CHAPTER 8

Failover

This chapter describes failover in a Lustre system and includes the following sections:

8.1 What is Failover?

A computer system is "highly available" when the services it provides are available with minimal downtime. In a highly-available system, if a failure condition occurs, such as the loss of a server or a network or software fault, the system's services continue without interruption. Generally, we measure availability by the percentage of time the system is required to be available.

Availability is accomplished by replicating hardware and/or software so that when a primary server fails or is unavailable, a standby server can be switched into its place to run applications and associated resources. This process, called *failover*, should be automatic and, in most cases, completely application-transparent.

A failover hardware setup requires a pair of servers with a shared resource (typically a physical storage device, which may be based on SAN, NAS, hardware RAID, SCSI or FC technology). The method of sharing storage should be essentially transparent at the device level in that the same physical logical unit number (LUN) should be visible from both servers. To ensure high availability at the physical storage level, we encourage the use of RAID arrays to protect against drive-level failures.

8.1.1 Failover Capabilities

To establish a highly-available Lustre file system, power management software or hardware and high availability (HA) software are used to provide the following failover capabilities:

- **Resource fencing** - Protects physical storage from simultaneous access by two nodes.
- **Resource management** - Starts and stops the Lustre resources as a part of failover, maintains the cluster state, and carries out other resource management tasks.
- **Health monitoring** - Verifies the availability of hardware and network resources and responds to health indications provided by Lustre.

Although these capabilities can be provided by a variety of software and/or hardware solutions, the currently supported solution for Lustre is Heartbeat. For information about accessing the latest version of Heartbeat, see:

www.sun.com/software/products/hpcsoftware/getit.jsp

HA software is responsible for detecting failure of the primary Lustre server node and controlling the failover. Lustre works with any HA software that supports resource (I/O) fencing. For proper resource fencing, the HA software must be able to completely power off the failed server or disconnect it from the shared storage device. If two active nodes have access to the same storage device, data may be severely corrupted.

8.1.2 Types of Failover Configurations

Nodes in a cluster can be configured for failover in several ways. They are often configured in pairs (for example, two OSTs attached to a shared storage device), but other failover configurations are also possible. Failover configurations include:

- **Active/passive pair** - In this configuration, the active node provides resources and serves data, while the passive node is usually standing by idle. If the active node fails, the passive node takes over and becomes active.
- **Active/active pair** - In this configuration, both nodes are active, each providing a subset of resources. In case of a failure, the second node takes over resources from the failed node.

The active/passive configuration is seldom used for OST servers as it doubles hardware costs without improving performance. On the other hand, an active/active cluster configuration can improve performance by serving and providing arbitrary failover protection to a number of OSTs.

8.2 Failover Functionality in Lustre

The failover functionality provided in Lustre supports the following failover scenario. When a client attempts to do I/O to a failed Lustre target, it continues to try until it receives an answer from any of the configured failover nodes for the Lustre target. A user-space application does not detect anything unusual, except that the I/O may take longer than usual to complete.

Lustre failover requires two nodes configured as a failover pair, which must share one or more storage devices. Lustre can be configured to provide MDT or OST failover.

- For MDT failover, two MDSs are configured to serve the same MDT. Only one MDS node can serve an MDT at a time.
- For OST failover, multiple OSS nodes are configured to be able to serve the same OST. However, only one OSS node can serve the OST at a time. An OST can be moved between OSS nodes that have access to the same storage device using `umount/mount` commands.

To add a failover partner to a Lustre configuration, the `--failnode` option is used. This can be done at creation time (using `mkfs.lustre`) or later when the Lustre system is active (using `tunefs.lustre`). For explanations of these utilities, see mkfs.lustre and tunefs.lustre.

For a failover example, see More Complicated Configurations.

Note – Failover is supported in Lustre only at the file system level. In a complete failover solution, support for system-level components, such as node failure detection or power control, is provided by a third party tool.

Caution – OST failover functionality does not protect against corruption caused by a disk failure. If the storage media (i.e., physical disk) used for an OST fails, Lustre cannot recover it. We strongly recommend that some form of RAID be used for OSTs. Lustre functionality assumes that the storage is reliable, so it adds no extra reliability features.

