Defining Multiple LNet Interfaces for Multi-Homed Servers and High Availability

In order to communicate, all Lustre servers and clients must be configured as peers in one or more Lustre networks, or LNets. A host may be configured with multiple LNet end-points, each represented by a unique Lustre Network Identifier, or NID. Lustre nodes can be joined to multiple LNets simultaneously, a configuration commonly referred to as "multi-homing".

Lustre services (MGT, MDTs, OSTs) must be configured with the explicit list of hosts that can mount and serve a given Lustre storage target. The storage target is mounted by a single host from the list at any one time. Similarly, each Lustre service (with the exception of the MGS itself) must be configured with the list of hosts that provide the MGS, so that the services can register with the MGS on startup.

Each host in the list is represented by a list of the interfaces that can be used for LNet communication. This provides Lustre with the flexibility to serve multiple networks simultaneously. The following provides an overview of the Lustre syntax for defining LNet host lists, and how to apply that information when creating high availability and multi-homed Lustre servers.

Lustre has a specific syntax for defining lists of hosts, where each host entry in the list is itself comprised of one or more LNet network identifiers (NIDs). The set of NIDs for a single host or node is formatted as a comma-separated list as follows:

```
<NID>[,<NID> …]
```

The NID is a representation of a Lustre network (LNet) end-point, formatted as follows:

```
<IPv4 address>@<lnd><lnet #>
```

The simplest definition is one for a host with a single NID. For example:

```
10.0.0.12@tcp0
```

The next example defines a multi-homed host with 2 NIDs:

```
10.10.2.11@o2ib0,192.168.2.11@tcp1
```

A multi-homed host with 3 NIDs:

```
10.10.2.11@o2ib0,192.168.2.11@tcp1,10.20.2.11@o2ib1
```

Multiple host specifications can be concatenated together to create a list of nodes. The node list is delimited by colons, and each node entry on the list conforms to the host specification:

```
<host spec>[:<host spec> ...]
```

This can be expanded as follows:

```
<NID>[,<NID> ...][:<NID>[,<NID> ...]]
```

For example, a node list representing two hosts, each with a single NID:

```
10.10.2.11@o2ib0:10.10.2.12@o2ib0
```

A list of 2 hosts, each with 2 NIDs:

```
10.10.2.11@o2ib0,192.168.2.11@tcp1:10.10.2.12@o2ib0,192.168.2.12@tcp1
```

Host A Host B

For consistency, the list of hosts will be referred to as the *node list specification*.

Host Lists and High Availability

Each storage target in Lustre is defined with a list of service nodes, usually when the target is first formatted, representing the target's failover configuration. The failover configuration is supplied using the `--servicenode` or `--failnode` command line options.

All of the examples implicitly use an LDISKFS OSD for the back-end storage, but the syntax for specifying service nodes, failover nodes and MGS nodes is the same for all OSD types, including ZFS. For a more complete description of the `mkfs.lustre` command syntax and creating storage targets for Lustre, refer to the `mkfs.lustre(8)` man page.

For example, to create an MGT that runs only on a single host and has a single LNet NID:

```
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
  --servicenode 192.168.227.11@tcp1 \
  /dev/sda
```

In this particular case, the `--servicenode` flag is, strictly speaking, optional and will be assigned implicitly when the MGT is mounted for the first time.

To create a multi-homed MGS service on a single host, use the host specification format to supply a comma-separated list:

```
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
  --servicenode 192.168.227.11@tcp1,10.10.2.11@o2ib0 \
  /dev/sda
```

The above example is not an HA configuration because it represents only a single server, but it does mean that the server can be accessed on two LNets: `tcp1` and `o2ib0`.

The following two examples show how to create a high availability configuration with two machines, each represented by a single NID:

```
# Example 1
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
```

```
  --servicenode 192.168.227.11@tcp1 \
  --servicenode 192.168.227.12@tcp1 \
  /dev/sda

# Example 2
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
  --servicenode 192.168.227.11@tcp1:192.168.227.12@tcp1 \
  /dev/sda
```

The first example uses multiple `--servicenode` options, one for each host, while the second, more compact syntax uses the node list specification to create a single list delimited by the colon character. The first syntax is generally more readable and is preferred.

