XC™ Series Lustre® Administration Guide

(CLE 6.0.UP07)

S-2648

Contents

1 About the XC Series Lustre Administration Guide

The XC® Series Lustre® Administration Guide (CLE 6.0 UP07) S-2648 includes information and procedures for configuring direct-attached Lustre (DAL) or external routed configurations (LNet).

This release supports Lustre 2.7.3 for CLE 6.0 UP07. Additional Information about Lustre is available from Intel®: *https://wiki.hpdd.intel.com/display/PUB/Documentation*.

Lustre information in this guide is based, in part, on documentation from Oracle®, Whamcloud®, and Intel. Lustre information contained in Cray publications supersedes information found in Intel publications.

Table 1. Record of Revision

Publication Title	**Date**	**Updates**
XC™ Series Lustre® Administration Guide (CLE 6.0.UP07) S-2648	July 2018	Minor updates to Change Lustre Versions.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP06) S-2648	March 2018	Use Simple Sync to emplace Lustre kernel module parameters.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP05) S-2648	October 2017	Minor updates for CLE 6.0 UP05 and Lustre 2.7.2.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP04) S-2648	June 2017	Updates include resolved bugs, updated fine-grain routing procedures and examples, and documentation to support Lustre 2.7.2 and change to SLES 12 SP2 operating system.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP03) S-2648	February 2017	Updates included resolved bugs, updated fine-grain routing procedures and examples, and documentation to support Lustre 2.7.1.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP02) S-2648	November 2016	Updates included resolved bugs, updated fine-grain routing procedures and examples, and documentation to support Lustre 2.7.1.
XC™ Series Lustre® Administration Guide (CLE 6.0.UP01) S-2648	July 2016	Content in this publication was previously released in April 2015. It was included in the *Manage Lustre for the Cray Linux Environment (CLE) S-0010* and the *XC CLE System Administration Guide S-2393*, which were combined in this single publication.

Related Publications

- *XC™ Series System Administration Guide S-2393*
- *XC™ Series Software Installation and Configuration Guide S-2559*
- *Lustre® Client Software Build and Installation Guide for Cray® Cluster Connect S-2550*

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, key strokes (e.g., `Enter` and `Alt-Ctrl-F`), and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a graphical user interface (GUI) window or element.
`\` (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line). Do not type anything after the backslash or the continuation feature will not work correctly.

Scope and Audience

This publication is written for experienced Cray system software administrators.

Trademarks

2 Introduction to Lustre on CLE 6.0

The Lustre file system is optional on Cray systems with the Cray Linux environment (CLE 6.0). Storage RAID may be configured with other file systems as site requirements dictate. Lustre is a scalable, high-performance, POSIX-compliant file system. It consists of software subsystems, storage, and an associated network (LNet).

Lustre uses the `ldiskfs` file system for back-end storage. The `ldiskfs` file system is an extension to the Linux `ext4` file system with enhancements for Lustre. The packages for the Lustre file system are installed during the CLE 6.0 software installation.

There are two different options for installing a Lustre file system on a Cray system: direct-attached Lustre (DAL) configurations and external Lustre configurations.

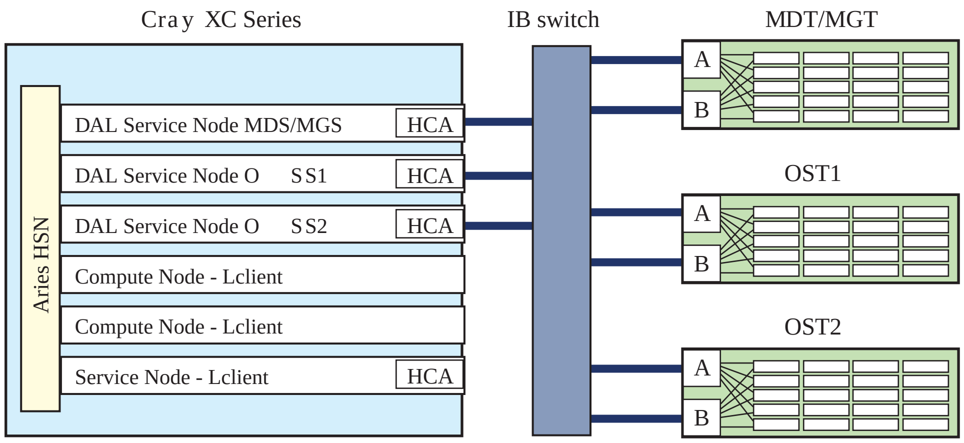
Direct-attached Lustre (DAL)

DAL configurations use specific software images and config sets to configure Cray service nodes to function as Lustre metadata or OSS nodes. DAL configurations require some manual setup during the system installation to configure the Lustre monitoring tool (LMT).

Limitations of DAL Systems Without RSIP

CLE 6.0 UP07 does not support Ethernet cards in DAL MDS/MGS nodes for authentication. RSIP nodes must be used to connect to LDAP servers when users must access directories and files that belong to secondary groups.

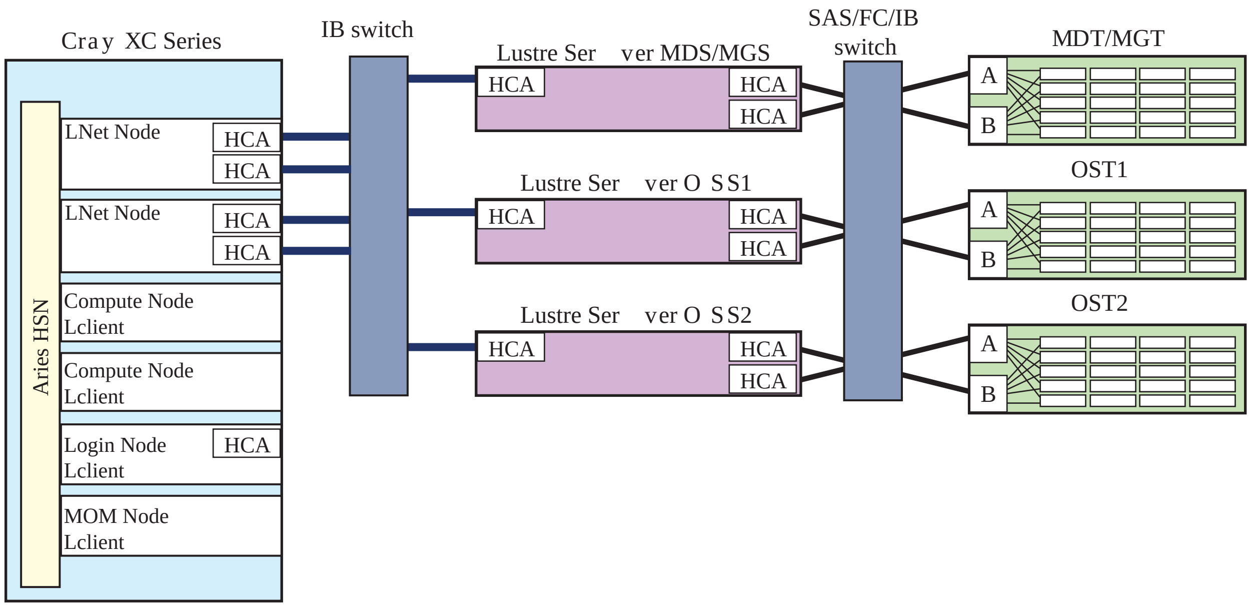
Figure 1. Direct-Attached Lustre Block Diagram



External Lustre

External Lustre file systems implement Cray service nodes as LNet routers. LNet router nodes connect the external Lustre servers on the storage network (IB/FC/SAS) to the Aries HSN.

Figure 2. External Lustre Block Diagram



Configuration Management Framework (CMF)

Lustre setup and management on CLE 6.0 is accomplished using the CMF. The CMF manages configuration data (config set) for the entire system, the tools to manage and distribute that data, and the software to apply the configuration data to the running image at boot time.

The `cfgset` command and the configurator that it invokes are the primary tools that Cray provides for managing configuration data. Some services are not yet supported with the configurator—such as managing Cerebro and creating the Lustre monitoring tool (LMT) MySQL database on the SMW—and must be configured manually.

Image Management and Provisioning System (IMPS) Image Distribution Service (IDS)

IMPS and IDS enable config sets on the management node to be shared to all Lustre client and server nodes in DAL systems or LNet nodes. Config set data is consumed by Ansible plays—another component of the CMF—which act upon that data during the boot phase to enable each node to dynamically self-configure. Lustre services are configured for a config set that dynamically self-configure Lustre nodes.

The services in the config set for managing Lustre are:

- `cray_lustre_server` – Lustre servers
- `cray_lustre_client` – Lustre clients
- `cray_lnet` – LNet routers
- `cray_net` – Network configuration
- `cray_lmt` – Lustre monitoring tool (LMT)

IMPS Recipes

IMPS recipes are included with the system to support the various Lustre node types.

List the IMPS recipes for Lustre:

```
smw# recipe list |grep lustre
compute-large-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari
compute-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari
dal-lustre-2.x_cle_rhine_centos_6.5_x86-64_ari
elogin-large-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari
elogin-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari
...
```

Show more information about a specific Lustre IMPS recipe:

```
smw# recipe show service-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari --fields description
service-lustre-2.x_cle_rhine_sles_12sp2_x86-64_ari:
  description: Generic service node with Lustre 2.x, specifically excludes the login node.
```

2.1 Lustre Software Components and Node Types

The following Lustre software components can be implemented on selected nodes of the Cray system running CLE 6.0 UP07.

Clients	Services or programs that access the file system. On Cray systems, clients are typically associated with login or compute nodes.
LNet Routers	Transfers LNet traffic between Lustre servers and Lustre clients that are on different networks. On Cray systems, LNet routers are typically used to connect external Lustre file systems on InfiniBand (IB) networks to the HSN.
Object Storage Target (OST)	Software interface to back-end storage volumes. There may be one or more OSTs. The client performs parallel I/O operations across multiple OSTs.
Object Storage Server (OSS)	Node that hosts the OSTs. Each OSS node is referenced by node ID (NID) and provides a Fibre Channel or IB connection to a RAID controller. The OST is a logical device; the OSS is the physical node.
Metadata Server (MDS)	Owns and manages information about the files in the Lustre file system. Handles namespace operations such as file creation, but does not contain any file data. Stores information about which file is located on which OSTs, how the blocks of files are striped across the OSTs, the date and time the file was modified, and so on. The MDS is consulted whenever a file is opened or closed. Because file namespace operations are done by the MDS, they do not impact operations that manipulate file data.
Metadata Target (MDT)	Software interface to back-end storage volumes for the MDS. Stores metadata for the Lustre file system.
Management Server (MGS)	Controls the configuration information for all Lustre file systems running at a site. Clients and servers contact the MGS to retrieve or change configuration information.

2.2 Lustre Framework

The system processes (Lustre components) that run on Cray nodes are referred to as Lustre services throughout this topic. The interactions of these services make up the framework for the Lustre file system as follows. The metadata server (MDS) transforms client requests into journaled, batched, metadata updates on persistent storage. The MDS can batch large numbers of requests from a single client. It can also batch large numbers of requests generated by different clients, such as when many clients are updating a single object. After objects have been created by the MDS, the object storage server (OSS) handles remote procedure calls from clients and relays the transactions to the appropriate objects. The OSS read cache uses the Linux page cache to store data on a server until it can be written. Site administrators and analysts should take this into consideration as it may—on direct-attached Lustre (DAL) systems—impact service node memory requirements. For more information, see *OSS Read Cache and Writethrough Cache* on page 32.

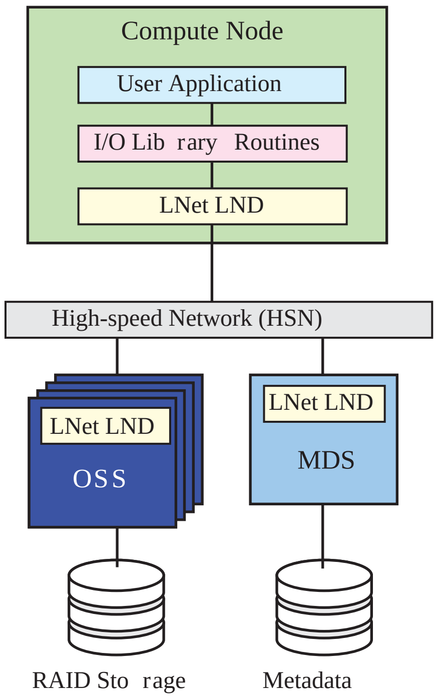
The characteristics of the MDS—such as how files are stored across object storage targets (OSTs)—can be configured as part of the Lustre setup. See the *XC™ Series Software Installation and Configuration Guide S-2559* for Lustre setup procedures.

Each pair of subsystems acts according to protocol.

MDS-Client	The MDS interacts with the client for metadata handling such as the acquisition and updates of inodes, directory information, and security handling.
OST-Client	The OST interacts with the client for file data I/O, including the allocation of blocks, striping, and security enforcement.
MDS-OST	The MDS and OST interact to pre-allocate resources and perform recovery.

The Lustre framework enables files to be structured at file system installation to match data transfer requirements. One MDS plus one or more OSTs make up a single instance of Lustre and are managed together. Client nodes mount the Lustre file system over the network and access files with POSIX file system semantics. Each client mounts Lustre, uses the MDS to access metadata, and performs file I/O directly through the OSTs.

Figure 3. Layout of Lustre File System



3 Configuring Lustre File Systems

3.1 Default Lustre Services Parameter Settings for CLE

Refer to the *XC™ Series Configurator User Guide* S-2560 and the individual worksheet YAML files for information about how to edit the parameters for each node type and update CLE 6.0 UP07 config sets. Always check the worksheet YAML files for new settings and recommended default values for the current CLE release.

Lustre Client Services

`cray_lustre_client.enabled`

Enable or disable the Lustre client service.

`module_params:libcfs_panic_on_lbug`

Set to `true` to enable panic on LBUG.

`module_params:ptlrpc_at_min`

Default **40**. Adaptive timeout minimum in seconds. Cray recommends setting this to 40 on a CLE client.

`module_params:ptlrpc_at_max`

Default **400**. Adaptive timeout maximum in seconds. Cray recommends setting this to 400 on a CLE client.

`module_params:ptlrpc_ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds. This is the minimum amount of time a server will wait to see traffic on a lock before it assumes a client is misbehaving and takes action to revoke the lock by evicting the client. Cray recommends setting this to 260 for CLE clients.

Multiple values of `client_mounts:fs_name` can be configured by giving each entry a unique key identifier.

`client_mounts:lustre_fs_name`

Name for this client mount entry. This name is arbitrary and does not need to correspond to the name of the Lustre file system. It just needs to be unique among the `client_mounts` entries.

`client_mounts:mgs_lnet_nids`

List of MGS LNet NIDs in primary to secondary order. For example, for a DAL file system, the primary MGS LNet NID might be `13@gni`, and the secondary MGS LNet NID might be `14@gni`. For an external file system, the primary MGS LNet NID might be `10.149.0.3@o2ib`, and the secondary MGS LNet NID might be `10.149.0.4@o2ib`.

`client_mounts:mount_options`

List of client mount options which will be passed to `mount.lustre` when the file system is mounted. For example, `rw,flock,lazystatfs` or `rw,flock,user_xattr`.

`client_mounts:mount_at_boot`

Set to `true` for this file system to be mounted at boot time, `false` otherwise.

`client_mounts:client_groups`

A list of node groups which will mount the file system.

Lustre Server Services

`cray_lustre_server.enabled`

Enable or disable the Lustre server service.

`lustre_servers:mgs_group`

Name(s) of the node group(s) containing your Lustre metadata server (MDS) node(s), including failover nodes, if applicable.

`lustre_servers:mds_groups`

Name(s) of the node group(s) containing your Lustre MDS node(s), including failover nodes, if applicable.

`lustre_servers:oss_groups`

Name(s) of the node group(s) containing your Lustre Object Storage Server (OSS) node(s), including failover nodes, if applicable.

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

External Login Node LNet Services

`cray_elogin_lnet:enabled`

Enable or disable eLogin node LNet service.

`local_lnets`

Multiple values of `local_lnets`.

`local_lnets:lnet_name`

Name of the LNet network e.g., `o2ib`. This should be of the form `o2ib`*X* where *X* is some number or omitted.

`local_lnets:ip_wildcard:`

IP address wildcard that matches the IP addresses of all interfaces which should be on this LNet. This must be in the form accepted by the LNet kernel module's `ip2nets` module parameter. For example, `10.149.*.*`.

`local_lnets:interface`

Default **ib0**. LNet interface in the form `ib`*X*, where *X* is the interface number.

`routes:routes_entries`

All routes which are needed to get from the external login nodes to the Lustre file system servers. For example, `o2ib1000 10.149.0.1@o2ib`.

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

LNet Service

`cray_lnet:enabled`

Enable or disable the LNet services.

NOTE: If values are set to an empty string, the corresponding kernel module parameter will be omitted from the `/etc/modprobe.d/*.conf` file generated by the Cray LNet configuration, which will result in LNet using its internal defaults.

`kgnilnd:credits`

Default **2048**. Number of concurrent sends allowed by the `gnilnd`. Cray recommends setting this to 2048.

`kgnilnd:peer_health`

Default **true**. A router-only option which indicates whether or not to enable the peer timeout used for LNet peer health. Cray recommends setting this to true.

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **40**. Number of seconds without aliveness news it takes to declare a peer dead. A value less than or equal to 0 disables the peer aliveness feature. Cray recommends setting this to 40 seconds.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPT. Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends allowed to a single peer. Cray recommends setting this to 126. `peer_credits` must be consistent across all peers on the IB network (all routers and the Lustre servers). If there is a mismatch, the file system will be unmountable. This value can be overridden for each router group in the `flat_routes` or `fgr_routes` settings.

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work-queue sizing. If this option is omitted, the default is calculated based on `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. `concurrent_sends` must be the same for all Lustre clients and servers on the IB network. If there is a mismatch, the file system will be unmountable. This value can be overridden for each router group in the `flat_routes` or `fgr_routes` settings.

`ko2iblnd:peer_buffer_credits`

Default **128**. Number of per-peer router buffer credits. Cray recommends setting this to 128.

`ko2iblnd:peer_credits_hiw`

Default **0**. The `peer_credits_hiw` parameter defines when to eagerly return credits. Cray recommends the default setting of 0 for InfiniBand HCAs or a value of 64 for OPA HFIs.

`ko2iblnd:map_on_demand`

Default **0**. Controls the use of fast memory registration (FMR). Cray recommends setting this value to 0 for InfiniBand HCAs or a value of 32 for Intel® Omni-Path (OPA) host fabric interfaces (HFI).

`ko2iblnd:fmr_pool_size`

Default **512**. Size of FMR pool on each CPT (>=`ntx/4`). Cray recommends setting this value to 512 for InfiniBand HCAs or a value of 2048 for OPA HFIs.

`ko2iblnd:fmr_flush_trigger`

Default **384**. Number of dirty FMRs that triggers a pool flush. Cray recommends setting this value to the default of 384 for InfiniBand HCAs or a value of 512 for OPA HFIs.

`ko2iblnd:fmr_cache`

Default **1**. Used to enable FMR caching. Cray recommends setting this to 1. Cray recommends this default setting for both InfiniBand HCA's or OPA HFIs.

`lnet:router_ping_timeout`

Default **50**. Number of seconds to wait for the reply to a router health query. Cray recommends using the default value of 50 seconds.

`lnet:dead_router_check_interval`

Default **60**. Number of seconds between dead router health checks. Cray recommends using the default value of 60 seconds. A value less than or equal to 0 disables pinging of dead routes.

`lnet:live_router_check_interval`

Default **60**. Number of seconds between live router health checks. Cray recommends leaving this at the default value of 60 seconds. A value less than or equal to 0 disables pinging of live routes.

`lnet:large_router_buffers`

Default **1024**. Number of large (greater than 1 page) messages to buffer in the router. Cray recommends setting this to 1024 on LNet routers.

`lnet:small_router_buffers`

Default **16384**. Number of small (1 page) messages to buffer in the router. Cray recommends setting this to 16384 on LNet routers. Small router buffers are cheap (4KiB), so do not be afraid to increase this value.

`local_lnet:lnet_name`

Specify the local LNet name on this system. A local LNet is one which is used only for communication within this system, and no routes are required to reach an address on this network. For a system which communicates locally over the Aries® or Intel® OPA HSN, this will be of the form `gni`*X* where *X* is a small nonnegative number or omitted. For a system which communicates locally over InfiniBand (e.g. external Lustre servers), this will be of the form `o2ib`*X* where *X* is a small nonnegative number or omitted.

`local_lnet:ip_wildcard`

Enter the IP address wildcard which matches the IP addresses of all interfaces on the local LNet. For example, if the local HSN interfaces are all on the network `10.128.0.0` with netmask `255.252.0.0`, then the IP wildcard matching all local interfaces would be `10.[128-131].*.*`

`local_lnet:ip2nets_file`

Enter the name of the local `ip2nets` file. This is optional. It is useful if configuring multiple local LNets, for example, when configuring multiple LNets on Lustre servers for Fine-grained routing. The file containing the `ip2nets` entries should be placed in *`config_set`* at `smw:/var/opt/cray/imps/config/sets/`*`config_set`*`/files/roles/lnet/`. The file must be generated by an external process or tool, such as `clcvt`. See *Use CLCVT to Configure Fine-Grained Routing Files* on page 65.

`flat_routes:dest_lnet`

Enter the destination LNet name, for example, `o2ib`*`N`* where *`N`* is either a small number or omitted.

`flat_routes:dest_lnet_ip_wildcard`

Enter the IP address wildcard which matches the IP addresses of all router interfaces to be instantiated with the above destination LNet. For example, for a flat route from CLE clients to an external Lustre file system, the destination LNet might be `o2ib`, and the wildcard `10.149.*.*` would match the IB interfaces on the router nodes which are to be on the `o2ib` LNet.

`flat_routes:router_groups`

Enter a list of node groups containing the routers that will route from the source LNet to the destination LNet.

`flat_routes:src_lnet`

Enter the local (source) LNet name at which the routers can be reached. For example, the CLE nodes can reach the routers at `gni`*`N`* where *`N`* is either a small number or omitted.

`flat_routes:ko2iblnd_peer_credits`

Default **126**. The number of concurrent sends allowed to a single peer. Cray recommends setting this to 126. `peer_credits` must be consistent across all peers on the IB network (all routers and the Lustre servers). If there is a mismatch, the file system will be unmountable. This value is specific to the routers specified in this flat route, and it will override the general `ko2iblnd peer_credits` setting specified earlier.

`flat_routes:ko2iblnd_concurrent_sends`

Default **63**. Determines send work-queue sizing. If this option is omitted, the default is calculated based on `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. `concurrent_sends` must be consistent across all peers on the IB network. This means it must be the same on the routers and the Lustre servers. If there is a mismatch, the file system will be unmountable. This value is specific to the routers specified in this flat route, and it will override the general `ko2iblnd` `concurrent_sends` setting specified earlier.

`fgr_routes:dest_name`

Enter the name of the destination. This is not functionally important. A good convention is to use the name of the destination. For example, if the destination is the `husk2` external file system, enter `husk2`.

`fgr_routes:router_groups`

Enter a list of node groups containing the routers that will route from the source LNet to the destination LNet.

`fgr_routes:ip2nets_file`

Enter the name of the `ip2nets` file for this FGR config. The file must be placed in the *`config_set`* at `smw:/var/opt/cray/imps/config/sets/`*`config_set`*`/files/roles/lnet/`. This file must be generated using an external tool, such as `clcvt`. See *Use CLCVT to Configure Fine-Grained Routing Files* on page 65.

`fgr_routes:routes_file`

Enter the name of the routes file for this FGR config. The file must be placed in the *`config_set`* at `smw:/var/opt/cray/imps/config/sets/`*`config_set`*`/files/roles/lnet/`. This file must be generated using an external tool, such as `clcvt`.

`fgr_routes:ko2iblnd_peer_credits`

Default **126**. The number of concurrent sends allowed to a single peer. Cray recommends setting this to 126. `peer_credits` must be consistent across all peers on the IB network (routers and tLustre servers). If there is a mismatch, the file system will be unmountable. This value is specific to the routers specified in this FGR config, and it will override the general `ko2iblnd` `peer_credits` setting specified earlier.

`fgr_routes:ko2iblnd_concurrent_sends`

Default **63**. Determines send work-queue sizing. If this option is omitted, the default is calculated based on `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. `concurrent_sends` must be consistent across all peers on the IB network (routers and Lustre servers). If there is a mismatch, the file system will be unmountable. This value is specific to the routers specified in this FGR config, and it will override the general `ko2iblnd` `concurrent_sends` setting specified earlier.

3.2 Configure DAL File Systems on CLE

The Cray Linux Environment (CLE 6.0 UP07) software includes Lustre control utilities from Cray to support direct-attached Lustre (DAL) file systems. These utilities access site-specific parameters stored in a file system definition (*`fs_name`*`.fs_defs`) file and use that information to interface with the Lustre Mountconf system and management server (MGS). When using the Lustre control configuration utilities, system administrators do not need to access the MGS directly. The `lustre_control` command and `fs_defs` are used to manage DAL file systems. Sonexion systems use the `cscli` commands from the primary management node.

The file system definition file (*`fs_name`*`.fs_defs`) describes the characteristics of the DAL file system—MDS, OST, clients, network, and storage specifications—and configures the `cray_lnet`, `cray_lustre_client`, `cray_lustre_server`, and `cray_net` services in the configuration set. The first task in setting up a Lustre file system on a Cray system is to create a unique file system definition file with values appropriate for the site. Each `fs_defs` file represents one file system. If there is more than one Lustre file system, a `fs_defs` file must be created for each file system.

An optional file system tuning file (*`fs_name`*`.fs_tune`) contains commands for setting Lustre tunable parameters. It is passed as an argument to the `lustre_control set_tune` command. This command can be used to set parameters for multiple file systems. It is available as a convenience feature for administrators who wish to modify their file system settings.

The `lustre_control` utility generates the appropriate commands to manage the DAL file system. By convention, the Lustre control utilities and example `fs_defs` and `fs_tune` files are located in `/opt/cray-xt-lustre-utils/default/bin`.

Service node and compute node clients reference Lustre like a local file system. References to Lustre are handled transparently through the virtual filesystem switch (VFS) in the kernel. Lustre file systems are mounted on compute node clients automatically during startup and can be mounted and unmounted with the `mount_clients` and `umount_clients` actions of `lustre_control`. They can also be manually mounted using the `mount` command.

Use `cfgset` to modify the various Lustre node services in the configuration set as well as Lustre client mount points and settings for DAL.

3.2.1 Ensure a File System Definition is Consistent with Cray System Configurations

It is possible for `/dev/sd*` type device names to change upon reboot of a Cray Linux environment (CLE) system. Host names and node identifiers (NIDs) are dynamically allocated in Cray systems running CLE. They will not change otherwise.

CAUTION: Use persistent device names in the Lustre file system definition. Non-persistent device names (for example, `/dev/sdc`) can change when the system reboots. If non-persistent names are specified in the *`fs_name`*`.fs_defs` file, then Lustre may try to mount the wrong devices and fail to start when the system reboots.

For more information about Lustre control utilities, see the `lustre_control(8)` and `lustre.fs_defs(5)` man pages.

Several options within `lustre_control` enable an administrator to prepare for hardware and software upgrades, link failover, and other dynamics one may encounter that can render the Lustre file system unusable. It is possible that host names, NIDs, and/or device names of either Lustre servers or their storage targets will reflect a configuration different than what is found in the file system definition file.