8.2.1 MDT Failover Configuration (Active/Passive)

Two MDSs are usually configured as an active/passive failover pair. Note that both nodes must have access to shared storage for the MDT(s) and the MGS. The primary (active) MDS manages the Lustre system metadata resources. If the primary MDS fails, the secondary (passive) MDS takes over these resources and serves the MDTs and the MGS.

Note – In an environment with multiple file systems, the MDSs can be configured in a quasi active/active configuration, with each MDS managing metadata for a subset of the Lustre file system.

8.2.2 OST Failover Configuration (Active/Active)

OSTs are usually configured in a load-balanced, active/active failover configuration. A failover cluster is built from two OSSs.

Note – OSSs configured as a failover pair must have shared disks/RAID.

In an active configuration, 50% of the available OSTs are assigned to one OSS and the remaining OSTs are assigned to the other OSS. Each OSS serves as the primary node for half the OSTs and as a failover node for the remaining OSTs.

In this mode, if one OSS fails, the other OSS takes over all of the failed OSTs. The clients attempt to connect to each OSS serving the OST, until one of them responds. Data on the OST is written synchronously, and the clients replay transactions that were in progress and uncommitted to disk before the OST failure.

8.2.3 Lustre Failover and MMP

The failover functionality in Lustre is supported by the multiple mount protection (MMP) feature, which protects the file system from being mounted simultaneously to more than one node. This feature is important in a shared storage environment (for example, when a failover pair of OSTs share a partition).

Lustre's backend file system, `ldiskfs`, supports the MMP mechanism. A block in the file system is updated by a `kmmpd` daemon at one second intervals, and a sequence number is written in this block. If the file system is cleanly unmounted, then a special "clean" sequence is written to this block. When mounting the file system, `ldiskfs` checks if the MMP block has a clean sequence or not.

Even if the MMP block has a clean sequence, `ldiskfs` waits for some interval to guard against the following situations:

- If I/O traffic is heavy, it may take longer for the MMP block to be updated.
- If another node is trying to mount the same file system, a "race" condition may occur.

With MMP enabled, mounting a clean file system takes at least 10 seconds. If the file system was not cleanly unmounted, then the file system mount may require additional time.

Note – The MMP feature is only supported on Linux kernel versions >= 2.6.9.

Note – On a new Lustre file system, MMP is automatically enabled by `mkfs.lustre` at format time if failover is being used and the kernel and e2fsprogs version support it. On an existing file system, a Lustre administrator can manually enable MMP when the file system is unmounted.

8.2.3.1 Working with MMP

Use the following commands to determine whether MMP is running in Lustre and to enable or disable the MMP feature.

To determine if MMP is enabled, run:

```
dumpe2fs -h <device>|grep features
```

Here is a sample command:

```
dumpe2fs -h /dev/sdc | grep features
Filesystem features: has_journal ext_attr resize_inode dir_index
filetype extent mmp sparse_super large_file uninit_bg
```

To manually disable MMP, run:

```
tune2fs -O ^mmp <device>
```

To manually enable MMP, run:

```
tune2fs -O mmp <device>
```

When MMP is enabled, if `ldiskfs` detects multiple mount attempts after the file system is mounted, it blocks these later mount attempts and reports the time when the MMP block was last updated, the node name, and the device name of the node where the file system is currently mounted.

8.3 Configuring and Using Heartbeat with Lustre Failover

This section describes how to configure Lustre failover using the Heartbeat cluster infrastructure daemon.

8.3.1 Creating a Failover Environment

Lustre provides failover mechanisms only at the file system level. No failover support is provided for system-level components, such as node failure detection or power control, as would typically be provided in a complete failover solution. Additional tools are also needed to provide resource fencing, control and monitoring.

8.3.1.1 Power Management Software

Lustre failover requires power control and management capability to verify that a failed node is shut down before I/O is directed to the failover node. This avoids double-mounting the two nodes, and the risk of unrecoverable data corruption. A variety of power management tools will work, but two packages that are commonly used with Lustre are STONITH and PowerMan.

Shoot The Other Node In The HEAD (STONITH), is a set of power management tools provided with the Linux-HA package. STONITH has native support for many power control devices and is extensible. It uses expect scripts to automate control.

PowerMan, available from the Lawrence Livermore National Laboratory (LLNL), is used to control remote power control (RPC) devices from a central location. PowerMan provides native support for several RPC varieties and expect-like configuration simplifies the addition of new devices.