The next example creates a failover configuration where each node is also multi-homed:

```
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
  --servicenode 192.168.227.11@tcp1,10.10.2.11@o2ib0 \
  --servicenode 192.168.227.12@tcp1,10.10.2.12@o2ib0 \
  /dev/sda
```

Again, this can be expressed with the more concise syntax using a colon-delimited node list:

```
[root@rh7z-mds1 ~]# mkfs.lustre --mgs \
 --servicenode \
 192.168.227.11@tcp1,10.10.2.11@o2ib:192.168.227.12@tcp1,10.10.2.12@o2ib \
 /dev/sda
```

Reading the Failover Configuration from a Lustre OSD

To determine the failover configuration of a storage target, use `tunefs.lustre`:

```
tunefs.lustre [--dryrun] <dev>|<zfs dataset>
```

For example:

```
[root@rh7z-mds1 ~]# tunefs.lustre /dev/sda
checking for existing Lustre data: found
Reading CONFIGS/mountdata

   Read previous values:
Target:     MGS
Index:      unassigned
Lustre FS:
Mount type: ldiskfs
Flags:      0x1044
              (MGS update no_primnode )
Persistent mount opts: user_xattr,errors=remount-ro
Parameters:
failover.node=192.168.227.11@tcp1,10.10.2.11@o2ib:192.168.227.12@tcp1,
10.10.2.12@o2ib
```

```
   Permanent disk data:
Target:     MGS
Index:      unassigned
Lustre FS:
Mount type: ldiskfs
Flags:      0x1044
              (MGS update no_primnode )
Persistent mount opts: user_xattr,errors=remount-ro
Parameters:
failover.node=192.168.227.11@tcp1,10.10.2.11@o2ib:192.168.227.12@tcp1,
10.10.2.12@o2ib

Writing CONFIGS/mountdata
```

The parameter `failover.node` contains the failover information that is supplied using the `--servicenode` or `--failnode` flags. Note that when the older `--failnode` syntax is used, only the failover target gets written to the `failover.node` parameter. Lustre creates an implicit primary node based on the NIDs of the host that first mounts the OSD. This implicit configuration can create issues if the OSD is mounted on the "wrong" primary node, because the NIDs will not be those of the intended primary. The intended primary will then be unable to mount the storage. The `--failnode` usage also has an impact on the startup procedures for the first time startup of new OSDs. For these reasons, the `--servicenode` syntax is recommended for all Lustre installations.

Using Host Lists to Define the MGS Specification

For storage targets other than the MGT, and for Lustre clients mounting a file system, the node list specification syntax is also used to define the list of machines that serve the MGS for a file system. The MGS specification is usually supplied when the storage target is formatted, using the `--mgsnode` flag. For example, the following command creates a standalone MDT with a single MGS node reference:

```
mkfs.lustre --mdt \
  --fsname demo \
  --index 0 \
  --mgsnode 192.168.227.11@tcp1 \
  --servicenode 192.168.227.12@tcp1 \
  /dev/sdb
```

This next example also only specifies a single MGS node but this time the MGS is on a multi-homed server (note the comma):

```
mkfs.lustre --mdt \
  --fsname demo \
  --index 0 \
  --mgsnode 192.168.227.11@tcp1,10.10.2.11@o2ib0 \
```

```
    --servicenode 192.168.227.12@tcp1,10.10.2.12@o2ib0 \
    /dev/sdb
```

Also note that in the above example, the MDT is also on a single multi-homed server.

To specify that the MGS is part of a high availability failover configuration, use multiple `--mgsnode` parameters, one for each MGS host:

```
mkfs.lustre --mdt \
  --fsname demo \
  --index 0 \
  --mgsnode 192.168.227.11@tcp1 \
  --mgsnode 192.168.227.12@tcp1 \
  --servicenode 192.168.227.12@tcp1 \
  --servicenode 192.168.227.11@tcp1 \
  /dev/sdb
```

or use the colon-separated list syntax:

```
mkfs.lustre --mdt \
  --fsname demo \
  --index 0 \
  --mgsnode 192.168.227.11@tcp1:192.168.227.12@tcp1 \
  --servicenode 192.168.227.12@tcp1:192.168.227.11@tcp1 \
  /dev/sdb
```

Note: In old versions of Lustre, prior to Lustre 2.7.0, there are two bugs in the ZFS OSD that prevent the correct setting of multiple MGS nodes (LU-4334), and failover NIDs when using the `--servicenode` syntax (LU-4749).

In the next example, each MGS node has two LNets:

```
mkfs.lustre --mdt --reformat \
  --fsname demo \
  --index 0 \
  --mgsnode 192.168.227.11@tcp1,10.10.2.11@o2ib0 \
  --mgsnode 192.168.227.12@tcp1,10.10.2.12@o2ib0 \
  --servicenode 192.168.227.12@tcp1,10.10.2.12@o2ib0 \
  --servicenode 192.168.227.11@tcp1,10.10.2.11@o2ib0 \
  /dev/sdb
```

Note: In versions of Lustre prior to 2.9, or Intel® EE for Lustre software version 3.1, there is a parsing bug documented in JIRA ticket LU-8311 that affects the MGS node list definition when the MGS has multiple NIDs. To work around this issue, use the above syntax where each MGS node is listed separately with multiple `--mgsnode` flags, rather than using the colon-separated syntax.

