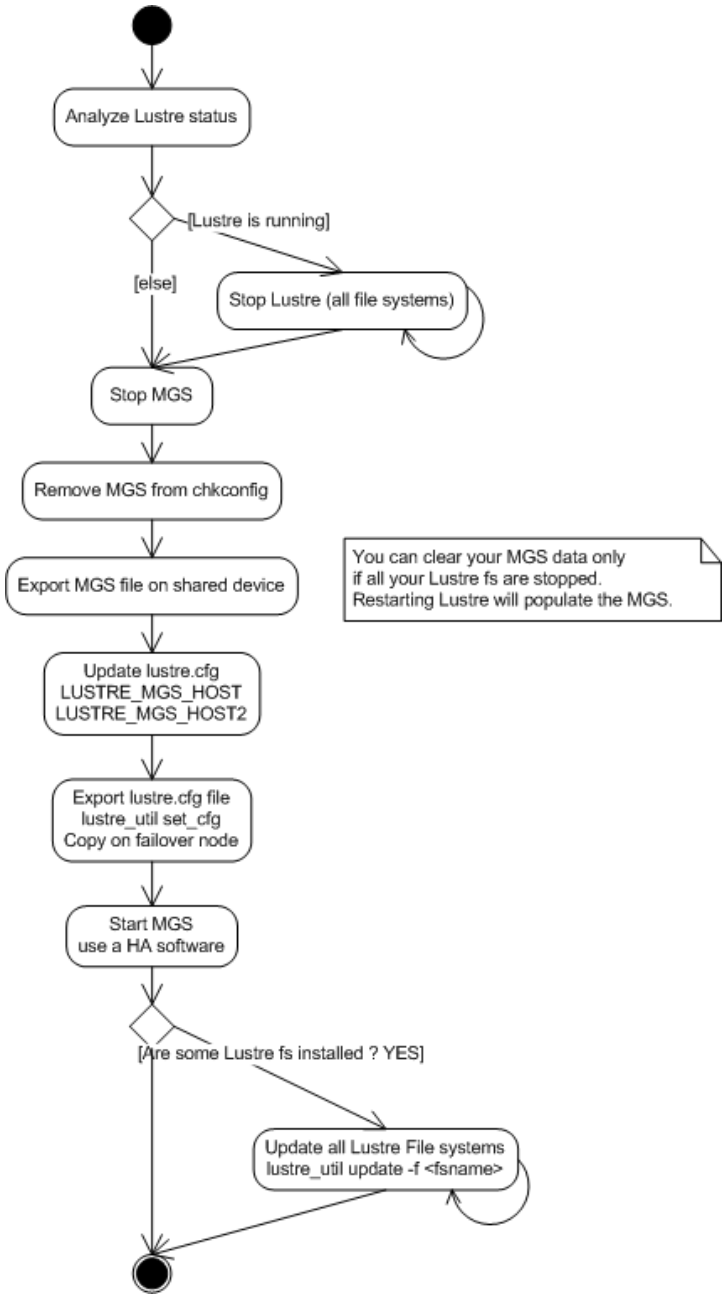

Figure 3-2. MGS Service configuration on Highly Available Management Nodes

WARNING

The MGS service does not support the use of IP address aliases. Always use two different IP addresses or hostnames in the lustre.cfg file.

Chapter 4. Configuring the Lustre File System

This chapter describes how to configure Lustre. It includes the configurations necessary for Lustre High Availability systems and, where necessary, references are included to sections in *Chapter 5* for **Lustre HA** specific configurations.

Important

These tasks must be performed after installation of Lustre software and after the deployment of the I/O Nodes as described in the *Installation and Configuration Guide* related to your delivery.

Unless specified, all the operations described in this chapter must be performed on the cluster Management Node, from the root account.

See *Appendix A. Configuration Files and Reference* for the syntax of the Lustre files and commands used in this chapter.

4.1 Enabling Lustre Management Services on the Management Node

1. Restore the Lustre system configuration information if performing a software migration:
   - **/etc/lustre** directory,
   - **/var/lib/ldap/lustre** directory if Lustre High Availability is included
2. Verify that the I/O and metadata nodes information is correctly initialized in the ClusterDB by running the command below:

```
lustre_io_node_dba list
```

This will give output similar to that below, displaying the information specific to the I/O and metadata nodes. There must be one line per I/O or metadata node connected to the cluster.

```
IO nodes caracteristics
id name type netid clus_id HA_node net_stat stor_stat lustre_stat
4  ns6 --I-- 6     -1      ns7     100.0    100       OK
5  ns7 --IM- 7     -1      ns6     100.0    100       OK
```

The most important things to check are that:

- ALL the I/O nodes are listed with the right type: I for OSS and/or M for MDS.
- The High Availability node is the right one.

It is not a problem if **net_stat, stor_stat, lustre_stat** are not set. However, these should be set when the file systems are started for the first time.

If there are errors, the ClusterDB information can be updated using the command:

```
lustre_io_node_dba set
```

3. Check that the file **/etc/cron.d/lustre_check.cron** exists on the Management Node and that it contains lines similar to the ones below:

```
# lustre_check is called every 15 mn
*/15 * * * * root /usr/sbin/lustre_check >> /var/log/lustre_check.log 2>&1
```

4.2 Configuring I/O Resources for Lustre

Skip this phase when carrying out an update, or a reinstall of the distribution as the Lustre configuration and data files will have been saved.

At this point, the storage resources should have already been configured, either automatically or manually, and in accordance with the type of storage system.

See *Configuring I/O Resources for the cluster* - in the *Installation and Configuration Guide* for configuration details (manual and automatic) for each type of storage system.

4.2.1 Configuring I/O Resources for Lustre after Automatic Deployment of I/O Configuration

This phase must take place after executing the procedures described in the sections *Automatic Configuration of a Storage System* and *Automatic Deployment of the configuration of I/O resources for the nodes* in the *Installation and Configuration Guide*.

The automated configuration of **Lustre** I/O resources uses the storage model file described in *Automatic Deployment of the I/O Configuration* section in the *Installation and Configuration Guide*. This model file details how Lustre uses the configured **LUN**s (description of **OST** and **MDT** data and journal LUNs).

When carrying out an update to, or a reinstall of, the distribution, do not run the following two stormodelctl commands, as the Lustre configuration and data files will have been saved.

The **Lustre** tables in the Cluster database should be populated with the information found in the model file, as described in this section.

1. Declare the Lustre OST configuration:

```
stormodelctl -m <model_name> -c generateost
```

2. Declare the Lustre MDT configuration:

```
stormodelctl -m <model_name> -c generatemdt
```

3. Make the OSTs and MDTs available for the Lustre File System:

```
lustre_investigate check
```

4.2.2 Configuring I/O Resources for Lustre after Manual I/O Configurations

Skip this phase for a migration to, or a reinstall of the distribution, as the /etc/lustre directory will have been saved.

This phase must take place after executing the procedures described in the *Manual Configuration of I/O Resources* in the *Installation and Configuration Guide*.
It should be done if no management tools are provided for the storage devices being used.

1. The **storage.conf file** allows you to provide required information to populate the **lustre_ost** and **lustre_mdt** tables. To fill-in this file, refer to the **storage.conf file** syntax, on page 87.

2. Update the Lustre tables in the ClusterDB using the **/usr/lib/lustre/load_storage.sh** script..
   See 6.1.3 *Loading storage.conf information into the ClusterDB* for details on using **load_storage.sh** script.

4.3 Configuring the High Availability Services (Lustre High Availability clusters only)

Lustre HA Carry out the actions indicated in the sections:
5.1 *Checking the Cluster Environment* and
5.2 *Using HA Cluster Suite*.

4.4 Running Lustre Pre-Configuration Operations

1. **Change the Lustre user password.**

   The **lustre_mgmt** rpm creates the **lustre** user on the Management node with « lustre » as the password. It is strongly advised to change this password by running the following from the root command line on both Primary and Secondary Management nodes for High Availability systems.

```
passwd lustre
```

   The **lustre** user is allowed to carry out most common operations on Lustre File Systems by using **sudo**. In the next part of this document, the commands can also be run as **lustre** user using the **sudo <command>**. For example:

```
sudo lustre_util status
```

2. **Set Lustre Network layers.**

   Lustre runs on all network layers that can be activated in the kernel, for example **InfiniBand** or **Ethernet**.

Important

By default the delivered **Lustre** model file is set to the **elan** net type. The **net type** parameter in the **/etc/lustre/models/fs1.lmf** file must be changed to **o2ib** for **InfiniBand** networks, and **tcp** for **Ethernet** networks.

If **Ethernet** is used as the **Lustre** network layer, and there are several physical links, you must select the links to be used by **Lustre**. This is done by editing the **/etc/modprobe.d/lustre** file.

See The *Lustre Operations Manual* (Section *Multihomed Servers*, sub-section *modprobe.conf*) available from http://manual.lustre.org/, for more details.

3. **Set the /etc/lustre/lustre.cfg file.**

   a. Edit the **/etc/lustre/lustre.cfg file** of the Management Node.

   b. Set **LUSTRE_MODE** to **XML**. (This should already have been done).

   c. Set **CLUSTERDB** to **yes** (if not already done).

Lustre HA Carry out the actions indicated in the sections:
5.3.1 *Installing the Lustre LDAP Directory* and
5.3.2 *Synchronizing ClusterDB*.

4.5 Configuring the Lustre MGS service

The Lustre **MGS service** is not managed by the **lustre_util command**. It is an independent service which has to be run separately. You can only have one MGS running per node.

Important

- **The Lustre MGS service must be installed and configured on the Management Node before Lustre is installed.**
- **Skip this phase for a migration to, or a reinstall of the distribution, as the /etc/lustre directory will have been saved.**

MGS is delivered as a service matching the Cluster Suite layout. The service is located in **/etc/init.d/mgs**.

1. **Configure /etc/lustre/lustre.cfg file**
   Some fields have to be filled in to link the MGS with the Lustre core.

Important

Before the LUSTRE_MGS_HOST and LUSTRE_MGS_NET fields are filled, check that the host node is valid by running the command gethostip -dn <host_name>. This will list the host name and its IP address. This is particularly recommended when there are multiple interfaces for a node.

   - **LUSTRE_MGS_HOST**=name of the Management Node where the MGS service is installed. This value is used by the **lustre_util** tool to link the **MGS** with other Lustre entities, for example, **MDS, OSS**.
   - **LUSTRE_MGS_NET**= the name of the network used to read the **MGS**, for example, TCP or o2ib. When the **o2ib** net type is used the **LUSTRE_MGS_HOST** name value has to be suffixed with'-ic0' which is hostname suffix for IB networks.
     For example, if you need to use an **InfiniBand** network to reach the MGS entity that runs on the node **zeus6** you have to:
     - set LUSTRE_MGS_NET to o2ib
     - set LUSTRE_MGS_HOST to zeus6-ic0
   - **LUSTRE_MGS_ABSOLUTE_LOOPBACK_FILENAME** = file for mgs loop device. The default is **/home/lustre/run/mgs_loop**.
     When High Availability exists for the Management Node, select a directory which is shared for the Management Node pairs. This value is used by the **MGS** service when **lustre_util** is not used.

2. **Verify your network.**
   Lustre requires IP interfaces to be in place. On your **Lustre MGS** node, make sure that **IPOIB** is configured if the **InfiniBand** modules are available.

3. **Install the MGS service on the Management Node:**

```
service mgs install
```

```
mgs installed [OK]
```

Run the command below to ensure that the **MGS** service restarts:

```
chkconfig --add mgs
```

4. **Start the MGS service on the Management Node:**

```
service mgs start
```

```
Starting mgs: on xena0
mgs xena0 is not running
mgs started [OK]
```

When there is no High Availability on the Management Node, the service must be started at boot time.

4.6 Checking Lustre Pre-Configuration

1. Once the **lustre.cfg file** has been edited and saved, copy it to the Secondary Management Node for clusters which feature High Availability for the Management Node.

2. Check that **mgs service** is running on the Management Node:

```
service mgs status
```

```
/dev/loop0 on /mnt/srv_lustre/MGS type lustre (rw)
mgs xena0 is running
```

3. Check the consistency of the database.

```
lustre_investigate check
```

This command checks which storage devices in the **lustre_ost** and **lustre_mdt** tables can be used. A clean output means that the command has been successful.

Run the command below to list the **OST**s. There must have be at least one OST with **cfg_stat** set to "**available**":

```
lustre_ost_dba list
```

Run the command below to list the **MDT**s. There must have be at least one MDT with **cfg_stat** set to "**available**":

```
lustre_mdt_dba list
```

Lustre HA Carry out the actions indicated in the section:
5.3.3 Managing Lustre Failover Services on the I/O and Metadata Nodes.

4.7 Configuring Lustre

1. **Configure Lustre on the I/O nodes.**

   Run the following command, and answer 'yes':

```
lustre_util set_cfg
```

   An output similar to the following is displayed:

```
lustre.cfg copied on < I/O nodes >
snmpd enabled on < I/O nodes >
ldap database enabled on < mgmt node >
```

2. **Create the file system configuration.**

   The **/etc/lustre/models/fs1.lmf** file is a default model file which comes with the Lustre RPMs. It implements a file system which uses all the available OSTs and the first available MDT, with no failover. If you want to create more than one file system and/or with failover capability, refer to *Chapter 6* for more details about the Lustre model files.

   Run the following command:

```
lustre_util info -f /etc/lustre/models/fs1.lmf
```

   This command prints information about the **fs1** file system. It allows you to check that the MDT and OSTs are actually those you want to use. Ensure that no warning occurs.

Lustre HA Carry out the actions indicated in the section: *5.3.4 Configuring File Systems for Failover*.

3. **Check what happened.**

   At this point it is possible to run the following command on a second terminal (checking terminal) to see what happened during the installation process.

```
watch lustre_util info -f all
```

   The following message should be displayed:

```
No filesystem installed
```

   It is also possible to look at http://<mngt_node>/lustre from a Web browser.

4. **Install the file system.**

Important

Do not perform this step when performing a software migration as the Lustre configuration details and data will have been preserved.

Run the following command:

```
lustre_util install -f /etc/lustre/models/fs1.lmf -V
```

This operation is quite long as it formats the underlying file system (about 15 minutes for a 1 TB file system). Do not use the **-V** option if a less verbose output is required.

