# Creating a High-Availability Lustre* Storage Solution over a ZFS File System

Partner Guide

High Performance Data Division

Software version: Intel® EE for Lustre* Software 2.4.n.n

December 21, 2015

World Wide Web: http://www.intel.com

# Disclaimer and legal information

# Contents

About this Document ..... iii
Document Purpose ..... iii
Intended Audience ..... iii
Conventions Used ..... iii
Related Documentation ..... iii

Introduction - Lustre* over ZFS ..... 1
Requirements ..... 1
Supported Configurations ..... 1
Environment Used in this Example ..... 2
Software Stack ..... 2

Installation ..... 3
Mellanox OFED Installation ..... 3
Lustre* and ZFS Installation ..... 4
Lustre Network Configuration ..... 5

Implementing High Availability ..... 5
HA Framework Installation on Metadata Servers ..... 5
ZFS Configuration ..... 6
Lustre Formatting and Target Configuration ..... 7
HA Framework Installation on Object Storage Servers ..... 8
ZFS Configuration ..... 9
Lustre Formatting and Target Configuration ..... 11
Mount the File System on a Client ..... 12
Compression ..... 12

Detecting and Monitoring the File System at the Dashboard ..... 12
Add Servers ..... 12
Detect the New File System ..... 12
Mounting the Lustre File System on a Client via the Dashboard ..... 13

Troubleshooting ..... 13
Procedure for Updating ZFS or Mellanox OFED ..... 13
Mellanox OFED not installing ..... 14

Appendix: LustreZFS ..... 14

# About this Document

## Document Purpose

Using an example configuration, this document describes how to create a ZFS file system and use that as the backend for a Lustre* file system monitored by Intel® Manager for Lustre* software. The document then describes how to use pacemaker and corosync to configure high-availability. The system configuration described herein is an example only; your configuration will likely be different.

## Intended Audience

The intended audience for this guide are partners who are designing storage solutions based on Intel® Enterprise Edition for Lustre* Software. Readers are assumed to be full-time Linux system administrators or equivalent, who have:

- experience administering file systems and are familiar with storage components such as block storage, SAN, and LVM
- experience or knowledge about Lustre* installation and setup
- proficiency in setting up, administering and maintaining networks
- familiarity in setting up and administering ZFS file systems
- familiarity with corosync and pacemaker configuration and PCS.

## Conventions Used

Conventions used in this document include:

- # preceding a command indicates the command is to be entered as root
- $ indicates a command is to be entered as a user
- *<variable_name>* indicates the placeholder text that appears between the angle brackets is to be replaced with an appropriate value

## Related Documentation

The following documents are pertinent to Intel® Enterprise Edition for Lustre* software. This list is not all-inclusive.

- *Intel® Enterprise Edition for Lustre* Software Help*. Also available as a user guide.
- *Intel® Enterprise Edition for Lustre* Software Partner Installation Guide*
- *Creating a Scalable File Service for Windows Networks using Intel® EE for Lustre* Software*
- *Hierarchical Storage Management Configuration Guide*

- *Intel® EE for Lustre* Hierarchical Storage Management Framework White Paper*
- *Architecting a High-Performance Storage System White Paper*

# Introduction – Lustre* over ZFS

ZFS is an attractive technology that can meet the requirements of the next generation of HPC storage solutions. In recent years, the use of ZFS as a backend file system for Lustre* storage targets has grown in popularity. There are many additional features to ZFS when compared to the current common file system, ldiskfs. These features include mirroring, striping with parity, and compression, to name a few.

Intel® Enterprise Edition for Lustre* software supports ZFS as a backend file system replacement for ldiskfs. The Intel® Manager for Lustre* dashboard, the GUI-based configuration and management tool of Intel® EE for Lustre* software, is able to configure and manage high-availability Lustre storage solutions. However, this release, version 2.4.*n.n* of Intel® EE for Lustre* software supports ZFS file systems in *monitor mode only*, and thus the dashboard cannot be used to manage high-availability.

Using an example configuration, this document describes how to create a ZFS file system and use that as the backend for a Lustre file system monitored by Intel® Manager for Lustre* software. The document then describes how to use pacemaker and corosync to configure high-availability. The system configuration described herein is an example only; your configuration will likely be different. For "production" Lustre file systems, Intel® encourages high-availability configuration.