Clients must also reference the MGS when mounting the Lustre file system. The MGS NIDs are part of the mount command, which uses the same syntax to define the node list. In the following example, the client and the `/demo` file system are connected via LNet `tcp1`:

```
[root@rh7z-c1 ~]# lctl list_nids
192.168.227.77@tcp1

[root@rh7z-c1 ~]# mount -t lustre \
  192.168.227.11@tcp1:192.168.227.12@tcp1:/demo /lustre/demo
```

It is unusual for a Lustre client to be multi-homed. Nevertheless, one can supply the full node list specification to the mount command and Lustre will determine the most appropriate LNet to use for communication.

In the next example, the file system is available on LNets `tcp1` and `o2ib0`, and the client is on `tcp1`:

```
[root@rh7z-c1 ~]# lctl list_nids
192.168.227.77@tcp1
[root@rh7z-c1 ~]# mount -t lustre \

192.168.227.11@tcp1,10.10.2.11@o2ib0:192.168.227.12@tcp1,10.10.2.12@o2
ib0:/demo /lustre/demo
```

This does increase the complexity of the mount syntax and it is often easier to only specify the MGS NIDs for the LNet that the client is connected to, or to specify the preferred LNet NIDs, if the client is itself multi-homed.

In this example, there are two clients, one on `tcp1`, and the other on `o2ib0`. The servers are multi-homed on both LNets. The MGS host is on LNets `tcp1` and `o2ib0`:

```
[root@rh7z-mds1 ~]# tunefs.lustre --dryrun /dev/sda|grep ^Parameters |
tail -1
Parameters:
failover.node=192.168.227.11@tcp1,10.10.2.11@o2ib:192.168.227.12@tcp1,10.10.2
.12@o2ib
```

The first client is on LNet `tcp1`:

```
[root@rh7z-c1 ~]# lctl list_nids
192.168.227.77@tcp1
[root@rh7z-c1 ~]# mount -t lustre \
  192.168.227.11@tcp1:192.168.227.12@tcp1:/demo /lustre/demo
```

The second client is on LNet `o2ib0`:

```
[root@rh7z-c2 ~]# lctl list_nids
10.10.2.78@o2ib0
[root@rh7z-c2 ~]# mount -t lustre \
```

```
10.10.2.11@o2ib0:10.10.2.12@o2ib0:/demo /lustre/demo
```

Changing the Network Configuration of Lustre Storage Servers

The LNet service node and MGS node list specifications for Lustre servers are stored persistently on the storage targets, and are written into file system attributes of the OSD back-end when the storage is formatted. Lustre attributes are stored as file system parameters on LDISKFS (EXT4) OSDs, and as dataset properties on ZFS OSDs.

When the server's NID specification changes – for example, if the servers are moved to a different network, or a new LNet is added – then the storage target attributes that contain network information must be updated to match the changes.

The most common reasons for altering the network configuration of a Lustre file system's servers are:

1. Adding additional LNets to servers, to create a multi-home file system accessible from multiple network fabrics.
2. Migrating a Lustre file system to a new network range, subnet or fabric.
3. Correcting an error in the node list specification of an individual server.

Note: Changing the network configuration of Lustre services cannot be executed while the file system is online. All clients must unmount the affected file system and all MDS and OSS servers for the file system must also unmount the storage targets. The MGS only needs to be stopped if the `servicenode` specification for the MGT itself needs to be changed (e.g. because the network configuration of the MGS host is being changed).

Migrating a Lustre File System to a New Network or Adding Additional LNets to Create Multi-Home Lustre Servers

The processes for migrating a file system to a new address range or subnet or for adding a multi-home configuration are largely the same, and so are presented as a single procedure in this guide. The chief differences are in the node specification of the service nodes and the MGS nodes when writing the new configuration to the Lustre storage targets, as well as in the definition of the LNets themselves.

The high-level procedure is below. Details are provided in subsequent sections.

1. Unmount the affected Lustre file system from all Lustre clients.
2. Stop the file system (unmount MDTs and OSTs; umount MGT if required).
3. On each server, unload the Lustre and LNet kernel modules.
4. On each server, update the OS network configuration.
5. On each server, update the Lustre Network (LNet) configuration.
6. Update the `servicenode` and `mgsnode` specifications for each Lustre storage target in the file system.
7. Update the Lustre clients.