At the top of the checking terminal, the following should appear:

```
   Filesystem fs1:
      Cfg status  : formating
      Status      : offline
      Mounted     : 0 times
```

Wait until the following appears:

```
Filesystem fs1:
      Cfg status  : installed
      Status      : offline
      Mounted     : 0 times
```

The last line printed at the execution terminal must be:

```
  Filesystem fs1 SUCCESSFULLY installed
```

5. **Enable the file system**

Run the following command:

```
lustre_util start -f fs1 -V
```

This operation is quite long (about 10 minutes for a 1TB file system). Do not use the **-V** option if a less verbose output is required.

At the top of the checking terminal, the following should appear:

```
Filesystem fs1:
      Cfg status  : installed
      Status      : starting
      Mounted     : 0 times
```

Wait until the following appears:

```
   Filesystem fs1:
      Cfg status  : installed
      Status      : online
      Mounted     : 0 times
```

The "running status" of the OSTs/MDT must also be 'online'.

The last lines printed at the execution terminal must be:

```
   FILESYSTEMS STATUS
   +-----------+----------+--------+--------+----------------+
   |filesystem |  config  |running | number |   migration    |
   |           |  status  | status |of clts |                |
   +-----------+----------+--------+--------+----------------+
   |fs1        |installed |online  |0       |0 OSTs migrated |
   +-----------+----------+--------+--------+----------------+
```

6. **Mount the file system on clients.**

   Run the following command:

```
lustre_util mount -f fs1 -n <list_of_client_nodes_using_pdsh_syntax>
```

   For example, if the client nodes are `ns0` and `ns2`, then run:

```
lustre_util mount -f fs1 -n ns[0,2]
```

   At the top of the checking terminal, the following should appear:

```
Filesystem fs1:
   Cfg status  : installed
   Status      : online
   Mounted     : 2 times
```

   The last line printed at the execution terminal must be:

```
Mounting filesystem fs1 succeeds on ns[0,2]
```

The file system is now available. As administrator it will be possible to create user directories and to set access rights.

It is possible to check the health of the file system, at any time, by running:

```
lustre_util status
```

This will display a status, as below:

```
FILESYSTEMS STATUS
+-----------+----------+--------+--------+----------------+
|filesystem |  config  |running | number |   migration    |
|           |  status  | status |of clts |                |
+-----------+----------+--------+--------+----------------+
|fs1        |installed |online  |2       |0 OSTs migrated |
+-----------+----------+--------+--------+----------------+
  CLIENTS STATUS
+-----------+----------+
|filesystem |correctly |
|           | mounted  |
+-----------+----------+
|fs1        |ns[0,2]   |
+-----------+----------+
```

If more details are required, then run:

```
lustre_util all_info -f all
```

The file system health can also be checked in the **Nagios** view of the Management Node.

Chapter 5. Configuring HA for Lustre

This chapter complements *Chapter 4,* for clusters which implement Lustre High Availability.

Important

Lustre File System 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.

5.1 Checking the Cluster Environment

Important

- **These checks are mandatory. A cluster may start but will generate errors at runtime if they are not all verified.**
- **Lustre High Availability is only supported on clusters which include a dedicated Management Node with the Cluster Database in place.**

5.1.1 Checking the High Availability Paired Nodes

Run the command below to check the status of the HA paired nodes:

```
lustre_migrate nodestat -n all
lustre_migrate nodestat -n <nodes list>
```

This will give output similar to that below:

```
HA paired nodes status
----------------------
node name    node status       HA node name    HA node status
node1        MIGRATED          node2               OK
node10            OK               node12          MIGRATED
```

If the nodes are not correctly paired then edit the ClusterDB using the command below:

```
lustre_io_node_dba set  --nid <node_id>
                        [--fid <ha_node_id>] [--cid <cluster_id>]
                        [--net <net_status>] [--stor <storage_status>]
                        [--stat <lustre_status>] [--db <db_name>]
```

Important

If you are migrating to a new system release the existing HA paired nodes will be used.

5.1.2 Checking the connection between the I/O nodes and the Management Node

The I/O nodes must be connected to the Management Node and have the right to restart some of the services on the Management Node, as and when needed.

Run the command below to check the connection between the I/O nodes and the Management Node (node0 in the example, below)

```
pdsh -w <all IO nodes> "ssh <management node> echo OK"
```

Example

```
pdsh -w node[1-2,10,12] "ssh node0 echo OK"
```

Output (example):

```
node10: OK
node2:  OK
node1:  OK
node12: OK
```

5.1.3 Verifying the connection between the I/O nodes and the shared storage device

The same storage links must be available for both High Availability paired I/O nodes. Run the command, below, from the Management Nodes on all the I/O paired nodes to test that the links are OK:

```
pdsh -w <node1,node2> "stormap -l | sort" | dshbak -c
```

If any links are missing, then run the command, below, on the I/O node concerned to repair its links:

```
stormap -c
```

If the problem persists, check that the lpfc driver is correctly loaded.

5.2 Using HA Cluster Suite

Large clusters may contain multiple I/O cells, and within each I/O cell, the HA Cluster Suite must be configured. This process is fully automated by Bull cluster management tools. All the necessary information is extracted from the ClusterDB and used to generate the HA Cluster Suite configuration files. These files are pushed to each node which must be controlled by the HA Cluster Suite.

HA Cluster Suite commands not described in the present paragraph must not be used, as they may lead to fatal inconsistency for the Lustre file system. The GUI must not be used as well. All the HA Cluster Suite setup is predefined to enable the failover process expected by the Lustre file system, and prevent any risk of split brain (i.e. both nodes of the I/O cell running simultaneously the same Lustre service). Administrator must not attempt to modify the HA Cluster Suite's configuration within I/O cells.

The management tasks for **HA Cluster Suite** are:

- Distributing the configuration file
- Starting the HA Cluster Suite

By default, there is not automatic start at boot time, and it is not recommended to enable this.

5.2.1 Distributing the cluster.conf file on the I/O Node

The **/etc/cluster/cluster.conf** is generated using node HA pair defined in the ClusterDB. The following options are selected, and must not be changed:

- Name of the services to be managed.
- Manual start of services (to avoid split brain if a node can not join its peer node).
- List of nodes.
- Heartbeat through the Management Network for **NovaScale R440** and **R460** machines.

If the Heuristic function is to be used in the HA Cluster Suite configuration, **stordepha** must be used with the **–H** option, as below:

```
stordepha -a -c configure -o lustre [-H]
```

The **–a** flag refers to all nodes specified with the **-o** option. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the **stordepha** man page for more information.

Restoring a node

The **cluster.conf** file is not preserved when a node is reinstalled by **KSIS.** It must also not be integrated in a node image. After each node has been deployed, the **stordepha** command must be used to reset the node's configuration.

5.2.2 Starting / Stopping HA Cluster Suite's Daemons

HA Cluster Suite's daemons can be configured from the Management Node on all or on a subset of the High Availability I/O nodes, or configured locally on each node. In both cases, all the daemons required must be started and stopped in the right order.
Starting **HA Cluster Suite** will start the HA Cluster Suite daemons but not the HA Cluster Suite services as automatic start is disabled. Stopping **HA Cluster Suite** on a node causes its services to fail on the active node, if there is one.

- Run the command below from the Management Node:

```
stordepha -c start|stop -a -o lustre
```

  The **–a** flag means all nodes. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the **stordepha** man page for more information.
  OR

- Run the command below on the node locally:

```
storioha -c start|stop
```

5.2.3 Checking the Status of HA Cluster Suite

HA Cluster Suite's status can be verified from the Management Node on all or on a subset of the High Availability I/O nodes, or locally on each node.

- Run the command below from the Management Node:

```
stordepha -c status -a
```

  The **–a** flag means all nodes. It is possible to exclude some nodes (**-e** flag) or to specify a list of target nodes (**–i** flag, exclusive with **–a**). See the **stordepha** man page for more information.

- Or:
  Run the command on a node locally:

```
storioha -c status
```

Alternatively, it is also possible to use the HA Cluster Suite's **clustat** command:

```
clustat
```

Or:

```
clustat -i <refresh period>
```

5.3 Configuring Lustre High Availability

5.3.1 Installing the Lustre LDAP Directory

1. Create the **/var/lib/ldap/lustre** directory:

```
mkdir -p /var/lib/ldap/lustre
chown ldap.ldap /var/lib/ldap/lustre
```

2. Enable and start the **LDAP** service:

```
chkconfig --level 3 ldap
chkconfig --level 5 ldap
service ldap start
```

3. For a first installation initialize the **LDAP** directory, using the command below:

```
ldapadd -D cn=Manager,fs=lustre -w secret -x -H ldap://<hostname>/
-f /usr/share/lustre/top.ldif
```

4. Go to the **/etc/sudoers** configuration file and verify that the **ldap** user has access to the **lustre_<table>_dba** commands:

   The following should appear:

```
dap ALL=(root)NOPASSWD:/usr/sbin/lustre_ost_dba *
ldap ALL=(root)NOPASSWD:/usr/sbin/lustre_mdt_dba *
ldap ALL=(root)NOPASSWD:/usr/sbin/lustre_io_node_dba *
ldap ALL=(root)NOPASSWD:/usr/sbin/lustre_ldap *
ldap ALL=(root)NOPASSWD:/usr/sbin/lustre_util tune_servers ,
/usr/sbin/lustre_util tune_servers  *
```

   If not, use the **visudo** tool to update the **/etc/sudoers** file.

5. Edit the **/etc/lustre/lustre.cfg** management configuration file.
   a. Set the **LUSTRE_LDAP_URL** with the name of the Management Node for the failover file systems e.g. **ldap://<Mgmt_Node_name>/**.
   b. Change the **ldap URL** in the **/etc/lustre/lustre.cfg** file and provide an alias IP address for both Management Nodes.

5.3.2 Synchronizing ClusterDB

1. Set the **/etc/lustre/lustre.cfg** parameters, shown below, to allow the replication of **LDAP** in the ClusterDB:

   LUSTRE_DB_DAEMON_HOST=<hostname>

   LUSTRE_DB_DAEMON_PORT=<tcp port>

2. Set **LUSTRE_DEBUG** to "yes" to enable the failover tools trace feature.
   On each I/O node, the Lustre failover scripts will log events in the **/var/log/lustre** directory. On the Management Node, the **lustre_ldap** daemon will log events in the **/tmp/log/lustre** directory.

3. Start **lustredbd**, the **LDAP** replication daemon:

```
service lustredbd.sh start
```

```
Starting lustredbd:
.....
lustredbd started [OK]
```

4. Check the status for the **LDAP** replication daemon:

```
service lustredbd.sh status
```

```
lustredbd is running
```

Note A daily log file will be created daily in the **/var/log/lustre** directory on the Management Node.

5. If High Availability for the Management Node is supported, you must also:
   a. Stop the **lustredbd** service, if it is running on the Management Node:

```
service lustredbd.sh stop
```

   The **lustredbd** data is contained in the ClusterDB and does not need to be moved.
   b. Change the **lustredbd** URL in the **/etc/lustre/lustre.cfg** file and provide an alias IP address for both Management Nodes.
   c. Restart the **lustredbd** service using the High Availability software:

```
service lustredbd.sh start
```

5.3.3 Managing Lustre Failover Services on the I/O and Metadata Nodes

Lustre failover services are used by **HA Cluster Suite** to control the migration of the Lustre OST/MDT services.

WARNING
The failover services have to be started before the Lustre file systems are started. They can be stopped only when all Lustre file systems have stopped running.

The **lustre_migrate** command manages the **Lustre** failover services on the cluster.

If no file system parameters are specified, the command will act on all the I/O and Metadata nodes.

Start/Stop/Status failover services

1. Use the command below to display the status of the **Lustre** failover services on the I/O and Metadata Nodes specified. If no file system parameters are specified, the command will act on all the I/O and Metadata nodes.

```
lustre_migrate hastat -n <node_list>
```

2. Use the command below to start the **lustre_failover** services on the nodes listed. The use of **node_list** is mandatory; **-n all,** which uses the nodes listed in the ClusterDB, is also supported.

```
lustre_migrate hastart -n <node_list>
```

   Before starting High Availability with the **-n all** option, try the option with the **lustre_migrate hastat** command to see which nodes are listed. The command will fail if one of the nodes is missing.

Example

node1 and **node2** are a pair of HA nodes. If for some reason **node1** is down, then there will be problems when **HA Cluster Suite** is launched for the first time, and if **node1** was up the last time **HA Cluster Suite** launched the **node1** service.

When you run the command, below, to start the **node1** service on **node1** or on the owner of the **node1** service, the command will fail.

```
lustre_migrate hastart -n node1
```

In this example when the command fails try running the command:

```
lustre_migrate hastart --force -n node1
```

This command will try to force **node1** to start. If it is unable to start **node1** it will try to start the **node1** service on its failover node.

3. Use the command below to stop the **Lustre** failover services on the I/O and metadata nodes listed. If no file system parameters are specified, the command will act on all the I/O and Metadata nodes.

```
lustre_migrate hastop -n <node_list>
```

5.3.4 Configuring File Systems for Failover

Configuring file systems for failover means configuring two HA paired **OSS/MDS** Servers to support either **OST** or **MDT** targets, one server will be the primary node, the other the secondary node.

The failover feature is declared in the **/etc/lustre/models/<file_system_name>.lmf** model file. In the file system model update the following parameter:

- **failover=yes** Enables the failover configuration to be generated.

A file system is usually described as being composed of one MDT and several OSTs taken from the ClusterDB.

The secondary node declared in the ClusterDB will be taken into account for the secondary OST/MDT declaration.

The file system is managed using the lustre_util utility in a standard way. The failover specifics (LDAP directory interaction, status display, alternative mount, etc.) are automatically supported by the Lustre management tools.

Chapter 6. Administrating Lustre File System

This chapter describes the following administrator's functions for Lustre File System.

- 6.1 *Lustre in the ClusterDB*
- 6.2 *Lustre Networks*
- 6.3 *Lustre Services Definition*
- 6.4 *Lustre File Systems Definition*
- 6.5 *Lustre File Systems Installation and Removal*
- 6.6 *LDAP Management*
- 6.7 *Lustre File Systems Reconfiguration*
- 6.8 *Migration Failover Services*
- 6.9 *Quotas in Lustre File Systems*

6.1 Lustre in the ClusterDB

6.1.1 Lustre Tables

The Lustre management tools rely on the ClusterDB to store and get information about:

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

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

ClusterDB Information for Lustre High Availability

Two kinds of information are included in the **Lustre** ClusterDB tables to allow failover management and the monitoring of **Lustre** services:

- Static information linked with the cabling schema for the paired nodes.
- Dynamic information about the migration of the nodes and the distribution of **Lustre** services.

The **lustre_io_node** table for each M(etadata) and I(/O) node lists:

- The identity of the paired nodes, pre-loaded when the cluster is installed.
- The current migration status, this is kept up to date by the failover management tools.