## Requirements

The Lustre file system deployed with a ZFS file system backend must be created outside of Intel® Manager for Lustre* software. The manager software will not provision a Lustre file system with a ZFS backend in this release, version 2.4.n.n of Intel® EE for Lustre* software. Using procedures provided and other sources referenced herein, the partner will be responsible for installing and manually configuring ZFS and Lustre.

## Supported Configurations

In this implementation using ZFS as the file system backend, we are employing the high-availability framework included in the Red Hat distribution of Linux. For more information: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Configuring_the_Red_Hat_High_Availability_Add-On_with_Pacemaker/index.html

**Note**: Before using the Red Hat or RHEL software referenced herein, please refer to Red Hat's website for more information, including without limitation, information regarding the mitigation of potential security vulnerabilities in the Red Hat software.

ZFS is a file system and logical volume manager designed by Sun Microsystems. The features of ZFS include protection against data corruption, support for high storage capacities, efficient

data compression, integration of the concepts of file system and volume management, snapshots and copy-on-write clones, continuous integrity checking and automatic repair, RAID-Z, and native NFSv4 ACLs.

Not all of the above features are supported by Lustre, and some of the features supported by Lustre may not be supported by Intel EE for Lustre* software. These ZFS configurations are supported by Intel® High Performance Data Division:

- ZFS in high availability mode using Pacemaker/Corosync
- Mirror + Stripe zpool (recommended for MDT and MGT)
- Compression
- L2ARC
- RAID-Z
- RAID-Z2 (recommended for OSTs)
- RAID-Z3 (recommended for very high-capacity drives)

## Environment Used in this Example

Following are the file system servers, with their roles and addresses, that we will use in this example.

| Node Name | Role | LNET – ib0 | Management and Ring0 – eth0 | Ring1 – eth1 | IPMI device |
|---|---|---|---|---|---|
| Kzmds01 | MDS | 192.168.211.211 | 192.168.210.211 | 192.168.214.211 | 192.168.212.211 |
| Kzmds02 | MDS | 192.168.211.212 | 192.168.210.212 | 192.168.214.212 | 192.168.212.212 |
| Kzoss01 | OSS | 192.168.211.213 | 192.168.210.213 | 192.168.214.213 | 192.168.212.213 |
| Kzoss02 | OSS | 192.168.211.214 | 192.168.210.214 | 192.168.214.214 | 192.168.212.214 |

## Software Stack

This guide references the software listed in the following table.

| Software | Version |
|---|---|
| Red Hat Enterprise Linux or CentOS Linux | 6.7 |
| Pacemaker | 1.1.12 |

| | |
|---|---|
| Corosync | 1.4.7 |
| PCS (Pacemaker/Corosync CLI) | 0.9.139 |
| CMAN (Cluster Manager) | 3.0.12.1 |
| OFED (optional) | Either kernel built-in OFED drivers, or Mellanox OFED 2.4 or 3.0 |
| ZFS | Included with Intel® EE for Lustre* software version 2.4.0.0 and later. |
| Lustre | Included with Intel® EE for Lustre* software version 2.4.0.0 and later. |
| Fence agents | 4.0.15 |

# Installation

## Mellanox OFED Installation

Mellanox provides an OFED package that supports RDMA and kernel-bypass APIs called "OFED verbs" over Infiniband and Ethernet. This software stack supports Ethernet and Infiniband connectivity on the same card. As of this writing, MLNX_OFED versions 2.4 and 3.0 are supported. Mellanox recommends the most current release, and as of this writing, version MLNX_OFED 3.0 is the most current.

If you're using Mellanox network adapter cards, download and install the Mellanox OFED on *each server before installing Lustre and ZFS*. In this way, the Lustre installation will compile against the Mellanox OFED on each server. Download and install the OFED from the following website, which is active as of this writing:

http://www.mellanox.com/page/mlnx_ofed_matrix?mtag=linux_sw_drivers

If for some reason the installation utility fails, perform these steps:

1. Ensure you have the correct operating system and version. Mellanox OFED for Red Hat version 6.6 does not support systems running Red Hat version 6.7.
2. Force the install. Enter the following command:

```
./mlnxofedinstall -k $(head -1 .supported_kernels)
```

If you are adding Mellanox OFED after having installed Lustre software, perform the following steps. Run these steps on each server (MDS/MGS/OSS):