8. Restart the file system services: MGS, MDTs, then OSTs.
9. Mount the file system on the clients.

Unmount the affected Lustre file system from all Lustre clients

1. Log into each Lustre client node and unmount the file system:

   ```
   umount <path>
   ```
2. Verify that the file system has unmounted:

   ```
   mount -t lustre
   ```

   or

   ```
   df -t lustre
   ```

   The file system should not appear in either the `mount` or `df` command output.
3. If the client is also being migrated to a new network, then remove the Lustre kernel modules as well:

   ```
   lustre_rmmod
   ```

Stop the File System

1. On the metadata servers, unmount all of the MDTs for the file system, making sure that MDT0 is the last target that is unmounted.
2. Unmount all of the OSTs on each of the object storage servers.
3. If the MGS network configuration is also going to be updated, then unmount the MGT as well. This is usually hosted by one of the metadata server nodes.

Note: If the storage targets are running in a cluster framework such as Pacemaker, then use the cluster tools to stop the resources from running. For example, `pcs` provides the `resource disable` command:

```
pcs resource disable <resource id>
```

An alternative is to set the resources as `unmanaged` by the cluster framework, then unmount the storage by hand:

```
pcs resource unmanage <resource id> [<resource id> ...]
umount <path>
```

One each server, unload the Lustre and LNet kernel modules

Once the file system is offline, unload the Lustre and LNet kernel modules from each server:

```
lustre_rmmod
```

On each server, update the OS network configuration

Use the supplied operating system utilities to update the network configuration for all of the servers, assigning addresses to each of the network interfaces that will be used by LNet. Verify that the configuration is correct before proceeding. In particular, make sure that each interface can establish a network connection to another host on the same network.

On each server, update the Lustre Network (LNet) configuration

Lustre provides two methods for configuring LNet interfaces: static configuration using kernel module options and the newer Dynamic LNet Configuration software. Because Dynamic LNet configuration is a newer feature and is not present in all versions of Lustre, the following examples will use the older static configuration method. For a more complete description on LNet configuration, refer to the Lustre wiki, <ref TBD>.

1. Identify the network interfaces that will be used for Lustre. Use the output from the `ip address show` command to list all available network devices on the host.
2. Create or edit the module options configuration file, e.g., `/etc/modprobe.d/lustre.conf`, ensuring that there is a line describing the new LNet configuration. For example:

   ```
   options lnet networks="o2ib0(ib0),tcp1(eth1)"
   ```

   This configuration tells the `lnet` driver that there are two LNet NI's: the `ib0` interface using the RDMA driver on LNet `o2ib0` and the `eth1` interface using the sockets driver on LNet `tcp1`.

   Note: for a straightforward IP address migration, using the same network interfaces and LNet drivers as before, the LNet configuration may not need to be changed.
3. Load the `lnet` kernel module:

   ```
   modprobe [-v] lnet
   lctl network configure
   ```
4. Verify that the new NIDs have been loaded correctly:

   ```
   lctl list_nids
   lctl net show
   ```
5. Use the `lctl ping` command to verify connectivity to other hosts on the same LNets. For example:

   ```
   lctl ping 10.10.2.11@tcp1
   lctl ping 192.168.2.11@o2ib0
   ```

If there are any errors, or the configuration does not appear to be correct, review the following:

1. IP Addresses and link status of all IP devices:

   ```
   ip addr show
   ip link show
   ```
2. For InfiniBand devices, check the link status and error counters:

   ```
   ibstat
   perfquery
   ```
3. For Intel Omni-Path interfaces:

   ```
   opainfo
   ```
4. Check that the LNet devices match the intended host interfaces.

5. Check that both the driver and LNet number are correct and match those of the other Lustre hosts.

Update the `servicenode` specification for the MGT

Update the configuration of the MGT storage for the MGS, if required. This is not always necessary, but if there is a change that affects the network configuration of the MGS, then the MGT storage must be updated with new `servicenode` definitions.

1. Make sure that the MGT is not mounted.
2. Create a backup of the current configuration, using the `tunefs.lustre` command to capture the existing OSD parameters. **Do not skip this step**: subsequent commands will completely overwrite the original configuration, so this is the only opportunity to capture a backup of the configuration.