The **lustre_ost** and **lustre_mdt** tables for each Lustre service list:

- The primary and secondary nodes installed by the storage/Lustre deployment process
- The node that is active. This is set dynamically by the failover management tools.

Important

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

6.1.2 lustre_<table>_dba Commands

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

- **lustre_ost_dba**
- **lustre_mdt_dba**
- **lustre_fs_dba**
- **lustre_io_node_dba**

See **lustre_<table>_dba commands**, on page 101 for the syntax of these commands.

6.1.3 Loading storage.conf information into the ClusterDB

load_storage.sh is a script that is used to load **storage.conf file** information into the **lustre_ost** and **lustre_mdt** tables of the cluster database.
This may be useful if no management tools are provided for the storage devices, for example for **NEC** devices.

SYNOPSIS

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

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

Recommendation for High Availability MDS Node

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

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

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

6.2 Lustre Networks

By default **Lustre** runs on all network layers that may be active in the kernel, for example **InfiniBand** or **Ethernet**. If you do not want **Lustre** to run on certain network layers, these network layers must be deactivated for the nodes in question.

If **Ethernet** is used as the **Lustre** network layer, it is possible to select the link on which Lustre will run. This is done by editing the **/etc/modprobe.d/lustre** file. For details see the *Lustre Operations Manual* (Section *Multihomed Servers*, sub-section *modprobe.conf*) at http://manual.lustre.org/

6.3 Lustre Services Definition

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

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

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

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

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

See **lustre_investigate command**, on page 99 for the syntax of this command.

6.4 Lustre File Systems Definition

6.4.1 Prerequisites

- Lustre management tools use the **/etc/lustre/lustre.cfg file** to get configuration information. This file must reside on all OSS and MDS nodes. The **lustre_util command** enables to distribute this file easily.
  /etc/lustre/lustre.cfg is assumed to be updated correctly
- The **lustre_ost** and **lustre_mdt** tables are assumed to be updated correctly in the ClusterDB (use **lustre_investigate check** to verify).
- Lustre tools use **ssh** to execute remote commands, so users must be allowed to log into nodes without being prompted for a password. This can be done by appending the right keys in **/root/.ssh/authorized_keys2**.

Note For details on the configuration tasks refer to Chapter 4 and 5.

See **lustre.cfg file** and **lustre_util command** in *Appendix A. Configuration Files and Reference* for the file and command syntax.

6.4.2 Lustre Model File (.lmf) and Extended Model Files (.xmf)

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

See **Lustre Model file (.lmf)**, on page 85 for the syntax of this file.

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

Lustre Model Sample File

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

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

# To prevent failover
```

```
failover: no

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

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

# Network is elan
nettype: elan

# New mount options
ost_mount_options: extents,mballoc

######################################################
# First file system : fs1

# File system name is fs1
fs_name: fs1

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

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

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

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

######################################################
# Second file system : fs2

# File system name is fs2
fs_name: fs2

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

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

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

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

6.5 Lustre File Systems Installation and Removal

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

See **lustre_util command**, on page 88 for the syntax of this command.

6.5.1 Installing Lustre File Systems

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

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

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

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

6.5.2 Removing Lustre File Systems

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

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

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

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

6.6 LDAP Management

The **Lustre LDAP** directory on the Management Node contains the description of all **Lustre** file systems installed on the cluster's I/O nodes, including details of the distribution of their services.
When a file system starts, it is noted as active in the **LDAP** directory. The **lustre_util** utility uses failover scripts to manage the takeover of the services for a **Lustre** file system.

Lustre failover scripts check the **LDAP** directory each time a failover event occurs on a node, in order to obtain the list of **Lustre** services to act on. The scripts update the **LDAP** directory each time there is a migration. A synchronization mechanism transfers this information to the cluster database.

The **lustre_ldap** utility manages the LDAP directory, using:

- Callback for **lustre_util** and for the **LDAP** server to ensure the ClusterDB is synchronized.
- An online interface to display the **LDAP** directory contents.

Lustre file system HA status

The command below displays the High Availability status of the **Lustre** file system:

```
lustre_ldap show [-f <file_system_name>]
```

The **active** status is shown as either:

- **Yes** if the file system is managed by High Availability
- **No** if the file system is NOT managed by High Availability

If no file system parameters are specified, the command will show the status of all the file systems loaded in the **LDAP** directory.

Lustre file system description

The command below lists the **LDAP** descriptor of the file system in **LDIF** format:

```
lustre_ldap list [-f <file_system_name>]
```

If no file system parameters are specified, the command will list the descriptors for all the file systems loaded in the **LDAP** directory. The **LDIF** display format is mainly used for maintenance purposes.

Important

lustre_ldap can also be used to update the **LDAP** directory. This must be done very carefully otherwise the consistency of the failover information may be broken.

6.7 Lustre File Systems Reconfiguration

This procedure allows you to change the distribution of the Lustre services which are defined on the I/O nodes, without having to re-deploy (which involves configuring the DDN storage systems and High Availability). The file systems involved in the new distribution are stopped; the others continue to be operational.

The following example describes how to stop the `fs1` and `fs2` file systems.

1. If needed save the data of the `fs1` and `fs2` file systems.

2. Unmount the `fs1` and `fs2` file systems:

```
lustre_util umount -f fs1 -n all [-F]
lustre_util umount -f fs2 -n all [-F]
```

3. Stop the `fs1` and `fs2` file systems:

```
lustre_util stop -f fs1 [-F]
lustre_util stop -f fs2 [-F]
```

4. Remove the `fs1` and `fs2` file systems:

```
lustre_util remove -f fs1
lustre_util remove -f fs2
```

5. Make the required modifications in the models associated with the file systems. In our example `fs1` and `fs2` are grouped together in only one `fs3` file system.

6. Configure the new `fs3` file system (this operation erases the `fs1` and `fs2` file systems data).

```
lustre_util install -f /etc/lustre/model/fs3.lmf
```

7. Start the new `fs3` file system:

```
lustre_util start -f fs3
```

8. Mount the new `fs3` file system:

```
lustre_util mount -f fs3 -p p2
```

9. If needed, restore the saved data.

6.8 Migration Failover Services

The command, below, will migrate the **Lustre** services for a node to its HA paired node, so that it can be stopped without disturbing the **Lustre** file system. This is used for maintenance purposes.

```
lustre_migrate export -n <node_name>
```

Specifically, this command will stop the **lustre_<node_name>** failover service (**<node_name>** node is the primary node), and restart it on the secondary node. The secondary node information is taken from the ClusterDB.

Note If the **lustre_<node_name>** failover service was already running on its secondary node, the command has no effect.

Use the command below to relocate a failover service to its primary node, once it has been repaired:

```
lustre_migrate relocate -n <node_name>
```

This command will stop the **lustre_<node_name>** failover service (**<node_name>** node is the primary node) on the secondary node, and restart it on the primary node. The primary node information is taken from the ClusterDB.

Note If the **lustre_<node_name>** failover service was already running on its primary node, the command has no effect.

6.9 Quotas in Lustre File Systems

By default no quota is set in the Lustre configuration. This section provides information for Lustre administrator who needs to use quotas.

6.9.1 Quota Settings in Model Files

Quotas are enabled by setting "quota" to "yes" in lmf file:

```
quota: yes
```

The default quota options are as follows:

```
quota_options: quotaon=ug,iunit=5000,bunit=100,itune=50,btune=50
```

quotaon=<u|g|ug> Enable quota for user|group|user and group.

iunit=<number of inodes> **iunit** is the granularity of inodes quotas. Inodes are acquired and released by a slice of inuit. iunit is a int type (>0), the default value in Lustre is 5000 inodes.

bunit=<size in MB> **bunit** is the granularity of block quotas. Blocks are acquired and released by a slice of bunit MBs on each OSTs. bunit is expressed in MBs (>0), the default value in Lustre is 100 MBs.

itune=<percentage> **itune** sets the threshold to release and acquire iunit inodes. For example, if a user/group owns n*iunit+m inodes, iunit inodes will be acquired for this user as soon as m goes above itune*iunit/100. If a user/group owns n*iunit-m inodes, iunit inodes will be released for this user/group as soon as m goes above itune*iunit/100. itune is a int type (100 > itune > 0), the default value in Lustre is 50.

btune=<percentage> **btune** sets the threshold to release and acquire bunit block MBs for each OST. For instance, if a user/group owns n*bunit+m MB on one OST, bunit MBs will be acquired on this OST for this user/group as soon as m goes above btune*bunit/100. If a user/group owns n*bunit-m MBs on one OST, bunit MBs will be released on this OST for this user/group as soon as m goes above btune*bunit/100 MB. btune is a int type (100 > btune > 0), the default value in Lustre is 50.

6.9.2 Starting Quota: lfs quotacheck

Once the file system is installed, started and mounted, run the following command on a client:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

This means that if quota_options are as follows:
`quotaon=ug,iunit=5000,bunit=100,itune=50,btune=50`
and mountpoint is `/mnt/lustre`, then it will be necessary to run:

```
lfs quotacheck -ug /mnt/lustre
```

The time taken by **quotacheck** depends on the size of the biggest device used by the file system as OST or MDT. On average, it takes 160s for a 1TB OST/MDT check.

6.9.3 Setting the Limits: lfs Setquota

lfs setquota sets limits on blocks and files.

```
lfs setquota [-u|-g] <name> <block-softlimit> <block-hardlimit>
<inode-softlimit> <inode-hardlimit> <mount_point>
```

- **block-softlimit** and **block-hardlimit** are expressed in kB.
- **inode-softlimit** and **inode-hardlimit** are expressed in number of inodes.
- Limits on blocks/inodes MUST be greater than bunit/iunit. This means, for example, bunit=100MB, block-softlimit and block-hardlimit must be greater than 102400kB. If you have iunit=5000, inode-softlimit and inode-hardlimit must be greater than 5000.
- Limits on blocks must be greater than the number of OST * bunit. This means, for example, if there are 9 OSTs and bunit=100 MBs, block-softlimit and block-hardlimit must be greater than 9*100*1024=921600 kBs.

For example, the following command will set a **block-softlimit** to **900MB**, **block-hardlimit** to **1GB**, **inode-softlimit** to **5000**, **inode-hardlimit** to **10000** for user **testfs**, for a lustre file system mounted on /mnt/luster:

```
lfs setquota -u bob 900000 1000000 5000 10000 /mnt/lustre
```

The following command will implement the same settings for all users of group **dba**:

```
lfs setquota -g dba 900000 1000000 5000 10000 /mnt/lustre
```

Restrictions

- At present, soft limits are not supported in Lustre. So set **block-softlimit** and **inode-softlimit** to 0.
- It is strongly recommended to run **setquota** on a Lustre file system which is not busy. Otherwise an incorrect **block- hardlimit** value may be set.

6.9.4 Updating/Rescuing a File system with Quota enabled

If a file system is rescued, quota will have to be enabled again using the command below.

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

If a file system is updated and new OSTs are not added the following command will have to be run again:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

If a file system is updated and **new OSTs are added** then the **fs** will have to be updated, started and mounted and then the following command run:

```
lfs quotacheck -<quotaon parameter> <mount_point>
```

For ***ALL*** groups and users, all the limits may be set to 0 with the following command:

```
lfs setquota -u <user> 0 0 0 0 <mount_point>
lfs setquota -g <group> 0 0 0 0 <mount_point>
```

For ***ALL*** groups and users, the limits may be set to their former values with the following command.

```
lfs setquota [-u|-g] <name> <block-softlimit> <block-hardlimit>
<inode-softlimit> <inode-hardlimit> <mount_point>
```

Chapter 7. Monitoring Lustre

This chapter explains how to monitor get information about the Lustre system, using **Bull System Manager – HPC Edition**, a graphical user interface for performance and health monitoring.

7.1 Monitoring Lustre System

Status information about the Lustre file system and I/O nodes is kept up to date in the ClusterDB by the Lustre management tools.

Using this information and that collected by performance daemons, the **Bull System Manager - HPC Edition** supervision tool offers items specific to the Lustre system allowing the health and performance to be monitored from the management station – see the chapter on monitoring for more details.

7.1.1 Lustre System Health Supervision

7.1.1.1 The all status Map view

This includes global status indicators which provide the administrator with information about the global I/O system availability.

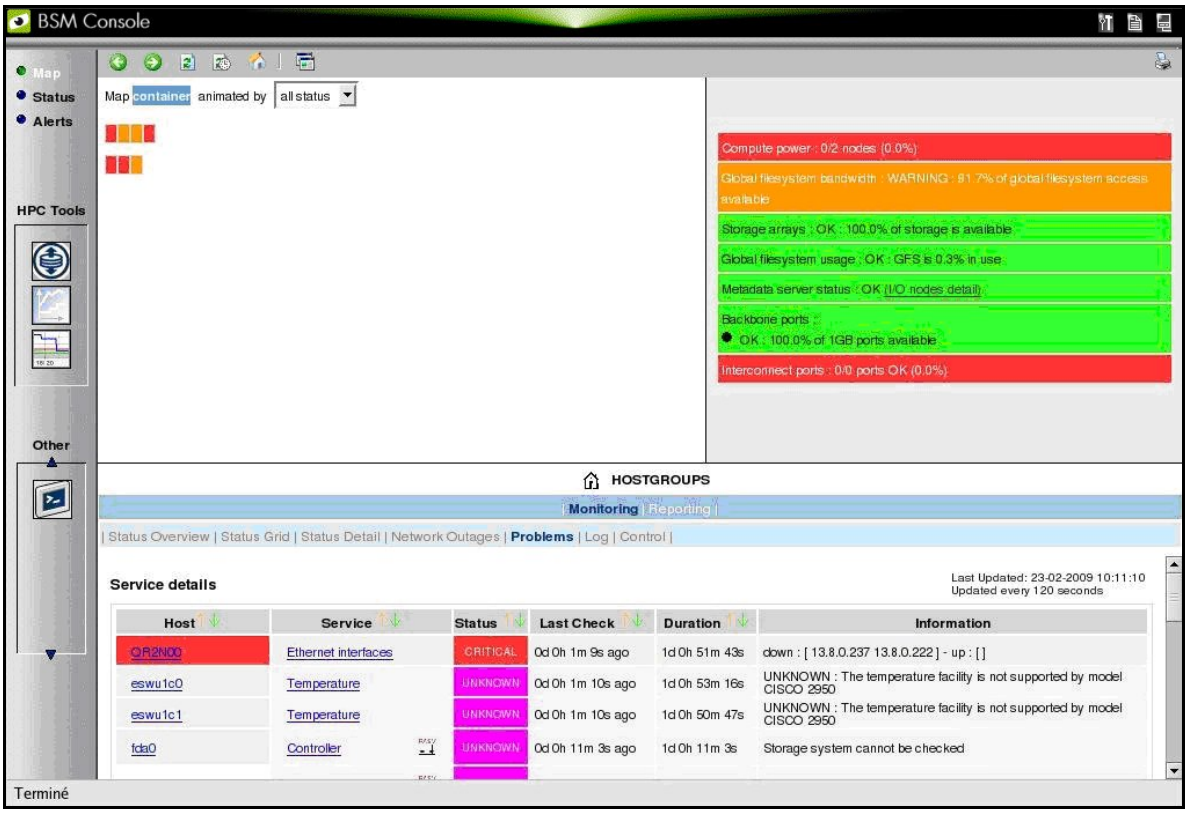

Figure 7-1. Bull System Manager - Map view

System Availability Indicators are located at the right top of the topological view and provide a status to the Administrator at a glance. These include:

Available Global File System Bandwidth as a Percentage

This is indicated as a percentage of I/O nodes available. An I/O node is fully available if it has its three Quadrics rails and its height fibre links up and if its Lustre status is OK. If not, a degradation factor is applied as follows:

- **cancel** the node if Lustre is not OK
- **apply** a 30% factor of degradation per quadrics rail missing
- **apply** a 12% factor of degradation per fibre link missing

Available Storage Arrays as a Percentage

The ratio of running storage appliances against the total number is indicated.

