# CONTENTS

Abbreviations ................................................................................................................................. v
Glossary ............................................................................................................................................... vii
Lustre Commands ............................................................................................................................... viii
Introduction ......................................................................................................................................... 1
  Who Should Use this Manual? ........................................................................................................ 1
  Lustre Version Used in this Document ......................................................................................... 1
  Other Sources of Lustre Information ............................................................................................ 2
  About the Authors .......................................................................................................................... 2
Obtaining the Lustre Software .......................................................................................................... 2
  Dependencies ................................................................................................................................. 2
  Downloading Packages .................................................................................................................. 3
Installing Lustre .................................................................................................................................. 3
  Prepatched Kernel RPM with Matching Lustre-utils RPM .......................................................... 3
  Instructions for Installation ........................................................................................................... 3
  Quick Configuration of Lustre ....................................................................................................... 3
  Using the Lustre Wizard for Configuration .................................................................................. 4
Basic Lustre Configuration .............................................................................................................. 6
Setting Up Various Configurations .................................................................................................. 7
  Single Node Client, MDS, and Two OSTs ..................................................................................... 7
  Multiple Nodes ............................................................................................................................. 11
    Multiple Machines – Both OSTs and Clients ............................................................................. 11
    Adding OSTs to Existing File Systems ..................................................................................... 19
    Adding an OST on a New OSS ..................................................................................................... 19
    Adding an OST to an Existing OSS ........................................................................................... 23
  Shutting Down Lustre .................................................................................................................... 26
  Starting Lustre ............................................................................................................................... 27
Performance Considerations ............................................................................................................ 27
  Estimating Hardware Performance .............................................................................................. 27
    Single OST ................................................................................................................................. 28
    Multiple OSTs ............................................................................................................................ 28
  Application Performance .............................................................................................................. 28
  Testing Tools ................................................................................................................................. 28
Monitoring and Administering with Lustre Manager ........................................................................ 30
  Installation and Configuration ....................................................................................................... 31
  Lustre Manager Collector .............................................................................................................. 31
    Connecting to the Lustre Manager ............................................................................................. 32
    Importing the Lustre Configuration into the Lustre Manager .................................................. 32
Best Practices .................................................................................................................................... 40
  Naming Conventions ...................................................................................................................... 40
  XML/Shell Configuration File Management ............................................................................... 40
  MDS/OSS/Client Configuration ..................................................................................................... 41
  Publish Your Experiences ............................................................................................................. 41
  LDAP Support .............................................................................................................................. 41
# ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFS</td>
<td>Cluster File System, Inc.</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Version System</td>
</tr>
<tr>
<td>dcache</td>
<td>Directory Cache</td>
</tr>
<tr>
<td>GNU</td>
<td>GNU is Not Unix</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>LMC</td>
<td>Lustre Configuration Maker</td>
</tr>
<tr>
<td>LOV</td>
<td>Logical Object Volume</td>
</tr>
<tr>
<td>LUG</td>
<td>Lustre Users Group</td>
</tr>
<tr>
<td>MDS</td>
<td>Metadata Server</td>
</tr>
<tr>
<td>MSCP</td>
<td>Mass Storage Communications Protocol</td>
</tr>
<tr>
<td>NAL</td>
<td>Network Abstraction Layer</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>nid</td>
<td>Network ID</td>
</tr>
<tr>
<td>OBD</td>
<td>Object Based Devices</td>
</tr>
<tr>
<td>OSS</td>
<td>Object Storage Servers</td>
</tr>
<tr>
<td>OST</td>
<td>Object Storage Target</td>
</tr>
<tr>
<td>RPM</td>
<td>Red Hat Package Manager</td>
</tr>
<tr>
<td>SAN</td>
<td>Storage Area Network</td>
</tr>
<tr>
<td>SCA</td>
<td>System Communication Architecture</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>VFS</td>
<td>Virtual File System</td>
</tr>
</tbody>
</table>
# GLOSSARY

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOV</td>
<td>A Logical Object Volume (LOV) is a collection of OSTs into a single volume.</td>
</tr>
<tr>
<td>MDS</td>
<td>The Metadata Server (MDS) maintains a transactional record of high-level file and file system changes. The MDS supports all file system namespace operations, such as file lookups, file creation, and file and directory attribute manipulation. It does not contain any file data, instead redirecting actual file I/O requests to OSTs.</td>
</tr>
<tr>
<td>NAL</td>
<td>The Network Abstraction Layer (NAL) provides out-of-the-box support for multiple types of networks. This layer makes it easy to integrate new network technologies.</td>
</tr>
<tr>
<td>OSS</td>
<td>An Object Storage Server (OSS) is a server node that runs the Lustre software stack. It has one or more network interfaces and usually one or more disks.</td>
</tr>
<tr>
<td>OST</td>
<td>An Object Storage Target (OST) is a software interface to a single exported backend volume. It is conceptually similar to an network file system (NFS) export, except that an OST does not contain a whole namespace, but rather file system objects.</td>
</tr>
</tbody>
</table>
LUSTRE COMMANDS

lconf  Lustre file system configuration utility – This utility configures a node following directives in the `<XML-config file>`. There is a single configuration file for all the nodes in a single cluster. This file should be distributed to all the nodes in the cluster or kept in a location accessible to all the nodes. One option is to store the cluster configuration information in lightweight directory access protocol (LDAP) format on an LDAP server that can be reached from all of the cluster nodes.

lctl  Low level Lustre file system configuration utility – This utility provides very low level access to the file system internals.

lfs  Lustre utility to create a file with a specific striping pattern and find the striping pattern of exiting files – This utility can be used to create a new file with a specific striping pattern, determine the default striping pattern, and gather the extended attributes (object numbers and location) for a specific file. It can be invoked interactively without any arguments or in a noninteractive mode with one of the arguments supported.

lmc  Lustre configuration maker – This utility adds configuration data to a configuration file. In the future, lmc will also be able to remove configuration data or convert its format. A Lustre cluster consists of several components: metadata servers (MDSs), client mount points, object storage targets (OSTs), logical object volumes (LOVs), and networks. A single configuration file is generated for the complete cluster. In the lmc command line interface, each of these components is associated with an object type.

lwizard  Lustre configuration wizard – The configuration files for Lustre installation are generally created through a series of lmc commands; this generates an XML file that describes the complete cluster. The lwizard eliminates the need to learn lmc to generate configuration files, instead, lwizard achieves the same result through asking some simple questions. The XML configuration file generated using lwizard still has to be made accessible to all the cluster nodes either by storing it on an LDAP server, network file system (NFS), or by copying it to all the involved nodes. Then lconf is run on all nodes to start the various Lustre services and device setups or to mount the file system. Using lwizard allows the user to simply answer a series of questions about the various pieces of the cluster, and the lwizard completes the configuration.
INTRODUCTION

Over the years, a number of attempts have been made to provide a single shared file system among a number of nodes. Some of these, such as Network File System (NFS), are useful only for a small number of clients, or in limited input/output (I/O) situations where high-speed I/O is not needed. In the past several years, due to the emergence of clusters in the high performance computing market, several projects have emerged to allow many clients to write in parallel to a global storage pool. A recent entry into this area of parallel file systems is the Lustre\textsuperscript{TM} file system.

The Lustre file system is parallel object-based and aggregates a number of storage servers together to form a single coherent file system that can be accessed by a client system. Data about the files being stored in the file system are stored on a metadata server (MDS), and the storage being aggregated is connected to a number of object storage targets (OSTs). The Lustre file system addresses some of the scalability issues of its predecessors by striping file reads and writing across multiple server systems. Lustre is being developed by Cluster File Systems, Inc. (CFS) with a number of corporate partners.

The Lustre Users Group (LUG) is a self organizing group of sites that run the Lustre file system. The mission of LUG is to promote the adoption of the Lustre file system. During the Spring 2005 LUG meeting it was clear that one of the hurdles to get Lustre more widely adopted was the state of the existing Lustre documentation. The purpose of this document is to close the gaps in the existing Lustre documentation. For more information about participation in LUG please email studham@ORNL.gov.

This document (1) gives the web site for Lustre software and additional information about installing Lustre, (2) shows how to install and configure all the components of a Lustre cluster from prebuilt RPMs, and (3) describes how to build Lustre from scratch using source code tarballs. Other topics covered are performance, monitoring and administration, and best practices.

WHO SHOULD USE THIS MANUAL?

The target audience for this document is the system administrator who is Linux literate, but who has never used Lustre before. This document is intended for administrators who are just setting up Lustre.

LUSTRE VERSION USED IN THIS DOCUMENT

Lustre (Linux + Cluster) is a storage and file system architecture and implementation designed for use with very large clusters. Public Open Source releases of Lustre are made under the GNU General Public License. This document uses version 1.4.1, which is available as a supported version (with a paid support contract from CFS) or as an evaluation version (with a limited license). The team chose this version of Lustre because CFS has committed to making this version available for free to the public in the future. This availability gives this document the longest usable life. Additional information on the Lustre licensing is available at http://clusterfs.com/lustre_is_oss.html
OTHER SOURCES OF LUSTRE INFORMATION

Lustre is a new file systems with an emerging support system. CFS develops Lustre, and provides enterprise support to end-users. In addition, various high performance computing companies are beginning to provide level one support for end customers. CFS-provided documentation is available at https://wiki.clusterfs.com/lustre/LustreHowto.

ABOUT THE AUTHORS

The team met in May 2005 at Oak Ridge National Laboratory (ORNL) in Oak Ridge, Tennessee, to perform a test installation of Lustre. Team members are Richard Alexander, ORNL; Chad Kerner, National Center for Supercomputing Applications (NCSA); Jeffery Kuehn, ORNL; Jeff Layton, Linux Networx.; Patrice Lucas, Commissariat à l’Energie Atomique (CEA), France; Hong Ong, ORNL; Sarp Oral, ORNL; Lex Stein, Harvard University; Scott Studham, ORNL; Joshua Schroeder, Chevron; and Steve Woods, MCNC.

OBTAINING THE LUSTRE SOFTWARE

The installation and configuration methods discussed in this document are (1) using the Lustre Wizard, (2) using a single node, and (3) using multiple nodes. These are the techniques recommended for new users of Lustre. Appendix C discusses a more complex, customized method of installation accomplished by building the kernels from sources. This method is quite involved and should not be attempted by novice Lustre users.

Lustre requires the following two items:

1. Linux kernel patches with some Lustre-specific patches and
2. Lustre utilities required for configuration.

In the subsequent sections we will highlight the various methods for satifying these two requirements.

DEPENDENCIES

Lustre uses the Portals software originally developed at Sandia National Laboratories to provide a network abstraction layer that simplifies using Lustre across multiple types of networks. Portals and Lustre’s administrative utilities require the following packages:

1. readline: Install the -devel version of this package, which includes the header files
2. libxml2: Install the -devel version of this package

The RedHat Package Manager (RPM) install of Lustre utilities does not actually check for these four requirements/dependencies.
Some of the more advanced Lustre functionality, such as server failover, require packages and configuration that are outside of the scope of this document.

**DOWNLOADING PACKAGES**

The list below shows the RPMs/tarballs available on the download site for every Lustre release.

1. **lustre-<release-ver>.tar.gz** — This tarball contains the Lustre source code (which includes the kernel patch).
2. **kernel-bigsmp-2.6.5-7.141_lustre.<release-ver>.i686.rpm** — Lustre-patched Linux 2.6.5-7.141 kernel RPM based on Red Hat’s 2.6.5-7.141 kernel package, includes Lustre and Portals kernel models and is used with matching lustre-lite-utils package.
3. **lustre-lite-utils-2.6.5-7.141_lustre.<release-ver>.i686.rpm** — Lustre Utilities – user space utilities for configuring and running Lustre and is used only with the matching kernel RPM listed above.
4. **kernel-source-2.6.5-7.141_lustre.<release-ver>.rpm** — Lustre patched Linux 2.6.5-7.141 kernel source RPM - companion to the kernel package; it is not required to build or use Lustre.