   ```
   tunefs.lustre --dryrun <dev>|<zfs dataset> > tunefs-MGT-backup.out
   ```

3. Prepare the `tunefs.lustre` command line that will be used to update the MGT:

   ```
   tunefs.lustre --dryrun --erase-params --writeconf \
     --servicenode <host spec> [--servicenode <host spec> ...] \
     [--param="..."] \
     <dev>|<zfs dataset>
   ```

   At a minimum, the list of the MGS NIDs for the service nodes of the MGT are required. Review the output of the `tunefs.lustre` backup for any additional parameters, such as tuning optimizations, and add those to the command line. In the following example, the MGT configuration is being updated to support high availability and multi-homing (there are two `servicenode` definitions, each representing a host that has two LNet NIDs):

   ```
   tunefs.lustre --dryrun --erase-params --writeconf \
     --servicenode 192.168.2.11@o2ib0,10.10.2.11@tcp1 \
     --servicenode 192.168.2.12@o2ib0,10.10.2.12@tcp1 \
     /dev/sdb
   ```

   Note: The `--dryrun` flag prevents the command from updating the persistent configuration and is a good way to verify that the command line syntax is correct and will achieve the desired outcome before committing the change. The `--erase-params` flag will delete all existing parameters, and the `--writeconf` flag erases and regenerates the configuration logs, but will not have any effect if the `--dryrun` flag is included in the command line.

4. To actually execute the update, re-run the above command without the `--dryrun` flag.
5. The MGT should now have the updated configuration. Verify as follows:

   ```
   tunefs.lustre --dryrun <dev>|<zfs dataset>
   ```

Update the `servicenode` and `mgsnode` specifications for each Lustre storage target in the file system

This procedure must be executed against every MDT and OST storage device across all of the Lustre servers in the file system. Each storage target will be reconfigured with an updated MGS

node specification and new `servicenode` specification based on the new NIDs for the hosts that manage the storage target.

1. Use the `tunefs.lustre` command to capture the existing OSD parameters, creating a backup of the current configuration. **Do not skip this step**: subsequent commands will completely overwrite the original configuration, so this is the only opportunity to capture a backup of the configuration.

   ```
   tunefs.lustre --dryrun <dev>|<zfs dataset> > tunefs-<OSD NAME>-bkup.out
   ```

   For example:

   ```
   tunefs.lustre --dryrun /dev/sda > tunefs-demo-MDT0000-bkup.out
   ```

2. Prepare the `tunefs.lustre` command line that will be used to update the OSD. The preferred syntax closely matches the `mkfs.lustre` command line used to format Lustre targets:

   ```
   tunefs.lustre --dryrun --erase-params --writeconf \
     --mgsnode <MGS host spec> [--mgsnode <MGS host spec> ...] \
     --servicenode <host spec> [--servicenode <host spec> ...] \
     [--param="..."] \
     <dev>|<zfs dataset>
   ```

   At a minimum, the list of the MGS NIDs for the file system, along with the correct NID list for the service nodes of the OSD are required. Review the output of the `tunefs.lustre` backup for any additional parameters, such as tuning optimizations. In the next example, the Lustre storage target is managed by two multi-homed servers, each with two NIDs (each `--servicenode` entry on the command line represents one server, each with two NIDs written as a comma-separated list. There are also two multi-homed MGS servers on `o2ib0` and `tcp1`).

   ```
   tunefs.lustre --dryrun --erase-params --writeconf \
     --mgsnode 192.168.2.11@o2ib0,10.10.2.11@tcp1 \
     --mgsnode 192.168.2.12@o2ib0,10.10.2.12@tcp1 \
     --servicenode 192.168.2.23@o2ib0,10.10.2.23@tcp1 \
     --servicenode 192.168.2.24@o2ib0,10.10.2.24@tcp1 \
    /dev/sdb
   ```

   Note: The `--dryrun` flag prevents the command from updating the persistent configuration and is a good way to verify that the command line syntax is correct and will achieve the desired outcome, before committing the change. The `--erase-params` flag will delete all existing parameters, and the `--writeconf` flag erases and regenerates the configuration logs, but will not have any effect if the `--dryrun` flag is included in the command line.

3. To execute the update, re-run the command without the `--dryrun` flag.

4. The target OSD should now have the updated configuration. Verify using the `tunefs.lustre --dryrun` command:

   ```
   tunefs.lustre --dryrun <dev>|<zfs dataset>
   ```

5. **Caution**: **Do not mount the updated OSD** until all of the OSDs across the entire file system have been changed.
6. Repeat this procedure for all of the MDT and OST storage targets across all of the Lustre servers in the file system.