Global File System Usage

This gives the current usage rate of the **Lustre** system for all the Lustre file systems together.

MDS Migration Alert

If **High Availability** is configured, this alerts the administrator to a **MDS** failover migration. The Lustre system then no longer has the High-Availability status.

7.1.1.2 File systems Health Monitoring

This is done by the script **/usr/bin/lustre_fs_nagios**. It checks the state of each OSTs/MDTs, and sets the status of the file systems into the ClusterDB according to whether they are online or offline. This script is called every 15 min on the Management Node using **/etc/cron.d/lustre_fs_nagios.cron**, which is automatically installed and enabled by lustre_utils RPM.

lustre_fs_nagios should not be used online by the administrator; however, it can be used to force a refresh of the **nagios** lustre file system status entry.

7.1.1.3 The lustre_check Tool

The **lustre_check** tool keeps the I/O node availability information up to date in the **Cluster Database**. It runs on the Management Node, scheduled by a **cron** every 15 min.

When called, it checks the I/O nodes and collects network and storage information. This information is stored for each node in the **lustre_io_node** table of the database, where it is scanned regularly by the supervision tools.

The **lustre_check** tool is unlikely to be used on-line by the Administrator; however, it can be used to force a refresh of the Cluster database information and to get a node by node status instantly.

7.1.2 Lustre File system Indicator

Within Bull System Manager the **Nagios** service plug-ins include a plug to monitor the health for the Lustre file system.

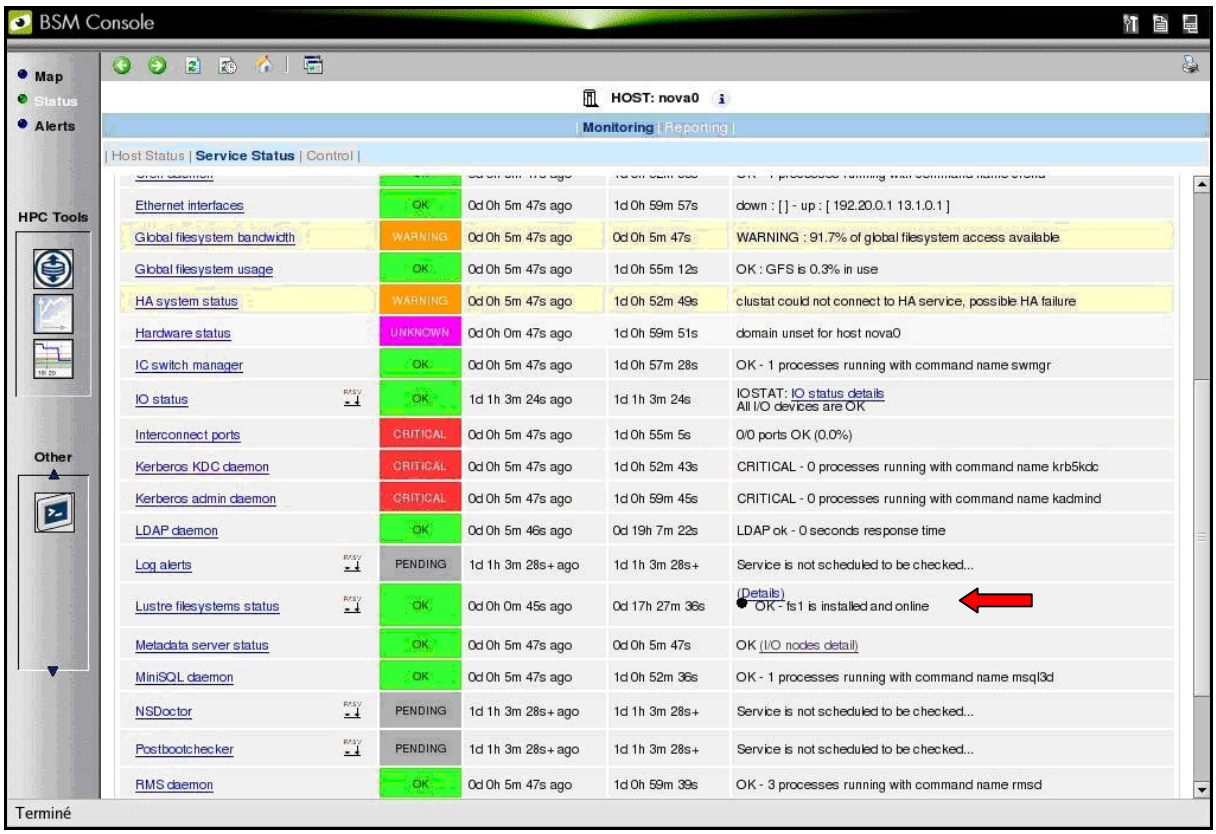

Figure 7-2. NovaScale Lustre File Systems Status

The **Lustre** file system indicator relates to the Lustre file systems health as a whole. Clicking on the info link will displays a detailed status for each file system running.

Lustre Management Node Web Interface

With a web browser, you can easily check the Lustre file system status using the following URL: http://<mangement node>/lustre

By clicking on the file system name, you can get details about the file system, using an interface that allows you to sort OSTs by name, active node, primary node, secondary node, device size, journal device, Config status, status or migration status.

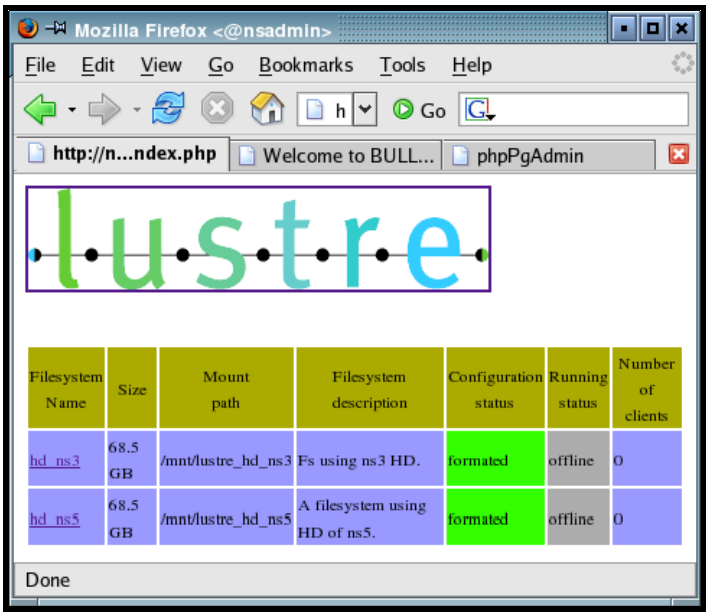

Figure 7-3. Lustre Management Node web interface

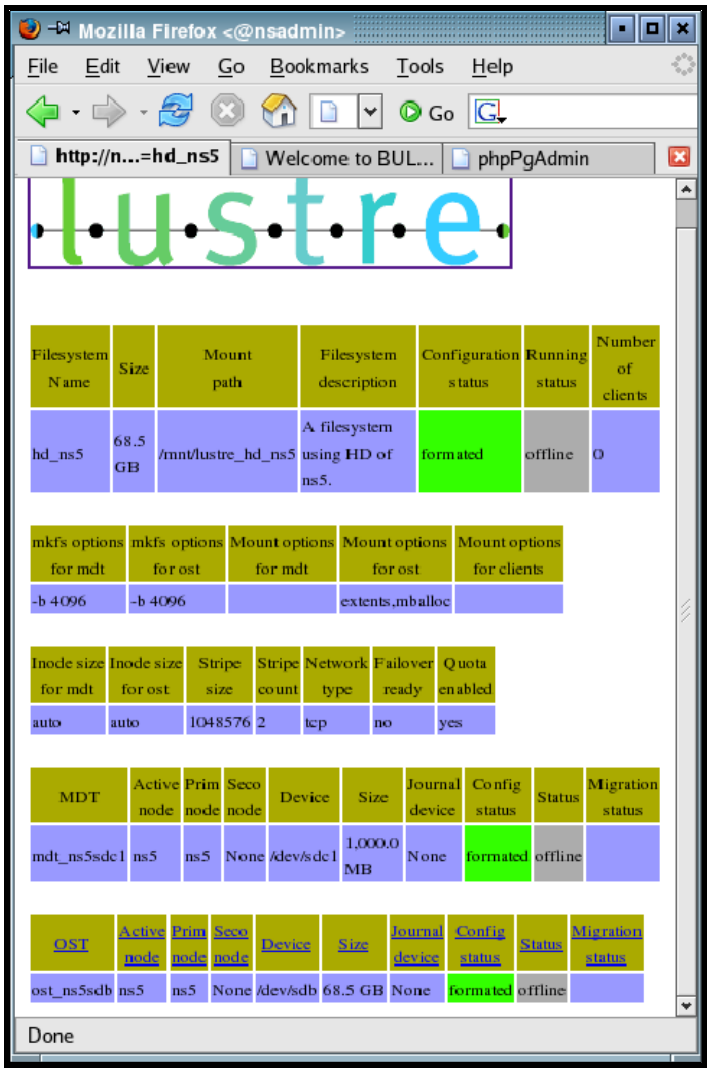

Figure 7-4. Detailed view of Lustre File Systems

7.1.3 Lustre System Performance Supervision

7.1.3.1 Group Performance Views

By clicking on the Group performance button in the Bull System Manager console the administrator is provided with an at-a-glance view of the transfer rates of the Lustre system for the file systems all together. The information period can be specified.

Clicking on the compiled view will display a dispatched view giving the performance rates node by node for the same time period.

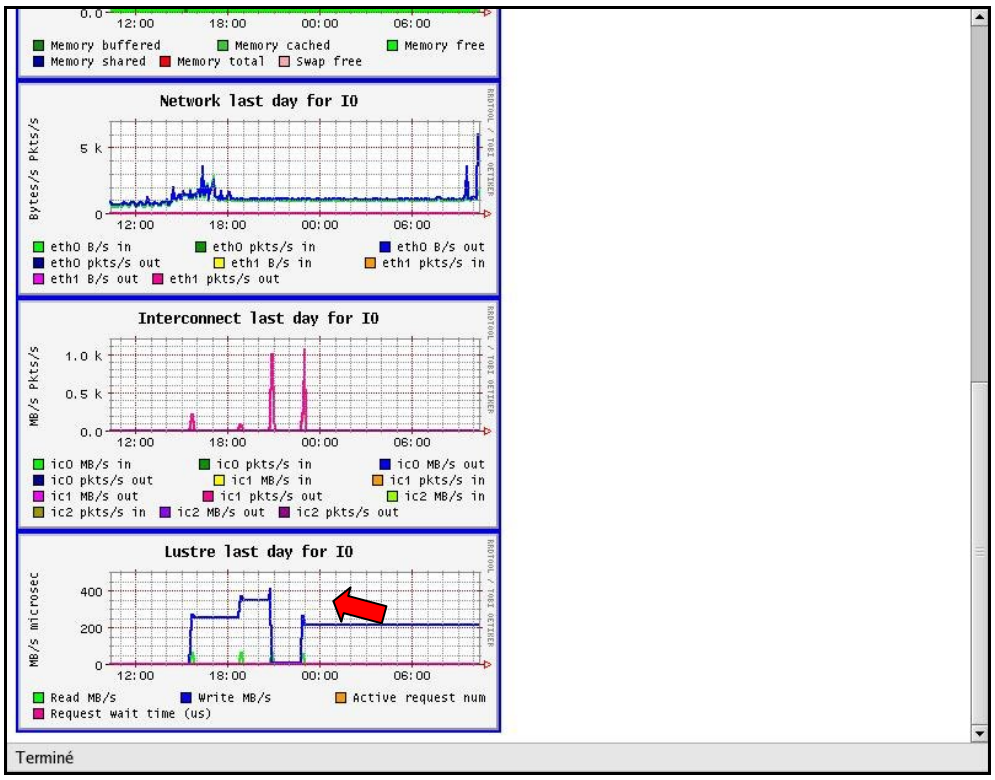

Figure 7-5. Group performance global view pop up window

Figure 7-6. Dispatched performance view pop up window

7.1.3.2 Node Performance Views

Views related to Lustre system local transfer and filling rates are available for each I/O node from the Global Performance view in the Bull System Manager Console.