1. Remove existing Lustre*: dkms remove lustre/2.5.39 –all
2. Run the Mellanox OFED installer as normal.
3. Re-install Lustre*: dkms install lustre/2.5.39

## Lustre* and ZFS Installation

**Note**: When creating a standard Lustre file system (*without* ZFS for the backend), installing Intel® EE for Lustre* on each server installs a modified Linux kernel. This is required to support certain features provided by Intel® EE for Lustre* software. However, when creating this ZFS-based Lustre file system, the unmodified Linux kernel included with RHEL or CentOS must remain. Accordingly, **do *not* perform** the installation as described in the Intel® EE for Lustre* Partner Installation Guide. Perform the following procedure to install ZFS and Lustre.

**Note**: If you are installing ZFS on servers that have been previously configured as Lustre file system servers, first re-provision all servers with Red Hat 6.7. Then install ZFS as described next. Any existing file system data will be lost.

**Note**: References herein to the *manager server* is that server running the Intel® Manager for Lustre* software dashboard.

1. Download the installation archive to a directory on the manager server (e.g. /tmp).
2. Unpack the installation archive using tar: ieel-2.4.0.0.tar.gz (version is an example)

   ```
   $cd /tmp
   $tar -xzvf ieel-2.4.0.0.tar.gz (version is an example)
   ```

3. Create the ZFS installer.

   ```
   $cd ieel-2.4.0.0 (version is an example)
   $./create_installer zfs
   ```

4. Copy the installer to all servers that will comprise the ZFS file system.

   ```
   $scp lustre-zfs-installer.tar.gz storage-server:/tmp
   ```

5. On each planned ZFS server, log in as root, untar the ZFS installer, and install ZFS using these steps. Note that for each server, the installation may take thirty minutes or longer. This process can also be scripted.

   ```
   $ssh root@storage-server "cd /tmp
   tar xzvf lustre-zfs-installer.tar.gz
   cd lustre-zfs
   ./install"
   ```

### Lustre Network Configuration

Enter the following command on all Lustre servers and clients (with appropriate network parameters):

```
# cat /etc/modprobe.d/lustre.conf
options lnet networks= o2ib0(ib0)
```

## Implementing High Availability

### HA Framework Installation on Metadata Servers

1. On each metadata server:

```
#yum install pcs fence-agents pacemaker cman
```

2. Change the hacluster password on each metadata server:

```
#passwd hacluster
```

3. Start the PCSD daemon and enable it at boot:

```
#service pcsd start
#chkconfig pcsd on
```

4. Disable corosync at start on each metadata server:

```
# service corosync stop
# chkconfig corosync off
```

5. Perform the remainder of these steps at a single server. Insert the credentials for the hacluster user:

```
# pcs cluster auth –u hacluster kzmds01 kzmds02
```

6. Create the cluster:

```
# pcs cluster setup --start --name MDS01-02 kzmds01 kzmds02
```

7. Enable the autostart of the cluster:

```
# pcs cluster enable –all
```

8. Create the stonith resources:

```
# pcs stonith create kzmds01-ipmi fence_ipmilan
ipaddr="192.168.212.211" lanplus=true passwd="XXXXX" login="XXXX"
pcmk_host_list="kzmds01"

# pcs stonith create kzmds02-ipmi fence_ipmilan
ipaddr="192.168.212.212" lanplus=true passwd="XXXXX" login="XXXX"
pcmk_host_list="kzmds02"
```

9. Add an additional network to increase the reliability:

```
# ccs -f /etc/cluster/cluster.conf --addalt kzmds01 192.168.214.211
port=7909 mcast=239.192.2.2 ttl=2

# ccs -f /etc/cluster/cluster.conf --addalt kzmds02 192.168.214.212
port=7909 mcast=239.192.2.2 ttl=2

# ccs -f /etc/cluster/cluster.conf --settotem rrp_mode="active"
```

10. Sync the cluster:

```
#pcs cluster sync
```

11. Reboot both the servers.

Storage servers can become overloaded, which may cause failover to occur. It may be necessary to increase the token value of the cluster. If no token value is specified in the cluster configuration, the default is 10000 ms, or 10 seconds. To use a value other than the default, add or edit the totem line in `/etc/cluster/cluster.conf` as a child of the <cluster> element. Please refer to the Red Hat Knowledgebase for more information.