The latest versions of PowerMan are available at:

sourceforge.net/projects/powerman

For more information about PowerMan, go to:

computing.llnl.gov/linux/powerman.html

8.3.1.2 Power Equipment

Lustre failover also requires the use of RPC devices, which come in different configurations. Lustre server nodes may be equipped with some kind of service processor that allows remote power control. If a Lustre server node is not equipped with a service processor, then a multi-port, Ethernet-addressable RPC may be used as an alternative. For recommended products, refer to the list of supported RPC devices on the PowerMan website.

computing.llnl.gov/linux/powerman.html

8.3.2 Setting up the Heartbeat Software

Lustre must be combined with high-availability (HA) software to enable a complete Lustre failover solution. Lustre can be used with different HA packages, including Heartbeat, the Linux-HA software.

For current information about Heartbeat, see linux-ha.org/wiki.

The Heartbeat package is one of the core components of the Linux-HA project. Heartbeat is highly-portable and runs on every known Linux platform, as well as FreeBSD and Solaris.

This section describes how to install Heartbeat v2 and configure it with and without STONITH. Because Heartbeat v1 has simpler configuration files, which can be used with both Heartbeat v1 and v2, the configuration examples show how to configure Heartbeat using Heartbeat v1 configuration files.

Heartbeat v2 adds monitoring and supports more complex cluster topologies, and the Heartbeat v2 configuration is stored as an XML file. To support users with Heartbeat v2, this section also includes a procedure to migrate Heartbeat v1 configuration files to v2.

8.3.2.1 Installing Heartbeat

1. **Install Lustre (see** Installing Lustre**).**

2. **Install the Heartbeat packages.**

   Heartbeat v2 requires several packages. This example uses Heartbeat v. 2.1.4. The required Heartbeat packages are, in order:

   - **heartbeat-stonith** -> heartbeat-stonith-2.1.4-1.x86_64.rpm
   - **heartbeat-pils** -> heartbeat-pils-2.1.4-1.x86_64.rpm
   - **heartbeat** -> heartbeat-2.1.4-1.x86_64.rpm

   You can download the Heartbeat packages and guides covering basic setup and testing here:

   www.sun.com/software/products/hpcsoftware/getit.jsp

   Heartbeat packages are available for many Linux distributions. Additionally, Heartbeat has some dependencies on other packages. It is recommended that you use a package manager like yum, yast or aptitude to install the Heartbeat packages and resolve their package dependencies.

8.3.2.2 Configuring Heartbeat

This section describes Heartbeat configuration and provides a worked example to illustrate the configuration steps.

Note – Depending on the particular packaging, Heartbeat files may be located in a different directory or path than indicated in the following procedures.

For remote power control, both OSS nodes are equipped with a service processor (SP). The SPs are accessible over the network via their hostnames. Individual node parameters are listed below.

Parameters	Value	Description
First OSS node		
OSS node	`oss01`	First OSS node in the Lustre file system
OST	`ost01`	First OST in the Lustre file system
block device	`/dev/sda`	Block device for the first OSS node (`oss01`)
mount point	`/mnt/ost1`	Mount point for the oss01 block device (`/dev/sda`) on the `oss01` node
hostname	`oss01sp`	Hostname for the first OSS node's SP
Second OSS node		
OSS node	`oss02`	Second OSS node in the Lustre file system
OST	`ost02`	Second OST in the Lustre file system
block device	`/dev/sdb`	Block device for the second OSS node (`oss02`)
mount point	`/mnt/ost02`	Mount point for the ost02 block device (`/dev/sdb`) on the `oss02` node
hostname	`oss02sp`	Hostname for the second OSS node's SP

Configuring Heartbeat without STONITH

Note – This procedure describes Heartbeat configuration using a v1 configuration file, which can be used with both Heartbeat v1 and v2. See (Optional) Migrating a Heartbeat Configuration (v1 to v2) for an optional procedure to convert the v1 configuration file to an XML-formatted v2 configuration file.

Note – Depending on the particular packaging, Heartbeat files may be located in a different directory or path than indicated in the following procedure. For example, they may be located in `/etc/ha.d/` or `/var/lib/heartbeat`.

To configure Heartbeat without STONITH:

1. **Create (or edit) the Heartbeat configuration file, /etc/ha.d/ha.cf.**

   This file must be identical on both nodes.