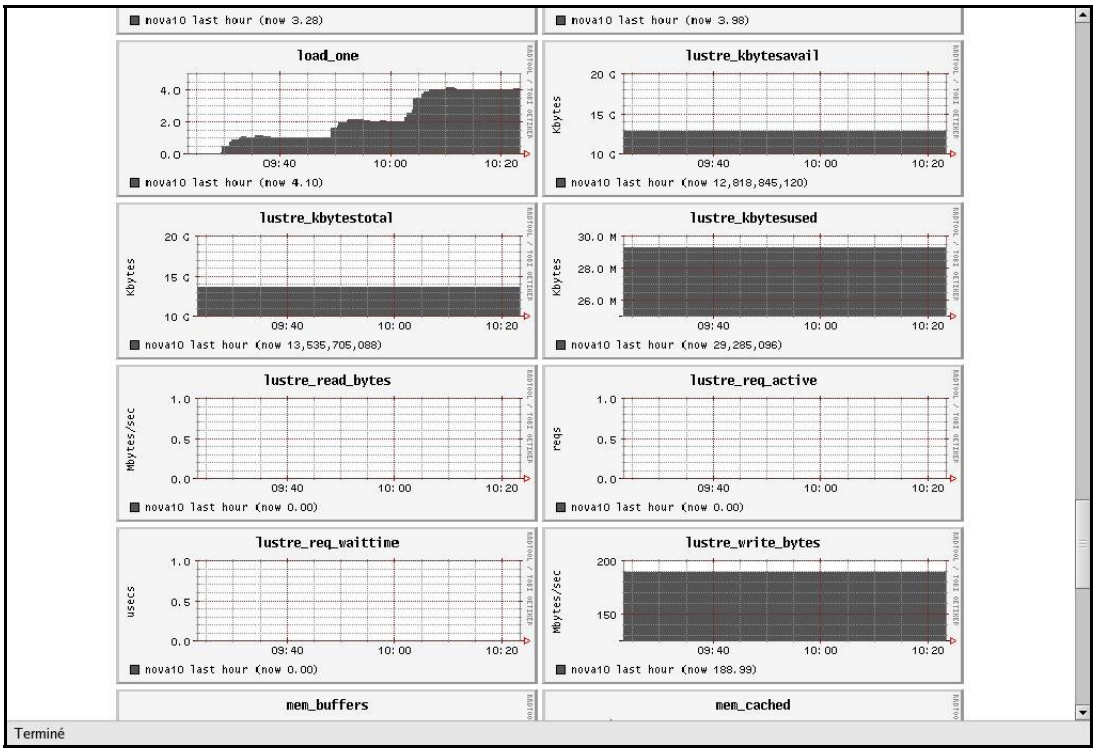

Figure 7-7. Global performance view pop up window

7.2 Monitoring Lustre High Availability

Two approaches are available from the monitoring tools: nodes migrations and the resulting OST/MDT file systems actual distribution.

On line commands allow the administrator to get an instant status of the Lustre High-Availability system.

If the cluster has a Management Node, important global health indicators are available via **Bull System Manager - HPC Edition** main view. They constitute a warning system for the administrator.

A trace system can be activated for debug and problem resolution purpose.

7.2.1 Command Line Monitoring

The following command displays the current failover paired nodes status under the form of an array with one line for each pair of nodes, as follows:

```
lustre_migrate nodestat
```

```
node name      node status    node HA name      node HA status
ns6            OK                   ns7                  MIGRATED
```

For each node, the status is that of the Lustre failover service it is primary for:

KO the Lustre failover service is UP

WARNING the Lustre failover service is UP some Lustre services are missing. A node migration may be in progress

MIGRATED the Lustre failover service has successfully migrated to the paired node and is now running on it

CRITICAL the Lustre failover service is no longer operating. The node migration has failed

The following command displays the current Lustre failover services distribution and status as seen by the HA Cluster Suite.

```
lustre_migrate hastat
```

For each Lustre file system installed on the cluster, the following command displays the detailed distribution of the MDTs and OSTs.

```
lustre_util info –f <File system name>
```

7.2.2 Graphic Monitoring

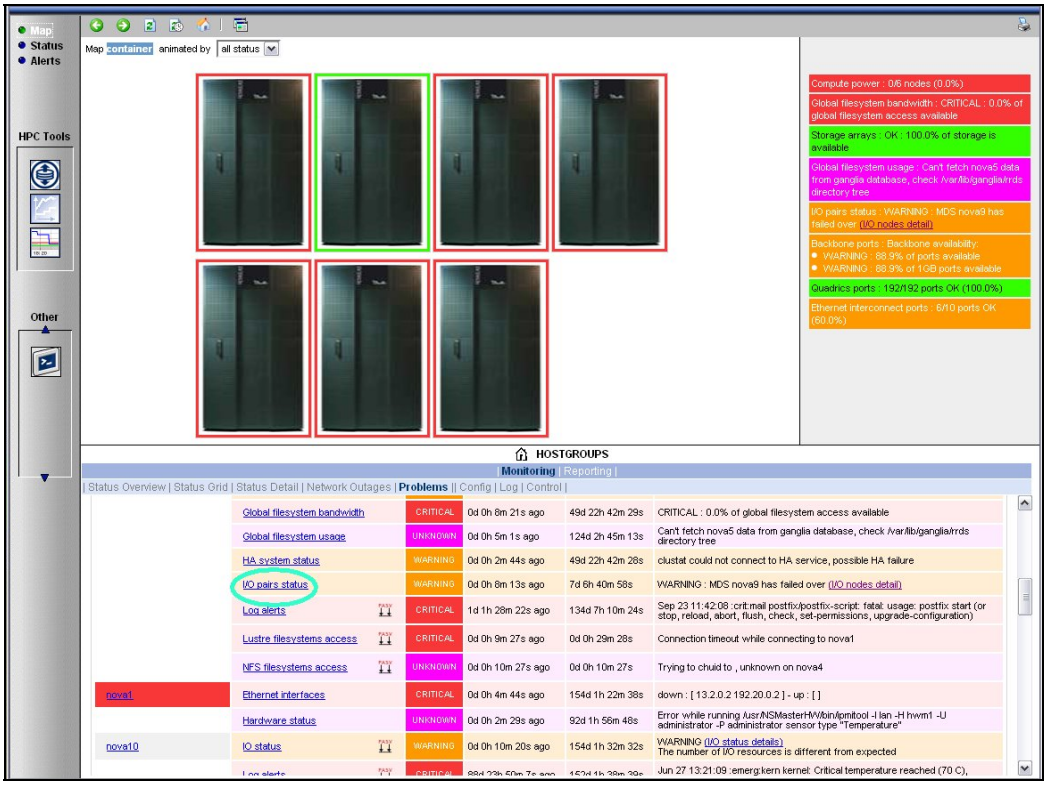

Figure 7-8. Bull System Manager Map all status screen with the I/O pairs status service highlighted for a host

The **I/O pairs status** alert indicates if a migration of the Metadata server has occurred. If a Warning is displayed click the **IO nodes details** link to see more details. The HA status for the I/O and Metadata nodes appears as shown in the example below

I/O and Metadata nodes High-Availability status

Node Name	Node Status	Paired Node Name	Paired Node Status
nova5	OK	nova9	MIGRATED
nova6	MIGRATED	nova10	OK

Figure 7-9. I/O and MetaData Node HA status

If a migration has occurred, the **Lustre** system is no longer Highly Available and an intervention by the Administrator to restore will be required highly urgently.

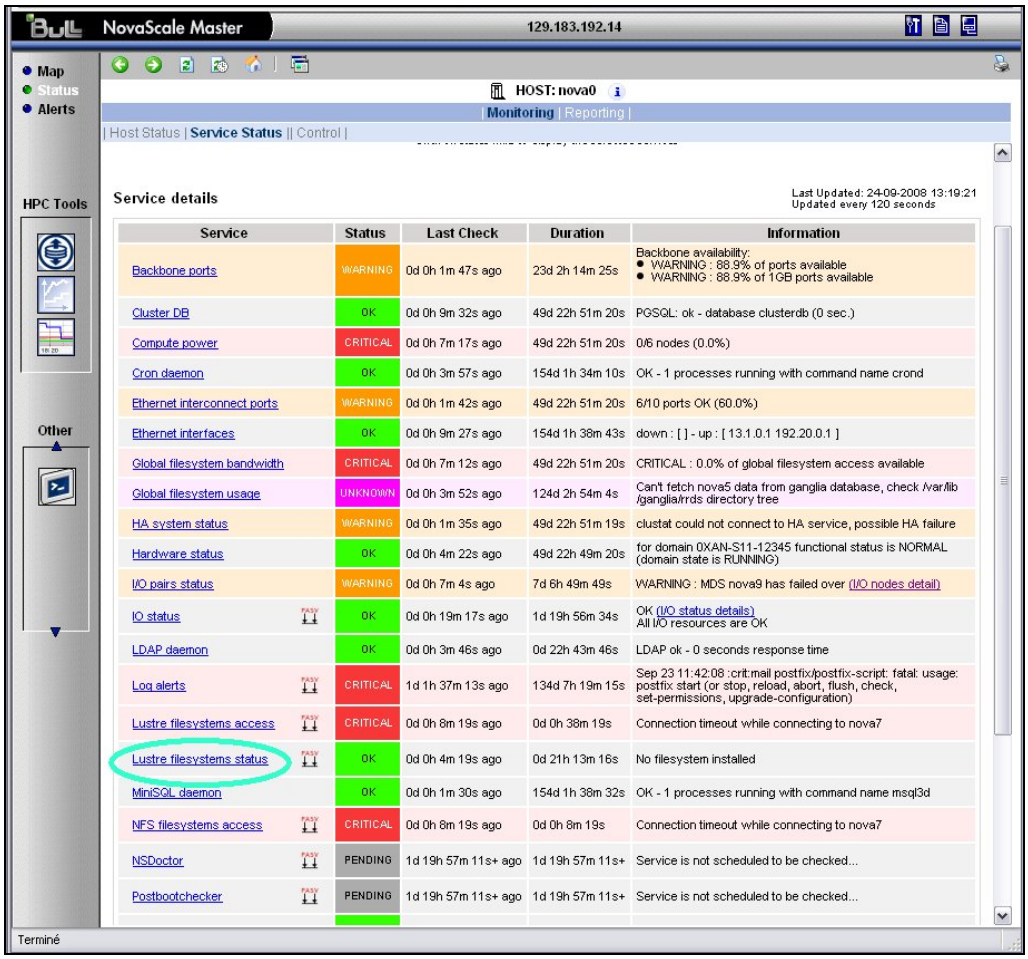

Figure 7-10. Lustre File System status indicator in the Host service status window

The **Lustre filesystems status** indicator warns about failures. Clicking on the info link will display a detailed status for MDTs/OSTs.

7.2.3 Traces and Debug

Failover Tools Traces

These are enabled by setting the **LUSTRE_DEBUG** parameter of the **/etc/lustre/lustre.cfg** file to **yes**.

On the Management Node, a daily log file is recorded under the **/tmp/log/lustre** directory by the **lustre_ldap** daemon, for example **//tmp/log/lustre/LDAP-<dd mm>.log**. It gives information about migration events transmitted to the **LDAP** directory.

On the Management Node, a daily log file is recorded under the **/var/log/lustre** directory by the **lustredbd** daemon. It records all the connections and update requests to the ClusterDB.

On the I/O and metadata nodes, a daily log file is recorded under the **/var/log/lustre** directory by the **lustre_failover** scripts. It gives information about failover events and their management.

Note The Administrator should remove the old log files on a regular basis.

System Log Files

On each I/O and Metadata Node, **HA Cluster Suite** and the failover scripts log events in the **/var/log/messages** and the **/var/log/syslog** files. These files are centralized on the Management Node by the **syslog-ng** system.

Chapter 8. Optimizing Lustre

This chapter explains how the **Lustre** parallel file system should be optimized. The following topics are described:

- *8.1 Monitoring Lustre Performance*
- *8.2 Lustre Optimization - Administrator*
- *8.3 Lustre Optimization – Application Developer*
- *8.4 Lustre File System Tunable Parameters*

To be fully optimized large cluster needs all its storage devices, and file systems of the input/output sub-system, to work in parallel with very high I/O rates and capable of accessing many processors at once. A distributed file system such as NFS is not sufficient for the requirements of the system.

There are separate sections in this chapter for the system administrator and the application developer. However, these are not exclusive, as any optimization of the Lustre file system will involve collaboration between these two. A lot of the optimizations need to be put in place when the platform is configured initially, and many aspects of application tuning cannot be done with user rights alone.

8.1 Monitoring Lustre Performance

The I/O performance of an application depends on:

1. The application itself, in particular how Input/Output data is sent to the File System
2. The performance of the system including the Linux kernel and drivers, and the Lustre file system.
3. Hardware performance including networking cards, disk arrays and storage devices.

These three aspects are interrelated and the overall performance for an application on a cluster depends on having a balance between all three. Different means exist for monitoring the performance of the file system and for identifying potential improvements.

8.1.1 Ganglia

Ganglia is a scalable, distributed, open-source monitoring tool, and is included as part of **Bull System Manager – HPC Edition**. This provides information about the cluster distribution for the **Lustre** file system and can be used for monitoring its performance in real time. This is important as the file system may be used by several different users at the same time, and if there is a perceptible drop in performance then **Ganglia** will indicate where there is uneven access to the files. Ganglia also collects Lustre statistics from **/proc/fs/lustre** in order to measure different aspects of file system performance.

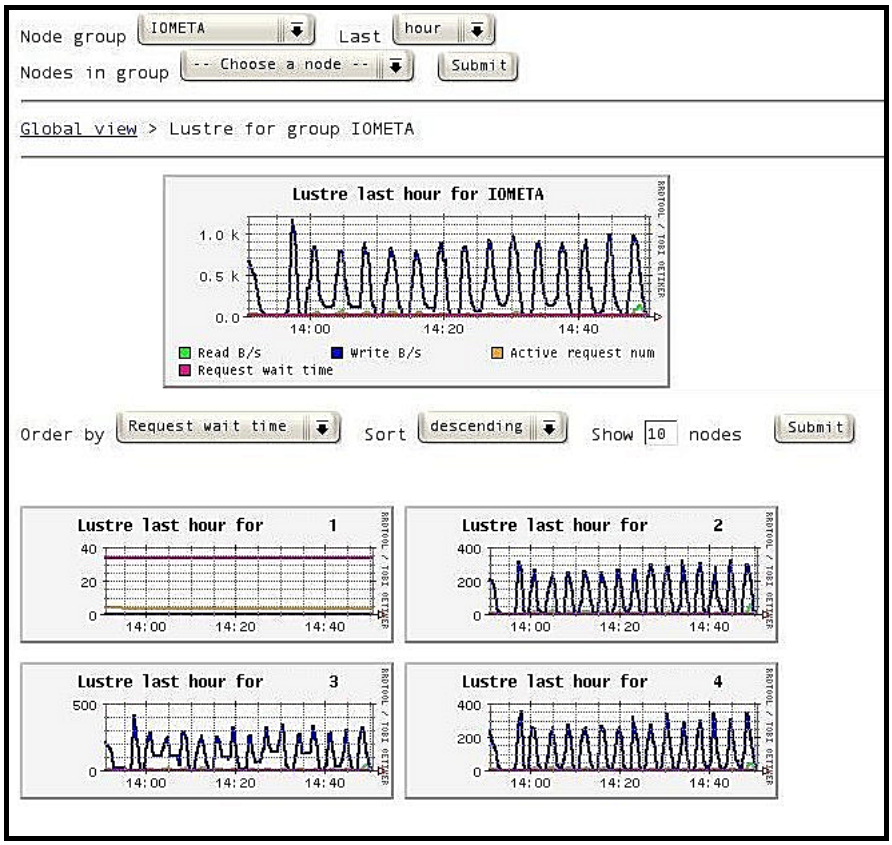

Figure 8-1 Ganglia Lustre monitoring statistics for a group of 4 machines with total accumulated values in top graph

8.1.2 Lustre Statistics System

Lustre itself collects a range of statistics. These are available in files in the **proc** File System. A list of these files can be retrieved by using the command:

```
find /proc/fs/lustre –name “*stats*”
```

8.1.3 Time

Time command – see the *Application Tuning Guide* for details. Here are two examples with the **dd** command which is used exclusively for I/O operations.