SCSI device names (`/dev/sd*`) are not guaranteed to be numbered the same from boot to boot. This inconsistency can cause serious problems following a reboot (the Lustre configuration specified in the Lustre file system definition file may differ from actual device names, resulting in a failure to start the file system). Because of this behavior, Cray strongly recommends that persistent device names for Lustre are configured.

Cray supports and tests the `/dev/disk/by-id` persistent device naming conventions. The `by-id` names typically include a portion of the device serial number in the name. For example, `/dev/disk/by-id/scsi-3600a0b800026e1407000192e4b66eb97`.

A separate `udev` rule can be used to create aliases for these devices.

3.2.2 File System Definition Parameters

A Lustre file system definition file (*`fs_name`*`.fs_defs`) for each Lustre file system is required when the `lustre_control` utility is used with CLE. A sample file system definition file is provided in `/etc/opt/cray-xt-lustre-utils/example.fs_defs` on the SMW.

File system definition parameters use the following conventions for node and device naming:

- *`nodename`* is a host or node name using the format *`nidxxxxx`*; for example, `nid00008`
- *`device`* is a device path using the format `/dev/disk/by-id/`*`ID-partN`* where *`ID`* is the volume identifier and *`partN`* is the partition number (if applicable); for example:

  ```
  /dev/disk/by-id/scsi-3600a0b800026e1400000192e4b66eb97-part2
  ```

  CAUTION: Use persistent device names in the Lustre file system definition. Non-persistent device names (for example, `/dev/sdc`) can change when the system reboots. If non-persistent names are specified in the *`fs_name`*`.fs_defs` file, then Lustre may try to mount the wrong devices and fail to start when the system reboots.

  For more information about Lustre control utilities, see the `lustre_control(8)` and `lustre.fs_defs(5)` man pages.
- *`target_type`* can be one of `ost`, `mdt`, or `mgt` (if *`fs_name`*`.fs_defs` parameters are changed, always run the `lustre_control install` command to regenerate the Lustre configuration and apply the changes)

3.2.2.1 Required File System Definitions

The following parameters must be defined in a *`fs_name`*`.fs_defs` file.

`fs_name:` ***`example`***	Specify the unique name for the Lustre file system defined by this *`fs_name`*`.fs_defs` file. This parameter is limited to eight characters. Used internally by Lustre.

`nid_map:` ***`nodes=nid000[27-28,31] nids=[27-28,31]@gni`***	Lustre server hosts to LNet NID mapping. Each line listed here should have a 1:1 mapping between the node name and its associated LNet NID. Use multiple lines for node names that are mapped to multiple LNet NIDs. Multiple lines are additive. Use `pdsh` hostlist expressions. For example, *`prefix`*`[a,k-l,...]` where `a,k,l` are integers with `k < l`.

3.2.2.2 Device Configuration Parameters

Device configuration parameters for the metadata target (MDT), management target (MGT), and object storage targets (OSTs) must be defined. Target device descriptions can span multiple lines and they accept the components listed in the table.

Table 2. `fs_name.fs_defs` Device Configuration Components

`node`	Specifies the primary device host.
`dev`	Specifies the device path.
`fo_node`	Specifies the backup (failover) device host.
`fo_dev`	Specifies the backup (failover) device path. (Only required if different from the primary device path.)
`jdev`	Specifies the external journal device (for OST configuration only).
`index`	Force a particular OST or MDT index. If this component is specified for one OST or MDT, it should be specified for all of them. By default, the index is zero-based and is assigned based on the order in which devices are defined in this file. For example, the first OST has an index value of `0` and the second has an index value of `1`, etc.

```
mdt: node=nodename dev=device fo_node=nodename
```

Specify at least the node and device for the metadata target. For failover configurations, also specify the failover node.

```
mgt: node=nodename dev=device fo_node=nodename
```

Specify at least the node and device for the management target. For failover configurations, also specify the failover node.

```
ost: node=nodename dev=device fo_node=nodename
```

Specify at least the node and device for the OST(s). For failover configurations, also specify the failover node. Including an `index` value makes managing a large number of targets much easier.

3.2.2.3 Mount Path Variables

Device target mount paths must be defined in a configuration. The table describes variables that may be used in mount path definitions. (If using a single shared management target (MGT) device, ensure that the MGT mount path definition will resolve to the same path across all `fs_defs` files; e.g. `mgt_mount_path`: `/tmp/lustre/sharedmgs`.)

Table 3. `fs_name.fs_defs` Mount Path Variables

`__fs_name__`	File system name defined in `fs_name:`.
`__label__`	Component label. For example, `foo-OST0002`.
`__type__`	Component type. For example, `mdt`, `mgt`, or `ost`.
`__index__`	Target index. For example, 1, 2, 36, etc.

```
mgt_mount_path: /tmp/lustre/__fs_name__/__type__
```

Specify the mount path to the MGT.

```
mdt_mount_path: /tmp/lustre/__fs_name__/__type__
```

Specify the mount path to the MDT.

```
ost_mount_path: /tmp/lustre/__fs_name__/__type____index__
```

Specify the mount path to the OSTs.

3.2.2.4 Optional File System Definitions

`auto_fo: yes`	Set this parameter to `yes` to enable automatic failover when failover is configured. Set to `no` to select manual failover. The default setting is `yes`. Automatic failover on direct-attached file systems is only supported for combined MGS/MDT configurations.
`stripe_size: 1048576`	Stripe size in bytes. This is automatically added to the relevant `mkfs.lustre` format parameters. Cray recommends a default value of `1048576` (1MB).
`stripe_count: 1`	Integer count of the default number of OSTs used for a file. This is automatically added to the relevant `mkfs.lustre` format parameters. Valid range is `1` to the number of OSTs. A value of `-1` specifies striping across all OSTs. Cray recommends a stripe count of 2 to 4 OSTs.
`journal_size: 400`	Journal size, in megabytes, on underlying `ldiskfs` file systems. This is automatically added to the relevant `mkfs.lustre` format parameters. The default value is `400`. In addition to the general `journal_size` keyword, users can define specific journal values for `mdt_journal_size` and/or `ost_journal_size`. If a target specific journal size is not defined then the value of `journal_size` is used.
`journal_block_size: 4096`	Journal block size, in bytes, for the journal device. This is automatically added to the relevant `mkfs.lustre` format parameters. The default value is `4096` (4KB).
`back_fs_type: ldiskfs`	Lustre backing file system type. This is automatically added to the relevant `mkfs.lustre` format parameters. The default is `ldiskfs`.

***mgt*|*mdt*|*ost*_format_params**	These "catchall" options, such as `--device-size` or `--param` are passed to `mkfs.lustre`. Multiple lines are additive. For more information on available options, see the `mkfs.lustre` man page.

***mgt*|*mdt*|*ost*_mkfs_mount_options:**	These options are passed to `mkfs.lustre` via `--mkfsoptions="`*`options`*`"`. Mount options specified here replace the default mount options. Multiple lines are additive. The defaults for `ldskfs` are:
	`OST: errors=remout-ro`
	`MGT/MDT: error=remout-ro,iopen_nopriv,user_xattr`

***mgt*|*mdt*|*ost*_mkfs_options:**	Format options for the backing file system. `ext3` options can be set here. They are wrapped with `--mkfsoptions=""` and passed to `mkfs.lustre`. Multiple lines are additive. For more information on options to format backing `ext3` and `ldiskfs` file systems, see the `make2fs(8) man` page.
***mgt*|*mdt*|*ost*_mount_options**	Optional arguments used when starting a target. Default is no options. For more information on mount options for Lustre file systems, see the `mount.lustre man` page. If OSTs are larger than 8TB in size, the `force_over_8tb` option may need to be added to this parameter for Lustre to start properly. For Lustre 2.x, if OSTs are larger than 128TB in size, add the `force_over_128tb` option.
quota: no	Deprecated for Lustre 2.4.0 or greater. To enable quota support, set `quota`: `yes` (the default value is `no`). For more information on quotas in Lustre file systems, see the `lfs(8) man` page.
quota_type: ug	Deprecated for Lustre 2.4.0 or greater. If quotas are enabled, set `quota_type` to u for user quotas, g for group quotas, or ug for both user and group quotas.

3.2.3 Mount and Unmount Lustre Clients Manually

While boot time mounting is handled automatically, Lustre clients occasionally must be mounted or unmounted while the system is running. The `mount_clients` and `umount_clients` actions of the `lustre_control` command supports these options. By adding the `-c` option, Lustre can be mounted or unmounted from the compute node clients. This can prevent them from flooding the system with connection RPCs (remote procedure calls) when Lustre services on an MDS or OSS node are stopped. T

he `mount_clients` and `umount_clients` operations invoke a script on each targeted node that inspects the local `/etc/fstab` file to determine the mount point of the filesystems being mounted/unmounted. For more flexibility, the `-w` option allows a list of nodes to receive the mount or unmount commands to be specified.

For more information, see the `lustre_control(8) man` page.

3.2.4 Unmount Lustre from Compute Node Clients Using lustre_control

Prerequisites

The `busybox mount` command available on the Cray compute nodes is not Lustre-aware, so `mount.lustre` must be used to manually mount compute node clients.

Procedure

1. To unmount Lustre from all compute node clients:

   ```
   boot# lustre_control umount_clients -c -a
   ```

2. To mount Lustre manually on a compute node using `mount.lustre` (substituting values from the particular site).

   ```
   boot# mount.lustre -o rw.flock 12@gni:/lus0 /mnt/lustre/lus0
   ```

3.2.5 Configure an NFS client to Mount the Exported Lustre File System

About this task

Depending on the site client system, the configuration may be different. This procedure contains general information that will help configure the client system to properly mount the exported Lustre file system. Consult the client system documentation for specific configuration instructions.

Procedure

1. As `root`, verify that the `nfs` client service is started at boot.

2. Add a line to the `/etc/fstab` file to mount the exported file system. (For more information on NFS mount options, see the `mount(8)` and `nfs(5)` man pages.)

   ```
   server@network:/filesystem /client/mount/point lustre file_system_options 0 0
   ```

   Recommended file system mount options.

`rsize=1048576,wsize=1048576`	Set the read and write buffer sizes from the server at 1MiB. These options match the NFS read/write transaction to the Lustre filesystem block size, which reduces cache/buffer thrashing on the service node providing the NFS server functionality.
`soft,intr`	Use a soft interruptible mount request.
`async`	Use asynchronous NFS I/O. Once the NFS server has acknowledged receipt of an operation, let the NFS client move along even though the physical write to disk on the NFS server has not been confirmed. For sites that need end-to-end write-commit validation, set this option to `sync` instead.

`proto=tcp`	Force use of TCP transport—this makes the larger `rsize`/`wsize` operations more efficient. This option reduces the potential for UDP retransmit occurrences, which improves end-to-end performance.
`relatime,timeo=600,local_lock=none`	Lock and time stamp handling, transaction timeout at 10 minutes.
`nfsvers=3`	Use NFSv3 specifically. NFSv4 is not supported at this time.

3. Mount the file system manually or reboot the client to verify that it mounts correctly at boot.

3.2.6 Lustre Option for Panic on `LBUG` for CNL and Service Nodes

A Lustre configuration option, `panic_on_lbug`, can be enabled on nodes to control Lustre behavior when processing a fatal file system error. This option is controlled in the `cray_lustre_client` worksheet for the configuration set.

When a Lustre file system hits an unexpected data condition internally, it produces an `LBUG` error to guarantee overall file system data integrity. This renders the file system on the node inoperable. In some cases, an administrator wants the node to remain functional—such as when there are dependencies like a login node that has several other mounted file systems. There are, however, also cases where the desired effect is for the `LBUG` to cause a node to panic.

Compute nodes are good examples, because when this state is triggered by a Lustre or system problem, a compute node is inoperable.

See *Default Lustre Services Parameter Settings for CLE* on page 11

3.2.7 Verify Lustre File System Configuration

The `lustre_control verify config` command compares the `mdt`, `mgt`, and `ost` definitions in the file system definition file (*`fs_name`*`.fs_defs`) to the configured Lustre file system and reports any differences. If failover is configured, the contents of the *`fs_name`*`.fs_defs` file will also be verified to match the contents of the failover tables in the SDB. The failover configuration check will be skipped if `auto_fo: no` in the *`filesystem`*`.fs_defs` file.

Verifying Lustre File System Configuration with `lustre_control verify_config`

Execute the following command to verify all installed Lustre file systems.

```
boot# lustre_control verify_config -a
Performing 'verify_config' from boot at Thu Aug  2 17:29:16 CDT 2012

No problems detected for the following file system(s):
fs_name
```

3.3 Configure Striping on Lustre File Systems

Striping is the process of distributing data from a single file across more than one device. To improve file system performance for a few very large files, files can be striped across several or all OSTs.

The file system default striping pattern is determined by the `stripe_count` and `stripe_size` parameters in the Lustre file system definition file. These parameters are defined as follows.

`stripe_count`	The number of OSTs that each file is striped across. Any number of OSTs can be striped across, from a single OST (the default is one) to all available OSTs.
`stripe_size`	The number of bytes in each stripe. This much data is written to each stripe before starting to write in the next stripe. The default is 1048576.

Striping can also be overridden for individual files. See *Override File System Striping Defaults* on page 24.

CAUTION: Striping can increase the rate that data files can be read or written. However, reliability decreases as the number of stripes increases. Damage to a single OST can cause loss of data in many files.

When configuring striping for Lustre file systems, Cray recommends:

- Striping files across one to four OSTs
- Setting stripe count value greater than 2 (this gives good performance for many types of jobs; for larger file systems, a larger stripe width may improve performance)
- Choosing the default stripe size of 1MB (1048576 bytes)

Stripe size can be increased by powers of two but there is rarely a need to configure a stripe size greater than 2MB. Stripe sizes smaller than 1MB, however, can result in degraded I/O bandwidth. They should be avoided, even for files with writes smaller than the stripe size.

3.3.1 Configuration and Performance Trade-off for Striping

For maximum aggregate performance, it is important to keep all OSTs occupied. The following circumstances should be considered when striping a Lustre file system.

Single OST	When many clients in a parallel application are each creating their own files, and where the number of clients is significantly larger than the number of OSTs, the best aggregate performance is achieved when each object is put on only a single OST.
Multiple OSTs	At the other extreme, for applications where multiple processes are all writing to one large (sparse) file, it is better to stripe that single file over all of the available OSTs. Similarly, if a few processes write large files in large chunks, it is a good idea to stripe over enough OSTs to keep the OSTs busy on both the write and the read path.

3.3.2 Override File System Striping Defaults

Each Lustre file system is built with a default stripe pattern that is specified in `*fs_name*.fs_defs`. However, users may select alternative stripe patterns for specific files or directories with the `lfs setstripe` command. For more information, see the `lfs(1) man` page.

File Striping

The `lfs setstripe` command has the following syntax, `lfs setstripe -s` *`stripe_size`* `-c` *`stripe_count`* `-i` *`stripe_start`* *`filename`*.

This example creates the file, `npf`, with a 2MB (`2097152` bytes) stripe that starts on OST0 (`0`) and stripes over two object storage targets (OSTs) (2).

```
$ lfs setstripe -s 2097152 -c 2 -i 0 npf
```

Here the `-s` specifies the stripe size, the `-c` specifies the stripe count, and the `-i` specifies the index of the starting OST.

The first two megabytes, bytes `0` through `2097151`, of `npf` are placed on `OST0`, and then the third and fourth megabytes, `2097152-4194303`, are placed on `OST1`. The fifth and sixth megabytes are again placed on `OST0` and so on.

The following special values are defined for the `lfs setstripe` options.

`stripe_size=0`	Uses the file system default for stripe size.
`stripe_start=-1`	Uses the default behavior for setting OST values.
`stripe_count=0`	Uses the file system default for stripe count.
`stripe_count=-1`	Uses all OSTs.

4 Lustre System Administration

4.1 Lustre Commands for System Administrators

Cray provides administrative commands that configure and maintain Lustre file systems as shown in *Lustre Administrative Commands Provided with CLE*. The `man` pages are accessed by using the `man` command on a Cray system.

For more information about standard Lustre system administration, see the following `man` pages: `Lustre(7)`, `mount(8)`, `mkfs.lustre(8)`, `tunefs.lustre(8)`, `mount.lustre(8)`, `lctl(8)`, and `lfs(1)`.

Table 4. Lustre Administrative Commands Provided with CLE

Command	Function
`lustre_control`	Manages Lustre file system using standard Lustre commands and a customized Lustre file system definition file.
`update_fs_defs`	Updates an older `fs_defs` file to the format required in CLE 4.1 and beyond.
`xtlusfoadmin`	Displays the contents of the `lustre_failover`, `lustre_service`, and `filesystem` database tables in service database (SDB). It is also used by the system administrator to update database fields to enable or disable automatic Lustre service failover handled by the `xt-lustre-proxy` daemon. (Direct-attached Lustre file systems only.)
`xtlusfoevntsndr`	Sends Lustre failover imperative recovery events to start the Lustre client connection switch utility on login and compute nodes. (Direct-attached Lustre file systems only.)

4.2 Identify MDS and OSSs

Identifying MDS and OSSs

Use the `lustre_control status` command to identify the OSSs and MDS for direct-attached file systems. This command must be `root` to be used.

```
boot# lustre_control status -a
```

To identify the OSSs and MDS on all (including external) Lustre file systems, as `root`, use the `lfs check servers` command.

```
login# lfs check servers
```

If there is more than one Lustre file system, the `lfs check servers` command does not necessarily sort the OSSs and MDSs by file system.

Checking the Status of Individual Nodes

The status of targets on an individual node can be checked with the `lustre_control status` command.

```
boot# lustre_control status -a -w nodename
```

4.3 Start Lustre

About this task

Lustre file systems are started at boot time by CLE boot automation files. Lustre file systems can be manually started using the `lustre_control` command.

Procedure

1. Start the file systems using `lustre_control`.
   a. Load module.

      ```
      boot# module load lustre-utils
      ```

   b. Start filesystem.

      ```
      boot# lustre_control start -a
      ```

2. Mount the service node clients.

   ```
   boot# lustre_control mount_clients -a
   ```

3. Mount the compute node clients. (If the appropriate `/etc/fstab` entries for the Lustre file system are present in the CNL boot image, then the compute nodes—at boot—will mount Lustre automatically.)

   To manually mount Lustre on compute nodes that are already booted, use the following command.

   ```
   boot# lustre_control mount_clients -a -c
   ```

4.4 Stop Lustre

About this task

Lustre file systems are stopped during shutdown by CLE system boot automation files. Lustre file systems can be manually stopped using the `lustre_control` command.

If all compute nodes and service node clients must be unmounted and all services stopped, alternatively the `lustre_control shutdown -a` command can be executed. The following procedure breaks this process up into three steps.

Procedure

1. Unmount Lustre from the compute node clients.

   ```
   boot# lustre_control umount_clients -a -c
   ```

2. Unmount Lustre from the service node clients.

   ```
   boot# lustre_control umount_clients -a
   ```

3. Stop Lustre services.

   ```
   boot# lustre_control stop -a
   ```

   For more information, see the `lustre_control(8) man` page.

4.5 Add OSSs and OSTs

About this task

New object storage servers (OSSs) and object storage targets (OSTs) can be added (or new targets can be added to existing servers) by performing the following procedure.

Procedure

1. Unmount Lustre from the compute node clients.

   ```
   boot# lustre_control umount_clients -f fs_name -c
   ```

2. Unmount Lustre from the service node clients.

   ```
   boot# lustre_control umount_clients -f fs_name
   ```

3. Stop Lustre services.

   ```
   boot# lustre_control stop -f fs_name
   ```

4. Update the Lustre file system definition file, `/etc/opt/cray/lustre-utils/`*`fs_name`*`.fs_defs` on the SMW node.

   Add the Lustre server host to LNet `nid` mapping (unless there is already a `nid_map` listing for this OSS).

   ```
   nid_map: nodes=nid00026 nids=26@gni
   ```

5. Add the OSTs.

   This example shows multiple OSTs on the same OSS being added.

   ```
   ost: node=nid00026
        dev=/dev/disk/by-id/IDa
   ```

```
     index=n

ost: node=nid00026
     dev=/dev/disk/by-id/IDb
     index=n+1
```

The new index numbers (`index=n`) in this sequence must follow the pre-existing index sequence numbers.

6. Remove the existing file system configuration and rerun the `lustre_control install` command with the updated `fs_defs` file.

   a. Remove current configuration file.

   ```
   smw# lustre_control remove -c p0 -f fs_name
   ```

   b. Install new configuration file.

   ```
   smw# lustre_control install –c p0 /home/crayadm/fs_name.fs_defs
   ```

7. Format the new targets on the OSS.

   a. Format the first new OST.

   ```
   nid00026# mkfs.lustre --fsname=filesystem --index=n --ost --mgsnode=12@gni /dev/disk/by-id/IDa
   ```

   b. Format any additional new OSTs.

   ```
   nid00026# mkfs.lustre --fsname=filesystem --index=n+1 --ost --mgsnode=12@gni /dev/disk/by-id/IDb
   ```

   If desired, other file system options can be added to this command, such as `--mkfsoptions`, `--index`, or `--failnode`. For more information, see the `mkfs.lustre(8) man` page.

8. Regenerate the Lustre configuration logs with the `lustre_control` script.

```
boot# lustre_control write_conf -f fs_name
```

Lustre services will be started and then stopped as part of this command to allow for proper registration of the new OSTs with the MGS.

9. Start the Lustre file system.

```
boot# lustre_control start -p -f fs_name
```

10. Mount Lustre on the service node clients.

```
boot# lustre_control mount_clients -f fs_name
```

11. Check that the OST is among the active targets in the file system on the login node from the Lustre mount point.

   a. Connect to login node.

   ```
   boot# ssh login
   ```

   b. Change working directory to mounted filesystem.

   ```
   login# cd /fs_name
   ```

   c. List active targets.

   ```
   login:/fs_name# lfs check servers
   fs_name-MDT0000-mdc-ffff8100f14d7c00 active.
   fs_name-OST0001-osc-ffff8100f14d7c00 active.
   fs_name-OST0002-osc-ffff8100f14d7c00 active.
   ```

```
fs_name-OST0003-osc-ffff8100f14d7c00 active.
fs_name-OST0004-osc-ffff8100f14d7c00 active.
```

12. Write a file to the Lustre file system from the login node to test if a new target is receiving I/O.

a. Create a new directory.

```
login:/fs_name# mkdir mydirectory
```

b. Move into directory.

```
login:/fs_name# cd mydirectory
```

c. Create a file that is striped across all OSTs.

```
login:/fs_name/mydirectory# lfs setstripe testfile -s 0 -c -1 -i -1
```

d. Write data to the new file.

```
login:/fs_name/mydirectory# dd if=/dev/zero of=testfile bs=10485760 count=1
1+0 records in
1+0 records out
10485760 bytes (10 MB) copied, 0.026317 seconds, 398 MB/s
```

e. Check that the file object is stored on the new target using `lfs`.

```
login:/fs_name/mydirectory# lfs getstripe testfile
OBDS:
0: ost0_UUID ACTIVE
1: ost1_UUID ACTIVE
2: ost2_UUID ACTIVE
3: ost3_UUID ACTIVE
4: ost4_UUID ACTIVE
testfile
        obdidx           objid          objid            group
             4         1237766       0x12e306                0              3
564292        0x89c44                0
             1          437047        0x6ab37                0
             0          720254        0xafd7e                0
             2          487517        0x7705d                0
```

13. Mount Lustre on the compute node clients.

```
boot# lustre_control mount_clients -f fs_name -c
```

4.6 Recover from a Failed OST

Use these procedures when an OST has failed and is not recoverable by `e2fsck`. In this case, the individual OST can be reformatted and brought back into the file system. Before reformatting, the OST must be deactivated and any striped files residing on it must be identified and removed.

4.6.1 Deactivate a Failed OST and Remove Striped Files

Procedure

1. Log in to the MDS and deactivate the failed OST in order to prevent further I/O operations on the failed device.

```
nid00012# lctl --device ostidx deactivate
```

The *ostdix* is displayed in the left column of the output generated by the `lctl dl` command.

2. Regenerate the list of Lustre devices and verify that the state for the deactivated OST is `IN` (inactive) and not `UP`.

```
nid00012# lctl dl
```

3. Identify the *ostname* for the OST by running the following command.

```
login> lfs df
UUID                 1K-blocks      Used Available  Use% Mounted on
lustre-MDT0000_UUID 358373232   1809780 336083452    0% /lus/nid00012[MDT:0]
lustre-OST0000_UUID 2306956012 1471416476 718352736   63% /lus/nid00018[OST:0]
lustre-OST0001_UUID 2306956012 1315772068 873988520   57% /lus/nid00018[OST:1]
```

The *ostname* will be similar to *fsname-OSTxxxx_UUID*.

4. Log in to a Lustre client, such as a login node, and search for files on the failed OST.

```
login> lfs find /mnt/filesystem --print --obd ostname
```

5. Remove (`unlink` or `rm`) any striped files on the OST before reformatting.

4.6.2 Reformat a Single OST

About this task

Refer to this procedure if there is a failed OST on a Lustre file system. For example, if the OST is damaged and cannot be repaired by `e2fsck`. This procedure can be used for an OST that is available and accessible. But—prior to completing the remaining steps—*Deactivate a Failed OST and Remove Striped Files* on page 30 should be completed to generate a list of affected files and unlink or remove them.

Procedure

1. Unmount Lustre from the compute node clients.

```
boot# lustre_control umount_clients -f fs_name -c
```

2. Unmount Lustre from the service node clients.

```
boot# lustre_control umount_clients -f fs_name
```

3. Stop Lustre services.

```
boot# lustre_control stop -f fs_name
```

4. Reformat the OST from the OSS node that serves it.

Use values from the file system definition file for the following options: *nid* is the `node` value for the `mgt`, *ostidx* is the `index` value for this OST, and *ostdevice* is the `dev` device name for this OST. If there are any additional `ost_mkfs_options` in the *fs_name*`.fs_defs` file, append them to the `-J size=400` value of `--mkfsoptions` in the following command. Make sure to append and "catchall" (such as `--param`) options to this command, as well.

```
nid00018# mkfs.lustre --reformat --ost --fsname=fs_name --mgsnode=nid@gni \
--index=ostidx --param sys.timeout=300 --mkfsoptions="-J size=400" ostdevice
```

5. Regenerate the Lustre configuration logs on the servers by invoking the following command from the boot node.

   ```
   boot# lustre_control write_conf -f fs_name
   ```

6. On the MDS node, mount the MDT device as `ldiskfs`, and rename the `lov_objid` file.

   a. Mount the MDT as `ldiskfs`.

   ```
   nid00012# mount -t ldiskfs mdtdevice /mnt
   ```

   b. Change the name of the `lov_objid` file.

   ```
   nid00012# mv /mnt/lov_objid /mnt/lov_objid.old
   ```

   c. Unmount the MDT.

   ```
   nid00012# umount /mnt
   ```

7. Start Lustre on the servers.

   ```
   boot# lustre_control start -p -f fs_name
   ```

8. Activate the newly reformatted OST on the MDS device.

   a. Generate a list of all the Lustre devices with the `lctl dl` command. (Note the device index for the OST that was reformatted in the far left column.)

   ```
   nid00012# lctl dl
   ```

   b. Activate the OST using the index from the previous step as *`ostidx`*.

   ```
   nid00012# lctl --device ostidx activate
   ```

   c. Regenerate the list of Lustre devices and verify that the state for the activated OST is `UP` and not `IN`.

   ```
   nid00012# lctl dl
   ```

9. Mount Lustre on the clients.

4.7 OSS Read Cache and Writethrough Cache

This section describes several commands that can be used to tune, enable, and disable object storage servers (OSS) server cache settings. If these settings are modified on a system, the modification commands must be run on each of the OSSs.