Lustre requires a number of patches to the core Linux kernel, mostly to export new functions, add features to ext3, and add a new locking path to the virtual file system (VFS). You can either patch your own kernel using patches from the Lustre source tarball or download the prepatched kernel RPM along with matching lustre-lite-utils RPM (items 2 and 3 in the above list).

**INSTALLING LUSTRE**

**PREPATCHED KERNEL RPM WITH MATCHING LUSTRE-UTILS RPM**

**Instructions for Installation**

1. Install the kernel-smp-2.6.5-7.141_lustre.<release-ver>.i686.rpm RPM and matching lustre-lite-utils-2.6.5-7.141_lustre.<release-ver>.i686.rpm RPM
2. Update lilo.conf or grub.conf to boot the new kernel
3. Reboot

**Quick Configuration of Lustre**

Lustre system consists of three types of subsystems: clients, a metadata server (MDS), and object storage servers (OSSs). All of these can coexist on a single system or can be running on different systems. A logical object volume manager (LOV) can transparently manage several object storage targets (OSTs) to make it appear that they are a single larger OST; this component is required for achieving file striping. As a side note, if LOV is not defined on a single-OST configuration, Lustre will automatically create one.

It is possible to set up the Lustre system in many different configurations using the administrative utilities provided with Lustre. Lustre includes some sample scripts in the /usr/lib/lustre/examples directory on a system on which Lustre has been installed (or
the lustre/tests subdirectory of a source code installation) that enable quick setup of some simple, standard configurations.

**NOTE:** If your distribution does not contain these examples, go to the Using Supplied Configuration Tools section. Verify your mounted system as described below.

**USING THE LUSTRE WIZARD FOR CONFIGURATION**

Lustre Wizard (lwizard) is used to build a XML file from user-input values. This utility does a good job of providing a base configuration. Depending on your requirements, this script may need to be updated.

**BE CAREFUL!** It is highly recommended to verify the configuration and not assume that the generated XML file is correct.

The lwizard tool supports the following (optional) command set:

```
--batch=FILE
  save lmc batch commands to FILE
-o, --file=FILE
  write Lustre configuration to FILE (default: lwizard.xml)
-f, --force
  force existing files to be overwritten
--help
  to get this help
--stripe_size=SIZE
  size (in KB) of each stripe on an OST (default: 64)
--stripe_count=COUNT
  the number of OSTs files are striped to (default: 1)
--lustre_upcall=LUSTRE_UPCALL
  Set location of lustre upcall script
--portals_upcall=PORTALS_UPCALL
  Set location of portals upcall script
--upcall=UPCALL
  Set both lustre and portals upcall script
```

The lwizard asks a series of questions about the various pieces of the cluster:

- MDS hostname
- MDS device information
- MDS Size
- OST hostname(s)
- OST device information for each OST
- OST Size
- Mount points for lustre on the clients (default - /mnt/lustre)

The utility saves the XML file to the filename specified using the 

```
-o or --file option or the default file lwizard.xml. It will also save the lmc commands used to create the XML file in
<specified-file-name> when used with the --batch option. It is highly recommended to write out the --batch file for reference, or to modify it for your own use.

The lwizard tool currently assumes the following defaults, but it can be changed by either editing the script file or by setting those environment variables prior to running the wizard:

```
Network type : tcp
Filesystem type : ext3
```

4
The following is a list of the environment variables (with brief descriptions) that can be set prior to running lwizard:

- **DEFAULT_FSTYPE**
  - This will set what type of filesystem to use.
- **DEFAULT_NETTYPE**
  - This will choose what type of network protocol being used.
- **DEFAULT_MNTPT**
  - Default mount point that the client will mount
- **STRIPE_SIZE**
  - Stripe size across ost’s
- **STRIPE_CNT**
  - how many ost’s to stripe across
- **STRIPE_PATTERN**
  - This can only be “0” right now, it is raid-0
- **LMC**
  - utility that generates the xml file
- **LUSTRE_UPCALL**
  - Set the location of the Lustre upcall scripts used by the client for recovery. This can be used to proactively notify the other nodes of a problem.
- **PORTALS_UPCALL**
  - Set the location of the Lustre upcall scripts used by the client for recovery. This can be used to proactively notify the other nodes of a problem.
- **UPCALL**
  - Sets both the lustre and portals upcall script.

Below is an example run of lwizard execution to set up the following: 1 MDS server, 1 OSS server with one OST, and N number of clients that will mount /mnt/lustre-example.

```
host:/tmp # /usr/sbin/lwizard --batch=/tmp/example.sh --
file=/tmp/example.xml
lwizard will help you create a Lustre configuration file.

Creating mds "mds1"...
Please enter the hostname(s) for mds1: ex-mds
Please enter the device or loop file name for mds1 on ex-mds: /dev/hdb1
Please enter the device size or 0 to use entire device: 0
Do you want to configure failover mds1?n
Creating ost "ost1"...
Please enter the hostname(s) for ost1: ex-ost
Please enter the device or loop file name for ost1 on ex-ost: /dev/hdc3
Please enter the device size or 0 to use entire device: 0
Do you want to configure failover ost1?n
Creating ost "ost2"...
Please enter the hostname(s) for ost2, or just hit enter to finish:
Please enter the clients' mountpoint (/mnt/lustre): /mnt/lustre-example
Creating mds "mds2"...
Please enter the hostname(s) for mds2, or just hit enter to finish: mds1 lov1 ost1 client
The Lustre configuration has been written to /tmp/example.xml.
The lmc batch file has been written to /tmp/example.sh.
```

Below is a listing of the /tmp/example.sh file that was generated from the above example.

```
--add node --node ex-mds
--add net --node ex-mds --nid ex-mds --nettype tcp
--add mds --node ex-mds --mds mds1 --fstype ldiskfs --dev /dev/hdb1 --size 0
```
BASIC LUSTRE CONFIGURATION

This section provides an overview of using the scripts shown above to set up a simple Lustre installation. The single system test (MDS, OSS, and client are all in the same system), which is the simplest Lustre installation, is a configuration in which all three subsystems execute on a single node. To set up, initialize, and start the Lustre file system for a single node system, follow steps 1–3 below.

1. Execute the script local.sh located in the lustre--<release-ver>.tar.gz under lustre/tests).
   - This script first executes a configuration script identified by a 'NAME' variable. This configuration script uses the lmc utility to generate a XML configuration file, which is then used by the lconf utility to do the actual system configuration.
   - You can change the size and location of these files by modifying the configuration script. (See Appendix A for an example of a local.sh file).
   - After the local.sh script is modified to meet your requirements, run that script. Running the script will create an XML file defined by the config variable in local.sh. (See Appendix B for a sample XML file).
   - When you run local.sh, you will see only the XML file—you will not receive any confirmation.
   - After you have the XML file, run lconf --reformat --verbose <name>.xml.
   - Finally, the lconf command mounts the Lustre file system at the mount point specified in the initial configuration script, the default used is /mnt/lustre.

You can the verify that the file system has been mounted from the output of df:

<table>
<thead>
<tr>
<th>File system</th>
<th>1K-blocks</th>
<th>Used</th>
<th>Available</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/ubd/0</td>
<td>1011928</td>
<td>362012</td>
<td>598512</td>
<td>38%</td>
<td>/</td>
</tr>
<tr>
<td>/dev/ubd/1</td>
<td>6048320</td>
<td>3953304</td>
<td>1787776</td>
<td>69%</td>
<td>/r</td>
</tr>
<tr>
<td>none</td>
<td>193712</td>
<td>16592</td>
<td>167120</td>
<td>10%</td>
<td>/mnt/lustre</td>
</tr>
</tbody>
</table>

**NOTE:** The output of the df command following the output of the script shows that a Lustre file system as been mounted on the mount point /mnt/lustre. The actual output of the script included with the Lustre installation may have changed due to enhancements or additional messages, but it should resemble the example.

You can also verify that the Lustre stack has been set up correctly by observing the output of find /proc/fs/lustre:
# find /proc/fs/lustre

/proc/fs/lustre
/proc/fs/lustre/llite
....
/proc/fs/lustre/ldlm/ldlm/ldlm_canceld/service_stats
/proc/fs/lustre/ldlm/ldlm/ldlm_cbd
/proc/fs/lustre/ldlm/ldlm/ldlm_cbd/service_stats

**NOTE:** The actual output may depend on what modules are being inserted and what internal Lustre (OBD) devices have been instantiated. Also, the file system statistics presented from /proc/fs/lustre are expected to be the same as those obtained from df.

1. Bring down a cluster and cleanup using script `llmountcleanup.sh`:

   Cleanup and unmounting of the file system can be done as shown:

   NAME=<local/lov> sh llmountcleanup.sh

2. Remount the file system using script `llrmount.sh`:

   Remounting can be done as shown:

   NAME=<local/lov> sh llrmount.sh

As described in earlier sections, Lustre uses clients, a metadata server, and object storage targets. It is possible to set up Lustre on either a single system or on multiple systems. The Lustre distribution comes with utilities that can be used to create configuration files easily and to set up Lustre for various configurations. Lustre uses three administrative utilities—lmc, lconf, and lctl—to configure nodes for any of these topologies. The lmc utility can be used to create configuration files in the form of XML files that describe a configuration. The lconf utility uses the information in this configuration file to invoke low-level configuration utility lctl to actually configure systems. Further details on these utilities can be found in their respective main pages. The complete configuration for the whole cluster should be kept in a single file. The same file is used on all the cluster nodes to configure the individual nodes.

**SETTING UP VARIOUS CONFIGURATIONS**

The next few sections describe the process of setting up a variety of configurations.

**SINGLE NODE CLIENT, MDS, AND TWO OSTs**

For the purpose of this test, we created three partitions on the server’s disk.

```bash
host:~ # fdisk -l /dev/hda
/dev/hda7                  2693 3500  6490228+  83 Linux  == MDS
/dev/hda8                  3501 6500  24097468+  83 Linux  == OST
/dev/hda9                  6501 9729  25936911  83 Linux  == OST
host:~#
```
This is a simple configuration script in which the client, MDS, and the OSTs are running on a single system. The lmc utility can be used to generate a configuration file for this as shown below. The size option is required only for the loopback devices; for real disk, the utility will extract the size from the device parameters.

```
#!/bin/bash

cfg="native.xml"
LMC="${LMC:-lmc}"
TMP=${TMP:-/tmp}
host=`hostname`
MOUNT=${MOUNT:-/mnt/lustre}
MOUNT2=${MOUNT2:-${MOUNT}2}
NETTYPE=${NETTYPE:-tcp}
JSIZE=${JSIZE:-0}
STRIPES_PER_OBJ=0 # 0 means stripe over all OSTs

# create nodes
${LMC} -o ${cfg} --add node --node $host || exit 10
${LMC} -m ${cfg} --add net --node $host --nid $host --nettype $NETTYPE || exit 11
${LMC} -m ${cfg} --add net --node client --nid '*' --nettype $NETTYPE || exit 12

# configure mds server
${LMC} -m ${cfg} --add mds --node $host --mds mds1 --fstype ldiskfs --dev /dev/hda7 --journal_size 400 || exit 20

# configure ost
${LMC} -m ${cfg} --add lov --lov lov1 --mds mds1 --stripe_sz $STRIPES_PER_OBJ --stripe_cnt $STRIPES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20

# create client config
${LMC} -m ${cfg} --add mtpt --node $host --path $MOUNT2 --mds mds1 --lov lov1

# Configuration using lconf
$ sh native.sh
$lconf --reformat native.xml
```

When this script is run, these commands create a native.xml file describing the specified configuration. The actual configuration could then be executed using the following command:

```
# Configuration using lconf
$ sh native.sh
$lconf --reformat native.xml
```

loading module: libcfs sradir None devdir libcfs
loading module: portals sradir None devdir portals
loading module: ksocknal sradir None devdir knals/socknal
loading module: lvfs srcdir None devdir lvfs
loading module: odbclass srcdir None devdir odbclass
This command loads all the required Lustre and Portals modules and also does all the low-level configuration of every device using `lctl`. The reformat option here is essential at least the first time to initialize the file systems on OSTs and MDSs. If it is used on any subsequent attempts to bring up the Lustre system, it will reinitialize the file systems.