Restart the file system services: MGS, MDTs, then OSTs

1. Start the MGS.
2. Start the MDS for MDT0.
3. Start the remaining MDTs.
4. Start the OSTs on the OSS servers.

Update the Lustre clients

1. Use the supplied operating system utilities to update the network configuration for all of the clients, if required, assigning addresses to each of the network interfaces that will be used by LNet. Verify that the configuration is correct, and in particular, make sure that each interface can establish a network connection to other hosts on the same network.
2. Update the LNet configuration to match the change, using either the static kernel module options or the newer dynamic LNet configuration software. For example, edit `/etc/modprobe.d/lustre.conf` and add an entry for the `lnet` module:

   ```
   options lnet networks="o2ib0(ib0)"
   ```

3. Load the LNet kernel module:

   ```
   modprobe -v lnet
   lctl network configure
   ```

4. Use `lctl ping` to verify connectivity to the Lustre servers. For example:

   ```
   lctl ping 192.168.2.11@o2ib0
   ```

Mount the file system on the clients

If the MGS NIDs have been changed, then the Lustre clients will need to have their mount options updated to match the new MGS node list specification. The Lustre mount may be recorded in `/etc/fstab` or it may be part of a startup script. Mount the file system on the clients as follows:

```
mount -t lustre \
  <mgsnode>[,<mgsnode> ...][:<mgsnode>[,<mgsnode> ...]]:/<fsname> \
  /<mount point>
```

Alternative Method for Specifying Lustre Parameters with `tunefs.lustre`

It is possible to use the `--param` option to completely specify all of the OSD parameters. However, parameters are passed literally to the persistent configuration, and are not subjected to any additional validation. To illustrate, consider the following example:

```
[root@rh7z-mds2 ~]# tunefs.lustre --dryrun --erase-params \
```

```
>  --writeconf --mgsnode xyzzy@adv3 /dev/sdb
checking for existing Lustre data: found
Reading CONFIGS/mountdata

   Read previous values:
Target:     demo-MDT0000
Index:      0
Lustre FS:  demo
Mount type: ldiskfs
Flags:      0x141
              (MDT update writeconf )
Persistent mount opts: user_xattr,errors=remount-ro
Parameters: mgsnode=192.168.2.11@tcp1

tunefs.lustre: Cannot resolve hostname 'xyzzy@adv3'.
tunefs.lustre: exiting with 1 (Operation not permitted)
```

In the above example, `tunefs.lustre` will not permit the mgsnode option to be written out, because it does not recognize the NID. However, look at what happens when the parameter is written out using the `--param` flag:

```
[root@rh7z-mds2 ~]# tunefs.lustre --dryrun --erase-params \
>  --writeconf --param="mgsnode=xyzzy@adv3" /dev/sdb
checking for existing Lustre data: found
Reading CONFIGS/mountdata

   Read previous values:
Target:     demo-MDT0000
Index:      0
Lustre FS:  demo
Mount type: ldiskfs
Flags:      0x141
              (MDT update writeconf )
Persistent mount opts: user_xattr,errors=remount-ro
Parameters: mgsnode=192.168.2.11@tcp1


   Permanent disk data:
Target:     demo=MDT0000
Index:      0
Lustre FS:  demo
Mount type: ldiskfs
Flags:      0x141
              (MDT update writeconf )
Persistent mount opts: user_xattr,errors=remount-ro
Parameters: mgsnode=xyzzy@adv3

exiting before disk write.
```

The parameter would be written out to the persistent OSD configuration, even though it is not a valid NID. Not only that, one can use this method to write any `key=value` data to the OSD storage. If you choose this method for writing out a new configuration, be very careful that the syntax and value of each parameter are correct.

Lustre Parameters on ZFS OSDs

The process for changing Lustre server storage parameters is the same, regardless of the choice of OSD. Both `mkfs.lustre` and `tunefs.lustre` are intended to work with all OSD types. However, each OSD type implements its own internal data structures for storing parameters.

Lustre OSD parameters are written to ZFS datasets as ZFS user properties. These can be examined with the `zfs get` command:

```
zfs get all | awk '/lustre:/'
```

In ZFS, user properties are arbitrary properties that do not have any effect on ZFS itself but can be defined by applications to annotate datasets (see the man page for `zfs(8)`). User properties are easily recognisable as they must include a colon character in the name. Lustre properties are all prefixed with the string "`lustre:`".

While it is possible to use the `zfs set` or `zfs inherit` commands to change or erase Lustre properties, this is not recommended. The correct tool for managing Lustre configuration changes to the OSD is `tunefs.lustre`; it is designed and developed to support Lustre storage devices. ZFS is not itself "Lustre-aware", and does not have the tools needed to maintain the Lustre data structures and cannot, for example, regenerate the configuration logs after a parameter change.

Caution: Incorrect application of the `zfs` commands can cause irreparable harm to the OSD.