Lustre uses the Linux page cache to provide read-only caching of data on OSSs. This strategy reduces disk access time caused by repeated reads from an object storage target (OST). Increased cache utilization, however, can evict more important file system metadata that subsequently needs to be read back from the disk. Very large files are not typically read multiple times, so their pressure on the cache is unwarranted.

Administrators can control the maximum size of files that are cached on the OSSs with the `readcache_max_filesize` parameter. To adjust this parameter from the default value for all OSTs on `nid00018`, invoke the following command.

```
nid00018# lctl set_param obdfilter.*.readcache_max_filesize=value
```

The asterisk in the above command is a wild card that represents all OSTs on that server. If administrators wish to affect a single target, individual OST names can be used instead.

This command sets `readcache_max_filesize` to *`value`*, so that files larger than *`value`* will not be cached on the OSS servers. Administrators can specify *`value`* in bytes or shorthand such as `32MiB`. Setting *`value`* to `-1` will cache all files regardless of size (this is the default setting).

OSS read cache is enabled by default. It can be disabled, however, by setting `/proc` parameters. For example, invoke the following on the OSS.

```
nid00018# lctl set_param obdfilter.*.read_cache_enable=0
```

Writethrough cache can also be disabled. This prevents file writes from ending up in the read cache. To disable writethrough cache, invoke the following on the OSS.

```
nid00018# lctl set_param obdfilter.*.writethrough_cache_enable=0
```

Conversely, setting `read_cache_enable` and `writethrough_cache_enable` equal to `1` will enable them.

4.8 Lustre 2.x Performance and `max_rpcs_in_flight`

Performance comparisons of 1.8.*x* and 2.*x* Lustre clients have brought to light large differences in direct I/O (DIO) rates. Lustre clients use a semaphore initialized to `max_rpcs_in_flight` to throttle the amount of I/O RPCs to each storage target. However, Lustre 1.8.*x* clients do not abide by the tunable for DIO request and there is no limit to the number of DIO requests in flight. This results in increased single-client DIO performance compared to 2.*x* clients. Aggregate performance is comparable given enough clients.

To increase single client DIO performance of Lustre 2.*x*, modify the `max_rpcs_in_flight osc` tunable. The tunable can be configured permanently for the file system via the `lctl conf_param` command on the MGS. Alternatively, for direct-attached and esFS Lustre file systems, the `lustre_control set_tune` can be used to easily change it in a non-permanent fashion. The default value is `8`. The maximum value is `256`.

```
# lctl conf_param osc.*.max_rpcs_in_flight=value
```

Be aware that tune-up of the value increases client memory consumption. This might be a concern for file systems with a large number of object storage targets (OSTs). There will be an additional 9k/RPC/target increase for each credit. Note that increasing the tunable to increase single client DIO performance will also allow more RPCs in flight to each OST for all other RPC request types. That means that buffered I/O performance should increase as well.

Also, be aware that more operations in flight will put an increased load onto object storage servers (OSSs). Increased load can result in longer service times and recovery periods during failover. Overloaded servers are identified by slow response and/or informational messages from clients. However, even large increases in `max_rpcs_in_flight` should not cause overloading. Cray has completed large-scale testing of increased `max_rpcs_in_flight` values using file systems with tens of thousands of simulated clients (that were simulated with multiple mounts per client) under extreme file system load. No scale issues have been found.

Set `max_rpcs_in_flight` and `max_dirty_mb`

The `osc.`*`osc_instance`*`.max_dirty_mb` setting controls how many MBs of dirty data can be written and queued up in the OSC. POSIX file writes that are cached contribute to this count. When the limit is reached, additional writes stall until previously cached writes are written to the server. This may be changed by writing a single ASCII integer to the file. Only values between 0 and 2048 or 1/4 of RAM are allowable. If 0 is specified, no writes are cached. Performance suffers noticeably unless large writes (1MB or more) are used. To maximize performance, the value for `max_dirty_mb` is recommended to be 4 * `max_pages_per_rpc` * `max_rpcs_in_flight`.

The `max_dirty_mb`, `max_rpcs_in_flight` parameters are client parameters. The `lctl set_param` command can be used to adjust these directly on clients (they would not be permanent settings but can be included as boot-time commands run after Lustre file systems are mounted). It is possible that the parameters should not be changed globally for a file system (global, persistent change is done on the MGS node using `lctl conf_param` or `lctl set_param -P`). Non-default values may be appropriate for some clients but not necessarily all (e.g. compute vs. login node).

Use `lctl set_param` to set temporary parameters on the node where it is run. These parameters map to items in `/proc/`*`fs, sys/ lnet, lustre`*. The `lctl set_param` command uses this syntax:

```
lctl set_param [-n]
obdtype.
obdname.
proc_file_name=
value
```

For example:

```
# lctl set_param osc.*.max_dirty_mb=1024
osc.myth-OST0000-osc.max_dirty_mb=32
osc.myth-OST0001-osc.max_dirty_mb=32
osc.myth-OST0002-osc.max_dirty_mb=32
osc.myth-OST0003-osc.max_dirty_mb=32
osc.myth-OST0004-osc.max_dirty_mb=32
```

4.9 Check Lustre Disk Usage

From a login node, type the following command.

```
login$ df -t lustre
Filesystem          1K-blocks      Used Available Use% Mounted on
12@gni:/lus0        2302839200   1928332 2183932948   1% /lustre
```

The `lfs df` command can be used to view free space on a per-OST basis.

```
login$ lfs df
UUID                1K-blocks      Used Available  Use% Mounted on
mds1_UUID           958719056  57816156 900902900    6% /scratch[MDT:0]
ost0_UUID           1061446736 570794228 490652508   53% /scratch[OST:0]
ost1_UUID           1061446736 571656852 489789884   53% /scratch[OST:1]
ost2_UUID           1061446736 604100184 457346552   56% /scratch[OST:2]
ost3_UUID           1061446736 604444248 457002488   56% /scratch[OST:3]
ost4_UUID           1061446736 588747532 472699204   55% /scratch[OST:4]
ost5_UUID           1061446736 597193036 464253700   56% /scratch[OST:5]
ost6_UUID           1061446736 575854840 485591896   54% /scratch[OST:6]
ost7_UUID           1061446736 576749764 484696972   54% /scratch[OST:7]
ost8_UUID           1061446736 582282984 479163752   54% /scratch[OST:8]
ost9_UUID           1061446736 577588324 483858412   54% /scratch[OST:9]
ost10_UUID          1061446736 571413316 490033420   53% /scratch[OST:10]
ost11_UUID          1061446736 574388200 487058536   54% /scratch[OST:11]
ost12_UUID          1061446736 593370792 468075944   55% /scratch[OST:12]
ost13_UUID          1061446736 585151932 476294804   55% /scratch[OST:13]
ost14_UUID          1061446736 564455796 496990940   53% /scratch[OST:14]

filesystem summary:  15921701040 8738192028 7183509012   54% /scratch
```

4.10 Lustre User and Group Quotas

Disk quotas provide system administrators with the ability to set the amount of disk space available to users and groups. Cray Lustre utilities allow administrators to easily enable user and group quotas on a system.

4.11 Check the Lustre File System

Lustre makes use of a journaling file system for its underlying storage (OSTs and MDT). This journaling feature allows for automatic recovery of file system data following a system crash. While not normally required for recovery, the `e2fsck` command can be run on individual OSTs and the MDT to check integrity. Before Lustre is restarted, administrators should always run an `e2fsck` on a target associated with any file system failures. In the case of a catastrophic failure, a special Lustre file system check utility, `lfsck`, is also provided. The `lfsck` command can be used to check coherency of the full Lustre file system.

The `lustre_control` command provides an `fs_check` action. It performs an `e2fsck` on specified devices by label, all devices on a specified host, or even all target types in a specified file system.

4.11.1 Perform an `e2fsck` on All OSTs in a File System with `lustre_control`

Procedure

Check all object storage targets (OSTs) in *`fs_name`* with `lustre_control`.

```
boot# lustre_control fs_check -f fs_name -t OST
```

For more information, see the `lustre_control(8)`, `e2fsck(8)` and `lfsck(8)` `man` pages.

4.12 Change Lustre Versions

Prerequisites

CLE software must be installed.

About this task

This procedure describes how to build image root and boot images with a version of Lustre other than the default version in the standard image recipes for XC systems running CLE 6.0.UP06 or a later release.

Procedure

1. Identify recipes with non-default Lustre.

With CLE 6.x, the default Lustre version is Lustre 2.7.2, and that version is included in the package collections for all recipes. However, a second set of recipes exist that use package collections with the non-default version of Lustre RPMs (2.7.3). These additional recipes can be located by this command.

```
smw# recipe list | grep lustre
compute-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
compute-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
elogin-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
elogin-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
initrd-compute-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
initrd-login-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
initrd-login-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
login-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
login-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
service-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
```

BUILD CLE IMAGES WITH NON-DEFAULT LUSTRE VERSION

2. Customize the `cray_image_groups` configuration file, as needed, by editing `/var/opt/cray/imps/config/sets/global/config/cray_image_groups.yaml` and by adding stanzas for standard and/or netroot images to be created.

Choose only the set of tmpfs recipes or the set of netroot recipes which matches the existing recipes in use on this system. There is a single recipe for service since it is always tmpfs.

```
smw# vi /var/opt/cray/imps/config/sets/global/config/cray_image_groups.yaml
```

For tmpfs.

```
cray_image_groups:
  lustre25:
     - recipe: "compute-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "compute-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-x86_64-
created{date}"
       arch: "x86_64"
       export_format: "cpio"
       nims_group: "compute"
     - recipe: "compute-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "compute-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-12sp2-
aarch64-created{date}"
       arch: "aarch64"
       export_format: "cpio"
       nims_group: "compute_aarch64"
     - recipe: "login-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "login-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-x86_64-
created{date}"
       arch: "x86_64"
       export_format: "cpio"
       nims_group: "login"
     - recipe: "login-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "login-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-aarch64-
created{date}"
       arch: "aarch64"
       export_format: "cpio"
       nims_group: "login_aarch64"
     - recipe: "service-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "service-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
created{date}"
       export_format: "cpio"
       nims_group: "service"
```

For netroot.

```
cray_image_groups
  lustre25:
     - recipe: "initrd-compute-large-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "initrd-compute-large-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
x86_64-created{date}"
       arch: "x86_64"
       export_format: "cpio"
       nims_group: "compute"
     - recipe: "initrd-compute-large-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "initrd-compute-large-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
aarch64-created{date}"
       arch: "aarch64"
       export_format: "cpio"
       nims_group: "compute_aarch64"
     - recipe: "initrd-login-large-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "initrd-login-large-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
x86_64-created{date}"
       arch: "x86_64"
       export_format: "cpio"
       nims_group: "login"
     - recipe: "initrd-login-large-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "initrd-login-large-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
aarch64-created{date}"
       arch: "aarch64"
       export_format: "cpio"
       nims_group: "login_aarch64"
     - recipe: "service-lustre-2.5_cle_{cle_release_lowercase}_sles_12sp3_ari"
       dest: "service-lustre-2.5{note}_cle_{cle_release_lowercase}-build{cle_build}{patch}_sles_12sp2-
created{date}"
       export_format: "cpio"
       nims_group: "service"
```

3. Run `imgbuilder` to make Lustre images.

```
smw# imgbuilder -g lustre25
```

4. (For netroot images only) Push netroot image roots to the boot node.

   Note that the following example uses `image sqpush`. If these image roots will be modified and pushed more than once to test them, consider using `image push` instead. For more information, see *About Image Pushes: push versus sqpush*.

```
smw# image sqpush -d boot login-large-lustre-2.5_cle_version_and_build.cpio
smw# image sqpush -d boot compute-large-lustre-2.5_cle_version_and_build.cpio
```

ASSIGN NEW LUSTRE IMAGES TO TEST NODES

5. Determine names of boot images built above.

```
smw# image list | grep lustre
```

6. Use `cnode` to assign the boot images to nodes for testing.

   a. Assign service image.

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/service-lustre-2.5_cle_version_and_build.cpio c9-9c0s0n1
```

   b. Assign DAL image.

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/dal-lustre-2.5_cle_version_and_build.cpio c8-8c0s0n2
```

   c. Assign login image.

   - For tmpfs:

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/login-lustre-2.5_cle_version_and_build.cpio c8-8c0s0n1
```

   - For netroot:

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/initrd-login-large-lustre-2.5_cle_version_and_build.cpio \
-k netroot=login-large-lustre-2.5_cle_version_and_build c8-8c0s0n1
```

   d. Assign compute image.

   - For tmpfs:

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/compute-lustre-2.5_cle_version_and_build.cpio c7-7c0s0n1
```

   - For netroot:

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/initrd-compute-large-lustre-2.5_cle_version_and_build.cpio \
-k netroot=compute-large-lustre-2.5_cle_version_and_build c7-7c0s0n1
```

WARMBOOT TEST NODES WITH NEW LUSTRE IMAGES

7. Warmboot test nodes with new Lustre images.

a. Log in as `crayadm`.

```
smw# su - crayadm
```

b. Shut down node.

```
crayadm@smw> xtcli shutdown c8-8c0s0n1
```

c. Wait 60 seconds.

```
crayadm@smw> sleep 60
```

d. Reboot node with new image.

```
crayadm@smw> xtbootsys --reboot -r "warmboot to test Lustre-2.5" c8-8c0s0n1
```

e. Repeat for each type of node to be tested.

REBOOT ENTIRE CLE SYSTEM WITH NEW LUSTRE IMAGES

8. Assign Lustre 2.5 images to all nodes.

a. Assign to all service nodes.

```
smw# cnode update --filter group=service -i \
/var/opt/cray/imps/boot_images/service-lustre-2.5_cle_version_and_build.cpio
```

b. Assign to all login nodes.

- For tmpfs:

```
smw# cnode --filter group=login update -i \
/var/opt/cray/imps/boot_images/login-lustre-2.5_cle_version_and_build.cpio
```

- For netroot:

```
smw# cnode update \
-i /var/opt/cray/imps/boot_images/initrd-login-large-lustre-2.5_cle_version_and_build.cpio \
-k netroot=login-large-lustre-2.5_cle_version_and_build --filter group=login
```

c. Assign to all compute nodes.

- For tmpfs:

```
smw# cnode update --filter group=compute -i \
/var/opt/cray/imps/boot_images/compute-lustre-2.5_cle_version_and_build.cpio
```

- For netroot:

```
smw# cnode update -i \
/var/opt/cray/imps/boot_images/initrd-compute-large-lustre-2.5_cle_version_and_build.cpio \
-k netroot=compute-large-lustre-2.5_cle_version_and_build.cpio --filter group=compute
```

For netroot only, if the compute-large and login-large image roots were not pushed to the boot node when testing the new images, push them to the boot node before rebooting the entire system. The boot node must be booted for the `image sqpush` command to succeed.

9. Shut down CLE.

```
crayadm@smw> xtbootsys -s last -a auto.hostname.stop
```

10. Boot CLE.

```
crayadm@smw> xtbootsys -a auto.hostname.start
```

11. Build recipes and deploy boot image for eLogin nodes using non-default Lustre.

    To build eLogin images, use the procedure in *XC Series SMW-managed eLogin Installation Guide* (S-3020), but remember to choose the appropriate image recipe which includes the non-default version of Lustre. This example shows the eLogin recipes that match the netroot and tmpfs login recipes configured for the XC system. Select the image recipe that most closely resembles what the XC login node uses.

    ```
    smw# recipe list | grep lustre | grep elogin
    elogin-large-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
    elogin-lustre-2.5_cle_6.0.up07_sles_12sp3_ari
    ```

4.13 Use Simple Sync to Emplace Lustre Kernel Module Parameters

Prerequisites

Simple Sync service (`cray_simple_sync`) must be enabled. The Simple Sync service is executed on all CLE nodes at boot time and whenever the administrator executes `cray-ansible start` on the node. Refer to the *XC™ Series System Administration Guide S-2393* for more information about the Simple Sync service.

About this task

It may be necessary to set Lustre kernel module parameters that are not exposed via the cfgset infrastructure. This example sets a `ko2iblnd` parameter that controls the number of scatter/gather entries in each InfiniBand work request. The parameter `wrq_sge` is set for LNet routers belonging to the `lnet_flat_routes` nodegroup.

Procedure

1. Create the directory `etc/modprobe.d` under the appropriate nodegroup location in the desired config set directory:.

   ```
   smw# mkdir –p \
   /var/opt/cray/imps/config/sets/configset/files/simple_sync/\
   nodegroups/lnet_flat_routes/files/etc/modprobe.d
   ```

   This creates an `/etc/modprobe.d` directory on the nodes belonging to the`lnet_flat_routes` node group.

2. Create a file containing the parameter setting in the `/etc/modprobe.d` directory. This file will be available on all nodes belonging to the `lnet_flat_routes` nodegroup.

   ```
   smw# echo “options ko2iblnd wrq_sge=2” > \
   /var/opt/cray/imps/config/sets/configset/files/simple_sync/nodegroups/lnet_flat_routes\
   /files/etc/modprobe.d/wrq_sge.conf
   ```

3. Login to the node and run ansible.

   ```
   node# /etc/init.d/cray-ansible start
   ```

4.14 Dump Lustre Log Files

When Lustre encounters a problem, internal Lustre debug logs are generated on the MDS and OSS nodes in `/tmp` directory log files. These files can be dumped on both server and client nodes. Log files are named by a timestamp and PID. For example, `/tmp/lustre-log-nid00135.1122323203.645`.

The `xtdumpsys` command does not collect this data automatically. Since the files reside in `/tmp`, they disappear on reboot. Create a script to retrieve the dump files from all MDS and OST nodes and store them in the dump directory. The files can be collected by invoking the script at the end of the `xtdumpsys_mid` function in an `xtdumpsys` plugin file.

Lustre debug logging can also be enabled on compute node clients. To do this, execute the following command on the compute nodes:

```
# echo 1 > /proc/sys/lustre/dump_on_eviction
```

Collect these logs before shutdown as they will also disappear on reboot.

4.15 File System Error Messages

Lustre errors are normally reported in both the `syslog` messages file and in the Cray system console log.

```
Found inode with zero generation or link
Free block count wrong
Free inode count wrong
```

If there are errors, run `e2fsck` to ensure that the `ldiskfs` file structure is intact.

4.16 Lustre Users Report ENOSPC Errors

When any of the object storage targets (OSTs) that make up a Lustre file system become filled, subsequent writes to the OST may fail with an `ENOSPC` error (errno 28). Lustre reports that the file system is full even though there is space on other OSTs. This can be confusing to users, as the `df` command may report that the file system has free space available. Although new files will not be created on a full OST, write requests to existing files will fail if the write would extend the file on the full OST.

Use the `lfs setstripe` command to place files on a specific range of OSTs to avoid this problem. Disk usage on individual OSTs can also be checked by using the `lfs df` command.

4.17 Compile with Lustre API

The Lustre Application Programming Interface (API) is a library of commands used to set Lustre file properties. This API provides functions to access and/or modify settings specific to the Lustre filesystem (allocation policies, quotas, etc.).

Add Lustre API Module

Use the Linux `module` command to add the CLE 6.x Lustre API (`lustre-cray_ari_s_rhine`) to the current environment.

```
login $ module load lustre-cray_ari_s_rhine
```

Compile with Lustre API

Specify the `--libs cray-lustre-api-devel` option to compile with the Lustre API.

```
login $ cc program.c `pkg-config --cflags --libs cray-lustre-api-devel`
```

4.18 Use API to Get/Set Stripe Info

Procedure

1. Create the file with the specified stripe.

   ```
    ret = llapi_file_create (fn, stripe_size, create_offset, 1, 0);
           printf ("llapi_file_create: fn %s, size %d, offset %d, count %d, "
                    "pattern %d: returned %d\n", fn, stripe_size, create_offset,
                    1, 0, ret);
           memset (&llapi_param, 0, sizeof (llapi_param));
   ```

2. Get stripe.

   ```
   ret = llapi_getstripe (fn, &llapi_param);
           if (ret != 0) {
             my_errno = errno;
             printf ("Error %d getting stripe info for %s\n", my_errno, fn);
             line = __LINE__;
   ```

3. Validate count.

   ```
   stripe_count = llapi_param.fp_lmd->lmd_lmm.lmm_stripe_count;
           if (stripe_count != 1) {
             printf ("Error: expected stripe count 1 for %s, got %d\n",
               fn, stripe_count);
             line = __LINE__;
   ```

4. Get stripe offset of the first object.

   ```
   stripe_offset = llapi_param.fp_lmd->lmd_lmm.lmm_objects[0].l_ost_idx;
           if (stripe_offset != create_offset) {
             printf ("Error: expected stripe offset %d for %s, got %d\n",
               create_offset, fn, stripe_offset);
             line = __LINE__;
   ```

If certain I/O workloads result in RDMA fragmented errors, increase the number of scatter/gather entries (SGEs) in each InfiniBand work request. The number of SGEs is controlled with the `wrq_sge ko2iblnd` kernel module parameter, see *Use Simple Sync to Troubleshoot LNet Routers* for further guidance. Increasing the number of SGEs may result in failure to create Queue Pairs since the size of the data structures has increased.

4.19 Separation of Mount Options in lustre-utils 2.3.3

The upgrade from lustre-utils 2.3.2 to 2.3.3 separated the persistent backend file system mount options (passed to `mkfs.lustre` via "`--mountfsoptions`") from the server-specific mount options (passed to `mount.lustre` via "`-o`"). Three new definitions were created in the `fs_defs` file. These definitions are for the persistent backend file system mount options:

```
mgt_mkfs_mount_options:
mdt_mkfs_mount_options:
ost_mkfs_mount_options:
```

The following definitions will be used for only server-specific mount options:

```
mgt_mount_options:
mdt_mount_options:
ost_mount_options:
```

To upgrade from a previous release to 2.3.3, any persistent backend filesystem mount options in `{mgt,mdt,ost}_mount_options` will need to be moved to the corresponding `{mgt,mdt,ost}_mkfs_mount_options` definition prior to re-installing the `fs_defs` file. For example, if a file contains the following entry:

```
ost_mount_options: errors=remount-ro,extents,mballoc
```

Move the `errors=remount-ro`, `extents`, `mballoc` to a new `ost_mkfs_mount_options` definition, so the file contains the following two lines:

```
ost_mount_options:
```

```
ost_mkfs_mount_options: errors=remount-ro,extents,mballoc
```

Once the backend file system mount options have been moved to the appropriate definitions, reinstall the `fs_defs` using `lustre_control`:

```
lustre_control install /path/to/fs_defs
```

5 LNet Routing

LNet is a Lustre® networking layer that translates the communication protocols between the external Lustre server network (typically InfiniBand®) and the compute, service, and management node high-speed network (for example, Intel® OPA or the Cray developed Aries® high-speed network).

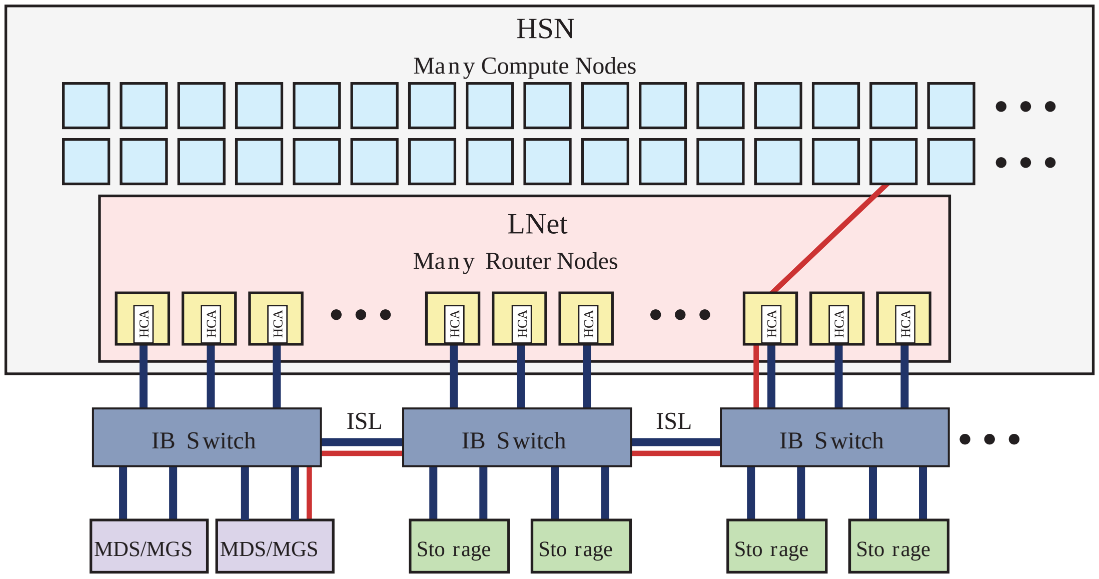
HPC systems may employ different high-speed network (HSN) technologies and storage network technologies, therefore, the LNet communication layer is required to enable each network to exchange data.

Another important role of LNet is to define what pool of LNet routers can be used to communicate with a given set of Lustre MGS, MDS, or OSS servers.

Flat Routing

In flat routing, all Lustre servers (MGS, MDS, OSSs) are assigned to a single LNet. A flat LNet is typically used by a small number of eLogin nodes or whitebox Lustre clients. All LNet routers use a round-robin method to access any Lustre server and there are no preferred pathways in flat routing. Flat routing can provide the same performance as fine-grained routing when the LNet routers are connected into the same InfiniBand leaf as the Lustre server. Communication paths will show poor performance when multiple inter-switch-link (ISL) hops are required reach the desired Lustre server. If there is a large number of routers and servers, (very common in HPC systems), flat routing limits the I/O bandwidth to/from a high-performance storage system due to the ISLs.

Figure 4. Flat Routing Block Diagram



Fine-grained Routing (FGR)

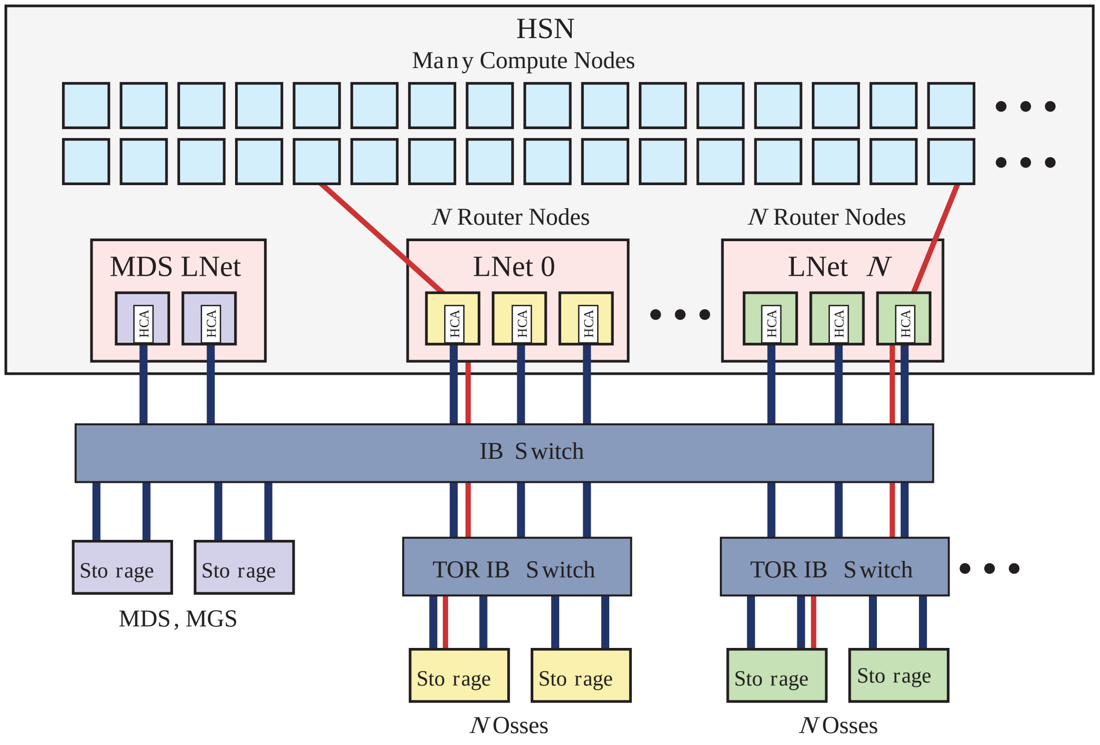
`

Fine-grained routing defines the highest performance pathway from a compute node to a Lustre server (MDS, MDS, or OSS) to:

- Make efficient use of the storage network hardware
- Maximize bandwidth between a set of LNet routers to a set of OSSs to drive I/O at the highest rate possible
- Avoid inter-switch links (ISLs)
- Scale LNet bandwidth to a group of servers linearly

Fine-grained routing divides the HSN and storage networks into groups or pools to make more efficient use of the HSN network hardware and storage system I/O bandwidth.