```bash
# Create a test file
dd if=/dev/zero of=/mnt/lustre/test bs=1024 count=50
```

```bash
## Get stripe information for file
host:/mnt/lustre # ls -l test
total 242944
drwxr-xr-x  2 root root      4096 May  3 09:41 .
drwxr-xr-x  5 root root      4096 May  2 16:49 ..
-rw-r--r--  1 root root     51200 May  3 09:42 test

host:/mnt/lustre # lfs getstripe ./test
```

<table>
<thead>
<tr>
<th>obdidx</th>
<th>objid</th>
<th>objid</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0x1</td>
<td>0</td>
</tr>
</tbody>
</table>
This shows that the file has been striped over two OSTs.

Create a file that is striped to a single OST using the default stripe size:

```
host:/mnt/lustre # lfs setstripe stripe_1 0 -1 1
host:/mnt/lustre # lfs getstripe ./stripe_1
```

```
OBDS:
  0: ost1_UUID
  1: ost2_UUID
 ./stripe_1
   obdidx  objid  objid  group
     0      6  0x6     0
```

This shows that the file was tied to the first OST (identified by the `obdidx` field).

**BE CAREFUL!** If you override the stripe count and set it to 1 and then fill that OST, you will not be able to create files that try to stripe across that OST.

Set the stripe count on a directory:

```
host: mkdir dirtest
host:/mnt/lustre # lfs setstripe dirtest 0 -1 2
host:/mnt/lustre # lfs getstripe ./dirtest
```

```
OBDS:
  0: ost1_UUID
  1: ost2_UUID
 ./dirtest/
```

Currently, new directories created in a directory that specifies a default striping do not inherit the properties of the parent directory. This is expected to be fixed in version 1.4.2.
MULTIPLE NODES

Multiple Machines – Both OSTs and Clients

This section introduces the concepts of putting the OSTs, MDSs, and clients on separate machines. This is a basic configuration for a more realistic Lustre deployment. The configuration also specifies multiple Lustre file systems served by each node and mounted on the client.

For this example, Figure 1 illustrates the partition layout of the two OSS machines and the MDS machine.

We used two object storage server (OSS) machines each with three partitions that we used for OSTs. From the three partitions, we built three LOVs. The MDS and the clients are on different machines from the OSS and MDS machines. We used one MDS service per Lustre file system (one MDS service per LOV).

Node n01 was the MDS machine for all three MDS services (mds1, mds2, and mds3), nodes n02 and n03 were the OSS machines, and node n05 was the client. On each OSS, we made three partitions, /dev/hda6, /dev/hda7 (about 20 GB each) and /dev/hda8 (about 31 GB), which filled the rest of the hard drive.

**Step 1: Repartition hard drives in the OSS machines**

Our source for storage space was /dev/hda6 that was mounted as /scratch. If you have such a device mounted, first unmount it (for example, we unmounted /dev/hda6). Then we used the fdisk utility to delete /dev/hda6 and create a new /dev/hda6 and /dev/hda7 that are 20 GB (2432 blocks) each. The last partition, /dev/hda8, was made to use the remaining 31 GB (3897 blocks).

---

**Figure 1. OSS and Partition Layout for the Example Configuration.**
For node \textit{n02}, the following is a listing of the partition table.

\begin{verbatim}
For node n02, the following is a listing of the partition table.
n02:~ # fdisk -l

Device Boot Start      End      Blocks   Id  System
/dev/hda1   1        7       56227   83  Linux
/dev/hda2   8       645      5124735   83  Linux
/dev/hda3  646       837     1542240   82  Linux swap
/dev/hda4  838      9729    71424990    5  Extended
/dev/hda5  838      965     1028159+  83  Linux
/dev/hda6  966     3398    19543041   83  Linux
/dev/hda7 3399     5831    19543041   83  Linux
/dev/hda8  5832    9729     31310653+  83  Linux
\end{verbatim}

You also need to check \texttt{/etc/fstab} to remove the previous mount point for \texttt{/dev/hda6}.

\textbf{NOTE:} You will not add mount points for \texttt{/dev/hda6}, \texttt{/dev/hda7}, and \texttt{/dev/hda8} because Lustre will be using those partitions.

\textbf{Step 2: Edit the shell script, multilustre.sh}

This will create the Lustre XML file. Below is the shell script:

\begin{verbatim}
#!/bin/bash

LMC="${LMC:-lmc}"
NFSDIR="${NFSDIR:-/home/tmp_lustre}"
config="${config:-${NFSDIR}/multilustre.xml}"
host='hostname'
MOUNT=${MOUNT:-/mnt/lustre}
MOUNT2=${MOUNT2:-${MOUNT}2}
MOUNT3=${MOUNT3:-${MOUNT}3}
NETTYPE=${NETTYPE:-tcp}

STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPES_PER_OBJ=0  # 0 means stripe over all OSTs

# Create nodes
${LMC} -o $config --add node --node n01 || exit 10
${LMC} -m $config --add node --node n02 || exit 10
${LMC} -m $config --add node --node n03 || exit 10
${LMC} -m $config --add node --node generic-client || exit 10

# Add net
${LMC} -m $config --add net --node n01 --nid n01 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n02 --nid n02 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n03 --nid n03 --nettype $NETTYPE || exit 11

# Generic client definition
${LMC} -m $config --add net --node generic-client --nid '*' --nettype $NETTYPE || exit 12

# Configure mds server
${LMC} -m $config --add mds --node n01 --mds n01-mds1 --fstype ldiskfs --dev /dev/hda6 --journal_size 400 || exit 20
${LMC} -m $config --add mds --node n02 --mds n02-mds2 --fstype ldiskfs --dev /dev/hda7 --journal_size 400 || exit 20
\end{verbatim}
There are a few things to note in this script. First, near the top we defined the variable `NFSDIR` that is used as the location for this script.

```bash
LMC="\${LMC:-lmc}"
NFSDIR="\${NFSDIR:-/home/tmp_lustre}"
config="\${config:-\${NFSDIR}/multilustre.xml}"
host='\$hostname'
```

We also defined the variable `config` to point to the XML file produced from the `multilustre.sh`.

Next, we defined the mount point from which Lustre will be mounted with the variables `MOUNT`, `MOUNT2`, and `MOUNT3`.

```bash
MOUNT=\${MOUNT:-/mnt/lustre}
MOUNT2=\${MOUNT2:-\${MOUNT}2}
MOUNT3=\${MOUNT3:-\${MOUNT}3}
NETTYPE=\${NETTYPE:-tcp}
```
Note that we defined three mount points: /mnt/lustre, /mnt/lustre2, and /mnt/lustre3, one for each LOV. We also defined the networking protocol we used with the variable NETTYPE (in this case, TCP).

We also set the variable STRIPES_PER_OBJ to be 0 so that the data is striped across all of the OSTs.

STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPE_PER_OBJ=0 # 0 means stripe over all OSTs

Then we created the nodes using the Lustre command lmc (the definition of ${LMC} is located at the top of the script).

# Create nodes
${LMC} -o $config --add node --node n01 || exit 10
${LMC} -m $config --add node --node n02 || exit 10
${LMC} -m $config --add node --node n03 || exit 10

# Add net
${LMC} -m $config --add net --node n01 --nid n01 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n02 --nid n02 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n03 --nid n03 --nettype $NETTYPE || exit 11

The second set of commands creates the network for each of the parts of Lustre: the MDS and the two OSSs machines.

Next, we defined a generic client that allowed us to easily add clients after the creation of the MDS and OSTs.

# Generic client definition
${LMC} -m $config --add net --node generic-client --nid '*' --nettype $NETTYPE || exit 12

By defining a generic client, we could easily add clients without having to specify the specific name of the client node.

The next section of the script defines the MDS for each of the three LOVs. You need one MDS service per LOV. Figure 1 shows how the MDS services are mapped to the LOVs. In this case, we have three LOVs, so we need three MDS services.

# Configure mds server
${LMC} -m $config --add mds --node n01 --mds n01-mds1 --fstype ldiskfs --dev /dev/hda6 --journal_size 400 || exit 20
${LMC} -m $config --add mds --node n01 --mds n01-mds2 --fstype ldiskfs --dev /dev/hda7 --journal_size 400 || exit 20
${LMC} -m $config --add mds --node n01 --mds n01-mds3 --fstype ldiskfs --dev /dev/hda8 --journal_size 400 || exit 20

NOTE: We defined three MDS services (n01-mds1, n01-mds2, and n01-mds3) in which the first part of the name, n01, is the name of the node, and the last part is the MDS. Also, each MDS
service uses a separate partition on *n01*. So, *n01-mds1* uses /dev/hda6, *n01-mds2* uses /dev/hda7, and *n01-mds3* uses /dev/hda8.

In the next part of the script, we created each of the three LOVs that are defined in Figure 1.

```
# Create LOVs
$LMC -m $config --add lov --lov lov1 --mds n01-mds1 --stripe_sz
$STRIPE_BYTES  
     --stripe_cnt $STRIPE_BYTES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20
$LMC -m $config --add lov --lov lov2 --mds n01-mds2 --stripe_sz
$STRIPE_BYTES  
     --stripe_cnt $STRIPE_BYTES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20
$LMC -m $config --add lov --lov lov3 --mds n01-mds3 --stripe_sz
$STRIPE_BYTES  
     --stripe_cnt $STRIPE_BYTES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20
```

We used the --mds option to specify the MDS service for the LOV.

**BE CAREFUL!** Do not use a MDS for more than one LOV or your data can become corrupted. Also, be careful not to use the same disk device more than once in your configuration.

After creating the LOVs, we need to configure the OSTs.

```
# Configure OSTs
$LMC -m $config --add ost --node n02 --lov lov1 --fstype ldiskfs --dev /dev/hda6 --ost n02-ost1 || exit 30
$LMC -m $config --add ost --node n02 --lov lov2 --fstype ldiskfs --dev /dev/hda7 --ost n02-ost2 || exit 30
$LMC -m $config --add ost --node n02 --lov lov3 --fstype ldiskfs --dev /dev/hda8 --ost n02-ost3 || exit 30
$LMC -m $config --add ost --node n03 --lov lov1 --fstype ldiskfs --dev /dev/hda6 --ost n03-ost1 || exit 30
$LMC -m $config --add ost --node n03 --lov lov2 --fstype ldiskfs --dev /dev/hda7 --ost n03-ost2 || exit 30
$LMC -m $config --add ost --node n03 --lov lov3 --fstype ldiskfs --dev /dev/hda8 --ost n03-ost3 || exit 30
```

We have to run the `$LMC` command for each LOV for each OSS. Because we have two OSS machines and three LOVs, we have six `$LMC` commands. If you have a large number of OSS machines and a large number of LOVs, you will have a large number of `$LMC` commands.

**BE CAREFUL!** Plan your LOVs and naming convention before you run the script (see Best Practices section for a recommended naming convention).

The last piece of the script defines the generic client.