## ZFS Configuration

In a ZFS configuration we have access to all the individual disks available in the JBOD attached to the cluster. Each individual disk is reachable thought two or more paths for high availability. You can use the device multi-mapper technology included in Red Hat Linux to create the ZFS pool and maintain a high-availability solution.

A ZFS pool will represent a Lustre Target. Each of the entities in a pool are VDEVs. A VDEV can be a single storage device organized in a RAID group, as in the example below.

```
# zpool create mgt mirror mpatha mpathb mirror mpathc mpathd
# zpool create mdt0000 mirror mpathe mpathf mirror mpathg mpathh
```

We need to be sure that each zpool is available on both nodes of the cluster. Please perform the following procedure for the mgt and mdt0000 pools:

```
# zpool export mdt0000
```

Now go to the other node:

```
# partprobe
# zpool import -o cachefile=none mdt0000
```

We suggest creating the MDT in mirror to get best performance. In this example we will create one ZFS pool for each server:

- mgt on kzmds01
- mdt0000 on kzmds02

## Lustre Formatting and Target Configuration

1. Manually import the MGT on kzmds01 and format the zfs pool as follows:

```
# mkfs.lustre --mgs --backfstype=zfs --fsname=lustrefs --
servicenode=192.168.211.211@o2ib0 --
servicenode=192.168.211.212@o2ib0 --reformat mgt/mgt
```

2. Manually import the MDT on kzmds02 and format the zfs pool as follows:

```
# mkfs.lustre --mdt --backfstype=zfs --fsname=lustrefs --index=0 --
mgsnid=192.168.211.211@o2ib0 --mgsnid=192.168.211.212@o2ib0 --
servicenode=192.168.211.212@o2ib0 --
servicenode=192.168.211.211@o2ib0 --reformat mdt0000/mdt0000
```

3. Create the mount points on all the servers:

```
# mkdir -p /lustrefs/mgt
# mkdir -p /lustrefs/mdt0000
```

4. Mount manually for the first time to verify each node:

`# mount -t lustre mgt/mgt /lustrefs/mgt` (on kzmds01)

`# mount -t lustre mdt0000/mdt0000 /lustrefs/mdt0000` (on kzmds02)

5. Then unmount.
6. Add the LustreZFS script available in the Appendix *to each host* at the location `/usr/lib/ocf/resource.d/heartbeat/`. Set the script's filename to LustreZFS and set the permissions for the script to 755.
7. Create the resources in pacemaker:

```
# pcs resource create lustrefs-MGS ocf:heartbeat:LustreZFS
pool="mgt" volume="mgt" mountpoint="/lustrefs/mgt”
```

```
# pcs resource create lustrefs-MDT0000 ocf:heartbeat:LustreZFS
pool="mdt0000" volume="mdt0000" mountpoint="/lustrefs/mdt0000"
```

... where `pool` is the name of the ZFS pool created previously, `volume` is the name of the Lustre volume in the pool and `mountpoint` is the directory where the volume will be mounted.

8. To configure a preferred server for this resource, please use the primary and secondary node selected during the format phase:

```
# pcs constraint location lustrefs-MGS prefers kzmds01=20
# pcs constraint location lustrefs-MGS prefers kzmds02=10
# pcs constraint location lustrefs-MDT0000 prefers kzmds02=20
# pcs constraint location lustrefs-MDT0000 prefers kzmds01=10
```

9. Verify the final configuration:

```
# pcs config --full
```

## HA Framework Installation on Object Storage Servers

1. Enter this command on all the Object Storage Servers in the pair:

```
# yum install pcs fence-agents pacemaker cman
```

2. Change the hacluster password on all the servers:

```
# passwd hacluster
```

3. Start the PCSD daemon and enable it at boot:

```
# service pcsd start
# chkconfig pcsd on
```

4. Disable corosync at start from all the servers:

```
# service corosync stop
# chkconfig corosync off
```

5. Perform the remainder of these steps at a single server. Add the credentials for the hacluster user:

```
# pcs cluster auth –u hacluster kzoss01 kzoss02
```