Figure 5. Fine-grained Routing (FGR) Block Diagram



Network Protocols

The exchange of requests and replies between hosts forms the basis of the Lustre protocol. The underlying network protocols used to exchange messages are abstracted away via the Lustre networking (LNet) software layer to isolate the file system code from the Lustre networking drivers (LNDs). LNet supports the Aries network and OPA networks using the generic network interface (GNI) Lustre network driver (`gnilnd`). Other networking technologies such as InfiniBand are supported via `o2iblnd`, and Ethernet (`socklnd`, although `socklnd` does not support RDMA). A single Lustre network can encompass multiple sub-networks through the use of LNet routers.

Routing Lustre involves configuring router nodes to route traffic between Lustre servers and Lustre clients which are on different networks. Routing Lustre requires that three types of nodes be configured: the router, the client, and the InfiniBand server. LNet assigns node IDs to each node on the network. While `gnilnd` uses node IDs (for example, *nnn*`@gni`) to enumerate ports on the Cray side, `o2iblnd` uses IB IP addresses (for example, *nn.nn.nnn.nnn*`@o2ib`) for hosts. Therefore, IPoIB must be configured on each IB port.

LNet Node Names

CLE systems use the LNet router node naming convention of `lnetN` where *N* is an ordinal starting with 1, no matter how many file systems are connected. The LNet node name may include identifying characters to indicate the place where it is used—in that case, the ordinals usually start at 1 for each set of mostly identical names. Some administrators may use the `cname`.

When routers initialize, CLE assigns their NID value as their hostname, such as `nid00022`. There is no indication of their `lnetN` name, except when translated through aliases in the `/etc/hosts` file.

5.1 LNet Tuning for CLE

Only deviations from defaults are shown. If a parameter is not specified, Cray reccomends keeping the default value specified during installation. See *Default Lustre Services Parameter Settings for CLE* on page 11 for CLE 6.0 UP07.

LNet Router Nodes

Set CLE LNet router nodes as follows:

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

LNet Parameters

`lnet:router_ping_timeout`

Default **50**. Number of seconds to wait for the reply to a router health query. Cray recommends using the default value of 50 seconds.

`lnet:live_router_check_interval`

Default **60**. Number of seconds between live router health checks. Cray recommends leaving this at the default value of 60 seconds. A value less than or equal to 0 disables pinging of live routes.

`lnet:dead_router_check_interval`

Default **60**. Number of seconds between dead router health checks. Cray recommends using the default value of 60 seconds. A value less than or equal to 0 disables pinging of dead routes.

`lnet:avoid_asym_router_failure` 1

Sonexion only (if off by default). Avoid asymmetrical router failures (0 to disable; 1 to enable).

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

Cray also recommends an OBD timeout of 100 seconds. This is the default value. It can be set via `lctl conf_param` on the MGS:

```
$ lctl conf_param fs_name.sys.timeout=100
```

e.g.

```
$ lctl conf_param husk1.sys.timeout=100
$ cat /proc/sys/lustre/timeout
100
```

Sonexion 3000 Nodes with EDR InfiniBand

The recommendations are the same as for the Sonexions with FDR InfiniBand except:

`ko2iblnd:peer_credits` 16

Enter the value for the `ko2iblnd` parameter `peer_credits`. This is the number of concurrent sends to a single peer. This value must be consistent across all peers on the IB network, meaning it must be the same on the external login clients and the Lustre file system servers.

`ko2iblnd:map_on_demand`

Default **0**. Controls the use of fast memory registration (FMR). Cray recommends setting this value to 0 for InfiniBand HCAs or a value of 32 for Intel® Omni-Path (OPA) host fabric interfaces (HFI).

Internal DAL Server Nodes

Set internal DAL server nodes as follows:

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

Internal DAL Clients (Compute and Login Nodes)

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

DVS Nodes

Cray DVS nodes should use defaults settings CLE 6.0 UP07. See *Default Lustre Services Parameter Settings for CLE* on page 11.

External Clients (eLogin or Whitebox)

When connecting and external eLogin or whitebox node to a Sonexion 3000 with EDR InfiniBand, or if using EDR HCAs, set `peer_credits=16` and `map_on_demand=0`.

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

LNet Router Buffers

This section provides additional detail on general LNet parameters.

- Routers only
- Sets the number of large (> 1 page) buffers on a router node
- Divided equally among CPTs

`lnet:large_router_buffers`

Default **1024**. Number of large (greater than 1 page) messages to buffer in the router. Cray recommends setting this to 1024 on LNet routers.

- Routers only
- Sets the number of small (1 page) router buffers on a router node
- Divided equally among CPTs

`lnet:small_router_buffers`

Default **16384**. Number of small (1 page) messages to buffer in the router. Cray recommends setting this to 16384 on LNet routers. Small router buffers are cheap (4KiB), so do not be afraid to increase this value.

PTLPRC Module Parameters

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

`ldlm_enqueue_min` sets the minimum amount of time a server waits to see traffic on a lock before it assumes a client is misbehaving and takes action to evict the client and revoke the lock. This value should be large enough that clients are able to resend an RPC from scratch without—in the event that the first RPC was lost—being evicted. The time it takes for an RPC to be sent is the sum of the network latency and the time needed for the server to process the request. In Lustre, both of these variables have a lower bound of `at_min`. Additionally, it should be large enough such that clients are not evicted as a result of HSN quiesce period. Thus the minimum value is calculated as:

```
ldlm_enqueue_min = max(2*net latency, net latency + quiesce duration) \
+  2*service time = max(2*40, 40 + 140) + 2*40 = 180 + 80 = 260
```

The quiesce duration of 140 in the above equation was determined experimentally. It could be smaller or larger depending on the nature of the HSN failure or the size of the system. We strive to strike a balance between resiliency of Lustre against extended network flaps (larger `ldlm_enqueue_min`) and the ability for Lustre to detect misbehaving clients (smaller `ldlm_enqueue_min`).

o2iblnd Module Parameters

- Router only
- Gives more peer buffers to each of the **server** nodes that the router is talking too
- Recommended value also 256 in bz 23575

`ko2iblnd:peer_buffer_credits`

Default **128**. Number of per-peer router buffer credits. Cray recommends setting this to 128.

- Servers and clients ONLY
- Peer Health is enabled by default in the code and is not what we want for servers or clients that are on Infiniband

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

- Routers ONLY - this controls how long the router waits to see traffic on the IB interfaces before it decides that interface is 'dead'
- ○ Feeds to asymmetric router failure detection data that is returned to clients
- Tunes default `peer_health` value (180s) to correct values
  - router `ko2iblnd_peer_timeout = (server's ko2iblnd_timeout + server's ko2iblnd_keepalive)` (`ko2iblnd peer_timeout` value should be at least twice the value of `ko2iblnd keepalive` option)

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

- Router only
- Increase number of concurrent sends

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

See *https://cug.org/proceedings/attendee_program_cug2012/includes/files/pap166.pdf*

If `concurrent_sends` is set to 0 (the default), then its value is calculated in the following way:

- If `map_on_demand > 0` and `map_on_demand <= LNET_MAX_IOV / 8 (256/8=32)`
  - Then `concurrent_sends` is set to `peer_credits` X 2, (8X2=16 with default values)
- else
  - `concurrent_sends` is set to `peer_credits` (8 with default values)

If `concurrent_sends` is non-zero, it has a max value of 2 X `peer_credits` (8X2=16 with default values), and a minimum value of `peer_credits / 2)` (8/2=4 with default values). If there are many peers on an IB fabric `concurrent_sends` may need to be lowered.

kgnilnd Module Parameters

`kgnilnd:peer_health`

Default **true**. A router-only option which indicates whether or not to enable the peer timeout used for LNet peer health. Cray recommends setting this to true.

`kgnilnd:credits`

Default **2048**. Number of concurrent sends allowed by the `gnilnd`. Cray recommends setting this to 2048.

5.1.1 Recommended LNet Router Node Parameters

LNet routers are service nodes connected to both the Aries high-speed network (HSN) and an external network, such as InfiniBand (IB). LNet routers route Lustre traffic and are dedicated to bridging the different networks to connect Lustre clients on the high-speed network (HSN) (compute, service, and management nodes) to Lustre servers on the external network.

Recommended LNet parameters for XC™ Series router nodes are shown below.

If the LNet router contains an mlx5-based host channel adapters (HCAs), `peer_credits` and `concurrent_sends` cannot be set to 126 and 63, respectively. For such a configuration, Cray recommends that `peer_credits` and `concurrent_sends` both be set to 16. Any IB peer on the Lustre network, such as a Lustre server or external Login node, must have `peer_credits` and `concurrent_sends` set to match the values on the LNet routers. For some mlx5 HCAs, it may be possible that `peer_credits` can be increased to 84 and `concurrent_sends` can be increased to 42.

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:peer_buffer_credits`

Default **128**. Number of per-peer router buffer credits. Cray recommends setting this to 128.

`kgnilnd:credits`

Default **2048**. Number of concurrent sends allowed by the `gnilnd`. Cray recommends setting this to 2048.

`lnet:large_router_buffers`

Default **1024**. Number of large (greater than 1 page) messages to buffer in the router. Cray recommends setting this to 1024 on LNet routers.

`lnet:small_router_buffers`

Default **16384**. Number of small (1 page) messages to buffer in the router. Cray recommends setting this to 16384 on LNet routers. Small router buffers are cheap (4KiB), so do not be afraid to increase this value.

OPA LND Default Module Parameters for `ko2iblnd`

`ko2iblnd:peer_credits_hiw`

Default **0**. The `peer_credits_hiw` parameter defines when to eagerly return credits. Cray recommends the default setting of 0 for InfiniBand HCAs or a value of 64 for OPA HFIs.

`ko2iblnd:map_on_demand`

Default **0**. Controls the use of fast memory registration (FMR). Cray recommends setting this value to 0 for InfiniBand HCAs or a value of 32 for Intel® Omni-Path (OPA) host fabric interfaces (HFI).

`ko2iblnd:fmr_pool_size`

Default **512**. Size of FMR pool on each CPT (>=`ntx/4`). Cray recommends setting this value to 512 for InfiniBand HCAs or a value of 2048 for OPA HFIs.

`ko2iblnd:fmr_flush_trigger`

Default **384**. Number of dirty FMRs that triggers a pool flush. Cray recommends setting this value to the default of 384 for InfiniBand HCAs or a value of 512 for OPA HFIs.

`ko2iblnd:fmr_cache`

Default **1**. Used to enable FMR caching. Cray recommends setting this to 1. Cray recommends this default setting for both InfiniBand HCA's or OPA HFIs.

5.1.2 External Server Node Recommended LNet Parameters

These LNet parameters are recommended for external Lustre server settings to configure Lustre nodes connected to external Sonexion storage systems. They are set from the Sonexion management node. Refer to the *Sonexion Administrator Guide* for more information. If any host channel adapters (HCAs) in the Lustre network are mlx5-based, `peer_credits` and `concurrent_sends` cannot be set to 126 and 63. Cray recommends that `peer_credits` and `concurrent_sends` both be set to 16.

Cray recommends an object-based disk (OBD) timeout of 100 seconds, which is the default value. Set this parameter using the `lctl conf_param` on the management server (MGS).

```
$ lctl conf_param fs_name.sys.timeout=100
```

For example:

```
$ lctl conf_param husk1.sys.timeout=100
$ cat /proc/sys/lustre/timeout
100
```

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`lnet:router_ping_timeout`

Default **50**. Number of seconds to wait for the reply to a router health query. Cray recommends using the default value of 50 seconds.

`lnet:live_router_check_interval`

Default **60**. Number of seconds between live router health checks. Cray recommends leaving this at the default value of 60 seconds. A value less than or equal to 0 disables pinging of live routes.

`lnet:dead_router_check_interval`

Default **60**. Number of seconds between dead router health checks. Cray recommends using the default value of 60 seconds. A value less than or equal to 0 disables pinging of dead routes.

`lnet:avoid_asym_router_failure` 1

Avoid asymmetrical router failures (0 to disable; 1 to enable).

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

5.1.3 External Server Node: Sonexion 3000 or ClusterStor L300 System Recommended Parameters

For Sonexion 3000 or ClusterStor L300 systems, the `peer_credits` setting must be consistent across all InfiniBand (IB) peers on the Lustre network. When routers and/or external Lustre clients have mlx5-based Host Channel Adapters (HCAs), `map_on_demand` must be set to 0. In addition, Cray recommends `peer_credits` and `concurrent_sends` be set to 16. Thus, when connecting to a Sonexion 3000/ClusterStor L300, the recommended parameters are generally the same as those for the Sonexion 900, 1600, and 2000—except that `map_on_demand` must be set to 0, and `peer_credits` and `concurrent_sends` should be set to 16 for all IB peers on the Lustre network.

If an IB peer must have access to an mlx4-based file system (i.e. Sonexion 900, Sonexion 1600, and Sonexion 2000) and an mlx5-based file system (i.e. Sonexion 3000), the `ko2iblnd` parameters of all mlx4 peers must match the `ko2iblnd` mlx5-peer parameters to ensure shared mlx4- and mlx5-peer function. For example, in a system where an external Login node needs access to a Sonexion 2000 and Sonexion 3000, all mlx4- and mlx5-peer `ko2iblnd` parameters should match the LNet parameters recommended for a Sonexion 3000.

For systems that have a Sonexion 3000/ClusterStor L300 running at running software version 2.1-SU003 or greater, `peer_credits` can be increased to 84 and `concurrent_sends` can be increased to 42. All IB peers within the Lustre network must be able to support these same values if they are to be used.

Cray recommends an object-based disk (OBD) timeout of 100 seconds, which is the default value. Set this parameter using the `lctl conf_param` command on the management server (MGS). For example:

```
$ lctl conf_param fs_name.sys.timeout=100
$ cat /proc/sys/lustre/timeout
100
```

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits` 16

Enter the value for the `ko2iblnd` parameter `peer_credits`. This is the number of concurrent sends to a single peer. This value must be the same on all external login clients and the Lustre file system servers.

`ko2iblnd:concurrent_sends` 16

Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. This value must be the same on the external login clients and the Lustre file system servers.

`ko2iblnd:map_on_demand`

Default **0**. Controls the use of fast memory registration (FMR). Cray recommends setting this value to 0 for InfiniBand HCAs or a value of 32 for Intel® Omni-Path (OPA) host fabric interfaces (HFI).

`lnet:router_ping_timeout`

Default **50**. Number of seconds to wait for the reply to a router health query. Cray recommends using the default value of 50 seconds.

`lnet:live_router_check_interval`

Default **60**. Number of seconds between live router health checks. Cray recommends leaving this at the default value of 60 seconds. A value less than or equal to 0 disables pinging of live routes.

`lnet:dead_router_check_interval`

Default **60**. Number of seconds between dead router health checks. Cray recommends using the default value of 60 seconds. A value less than or equal to 0 disables pinging of dead routes.

`lnet:avoid_asym_router_failure 1`

Avoid asymmetrical router failures (0 to disable; 1 to enable).

5.1.4 External Client Recommended LNet Parameters

The recommended LNet parameter settings for eLogin or external clients on XC™ Series systems running CLE 6.0 UP07 are:

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits`

Default **126**. Number of concurrent sends to a single peer. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

`ko2iblnd:concurrent_sends`

Default **63**. Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. Cray recommends setting this to 63. **This value must be the same on all external login clients and the Lustre file system servers on the IB network.**

5.1.5 Internal DAL Client Recommended LNet Parameters

Recommended LNet parameters for internal DAL client settings are the default settings for Cray CLE 6.0 UP07 systems.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

5.1.6 Internal Server (DAL) Recommended LNet Parameters

Recommended CLE 6.0 UP07 internal DAL server settings are the default settings.

`ptlrpc:data.at_max`

Default **400**. Adaptive timeout maximum in seconds. Set to 0 to disable adaptive timeouts.

`ptlrpc:data.at_min`

Default **40**. Adaptive timeout minimum in seconds.

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

5.1.7 Configure Lustre PTLRPC `ldlm_enqueue_min` Parameter

`ptlrpc:ldlm_enqueue_min`

Default **260**. Lock enqueue timeout minimum in seconds.

The `ldlm_enqueue_min` parameter sets the minimum amount of time a server waits to see traffic on a lock before assuming a client is malfunctioning, revoking the lock, and evicting the client. Set this value large enough such that clients are able to resend an RPC from scratch without being evicted in the event that the first RPC was lost. The time it takes for an RPC to be sent is the sum of the network latency and the time it takes for the server to process the request. Both of these variables have a lower bound of `at_min`. Additionally, it should be large enough so that clients are not evicted as a result of the high-speed network (HSN) quiesce period. Thus, the minimum value is calculated as:

```
ldlm_enqueue_min = max(2*net latency, net latency + quiesce duration) + 2*service time = max(2*40, 40
+ 140) + 2*40 = 180 + 80 = 260
```

The quiesce duration of 140 in the above equation was determined experimentally. It could be smaller or larger depending on the nature of the HSN failure or the size of the system. The quiesce duration of 140 strikes a balance between Lustre resiliency against extended network flaps (larger `ldlm_enqueue_min`) and Lustre ability to detect malfunctioning clients (smaller `ldlm_enqueue_min`).

The default setting for `ldlm_enqueue_min` on CLE 6.0 UP07 is 260.

5.1.8 Lustre Performance Variability and Network Congestion

This issue was patched in releases CLE 6.0 UP01 and above.

LNet router nodes do not coordinate get requests when relatively few compute nodes write data to a significant number of nodes. If a compute node cannot satisfy the many "get" requests, these requests can backup onto the Aries network. The get requests weill attempt to re-route around the Aries network congestion and cause further Aries network congestion. To correct this issue, change the settings for kgnilnd credits for compute nodes and configure kgnilnd `bte_get_dlvr_mode` parameter for LNet nodes.

Modify the configuration set using `cfgset`, use the Cray Simple Sync service (`cray_simple_sync`), or an ansible play to re-configure LNet routers. See the *XC Series System Administration Guide* for information about how to re-configure the system.

To configure new `kgnilnd` parameters for compute nodes and LNet nodes to use adaptive_3 routing:

Configure kgnilnd Credits

The first task is to configure compute node credits. Add the following to `test_lnet.conf` for compute nodes on the SMW:

```
options kgnilnd credits=64
```

```
smw:/var/opt/cray/imps/config/sets/Config_Set/files/simple_sync/platform/compute/
files/etc/modprobe.d/test_lnet.conf
```

Reboot the compute nodes and verify the option is set:

```
# cat /sys/module/kgnilnd/parameters/credits
64
```

Configure kgnilnd bte_get_dlvr_mode Parameter

Use the LNet node group to apply the feature to only routers within the `lnet_flat_routers` node group. For example, add the following line to `test_lnet.conf` for the node group configuration:

```
options kgnilnd bte_get_dlvr_mode=0x0080
```

```
smw:/var/opt/cray/imps/config/sets/Config_Set/files/simple_sync/nodegroups/
lnet_flat_routers/files/etc/modprobe.d/test_lnet.conf
```

Reboot the LNet routers and verify the settings:

```
# cat /sys/module/kgnilnd/parameters/bte_get_dlvr_mode
 128
```

Also see *Use Simple Sync to Troubleshoot LNet Routers*.

5.1.9 DVS Server Node Recommended LNet Parameters

Use the default settings for DVS server nodes on Cray CLE 6.0 UP07 systems.

5.1.10 LNet Router Pinger Parameter Settings

The router pinger determines the status of configured routers so that bad (dead) routers are not used for LNet traffic.

The ping timeout is set by the `router_ping_timeout`, `dead_router_check_interval`, and `live_router_check_interval` module parameters.

The router pinger is enabled on clients and servers when the LNet module parameters `live_router_check_interval` and `dead_router_check_interval` have values greater than `0` (default setting for CLE 6.0 UP07 is 60). The router pinger is always enabled on routers, though it is typically only used to update the status of local network interfaces. This means it **does not** do any pinging. In multi-hop configurations (server->router1->router2->client), the router pinger on a router behaves similarly to its behavior on other nodes, meaning it **does** do pinging.

The `router_checker` thread (router pinger) periodically sends traffic (an LNet ping) to each known router. Live routers are pinged (in seconds) every `live_router_check_interval`. Dead routers are pinged (in seconds) every `dead_router_check_interval`. If a response is not received from an alive route after a timeout period, then the route is marked down and is not used for further LNet traffic. Dead routes are marked alive once a response is received.

The `router_checker` is also integral in the use of asymmetric routing failure (ARF). The payload of the ping reply contains the status (up or down) of each router network interface. This information is used to determine whether a particular router should be used to communicate with particular remote network.

The effective maximum timeout is `router_pinger_timeout` + MAX(`dead_router_check_interval`, `live_router_check_interval`).

In the recommended tunings, 50 + MAX(60, 60) = 110 seconds.

5.1.11 LNet Peer Health Parameter Settings

Peer health queries the interface when a peer is on to determine whether it is alive or dead before allowing traffic to be sent to that peer. If a peer is dead, then LNet aborts the send. This functionality is needed to avoid communication attempts with known dead peers (which wastes network interface credits, router buffer credits, and other resources that could otherwise be used to communicate with alive peers).

Enable peer health by setting these Lustre network driver (LND) module parameters:

- `kgnilnd` (set `peer_health` to `true`)
- `ko2iblnd` (set the `peer_timeout` parameter to `0` to disable peer health)

When the Lustre network driver (LND) completes a transmit, receive, or connection setup operation for a peer, it records the current time in a `last_alive` field associated with the peer. When a client of LNet (for example, `ptlrpc`) attempts to send anything to a particular peer, the `last_alive` value for that peer is inspected and, if necessary, updated by querying the LND. The LND query serves a dual purpose—in addition to dropping a message, it causes the LND to attempt a new connection to a dead peer. If the `last_alive` is more than `peer_timeout` seconds (plus a fudge factor for `gnilnd`), then the peer is considered dead and the message is dropped.

Routed Configurations

For routed configurations, disable peer health on clients and servers. Clients and servers always have a peer in the middle (the router) and router aliveness is determined by the router checker feature. Peer health interferes with the normal operation of the router checker thread by preventing the router checker pings from being sent to dead routers. Thus, it would be impossible to determine when dead routers become alive again.

5.1.12 LNet Asymmetric Router Failure (ARF) Detection

Asymmetric router failure (ARF) detection enables Lustre networking (LNet) peers to determine if a route to a remote network is alive.

Peers use only known good routes. Attempting to send a message via a bad route generally results in a communication failure and requires a resend. Sending via a bad route also consumes router resources that could be utilized for other communication.

ARF detection is enabled by setting `avoid_asym_router_failure=1` in the LNet module settings. This feature piggy-backs off the router pinger feature. The ping reply sent from router to clients and servers contains information about the status of the router network interfaces for remote networks. Clients and servers then use this information to determine whether a particular router should be used when attempting to send a message to a remote network.

For example, assume a router at `454@gni` with an IB interface on `o2ib@1000` and another IB interface on `o2ib@1002`. Suppose this router responds to a router checker ping with the following information:

```
o2ib@1000 -> down
o2ib@1002 -> up
```

When a client wants to send a message to a remote network, it considers each configured router in turn. When considering `454@gni`, the client knows that this router can be used to send a message to `o2ib@1002` but not to `o2ib@1000`.