```
# Create client config
$LMC -m $config --add mtpt --node generic-client --path $MOUNT --mds n01-mds1 --lov lov1 \  
     $CLIENTOPT || exit 41
$LMC -m $config --add mtpt --node generic-client --path $MOUNT2 --mds n01-mds2 --lov lov2 \  
     $CLIENTOPT || exit 41
$LMC -m $config --add mtpt --node generic-client --path $MOUNT3 --mds n01-mds3 --lov lov3 \  
     $CLIENTOPT || exit 41
```
You must run the `${LMC}` command for each LOV. Also, you have to specify the MDS for the particular LOV as well as the mount point. We used shell variables for the mount point to make things easier to read.

**BE CAREFUL!** Do not to use a MDS for more than one LOV or your data can become corrupted. Also, be careful not to use the same mount point for more than one file system.

Now, run the script.

```
n01:~ # ./multilustre.sh
```

This script produces the XML file that will be used for the Lustre configuration. In this case, the XML file is called `multilustre.xml`.

**Step 3: Use lconf to reformat and start the MDS and OSTs**

After the Lustre XML configuration file has been created, you can reformat the MDS and the OSTs. This will also start the services on the MDS and OST machines.

**BE CAREFUL!** Reformatting will actually delete all the existing data and data structure from the OSTs.

**BE CAREFUL!** If you define multiple services (MDSs or OSTs) on a single node, --reformat will reformat all services on the node. This can be prevented with the more advanced --group options, which are outside of the scope of this document.

**BE CAREFUL!** Always start the OSTs first and the MDS last. We will discuss why you have to do this later, but for now, take this guidance as gospel. You have been warned!

Taking our own advice, we started the OST on `n02` first. Then we logged into node `n02` and used the `lconf` command as below. We used a NFS-mounted file system so the XML file, `multilustre.xml`, is available to all of the nodes. If you don’t do this, you will have to copy the XML file to each node.

```
n02:~ # lconf --reformat /home/tmp_lustre/multilustre.xml
loading module: libcfs srmdir None devdir libcfs
loading module: portals srmdir None devdir portals
loading module: ksocknal srmdir None devdir knals/socknal
loading module: lvfs srmdir None devdir lvfs
loading module: obdclass srmdir None devdir obdclass
loading module: ptrlpc srmdir None devdir ptrlpc
loading module: ost srmdir None devdir ost
loading module: ldiskfs srmdir None devdir ldiskfs
loading module: fsfilt_ldiskfs srmdir None devdir lvfs
loading module: obdfilter srmdir None devdir obdfilter
NETWORK: NET_n02_tcp NET_n02_tcp_UUID tcp n02 988
OST: n02-ost1 n02-ost1_UUID obdfilter /dev/hda6 0 ldiskfs no 0 0
OST: n02-ost1 n02-ost1_UUID obdfilter /dev/hda7 0 ldiskfs no 0 0
OST: n02-ost1 n02-ost1_UUID obdfilter /dev/hda8 0 ldiskfs no 0 0
OST mount options: errors=remount-ro
```
This last line from the lconf output is not an error. Its purpose is to tell you that if it had any errors, it would mount the partition as read-only (ro). Then, if you try to write to this OST, you will be unable to do so, indicating that you have a problem.

Run the same lconf command on all the OSS machines.

Finally, we started the MDS. In this case, we logged into the MDS node, n01, and ran the same lconf command.

n01:~ # lconf --reformat /home/tmp_lustre/multilustre.xml

Unlike other file systems, you don’t need to mount anything on the MDS or OSS machines.

**Step 4: Creating the clients and mounting the LOVs**

After the MDS and OSTs were up and functioning, we brought up the clients. We logged into the client node, in this case node n05, and used the lconf command to create n05 as a generic client for the three LOVs.

n05:~ # lconf --node generic-client

**NOTE:** Although not tested by the authors of this document, CFS suggests using the below command instead of the above one, as it is claimed to be faster on large number of clients and also as using lconf for mounting clients is being deprecated. Also suggested by CFS is to put the below command in the /etc/fstab file with an option of noauto which will further ease up starting clients and mounting Lustre file system on clients. More details can be found at the “Lustre mount using 0-config” section of the [https://wiki.clusterfs.com/lustre/LustreHowto web page.](https://wiki.clusterfs.com/lustre/LustreHowto)

```
mount -t lustre mdshost:/mdsname/client-profile <lustre-mount-point>
```

Following from where we left with the above lconf command, you can add clients anytime by using the n05:~ # lconf --node generic-client command on the new client node.

Do not use the --reformat option with this lconf because the LOVs are not local file systems and have already been formatted by Lustre. Also, performing the lconf command on the clients creates the mount point and mounts the LOVs for you.

A simple check to make sure everything was working and to look at what file systems that were mounted on the client node.

n05:~ # mount
```
/dev/hda2 on / type ext3 (rw)
proc on /proc type proc (rw)
tmpfs on /dev/shm type tmpfs (rw)
/dev/hdal on /boot type ext3 (rw)
/dev/hda5 on /var type ext3 (rw)
usbfss on /proc/bus/usb type usbfs (rw)
host:/home on /home_nfs type nfs (rw,addr=192.168.0.250)
automount(pid2854) on /home type autofs (rw,fd=5,pgrp=32854,minproto=2,maxproto=3) multilustre on /mnt/lustre3 type lustre_lite (rw,osc=lov3,mdc=MDC_n05_n01-mds3_MNT_generic-client_3)
```
multilustre on /mnt/lustre2 type lustre_lite (rw,osc=lov2,mdc=MDC_n05_n01-
mds2_MNT_generic-client_2)
multilustre on /mnt/lustre type lustre_lite (rw,osc=lov1,mdc=MDC_n05_n01-
mds1_MNT_generic-client)
Step 5: Testing the configurations
To test the configuration, use the directions in the Simple Lustre Configuration section of this document. Particularly, read about using the `lfs getstripe` and `lfs setstripe` commands to write files to the Lustre file systems. Be sure to test all three of the LOVs.

Adding OSTs to Existing File Systems
One configuration change that many users may want to do is to add space to an existing file system by adding OSTs. Simply, a straightforward OST addition to the existing file system does not work, but there are workarounds to this problem! We tried to do this by adding an OST on node `n04`. The clients could see the added space from the new OST, but the MDS did not recognize the added OST. The only way to get the MDS to recognize the OST was to reformat it – but then you will lose your data! It’s basically starting over.

**NOTE:** Although not tested by the authors of this document, CFS suggests the following procedure to add OSTs to existing file systems.
1. Stop and unmount the Lustre file system on all clients.
2. Use `lconf --cleanup <your.xml.file>` to stop MDS service on the MDS node.
3. Verify the MDS service is stopped on the MDS node by checking the `/proc/fs/lustre`. If this path is not existent on your system than the Lustre is stopped and unloaded on your system successfully. However, if this path is existent on your system, try `lconf --cleanup --force <your.xml.file>` and make sure the MDS services is stopped before continuing with the following steps.
4. Add the new OST using the appropriate `lconf` command with the `--reformat` option on the OSS and than the appropriate `lmc` command and the `lconf` command without the `--reformat` option on the MDS node.
5. Do a `lconf --write_conf <your.xml.file>` on the MDS node so MDS service can re-read the configuration updates.
6. Start the MDS service on the MDS node using the `lconf -v <your.xml.file>` command without the `--reformat` option.
7. Start and mount the Lustre file system on your clients as usual.

Other than the CFS suggested method above, there are several ways to add a new OST to an existing LOV without losing data. One way is to use a new OST, but it must be on a new OSS (a new service). You also need to stop the client and the MDS services to make this addition.

Adding an OST on a New OSS
For this experiment, we ran one MDS (mds1) on node `n01` and one OST (`/dev/hda6`) on node `n02` (the OSS). A generic client was run on node `n04`. Assume the configuration XML file is called `oldConf.xml`. This XML file is the current configuration that is running before the new OST was added. The following is the script we used for building the initial Lustre file system.

```bash
#!/bin/bash
# export PATH=`dirname $0`/../utils:$PATH
# config=${1:-local.xml}
LMC="${LMC:-lmc}"
```
#TMP=${TMP:-/tmp}
NFSDIR="${NFSDIR:-/home/tmp_lustre}"
config="${config:-${NFSDIR}/oldConf.xml}"
host= `hostname`'

# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname'}`
# MDSSIZE=${MDSSIZE:-400000}
# FSTYPE=${FSTYPE:-ext3}
MOUNT=${MOUNT:-/mnt/lustre}
NETTYPE=${NETTYPE:-tcp}
STRIPES_PER_OBJ=0 # 0 means stripe over all OSTs

$(LMC) -o $config --add node --node n01 || exit 10
$(LMC) -m $config --add node --node n02 || exit 10
$(LMC) -m $config --add node --node generic-client || exit 10

# Add net
$(LMC) -m $config --add net --node n01 --nid n01 --nettype $NETTYPE || exit 11
$(LMC) -m $config --add net --node n02 --nid n02 --nettype $NETTYPE || exit 11

# Generic client definition
$(LMC) -m $config --add net --node generic-client --nid '*' --nettype $NETTYPE || exit 12

# Configure mds server
$(LMC) -m $config --add mds --node n01 --mds n01-mds1 --fstype ldiskfs --dev /dev/hda6 --journal_size 400 || exit 20

# Create LOVs
$(LMC) -m $config --add lov --lov lov1 --mds n01-mds1 --stripe_sz $STRIPE_BYTES --stripe_cnt $STRIPES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20

# Configure OSTs
$(LMC) -m $config --add ost --node n02 --lov lov1 --fstype ldiskfs --dev /dev/hda6 --ost n02-ost1 || exit 30

# Create client config
$(LMC) -m $config --add mtpt --node generic-client --path $MOUNT --mds n01-mds1 --lov lov1 --clientopt $CLIENTOPT || exit 41

**Step 1: Stop the client.**
We logged into the client node and stopped the client using lconf.
n04:~ lconf -d --node generic-client oldConf.xml

**Step 2: Stop the MDS.**
We logged into the MDS node and stopped the MDS using lconf.
n01:~ lconf -d oldConf.xml

We did not shut down the exiting OSTs. They can be running or shutdown.

**Step 3: Create a new XML configuration file.**
The new XML file should reflect the added OST on the new OSS (new machine). We added a
new OSS, node n03. On that node, we added one OST, /dev/hda6. The following script,
newConf.sh, was used to create the new XML configuration file.

#!/bin/bash

# export PATH=`dirname $0`/../utils:$PATH

# config=${1:-local.xml}
LMC="${LMC:-lmc}"
#TMP=${TMP:-/tmp}
NFSDIR="${NFSDIR:-/home/tmp_lustre}"
config="${config:-${NFSDIR}/newConf.xml}"
host="hostname"

# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname`}
# MDSSIZE=${MDSSIZE:-400000}
# FSTYPE=${FSTYPE:-ext3}
MOUNT=${MOUNT:-/mnt/lustre}
NETTYPE=${NETTYPE:-tcp}

STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPES_PER_OBJ=0 # 0 means stripe over all OSTs

# Create nodes
${LMC} -o $config --add node --node n01 || exit 10
${LMC} -m $config --add node --node n02 || exit 10
${LMC} -m $config --add node --node n03 || exit 10
${LMC} -m $config --add node --node generic-client || exit 10

# Add net
${LMC} -m $config --add net --node n01 --nid n01 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n02 --nid n02 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n03 --nid n03 --nettype $NETTYPE || exit 11

# Generic client definition
${LMC} -m $config --add net --node generic-client --nid '*' --nettype $NETTYPE

# Configure mds server
${LMC} -m $config --add mds --node n01 --mds n01-mds1 --fstype ldiskfs \
   --dev /dev/hda6 --journal_size 400 || exit 20

# Create LOVs
${LMC} -m $config --add lov --lov lov1 --mds n01-mds1 --stripe_sz $STRIPE_BYTES \
   --stripe_cnt $STRIPES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20

# Configure OSTs
${LMC} -m $config --add ost --node n02 --lov lov1 \ 
   --fstype ldiskfs --dev /dev/hda6 --ost n02-ost1 || exit 30

${LMC} -m $config --add ost --node n03 --lov lov1 \ 
   --fstype ldiskfs --dev /dev/hda6 --ost n03-ost1 || exit 30

# Create client config
${LMC} -m $config --add mtpt --node generic-client --path $MOUNT --mds n01-mds1 \ 
   --lov lov1 $CLIENTOPT || exit 41
Notice that everything in this script is the same up to the line in the script, Create Nodes, except for the name of the XML file (near the top of the script with the variable `config`). Then the new node, `n03`, needs to be defined using `lmc` as well as the network for `n03`. Then under the section for defining the OSTs, we define a new OST, `n03-ost1`, using `/dev/hda6` on node `n03`. This OST is tied to the current LOV using the `--lov` option.

Run the script to create the new XML file.

**Step 4: Format the new OST on the new OSS.**
Only the new OST needs to be reformatted. So, we went to the new node, `n03`, and ran the following command.