   In this example configuration (without STONITH configuration), the `/etc/ha.d/ha.cf` file looks like this:

```
# log file settings
# write debug output to /var/log/ha-debug
debugfile /var/log/ha-debug
# write log messages to /var/log/ha-log
logfile /var/log/ha-log
# use syslog to write to logfiles
logfacility local0

# set some time-outs. these values are only recommendations, which
depend e.g. on the OSS load
# send keep-alive packages every 2 seconds
keepalive 2
# wait 90 seconds before declaring a node dead
deadtime 90
# write a warning to the logfile after 30 seconds without an answer
from the failover node
warntime 30
# wait for 120 seconds before declaring a node dead after heartbeat
is brought up
initdead 120

# define communication channels
# use port 12345 to communicate with fail-over node
udpport 12345
# use network interfaces eth0 and ib0 to detect a failed node
bcast eth0 ib0

# Use manual failback
auto_failback off

# node names in this failover-pair. These names must match the
output of `hostname`
node oss01
node oss02
```

2. **Define the resources that will be controlled by Heartbeat by editing the /etc/ha/d/haresources file.**

   This file must be identical on both nodes.

   In this example configuration, the /etc/ha.d/haresources file looks like this:

   ```
   oss01 Filesystem::/dev/sda::/mnt/ost01::lustre
   oss02 Filesystem::/dev/sdb::/mnt/ost02::lustre
   ```

   The resource definition file tells Heartbeat that one file system resource is associated with oss01 and oss02. Each resource is defined on separate lines.

   The file system resource script takes three inputs separated by "::". The first parameter is the device name, the second is the mount point and the third is the file system type.

   Depending on the configuration, a resource can be more complex, e.g., software RAID needs to be assembled before the file system can be mounted. In this case, an haresources file may look like this:

   ```
   oss01 Raid1::/etc/mdadm.conf.oss::/dev/md1
   Filesystem::/dev/md1::/mnt/ost01::lustre
   oss02 Raid1::/etc/mdadm.conf.oss::/dev/md2
   Filesystem::/dev/md2::/mnt/ost02::lustre
   ```

   When a resource group is started by Heartbeat, the resources start from left to right. In this example, the RAID is assembled first, and the file system is mounted second. If the resource group is stopped, then the file system is unmounted first and the RAID is stopped second.

   Other resource scripts can be found in the /etc/ha.d/resource.d/ folder.

3. **Create the /etc/ha.d/authkeys file and fix its permissions.**

   This file must be identical on both nodes.

   In this example configuration, the authkeys file looks like this:

   ```
   auth 1
   1 sha1 PutYourSuperSecretKeyHere
   ```

   Make sure that the permissions for this files are set to 0600, by running `chmod 0600 /etc/ha.d/authkeys` on both nodes.

4. **Test the Heartbeat configuration.**

   Run the following command on both nodes:

   ```
   service heartbeat start
   ```

   Check the log files on both nodes to find any problems and fix them.

   After the initial deadtime interval, you should see the nodes discover each other's state and start the Lustre resources associated with them.

Configuring Heartbeat with STONITH

STONITH automates the process of power control and management. Expect scripts are dependent on the exact set of commands provided by each hardware vendor. As a result, any change in the power control hardware or firmware requires that STONITH be adjusted.

Note – This procedure describes configuring Heartbeat using a v1 configuration file, which can be used with both Heartbeat v1 and v2. See (Optional) Migrating a Heartbeat Configuration (v1 to v2) for an optional procedure to convert the v1 configuration file to an XML-formatted v2 configuration file.

Note – Depending on the particular packaging, Heartbeat files may be located in a different directory or path than indicated in the following procedure. For example, they may be located in `/etc/ha.d/` or `/var/lib/heartbeat`.

The heartbeat-stonith package comes with a number of pre-defined STONITH scripts for different power control hardware. Additionally, Heartbeat can be configured to run an external script. Heartbeat can be configured in two STONITH modes:

- One STONITH command for all nodes found in `ha.cf`:

  ```
  stonith <type> <config file>
  ```

- One STONITH command per-node:

  ```
  stonith_host <hostfrom> <stonith_type> <params...>
  ```

You can use an external script to kill each node, e.g.:

```
stonith_host oss01 external foo /etc/ha.d/reset-nodeB
stonith_host oss02 external foo /etc/ha.d/reset-nodeA
```

To get the proper STONITH syntax, run:

```
$ stonith -L
```

The above command lists supported models.