5.2 Configure LNet Fine-Grained Routing for XC Systems

Tasks for Configuring Fine-grained Routing

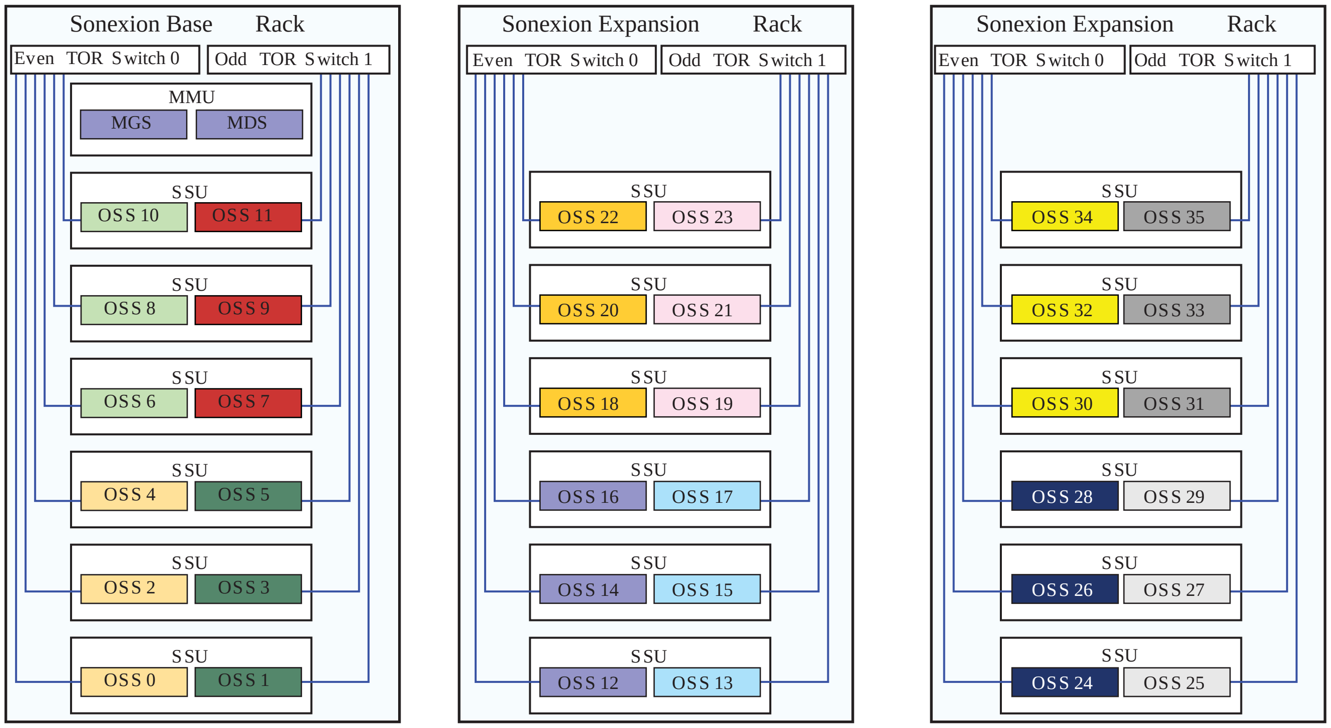
The following tasks must be completed to configure fine-grained routing (FGR) on Cray and Sonexion systems:

- Use bandwidth matching to get the router-to-server ratio
- Determine the IP addressing scheme to support the system configuration on the Cray and Sonexion systems
- Use CLCVT to generate the LNet configuration (`lnet.conf`, `routes.conf`, and `ip2nets.conf`)
- Configure IP addressing on IB interfaces (IPoIB)
- Place the LNet configuration the CLE system
- Place LNet configuration on the Sonexion
- Verify and test the configuration

Sonexion File Systems

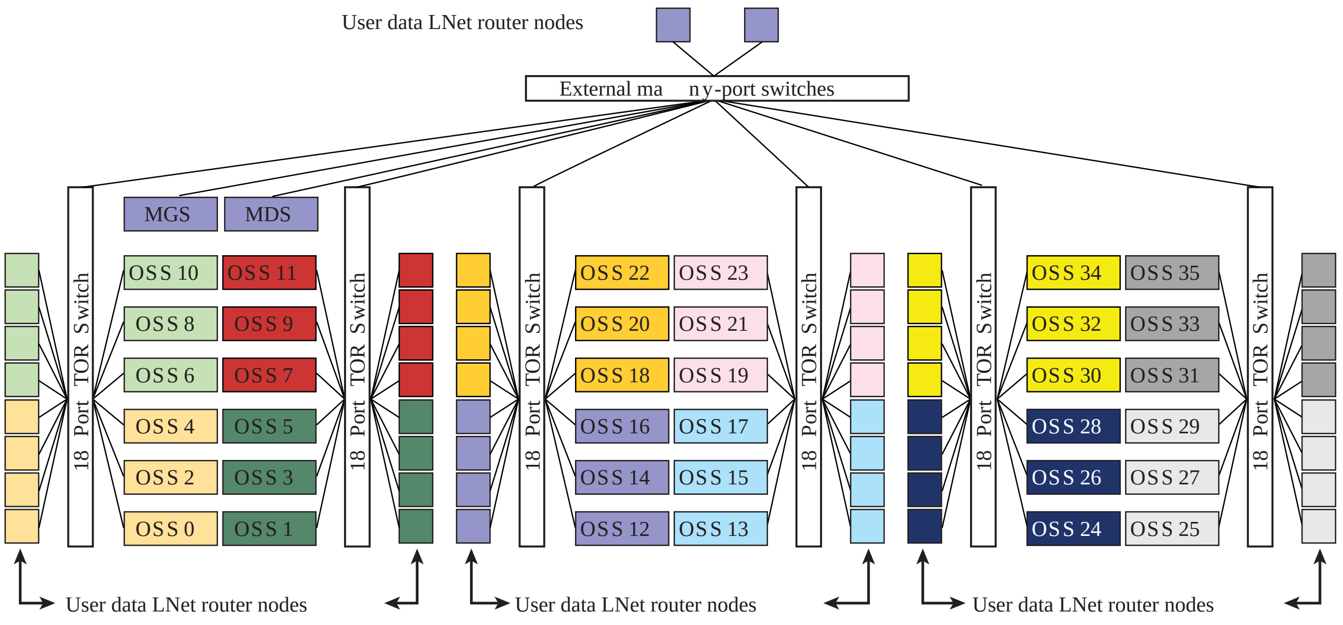
Sonexion file systems are made of two basic building blocks: scalable storage units (SSUs) and a metadata management unit (MMU). Each SSU houses disk drives and two OSSs that are connected to an even or odd top-of-rack IB switch and then to the system storage network. The MMU houses the MDS/MGS server pair and each file system includes an MDS/MGS server pair.

Figure 6. Sonexion File System Components



LNet FGR 3-Rack Example on page 59 shows how FGR is configured for three Sonexion racks. Each color represents an LNet group that is optimized to match the maximum bandwidth capacity for a Sonexion OSS. Four Cray LNet router nodes support three Sonexion OSSs in this example.

Figure 7. LNet FGR 3-Rack Example



InfiniBand Infrastructure

Cray uses industry standard fourteen data rate (FDR) InfiniBand infrastructure (although LNet supports other HSNs). The FDR InfiniBand network infrastructure (IBNI) is typically composed of dual InfiniBand core switches, and redundant pathways. There are two independent paths for all functionality accessed through the IBNI. This is accomplished with the use of active-active pairing for every function. In the event of an IBNI (or node) failure removing access to a node, the active partner(s) will automatically take over operation. Once the failure is resolved, the failed node is placed back into service. All components can be serviced, to include removal/ replacement, in the event of a single failure without loss of access to the file system.

InfiniBand subnet routing must be implemented that does not result in credit loops, (deadlock avoidance) and that high-bandwidth connections take paths through the subnet to avoid bandwidth conflicts. Connections from the HPC systems are typically split, to allow maximum bandwidth in the event of a total switch failure.

Lustre LNet routers can connect two distinct InfiniBand fabrics. Any number of LNet routers may be used in parallel to achieve the desired performance level. The hardware to create such an LNet router can be as simple as a 1U rackmount server with two InfiniBand HCA interfaces and standard Lustre software.

XC Series Client Interface

The connection from the XC Series system to the Sonexion is accomplished with a set of LNet router nodes attached to the internal Aries® HSN. LNet routers typically have FDR InfiniBand connections to director class switches (evenly split between the two). For any given file system, the Lustre clients use all appropriate and available LNet routers in a round-robin fashion. The effect of this is to add together the performance of each useable LNet router for a total available bandwidth number. The LNet routers demonstrate nearly linear performance scaling as they act in parallel.

A single XC Series I/O blade supports two independent I/O nodes that each run the Cray Linux Environment (CLE). Each I/O node supports two PCIe Gen3 x8 slots, each can support single or dual-port IB HCAs, to provide two compute node LNet routers.

A single port reaches the maximum available PCIe bandwidth. Each HCA is capable of approximately 6GB/s of raw IB data transfer. In the LNet router configuration, this translates into approximately 5.5GB/s of Lustre data traffic. However—due to interference between the two HCAs and the limits of the HSN interface—when both HCAs are active with Lustre traffic, the total is 8.3GB/s (or 4.15GB/s for each of the two IB links).

> **IMPORTANT:** Due to routing restrictions imposed by both Lustre LNet and the Internet Protocol (IP), which provides the underlying node addressing, each of the two HCA ports must be assigned to a different IP subnet and LNet o2ib instance number.

In addition, the IP subnet must be different for the HSN interface. This allows the LNet router to distinguish the appropriate exit path for any packet coming into the router. The IBNI typically includes dual-core switches and is designed to maintain the highest possible performance during any failure. Therefore, each of the two HCAs on a LNet router node are connected to different core switches. When a switch fails, the normal 4.15GB/s available to each port can become 5.5GB/s on the port surviving the failure.

The address assignments for each file system must use unique subnets. Therefore, any given LNet router IB link services only the file system that matches the router-assigned subnet. Because the two links on a single LNet router node must have different subnets, it follows that each LNet router node services two different file systems.

Metadata Server Restrictions

Lustre metadata servers (MDS/MGS) require special consideration. The traffic for metadata is very latency sensitive, but does not use much bandwidth. If this traffic were to be interleaved with normal read/write data traffic, it would be subject to the transmission delays imposed by large data transfers. These are redundant connections, and either of the two links for a single file system is sufficient to support operations. Each of the two links for a single file system is connected to a different core switch, allowing operations to proceed in the event of either a LNet router or a switch failure.

Distributing links across I/O blades accommodates system service procedures. If an LNet router node fails, the entire I/O blade may be serviced to resolve the problem. Placement of I/O blades purposed as LNet routers is distributed across different cabinets so that any failure affecting an entire cabinet will only take a single pair of LNet router nodes out of service.

5.2.1 Routing and Bandwidth Matching for Sonexion Systems

Cray recommends two IB links per router node FDR InfiniBand links on XC Series systems.

Sonexion bandwidth is evenly spread across all OSS nodes. Peak usable bandwidth is 2.4GB/s per OSS. External connections to the TOR switches must allow for this peak plus additional overhead, for a total of 3.3GB/s per OSS. A single InfiniBand FDR link can carry two OSSs worth of traffic (nominally 6.8GB/s). It is desirable to have an even number of links per TOR switch. The bandwidth of the HSN LNet nodes are matched to the bandwidth of the of the OSS nodes on the IB network for optimal throughput.

RESTRICTION: When two IB interfaces are used for each LNet router node, each IB interface must be on separate IPv4 subnets.

Sonexion storage systems are composed of two basic building blocks—a single Metadata Management Unit (MMU) and one or more Scalable Storage Units (SSUs). An optional building block, the additional distributed namespace environment (DNE) unit (ADU), is also available. The MMU consists of two management servers (MGS) and two metadata servers (MDSs) and either a 2U24 or 5U84 drive enclosure. An SSU consists of two OSSs with a 5U84 drive enclosure.

An ADU consists of two MDSs with a 2U24 drive enclosure. The Sonexion 1600 MDRAID and GridRAID systems provide 5GB/s per SSU sustained, and 6GB/s per SSU peak. The Sonexion 2000 system provides 7.5GB/s per SSU sustained, and 9 GB/s per SSU peak. Ensuring sufficient network bandwidth to each OSS is a key requirement in the design of Lustre networks.

An XC Series I/O blade has one Aries Network Interface Controller (NIC) that provides an I/O module (IBB) with 17GB/s of I/O bandwidth. That IBB includes two nodes (LNet routers), so each node can achieve 8.5GB/s. Each LNet router node can support one or two, single- or dual-port IB host channel adapters (HCAs). Each active HCA port (`ib0` and `ib2`, for instance) must be assigned to a different LNet (cannot bond them to a single LNet). Therefore, a single LNet router node services two different LNets. A single FDR (Fourteen Data Rate) IB HCA is capable of 5.5GB/s of LNet traffic. Therefore, if busy, the two IB HCAs on a single LNet router split 8.5GB/s and achieve 4.25GB/s per IB port.

Because a single FDR IB link provides sufficient bandwidth for a single Sonexion 2000 OSS, the ratio of *n* IB links to *n* servers would work. However, in the case of the Sonexion 1600, this results in wasted bandwidth. *Bandwidth Capabilities of 6 Sonexion 2000 OSSs (3 SSUs) (22.5GB/s) 5 IB Links from Single HCA Routers (27.50GB/s)* on page 62 indicates that six Sonexion 1600 OSSs can deliver 18GB/s peak I/O bandwidth. If six single HCA LNet router links are assigned to service the six servers, then the fabric is capable of 33GB/s. This is nearly double the bandwidth that six OSSs can deliver. If IB links from LNet routers with dual HCAs are assigned, then the configuration would provide 23% more bandwidth than what is needed.

Assigning two single HCA LNet router links (or three IB links from dual HCA routers) to every four Sonexion-1600 servers is ideal from the perspective of minimizing wasted bandwidth. This would either result, however, in FGR groups that span more than one TOR switch or would require additional sub-optimal FGR groups that contain just two servers. If only FGR groups are considered (where the number of servers in each FGR group is evenly divisible by the number of servers in each TOR switch) then this restricts the configurations to ratios with one, two, three or six servers.

With the above constraints in mind, suitable ratios are:

- Sonexion 1600
  - XC 40 Single HCA: 2:3
  - XC 40 Dual HCA: 5:6, or n:n+1
- Sonexion 2000
  - XC 40 Single HCA: n:n
  - XC 40 Dual HCA: n:n

Sonexion 2000 Bandwidth Matching

The bolded table cells in each table show the best grouping of Cray XC Series LNet nodes and Sonexion OSS nodes over single or dual InfiniBand HCAs. Always assign more bandwidth in LNet router nodes than is provided by the disk I/O to/from the OSSs.

- Based on the IOR benchmark with Cray chosen parameters and read/write averaging per SSU, the bandwidth is 6.0 GB/sec per SSU (3.0GB/s per OSS) for 2TB drives and requires an LNet ratio of 1:1
- Based on the IOR benchmark with Cray chosen parameters and read/write averaging per SSU, without an ESU, the bandwidth is 7.5GB/s per SSU (3.75GB/s per OSS) for 4TB and 6TB drives and requires an LNet ratio of 1:1
- Based on the IOR benchmark with Cray chosen parameters and read/write averaging per SSU, with an ESU, the bandwidth is 9.0 GB/sec per SSU (4.5GB/s per OSS) for 4TB and 6TB drives and requires an LNet ratio of 7:6

Table 5. Bandwidth Capabilities of 6 Sonexion 2000 OSSs (3 SSUs) (22.5GB/s) 5 IB Links from Single HCA Routers (27.50GB/s)

	1	**2**	**3**	**4**	**5**	**6**
Sonexion 1600 OSS	3.00	6.00	9.00	12.00	15.00	18.00
Sonexion 2000 OSS	3.75	7.50	11.25	15.00	18.75	***22.50***
Single HCA	5.50	11.00	16.50	22.00	***27.50***	33.00
Dual HCA	4.20	8.40	12.60	16.80	21.00	25.20

Table 6. 6 Sonexion 2000 OSSs (3 SSUs) (22.5GB/s) 6 IB Links from Dual-HCA Routers (25.20GB/s)

	1	**2**	**3**	**4**	**5**	**6**
Sonexion 1600 OSS	3.00	6.00	9.00	12.00	15.00	18.00
Sonexion 2000 OSS	3.75	7.50	11.25	15.00	18.75	***22.50***
Single HCA	5.50	11.00	16.50	22.00	27.50	33.00
Dual HCA	4.20	8.40	12.60	16.80	21.00	***25.20***

Sonexion 3000 Bandwidth Matching

SSU Performance for IB with EDR LDN - Based on the IOR benchmark with Cray parameters and read/write averaging:

- 9GB/s per SSU for all 7200 RPM drive sizes (4.5GB/s per OSS)
- 10GB/s per SSU with an ESU, for all drive types (5GB/s per OSS)
- 12GB/s per SSU with 10K RPM HPC drives, (6GB/s per OSS)

The following table shows the number of physical connections (i.e. cables) required for connecting the TOR switches in each rack to a XC system.

Table 7. Number of Physical Connections Required for Sonexion 3000 LNet Routing

Number of SSUs	Single FDR HCA LNet Connections	Dual FDR HCA LNet Connections	Single EDR HCA Connections
7.2 RPM Drives			
1	1	2	1
2	2	3	2
3	3	4	2
4	4	5	3
5	5	6	3
6	6	7	4
7	7	8	4
10K HPC Drives			
1	2	2	1
2	3	3	2
3	4	4	2
4	5	6	3
5	6	7	4
6	7	8	4
7	8	10	5

5.2.2 External Server Node: Sonexion 3000 or ClusterStor L300 System Recommended Parameters

For Sonexion 3000 or ClusterStor L300 systems, the `peer_credits` setting must be consistent across all InfiniBand (IB) peers on the Lustre network. When routers and/or external Lustre clients have mlx5-based Host Channel Adapters (HCAs), `map_on_demand` must be set to 0. In addition, Cray recommends `peer_credits` and `concurrent_sends` be set to 16. Thus, when connecting to a Sonexion 3000/ClusterStor L300, the recommended parameters are generally the same as those for the Sonexion 900, 1600, and 2000—except that `map_on_demand` must be set to 0, and `peer_credits` and `concurrent_sends` should be set to 16 for all IB peers on the Lustre network.

If an IB peer must have access to an mlx4-based file system (i.e. Sonexion 900, Sonexion 1600, and Sonexion 2000) and an mlx5-based file system (i.e. Sonexion 3000), the `ko2iblnd` parameters of all mlx4 peers must match the `ko2iblnd` mlx5-peer parameters to ensure shared mlx4- and mlx5-peer function. For example, in a system where an external Login node needs access to a Sonexion 2000 and Sonexion 3000, all mlx4- and mlx5-peer `ko2iblnd` parameters should match the LNet parameters recommended for a Sonexion 3000.

For systems that have a Sonexion 3000/ClusterStor L300 running at running software version 2.1-SU003 or greater, `peer_credits` can be increased to 84 and `concurrent_sends` can be increased to 42. All IB peers within the Lustre network must be able to support these same values if they are to be used.

Cray recommends an object-based disk (OBD) timeout of 100 seconds, which is the default value. Set this parameter using the `lctl conf_param` command on the management server (MGS). For example:

```
$ lctl conf_param fs_name.sys.timeout=100
$ cat /proc/sys/lustre/timeout
100
```

`ko2iblnd:timeout`

Default **10**. The `o2iblnd` timeout in seconds. Cray recommends setting this to 10 seconds.

`ko2iblnd:peer_timeout`

Default **0**. Number of seconds without aliveness news it takes to declare a peer dead. Cray recommends setting this to 0.

`ko2iblnd:keepalive`

Default **30**. Idle time in seconds before sending a keepalive. Cray recommends setting this to 30.

`ko2iblnd:credits`

Default **2048**. Number of concurrent sends allowed by `o2iblnd`. Shared by all CPU partitions (CPT). Cray recommends setting this to 2048.

`ko2iblnd:ntx`

Default **2048**. Number of message descriptors allocated for each pool. Cray recommends setting this to 2048.

`ko2iblnd:peer_credits` 16

Enter the value for the `ko2iblnd` parameter `peer_credits`. This is the number of concurrent sends to a single peer. This value must be the same on all external login clients and the Lustre file system servers.

`ko2iblnd:concurrent_sends` 16

Determines send work queue sizing. If this option is omitted, the default is calculated based on the values of `peer_credits` and `map_on_demand`. This value must be the same on the external login clients and the Lustre file system servers.

`ko2iblnd:map_on_demand`

Default **0**. Controls the use of fast memory registration (FMR). Cray recommends setting this value to 0 for InfiniBand HCAs or a value of 32 for Intel® Omni-Path (OPA) host fabric interfaces (HFI).

`lnet:router_ping_timeout`

Default **50**. Number of seconds to wait for the reply to a router health query. Cray recommends using the default value of 50 seconds.

`lnet:live_router_check_interval`

Default **60**. Number of seconds between live router health checks. Cray recommends leaving this at the default value of 60 seconds. A value less than or equal to 0 disables pinging of live routes.

`lnet:dead_router_check_interval`

Default **60**. Number of seconds between dead router health checks. Cray recommends using the default value of 60 seconds. A value less than or equal to 0 disables pinging of dead routes.

`lnet:avoid_asym_router_failure` **1**

Avoid asymmetrical router failures (0 to disable; 1 to enable).

5.2.3 Use CLCVT to Configure Fine-Grained Routing Files

The `clcvt` command, available on the boot node and the system management workstation (SMW), aids in the configuration of Lustre networking (LNet) fine-grained routing (FGR).

The `clcvt` command requires several file-system-specific input files and generates LNet kernel module configuration information that can be used to configure the servers, routers, and clients for that file system. The utility can also create cable maps (in HTML, CSV, and human-readable formats) and validate cable connection on Sonexion systems.

See the `clcvt(8) man` page for detailed information.

5.2.3.1 CLCVT Prerequisite Files

The `clcvt` command requires several prerequisite files in order to compute the `ip2nets` and `routes` information for the specific configuration. The prerequisite files must be placed in an empty directory on the boot node or SMW, depending on where `clcvt` will be run.

Deciding how to assign which routers to which object storage servers (OSSs), what fine grained routing (FGR) ratios to use, which interface on which router to use for a Lustre networking (LNet) group, and router placement are all things that can vary greatly from site to site. LNet configuration is determined as the system is ordered and configured. See a Cray representative for the site-specific values. Use *Routing and Bandwidth Matching for Sonexion Systems* on page 61 as a guide.

`info.`*`file-system-identifier`*	A file with global file system information for the *`cluster-name`* server machine and each client system that will access it.
***`client-system`*`.hosts`**	A file that maps the client system (such as the Cray mainframe) IP addresses to unique host names, such as the boot node `/etc/hosts` file. The *`client-system`* name must match one of the clients in the `info.`*`file-system-identifier`* file.
***`client-system`*`.ib`**	A file that maps the client system LNet router InfiniBand IP addresses to system hardware `cnames`. The *`client-system`* name must match one of the `clients` in the `info.`*`file-system-identifier`* file. This file must be created by an administrator.
***`clustername`*`.ib`**	A file that maps the Lustre server InfiniBand IP addresses to cluster (for example, Sonexion) host names. The *`clustername`* name must match the `clustername` in the `info.`*`file-system-identifier`* file. This file must be created by an administrator.

***client-system*.rtrIm**	A file that contains `rtr -Im` command output (executed on the SMW) for the *`client-system`*.

5.2.3.2 Create the CLCVT info.file-system-identifier File

Create the `info.`*`file-system-identifier`* file manually. This file contains global file system information for the Lustre server machine and each client system that must have access. Based on the ratio of OSS to LNet routers in the configuration, the `[`*`clustername`*`]` section (`snx11029n` in this example) and each `[`*`client-system`*`]` section (`hera` in this example) defines which servers and routers belong to each InfiniBand (IB) subnet.

This file is in the form of a `ini` style file, and the possible keywords in the `[info]` section include `clustername`, `ssu_count`, and `clients`. Refer to the `clcvt` man page for detailed information and usage.

clustername	Defines the base name used for all file system servers. The example show a Sonexion file system `snx11029n`. Thus, all server hostnames will be `snx11029n`*`NNN`*. *`NNN`* is a three-digit number starting at `000` and `001` for the primary and secondary Cray Sonexion management servers (MGMT), `002` for the MGS, `003` for the MDS, `004` for the first OSS, and counting up from there for all remaining OSSs.
ssu_count	Defines how many SSUs make up a Sonexion file system. If this is missing, then this is not a Sonexion file system but a CLFS installation.
clients	Defines a comma-separated list of mainframe names that front-end this file system (`hera` in this example).
fgr_ratio	Determines the Fine-Grained Routing ratio (see *Routing and Bandwidth Matching for Sonexion Systems* on page 61. Set this to *`M`*`:`*`N`*, where *`M`* is the number of routers per group and *`N`* is the number of servers per group. If set, `oss_per_switch` and `ssu_per_rack` must be equal to or greater than *`N`*, and they must be set to a whole-number multiple of *`N`*. Changing the `ssu_per_rack` and `oss_per_switch` default settings can accommodate seven SSUs per rack.

```
ssu_per_rack = 7
oss_per_switch = 7
```

The `info.`*`file-system-identifier`* file requires a `[`*`client-system`*`]` section for each client system listed in the `clients` line of the `[info]` section to describe the client systems and a `[`*`clustername`*`]` section to describe the Lustre server system.

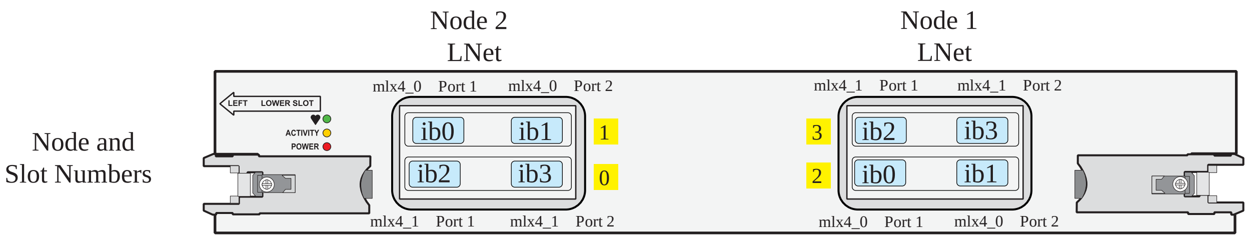
Each of these sections contain a literal LNet network wildcard in the format of *`LNET-name`*:*`IP-wildcard`* which instructs the LNet module to match a host IP address to *`IP-wildcard`* and, if it matches, instantiate LNet *`LNET-name`* on them.

The hostname fields in the `[`*`client-system`*`]` section of this file are fully qualified interface specifications of the form *`hostname`*`(ibn)`, where `(ib0)` is the assumed default if not specified.

XC systems support multiple IB interfaces per LNet router. Configure the second IB interface and append the interface names (`ib`*`n`*) to the `cname` for the routers (as shown in the example). These interface names must also be appended to the *`client-system`*`.ib` file.

XC40 IB port assignments are shown in the illustration.

Figure 8. XC40 IB Port Assignments



Example: `info.`*`file-system-identifier`* File

This example configures an `info.snx11029n` file that supports the example file system in *XC40 with Two FDR IB Interfaces Per LNet Router* on page 68. It is configured with four FDR IB interfaces (two per router node) from each XC40 LNet router blade and includes a total of 12 SSUs for the file system.