```bash
n03:~ lconf --reformat newConf.xml
```

What we found is that if the OST resides on a new OSS (a new machine) then only that OST gets reformatted. This keeps the original data intact. However, if you try to use a new OST on an existing OSS, Lustre will format all of the OSTs.

**NOTE:** There is a way to use an existing OSS, and that is covered in Adding an OST on an Existing OSS section in this document.

**Step 5: Update the MDS.**
This is a key step to adding the new OST without losing any data on the exiting Lustre file system. We used the `lconf` command, but we used an option so it would add the OST to the LOV.

```bash
n01:~ lconf --write-conf newConf.xml
```

Notice that this command was run on the MDS node, `n01`.

**Step 6: Set up MDS service without reformatting.**
If you stopped the existing OST machine, you should start it before restarting the MDS. We did not shut down the existing OST, so we are skipping that step. However, if you did, then be sure to start the OST without the `--reformat` option.

```bash
lconf newConf.xml
```

Restarting the MDS is fairly easy. We used `lconf`, but without the `--reformat` option.

```bash
n01:~ lconf newConf.xml
```

**Step 7: Mount the client(s)**
To mount the clients, we just logged into the client node and use the `lconf` command as we did before.

```bash
n04:~ lconf --node generic-client newConf.xml
```
If you run the `df` command on the file system, you will see that the space has grown.

**Adding an OST to an Existing OSS**

For this experiment, we ran one MDS (mds1) on node n01 and one OST (/dev/hda6) on node n02 (the OSS). A generic client was run on node n04. We added a new OST that was located on the existing OSS (node n02). The key to making this work is to use a new name for the OSS for only the new OST. Otherwise, when you reformat the new OST, Lustre will reformat all of the OSTs (and some of them have data already on them).

Assume the configuration XML file is called `oldConf2.xml`. This XML file is the current configuration that is running before the new OST was added. We used the following for building the initial Lustre file system.

```bash
#!/bin/bash
# export PATH=`dirname $0`/../utils:$PATH
# config=${1:-local.xml}
LMC="${LMC:-lmc}"
#TMP=${TMP:-/tmp}
NFSDIR="${NFSDIR:-/home/tmp_lustre}"
config="${config:-${NFSDIR}/oldConf2.xml}"
host='hostname'

# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname`}  
# MDSSIZE=${MDSSIZE:-400000}  
# FSTYPE=${FSTYPE:-ext3}  
# MOUNT=${MOUNT:-/mnt/lustre}  
# NETTYPE=${NETTYPE:-tcp}  

STRIPE_BYTES=${STRIPE_BYTES:-1048576}  
STRIPES_PER_OBJ=0  # 0 means stripe over all OSTs

# Create nodes  
${LMC} -o $config --add node --node n01      || exit 10
${LMC} -m $config --add node --node n02      || exit 10
${LMC} -m $config --add node --node generic-client      || exit 10

# Add net  
${LMC} -m $config --add net --node n01 --nid n01 --nettype $NETTYPE      || exit 11
${LMC} -m $config --add net --node n02 --nid n02 --nettype $NETTYPE      || exit 11

# Generic client definition  
${LMC} -m $config --add net --node generic-client --nid '*' --nettype $NETTYPE      || exit 12

# Configure mds server  
${LMC} -m $config --add mds --node n01 --mds n01-mds1 --fstype ldiskfs \  --dev /dev/hda6 --journal_size 400      || exit 20

# Create LOVs  
${LMC} -m $config --add lov --lov lov1 --mds n01-mds1 --stripe_sz $STRIPE_BYTES \  --stripe_cnt $STRIPE_BYTES_PER_OBJ --stripe_pattern 0 $LOVOPT      || exit 20

# Configure OSTs  
${LMC} -m $config --add ost --node n02 --lov lov1 \  --fstype ldiskfs --dev /dev/hda6 --ost n02-ost1      || exit 30
```
Step 1: Stop the client.
We logged into the client node and stopped the client using lconf.
n04:~ lconf -d --node generic-client oldConf.xml

Step 2: Stop the MDS.
We logged into the MDS node and stopped the MDS using lconf.
n01:~ lconf -d oldConf.xml

We did not shut down the exiting OSTs. They can be running or shutdown.

Step 3: Create a new XML configuration file.
The new XML file should reflect the added OST on the existing OSS (new machine). We used an existing OSS, node n02. On that node we added one OST, /dev/hda7. We used the following script, newConf2.sh, to create the new XML configuration file.

#!/bin/bash
# export PATH=`dirname $0`/../utils:$PATH
# config=${1:-local.xml}
LMC="${LMC:-lmc}"
#TMP=${TMP:-/tmp}
NFSDIR="${NFSDIR:-/home/tmp_lustre}"
config="${config:-${NFSDIR}/newConf2.xml}"
host=`hostname`
# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname`}
# MDSSIZE=${MDSSIZE:-400000}
# FSTYPE=${FSTYPE:-ext3}
MOUNT=${MOUNT:-/mnt/lustre}
NETTYPE=${NETTYPE:-tcp}

STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPES_PER_OBJ=0 # 0 means stripe over all OSTs

# Create nodes
${LMC} -o $config --add node --node n01 || exit 10
${LMC} -m $config --add node --node n02 || exit 10
${LMC} -m $config --add node --node n02a || exit 10
${LMC} -m $config --add node --node generic-client || exit 10

# Add net
${LMC} -m $config --add net --node n01 --nid n01 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n02 --nid n02 --nettype $NETTYPE || exit 11
${LMC} -m $config --add net --node n02a --nid n02 --nettype $NETTYPE || exit 11

# Generic client definition
This script is different from the previous one. We defined a new node called n02a in the Create Nodes section of the script.

We also added a network to this node.

But notice that the nid is the same for this node.

Everything else is the same until we defined the OSTs:

Run the script to create the new XML file.

**Step 4: Format the new OST on the new OSS.**

Only the new OST needs to be reformatted. So, we went to the existing node, n02, and ran the following command.

Because Lustre thinks that the new OST is on a new OSS, n02a, it will reformat only that OST.

**Step 5: Update the MDS.**
This is a key step to adding the new OST without losing any data on the exiting Lustre file system. We used the lconf command, but we used an option so it would add the OST to the LOV.

```
n01:~ lconf --write-conf newConf2.xml
```

Notice that this command was run on the MDS node, n01.

**Step 6: Set up MDS service without reformatting.**

If you stopped the existing OST machine, restart it before restarting the MDS. We did not shut down the existing OST, so we are skipping that step. However, if you did, then be sure to start the OST without the --reformat option.

```
lconf newConf2.xml
```

Restarting the MDS is fairly easy. We used lconf but without the --reformat option.

```
n01:~ lconf newConf2.xml
```

**Step 7: Mount the client(s).**

To mount the clients, we just logged into the client node and use the lconf command as we did before.

```
n04:~ lconf --node generic-client newConf2.xml
```

If you run the df command on the file system, you will notice that the space has grown.

**Shutting Down Lustre**

At this point in the experiment, the Lustre system is up and running, so we will first shut it down before restarting. To shut down a Lustre file system, repeat the above steps in reverse order.

1. First, shut down Lustre on the client:

```
n05:~ lconf -d /home/tmp_lustre/multilustre.xml
```

**NOTE:** You must log into the client to run this command. To make life easier, you could use a parallel command shell such as pdsh to do this.

2. Next, shut down the MDS by running the exact same command, but on the MDS machine.

```
n01:~ lconf -d /home/tmp_lustre/multilustre.xml
```

3. Finally, run the same command for the OSS machines.

```
n03:~ lconf -d /home/tmp_lustre/multilustre.xml
n02:~ lconf -d /home/tmp_lustre/multilustre.xml
```

**BE CAREFUL!** If you shut down the parts of Lustre in the wrong order, you may not be able to shut down the Lustre file system cleanly and my need to reboot your nodes.
Starting Lustre

Lustre has a script in /etc/init.d for starting Lustre. It is a service that can be treated as any other service using chkconfig. So to start Lustre, assuming it has been closed correctly, just start the service:

```
# pdsh /etc/init.d/lustre start
```

This method has only been tested with SuSE 9.1 Linux; other distributions may be different. Because of the startup ordering requirements, it is not recommended that the MDS service start automatically on boot, but this is okay for the OSTs because they have no startup dependency.

PERFORMANCE CONSIDERATIONS

ESTIMATING HARDWARE PERFORMANCE

Before we begin to test the performance of our configuration, it is important to set the expectations. Considering a simple I/O test, in which applications on client nodes of the configuration read and write data to the Lustre file system, we can observe bottlenecks in the following places:

- the I/O rate of the test application itself,
- the aggregate off-node bandwidth of client nodes,
- the aggregate off-node bandwidth of OST nodes, and
- the aggregate bandwidth of the disk subsystem of OST nodes.

Disk technology varies widely in performance with low-end or low-power drives sustaining roughly 10 MB/s and high-end drives reaching burst rates of 80 MB/s. If you’re in a hurry, the raw bandwidth of an individual disk can be roughly estimated at 40 MB/s (give or take a factor of two).

Network performance observed by applications never reaches the rated signaling speed of network hardware, because, in addition to the application data payload, network packets contain control data (Ethernet frame, IP header, and other protocol headers). Thus, the performance of a network can approach but never exceed the “wire speed” of the network.

Other limiting factors include network tuning details, such as the TCP window size used for acknowledging the receipt of packets. More detail on the topic of networking can be found in either TCP/IP Illustrated (3 volumes) by W. Richard Stevens, Addison-Wesley, 2002, or Internetworking with TCP/IP (2 volumes) by Douglas Comer, Prentice-Hall, 2000. So how fast can we go? Network performance for large packets will asymptotically approach the limit of 1 byte of payload data per 8 bits on the wire. This means that for 100 Mbit Ethernet (FastE), performance will approach (but never exceed) the limit of 12.5 MB/s; the limit of Gigabit Ethernet (GigE) is 125 MB/s. A rule of thumb is to expect an application to observe throughput of 80–90% of the wire speed. During our tests we were able to show 11.8 MB/s on a 100-Mbit network (94.4% of the maximum of 12.5 MB/s)—a very impressive performance from Lustre.
Single OST
If you have a node that you want to use as an OST with a FastE connection, the bandwidth of a single disk drive will exceed the available network bandwidth. Adding disks to this node’s configuration will not improve performance; it will be capped at a maximum of 12.5 MB/s by the network bottleneck. Switching from FastE to GigE will remove the network bottleneck (the network max is now ~125 MB/s) and improve the performance observed by an application by a factor of 3.2 to 40 MB/s—the speed of the individual disk drive. After the network is upgraded, the bottleneck moves to the disk, at which point adding a second disk would again double the performance to 80 MB/s.

Multiple OSTs
The performance of multiple OSTs cannot be adequately estimated by looking at only the aggregate bandwidth of either the disks or the network connections. Each class of OST must be examined individually to determine where the bottlenecks occur. You can estimate the unidirectional performance of multiple OSTs by simply summing their individual maximums. For example, if you have 10 OSSs with two disks each—four with a GigE connection and six OSSs with two FastE connections each (load balanced)—the GigE nodes will not exceed 80 MB/s each (disk limited) and the FastE nodes will not exceed 25 MB/s each (network limited), for a total of 470 MB/s aggregate bandwidth. Compare this performance estimate with that a simple estimate produced by examining the number of disks (20 disks × 40 MB/s = 800 MB/s) or by examining the network connections (4 × GigE + 6 × 2 × FastE = 650 MB/s).

APPLICATION PERFORMANCE
As with the OSTs, the client nodes can be limited in several ways. While application issues may dominate actual performance demands in production, the rates observed during testing will depend on various factors. These factors include the size and number of network connections that the application can be used to access the storage, the size of the I/O requests made by the application, the pattern of I/O requests (sequential vs. random), and the type of request (reread vs. read vs. write vs. rewrite).

Additionally, because of the operating systems’ caches, until the size of the files being tested exceeds the memory pool size, the test is predominantly measuring memory performance rather than disk performance. A good rule of thumb is to test the file system with files that are at least twice the size of the memory pool. Thus, with a client node with 512 MB of memory, you could start testing with 1 GB files.

TESTING TOOLS
The simplest approach is to use the UNIX dd command; for example,