To list required parameters and specify the configuration filename, run:

```
$ stonith -l -t <model>
```

To attempt a test, run:

```
$ stonith -l -t <model> <fake host name>
```

To test STONITH, use a real hostname. To work with Heartbeat correctly, the external STONITH scripts should take the parameters {start | stop | status} and return 0 or 1.

To add STONITH functionality (using an ipmi service processor) to the configuration example, add the following lines to the /etc/ha.d/ha.cf configuration file:

```
# define how a node can be powered off in case of a failure. more
details below
stonith_host oss01 external/ipmi oss02 oss02sp root changeme lanplus
stonith_host oss02 external/ipmi oss01 oss01sp root changeme lanplus
```

STONITH is only invoked if one of the failover nodes is no longer responding to Heartbeat messages and the cluster does stop resources in an orderly manner. If two cluster nodes can communicate, they usually shut down properly. This means that many tests do not produce a STONITH, for example:

- Calling init 0, shutdown, or reboot on a node will cause no STONITH
- Stopping Heartbeat on a node stops the resources cleanly and fails them over to the other node without invoking STONITH.

8.3.2.3 (Optional) Migrating a Heartbeat Configuration (v1 to v2)

Heartbeat includes a script that enables v1 configuration files to be migrated to v2 XML configuration files. The script reads the v1 configuration files (ha.cf and haresources), and then writes an XML file to STDOUT. The script is

```
$ /usr/lib/heartbeat/haresources2cib.py
```

or

```
$ /usr/lib64/heartbeat/haresources2cib.py
```

To redirect the script output after the cib.xml file has been generated, it is recommended that you check the XML file and change some parameters, such as resource-stickiness and timeouts, to more appropriate values. For example:

$ /usr/lib64/heartbeat/haresources2cib.py > cib.xml

Then the cib.xml file should than be copied to /var/lib/heartbeat/crm/cib.xml on both failover nodes.

To test the new configuration, start Heartbeat on both nodes and check the log files.

Note – If a Heartbeat v2 configuration file is available on the system, it is not necessary to remove the v1 configuration files, as they are ignored.

8.3.3 Working with Heartbeat

After Lustre and Heartbeat are correctly configured, the following commands can be used to control Heartbeat.

8.3.3.1 Starting Heartbeat

To start Heartbeat, run this command on both failover nodes:

```
service heartbeat start
```

After a node fails, start Heartbeat manually and analyze the cause of the problem before taking over the failed resources. You should NOT start Heartbeat automatically after a node failure.

8.3.3.2 Switching Resources Between Nodes

Depending on whether Heartbeat v1 or v2 configuration files are being used, there are different ways to switch resources between nodes.

For Heartbeat v1 configuration files, two scripts are provided (`hb_takeover` and `hb_standby`), that make it easy to switch resources between failover nodes. Depending on your system, these scripts are located in `/usr/lib/heartbeat/` or `/usr/lib64/heartbeat/`.

The `hb_takeover` and `hb_standby` scripts take the following arguments:

- **all** -- take/fail over all resources
- **foreign** -- take/fail over foreign resources
- **local** -- take/fail over local resources only
- **failback** -- fail/take over foreign resources

Performing an `hb_takeover` on the current node is equivalent to performing an `hb_standby` on the other node.

For Heartbeat v2 configuration files, the `crm_resource` command is used to interact with Heartbeat's Cluster Resource Manager and switch resources between nodes. For more information on crm_resource, see:

linux.die.net/man/8/crm_resource

To switch resources between nodes:

1. **Generate a complete list of resources known to the Heartbeat cluster resource manager. Run:**

   ```
   crm_resource --list
   ```

2. **From the list, identify the group name for the resource to fail over.**

3. **Determine if and where the specified resource is running. Run:**

   ```
   crm_resource -W -r <resource_name>
   ```

4. **Migrate the resource to the host. Run:**

   ```
   crm_resource -M -r <resource_name> -H <target_host_name>
   ```

5. **To un-migrate a resource, run:**

   ```
   crm_resource -U -r <resource_name>
   ```