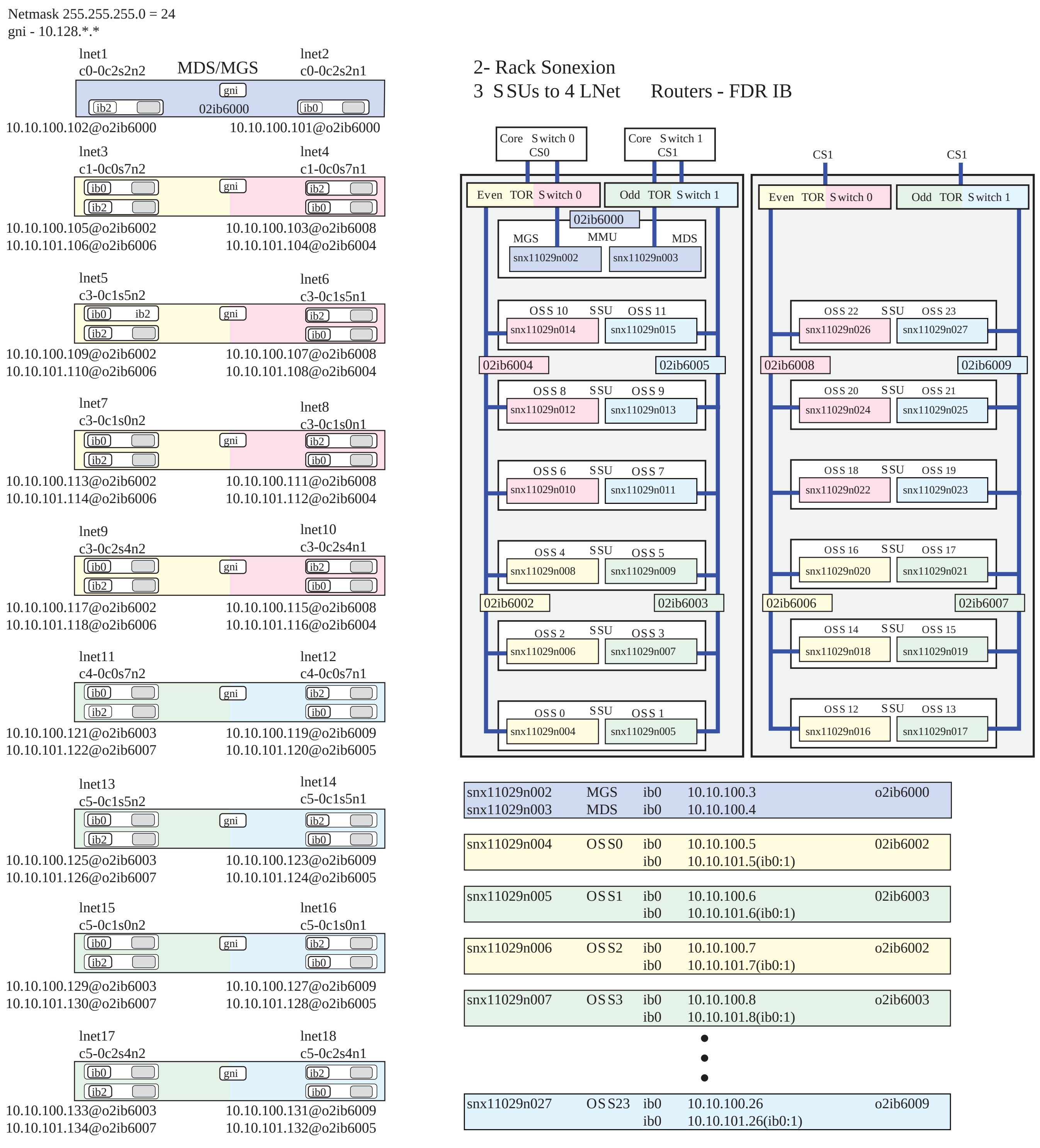
The Sonexion OSSs that connect to the ib2 interface of the LNet routers in this example must have an IP address on the ib2 subnet. The ib0:1 alias is added as label for the ib2 interfaces because there is only single IB interface on each Sonexion OSS.

```
# This section describes the size of this filesystem.
[info]
clustername = snx11029n
SSU_count = 12
clients = hera
fgr_ratio = 4:3
# other optional keywords not used in this example
# oss_count
# oss_per_switch
# sonexion
# ssu_per_rack
# fgr

[hera]
lnet_network_wildcard = gni1:10.128.*.*

# Because of our cabling assumptions and naming conventions, we only
# need to know which XIO nodes are assigned to which LNETs.  From that
# our tool can actually generate a "cable map" for the installation folks.
o2ib6000: c0-0c2s2n2(ib2), c0-0c2s2n1(ib0) ; MGS and MDS
o2ib6002: c1-0c0s7n2(ib0), c3-0c1s5n2(ib0), c3-0c1s0n2(ib0), c3-0c2s4n2(ib0) ; OSSs 0/2/4
o2ib6003: c4-0c0s7n2(ib0), c5-0c1s5n2(ib0), c5-0c1s0n2(ib0), c5-0c2s4n2(ib0) ; OSSs 1/3/5
o2ib6004: c1-0c0s7n1(ib0), c3-0c1s5n1(ib0), c3-0c1s0n1(ib0), c3-0c2s4n1(ib0) ; OSSs 6/8/10
o2ib6005: c4-0c0s7n1(ib0), c5-0c1s5n1(ib0), c5-0c1s0n1(ib0), c5-0c2s4n1(ib0) ; OSSs 7/9/11
o2ib6006: c1-0c0s7n2(ib2), c3-0c1s5n2(ib2), c3-0c1s0n2(ib2), c3-0c2s4n2(ib2) ; OSSs 12/14/16
o2ib6007: c4-0c0s7n2(ib2), c5-0c1s5n2(ib2), c5-0c1s0n2(ib2), c5-0c2s4n2(ib2) ; OSSs 13/15/17
o2ib6008: c1-0c0s7n1(ib2), c3-0c1s5n1(ib2), c3-0c1s0n1(ib2), c3-0c2s4n1(ib2) ; OSSs 18/20/22
o2ib6009: c4-0c0s7n1(ib2), c5-0c1s5n1(ib2), c5-0c1s0n1(ib2), c5-0c2s4n1(ib2) ; OSSs 19/21/23

[snx11029n]
lnet_network_wildcard = o2ib0:10.10.100.*

o2ib6000: snx11029n002(ib0), snx11029n003(ib0) ; MGS and MDS
o2ib6002: snx11029n004(ib0), snx11029n006(ib0), snx11029n008(ib0) ; OSSs 0/2/4
o2ib6003: snx11029n005(ib0), snx11029n007(ib0), snx11029n009(ib0) ; OSSs 1/3/5
o2ib6004: snx11029n010(ib0), snx11029n012(ib0), snx11029n014(ib0) ; OSSs 6/8/10
o2ib6005: snx11029n011(ib0), snx11029n013(ib0), snx11029n015(ib0) ; OSSs 7/9/11
o2ib6006: snx11029n016(ib0:1), snx11029n018(ib0:1), snx11029n020(ib0:1) ; OSS 12/14/16
o2ib6007: snx11029n017(ib0:1), snx11029n019(ib0:1), snx11029n021(ib0:1) ; OSS 13/15/17
o2ib6008: snx11029n022(ib0:1), snx11029n024(ib0:1), snx11029n026(ib0:1) ; OSS 18/20/22
o2ib6009: snx11029n023(ib0:1), snx11029n025(ib0:1), snx11029n027(ib0:1) ; OSS 19/21/23
```

The example shows two interfaces for the MDS/MGS nodes on a single blade. If the metadata server must be on a 3rd IPv4 subnet, then the additional interfaces would be placed on other LNet nodes.

Move the `info.snx11029n` file to the working directory where the `clcvt` command will be run.

Figure 9. XC40 with Two FDR IB Interfaces Per LNet Router



snx11029n002	MGS	ib0	10.10.100.3	o2ib6000
snx11029n003	MDS	ib0	10.10.100.4	
snx11029n004	OSS0	ib0	10.10.100.5	02ib6002
		ib0	10.10.101.5(ib0:1)	
snx11029n005	OSS1	ib0	10.10.100.6	02ib6003
		ib0	10.10.101.6(ib0:1)	
snx11029n006	OSS2	ib0	10.10.100.7	o2ib6002
		ib0	10.10.101.7(ib0:1)	
snx11029n007	OSS3	ib0	10.10.100.8	o2ib6003
		ib0	10.10.101.8(ib0:1)	
•				
•				
•				
snx11029n027	OSS23	ib0	10.10.100.26	o2ib6009
		ib0	10.10.101.26(ib0:1)	

5.2.3.3 Create the CLCVT client-system.hosts File

Use the `/etc/hosts` file from the XC boot node to create the *`client-system`*`.hosts` file (`hera.hosts`). Copy the `/etc/hosts` file from the boot node to a working directory and include the LNet router nodes. A typical `/etc/hosts` file is shown below.

Example *`client-system`*`.hosts` File

```
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost

# special IPv6 addresses
::1     ipv6-localhost  localhost       ipv6-loopback

fe00::0 ipv6-localnet

ff00::0 ipv6-mcastprefix
ff02::1 ipv6-allnodes
ff02::2 ipv6-allrouters
ff02::3 ipv6-allhosts

10.128.0.1      nid00000        c0-0c0s0n0      dvs-0
10.128.0.2      nid00001        c0-0c0s0n1      boot001 boot002
10.128.0.31     nid00030        c0-0c0s0n2      #ddn_mds
10.128.0.32     nid00031        c0-0c0s0n3      hera-rsip2
10.128.0.3      nid00002        c0-0c2s2n1      lnet2
10.128.0.4      nid00003        c0-0c2s2n2      lnet1
10.128.0.29     nid00018        c1-0c0s7n1      lnet4
10.128.0.30     nid00019        c1-0c0s7n2      lnet3
10.128.0.45     nid00024        c3-0c1s0n1      lnet8
10.128.0.46     nid00025        c3-0c1s0n2      lnet7
10.128.0.57     nid00036        c3-0c1s5n1      lnet6
10.128.0.58     nid00037        c3-0c1s5n2      lnet5
10.128.0.67     nid00046        c3-0c2s4n1      lnet10
10.128.0.68     nid00047        c3-0c2s4n2      lnet9
10.128.0.75     nid00054        c4-0c0s7n1      lnet12
10.128.0.76     nid00055        c4-0c0s7n2      lnet11
10.128.0.85     nid00067        c5-0c1s5n1      lnet14
10.128.0.86     nid00068        c5-0c1s5n2      lnet13
10.128.0.91     nid00074        c5-0c1s0n1      lnet16
10.128.0.92     nid00075        c5-0c1s0n2      lnet15
10.128.0.101    nid00093        c5-0c2s4n1      lnet18
10.128.0.102    nid00094        c5-0c2s4n2      lnet17
. . .
```

Move the *`client-system`*`.hosts` file to the working directory on the SMW or boot node where the `clcvt` command will be run.

5.2.3.4 Create the CLCVT client-system.ib File

The *`client-system`*`.ib` file contains a client-system LNet router InfiniBand (IB) IP address to `cname` mapping information in an `/etc/hosts` style format. The hostname field in this file is a fully-qualified interface specification of the form *`hostname`*`(ibn)`, where `(ib0)` is the assumed default if not specified. This file must be created by an administrator.

XC systems can support multiple IB interfaces per router—configure the second IB interface and append the interface names `(ib`*`n`*`)` to the `cname` for the routers. The LNet router IB IP addresses should be within the same subnet as the Lustre servers (MGS/MDS/OSS). One possible address assignment scheme would be to use a contiguous set of IP addresses, with `ib0` and `ib2` on each node having adjacent addresses. These interface names must be appended to the `info.`*`file-system-identifier`* file.

Example: *`client-system`*`.ib` File

This example (`hera.ib`) configures two FDR IB cards per XC system LNet router.

```
#
# This is the /etc/hosts-like file for Infiniband IP addresses
# on "hera".
#
10.10.100.101    c0-0c2s2n1(ib0)
10.10.100.102    c0-0c2s2n2(ib2)
10.10.100.103    c1-0c0s7n1(ib2)
10.10.101.104    c1-0c0s7n1(ib0)
10.10.100.105    c1-0c0s7n2(ib0)
10.10.101.106    c1-0c0s7n2(ib2)
10.10.100.107    c3-0c1s5n1(ib2)
10.10.101.108    c3-0c1s5n1(ib0)
10.10.100.109    c3-0c1s5n2(ib0)
10.10.101.110    c3-0c1s5n2(ib2)
10.10.100.111    c3-0c1s0n1(ib2)
10.10.101.112    c3-0c1s0n1(ib0)
10.10.100.113    c3-0c1s0n2(ib0)
10.10.101.114    c3-0c1s0n2(ib2)
10.10.100.115    c3-0c2s4n1(ib2)
10.10.101.116    c3-0c2s4n1(ib0)
10.10.100.117    c3-0c2s4n2(ib0)
10.10.101.118    c3-0c2s4n2(ib2)
10.10.100.119    c4-0c0s7n1(ib2)
10.10.101.120    c4-0c0s7n1(ib0)
10.10.100.121    c4-0c0s7n2(ib0)
10.10.101.122    c4-0c0s7n2(ib2)
10.10.100.123    c5-0c1s5n1(ib2)
10.10.101.124    c5-0c1s5n1(ib0)
10.10.100.125    c5-0c1s5n2(ib0)
10.10.101.126    c5-0c1s5n2(ib2)
10.10.100.127    c5-0c1s0n1(ib2)
10.10.101.128    c5-0c1s0n1(ib0)
10.10.100.129    c5-0c1s0n2(ib0)
10.10.101.130    c5-0c1s0n2(ib2)
10.10.100.131    c5-0c2s4n1(ib2)
10.10.101.132    c5-0c2s4n1(ib0)
10.10.100.133    c5-0c2s4n2(ib0)
10.10.101.134    c5-0c2s4n2(ib2)
```

Move the *`client-system`*`.ib` file to the working directory on the SMW or boot node where the `clcvt` command will be run.

5.2.3.5 Create the CLCVT cluster-name.ib File

The CLCVT *`cluster-name`*`.ib` file contains Lustre server InfiniBand (IB) IP addresses to cluster (Sonexion) host name mapping information in a `/etc/hosts` style format. This file must be created by an administrator.

The Sonexion servers that connect to the ib2 interface of the routers in this example must have an IP address on the ib2 subnet. The ib0:1 alias is added as a label for the ib2 interfaces to support the single IB interface on each Sonexion OSS.

Example *`cluster-name`*`.ib` File*`snx11029n`*.ib

```
#
# This is the /etc/hosts-like file for InfiniBand IP addresses
# on the Sonexion known as "snx11029n".
#
10.10.100.1    snx11029n000    #mgmnt
10.10.100.2    snx11029n001    #mgmnt
```

```
10.10.100.3    snx11029n002    #mgs
10.10.100.4    snx11029n003    #mds
10.10.100.5    snx11029n004    #first oss, oss0
10.10.100.6    snx11029n005    #oss1
10.10.100.7    snx11029n006    #oss2
10.10.100.8    snx11029n007    #oss3
10.10.100.9    snx11029n008    #oss4
10.10.100.10   snx11029n009    #oss5
10.10.100.11   snx11029n010    #oss6
10.10.100.12   snx11029n011    #oss7
10.10.100.13   snx11029n012    #oss8
10.10.100.14   snx11029n013    #oss9
10.10.100.15   snx11029n014    #oss10
10.10.100.16   snx11029n015    #oss11
10.10.101.17   snx11029n016(ib0:1)    #oss12
10.10.101.18   snx11029n017(ib0:1)    #oss13
10.10.101.19   snx11029n018(ib0:1)    #oss14
10.10.101.20   snx11029n019(ib0:1)    #oss15
10.10.101.21   snx11029n020(ib0:1)    #oss16
10.10.101.22   snx11029n021(ib0:1)    #oss17
10.10.101.23   snx11029n022(ib0:1)    #oss18
10.10.101.24   snx11029n023(ib0:1)    #oss19
10.10.101.25   snx11029n024(ib0:1)    #oss20
10.10.101.26   snx11029n025(ib0:1)    #oss21
10.10.101.27   snx11029n026(ib0:1)    #oss22
10.10.101.28   snx11029n027(ib0:1)    #oss23
```

Move the *`cluster-name`*`.ib` file to the working directory on the SMW or boot node where the `clcvt` command will be run.

5.2.3.6 Create the CLCVT client-system.rtrIm File

About this task

The *`client-system`*`.rtrIm` file contains output from the `rtr -Im` command as executed from the SMW. When capturing the command output to a file, use the `-H` option to remove the header information from `rtr -Im` or open the file after capturing and delete the first two lines.

Procedure

1. Log on to the SMW as `crayadm`.
2. Run the `rtr -Im` command and capture the output (without header information) to a file.

   ```
   crayadm@smw> rtr -Im -H > client-system.rtrIm
   ```

   Example *`client-system`*`.rtrIm` File

   ```
     4        4       c0-0c2s2n1    c0-0c2s2g0     0  0  0
     5        5       c0-0c2s2n2    c0-0c2s2g0     0  0  0
   221      221       c1-0c0s7n1    c1-0c0s7g0     0  3  7
   222      222       c1-0c0s7n2    c1-0c0s7g0     0  3  7
   641      769       c3-0c1s0n1    c3-0c1s0g0     1  4  0
   642      770       c3-0c1s0n2    c3-0c1s0g0     1  4  0
   661      789       c3-0c1s5n1    c3-0c1s5g0     1  4  5
   662      790       c3-0c1s5n2    c3-0c1s5g0     1  4  5
   721      849       c3-0c2s4n1    c3-0c2s4g0     1  5  4
   722      850       c3-0c2s4n2    c3-0c2s4g0     1  5  4
   797      1053      c4-0c0s7n1    c4-0c0s7go     2  0  7
   798      1054      c4-0c0s7n2    c4-0c0s7go     2  0  7
   1025     1281      c5-0c1s0n1    c5-0c1s0g0     2  4  0
   1046     1302      c5-0c1s0n2    c5-0c1s0g0     2  4  0
   1045     1301      c5-0c1s5n1    c5-0c1s5g0     2  4  5
   1046     1302      c5-0c1s5n2    c5-0c1s5g0     2  4  5
   1077     1365      c5-0c2s4n1    c5-0c2s4g0     2  4  13
   1078     1366      c5-0c2s4n2    c5-0c2s4g0     2  4  13
   ```

3. Move the `client-system.rtrIm` (`hera.rtrIm`) file to the working directory where the `clcvt` command will be run.

5.2.3.7 Generate `ip2nets` and `routes` Information

After the prerequisite files have been created, generate the `persistent-storage` file with the `clcvt generate` action. This portable file will then be used to create `ip2nets` and `routes` directives for the servers, routers, and clients.

The following procedures frequently use the `--split-routes=4` flag, which prints information that can be loaded into `ip2nets` and `routes` files. This method of adding `modprobe.conf` directives is particularly valuable for large systems where the directives might otherwise exceed the `modprobe` buffer limit.

- *Create the CLCVT persistent-storage File* on page 72
- *Create ip2nets and Routes Information for Compute Nodes* on page 72
- *Create ip2nets and routes Information for Service Node Lustre Clients (MOM and Internal Login Nodes)* on page 73
- *Create ip2nets and routes Information for LNet Router Nodes* on page 74
- *Create ip2nets and routes Information for Lustre Server Nodes* on page 74

5.2.3.7.1 Create the CLCVT persistent-storage File

Procedure

1. Move all prerequisite files to an empty directory on the boot node or SMW (the `clcvt` command is only available on the boot node or the SMW).

   The working directory should include the following files.

```
crayadm@smw$ ll
total 240
-rw-rw-r-- 1 crayadm crayadm  23707 Oct  8 14:27 hera.hosts
-rw-rw-r-- 1 crayadm crayadm    548 Oct  8 14:27 hera.ib
-rw-rw-r-- 1 crayadm crayadm  36960 Oct  8 14:27 hera.rtrIm
-rw-rw-r-- 1 crayadm crayadm   1077 Feb  8 14:27 info.snx11029n
-rw-rw-r-- 1 crayadm crayadm    662 Feb  8 14:27 snx11029n.ib
```

2. Create the `persistent-storage` file. (Use the `--debug` flag to display debugging information.)

```
crayadm@smw$ clcvt generate
INFO:LNET_filesystem.load: clustername = snx11029n
INFO:LNET_filesystem.load: This is a SONEXION
INFO:LNET_filesystem.load: 12 SSU, 6 SSU per rack
INFO:LNET_filesystem.load: You have chosen to use Fine Grained Routing.
INFO:LNET_filesystem.load: The ratio of LNET router connections to OSSs is '4:3'.
```

5.2.3.7.2 Create ip2nets and Routes Information for Compute Nodes

Procedure

1. Execute the `clcvt` command with the `compute` flag to generate directives for the compute nodes.

```
crayadm@smw$ clcvt compute --split-routes=4
# Place the following line(s) in the appropriate 'modprobe' file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
```

```
options lnet ip2nets=/path/to/ip2nets-loading/filename
options lnet routes=/path/to/route-loading/filename
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate ip2nets-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
gni1 10.128.*.*
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate route-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
o2ib6000 1 [4,5]@gni1
o2ib6002 1 [222,642,662,722]@gni1
o2ib6003 1 [798,1026,1046,1078]@gni1
o2ib6004 1 [221,641,661,721]@gni1
o2ib6005 1 [797,1025,1045,1077]@gni1
o2ib6006 1 [222,642,662,722]@gni1
o2ib6007 1 [798,1026,1046,1078]@gni1
o2ib6008 1 [221,641,661,721]@gni1
o2ib6009 1 [797,1025,1045,1077]@gni1
o2ib6000 2 [221,222,641,642,661,662,721,722,797,798,1025,1026,1045,1046,1077,1078]@gni1
o2ib6002 2 [221,641,661,721]@gni1
o2ib6003 2 [797,1025,1045,1077]@gni1
o2ib6004 2 [222,642,662,722]@gni1
o2ib6005 2 [798,1026,1046,1078]@gni1
o2ib6006 2 [221,641,661,721]@gni1
o2ib6007 2 [797,1025,1045,1077]@gni1
o2ib6008 2 [222,642,662,722]@gni1
o2ib6009 2 [798,1026,1046,1078]@gni1
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2. Place the command output in the appropriate `modprobe` configuration file.

5.2.3.7.3 Create ip2nets and routes Information for Service Node Lustre Clients (MOM and Internal Login Nodes)

Procedure

1. Execute the `clcvt` command with the `login` flag to generate directives for the service node Lustre clients.

```
crayadm@smw$  clcvt login --split-routes=4
# Place the following line(s) in the appropriate 'modprobe' file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
options lnet ip2nets=/path/to/ip2nets-loading/filename
options lnet routes=/path/to/route-loading/filename
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate ip2nets-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
gni1 10.128.*.*
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate route-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
o2ib6000 1 [4,5]@gni1
o2ib6002 1 [222,642,662,722]@gni1
o2ib6003 1 [798,1026,1046,1078]@gni1
o2ib6004 1 [221,641,661,721]@gni1
o2ib6005 1 [797,1025,1045,1077]@gni1
o2ib6006 1 [222,642,662,722]@gni1
o2ib6007 1 [798,1026,1046,1078]@gni1
o2ib6008 1 [221,641,661,721]@gni1
o2ib6009 1 [797,1025,1045,1077]@gni1
o2ib6000 2 [221,222,641,642,661,662,721,722,797,798,1025,1026,1045,1046,1077,1078]@gni1
o2ib6002 2 [221,641,661,721]@gni1
o2ib6003 2 [797,1025,1045,1077]@gni1
o2ib6004 2 [222,642,662,722]@gni1
o2ib6005 2 [798,1026,1046,1078]@gni1
o2ib6006 2 [221,641,661,721]@gni1
o2ib6007 2 [797,1025,1045,1077]@gni1
o2ib6008 2 [222,642,662,722]@gni1
o2ib6009 2 [798,1026,1046,1078]@gni1
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2. Place the command output in the appropriate `modprobe` configuration file.

5.2.3.7.4 Create ip2nets and routes Information for LNet Router Nodes

Procedure

1. Execute the `clcvt` command with the `router` flag to generate directives for the LNet router nodes.

```
crayadm@smw$  clcvt router --split-routes=4
# Place the following line(s) in the appropriate 'modprobe' file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
options lnet ip2nets=/path/to/ip2nets-loading/filename
options lnet routes=/path/to/route-loading/filename
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate ip2nets-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
gni1 10.128.*.*
o2ib6000 10.10.100.[101,105,109,113,117,121,125,129,133]
o2ib6000 10.10.101.[104,108,112,116,120,124,128,132]
o2ib6000(ib2) 10.10.100.102
o2ib6002 10.10.100.[105,109,113,117]
o2ib6002 10.10.101.[104,108,112,116]
o2ib6003 10.10.100.[121,125,129,133]
o2ib6003 10.10.101.[120,124,128,132]
o2ib6004 10.10.100.[105,109,113,117]
o2ib6004 10.10.101.[104,108,112,116]
o2ib6005 10.10.100.[121,125,129,133]
o2ib6005 10.10.101.[120,124,128,132]
o2ib6006(ib2) 10.10.100.[103,107,111,115]
o2ib6006(ib2) 10.10.101.[106,110,114,118]
o2ib6007(ib2) 10.10.100.[119,123,127,131]
o2ib6007(ib2) 10.10.101.[122,126,130,134]
o2ib6008(ib2) 10.10.100.[103,107,111,115]
o2ib6008(ib2) 10.10.101.[106,110,114,118]
o2ib6009(ib2) 10.10.100.[119,123,127,131]
o2ib6009(ib2) 10.10.101.[122,126,130,134]
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate route-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
o2ib6000 1 [4,5]@gni1
o2ib6002 1 [222,642,662,722]@gni1
o2ib6003 1 [798,1026,1046,1078]@gni1
o2ib6004 1 [221,641,661,721]@gni1
o2ib6005 1 [797,1025,1045,1077]@gni1
o2ib6006 1 [222,642,662,722]@gni1
o2ib6007 1 [798,1026,1046,1078]@gni1
o2ib6008 1 [221,641,661,721]@gni1
o2ib6009 1 [797,1025,1045,1077]@gni1
o2ib6000 2 [221,222,641,642,661,662,721,722,797,798,1025,1026,1045,1046,1077,1078]@gni1
o2ib6002 2 [221,641,661,721]@gni1
o2ib6003 2 [797,1025,1045,1077]@gni1
o2ib6004 2 [222,642,662,722]@gni1
o2ib6005 2 [798,1026,1046,1078]@gni1
o2ib6006 2 [221,641,661,721]@gni1
o2ib6007 2 [797,1025,1045,1077]@gni1
o2ib6008 2 [222,642,662,722]@gni1
o2ib6009 2 [798,1026,1046,1078]@gni1
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2. Place the command output in the appropriate `modprobe` configuration file.

5.2.3.7.5 Create ip2nets and routes Information for Lustre Server Nodes

Procedure

1. Execute the `clcvt` command with the `server` flag to generate directives for the Lustre server nodes.

```
crayadm@smw$ clcvt server --split-routes=4
# Place the following line(s) in the appropriate 'modprobe' file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
options lnet ip2nets=/path/to/ip2nets-loading/filename
options lnet routes=/path/to/route-loading/filename
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate ip2nets-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
o2ib6 10.10.*.*
o2ib6000 10.10.100.[3,4]
o2ib6002 10.10.100.[5,7,9]
o2ib6003 10.10.100.[6,8,10]
o2ib6004 10.10.100.[11,13,15]
o2ib6005 10.10.100.[12,14,16]
o2ib6006(ib0:1) 10.10.101.[17,19,21]
o2ib6007(ib0:1) 10.10.101.[18,20,22]
o2ib6008(ib0:1) 10.10.101.[23,25,27]
o2ib6009(ib0:1) 10.10.101.[24,26,28]
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Place the following line(s) in the appropriate route-loading file.
#vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
gni1 1 10.10.100.[101,102]@o2ib6000
gni1 1 10.10.100.[105,109,113,117]@o2ib6002
gni1 1 10.10.100.[121,125,129,133]@o2ib6003
gni1 1 10.10.101.[104,108,112,116]@o2ib6004
gni1 1 10.10.101.[120,124,128,132]@o2ib6005
gni1 1 10.10.101.[106,110,114,118]@o2ib6006
gni1 1 10.10.101.[122,126,130,134]@o2ib6007
gni1 1 10.10.100.[103,107,111,115]@o2ib6008
gni1 1 10.10.100.[119,123,127,131]@o2ib6009
gni1 2 10.10.100.[105,109,113,117,121,125,129,133]@o2ib6000
gni1 2 10.10.101.[104,108,112,116,120,124,128,132]@o2ib6000
gni1 2 10.10.101.[104,108,112,116]@o2ib6002
gni1 2 10.10.101.[120,124,128,132]@o2ib6003
gni1 2 10.10.100.[105,109,113,117]@o2ib6004
gni1 2 10.10.100.[121,125,129,133]@o2ib6005
gni1 2 10.10.100.[103,107,111,115]@o2ib6006
gni1 2 10.10.100.[119,123,127,131]@o2ib6007
gni1 2 10.10.101.[106,110,114,118]@o2ib6008
gni1 2 10.10.101.[122,126,130,134]@o2ib6009
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2. Place the command output in the appropriate `modprobe` configuration file for the Lustre servers. (For more information, refer to the site-specific Lustre server documentation.)

5.2.4 Place the LNet Configuration on a CLE6 System

5.2.4.1 Copy Files for External Lustre Fine-grained Routing

Prerequisites

This procedure is only for systems that use an external Lustre file system. It assumes the following:

- Fine-grained routing (FGR) files have been generated by `clcvt`
- Cray LNet configuration service (`cray_lnet`) has been configured with FGR

About this task

This procedure places the `ip2nets.conf` and `routes.conf` files in the CLE config set for the LNet routers.

Procedure

1. Create an `lnet` directory under `roles` in the CLE config set directory structure.

   This example uses a config set named *`p0`*. Substitute the correct config set name for this site.

   ```
   smw# mkdir -p /var/opt/cray/imps/config/sets/p0/files/roles/lnet
   ```

2. Confirm the file names of the fine-grained routing files.

   It is possible that these two files were created with names other than `ip2nets.conf` and `routes.conf`. Check these two settings in the `cray_lnet` configuration service to see what file names are used (example settings are for config set *`p0`* and a file system with key *`sonexion`*).