```
dd if=/dev/zero of=/lustre/testfile bs=16k count=65536
```

This will read /dev/zero and write a 1 GB file to /lustre/testfile as 65536 blocks of data with a blocksize of 16 KB/block. Then wrap this in timing to create a simple performance test:
time dd if=/dev/zero of=/lustre/testfile bs=16k count=65536
or
timex dd if=/dev/zero of=/lustre/testfile bs=16k count=65536

To map the performance of a file system, test the performance at a variety of block sizes. This can be automated with the tool `iozone` (http://www.iozone.org):

```shell
iozone -i0 -i1 -s 2g -r 16384 -f /lustre/testfile
```

This command will perform write/rewrite (-i0) and read/reread (-i1) tests on a 2 GB file (-s 2g) with a block size of 16384 K. (Jeff K., please confirm this is okay without the k after the number) using the file /lustre/testfile. The output might look something like the following sample:

```
iozone  -a -i0 -i1 -s 2048m -r 16384 -f iozone.$$_.tmp
Iozone: Performance Test of File I/O
   Version $Revision: 3.239 $
   Compiled for 32 bit mode.
   Build: linux

Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins, Al Slater, Scott Rhine, Mike Wisner, Ken Goss, Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,Randy Dunlap, Mark Montague, Dan Million, Jean-Marc Zucconi, Jeff Blomberg,Erik Habbinga, Kris Strecker, Walter Wong.

Run began: Wed May 4 16:05:07 2005
Auto Mode
File size set to 2097152 KB
Record Size 16384 KB
Command line used: iozone -a -i0 -i1 -s 2g -r 16384 -f /lustre/iozone.3182.tmp
Output is in Kbytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 Kbytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

   KB  reclen   write rewrite    read    reread
2097152   16384   11802   11850    11474    11475

iozone test complete.
```

The `iozone` utility also has an `-a` switch, which runs the same tests over a variety of block sizes. If you don’t mind waiting a bit, try the following:

```shell
iozone -az -i0 -i1 -s 2g -f /lustre/testfile
```

You can run `iozone` on several clients simultaneously to obtain a rough measure of aggregate I/O performance.

**BE CAREFUL!** If your bottleneck is the network, full duplex connections may affect your results by allowing one instance of `iozone` to read from the Lustre file system while another is writing to
it. In this case, the simple sum of the bandwidths for the reading iozone and the writing iozone will exceed the estimated unidirectional performance estimated using the technique described above.

Another tool which can be used to test the performance of our new file system is bonnie++ (http://www.coker.com.au/bonnie++/). The tool, bonnie++ has several nice features, including the ability to run a basic metadata test to examine file creation and deletion speeds, and the ability to generate results which are easily imported to a spreadsheet. However, in its default mode, it also runs a character I/O test, which is time consuming.

We are primarily interested in validating the correctness of our installation by comparing it to our best estimate of what we expected, therefore we disabled the character I/O test. Below is an example of the output of bonnie++ from a command that launches bonnie++ with the system label LustreCluster to run in the /mnt/lustre directory as the root user with a file size of 2048 MB, no write buffering, and no character I/O test.

```
./bonnie++ -m LustreCluster -d /mnt/lustre -u root -s 2048 -b -f | tee BONNIE.OUT
Using uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...
Create files in sequential order...Can't sync directory, turning off dir-sync.
done.
Stat files in sequential order...done.
Delete files in sequential order...done.
Create files in random order...done.
Stat files in random order...done.
Delete files in random order...done.
Version  1.03       ------Sequential Output------ --Sequential Input-- --
Random-
   -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --
Seeks--
Machine Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP /sec
%CP
LustreCluster  2G      11784   11 10881  31          11460  19  205.0
4
------Sequential Create----- --------Random Create--------
   -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
files /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP
16 48 1 661 28 715 5 45 0 673 29 715 5
LustreCluster,2G,,11784,11,10881,31,,11460,19,205.0,4,16,48,1,661,28,715,5,
45,0,673,29,715,5
```

As with iozone, bonnie++ was basically able to achieve 94% of wire speed from a single client on our 100-Mbit switched network. As with iozone, bonnie++ can also be run on multiple clients with the same caveats about aggregating read and write performance across multiple clients on a full duplex connection.

**MONITORING AND ADMINISTERING WITH LUSTRE MANAGER**
The Lustre Management Tool provides an overview of what is happening on all servers and clients. It provides early warnings, recent throughput, space utilization, and all other events and information that you need to know to monitor the servers.

**INSTALLATION AND CONFIGURATION**

**Lustre Manager Web Interface**

1. The Lustre Manager RPM can be obtained from the Cluster File System download area. After it is downloaded, it can be installed with the following command:

   ```bash
   rpm -ivh lustre-manager*.rpm
   ```

2. You must first install the following dependencies for the Lustre Manager interface:

   ```bash
   python and python-gd.
   ```

3. The python-gd module can be found via anonymous at ftp://ftp.clusterfs.com/pub/people/shaver/

4. Start the service by executing:

   ```bash
   /var/lib/lustre-manager/LustreManagerMaster. If you are using a Red Hat system, you can just use the service lustre-manager start command.
   ```

5. Find the administrator password at the top of the /var/lib/lustre-manager/log/manager.log file. You can change the password inside the web interface. This password is created the first time it is executed.

**LUSTRE MANAGER COLLECTOR**

The collector collects data from the MDSs, OSSs, and client systems and then passes that data to the manager so it can be correlated and displayed on the web site. The lustre-manager-collector RPMs can be downloaded from Lustre.

1. After it is downloaded, it can be installed with the following command:

   ```bash
   rpm -ivh lustre-manager-collector*.rpm. There were no dependencies for installing this RPM.
   ```

2. Set up the config file to enable the collector to work. This file (/etc/sysconfig/lustre-manager-collector) contains a single line telling the collectors on which host the Lustre monitor daemon is running.

   ```bash
   LMD_MONITOR_HOST="host"
   ```

3. After the collector has been configured, it can be started by executing:

   ```bash
   /etc/init.d/lmd start
   ```
Connecting to the Lustre Manager
To connect to the Lustre Manager, point the web browser to Port 8000 on the server that is running the Lustre Manager. In the figures shown in the examples below, the port is located at http://192.168.1.1:8000.

Importing the Lustre Configuration into the Lustre Manager
Three methods are available for installing the Lustre configuration into the Lustre Manager. The first is to use the Lustre Manager web interface to define the entire cluster configuration. This method is identical to the method used by the Lustre Wizard, with the exception of it being a graphical user interface (GUI) (see Figure 2). The second method is to take the existing XML file from a Lustre cluster and import it into the Manager (see Figure 3). The third method is to take the XML file from an existing cluster and copy it into /var/lib/lustre-manager/configs/ directory.

Figure 2. The Lustre Manager web interface can define the entire cluster configuration—the same method as used by the Lustre Wizard.

After everything is complete, you can generate the configuration. This will produce an XML file like the lmc command output.

NOTE: The available file system type is ext3 and not ldiskfs as it should be for 2.6 kernels.
The second method for installing the configuration is to simply import an existing XML file. To do this, select the Configurations tab on the GUI.

Figure 3. The second method for installing the configuration is to simply import an existing XML file.
Next, select the Import XML button to obtain the import screen. Enter the full path to the XML file and then click on the Import XML button to import the configuration (see Figure 4).

Figure 4. Import XML screen.
By selecting the Systems tab, you can see the nodes in your cluster, as well as some of the statistics about the server. This screen shows CPU usage, memory usage, and network usage both inbound and outbound for each of the servers in the cluster (see Figure 5).

Figure 5. The Systems tab shows various usages for each server in the cluster.
The Services tab displays the status of all of the Lustre services in the defined cluster. You can stop and start the MDT and OST services from the web interface if you have administrator-level access (see Figure 6).

Figure 6. The Services tab displays the status of all of the Lustre services.
The Overview tab displays the overview of the MDTs and OSTs (see Figure 7). It displays the amount of disk space available per OST and MDT and how many files can still be created in each MDT and OST.

Figure 7. The Overview tab displays the overview of the MDTs and OSTs.
The Performance tab displays information about the performance of the MDTs and OSTs (see Figure 8). It shows the throughput of the reads and writes in Mb/sec for each MDT and OST.

Figure 8. The Performance tab displays information about the performance of the MDTs and OSTs
The Lustre Manager allows you to add accounts to use the web interface. The Lustre Manager web interface provides a nice way to get an overview of the entire Lustre. It also allows you to give the operations staff accounts in which they can view the status and performance of the services, but they will not be able to stop and start any of the services (see Figure 9).

Figure 9. The operations staff can access accounts so they can view the status and performance of the services.
BEST PRACTICES

NAMING CONVENTIONS
It is best to keep a generic naming convention for the MDS and OSTs. This will make failover configuration much easier when you configure those components of Lustre. The client name should remain a generic name so when you start the client, you just need to specify the generic name as shown in the client section below.

MDS:
- Suggested name: mds<fsname>, where fsname is usually the file system mount point (e.g., mds-scratch, mds-usr_local, mds-home)
- The mds line in the configuration script would look like:
  
  ... --add mds --node <hostname> --mds mds1 ...

OST:
- Suggested name: ost<fsname>-# (e.g., ost-scratch-1)
- The ost line in the configuration script would look like:
  
  ... --add ost --node <hostname> --ost ost1 ...

LOV:
- Suggested name: lov<fsname> (e.g., lov-scratch, lov-usr_local)

Client:
- Suggested name: client -<fsname> (e.g., client-scratch)
- The client line in the configuration script would look like:
  
  ... --add net --node client --nid '*' ...
- When you start Lustre on the client, you just need to specify
  
  lconf --node client <xml file>

In summary, inconsistent naming will cause much havoc as your Lustre implementation continues to grow.

XML/SHL CONFIGURATION FILE MANAGEMENT
Always keep copies of your XML and sh configuration files. These should be kept on separate devices in case of hardware/administrative errors. If you are planning to regenerate the XML file, make a back up the original first. Because these files manage the layout of the Lustre file system, it is best to keep track of the history of changes made to the sh file, when those changes were made, and who made them. A possible solution is to use the concurrent version system (CVS) system. Also, although not tested by the authors of this document, it was pointed out by the CFS that using the lconf utility to read configuration files from a web server can ease the
centralization task. According to the CFS, replacing the `lconf <path.to.your.xml.file>` with `lconf http://URL-to-your-xml-configuration-file-on-your-web-server` is more convenient to centralize the configuration process, especially if another shared file system is not available for the system (e.g. NFS).

**MDS/OSS/CLIENT CONFIGURATION**

It is advisable to keep the MDS, OSS, and client on separate systems. This helps prevent system contention.

**PUBLISH YOUR EXPERIENCES**

Lustre documentation still lacks depth and clarity. Contributing to the overall documentation effort will help the Lustre community.

**LDAP SUPPORT**

CFS recommends not using LDAP integration with Lustre. CFS plans to phase out LDAP in future Lustre releases.

**LIBLUSTRE LIBRARY**

In version 1.4.1, CFS does not support Liblustre Library for platforms other than the Cray. It will not be available in Lustre version 1.4.2 either.

**ADMINISTRATIVE TIPS**

**Log Files**

Logging for Lustre is done in the standard operating system `/syslog/messages` file. If you have a problem, it is best to check the following items in this order:

1. Check client log file
2. Check OSS log file
3. Check MDS log file

If an error needs to be submitted to CFS, it is best to have the error entries from all of the above files.

**Useful Commands**

- `lconf`: This is used to configure a node following the directives that are in the XML file. There will be a single configuration file for all nodes in a single Lustre cluster. Some options for `lconf` are listed below.
  - `-d` or `--cleanup`
    - This is used to shut down Lustre, un-configuring the node. This will unload the kernel modules.
  - `--reformat`
This is used when initially setting up a Lustre file system. This is destructive, so do not run this command on a file system on which you want to maintain the data (e.g., a production system).

- **--write_conf**
  This saves all client configurations on the MDS. It is useful for reconfiguring the existing system without losing data (e.g., adding an OST to an existing LOV).

- **lmc**
  This adds configuration data to the configuration file.

- **lfs**
  This utility can be used to create a new file with a specific striping pattern, determine the default striping pattern, and gather the extended attributes (object numbers and location) for a specific file. Some options for lfs are listed below:
  - **Setstripe**: Creates a new file with a specific striping pattern, or changes the default striping pattern on new files created in a directory. In Lustre 1.4.1, a subdirectory with a specified stripe count will not inherit its parent directory’s stripe count.
  - **Getstripe**: Retrieves the current stripe count of a file.
  - **Check**: Allows you to look at the status of MDS/OSTs.
  - **Help**: Shows all of the available options.
APPENDIX A: SAMPLE  

This example uses files on already existing file systems. This sample demonstrates that Lustre is functioning correctly, even though it is not actually using device files.

#!/bin/bash

# export PATH=`dirname $0`/../utils:$PATH

# config=${1:-local.xml}
config=`hostname`.xml
LMC=${LMC:-lmc}
TMP=${TMP:-/tmp}
host=`hostname`

# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname'}`
# MDSSIZE=${MDSSIZE:-400000}
# FSTYPE=${FSTYPE:-ext3}
MOUNT=${MOUNT:-/mnt/lustre}
MOUNT2=${MOUNT2:-${MOUNT}2}
NETTYPE=${NETTYPE:-tcp}

# OSTDEV=${OSTDEV:-$TMP/ost1-`hostname'}`
# OSTSIZE=${OSTSIZE:-400000}

# specific journal size for the ost, in MB
JSIZE=${JSIZE:-0}
# [ "$JSIZE" -gt 0 ] && JARG="--journal_size $JSIZE"
# MDSSIZE=${MDSSIZE:-0}
# [ "$MDSISIZE" -gt 0 ] && IARG="--inode_size $MDSISIZE"
STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPES_PER_OBJ=0       # 0 means stripe over all OSTs

# rm -f $config

# create nodes
${LMC} -o $config --add node --node $host || exit 10
${LMC} -m $config --add net --node $host --nid $host --nettype $NETTYPE ||
exit 11
${LMC} -m $config --add net --node client --nid '*' --nettype $NETTYPE || exit 12

# configure mds server
${LMC} -m $config --add mds --node $host --mds mds1 --fstype ldiskfs \
    --dev /tmp/mds1 --size 50000 --journal_size 400 || exit 20

# configure ost
${LMC} -m $config --add lov --lov lov1 --mds mds1 --stripe_sz $STRIPE_BYTES \
    --stripe_cnt $STRIPE_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20
${LMC} -m $config --add ost --node $host --lov lov1 \
    --fstype ldiskfs --dev /scratch/ost1 --size 400000 $JARG $OSTOPT ||
exit 30

# create client config
${LMC} -m $config --add mtpt --node $host --path $MOUNT --mds mds1 --lov lov1 \
    $CLIENTOPT || exit 40
# ${LMC} --add mtpt --node client --path $MOUNT2 --mds mds1 --lov lov1 \n    $CLIENTOPT || exit 41
The previous example is for a single node running as an MDS/OSS/client. The following explains the above `local.sh` file.