6. Create the cluster:

```
# pcs cluster setup --start --name OSS01-02 kzoss01 kzoss02
```

7. Enable the autostart of the cluster:

```
# pcs cluster enable --all
```

8. Create the stonith resources:

```
# pcs stonith create kzoss01-ipmi fence_ipmilan
ipaddr="192.168.212.213" lanplus=true passwd="XXXXX" login="XXXX"
pcmk_host_list="kzoss01"

# pcs stonith create kzoss02-ipmi fence_ipmilan
ipaddr="192.168.212.214" lanplus=true passwd="XXXXX" login="XXXX"
pcmk_host_list="kzoss02"
```

9. Add an additional network to increase the reliability:

```
# ccs -f /etc/cluster/cluster.conf --addalt kzoss01 192.168.214.213
port=7809 mcast=239.192.1.2 ttl=2

# ccs -f /etc/cluster/cluster.conf --addalt kzoss02 192.168.214.214
port=7809 mcast=239.192.1.2 ttl=2

# ccs -f /etc/cluster/cluster.conf --settotem rrp_mode="active"
```

10. Sync the cluster:

```
#pcs cluster sync
```

11. Now reboot both the servers.

Storage servers can become overloaded and we don't want failover to occur for this reason. It may be necessary to increase the token value of the cluster. If no token value is specified in the cluster configuration, the default is 10000 ms, or 10 seconds. To use a value other than the default, add or edit the totem line in /etc/cluster/cluster.conf as a child of the <cluster> element, please refer to Red Hat Knowledgebase for more information

## ZFS Configuration

In a ZFS configuration, we have access to all the individual disks available in the JBOD attached to the cluster. Each individual disk is reachable through two or more paths for high availability. You can use the device multi-mapper technology included in Red Hat Linux to create the ZFS pool.

### Create zpool ost0000

A ZFS pool will represent a Lustre OST. All entities in a pool are VDEVs. A VDEV can be a single storage device or a RAIDZ2 group composed of multiple storage devices. Perform the following procedure to create zpool ost0000.

```
# zpool create ost0000 -o cachefile=none raidz2 mpathaa mpathab
mpathac mpathad mpathae mpathaf mpathag mpathah mpathai mpathaj
# zpool list
NAME      SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH
ALTROOT
ost0000  1.33T   152K  1.33T         -     0%     0%  1.00x  ONLINE  -

 [root@kzoss01 ~]# zpool status
  pool: ost0000
 state: ONLINE
  scan: none requested
config:

        NAME         STATE     READ WRITE CKSUM
        ost0000      ONLINE       0     0     0
          raidz2-0   ONLINE       0     0     0
            mpathaa  ONLINE       0     0     0
            mpathab  ONLINE       0     0     0
            mpathac  ONLINE       0     0     0
            mpathad  ONLINE       0     0     0
            mpathae  ONLINE       0     0     0
            mpathaf  ONLINE       0     0     0
            mpathag  ONLINE       0     0     0
            mpathah  ONLINE       0     0     0
            mpathai  ONLINE       0     0     0
            mpathaj  ONLINE       0     0     0
```

We need to be sure that the zpool is available on both the nodes of the cluster:

```
# zpool export ost0000
```

Now go to the other node:

```
# partprobe
# zpool import -o cachefile=none ost0000
```

For a perfect alignment with the Lustre typical “block” size, we suggest creating RAIDZ2 groups with 10 storage devices. It is also possible to organize a single pool with several RAIDZ2 groups, each of 10 storage devices.

**Create zpool ost0001**

Now, repeat the same procedure above to create zpool ost0001.

## Lustre Formatting and Target Configuration

Next, we will associate one ZFS pool for each server:

- ost0000 on kzoss01
- ost0001 on kzoss02

1. Manually import ost0000 on kzoss01 and format the zfs pool as follows:

```
# mkfs.lustre --ost --backfstype=zfs --fsname=lustrefs --index=0 --
mgsnid=192.168.211.211@o2ib0 --mgsnid=192.168.211.212@o2ib0 --
servicenode=192.168.211.213@o2ib0 --
servicenode=192.168.211.214@o2ib0 --reformat ost0000/ost0000
```

2. Manually import ost0001 on kzoss02 and format the zfs pool as follows:

```
# mkfs.lustre --ost --backfstype=zfs --fsname=lustrefs --index=1 --
mgsnid=192.168.211.211@o2ib0 --mgsnid=192.168.211.212@o2ib0 --
servicenode=192.168.211.214@o2ib0 --
servicenode=192.168.211.213@o2ib0 --reformat ost0001/ost0001
```