   ```
   smw# cfgset search -l advanced -s cray_lnet -t fgr_routes p0
   # 2 matches for 'fgr_routes' from cray_lnet_config.yaml
   #--------------------------------------------------------------
   cray_lnet.settings.fgr_routes.data.sonexion.ip2nets_file: ip2nets.conf
   cray_lnet.settings.fgr_routes.data.sonexion.routes_file: routes.conf
   ```

3. Copy the `ip2nets.conf` and `routes.conf` files to the `lnet` directory.

   ```
   smw# cd directory_containing_ip2nets.conf_and_routes.conf

   smw# cp -p ip2nets.conf routes.conf /var/opt/cray/imps/config/sets/p0/files/roles/lnet
   ```

5.2.4.2 Configure LNet Routers for Fine-Grained Routing

Prerequisites

Fine-grained routing files have been configured using `clcvt`.

About this task

This procedure modifies the *Cray System Management Configuration Worksheet* for the `cray_lnet` service in the config set to configure fine-grained routing (FGR).

Procedure

1. As `root`, use `cfgset` to modify the `cray_lnet` service in the configuration set.

   ```
   smw# cfgset update --service cray_lnet --mode interactive partition
     cray_lnet        [ status: enabled ]  [ validation: valid ]

   ---------------------------------------------------------------------------
     Selected     #      Settings               Value/Status (level=basic)
   ---------------------------------------------------------------------------
                         ko2iblnd
                 1)        peer_credits         63
                 2)        concurrent_sends     63
                         local_lnet
                 3)        lnet_name            gni1
                 4)        ip_wildcard          10.128.*.*
                 5)      flat_routes            [ 6 sub-settings unconfigured, select
                                                and enter C to add entries ]
                 6)      fgr_routes             [ 5 sub-settings unconfigured, select
                                                and enter C to add entries ]
   ---------------------------------------------------------------------------
   ```

2. Enter 6, then C to configure fine-grained routing (`fgr_routes`).

3. Enter + and type the Sonexion server designation (`snx11029n`) to configure `cray_lnet.settings.fgr_routes`.

   ```
   cray_lnet.settings.fgr_routes.data.dest_name
   [<cr>=set '', <new value>, ?=help, @=less] $ snx11029n
   CUG
   ```

4. Enter + to configure `cray_lnet.settings.fgr_routes.data.snx11029n.routers`.

   ```
   cray_lnet.settings.fgr_routes.data.snx11029n.routers
   [<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ +
   Add routers (Ctrl-d to exit) $ c0-0c2s2n1
   Add routers (Ctrl-d to exit) $ c0-0c2s2n2
   Add routers (Ctrl-d to exit) $ c1-0c0s7n1
   Add routers (Ctrl-d to exit) $ c1-0c0s7n2
   Add routers (Ctrl-d to exit) $ c3-0c1s0n1
   Add routers (Ctrl-d to exit) $ c3-0c1s0n2
   ...
   Add routers (Ctrl-d to exit) $
   ```

5. Specify the name of the `ip2nets.conf` file for this FGR configuration. (The file must be placed in the config set on the SMW in `/var/opt/cray/imps/config/sets/`*`config_set`*`/files/roles/lnet/`.)

   ```
   cray_lnet.settings.fgr_routes.data.snx11029n.ip2nets_file
   [<cr>=set '', <new value>, ?=help, @=less] $ ip2nets.conf
   ```

6. Specify the name of the `routes.conf` file for this FGR configuration. Place `routes.conf` in the `config_set` on the SMW `/var/opt/cray/imps/config/sets/`*`config_set`*`/files/roles/lnet/`.

   ```
   cray_lnet.settings.fgr_routes.data.snx11029n.routes_file
   [<cr>=set '', <new value>, ?=help, @=less] $ routes.conf
   ```

7. Follow the guidance in the `cray_lnet` worksheet and set each configuration value for the system as needed.

8. Review the configuration settings:

   ```
   Configured Values:
   1) 'snx11029n'
       a) routers:
           c0-0c2s2n1
           c0-0c2s2n2
           c1-0c0s7n1
           c1-0c0s7n2
           ...
       b) ip2nets_file: ip2nets.conf
       c) routes_file: routes.conf
       d) ko2iblnd_peer_credits: 63
       e) ko2iblnd_concurrent_sends: 63
   ```

9. Reboot the nodes associated with these configuration changes and integrate the new configuration into the default config set (`p0`).

Configure Sonexion system with this FGR scheme.

5.2.5 Create Custom LNet Configuration for Sonexion

Prerequisites

For a new system, first complete the setup procedures described in the *Sonexion Field Installation Guide*.

About this task

Follow this procedure to create a custom LNet configuration on the Sonexion system while in the "daily mode".

Procedure

1. Log in to the primary management mode.

2. Change to `root`.

   ```
   $ sudo su -
   ```

3. Stop the Lustre file system.

   ```
   # cscli unmount -f file_system_name
   ```

4. Use the following steps to change the `o2ib` index. First, start the MySQL client and connect to the `t0db` database.

   ```
   # mysql t0db
   ```

5. Display the `mgsNID`, `nidFormat`, and `nidIndex` entries.

   ```
   mysql> select * from property where name in ('nidFormat', 'mgsNID', 'nidIndex');
   +-----+--------------------+-----------+---------------------+-----------+
   | id  | context            | name      | value               | attr_type |
   +-----+--------------------+-----------+---------------------+-----------+
   |  22 | snx11000n:beConfig | nidFormat | l%s@o2ib%d          | str       |
   | 106 | snx11000n:beConfig | nidIndex  | 3                   | int       |
   | 109 | snx11000n:beConfig | mgsNID    | lsnx11000n002@o2ib0 | str       |
   +-----+--------------------+-----------+---------------------+-----------+
          3 rows in set (0.00 sec)
   ```

6. Set the `o2ib` index by modifying the `nidIndex` entry.

   ```
   mysql> update property set value=desired_odib_index where name='nidIndex';
   ```

   For example:

   ```
   mysql> update property set value=2 where name='nidIndex';
   Query OK, 1 row affected (0.02 sec)
   Rows matched: 1  Changed: 1  Warnings: 0
   ```

7. Set the `mgsNID` entry to match the `o2ib` index.

   ```
   mysql> update property set \
   value='original_value@o2ibdesired_o2ib_index' \
   where name='mgsNID';
   ```

   For example:

   ```
   mysql> update property set value='lsnx11029n002@o2ib0' where name='mgsNID';
   Query OK, 1 row affected (0.04 sec)
   Rows matched: 1  Changed: 1  Warnings: 0
   ```

8. Verify the changes.

   ```
   mysql> select * from property where name in ('nidFormat', 'mgsNID', 'nidIndex');
   +-----+--------------------+-----------+---------------------+-----------+
   | id  | context            | name      | value               | attr_type |
   +-----+--------------------+-----------+---------------------+-----------+
   |  22 | snx11000n:beConfig | nidFormat | l%s@o2ib%d          | str       |
   | 106 | snx11000n:beConfig | nidIndex  | 2                   | int       |
   | 109 | snx11000n:beConfig | mgsNID    | lsnx11000n002@o2ib0 | str       |
   ```

```
+-----+--------------------+-----------+--------------------+-----------+
       3 rows in set (0.00 sec)
```

9. Close the MySQL session.

```
mysql> quit
```

10. Run `puppet`.

```
# /opt/xyratex/bin/beUpdatePuppet -sa
```

11. Run the `beSystemNetConfig.sh` script on the primary management node and wait for it to finish.

```
# /opt/xyratex/bin/beSystemNetConfig.sh \
-c file_location/lnet.conf -r file_location/routes.conf \
-i file_location/ip2nets.conf clustername
```

12. Verify that the customized LNet configuration has been applied.

a. List the `nids`.

```
# pdsh -g lustre lctl list_nids | sort
```

b. List the nodes and targets.

```
# cscli fs_info
```

13. Start the Lustre file system and wait for the targets to mount on all system nodes.

```
# cscli mount -f file_system_name
```

14. Modify `modprobe.conf` to support IB Aliases.

The following examples intercept the LNet module and setup IB aliases before the module is loaded. **Each example below is a single line** in `lnet.conf`.

a. Add `ib0:1` to `10.10.101` subnet.

```
## Create aliases.
## Intercepts the Lnet module and sets up the aliases before the module is loaded install lnet

/sbin/ip -o -4 a show ib0 | /usr/bin/awk '/inet/{s=$4;sub("10\\.10\\.100\
\.","10.10.101.",s);print "/sbin/ip address add dev ib0 label ib0:1",s}' | /bin/sh ;/sbin/
modprobe --ignore-install lnet $CMDLINE_OPTS
```

b. Remove all `ib0:1` aliases when LNet module is unloaded.

```
## Remove all ib0:1 aliases when lnet module is unloaded

remove lnet /sbin/modprobe -r --ignore-remove lnet && /sbin/ip -o -4 a show label ib0:1 | awk
'{print "/sbin/ip address del dev ib0 label ib0:1",$4}' | /bin/sh
```

c. Configure a metadata server on a 3rd IPv4 subnet.

```
install lnet if nodeattr mds; then /sbin/ip -o -4 a show ib0 | /usr/bin/awk '/inet/{s=$4;sub("10\
\.10\\.100\\.","10.10.102.",s);print "/sbin/ip address add dev ib0 label ib0:1",s}' | /bin/sh ;
else /sbin/ip -o -4 a show ib0 | /usr/bin/awk '/inet/{s=$4;sub("10\\.10\\.100\
\.","10.10.101.",s);print "/sbin/ip address add dev ib0 label ib0:1",s}' | /bin/sh ; fi ; /sbin/
modprobe --ignore-install lnet $CMDLINE_OPTS
```

5.3 Manually Control LNet Routers

Procedure

If `/etc/init.d/lnet` is not provided, send the following commands to each LNet router node to control them manually:

- For startup:

```
modprobe lnet
lctl net up
```

- For shutdown:

```
lctl net down
lustre_rmmod
```

6 Lustre Failover on Cray Systems

To support Lustre failover, each LUN (logical unit) must be visible to two Lustre service nodes. For more information about setting up the hardware storage configuration for Lustre failover, contact a Cray service representative.

Failover is generally defined as a service that switches to a standby or secondary server when the primary system fails or the service is temporarily shutdown for maintenance. Lustre can be configured to automatically failover (see *Lustre Automatic Failover* on page 85), or failover can be set up to be done manually as described in *Configure Manual Lustre Failover* on page 81.

6.1 Configuration Types for Failover

The Lustre failover configuration requires two nodes (a failover pair) that must be connected to a shared storage device. Nodes can be configured in two ways, active/active or active/passive. An active node actively serves data and a passive node idly stands by to take over in the event of a failure. (Automatic failover is only supported on direct-attached Lustre file systems with combined MGS/MDS configurations.)

Active/ Passive In this configuration, only one node is actively serving data all the time. The other node takes over in the case of failure. Failover for the Lustre metadata server (MDS) is configured this way on a Cray system. For example, the active node is configured with the primary MDS, and the service is started when the node is up. The passive node is configured with the backup or secondary MDS service, but it is not started when the node is up so that the node is idling. If an active MDS fails, the secondary service on the passive node is started and takes over the service.

Active/ Active In this configuration, both nodes actively serve data all the time on different storage devices. In the case of a failure, one node takes over for the other, serving both storage devices. Failover for the Lustre object storage target (OST) is configured this way on a Cray system. For example, one active node is configured with primary OST services `ost1` and `ost3` and secondary services `ost2` and `ost4`. The other active node is configured with primary OST services `ost2` and `ost4` and secondary services `ost1` and `ost3`. In this case, both nodes are only serving their primary targets and failover services are not active. If an active node fails, the secondary services on the other active node are started and begin serving the affected targets.

CAUTION: To prevent severe data corruption, the physical storage must be protected from simultaneous access by two active nodes across the same partition. For proper protection, the failed server must be completely halted or disconnected from the shared storage device.

6.2 Configure Manual Lustre Failover

Manual failover may be attractive to administrators who require full control over Lustre servers. Manual failover is the only option for failing over file systems with separate MGS/MDS configurations.

Configure Lustre for manual failover using Cray Lustre control utilities to interface with the standard Lustre configuration system. For additional information about configuring Lustre for Cray systems, see *XC™ Series System Software Initial Installation and Configuration Guide CLE 6001*.

6.2.1 Configure DAL Failover for CLE 6.0

Prerequisites

This task assumes the direct-attached Lustre (DAL) file is configured without failover and that the system is currently running with the DAL file system.

About this task

Targets are configured for failover by specifying the `fo_node` component of the target definition in the `fs_defs` file. If new servers are added, then the `nid_map` in the `fs_defs` needs to be updated. For manual failover, `auto_fo: no` is set in the `fs_defs`. If necessary, the `cray_lustre_client` service is updated to specify an additional MGS LNet `nid`.

Procedure

1. Add `fo_node` definitions to each target in the `lustre_control` `lustre/.lctrl/fs_defs` file for the config set.

   `/var/opt/cray/imps/config/sets/p0/lustre/.lctrl/dal.fs_defs.2015052.1440768293`

2. Copy the `dal.fs_defs` file to a work area on the SMW.

3. Edit the file so that each Lustre target, management target (MGT), metadata target (MDT), and object storage target (OST) has a corresponding `fo_node` defined.

```
## MGT
 ## Management Target
 mgt: node=nid00009
+     fo_node=nid00010
      dev=/dev/disk/by-id/scsi-360080e50003ea4c20000065d5257da5d

 ## MDT
 ## MetaData Target(s)
 mdt: node=nid00009
+     fo_node=nid00010
      dev=/dev/disk/by-id/scsi-360080e50003ea4c20000065f52580c00
      index=0

 ## OST
 ## Object Storage Target(s)
 ost: node=nid00010
+     fo_node=nid00009
      dev=/dev/disk/by-id/scsi-360080e50003ea5ba000006605257db2f
      index=0
 ost: node=nid00010
+     fo_node=nid00009
      dev=/dev/disk/by-id/scsi-360080e50003ea5ba000006625257db5c
      index=1
 ost: node=nid00010
+     fo_node=nid00009
```

```
      dev=/dev/disk/by-id/scsi-360080e50003ea5ba000006645257db84
      index=2
```

4. Set `auto_fo` to `yes`.

   ```
   +
   +auto_fo: yes
   ```

5. Save the `fs_defs` file.

6. Install the `fs_defs` into the config set and overwrite the existing file system definition using `lustre_control` install on the SMW.

   a. Install the `fs_defs` with `lustre_control`.

   ```
   smw# lustre_control install -c p0 dal.fs_defs.20150828.1440768293
   Performing 'install' from smw at Wed Oct 21 16:28:35 CDT 2015

   Parsing file system definitions file: dal.fs_defs.20150828.1440768293
   Parsed file system definitions file: dal.fs_defs.20150828.1440768293
   A file system named "dal" has already been installed.
   ```

   b. Answer the prompt.

   ```
   Would you like to overwrite the existing file system definitions with those in dal.fs_defs.
   20150828.1440768293? (y|n|q)
   y
   Operating on file system - "dal"
   The 'dal' file system definitions were successfully removed.
   ```

   c. Update failover tables.

   ```
   Failover tables need to be updated. Please execute the following command from
   the boot node:
   lustre_control update_db
   The 'dal' file system definitions were successfully installed!
   ```

   d. Repeat previous step.

   ```
   Failover tables need to be updated. Please execute the following command from
   the boot node:
   lustre_control update_db
   ```

7. Update the SDB failover tables by running the following command on the boot node.

   ```
   boot# lustre_control update_db
   ```

8. Stop the file system.

   a. Unmount all clients before stopping the file system.

   ```
   boot# xtxqtcmd ALL_COMPUTE "umount $DAL_MOUNT_POINT"
   ```

   b. Unmount the file system on the login nodes and any other nodes where it is mounted.

   ```
   boot# lustre_control umount_clients -w $NODE_LIST -f $FS_NAME
   ```

   c. Stop the file system.

   ```
   boot# lustre_control stop -f $FS_NAME
   ```

9. Reboot the DAL nodes.

   (This step ensures DAL nodes have the most recent version of the config set containing the new version of the `fs_defs` file with failover configured. Do not reboot the whole system. The default auto boot file does not run the correct `lustre_control` commands needed to initially set up failover.)

10. Perform a `write_conf` before restarting the file system.

    ```
    boot# lustre_control write_conf -f $FS_NAME
    ```

11. Restart the file system.

    ```
    boot# lustre_control start -p -f $FS_NAME
    ```

12. Mount the file system on clients.

    a. Mount compute nodes.

    ```
    boot# xtxqtcmd ALL_COMPUTE "mount $DAL_MOUNT_POINT"
    ```

    b. Mount service nodes.

    ```
    boot# lustre_control mount_clients -f $FS_NAME -w $NODE_LIST
    ```

6.2.2 Perform Lustre Manual Failover

About this task

If the system is set up for failover and a node fails or an object storage target (OST) is not functional, perform the following steps to initiate Lustre manual failover.

Procedure

1. Halt the failing node with the `xtcli` command on the SMW.

   ```
   smw# xtcli halt -f c0-0c0s6n2
   ```

   The `-f` option is required to make sure that an alert flag does not prohibit the halt from completing.

   CAUTION: Prior to starting secondary OST services, ensure that the primary node is halted. Simultaneous access from multiple server nodes can cause severe data corruption.

2. Check the status of the OSS node to make sure the `halt` command executed properly.

   ```
   smw# xtcli status c0-0c0s6n2
   Network topology: class 0
   Network type: Gemini
             Nodeid: Service  Core Arch|  Comp state      [Flags]
   --------------------------------------------------------------------------
          c0-0c0s6n2: service  IB06   OP|        halt      [noflags|]
   --------------------------------------------------------------------------
   ```

3. Start the failover process with the `lustre_control failover` command.

   Here, `nid00018` is the failing primary server that was halted in.

   ```
   boot# lustre_control failover -w nid00018 -f lus0
   ```

   The recovery process may take several minutes, depending on the number of clients. Attempting warm boots during Lustre recovery is not advised as it will break recovery for the remaining clients. Recovery will begin automatically as clients reconnect (unless `abort_recovery` is specified via `lctl`).

   To monitor the status of recovery, see *Monitor Recovery Status* on page 85

4. Run, after recovery, the `df` or `lfs df` command on a login node.

Applications that use Lustre on login nodes should be able to continue to check if all services are working properly.

If there are a large number of clients doing Lustre I/O at the time that the failure occurs, the recovery time may become very long. But it will not exceed the value specified by the `recovery_time_hard` parameter in the *`fs_name`*`.fs_defs` file.

6.2.3 Monitor Recovery Status

The `lustre_control status` command may be used to monitor an OST that is in the recovery process. Upon completion, the status changes from `RECOVERING` to `COMPLETE`.

OST Recovery After Failover

List status of OSTs.

```
boot# lustre_control status -t ost -a
Performing 'status' from boot at Tue May  8 08:09:28 CDT 2012
File system: fs_name
Device             Host     Mount        OST Active  Recovery Status
fs_name-OST0000    nid00026 Unmounted    N/A         N/A
fs_name-OST0000*   nid00018 Mounted      N/A         RECOVERING
[...]
Note: '*' indicates a device on a backup server
ade#
```

Rerun command to verify recovery is complete.

```
boot# lustre_control status -t ost -a
Performing 'status' from boot at Tue May  8 08:22:38 CDT 2012
File system: fs_name
Device             Host     Mount        OST Active  Recovery Status
fs_name-OST0000    nid00026 Unmounted    N/A         N/A
fs_name-OST0000*   nid00018 Mounted      N/A         COMPLETE
Note: '*' indicates a device on a backup server
```

The recovery status is recorded in the following `/proc` entries.

For OSS

```
/proc/fs/lustre/obdfilter/lus0-OST0000/recovery_status
/proc/fs/lustre/obdfilter/lus0-OST0002/recovery_status
```

For MDS

```
/proc/fs/lustre/mds/lus0-MDT0000/recovery_status
```

6.3 Lustre Automatic Failover

This section describes the framework and utilities that enable Lustre services to failover automatically in the event that the primary Lustre services fail. Lustre automatic failover is only applicable to direct-attached Lustre (DAL) file systems.

The automatic Lustre failover framework includes the `xt-lustre-proxy` process, the service database, a set of database utilities and the `lustre_control` command. The Lustre configuration and failover states are kept in the service database (SDB). Lustre database utilities and the `xt-lustre-proxy` process are used in conjunction with `lustre_control` for Lustre startup/shutdown and for failover management. The `xt-lustre-proxy` process is responsible for automatic Lustre failover in the event of a Lustre service failure.

To enable automatic failover for a Lustre file system, set `auto_fo: yes` in the file system definition file. If automatic failover is enabled, `lustre_control` starts an `xt-lustre-proxy` process on the MDS and OSSs. It then monitors the health of MDS and OSS services through the hardware supervisory system (HSS) system. If there is a node-failure or service-failure event, HSS notifies the `xt-lustre-proxy` process on the secondary node to start up the backup services.

The primary and secondary configuration is specified in the *`fs_name`*`.fs_defs`. The failover configuration is stored in the SDB for use by `xt-lustre-proxy`. To avoid both primary and secondary services running at the same time, the `xt-lustre-proxy` service on the secondary node issues a `node reset` command to shut down the primary node before starting the secondary services. The proxy also marks the primary node as `dead` in the SDB so that, if the primary node is rebooted while the secondary system is still running, `xt-lustre-proxy` will not start on the primary node.

When Lustre automatic failover is configured, the `lustre_control` utility starts and stops the `xt-lustre-proxy` daemon each time Lustre services are started and stopped. `lustre_control` uses the configuration information in the *`fs_name`*`.fs_defs` file to start the `xt-lustre-proxy` daemon with options appropriate for the configuration. Typically, `xt-lustre-proxy` is not used directly by system administrators.

Services can be disabled to prevent some MDS or OST services from participating in automatic failover. See the `xtlusfoadmin(8) man` page and *Use the xtlusfoadmin Command* on page 88 for more information on enabling and disabling Lustre services.

The status of Lustre automatic failover is recorded in `syslog` messages.

6.3.1 Lustre Automatic Failover Database Tables

Three service database (SDB) tables are used by the Lustre failover framework to determine failover processes. The `lustre_control install` command creates and populates the `filesystem`, `lustre_service`, and `lustre_failover` database tables as described in the following sections. The `lustre_control remove` command updates these tables as necessary. After the `lustre_control install` and the `lustre_control remove` operations are performed, `lustre_control update_db` must be run on the `boot` node to modify the three SDB tables.

6.3.1.1 The `filesystem` Database Table

The `filesystem` table stores information about file systems. The fields in the `filesystem` table are shown in *filesystem SDB Table Fields*. For more information, see the `xtfilesys2db(8)` and `xtdb2filesys(8) man` pages.

Table 8. `filesystem` SDB Table Fields

Database Table Field	Description
`fs_fsidx`	File system index. Each file system is given a unique index number based on the time of day.
`fs_name`	Character string of the internal file system name. Should match the value of `fs_name` as defined in the *`fs_name`*`.fs_defs` file.
`fs_type`	File system type. Valid values are `fs_lustre` or `fs_other`. The `fs_other` value is currently not used.
`fs_active`	File system status snapshot. Valid values are `fs_active` or `fs_inactive`.

Database Table Field	Description
`fs_time`	Timestamp when any field gets updated. Format is `'yyyy-mm-dd hh:mm:ss'`.
`fs_conf`	Reserved for future use. Specify as a null value using `''`.

6.3.1.2 The `lustre_service` Database Table

The `lustre_service` table stores the information about Lustre services. The fields in the `lustre_service` table are shown in *lustre_service SDB Table Fields*. For more information, see the `xtlustreserv2db(8)` and `xtdb2lustreserv(8)` man pages.

Table 9. `lustre_service` SDB Table Fields

Database Table Field	Description
`lst_srvnam`	Name of the Lustre metadata server (MDS) or object storage target (OST) services.
`lst_srvidx`	Service index. For an MDS, use a value of `0`. For OSTs, use the index of the OSTs as defined in the *`fs_name`*`.fs_defs` file. Format is an integer number.
`lst_fsidx`	File system index. Format is a character string.
`lst_prnid`	Node ID (NID) of the primary node for this service. Format is an integer value.
`lst_prdev`	Primary device name, such as `/dev/disk/by-id/`*`IDa`*, for metadata target (MDT) or OST. Format is a character string.
`lst_bknid`	NID of the backup or secondary node for this service. Format is an integer value.
`lst_bkdev`	Backup or secondary device name, such as `/dev/disk/by-id/`*`IDa`*, for MDT or OST. Format is a character string.
`lst_ltype`	Lustre service type. Valid values are `lus_type_mds` or `lus_type_ost`.
`lst_failover`	Enables failover. Valid values are `lus_fo_enable` to enable the failover process and `lus_fo_disable` to disable the failover process.
`lst_time`	Timestamp when any field gets updated.

6.3.1.3 The `lustre_failover` Database Table

The `lustre_failover` table maintains the Lustre failover states. The fields in the `lustre_failover` table are shown in *lustre_failover SDB Table Fields*. For more information, see the `xtlustrefailover2db(8)`, `xtdb2lustrefailover(8)` and `lustre_control(5)` man pages.

Table 10. `lustre_failover` SDB Table Fields

Database Table Field	Description
`lft_prnid`	NID for primary node.
`lft_bknid`	NID for backup or secondary node. A value of `-1` (displays 4294967295) indicates there is no backup node for the primary node.
`lft_state`	Current state for the primary node. Valid states are `lus_state_down`, `lus_state_up` or `lus_state_dead`. The `lus_state_dead` state indicates that Lustre services on the node have failed and the services are now running on the secondary node. The services on this node should not be started until the state is changed to `lus_state_down` by a system administrator.
`lft_init_state`	Initial primary node state at the system boot. The state here will be copied to `lft_state` during system boot. Valid states are `lus_state_down` or `lus_state_dead`. For normal operations, set the state to `lus_state_down`. If Lustre services on this node should not be brought up, set the state to `lus_state_dead`.
`lft_time`	Timestamp when any field gets updated.

6.3.2 Back Up SDB Table Content

The following set of utilities can be used to dump the database entries to a data file.

CAUTION: By default, these utilities will create database-formatted files named `lustre_failover`, `lustre_serv`, and `filesys` in the current working directory. Use the `-f` option to override default names.

Table 11. Lustre Automatic Failover SDB Table Dump Utilities

Command	Description
`xtdb2lustrefailover`	Dumps the `lustre_failover` table in the SDB to the `lustre_failover` data file.
`xtdb2lustreserv`	Dumps the `lustre_service` table in the SDB to the `lustre_serv` data file.
`xtdb2filesys`	Dumps the `filesystem` table in the SDB to the `filesys` data file.

6.3.3 Use the `xtlusfoadmin` Command

The `xtlusfoadmin` command can be used to modify or display fields of a given automatic Lustre failover database table. When it is used to make changes to database fields, failover operation is impacted accordingly. For example, `xtlusfoadmin` is used to set file systems active or inactive or to enable or disable the Lustre failover process for Lustre services. For more information, see the `xtlusfoadmin(8)` man page.

Use the query option (`-q` | `--query`) of the `xtlusfoadmin` command to display the fields of a database table. For example:

1. Display `lustre_failover` table.

   ```
   # xtlusfoadmin -q o
   ```

2. Display `lustre_service` table.

   ```
   # xtlusfoadmin -q s
   ```

3. Display `filesystem` table.

   ```
   # xtlusfoadmin -q f
   ```

4. Display all three tables.

   ```
   # xtlusfoadmin -q a
   ```

Some invocations of `xtlusfoadmin` require the variables *`fs_index`* and *`service_name`*. The available values for these variables can be found by invoking the `xtdb2lustreserv -f -` command, which prints the `lustre_service` table to `stdout`.