```
${TMP:-/tmp}
```

This means that if the environment variable TMP is not set, then set it to “/tmp”.

```
config="`hostname`.xml"
LMC="${LMC:-lmc}"
```

The command `config` will set the name of the XML file. This can be changed to whatever you want. The utility, `lmc`, is provided from the lustre-lite-utils RPM package. The utility, `lmc`, is used to add configuration data to a configuration file.

```
# MDSDEV=${MDSDEV:-$TMP/mds1-`hostname`}
# MDSSIZE=${MDSSIZE:-400000}
# FSTYPE=${FSTYPE:-ext3}
MOUNT=${MOUNT:-/mnt/lustre}
NETTYPE=${NETTYPE:-tcp}
```

In this version of the script, MDSDEV, MDSSIZE, and FSTYPE were commented out. These could be set to

```
MDSDEV=${MDSDEV:-/tmp/mds1}
MDSSIZE=${MDSSIZE:-50000}  # note this size is in KBytes.
FSTYPE=${FSTYPE:-ldiskfs}
```

We used `ldiskfs` (CFS’s enhanced version of ext3) because SuSE 2.6 ldiskfs has features required by Lustre. Lustre's ext3 file system does not have these features. On a 2.4 kernel, the ext3 file system is modified directly, so it should be used there.

MOUNT defines the Lustre mount point.
NETTYPE has four options, tcp, elan, and gm. This determined the type of network interconnect. TCP is self explanatory, elan is for Quadrics, and gm is for Myrinet.

```
# specific journal size for the ost, in MB
JSIZE=${JSIZE:-0}
# [ "$JSIZE" -gt 0 ] && JARG="--journal_size $JSIZE"
# MDSSIZE=${MDSSIZE:-0}
# [ "$MDSSIZE" -gt 0 ] && IARG="--inode_size $MDSSIZE"

STRIPE_BYTES=${STRIPE_BYTES:-1048576}
STRIPES_PER_OBJ=0  # 0 means stripe over all OSTs
```

JSIZE or `journal_size` is optional. The value of JSIZE should be recognized by `mkfs`. If this variable is not set, the default journal size will be used. MDSSIZE is only needed for a loop device. STRIPE_BYTES is the stripe size. In this case, we set our stripe size to 1 MB. STRIPES_PER_OBJ is the stripe count. We set STRIPES_PER_OBJ to the total number of OSTs.

```
# create nodes
${LMC} -o $config --add node --node $host  || exit 10
${LMC} -m $config --add net --node $host --nid $host --nettype $NETTYPE  || exit 11
${LMC} -m $config --add net --node client --nid '*' --nettype $NETTYPE  || exit 12
```
In creating the nodes for the Lustre configuration, the 
\texttt{-o} will tell \texttt{lmc} to overwrite the existing 
\texttt{config} file and add a node for \texttt{host}. The \texttt{-m} tells \texttt{lmc} to add a new network device descriptor 
to the \texttt{config} file. In the first \texttt{add net} line, we added a network device descriptor for the node 
\texttt{host}, with a network id (\textit{nid}) of \texttt{host}. This \texttt{nid} must be unique for all nodes. And \texttt{nettype} 
describes the interconnect as defined earlier.

\begin{verbatim}
# configure mds server
${LMC} -m $config --add mds --node $host --mds mds1 --fstype ldiskfs 
   --dev /tmp/mds1 --size 50000 --journal_size 400 || exit 20
\end{verbatim}

We defined the MDS services \texttt{--mds1}, and indicated that this is a device pointing to a file located 
in the \texttt{/tmp} directory.

\begin{verbatim}
# configure ost
${LMC} -m $config --add lov --lov lov1 --mds mds1 --stripe_sz $STRIPE_BYTES 
   --stripe_cnt $STRIPES_PER_OBJ --stripe_pattern 0 $LOVOPT || exit 20
${LMC} -m $config --add ost --node $host --lov lov1 
   --fstype ldiskfs --dev /scratch/ost1 --size 400000 $JARG $OSTOPT ||
exit 30
\end{verbatim}

We configured one LOV and one OST. Unlike the configuration shown in the Multiple Nodes 
section of this document, this configuration uses a loop device, \texttt{/scratch/ost1} instead of a 
physical device.

\begin{verbatim}
# create client config
${LMC} -m $config --add mtpt --node $host --path $MOUNT --mds mds1 --lov lov1 
   $CLIENTOPT || exit 40
\end{verbatim}

The client mounts the Lustre file system using above line. The mount point is \texttt{--path}, using the 
\texttt{lov1} volume. The client is located on the same machine as the MDS and OST.
APPENDIX B: SAMPLE XML FILE

<?xml version="1.0" encoding="utf-8"?>
<lustre version="2003070801">
  <ldlm name="ldlm" uuid="ldlm_UUID"/>
  <node name="host" uuid="host_UUID">
    <profile_ref uuidref="PROFILE_host_UUID"/>
    <network name="NET_host_tcp" nettype="tcp" uuid="NET_host_tcp_UUID">
      <nid>host</nid>
      <clusterid>0</clusterid>
      <port>988</port>
    </network>
  </node>
  <profile name="PROFILE_host" uuid="PROFILE_host_UUID">
    <ldlm_ref uuidref="ldlm_UUID"/>
    <network_ref uuidref="NET_host_tcp_UUID"/>
    <osd_ref uuidref="OSD_OST_host_host_UUID"/>
    <mountpoint_ref uuidref="MNT_host_UUID"/>
  </profile>
  <node name="client" uuid="client_UUID">
    <profile_ref uuidref="PROFILE_client_UUID"/>
    <network name="NET_client_tcp" nettype="tcp" uuid="NET_client_tcp_UUID">
      <nid>*</nid>
      <clusterid>0</clusterid>
      <port>988</port>
    </network>
  </node>
  <profile name="PROFILE_client" uuid="PROFILE_client_UUID">
    <ldlm_ref uuidref="ldlm_UUID"/>
    <network_ref uuidref="NET_client_tcp_UUID"/>
  </profile>
  <mds name="mds1" uuid="mds1_UUID">
    <active_ref uuidref="MDD_mds1_host_UUID"/>
    <lovconfig_ref uuidref="LVCFG_lov1_UUID"/>
    <filesystem_ref uuidref="FS_fsname_UUID"/>
  </mds>
  <mdsdev name="MDD_mds1_host" uuid="MDD_mds1_host_UUID">
    <fstype>ldiskfs</fstype>
    <devpath>/tmp/mds1</devpath>
    <autoformat>no</autoformat>
    <devsize>50000</devsize>
    <journalsize>400</journalsize>
    <inodesize>0</inodesize>
    <node_ref uuidref="host_UUID"/>
    <target_ref uuidref="mds1_UUID"/>
  </mdsdev>
  <lov name="lov1" stripecount="0" stripepattern="0" stripesize="1048576" uuid="lov1_UUID">
    <mds_ref uuidref="mds1_UUID"/>
    <obd_ref uuidref="OST_host_UUID"/>
  </lov>
</lustre>
APPENDIX C: BUILDING THE LUSTRE KERNEL FROM SOURCE

Lustre binary RPMs are available from CFS. For some applications, you may want to build Lustre from sources; for example, you may want to integrate a new network driver with Lustre or simply add a new module to the kernel. Lustre consists of changes to kernel code, a number of kernel modules, and user-level utilities. We believe that CFS plans to organize the Lustre modules into one single kernel module at some time in the future. Currently, Lustre is supported on several kernels. If you are building Lustre from source, you need to build a patched Linux kernel, the kernel modules, and the user-level utilities. Patching the kernel can be quite involved, especially if you are trying to apply the patches to a kernel source that differs from the original kernel source used to generate the patches. That is the major difficulty of building from source.

In this section, we will describe building Lustre (1) with a kernel prepatched by CFS, (2) with a Red Hat or SuSE Linux release, and (3) from a vanilla kernel source. By vanilla, we mean main-line Linux sources released by Linus and available from, for example, http://kernel.org.

In this document we use Lustre 1.4.1 and different versions of Linux 2.6 kernel source trees.

BUILDING LUSTRE FROM CFS-SUPPLIED PREPATCHED KERNEL

In certain situations it may be necessary to build a kernel from scratch as described above. In order to accomplish this, it is necessary to build extra modules from sources using the kernels source tree.

The easiest way to build the Linux kernel is to start with a prepatched version of the kernel, which can be downloaded from the Cluster File Systems web site: http://www.clusterfs.com.

From this web site, a variety of prepatched kernel versions are available from various Linux distributions as well as for various hardware architectures such as x86, i686, x86_64 and ia64. In addition, a variety of Lustre release levels are available. Below is a snapshot from the CFS web site.

lustre-1.4.1.tar.gz 24-Mar-2005 19:28  3.6M
rhel-2.4-i686/ 24-Mar-2005 12:27 -
rhel-2.4-ia64/ 24-Mar-2005 12:28 -
rhel-2.4-x86_64/ 24-Mar-2005 12:27 -
rhel-2.6-i686/ 24-Mar-2005 12:40 -
rhel-2.6-x86_64/ 24-Mar-2005 12:41 -
sles-2.6-ia64/ 24-Mar-2005 12:32 -
sles-2.6-x86_64/ 24-Mar-2005 12:32 -

After downloading the RPM for the chosen release of Lustre, you will also need to download the source for Lustre itself. This includes the source for Lustre commands, drivers, utilities, and example scripts. In the case of the Lustre 1.4.1 release, it is the gzipped tar file lustre-1.4.1.tar.gz. Below is an example using the SuSE source release of Linux 2.6.5-7 for an Intel Xeon-based system.
In this case, the source is installed in `/usr/src/linux-2.6.5-7.141_lustre.1.4.1` directory. To build this, do the following:

```
host# cd /usr/src/linux-2.6.5-7.141_lustre.1.4.1
host# make distclean
```

At this point, either copy a version of the kernel configuration file to `.config` or use `make menuconfig` to assist in making the `.config` file. If you wish, you can start with a configuration file in the `/boot` directory for the currently running system and make any necessary changes.

```
host# cp my_saved_config .config
```

You can give the kernel you are building a special version number. This can be done by modifying the file `Makefile` and changing the variable `EXTRAVERSION`.

```
host# vi Makefile
```

After making this change, you are ready to build the kernel. The following example will build the kernel and all defined modules and then install them. The kernel modules will be installed in `/lib/modules/2.6.5-7.141_lustre.1.4.1custom`.

```
host# make oldconfig dep bzImage modules modules_install install
```

It is also possible to build a new kernel RPM with the `make rpm` command that can be installed with the `rpm` tool. After this is completed, install a kernel in `/boot/vmlinuz-2.6.5-7.141_lustre.1.4.1custom` and then modify the `/boot/grub/menu.lst` or `/boot/grub/grub.conf` file. By default, the install process links the newly built kernel to `/boot/vmlinuz`, which may be the default in your `menu.lst` or `grub.conf` file. It may be desired to make a separate entry for the newly built kernel in these files in the event there is a problem with the newly built kernel. Although it is not necessary, a reboot with the new kernel for a sanity check is advisable. Good luck...

```
host# vi /boot/grub/menu.lst
host# reboot
```

When the system reboots with the new Lustre-patched kernel, it will contain the entire Lustre-ready kernel infrastructure, but not the Lustre loadable modules. After the successful restart of the system, install the Lustre specific modules, commands, utilities, and examples. You will need to gunzip and extract the tar file. In this case, we extracted the source files in `/usr/src`.

```
host# cd /usr/src
host# gunzip /location_of_file/lustre-1.4.1.tar.gz
```
Now begin building the source. First run `configure` then do the `make` itself. As part of the `configure` run, specify the location of the Linux source you are building against.

```bash
cd /usr/src/lustre-1.4.1
./configure --with-linux=/usr/src/linux-2.6.5-7.141_lustre.1.4.1
make install
```

After this has successfully run, the Lustre modules and commands will have been installed. Next, either run `depmod` to setup the module dependencies or reboot the system. Finally, `configure` and customize the system as described in earlier chapters.

### Patching SuSE or Red Hat Kernel Source and Building Lustre

It is possible to patch and build a Linux kernel depending on the source release from which you are building. Most of the Lustre kernel patches are based on the SuSE and Red Hat releases. For the best chance of success, it would be advisable to use either one of these distributions and one of the more common and stable releases, for example, the SuSE 2.6.5-7 release mentioned above. It should be noted that the SuSE releases, especially the newer 2.6 releases, have started incorporating the Lustre kernel patches. This can also make it somewhat more difficult because the patches and series of patches may already be installed, or they may not point to the appropriate lines in the source, thus causing various problems for the build. It may be necessary to look at all patches associated with a particular release to see which ones pertain to that release and whether or not the patched kernel lines match up. The bottom line is to match the kernel release with the patch list for that kernel from CFS, who notes that there is an incremental patch series for the SuSE 2.6 kernel called 2.6-susi-Inxi.series that should be used to patch the latest SuSE-released kernel.

So we started with the desired kernel source. In this case, we based it on the SuSE 2.6.5-7.147 source RPM.

```bash
rpm -ivh kernel-source-2.6.5-7.147.i586.rpm
```

Again, install the Lustre source as shown above.

```bash
cd /usr/src
```

```bash
gunzip /location_of_file/lustre-1.4.1.tar.gz
tar -xvf /location_of_file/lustre-1.4.1.tar
```
releases that contain the needed kernel patches from the patches subdirectory that are used by the quilt utility during the installation of patches.

Since some releases (for example some of the SuSE releases) contain some of the Lustre kernel patches, it may become necessary to match up the series file to the desired kernel release. Then, walk through each of the patches in that file to see which patches are already installed and which are not needed. Also, check to see how accurately the patches that are missing from the kernel source tree align with the lines to be modified in the current source.

To install the patches using quilt, use the following as an example:

```
host# cd /usr/src/linux-2.6.5-7.147
host# quilt setup -l /usr/src/lustre-1.4.1/lustre/kernel_patches/series/desired_patch_series -d /usr/src/lustre-1.4.1/lustre/kernel_patches/patches
host# quilt push -av
```

After all kernel patches have been installed, follow the above sequence to build the Linux kernel then the Lustre modules and commands:

```
host# cd /usr/src/linux-2.6.5-7.147
host# make distclean
host# cp my_saved_config .config
host# cp my_saved_config .config
host# make oldconfig dep bzImage modules modules_install install
host# vi /boot/grub/menu.lst
host# reboot
host# cd /usr/src/lustre-1.4.1
host# ./configure --with-linux=/usr/src/linux-2.6.5-7.147
host# make install
```

During the Lustre build, you will notice that the appropriate ext3 source files get copied to a target directory, patched with the ldiskfs patches then built along with the other Lustre commands.

**BUILDING LUSTRE FROM VANILLA SOURCES**

Evan Felix of Pacific Northwest National Laboratory (PNNL) has created patches to the 2.6.9 and 2.6.8 vanilla Linux kernels. In this document, we describe our experience in successfully patching and building Lustre with these patches.

The 2.6.9 PNNL patches consist of a number of patch files and two patch series files:
ldiskfs-2.6-vanilla.series and vanilla-2.6.9.series.

The ldiskfs-2.6-vanilla.series patches are used by the Lustre-1.4.1 source, and the vanilla-2.6.9.series patches are used by the Linux-2.6.9 source.

The general process that we followed was to (1) apply the Linux source patch series, (2) build the Linux source, (3) apply the Lustre source patch series, and (4) build the Lustre source.

1. Download the l-2.6.9.tgz file from the clusterfs wiki at:

https://wiki.clusterfs.com/lustre/LustreStatusonLinux26

That page also has a link to a l-2.6.8.tgz file. The file contains series and patches for the vanilla 2.6.8 kernel. We have not tried applying those patches. This section describes applying the patches in l-2.6.9.tgz to the vanilla Linux 2.6.9 kernel source and the Lustre 1.4.1 source.

2. Unzip and untar the l-2.6.9.tgz file in /usr/src/lustre-1.4.1/lustre/kernel_patches/

   host# mv l-2.6.9.tgz lustre-1.4.1/lustre/kernel_patches/
   host# cd lustre-1.4.1/lustre/kernel_patches/
   host# tar zxf l-2.6.9.tgz

Patching and Building the Vanilla Linux Source

The l-2.6.9.tgz patches were built against the Linux 2.6.9 source and the Lustre 1.4.0 source. We used the Lustre 1.4.1 source and required one additional patch to the Linux 2.6.9 source.

1. Use the CFS version of quilt, available here:


   host# cd /usr/src/linux-2.6.9
   host# quilt setup \n       -l /usr/src/lustre-1.4.1/lustre/kernel_patches/series/vanilla-2.6.9.series \n       -d /usr/src/lustre-1.4.1/lustre/kernel_patches/patches/

   This creates symlinks in /usr/src/ext3rc/linux-2.6.9 for the series file and the patches directory.

   host# quilt push -av

2. Now make two further changes that are outside of the PNNL series and related to the export of kernel symbol filemap_populate. The filemap_populate.patch has been added to the LustreStatusonLinux26 wiki page. You can also download the patch here:

http://www.eecs.harvard.edu/~stein/lug/patches/filemap-populate.patch
3. Apply that patch with

```bash
host# patch -p0 < filemap-populate.patch
```

4. Now build using

```bash
host# make clean bzImage modules
```

5. And install the kernel and modules

```bash
host# make modules_install install
```

6. Check that the kernel has been put in the right place.

**Patching and Building the Lustre Source**

```bash
host# cd /usr/src/lustre-1.4.1
```

1. The configure script has to be pointed at the vanilla series file. In the configure script, change the line from:

```bash
LDISKFS_SERIES="2.6-rhel4.series"
```

to:

```bash
LDISKFS_SERIES="2.6-vanilla.series"
```

To do this, you can download the patch here

[http://www.eecs.harvard.edu/~stein/lug/patches/configure.patch](http://www.eecs.harvard.edu/~stein/lug/patches/configure.patch)

2. Apply the patch with

```bash
host# patch -p0 < configure.patch
```

Makefiles deeper down in the directory hierarchy will prepend `ldiskfs-` to the beginning of `LDISKFS_SERIES` so it is actually the series file named `ldiskfs-2.6-vanilla.series` that is applied by quilt. Quilt is run by the Makefiles and does not need to be run explicitly by the user here.

```bash
host# ./configure --with-linux=/usr/src/linux-2.6.9/
host# make
```

This builds the set of modules that implement Lustre.

3. Install these modules with

```bash
host# make install
```
4. Now reboot to bring up your Lustre-ready kernel.