Example 1

```
# time dd if=/dev/zero of=/tmp/testfile bs=1M count=100
```

```
100+0 records in
100+0 records out
104857600 bytes transferred in 0.738386 seconds (142009176 bytes/sec)

real    0m0.749s
user    0m0.001s
sys     0m0.476s
```

If the system time is high, as in example 1, then this is an indication that the resources being used for the application I/O are high.

Example 2

```
# time dd if=/dev/zero of=/home/testfile bs=1M count=1000
```

```
1000+0 records in
1000+0 records out
1048576000 bytes transferred in 44.987850 seconds (23307982 bytes/sec)

real    0m45.039s
user    0m0.012s
sys     0m5.262s
```

If the sum of the CPU user time and the system time is significantly lower than real time, as in example 2, then this may be an indication that, again, the resources being used for the application I/O operations are high.

8.1.4 Iostat

When the host kernel has been configured to provide detailed I/O stats per partition the following information is available.

iostat can provide insight into the nature of I/O bottlenecks. It provides the nature and concurrency of requests being made of the attached storage devices.

iostat -x is invaluable for profiling the load on the storage devices which are part of a server node. The raw throughput numbers (wkB/s) combined with the requests per second (w/s) gives the average size of I/O requests to the device.

The service time indicates the amount of time it takes the device to respond to an I/O request. This sets the maximum number of requests that can be handled in turn when requests are not issued concurrently. Comparing this with the requests per second gives a measure of the amount of storage device concurrency.

Refer to the **iostat** man page for details regarding the meaning of the various columns in the output.

8.1.5 Llstat

llstat is a command which allows the examination of some of the Lustre statistics files. It 'decodes' the content by calculating statistics (min, max, mean, standard deviation) based on the contents of the file (sum and sum of squares).

See The *Lustre Operations Manual* Chapter 32.5.11 available on http://manual.lustre.org for more information.

8.1.6 Vmstat

The CPU use columns in the **vmstat** output file can be used to identify a node whose CPUs are completely occupied. On Metadata servers (**MDS**) and Object Storage Server (**OSS**) nodes, the I/O columns tell you how many blocks are flowing through the node's I/O subsystem. Coupled with the details of the attached storage for the node, it is then possible to determine if a subsystem in the node is the bottleneck.

Vmstat does not provide I/O block flow details for clients.

The columns that report swap activity can identify nodes that are having trouble keeping their working applications in memory.

8.1.7 Top

This tool is used to identify tasks which are using a lot of system resources. It can also be used to identify tasks which are not generating file system load, because they are using CPU or server threads which are struggling to obtain system resources on an overloaded node.

See The *Application Tuning Guide* for more details.

8.1.8 Strace

The **strace** command intercepts and records the system calls called and received by a running process and is used to measure I/O activity at the system level.

Two options which are useful are:

-e trace=file	traces activity related to system calls for file activity
-tttT	gives a microsecond resolution.

These two options combined allow the measurement and evaluation of the performance of file system calls. However whilst **strace** is being used the performance of the application may be impacted. It is possible to use **strace** command for each system call. For example, use **–e trace=write** option to analyze the write performance.

8.1.9 Application Code Monitoring

It is possible to add system calls in the code during the development of the application which can then be used to measure the I/O file performance of the application itself.

Another option is that any I/O operation debugging traffic is included within the **NFS** system and not the **Lustre** system in order to minimize any additional overhead in the use of system resources.

Note Increasing the debug level can lead to a major performance impact. Full debugging can slow the system by as much as 90%, compared to the default settings.

Benchmarking tools are easier to work with to study performance. Once the tuning changes have been made, a general purpose benchmarking tool can then be used to check for any adverse effects.

8.2 Lustre Optimization - Administrator

The **Lustre** system administrator will need to monitor the file system to check the overall performance, and to identify any areas where there may be possible degradation in the service. Lustre includes tools to monitor I/O performance. These can be used to evaluate any changes that are made to the application, and also to see if, and where, performance could be improved.

The application developer and large cluster administrator will need to be clear in their definition of the needs for the application at the outset, as some of the system configuration settings made when installing and configuring the system will impact the performance of the application. Some flexibility is possible, as there are parameters which can be modified after the cluster has been configured to meet the particular requirements of the application.

The developer needs to be aware that the Input/Output algorithms, specified for the application will have a big impact on performance and some performance compromises may have to be made for different parts of the application with respect to the overall performance for the complete program.

The main bottleneck within the whole system normally is the I/O speed of the data storage devices, as this is usually the slowest part of a cluster.

Note It may be possible to observe superlinear speedups for the I/O throughput using **Lustre** client cache.

Raw benchmarking data, including control data should be available for the systems. The objective is to get as close to these performance figures as is possible with the application in place.

Lustre is ideal for large sequential write I/O operations as used, for example, by checkpoint/restart. Using Lustre it should be possible to obtain 80 to 90% of the raw I/O performance figures (Write operations generally perform better than read operations).

Attention is particularly needed when small random I/O Metadata operations are being performed. This is because the data may be unevenly distributed throughout the system.

Several points have to be kept in mind when attempting to tune the file system:

- Lustre is a part of a shared file system which means that it will be difficult to obtain exclusive use of Interconnects and data storage devices. For clients who need to have exclusive use of the file system, it is possible to do this by mounting it directly on the clients. This is in contrast to CPUs and memory where an application can be given exclusive use easily. Overall there are a lot of variables which can impact an application's performance.
- System caches can have a positive effect on performance if all the I/O traffic takes place in the client cache – in fact it is possible that the application bandwidth may appear greater than the disk bandwidth. System caches can also have a negative effect as Lustre's **readahead** option may impact performance.

- There are a lot of data pipelines within the Lustre architecture. Two in particular have a direct impact on performance. Firstly, the network pipe between clients and OSSs, and secondly the disk pipe between the OSS software and its backend storage. **Balancing these two pipes maximizes performances**.

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

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

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

Note It is recommended that MDT file systems be formatted with the inode large enough for the default number of stripes per file to improve performance and storage efficiency.

One needs to keep enough free space in the MDS file system for directories and external blocks. This represents ~512 Bytes per inode.

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

Note It is recommended to stripe over as few objects as possible to limit network overhead and reduce the risk of data loss when there is an OSS failure.

The stripe size must be a multiple of the page size. The smallest recommended stripe size is 1 MB because Lustre tries to batch I/O into 1 MB blocks on the network.

8.2.1 Stripe Tuning

Check that enough stripes are being used

It is important to remember that the peak aggregate bandwidth for I/O to a single file is restricted by the number of stripes multiplied by peak bandwidth per server. No matter how many clients try to write to that file, if it only has one stripe, all of the I/O will go to only one server.

Check that files are striped evenly over the Object Storage Targets

Lustre will create stripes on consecutive OSTs by default, so files created at one time will be optimally distributed among OSTs, assuming there are enough stripes and/or files created

at that time. However, files created at different times may not have an optimal distribution among OSTs. To ascertain the file distribution, use the following command:

```
lfs getstripe
```

For more information refer to **lfs** man page.

If some servers appear to be receiving a disproportionate share of the I/O load check that the files are striped evenly over the OSTs.

If the I/O load is unbalanced for servers then use the **lfs** command to create a balanced set of files before the application starts, or if applicable, restructure the application so that Lustre striping is more efficient.

See *Chapter 25 - Striping and I/O Options* in the *Lustre Operations Manual* available on http://manual.lustre.org, for more information on File Striping.

8.3 Lustre Optimization – Application Developer

The main determinant on performance for the Lustre file system is the file size and how this is handled by the I/O devices. **POSIX** will handle the parallel distribution of the file, however, if the file is large performance may be impacted. The application developer has to decide if it is possible to chunk the program and thus gain performance.

The optimal level of performance is when the cluster platform I/O device read/write operations is as near as possible to that of the raw Lustre file system performance without the application running. Depending on the application program it should be possible to achieve 80% to 90% of the performance of a 'clean' cluster system.

One of the key questions to look at is to ascertain if the application performs I/O from enough client nodes to take full advantage of the aggregate bandwidth provided by the Object Storage Servers.

8.3.1 Striping Optimization for the Developer

Default striping settings are usually in the hands of the Lustre administrator who will normally use the default values. However, the application developer can also change these settings by using the **lfs** command on a per directory or on a per file basis. This controls the way parallel I/O operations are carried out for the files. Refer to the man page displayed under **lfs(1)** for more information.

Optimal striping settings depend primarily on the file size. It does not make sense to stripe a small file over several OSTs, on the other hand it does make sense to stripe a big file over several OSTs.

It is recommended to use the default striping settings configured by the Lustre administrator.

If the striping is to be changed, it is best to perform I/O tests with different striping configurations in order to find the best possible striping configuration.

8.3.2 POSIX File Writes

Being a high-performance distributed file system makes Lustre especially complex. By being POSIX compliant this complexity is simplified and no code modification is required whether a code is run on a local file system (**ext3, xfs**) or on Lustre. Only performance is enhanced.

There are several points to be kept in mind for file writes. Writes flow from the application that generates them to OSTs where they are placed within the storage system. The network between the client and storage target needs to have capacity for the write traffic. It is also advisable to look for possible choke points along this path.

It should be said that these problems will only occur in extreme cases on the large cluster platform.

There must be enough write capacity for the application

If an application is to exploit a large network and disk pipes, it must generate a lot of write traffic, which can be cached on the client node and packaged into **RPC**s for the network. There must be enough free memory on the node for use as a write cache. If the kernel cannot keep at least 4 MB in use for Lustre write caching, it cannot keep an optimal number of network transactions in progress at once.
There must be enough CPU capacity for the application to do the work which generates data for writing.

There must be enough storage space available

To prevent a situation in which Lustre puts application data into its cache, but then cannot write it to disk because the disk is full, Lustre clients must reserve disk space in advance. However, if it is unable to reserve this space as the OSTs are almost full, less than 2% space available, it must execute its writes synchronously with the server, instead of caching them for efficient bundling.

The degree to which this affects performance depends on how much your application would benefit from write caching. The **cur_dirty_bytes** file in the subdirectory of each OSC of **/proc/fs/lustre/osc/** on a client records the amount of cached writes which are destined for a particular storage target.

The maximum amount of cached data per OSC is determined by the **max_dirty_mb** value in the same directory. This is usually 4 MBs by default. Increasing this value will allow more dirty data to be cached on a client before it needs to flush to the OST, but also increases the time needed for other clients to read or overwrite the cached data once it has been written to the **OST**.

Server thread availability

Write **RPC**s arrive at the server and are processed synchronously by kernel threads (named ll_ost_*). **ps** will help to identify the number of threads that are in the D state indicating that they're busy servicing a request.

Vmstat –see section 8.1.6 - can give a rough approximation of the number of threads that are blocked processing I/O requests when a node is busy servicing only I/O **RPC**s. The number of threads sets an upper bound on the number of I/O **RPC**s that can be processed concurrently, which in turn sets an upper limit for the number of I/O requests that will be serviced concurrently by the attached storage.

8.3.3 Fortran

Particular attention may be necessary for the I/O operations of Fortran as opposed to C as the Fortran run time library may modify the way the I/O operations are programmed. Please refer to the *Application Tuning Guide* for more information on Fortran compiler optimizations, or to the compiler documentation from Intel, or to the manual page for the **ifort** command with particular reference to the section on environmental variables. In particular, the environmental variables **FORT_BUFFERED**, **FORT_CONVERT*** and **F_UFMTENDIAN** should be looked at.

8.4 Lustre File System Tunable Parameters

WARNING
Changing tunable parameters of the lustre file system can render the file system non functional. It should be done with great care on production File Systems. The use of default values is recommended.

One scenario where tuning the file system is beneficial is a cluster with several file systems, some of which have clearly defined workloads. For these file systems, the file system can then be tuned and optimized for this clearly defined workload. For example, if a file system is used only for checkpoint/restart purposes; the workload for this file system will probably consist of large sequential write and read I/O operations. It is then beneficial to tune the file system for this particular workload, particularly if the cluster has a large amount of memory.

Another example is when performing benchmarking: (temporary) changes may be applied in order to optimize benchmark through put.

This section describes the tuneable parameters of the Lustre file system.
For the syntax see the **lustre_util** section in Appendix A.

See *Chapter 20.2 - Lustre I/O Tunables* in the *Lustre Operation Manual* on http://manual.lustre.org for more details.

8.4.1 Tuning Parameter Values and their Effects

max_read_ahead_mb

```
/proc/fs/lustre/llite/fs0/max_read_ahead_mb
```

This parameter defines the per-file read-ahead value for a client. Defaulting to 40MB **Read_ahead** is a two-edged sword: this can increase the read throughtput, but can be inefficient (if a file is read randomly rather than sequentially), and in turn detrimental as the memory which is wasted is not available elsewhere. The default value of 40MB is a general purpose value. It may be beneficial to increase this for sequential read workloads, whilst in other situations it may be better to disable it completely.

max_cache_mb

```
/proc/fs/lustre/llite/fs0/max_cache_mb
```

This parameter defines the maximum amount of inactive data cached by the client (the default value is ¾ of the RAM which is available).

max_dirty_mb

```
/proc/fs/lustre/osc/<...>/max_dirty_mb
```

This parameter has a value between 0 and 512MB.
This value controls the write back cache on the client per OSC. While it is beneficial to use larger values, the quantity of dirty data can become so high that, depending on the number of clients, it results in a significant amount time being needed to copy the data to disk.

max_page_per_rpc

```
/proc/fs/lustre/osc/<...>/max_page_per_rpc
```

This value should not be changed from the default value as the optimal value depends on the kernel page size.

max_rpc_in_flight

This value should not be changed from the default value as the optimal value depends on characteristics of the machine.

lru_size

```
/proc/fs/lustre/ldlm/ldlm/namespaces/<OSC name|MDC name>/lru_size
```

Increasing the default value is recommended for login nodes using **lustre** and for improving metadata performance.

debug

```
/proc/sys/lnet/debug
```

The debug level can impact the performance. This is the reason why it is disabled by default. When analyzing problems, different values may be used. The exact optimal value depends on the problem being analyzed: **full debugging** (**-1** value) can slow the file system noticeably and even mask the problem under diagnosis.