3. Create the mount points on all the servers:

```
# mkdir -p /lustrefs/ost0000
# mkdir -p /lustrefs/ost0001
```

4. Mount manually for the first time to verify on each node (MGT and MDT must be already up and running):

```
# mount -t lustre ost0000/ost0000 /lustrefs/ost0000
# mount -t lustre ost0001/ost0001 /lustrefs/ost0001
```

5. Then unmount.
6. Add the LustreZFS script available in the Appendix to each host at the location /usr/lib/ocf/resource.d/heartbeat/. Set the script's filename to LustreZFS and set the permissions for the script to 755.
7. Create the resources in pacemaker. Enter the following commands:

```
# pcs resource create lustrefs-OST0000 ocf:heartbeat:LustreZFS
pool="ost0000" volume="ost0000" mountpoint="/lustrefs/ost0000"

# pcs resource create lustrefs-OST0001 ocf:heartbeat:LustreZFS
pool="ost0001" volume="ost0001" mountpoint="/lustrefs/ost0001"
```

... where `pool` is the name of the ZFS pool created previously, `volume` is the name of the Lustre volume in the pool, and `mountpoint` is the directory where the volume will be mounted.

8. To configure a preferred server for this resource, please reflect the primary and secondary node selected during the format phase:

```
# pcs constraint location lustrefs-OST0000 prefers kzoss01=20
# pcs constraint location lustrefs-OST0000 prefers kzoss02=10
# pcs constraint location lustrefs-OST0001 prefers kzoss02=20
# pcs constraint location lustrefs-OST0001 prefers kzoss01=10
```

### Mount the File System on a Client

```
# mount -t lustre
192.168.211.211@o2ib0:192.168.211.212@o2ib0:/lustrefs /mnt/
```

### Compression

ZFS can compress data on file systems. While not appropriate for every situation, compression can increase system performance by improving IO at the cost of CPU. In most cases, disk IO, more than CPU, is the performance bottleneck for storage systems.

```
# zfs compression=on ost0000
# zfs compression=on ost0001
```

## Detecting and Monitoring the File System at the Dashboard

To be able to detect and monitor the ZFS-based Lustre file system using Intel® Manager for Lustre* software, perform the following procedures.

### Add Servers

To add all of the servers to your file system so that Intel® Manager for Lustre* software is aware of them, open the Open the Intel® Manager for Lustre* dashboard and click **Help**, or open the User Guide. Perform the procedure *Add servers to be monitored only.* In this procedure, add all of the ZFS servers that will comprise your file system as instructed. Remember to use the **Monitored storage server** profile for each server.

**Note**: You don't need to add all servers at the same time or in the same session, but be sure to add all of them before performing the next step: *Detect the new file system*.

**Note**: If you configured your file system for high availability, be sure to also add those servers you have configured as failover servers. Remember that for ZFS-based Lustre file systems, Intel® Manager for Lustre* software does not perform HA configuration.

### Detect the New File System

**Note**: Assuming that you have configured this system for HA, be sure to test failover for the system before detecting the new file system. This is so that all servers will be detected.

After adding all file system servers, open the Intel® Manager for Lustre* **Help** (or user guide) and perform the procedure *Detect file systems*. This section is located in *Detecting and monitoring existing Lustre file systems*. Be sure to select all listed servers that belong to this new file system.

The ZFS-based Lustre file system should now be visible on the Intel® Manager for Lustre* dashboard. See the integrated Help for extensive information about how to monitor the file system.

### Mounting the Lustre File System on a Client via the Dashboard

Make sure the Intel® Enterprise Edition for Lustre* client software has been installed on each client before attempting to mount the file system.

To obtain the command to use to mount your file system:

1. At the manager Dashboard menu bar, click the **Configuration** drop-down menu and click **File Systems**.
2. Each Lustre file system created or detected using Intel® Manager for Lustre* software is listed. Select the file system to be mounted. A window opens that shows information for that file system.
3. On the file system page, click **View Client Mount Information**.
4. The mount command to be used to mount the file system is displayed. On the client, enter the actual command. Following is an example only:

```
  mount -t lustre
192.168.214.211@o2ib0:192.168.214.212@o2ib0:/lustrefs
/mnt/lustrefs
```

## Troubleshooting

### Procedure for Updating ZFS or Mellanox OFED

This procedure is for updating the ZFS version or Mellanox OFED version without updating the version of Lustre* software. This procedure is also valid for adding Mellanox OFED support after Lustre* and ZFS have been installed. The following instructions are to be run on each server (MDS/MGS/OSS) that the update is required.