Changing or Repairing the LNet Configuration of a Single Lustre OSD (except MGT)

If a Lustre OSD has been created with the incorrect configuration for the MGS nodes or the list of service nodes contains an error, there are several options to enact a repair, depending on when the error is detected.

First, it is imperative to ensure that the LNet configuration for the host is correct. Check the current configuration using either of the following commands:

```
# lnetctl lnet show
# lctl list_nids
```

If the LNet configuration is wrong, repair using either `lnetctl` (the dynamic LNet configuration tool) or by updating the static kernel module configuration and reloading the LNet kernel modules (unloading the kernel modules will require that all Lustre services be stopped on the affected host).

If the storage target is a new addition and does not contain any data, the simplest thing to do is to reformat the OSD and apply the corrected configuration. If the target was mounted, but no

data was committed to the target, add the `--replace` flag to the `mkfs.lustre` command syntax, to allow the OSD index to be re-used.

If the storage target contains data, the procedure is more complex. It is rare for this situation to occur because incorrect NIDs are usually detected before data is loaded onto the storage device. However, incorrect specification of failover NIDs can go unnoticed, usually because the "primary" NID (that is, the NID of the preferred server, which appears first on the list of hosts in the node specification) for a storage target is correct. Where the affected target is an OST, it may be possible to migrate the data that it holds to other OSTs in the file system, and then reformat the target to correct the error. This is a rather drastic measure and is not recommended. Additionally, this method will not work for MDT0 (the file system root metadata target) or the MGT.

Instead, the recommended approach is to stop the file system by unmounting all clients and all of the server OSDs, excluding the MGT. When the file system is down, overwrite the old configuration of the affected storage target with a new, corrected configuration. Once this has been done, all storage targets for the entire file system must be forced to erase their configuration logs. The file system can then be restarted, and new logs will be generated. The MGS service can remain online for the duration of the operation.

Unfortunately, a change to even a single storage target's configuration will require an outage of the whole file system in order to correct.

The outline process is as follows:

1. Stop Lustre client activity and unmount the affected file system from all clients.
2. Stop the file system services for the file system:
    a. Unmount all of the MDTs.
    b. Unmount all of the OSTs.
    c. **Note**: if the storage targets are running in a cluster framework such as Pacemaker, then one must use the cluster tools to stop the resources from running. For example, `pcs` provides the `resource disable` command.
3. Leave the MGT mounted.
4. When all targets are offline, log into the preferred primary node for the Lustre storage target that is to be updated and execute the following procedure:
    a. Remove all Lustre kernel modules:

    ```
    lctl network unconfigure
    lustre_rmmod
    ```

    b. Update the LNet configuration, if required.
    c. Reload the `lnet` kernel module and verify that the NID[s] for the host are correct.
    d. Use the `tunefs.lustre` command to capture the existing OSD parameters, creating a backup of the current configuration.

    ```
    tunefs.lustre --dryrun <dev>|<zfs dataset> > \
      tunefs-<OSD NAME>-backup.out
    ```

e. Prepare the `tunefs.lustre` command line that will be used to update the OSD. The preferred syntax closely matches the `mkfs.lustre` command line used to format OSDs:

```
tunefs.lustre --dryrun --erase-params --writeconf \
  --mgsnode <MGS hostspec> [--mgsnode <MGS hostspec> ...] \
  --servicenode <hostspec> [--servicenode <hostspec> ...] \
  [--param="..."] \
  <dev>|<zfs dataset>
```

At a minimum, the list of the MGS NIDs for the file system along with the correct NID list for the service nodes of the OSD are required. Review the output of the `tunefs.lustre` backup for any additional parameters, such as tuning optimizations. For example:

```
tunefs.lustre --dryrun --erase-params --writeconf \
  --mgsnode 10.10.2.11@tcp1 \
  --mgsnode 10.10.2.12@tcp1 \
  --servicenode 10.10.2.23@tcp1 \
  --servicenode 10.10.2.24@tcp1 \
  /dev/sdb
```

The `--dryrun` flag prevents the command from updating the persistent configuration. It is a good way to verify that the command syntax is correct and will achieve the desired outcome, before committing the change.

f. To execute the update, re-run the command without the `--dryrun` flag.

g. The target OSD should now have the updated configuration. Verify using the `tunefs.lustre --dryrun` command:

```
tunefs.lustre --dryrun <dev>|<zfs dataset>
```

5. **Do not mount the updated OSD**. Before the OSD can be mounted, all the other OSDs in the file system must have their Lustre configuration logs erased and regenerated. For each OST and MDT on every Lustre server node in the file system , run the following command:

   ```
   tunefs.lustre --writeconf <device>|<zfs dataset>
   ```

6. The MGT does not need to be updated.
7. Start up the Lustre services, beginning with `MDT0`, then the remaining MDTs and finally the OSTs.
8. Start the clients.