Note Increasing the debug level can lead to a major performance impact. Full debugging can slow the system by as much as 90%, compared to the default settings.

Chapter 9. Troubleshooting Lustre

This chapter describes the following topics:

- 9.1 *Troubleshooting Lustre*
- 9.2 *Troubleshooting Lustre File System High Availability*

9.1 Troubleshooting Lustre

The following section helps you troubleshoot some of the problems affecting your Lustre file system. Because typographic errors in your configuration script or your shell script can cause many kinds of errors, check these files first when something goes wrong.

First be sure your File-system is mounted and you have mandatory user rights.

9.1.1 Hung Nodes

There is no way to clear a hung node except by rebooting. If possible, un-mount the clients, shut down the MDS and OSTs, and shut down the system.

9.1.2 Suspected File System Bug

If you have rebooted the system repeatedly without following complete shutdown procedures, and Lustre appears to be entering recovery mode when you do not expect it, take the following actions to cleanly shut down your system.

1. Stop the login nodes and all other Lustre client nodes. Include the **-F** option with the **lustre_util command** to un-mount the file system.

```
#lustre_util umount –F –f <file_system> -n <node_name>
```

2. Shut down the rest of the system.

3. Run the **e2fsck** command.

9.1.3 Cannot re-install a Lustre File System if the status is CRITICAL

If the status of a file system is CRITICAL (according to the **lustre_util status** command), and if the file system needs to be re-installed (for instance if some nodes of the cluster have been deployed and reconfigured), it is possible that the file system description needs to be removed from the cluster management database, as shown below:

1. Run the following command to install the `fs1` file system:

```
lustre_util install -f /etc/lustre/models/fs1.lmf
```

The command may issue an output similar to:

```
file system already installed, do "remove" first
```

2. Run the following command to remove the `fs1` file system:

```
lustre_util remove -f fs1
```

The command may fail with a message similar to:
`file system not loaded, try to give the full path`
If it is not possible to re-install neither remove the file system with force option (-F).

The **lustre_fs_dba** command can then be used to remove the file system information from the cluster management database.
For example, to remove the `fs1` file system description from the cluster management database, enter the following command:

```
lustre_fs_dba del -f fs1
```

After this command the file system can be re-installed using the **lustre_util install** command.

9.1.4 Cannot create file from client

If you get the following error message when you try to create a new file from a Lustre client, it simply means that the user (UID) you use to create the file is not recognized by the Lustre File System:

```
touch: cannot touch `/mnt/lustre/myfile': Identifier removed
```

To avoid such problems, all users (UID) that exist on the Lustre client nodes must also exist on the MDS server.

9.1.5 No such device

If the start of the Lustre File System fails with the following message, most of the time it is due to the fact that InfiniBand is not properly configured on the Lustre nodes:

```
mount.lustre: mount /dev/ldn.lustrefda2500.4 at
/mnt/srv_lustre/scratch/scratch-OST0003 failed: No such device
Are the lustre modules loaded?
```

This is confirmed by the following lines in the system logs of the machine from which the problem is coming:

```
LustreError: 11602:0:(o2iblnd.c:1569:kiblnd_startup()) Can't query
IPoIB interface ib0: it's down
LustreError: 105-4: Error -100 starting up LNI o2ib
```

Please pay particular attention to the fact that the **IPoIB** interface has to be fully functional in order to start and run Lustre. Despite that fact that Lustre data is not transmitted on the IPoIB interface, IPoIB is used by Lustre to create and manage InfiniBand connections.

9.2 Troubleshooting Lustre File System High Availability

Before using a Lustre file system configured with the High Availability (HA) feature, or in the event of abnormal operation of HA services, it is important to perform a check-up of the Lustre HA file system. This section describes the tools that allow you to make the required checks.

9.2.1 On the Management Node

The following tools must be run from the Management Node.

lustre_check

This command updates the **lustre_io_nodes** table in the ClusterDB. The **lustre_io_nodes** table provides information about the availability and the state of the I/O nodes and metadata nodes.

lustre_migrate nodestat

This command provides information about the node migrations carried out. It indicates which nodes are supposed to support the OST/MDT services.

In the following example, the MDS are `nova5` and `nova9`, the I/O nodes are `nova6` and `nova10`. `nova5` and `nova6` have been de-activated, so their services have migrated to their pair-nodes (`nova9` and `nova10`).

```
lustre_migrate nodestat
```

```
HA paired nodes status
----------------------
node name  node status    HA node name  HA node status
    nova5     MIGRATED          nova9              OK
    nova6     MIGRATED          nova10             OK
```

Note This table is updated by the **lustre_check** command.

lustre_migrate hastat [-n <node_name>]

This command indicates how the Lustre failover services are dispatched, after CS4 software has been activated.
Each node has a view on the paired failover services (the failover service dedicated to the node and the failover service dedicated to its pair node). If the pair-node has switched roles, the `owner` column of the command output will show that this node supports the two lustre_HA services.

In the following example, `nova6` and `nova10` are paired I/O nodes. The `lustre_nova6` service is started on `nova10` (owner node). This status is consistent on both `nova6` and `nova10` nodes.

```
lustre_migrate hastat -n nova[6,10]
```

```
----------------
nova10
----------------
Member Status: Quorate, Group Member
  Member Name                    State      ID
  ------ ----                    -----      --
  nova6                          Online     0x0000000000000001
  nova10                         Online     0x0000000000000002
  Service Name       Owner (Last)             State
  ------- ----       ----- ------             -----
  lustre_nova10      nova10                   started
  lustre_nova6       nova10                   started
----------------
nova6
----------------
Member Status: Quorate, Group Member
  Member Name                    State      ID
  ------ ----                    -----      --
  nova10                         Online     0x0000000000000002
  nova6                          Online     0x0000000000000001
  Service Name       Owner (Last)             State
  ------- ----       ----- ------             -----
  lustre_nova10      nova10                   started
  lustre_nova6       nova10                   started
```

To return to the initial configuration, you should stop `lustre_nova6` which is running on `nova10` and start it on `nova6`, using the `lustre_migrate relocate` command.

lustre_util status

This command displays the current state of the Lustre file systems.

Important

- **Sometimes this command can simply indicate that the recovery phase has not finished; in this situation the status will be set to "WARNING" and the remaining time will be displayed.**
- **When an I/O node have been completely re-installed following a system crash, the Lustre configuration parameters will have been lost for the node. They need to be redeployed from the Management Node by the system administrator. This is done by copying all the configuration files from the Management Node to the I/O node in question by using the scp command as shown below:**
  `scp/etc/lustre/conf/<fs_name>.xml<io_node_name>:/etc/lustre/conf/<fs_name>.xml`
  <fs_name> is the name for each file system that was included on the I/O node before the crash.

lustre_util info

This command provides detailed information about the current distribution of the OSTs/MDTs. The services and their status are displayed, along with information about the primary, secondary and active nodes.

/tmp/log/lustre/lustre_HA-*ddmm*.log

This file provides a trace of the commands issued by the nodes to update the LDAP and ClusterDB databases. This information should be compared with the actions performed by CS5.

Note In **lustre_HA-*ddmm*.log**, *dd* specifies the day and *mm* the month of the creation of the file.

/var/log/lustre/HA-DBDaemon=*yy-mm-dd*.log

This file provides a trace of any ClusterDB updates that result from the replication of LDAP. This could be useful if Lustre debug is activated at the same time.

9.2.2 On the Nodes of an I/O Pair

The following tools must be run from the I/O nodes.

ioshowall

This command allows the configuration to be checked.
Look at the **/etc/cluster/cluster.conf** file for any problems if the following error is displayed:

```
-- cannot connect to < PAP address> or HWMANAGER
```

If the following error appears, check if the node is an inactive pair-node, otherwise start the node again:

```
-- service lustre_ha inactif
```

clustat

Displays a global status for HA Cluster Suite 4, from the HA cluster point of view.

If there is a problem, the two pair nodes may not have the same view of the HA cluster state.

storioha -c status

This command checks that all the HA Cluster Suite 4 processes are running properly ("running state").

Notes

- This command is equivalent to the following one on the Management Node:
  `stordepha -c <status> -i <node>`
- This command is included in the global checking performed by the **ioshowall** command.

stormap -l

This command checks the state of the virtual links.

Note This command is included in the global checking performed by the **ioshowall** command.

lctl dl

This command checks the current status of the OST/MDT services on the node.
For example:

```
1 UP lov fs1_lov-e0000047fcfff680 b02a458d-544e-974f-8c92-23313049885e 4
2 UP osc OSC_nova9_ost_nova6.ddn0.11_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
3 UP osc OSC_nova9_ost_nova10.ddn0.5_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
4 UP osc OSC_nova9_ost_nova6.ddn0.3_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
5 UP osc OSC_nova9_ost_nova10.ddn0.21_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
6 UP osc OSC_nova9_ost_nova6.ddn0.19_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
7 UP osc OSC_nova9_ost_nova10.ddn0.7_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
8 UP osc OSC_nova9_ost_nova6.ddn0.1_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
9 UP osc OSC_nova9_ost_nova10.ddn0.23_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
10 UP osc OSC_nova9_ost_nova6.ddn0.17_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
11 UP osc OSC_nova9_ost_nova10.ddn0.13_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
12 UP osc OSC_nova9_ost_nova6.ddn0.9_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
13 UP osc OSC_nova9_ost_nova10.ddn0.15_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
14 UP mdc MDC_nova9_mdt_nova5.ddn0.25_MNT_clientelan-e0000047fcfff680
b02a458d-544e-974f-8c92-23313049885e 4
```

The last line indicates the state of the MDC, which is the client connecting to the MDT (on the MDS).
The other lines indicate the state of the OSC, which are the clients connecting to each OST (on the `nova6` and `nova10` OSS).

/var/log/lustre/HA_yy-mm-dd.log

This file provides a trace of the calls made by CS5 to the Lustre failover scripts.

Note In the **HA_yy-mm-dd.log** file, yy specifies the year, *mm* the month and *dd* specifies the day of the creation of the file.

/var/log/syslogfile

This file provides a trace of the events and activity of CS5 and Lustre.

Pair Node Consistency

In some very specific cases, it may be necessary to reset the HA system to a state which ensures consistency across the pair-nodes, **without stopping** the Lustre system.

1. Disconnect the `fs1` Lustre File System from the HA system:

```
lustre_ldap unactive -f fs1
```

2. Run **clustat** to view the location of the services:

```
clustat
```

3. Perform one of the following actions:
   - To switch a node from primary state to pair-node state, run:

```
lustre_migrate export -n <node_name>
```

   - Or, to reset the switched node back to its primary state, run:

```
lustre_migrate relocate -n <node_name>
```

4. Re-connect the Lustre File System to the Lustre HA system:

```
lustre_ldap active -f fs1
```

Appendix A. Configuration Files and Reference

This Appendix describes the syntax of the following files and commands:

lustre.cfg file

Lustre management tools use the **lustre.cfg** Lustre Management Configuration File (**/etc/lustre/lustre.cfg**) to get configuration information. This file must reside on all OSS and MDS nodes. Refer to *lustre_util command*, on page 88 to know how to distribute this file easily.

File Syntax:

```
VARIABLE=VALUE
```

Lines beginning with # are comments.

LUSTRE_MODE=XML

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

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

CLUSTERDB=yes

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

LUSTRE_CONFIG_DIR=/etc/lustre/conf/

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

Default value is /etc/lustre/conf/.

LUSTRE_NET=tcp or elan or o2ib

This variable specifies the kind of network used by Lustre for the whole cluster. It is deprecated and should be left set to the **tcp** default value. It is now used by the **lustre_check** monitoring tool only.

LUSTRE_ADMIN=hostname

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

No default value is defined.

LUSTRE_ADMIN2=hostname

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

No default value is defined.

LUSTRE_LDAP_URL=ldap://hostname/

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

No default value is defined.

LUSTRE_LDAP_URL2=ldap://hostname/

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

No default value is defined.

LUSTRE_DB_DAEMON_HOST=hostname
LUSTRE_DB_DAEMON_HOST2=hostname
LUSTRE_DB_DAEMON_PORT=tcp port

These variables should be set if High Availability is configured with the ClusterDB and are used to specify the http server daemon that updates the ClusterDB. LUSTRE_DB_DAEMON_HOST2 is to be used when the Management Node does not support the High Availability feature. An alternative LUSTRE_DB_DAEMON hostname will be provided as a backup.

No default values are defined for the hostnames. The default value for the tcp port is 56283, e.g. 0xDBDB so this appears as **LUSTRE_DB_DAEMON_PORT=56283**

LUSTRE_DEBUG=yes or no

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

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

Default value is no.

LUSTRE_MGS_HOST=name of the Management Node where the MGS service is installed.

This value is used by the **lustre_util** tool to link the **MGS** with others Lustre entities, for example, **MDS, OSS**.

LUSTRE_MGS_NET= the name of the network used to read the MGS, for example, TCP or o2ib.

When the **o2ib** net type is used the **LUSTRE_MGS_HOST** name value has to be suffixed with'-ic0' which is hostname suffix for IB networks. For example, if you need to use an **InfiniBand** network to reach the MGS entity that runs on the node **zeus6** you have to:

- set **LUSTRE_MGS_NET** to **o2ib**
- set **LUSTRE_MGS_HOST** to **zeus6-ic0**

LUSTRE_MGS_ABSOLUTE_LOOPBACK_FILENAME = file for mgs loop device

The default is **/home/lustre/run/mgs_loop**. When High Availability exists for the Management Node, select a directory which is shared for the Management Node pairs. This value is used by the **MGS** service when **lustre_util** is not used.

I/O scheduler for block devices

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

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

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

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

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

Default and recommended values are:

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

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

LUSTRE_SNMP=yes or no

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

Default value is no.

DISABLE_LUSTRE_FS_NAGIOS=yes or no

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

Default value is no

LUSTRE_TUNING_FILE=/etc/lustre/tuning.conf

This is the path to the tuning file.