1. Remove existing Lustre*: `dkms remove lustre/2.5.39 –all`
2. Update ZFS or Mellanox accordingly.
3. Re-install Lustre*: `dkms install lustre/2.5.39`

### Mellanox OFED not installing

1. Ensure you have the correct version
    a. Correct operating system.
    b. Correct version of OS (e.g. rhel6.6 does not support rhel6.7 systems)
2. Force the install: `./mlnxofedinstall -k $(head -1 .supported_kernels)`

This procedure is for updating ZFS version or Mellanox OFED version without updating Lustre* version.

## Appendix: LustreZFS

The following script can be used to manage ZFS and Lustre on shared storage. For each host, save this script as "LustreZFS" at the location: `/usr/lib/ocf/resource.d/heartbeat/` Set the script's permissions to 755.

```
#!/bin/sh
#
# License:      GNU General Public License (GPL)v2
# Description:  Manages ZFS and Lustre on a shared storage
# Written by:   Gabriele Paciucci
# Release Date: 01 Oct 2015
# Release Version: 0.9
# Copyright © 2015, Intel Corporation
#
# This program is free software; you can redistribute it and/or modify
# it under the terms and conditions of the GNU General Public License,
# version 2, as published by the Free Software Foundation.
#
# This program is distributed in the hope it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
#
# usage: ./LustreZFS {start|stop|status|monitor|validate-all|meta-data}
#
#               OCF parameters are as follows
#               OCF_RESKEY_pool - the pool to import/export
#               OCF_RESKEY_volume - the volume to mount/umount
#               OCF_RESKEY_mountpoint - the mountpoint to use
##########################################################################
# Initialization:
```

```
: ${OCF_FUNCTIONS_DIR=${OCF_ROOT}/lib/heartbeat}
. ${OCF_FUNCTIONS_DIR}/ocf-shellfuncs

# Defaults

# Variables used by multiple methods


##########################################################################

# USAGE

usage() {
        usage: $0 {start|stop|status|monitor|validate-all|meta-data}
}

# META-DATA

meta_data() {
        cat <<END
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="LustreZFS">
<version>0.9</version>
<longdesc lang="en">
This script manages ZFS pools and Lustre volumes. The script is able to
import and export ZFS pools and mount/umount Lustre.
</longdesc>
<shortdesc lang="en">Lustre and ZFS management</shortdesc>

<parameters>

<parameter name="pool" unique="1" required="1">
<longdesc lang="en">
The name of the ZFS pool to manage.
</longdesc>
<shortdesc lang="en">ZFS pool name</shortdesc>
<content type="string" default="" />
</parameter>

<parameter name="volume" unique="1" required="1">
<longdesc lang="en">
The name of the volume created during the Lustre format on the ZFS pool.
</longdesc>
<shortdesc lang="en">Lustre volume name in the pool</shortdesc>
<content type="string" default="" />
</parameter>


<parameter name="mountpoint" unique="1" required="1">
```

```
<longdesc lang="en">
The mount point where the Lustre target will be mounted.
</longdesc>
<shortdesc lang="en">Mount point for Lustre</shortdesc>
<content type="string" default="" />
</parameter>


</parameters>

<actions>
<action name="start"   timeout="300s" />
<action name="stop"    timeout="300s" />
<action name="monitor" depth="0"  timeout="300s" interval="20s" />
<action name="validate-all"  timeout="30s" />
<action name="meta-data"  timeout="5s" />
</actions>
</resource-agent>
END
        exit $OCF_SUCCESS
}

# FUNCTIONS

zpool_is_imported () {
    zpool list -H "$OCF_RESKEY_pool" > /dev/null
}

lustre_is_mounted () {
        # Verify if this is consistent
        cat /proc/mounts |grep "$OCF_RESKEY_mountpoint" >/dev/null 2>&1;

}

zpool_import () {
    if ! zpool_is_imported; then
         ocf_log info "Starting to import $OCF_RESKEY_pool"

# The meanings of the options to import are as follows:
#   -f : import even if the pool is marked as imported
#   -o cachefile=none : the import should be temporary
        if zpool import -f -o cachefile=none "$OCF_RESKEY_pool" ; then