Changing or Repairing the LNET Configuration of the Management Service MGT Storage

The process for altering the configuration of the MGT is very similar to that for any other OSD, with the exception that all file systems that are registered with the MGT must be stopped, the

updated MGS node list re-applied to every server in the cluster, and all client mount points updated to use the new MGS node list specification.

The outline process is:

1. Stop Lustre client activity and unmount all file systems that are registered with the MGS from all clients.
2. Stop the file system services for all file systems:
    a. Unmount all of the MDTs.
    b. Unmount all of the OSTs.
    c. Unmount the MGT.

    Note: if the storage targets are running in a cluster framework such as Pacemaker, then one must use the cluster tools to stop the resources from running.
3. When all targets are offline, log into the preferred primary node for the MGT.
    a. Remove all Lustre kernel modules:

    ```
    lctl network unconfigure
    lustre_rmmod
    ```

    b. Update the LNet configuration, if required.
    c. Reload the `lnet` kernel module and verify that the NID[s] for the host are correct.
    d. Use the `tunefs.lustre` command to capture the existing MGT parameters, creating a backup of the current configuration.

    ```
    tunefs.lustre --dryrun <dev>|<zfs dataset> > \
      tunefs-MGT-backup.out
    ```

    e. Prepare the `tunefs.lustre` command line that will be used to update the MGT OSD:

    ```
    tunefs.lustre --dryrun --erase-params --writeconf \
      --servicenode <hostspec> [--servicenode <hostspec> ...] \
      [--param="..."] \
      <dev>|<zfs dataset>
    ```

    f. At a minimum, the list of the MGS NIDs for the service nodes of the MGT are required. Review the output of the `tunefs.lustre` backup for any additional parameters, such as tuning optimizations and add those to the command line. For example:

    ```
    tunefs.lustre --dryrun --erase-params --writeconf \
      --servicenode 10.10.2.11@tcp1 \
      --servicenode 10.10.2.12@tcp1 \
      /dev/sdb
    ```

Note: the `--dryrun` flag prevents the command from updating the persistent configuration and is a good way to verify that the command line syntax is correct and will achieve the desired outcome, before committing the change.

g. To execute the update, re-run the above command without the `--dryrun` flag.

4. The MGT should now have the updated configuration. Verify as follows:

```
tunefs.lustre --dryrun <dev>|<zfs dataset>
```

5. Because the MGT configuration has been altered, all of the storage targets for all file systems will need to be updated with the new MGS node specification. Log into each Lustre server in turn and execute the following procedure:

   a. Create a backup of the current OSD configurations for each target on the server, for example:

```
tunefs.lustre --dryrun <dev>|<zfs dataset> > \
  tunefs-<OSD target>-backup.out
```

   b. Take note of the `failover.node` parameter. This is the persistent representation of the OSD's servicenode specification.

   c. Take note of any additional parameters, but ignore `mgsnode` because this will be replaced by a new MGS node list. Additional parameters, if present, can be applied using the `--param` option.

   d. Prepare the `tunefs.lustre` command line, using the `--dryrun` flag to prevent the target's current configuration from being accidentally overwritten.

```
tunefs.lustre --dryrun --erase-params --writeconf \
  --mgsnode <MGS hostspec> [--mgsnode <MGS hostspec> ...] \
  --servicenode <hostspec> [--servicenode <hostspec> ...] \
  [--param="..."] \
  <dev>|<zfs dataset>
```

   e. For example:

```
tunefs.lustre --dryrun --erase-params --writeconf \
  --mgsnode 10.10.2.11@tcp1 \
  --mgsnode 10.10.2.12@tcp1 \
  --servicenode 10.10.2.23@tcp1 \
  --servicenode 10.10.2.24@tcp1 \
  /dev/sdb
```

   f. To write the new configuration, re-run the `tunefs.lustre` command that was constructed in the last step, but without the `--dryrun` flag.

   g. Verify that the changes have been applied:

```
tunefs.lustre --dryrun <dev>|<zfs dataset>
```

6. When all of the OSDs across all of the servers have been updated, restart the Lustre services in sequence:
    a. Start the MGS.
    b. For each file system, start MDT0, followed by the remaining MDTs and finally the OSTs.
    c. For each client, edit the mount point of each Lustre file system so that it reflects the new MGS node list specification.
    d. Start the clients.