The default value is **/etc/lustre/tuning.conf**.

lustre_util.conf file

The **lustre_util.conf** file (**/etc/lustre/lustre_util.conf**) contains some additional settings for **lustre_util.** The following values are set by default:

- Timeout in seconds given to the **connect_timeout** parameter of SSH:

```
ssh_connect_timeout=20
```

- Timeout in seconds for install, update and rescue operations and can be overwritten by the **-t** option:

```
install_timeout=0
```

- Timeout in seconds for the start operation and can be overwritten by the **-t** option:

```
start_timeout=0
```

- Timeout in seconds for the mount operation and can be overwritten by the **-t** option:

```
mount_timeout=60
```

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

```
umount_timeout=60
```

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

```
stop_timeout=0
```

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

```
status_timeout=30
```

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

```
set_ioscheds_timeout=60
```

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

```
set_tuning_timeout=60
```

- **yes** will disable the update of the nagios pipe by **lustre_util**:

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

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

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

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

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

- Unless you explicitly want to use loop device, this should be set to no. This way, it prevents **lconf** to create huge loop devices in **/dev/** directory when some LUNS disappear:

```
allow_loop_devices=no [yes]
```

- If set to yes, only nodes that are assumed to mount a file system will be checked on **status** and **mnt_status** operation:

```
check_only_mounted_nodes_on_mnt_status=no [yes]
```

- Number of **ssh** connexions allowed to run at the same time. Can be overloaded using **--fanout** option:

```
default_fanout=128
```

tuning.conf file

The **tuning.conf** file (**/etc/lustre/tuning.conf**) contains tuning parameters.

File Syntax:

```
"<string>" <file> <target> [<delay>] [<file systems>]
```

"<string>" String to write in file, it can contain spaces, MUST be between double-quotes.

<file> Full path to the file where string will be written. Globbing is allowed. 2 macros can be used:
${mdt} stands for the name of the **MDT** of the file system.
${ost} stands for the name of ALL the **OSTs** (one line will be generated for each ost).

<target> A string composed of the **OSS, MDS**, or **CLT**, separated by semicolons. **OSS**, **MDS** and **CLT** can be followed by a nodes list (**pdsh syntax**) using colon.

<delay> A time in ms that we have to wait before continuing setting tuning parameters on a node. This is an optional argument, and the default is 0 ms.

<file system> A list of file system separated with semicolons. This is an optional argument, and the default is to allow this tuning for every file systems.

For OSS and MDS, tuning parameters are set when a file system is started. For Clients, tuning parameters are set when the file system is mounted, for example:

```
"1"   proc/sys/lnet/panic_on_lbug OSS;MDS;CLT
```

This line will enable panic on **lbug** on **ALL** types of node for all file systems by running `echo "1" >proc/sys/lnet/panic_on_lbug` on all nodes.

```
"0"   /proc/sys/lnet/panic_on_lbug   OSS:ns[5-6];MDS:ns3  fs1;fs2
```

This line will disable panic on **lbug**:

- on `ns5` and `ns6`, if they are used as an OSS of `fs1` and/or `fs2`
- on `ns3`, if it is used as MDS of `fs1` and/or `fs2`

String, **file** and **target** can be aliased using the following syntax:

alias <name>=<content>

alias can be declared anywhere in the file, but it also acts on the **WHOLE** file, not only on the lines that follow the declaration.

When you use **alias** on a string, the alias must also be in double quotes.

Example:

A **tuning.conf** example file is shown below:

```
####  ALIAS DECLARATION ###################################

alias health_check=/proc/fs/lustre/health_check
alias panic_on_lbug=/proc/sys/lnet/panic_on_lbug
alias ping_osc=/proc/fs/lustre/osc/*${ost}*/ping
alias debug=/proc/sys/lnet/debug
#### TUNING PARAMETER ###################################

"1"                    ping_osc        CLT

"0"                    panic_on_lbug   CLT

"0"                    panic_on_lbug   OSS;MDS

"524288"               debug           OSS;MDS;CLT
```

Lustre Model file (.lmf)

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

File Syntax:

```
keyword: <value>
```

Lines beginning with # are comments.

Possible Keywords:

stripe_size	Specify the stripe size in bytes. Default is 1048576 (1M).
stripe_count	Specify the number of OSTs each file should be striped onto. The default value is 2.
stripe_pattern	Only pattern 0 (RAID 0) is supported currently.
nettype	Possible values are **tcp** or **elan** or **o2ib**. The default is elan.
generic_client	The name of the directory from which the file system is mounted on the mds. If the network is elan, the default is clientelan. If network is tcp, default is **clienttcp**.
fstype	File system type. Possible values are **ldiskfs** or **ext3**. Default (and recommended) is **ldiskfs**.
failover	Enable failover support on OST. Possible values are yes or no. The default is no.
lustre_upcall	Location of the Lustre **upcall** script used by the client for recovery. No default script is defined.
portals_upcall	Location of the Portals **upcall** script used by the client for recovery. No default script is defined.
mdt_mkfs_options	Optional argument to **mkfs** for MDT. By default, no option is specified.
mdt_inode_size	Specify the new **inode** size for the underlying MDT ext3 file system. The default is self-evaluated.
mdt_mount_options	Optional argument to mount fs locally on the MDS. By default, no option is specified.
ost_mkfs_options	Optional argument to mkfs for OST. By default, no option is specified.
ost_inode_size	Specify the new inode size for the underlying OST ext3 file system. The default is self-evaluated.

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

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

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

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

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

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

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

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

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

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

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

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

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

Note Only one **MDT** line must be defined for each file system.

storage.conf file

The **/etc/lustre/storage.conf** file stores information about the storage devices available on the cluster and it records which ones are OSTs and which ones are MDTs. It must be located on the Management Node.

File Syntax:

The file is composed of lines with the following syntax:

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

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

Comments are lines beginning with # (hash).

lustre_util command

SYNOPSIS

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

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

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

lustre_util fsck -f < fs_name | all >

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

lustre_util rescue -f < fs_name | all >

lustre_util start -f < fs_name | all >

lustre_util tune_servers -f < fs_name | all >

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

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

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

lustre_util fs_status [-f < fs_name | all >]

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

lustre_util stop -f < fs_name | all >

lustre_util remove -f < fs_name | all >

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

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

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

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

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

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

lustre_util scan_storage

lustre_util check_storage

lustre_util show_tuning

lustre_util show_cfg

lustre_util show_conf

lustre_util list

ACTIONS

set_cfg	Copies **/etc/lustre/lustre.cfg** to OSS and MDS nodes.
install	Checks if file systems can be installed, and then install them.
update	Updates settings of an installed file system that do not require reformatting of **previously** formatted OSTs/MDT (New OSTs, different network type, etc).
fsck	Runs e2fsck on the OST/MDT. The file system must be offline.
chk_dev	Check the devices and their links on I/O nodes.
rescue	Makes a file system usable again by formatting OSTs that are assumed to be NOT correctly formatted.
tune_servers	Set the I/O schedulers of file systems devices regarding **lustre.cfg** content. Apply the server related tunings of **tuning.conf** on OSS/MDS.
start	Makes installed file systems available for mounting.
mount	Mounts file systems on specified nodes.
umount	Unmounts file systems on specified nodes.
fs_status	Updates and prints OSS and MDS status information.
mnt_status	Updates and prints information regarding client nodes.
status	Runs **fs_status** AND **mnt_status**.
stop	Disables file systems.
remove	Removes file systems.
info	Prints information (mdt, ost, path, etc.) about file systems.
short_info	Prints information (mdt, ost, path, etc.) about file systems, but sort OSTs by nodes.
Lfsck	Builds mdt, osts lfsck database and distributes coherency checking of a Lustre file system.
build_mdt_db	Build mdt lfsck database (first step of lfsck).
build_ost_db	Build osts lfsck database (second step of lfsck).
distribute_coherency	Distributes coherency checking of a Lustre file system (third step of lfsck).
scan_storage	Wizard to help configure **storage.conf**
check_storage	Check the consistency of the storage.conf or tables lustre_ost/lustre_mdt.
show_tuning	Display tuning parameter (from tuning.conf).

show_cfg Display lustre.cfg variable.

show_conf Display the lustre_util parameters.

list Prints name of installed file systems or those file systems which failed to be installed.

OPTIONS

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

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

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

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

--mount <[+]opt1,opt2,...>
This allows the mount options to be specified. For example, if +bar is specified for a file system which has foo as a default mount option, mount will be run on the client with -o foo,bar options. If only bar is specified (without +), mount will be run with -o bar options.

-p <rms_partition> Applies the command to the configured nodes of the RMS partition running.

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

-t <time_in_second>
Sets the limit on the amount of time a command is allowed to execute. 0 means no timeout.

-u <user> User name to use to log onto the nodes instead of root.

--fanout Number of SSH connections allowed to run at the same time, default is 128.

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

set_cfg: Distributing /etc/lustre/lustre.cfg

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

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

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

info: Printing Information about the file system

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

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

short_info: Printing Information about a file system

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

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

install: Installing a Lustre file system

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

This command formats the storage devices and performs operations required to install the file system such as loading file systems information into **ldap** database and/or cluster

database. If **-F** is used, no user acknowledge is required. If **-F** is not specified, user must enter "yes" to go on if a file system with the same name is already installed. An **xmf** file is also automatically generated in LUSTRE_CONFIG_DIR for each file system.

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

start: Enabling a Lustre file system

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

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

mount: Mounting Lustre file system

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

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

umount: Unmounting Lustre file system

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

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

stop: Disabling a Lustre file system

```
lustre_util stop -f fs_name
```

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

set_iosched: Set the I/O Schedulers of file system Devices

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

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

remove: Removing a Lustre file system

```
lustre_util remove -f fs_name
```

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

fs_status: Updating file system Status and Printing file system Information regarding OSS and MDS

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

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

```
FILE SYSTEM STATUS

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

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

not installed	**fs** is not installed (it should never be visible).
loaded but not installed	**fs** information is in a database but **lustre_util install** failed.
Formatting	**lustre_util install** is running.
checking	**lustre_util fsck** is running.
installed	fs is correctly installed.
not usable	**fs** is not correctly installed, because some devices could not be formatted or **fsck** failed to repair some devices.

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

offline	**fs** is correctly stopped.

Starting	**lustre_util** start is running.
online	**fs** is started and OSTs/MDT are healthy.
not correctly started	**fs** failed to start, some OSTs or MDT may be offline or unhealthy.
CRITICAL	**fs** started, but for unknown reasons, some OSTs or MDT may be offline or unhealthy.
WARNING	fs is started, but OSS or MDS may not be reachable and their states cannot be checked (IB or elan can work).
stopping	**lustre_util stop** is running.
not correctly stopped	**fs** failed to stop, some OSTs or MDT are still online or are in an unhealthy state.

mnt_status: Updating Clients Status and printing File system Information regarding Clients

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

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

```
CLIENT STATUS

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

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

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

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

fsck: running e2fsck on OSTs and MDT

```
lustre_util fsck -f fs_name
```

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

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

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

This command checks devices information on file systems I/O nodes:

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

lfsck: Builds mdt,osts lfsck database and distributes coherency checking of a Lustre file system

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

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

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

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

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

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

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

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

build_mdt_db, build_ost_db, distribute_coherency : step by step lfsck

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

These options are to be used:

- To restart an **lfsck** operation which has failed, avoiding the need to restart the process from the beginning. **Lustre_util** will provide information regarding which options should be used and when.
- If the directory is not a shared directory and there is a need to copy database files, **lustre_util** will provide information regarding which files should be copied and where.

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

update: Update File systems already Installed

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

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

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

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

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

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

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

Once the model file is updated, run:

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

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

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

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

rescue: Try to make the Installed File system Work again

```
lustre_util rescue -f fs_name -V
```

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

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

This command checks which OSTs have been successfully formatted and formats those that are assumed to be not correctly formatted. Theoretically, the file system should be usable again, **but data may be lost**.

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

```
lustre_util check_storage
```

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

show_tuning: Display the Tuning Parameters

```
lustre_util show_tuning
```

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

show_cfg: Display lustre.cfg Variable

```
lustre_util show_cfg
```

Display lustre.cfg variable.

show_conf: Display lustre_util Configuration

```
lustre_util show_conf
```

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

list: Gives the List of Installed File systems

```
lustre_util list
```

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

lustre_investigate command

SYNOPSIS

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

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

DESCRIPTION

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

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

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

ACTIONS:

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

OPTIONS:

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

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

MGS service

The **mgs** service is located in **/etc/init.d/mgs**.

SYNTAX

mgs {start|stop|restart|status|install|erase|reinstall|clear}

start Start the MGS service using the **mount.lustre** command
The mount point is: **/mnt/srv_lustre/MGS**
This returns 0 if successful or if the MGS service is already running

stop Stop the mgs service using the **umount.lustre** command. This returns 0 if successful or if the **MGS** service has already stopped.

status Status of the **MGS** service resulting from the **mount -t lustre** command. This returns 0 if successful.

restart Restart the **MGS** service using the stop and start target. This returns 0 if successful.

install Installs the **MGS** service if the service is not already installed or running. Creates a folder and file for the loopback device.

- Format using **mkfs.lustre**
- Size for loopback file is **512 MBs**
- Loopback file name is given by **/etc/lustre/lustre.cfg** file :
- target : **LUSTRE_MGS_ABSOLUTE_LOOPBACK_FILENAME**
- default value is : **/home/lustre/run/mgs_loop**

This returns 0 if successful.

erase Erase/Remove the MGS backend using the **rm** remove command on the loopback file. Check if service is stopped before. This returns 0 if successful.

reinstall Reinstall the MGS service using the erase and install target. Free the loopback reservation using the **losetup –d** command. This returns 0 if successful.

clear Clean the loopback map using **losetup -a** and **losetup –d** commands. This returns 0 if successful.

Run this command to display the **mgs** man page:

```
service mgs help
```

lustre_<table>_dba commands

SYNOPSIS

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

DESCRIPTION

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

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

These utilities are useful for checking the correctness of **ClusterDB** contents according to the last configuration updates. They allow the further adjustment of values in the instance of mistakes or failures of the Lustre management utilities thus avoiding a full repeat of the operation. They can also be used to force the Lustre services behaviour for some precise and controlled cases.
As these act on the global cluster configuration information, they must be used very carefully. The changes they allow may introduce fatal inconsistencies in the Lustre ClusterDB information.

ACTIONS

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

OPTIONS

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

Glossary

A

API

Application Programmer Interface

B

BSM

Bull System Manager

G

Ganglia

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

L

LDAP

Lightweight Directory Access Protocol

LOV

Logical Object Volume

M

MDS

MetaData Server

MDT

MetaData Target

O

OSS

Object Storage Server

OST

Object Storage Target

Index

I

L

M

N

O

P

Q

R

S

T

V

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

REFERENCE
86 A2 46FD 00