            ocf_log info "$OCF_RESKEY_pool imported successfully"
            return $OCF_SUCCESS
        else
            ocf_log err "$OCF_RESKEY_pool import failed"
            return $OCF_ERR_GENERIC
        fi
    fi
}
```

```
zpool_export () {
    if zpool_is_imported; then
         ocf_log info "Starting to export $OCF_RESKEY_pool"

# The meanings of the options to export are as follows:
#   -f : export in every case

        if zpool export -f "$OCF_RESKEY_pool" ; then
            ocf_log info "$OCF_RESKEY_pool exported successfully"
            return $OCF_SUCCESS
        else
            ocf_log err "$OCF_RESKEY_pool export failed"
            return $OCF_ERR_GENERIC
        fi
    fi
}


lustre_mount () {
        if ! lustre_is_mounted; then
                ocf_log info "Starting to mount $OCF_RESKEY_volume"

# The meanings of the options to export are as follows:
#

                if mount -t lustre "$OCF_RESKEY_pool/$OCF_RESKEY_volume"
$OCF_RESKEY_mountpoint ; then

                ocf_log info "$OCF_RESKEY_volume mounted successfully"
            return $OCF_SUCCESS
        else
            ocf_log err "$OCF_RESKEY_volume mount failed"
            return $OCF_ERR_GENERIC
        fi
    fi
}


lustre_umount () {

if lustre_is_mounted; then
                ocf_log info "Starting to unmount $OCF_RESKEY_volume"

# The meanings of the options to export are as follows:
#  -f : force umount

                if umount -f $OCF_RESKEY_mountpoint; then

                ocf_log info "$OCF_RESKEY_volume unmounted successfully"
```

```
            return $OCF_SUCCESS
        else
            ocf_log err "$OCF_RESKEY_volume unmount failed"
            return $OCF_ERR_GENERIC
        fi
    fi
}


zpool_monitor () {

# If the pool is not imported, then we can't monitor its health
    if ! zpool_is_imported; then
     ocf_log warn "$OCF_RESKEY_pool not imported"
        return $OCF_NOT_RUNNING
    fi

    # Check the pool status
    # Possible status:
    #           DEGRADED
    #   FAULTED
    #   OFFLINE
    #   ONLINE
    #   REMOVED
    #   UNAVAIL


    HEALTH=$(zpool list -H -o health "$OCF_RESKEY_pool")
    case "$HEALTH" in
        ONLINE)                 #to debug ocf_log info "$OCF_RESKEY_pool is
$HEALTH"
                                return $OCF_SUCCESS
                                ;;
        DEGRADED)               ocf_log warn "$OCF_RESKEY_pool is $HEALTH"
                                return $OCF_SUCCESS
                                ;;
        FAULTED)                ocf_log err "$OCF_RESKEY_pool is $HEALTH"
                                return $OCF_NOT_RUNNING
                                ;;
        *)                      ocf_log err "$OCF_RESKEY_pool is $HEALTH"
                                return $OCF_ERR_GENERIC
                                ;;
    esac
}


lustre_monitor () {

if ! lustre_is_mounted; then
                ocf_log err "$OCF_RESKEY_volume is not mounted"
```

```
                return $OCF_NOT_RUNNING

        else
                # to debug ocf_log info "$OCF_RESKEY_volume is mounted"
                return $OCF_SUCCESS

fi

}



all_start () {

        # Import first the pool
        zpool_import

        # Sleep few seconds
        sleep 5

        # Mount Lustre
        lustre_mount

}



all_stop () {

        # Unmount Lustre
        lustre_umount

        # Sleep few seconds
        sleep 5

        # Export the pool
        zpool_export

}


all_monitor () {

        zpool_monitor

        lustre_monitor

}
```

```
validate () {

    # Maybe we can implement some validation
    return $OCF_SUCCESS

}



case $1 in
    meta-data)          meta_data;;
    start)              all_start;;
    stop)               all_stop;;
    status|monitor)     all_monitor;;
    validate-all)       validate;;
    usage)              usage
                        exit $OCF_SUCCESS
                        ;;
    *)                  exit $OCF_ERR_UNIMPLEMENTED;;
esac

exit $?
```