6.3.3.1 Identify Lustre File System *`fs_index`* and *`service_name`* Values

Invoke the `xtdb2lustreserv -f -`command to display the `lustre_service` table contents. In this example (which has been truncated to save space) the *`fs_index`* for this file system is `79255228`, and the *`service_name`* for the device listed is *`fs_name`*`-MDT0000`.

```
boot# xtdb2lustreserv -f -
Connected
#
# This file lists tables currently in the system database,
#
# Each line contains a record of comma-delineated pairs of the form field1=val1, field2=val2, etc.
#
# Note: this file is automatically generated from the system database.
#
lst_srvnam='fs_name-MDT0000',lst_srvidx=0,lst_fsidx=79255228...
```

Use the following commands to modify fields in the database and impact failover operation.

Enable/Disable Failover Process for Whole File System

To either enable or disable the failover process for the whole file system, use the activate (`-a` | `--activate_fs`) or deactivate (`-d` | `--deactivate_fs`) options with the `xtlusfoadmin` command. These options set the value of the `fs_active` field in the `filesystem` table to either `fs_active` or `fs_inactive`.

- Set the filesystem active:

  ```
  boot# xtlusfoadmin -a fs_index
  ```

- Deactivate the filesystem:

  ```
  boot# xtlusfoadmin -d fs_index
  ```

This needs to be set before the `xt-lustre-proxy` process starts. If set while the proxy is running, `xt-lustre-proxy` needs to be restarted in order to pick up the change. Always shutdown `xt-lustre-proxy` gracefully before restart. A failover can be triggered if there is not a graceful shutdown. A graceful shutdown is a successful completion of the `lustre_control stop` command.

Enable or Disable Failover Process for Lustre Service on Specific Node

To enable or disable the failover process for a Lustre service on a specific node, use the `--enable_fo_by_nid` (`-e`) or `--disable_fo_by_nid` (`-f`) options.

- Enable failover for the node:

```
boot# xtlusfoadmin -e nid
```

- Disable failover for the node:

```
boot# xtlusfoadmin -f nid
```

Enable or Disable Failover Process for a Lustre Service by Name

To enable or disable the failover process for a Lustre service by name, use the enable (`-j` | `--enable_fo`) or disable (`-k` | `--disable_fo`) options. These options set the value of the `lst_failover` field in the `lustre_service` table to either `lus_fo_enable` or `lus_fo_disable`.

- Enable failover services:

```
boot# xtlusfoadmin -j fs_index service_name
```

- Disable failover services:

```
boot# xtlusfoadmin -k fs_index service_name
```

Change Initial State of Service Node

To change the initial state of a service node, use the `--init_state` (`-i`) option. This option sets the value of the `lft_init_state` field in the `lustre_failover` table to either `lus_state_down` or `lus_state_dead`.

- Set the initial node state to down:

```
boot# xtlusfoadmin -i nid n
```

- Set the initial node state to dead:

```
boot# xtlusfoadmin -i nid d
```

By setting a node as `dead`, Lustre services should not be started on that node after a reboot.

Reinitialize Current State of Service Node

To reinitialize the current state of a service node, use the `--set_state` (`-s`) option. This option would most commonly be used during failback, following a failover. Use the `set_state` option to change the state of a primary node from `dead` to `down` in order to failback to the primary node. This option sets the value of the `lft_state` field in the `lustre_failover` table to either `lus_state_down` or `lus_state_dead`.

- Set the current node state to down:

```
boot# xtlusfoadmin -s nid n
```

- Set the current node state to dead:

```
boot# xtlusfoadmin -s nid d
```

6.3.4 System Startup and Shutdown when Using Automatic Lustre Failover

Use the `lustre_control` command to start Lustre services. The `lustre_control` command starts both Lustre services and launches `xt-lustre-proxy`.

The following failover database information will impact startup operations as indicated.

Service Failover Enable/Disable In the event that there is a failure, the failover-disabled service does not trigger a failover process. If any services for the node have failover enabled, the failure of the service triggers the failover process. To prevent a failover process from occurring for an MDS or OSS, failover must be disabled for all the services on that node. Use the `xtlusfoadmin` command to disable failover on a service. For example, to disable failover for an entire file system, run this command:

```
xtlusfoadmin --disable_fo fs_index
```

To disable failover for all services on a node, type the following command:

```
xtlusfoadmin --disable_fo_by_nid nid
```

Initial State At system startup, the current state (`lft_state`) of each primary MDS and OSS node is changed to the initial state (`lft_init_state`), which is usually `lus_state_down`.

Current State Following an Automatic Failover When failing back the primary services from the secondary node after automatic failover, the primary node state will be `lus_state_dead` and will require re-initialization. The `xt-lustre-proxy` process will need the node to be in the `lus_state_down` state to start. Use the `xtlusfoadmin` command to change the current state of a node to `lus_state_down`. For example:

```
xtlusfoadmin --set_state nid n
```

6.3.4.1 Lustre Startup Procedures for Automatic Failover

Procedure

1. Log on to the boot node as `root`.
2. Start Lustre services and `xt-lustre-proxy`.

   Type the following commands for each Lustre file system that has been configured.

   a. Start filesystem.

   ```
   boot# lustre_control start -f fs_name
   ```

   b. Mount filesystem on clients.

   ```
   boot# lustre_control mount_clients -f fs_name
   ```

3. Optional: Mount the compute node Lustre clients at this time.

   ```
   boot# lustre_control mount_clients -c -f fs_name
   ```

6.3.4.2 Lustre Shutdown Procedures for Automatic Failover

About this task

CAUTION: The `lustre_control shutdown` command gracefully shuts down the `xt-lustre-proxy` process. Issuing `SIGTERM` will also work for a graceful shutdown. Any other method of termination, such as sending a `SIGKILL` signal, triggers the failover process and results in a failure event delivered to the secondary node. The secondary node then issues a `node reset` command to shut down the primary node and starts Lustre services.

Procedure

1. Log on to the boot node.
2. Shut down the Lustre file system.

```
boot# lustre_control shutdown -f fs_name
```

6.3.5 Configure Lustre Failover for Multiple File Systems

In order to support automatic Lustre failover for multiple file systems, the following limitation must be worked around—the `lustre_control stop` option terminates the `xt-lustre-proxy` process on servers that the specified file system uses. This includes servers that are shared with other file systems.

When shutting down only a single file system, `xt-lustre-proxy` must be restarted on the shared servers for the other file systems that are still active. Follow *Shut Down a Single File System in a Multiple File System Configuration* on page 92 to properly shut down a single file system in a multiple file system environment.

6.3.5.1 Shut Down a Single File System in a Multiple File System Configuration

About this task

This procedure is not required to shut down all Lustre file systems. It is only needed to shut down a single file system while leaving other file systems active.

After stopping Lustre on one file system, restart `xt-lustre-proxy` on the shared Lustre servers. Lustre services are still active for the file systems not stopped. The `xt-lustre-proxy` daemon on the shared servers, however, is terminated when a file system is shut down. In this example, `myfs2` is shut down.

Procedure

1. Unmount Lustre from the compute node clients.

```
boot# lustre_control umount_clients -c -f myfs2
```

2. Unmount Lustre from the service node clients.

```
boot# lustre_control umount_clients -f myfs2
```

3. Stop Lustre services.

```
boot# lustre_control stop -f myfs2
```

4. Restart `xt-lustre-proxy` on the shared Lustre servers by using `lustre_control`.

   The remaining active Lustre services are not affected when `xt-lustre-proxy` is started. The `lustre_control start` command is written to first start any MGS services, then any OST services, and finally any MDT services. If there are errors at any step (by trying to start an MGS that is already running), the script will exit before attempting to mount any subsequent targets. In order to successfully restart `xt-lustre-proxy`, choose the command(s) to execute next based on the role(s) of the shared servers.

   If only OSS servers are shared, execute this command.

   ```
   boot# lustre_control start -w oss1_node_id,oss2_node_id,... -f myfs1
   ```

   If only a combined MGS/MDT server is shared, execute this command.

   ```
   boot# lustre_control start -w mgs_node_id -f myfs1
   ```

   If a combined MGS/MDS server and OSS servers are shared, execute these commands.

   1. Start Lustre on MGS/MDS node.

      ```
      boot# lustre_control start -w mgs_node_id -f myfs1
      ```

   2. Start Lustre on all the OSS.

      ```
      boot# lustre_control start -w oss1_node_id,oss2_node_id,... -f myfs1
      ```

6.4 Back Up and Restore Lustre Failover Tables

About this task

To minimize the potential impact of an event that creates data corruption in the service database (SDB), Cray recommends creating a manual backup of the Lustre tables that can be restored after a reinitialization of the SDB.

Procedure

Manually Back Up Lustre Failover Tables

1. Log on to the boot node as `root`.

2. Back up the `lustre_service` table.

   ```
   boot# mysqldump -h sdb XTAdmin lustre_service > /var/tmp/lustre_service.sql
   ```

3. Back up the `lustre_failover` table.

   ```
   boot# mysqldump -h sdb XTAdmin lustre_failover > /var/tmp/lustre_failover.sql
   ```

Manually Restore Lustre Failover Tables

4. Log on to the boot node as `root`.

5. After the SDB is recreated, restore the `lustre_service` table.

   ```
   boot# mysqldump -h sdb XTAdmin < /var/tmp/lustre_service.sql
   ```

6. Restore the lustre_failover table.

```
boot# mysqldump -h sdb XTAdmin < /var/tmp/lustre_failover.sql
```

6.5 Perform Lustre Failback on CLE Systems

About this task

In this procedure, nid00018 (ost0 - /dev/disk/by-id/*IDa*, ost2 - /dev/disk/by-id/*IDc*) and nid00026 (ost1 - /dev/disk/by-id/*IDb*, ost3 - /dev/disk/by-id/*IDd*) are failover pairs. Here, nid00018 failed and nid00026 is serving both the primary and backup OSTs. After these steps are completed, ost0 and ost2 should failback to nid00018.

Procedure

1. Reset the primary node state in the SDB for an automatic failover. (There is no need to do this in manual failover since the SDB was not involved. If the system is not configured for automatic failover, skip ahead to step 4.)

During a failover, the failed node was set to the lus_state_dead state in the lustre_failover table. This prevents xt-lustre-proxy from executing upon reboot of the failed node. The failed node must be reset to the initial lus_state_down state. The following displays the current and initial states for the primary node. In this example, nid00018 has failed and nid00026 now provides services for its targets.

```
nid00026# xtlusfoadmin
sdb lustre_failover table
PRNID      BKNID      STATE             INIT_STATE       TIME
12         134         lus_state_up      lus_state_down   2008-01-16 14:32:46
18         26         lus_state_dead    lus_state_down   2008-01-16 14:37:17
26         18         lus_state_up      lus_state_down   2008-01-16 14:31:32
```

2. Reset the state using the following command.

```
nid00026# xtlusfoadmin -s 18 n
lft_state in lustre_failover table was updated to lus_state_down for nid 18
```

Here the command option -s 18 n sets the state for the node with nid 18 to n (lus_state_down). For more information, see the xtlusfoadmin(8) man page.

3. Run xtlusfoadmin again to verify that the state has been changed.

```
nid00026# xtlusfoadmin
sdb lustre_failover table
PRNID      BKNID      STATE             INIT_STATE       TIME
12         134         lus_state_up      lus_state_down   2008-01-16 14:32:46
18         26         lus_state_down    lus_state_down   2008-01-16 14:59:39
26         18         lus_state_up      lus_state_down   2008-01-16 14:31:32
```

4. Unmount the secondary OSTs from the remaining live OSS in the failover pair.

In this case, ost0 and ost2 are the secondary OSTs and nid00026 is the remaining live OSS.

a. Unmount the first target.

```
nid00026# umount /mnt/lustre/fs_name/ost0
```

b. Unmount additional targets.

```
    nid00026# umount /mnt/lustre/fs_name/ost2
```

It is acceptable if the node is unable to unload some Lustre modules. This is because they are still in use by the primary OSTs belonging to `nid00026`. In order to proceed, the `umount` commands have to finish successfully.

5. Verify that `ost0` and `ost2` are no longer showing up in the device list. (When this command is entered, the following message indicates the OSTs used.)

```
nid00026# lctl dl
  0 UP mgc MGC12@gni 59f5af70-8926-62b7-3c3e-180ef1a6d48e 5
  1 UP ost OSS OSS_uuid 3
  2 UP obdfilter mds1-OST0001 mds1-OST0001_UUID 9
  5 UP obdfilter mds1-OST0003 mds1-OST0003_UUID 3
```

6. Boot the failed node.

7. Optional: Start recovering Lustre using `lustre_control` from the boot node with the following command.

```
boot# lustre_control failback -w nid00018 -f fs_name
```

8. Check the `recovery_status` to see if it has completed.

```
boot# lustre_control status -w nid00018 -f fs_name
```

7 LMT Configuration for DAL

CAUTION: As stated in the "Migration Caveats" section of the introduction, this migration process does not include a tested procedure for preserving a DAL LMT database during migration. If this site wishes to preserve an existing LMT database, do not use this procedure as it will result in loss of existing data.

The Lustre® monitoring tool (LMT) for direct-attached Lustre (DAL) on Cray Linux environment (CLE 6.0) requires some manual configuration during the software installation process.

Configure Storage for the LMT Database	At least 40GB of storage space must be made available to the MGS node. See *LMT Disk Usage* on page 100.
Configure the LMT MySQL Database	The IMPS configuration does not set up this database, so this must be configured manually for CLE 6.0 UP01 and later releases. See *Configure LMT MySQL Database for DAL* on page 96.
Configure the LMT GUI (Optional)	See *Configure the LMT GUI* on page 98.

Use the configurator to configure the LMT for DAL on CLE 6.0. Guidance is provided for each LMT configuration setting in the `cfgset` utility.

The `cray_lmt` configurator template configures LMT settings for specific nodes when they are booted. The default system configuration value for the LMT service is disabled (`false`). Log in to the SMW as `root` and use the `cfgset` command to modify the `cray_lmt` configuration settings to configure LMT.

```
smw# cfgset update -s cray_lmt -m interactive CONFIG_SET
```

7.1 Configure LMT MySQL Database for DAL

Prerequisites

A MySQL server instance must be configured on the management server (MGS) node. All commands described below should be executed on the MGS for the direct-attached Lustre (DAL) file system.

About this task

A MySQL server instance on the management server (MGS) node stores real-time and historical Lustre monitoring tool (LMT) data. The configurator does not handle the initial setup of the LMT MySQL users and database. It must, therefore, be done manually. All commands described below should be executed on the MGS for the DAL file system.

Procedure

1. Log on to the MGS as `root`.

   (Where *`nidMGS`* is the node ID (NID) of the MGS node.)

   ```
   boot# ssh nidMGS
   ```

2. Start the MySQL server daemon (if not already running).

   ```
   mgs# /sbin/service mysqld start
   ```

3. Run the `mysql_secure_installation` script to improve MySQL server instance security.

   This sets the password for the `root` MySQL user, disallows remote `root` access to the database, removes anonymous users, removes the test database, and reloads privileges. If this is the first time configuring LMT, create a symlink before running `mysql_secure_installation` to ensure that MySQL uses the correct socket.

   a. Create a symbolic link.

   ```
   mgs# ln -s /var/run/mysql/mysql.sock /var/lib/mysql/mysql.sock
   ```

   b. Run `mysql_secure_installation` utility.

   ```
   mgs# mysql_secure_installation
   ```

   c. Respond to script prompts.

   Prompts and recommended responses generated by the script.

   ```
   Enter current password for root (enter for none): <Enter>

   Set root password? [Y/n] Y
   New password: Enter a secure password
   Re-enter new password: Enter the secure password again

   Remove anonymous users? [Y/n] Y

   Disallow root login remotely? [Y/n] Y

   Remove test database and access to it? [Y/n] Y

   Reload privilege tables now? [Y/n] Y
   ```

4. Ensure `root` only access to the LMT user configuration file, `/usr/share/lmt/mkusers.sql`.

   ```
   mgs# chmod 600 /usr/share/lmt/mkusers.sql
   ```

5. Edit the LMT user configuration file `/usr/share/lmt/mkusers.sql`.

   This file is not used at run time by LMT or MySQL processes. This script creates the MySQL users on the persistent storage configured for the MySQL databases. After it is run through MySQL, it is no longer needed.

   This file contains MySQL statements that create users named `lwatchclient` and `lwatchadmin`. It gives them privileges only on databases that start with `filesystem_`. Cray recommends making the following changes to `mkusers.sql`.

Edit the GRANT Statement	Edit the GRANT statements to grant privileges on only `filesystem_`*`fsname`*`.*` where *`fsname`* is the name of the file system. This will only grant permissions on the database for the file system being monitored.
Edit the Password	Edit the password for `lwatchadmin` by changing `mypass` to the desired password. Also add a password for the `lwatchclient` user.

```
CREATE USER 'lwatchclient'@'localhost' IDENTIFIED BY 'foo';
GRANT SELECT ON filesystem_scratch.* TO 'lwatchclient'@'localhost';

CREATE USER 'lwatchadmin'@'localhost' IDENTIFIED BY 'bar';
GRANT SELECT,INSERT,DELETE  ON filesystem_scratch.* TO 'lwatchadmin'@'localhost';
GRANT CREATE,DROP           ON filesystem_scratch.* TO 'lwatchadmin'@'localhost';

FLUSH PRIVILEGES;
```

6. Save the changes and execute the following command. (This prompts for the MySQL `root` user password, which was set when `mysql_secure_installation` was executed.)

```
mgs# mysql -u root -p < /usr/share/lmt/mkusers.sql
```

7. Create the database for the file system to be monitored.

   (Where *`fsname`* is the name of the DAL file system.)

```
mgs# lmtinit -a fsname
```

   LMT data will be inserted into the LMT MySQL database the next time the Cerebro service is restarted on the MGS.

8. Restart Cerebro.

```
mgs# service cerebrod restart
```

9. Verify that LMT is adding data to the MySQL database.

   a. Initiate the LMT shell.

```
mgs# lmtsh -f fsname
```

   b. List tables.

```
fsname> t
```

   c. List tables again after several seconds to verify that `Row Count` is increasing.

7.2 Configure the LMT GUI

About this task

The Lustre monitoring tool (LMT) graphical user interface (GUI) package is installed on login nodes. It contains a GUI called `lwatch` and a command-line tool for viewing live data called `lstat`. The configuration file `~/.lmtrc` must be set up prior to using either tool.

Procedure

1. Login to the MGS node as `root`.

2. Edit the sample configuration file `/usr/share/doc/packages/lmt-gui/sample.lmtrc` to reflect the site specific LMT configuration—where *`db_name`* is set to the name of the MySQL database used by LMT, that is, filesystem_*`fsname`*.

```
# LMT Configuration File - place in $HOME/.lmtrc
```

```
filesys.1.name=<insert_fsname_here>
filesys.1.mountname=<insert_/path/to/mountpoint_here>
filesys.1.dbhost=<insert_db_host_ip_here>
filesys.1.dbport=<insert_db_port_here>
filesys.1.dbuser=<insert_db_client_username_here>
# Leave dbauth blank if the given client has no password
filesys.1.dbauth=<insert_db_client_password_here>
filesys.1.dbname=<insert_db_name_here>
```

3. Save the updated `.lmtrc` as `~/.lmtrc`.

   Here is an example for configuring access to the LMT database for the file system named *`scratch_1`*, which was set up so that the user *`lwatchclient`* has no password. In this example, access is being configured on the LMT server node, so the database is local. Thus, the `db_host` is *`localhost`*.

```
filesys.1.name=scratch_1
filesys.1.mountname=/lus/scratch_1
filesys.1.dbhost=localhost
filesys.1.dbport=3306
filesys.1.dbuser=lwatchclient
filesys.1.dbauth=
filesys.1.dbname=filesystem_scratch_1
```

   After setting up `~/.lmtrc`, `lwatch` and `lstat` can be run on this node. To run the GUI from a remote node, the MySQL database must be configured to allow remote access for the read-only user, `lwatchclient`. See *Configure LMT MySQL for Remote Access* on page 99.

7.3 Configure LMT MySQL for Remote Access

In order to run the Lustre monitoring tool (LMT) graphical user interface (GUI) on a separate node from the LMT server, the MySQL server instance (running on the LMT server) must be configured to enable remote access for the LMT read-only user, `lwatchclient`. These MySQL statements can be added to `/usr/share/lmt/mkusers.sql` prior to executing the statements in that file. They can also be executed directly. In these examples, *FSNAME* is the name of the file system being monitored.

```
CREATE USER 'lwatchclient'@'%' IDENTIFIED BY 'foo';
GRANT SELECT ON filesystem_FSNAME.* TO 'lwatchclient'@'%';
```

To execute these statements directly, log on to the DAL MGS node, open a mysql shell as the root MySQL user, and run the statements as follows.

1. Connect to the database as `root`.

```
mgs# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

2. Create `lwatchclient` user.

```
mysql> CREATE USER 'lwatchclient'@'%';
Query OK, 0 rows affected (0.00 sec)
...
```

3. Grant privileges to `lwatchclient` user.

```
mysql> GRANT SELECT ON filesystem_FSNAME.* TO 'lwatchclient'@'%';
Query OK, 0 rows affected (0.00 sec)
```

This enables the user named `lwatchclient` to connect from any hostname.

To allow connections from a certain IP address, replace the '%' with an IP address in single quotes.

```
CREATE USER 'lwatchclient'@'10.11.255.252' IDENTIFIED BY 'foo';
GRANT SELECT ON filesystem_FSNAME.* TO 'lwatchclient'@'10.11.255.252';
```

7.4 LMT Disk Usage

LMT requires at least 40GB persistent storage attached to the LMT server (i.e., the MGS) to store historical data. If the storage becomes full, data can be deleted from the database using MySQL delete statements.

MySQL Tables

Five tables store general file system statistics. These tables are populated by `lmt_agg.cron` script.

Table 12. General File System Tables

Table Name	On-Disk Growth Rate
`FILESYSTEM_AGGREGATE_HOUR`	0.8 KB/hour
`FILESYSTEM_AGGREGATE_DAY`	0.8 KB/day
`FILESYSTEM_AGGREGATE_WEEK`	0.8 KB/week
`FILESYSTEM_AGGREGATE_MONTH`	0.8 KB/month
`FILESYSTEM_AGGREGATE_YEAR`	0.8 KB/year

Table 13. MDS Aggregate Tables and Growth Rates

Table Name	Approximate On-Disk Growth Rate
`MDS_AGGREGATE_HOUR`	0.5 KB/hour/MDS
`MDS_AGGREGATE_DAY`	0.5 KB/day/MDS
`MDS_AGGREGATE_WEEK`	0.5 KB/week/MDS
`MDS_AGGREGATE_MONTH`	0.5 KB/month/MDS
`MDS_AGGREGATE_YEAR`	0.5 KB/year/MDS

Table 14. OST Aggregate Tables and Growth Rates

Table Name	On-Disk Growth Rate
`OST_AGGREGATE_HOUR`	0.7 KB/hour/OST
`OST_AGGREGATE_DAY`	0.7 KB/day/OST
`OST_AGGREGATE_WEEK`	0.7 KB/week/OST
`OST_AGGREGATE_MONTH`	0.7 KB/month/OST
`OST_AGGREGATE_YEAR`	0.7 KB/year/OST

Calculate Expected Disk Usage for a File System

Use this formula to calculate the approximate rate of disk space usage for a file system. Disregard the `AGGREGATE` tables as they grow so much more slowly than the raw data tables.

```
(56 KB/hour/filesystem) * (# of filesystems) + (1000 KB/hour/MDS) * (# of MDSs)
    + (44 KB/hour/OSS) * (# of OSSs) + (70 KB/hour/OST) * (# of OSTs) = Total KB/hour
```

Calculate the Disk Usage for a File System for 1 Year

In this example, LMT is monitoring one file system with one MDS, four object storage servers (OSS), and eight object storage targets (OST). The amount of disk space used by the LMT database to is expected to grow at this hourly rate.

```
56 KB/hour/filesystem * 1 filesystem + 1000 KB/hour/MDS * 1 MDS
    + 44 KB/hour/OSS * 4 OSSs + 70 KB/hour/OST * 8 OSTs = 1792 KB/hour
```

Which translates to this yearly rate.

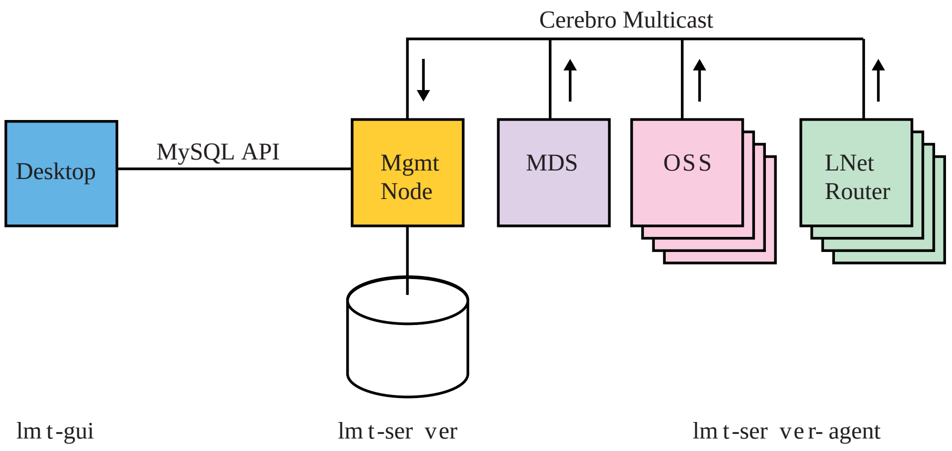
```
1792 KB/hour * 24 hours/day * 365 days/year * 1 MB/1024KB
       * 1 GB/1024MB = 15 GB / year
```

8 LMT Overview

The Lustre monitoring tool (LMT) monitors Lustre file system servers metadata target (MDT), object storage target (OST), and Lustre networking (LNet) routers. It collects data using the Cerebro monitoring system and stores it in a MySQL database. Graphical and text clients are provided which display historical and real time data pulled from the database.

There is currently no support for multiple MDTs in the same filesystem (DNE1).

Figure 10. LMT Block Diagram



View and Aggregate Data

Two commands display data provided by LMT:

- `ltop` displays live data
- `lmtsh` displays historical data from the MySQL database

Configuration of the data aggregation cron job is enabled by using the `cfgset` command.

```
smw# cfgset update -s cray_lmt -m interactive partition
```

Interfaces

An LMT MySQL database is accessed using a MySQL client. The database created is named filesystem_*`fsname`* where *`fsname`* is the name of the filesystem which LMT monitors.

Additional command-line interfaces (CLIs) to LMT are `ltop`, `lmtinit`, and `lmtsh`. These interfaces are only available on the LMT server and `lmtinit` and `lmtsh` can only be used by `root`.

- `ltop` provides access to live data collected by LMT
- `lmtinit` sets up a MySQL database for LMT

- `lmtsh` provides access to data in the LMT MySQL database

The LMT graphical user interface (GUI) package provides two other interfaces to LMT called `lwatch` and `lstat`. `lwatch` is the GUI, and `lstat` provides output similar to the output of `ltop`. Any user with network connectivity to the LMT server and credentials for a MySQL account with read access to the LMT database can use the CLI.

LMT also provides data aggregation scripts that act on raw data in the MySQL database and calculate hourly, daily, and monthly data. The main script for aggregation is `/usr/share/lmt/cron/lmt_agg.cron`.

Dependencies

The MySQL-server runs on the MGS node. The IMPS handles dependencies as long as the packages needed are in the CentOS image repository.

The two-disk RAID which is currently used as the management target (MGT) must split into two volumes in SANtricity. The MGT volume must be 1GB in size. The other volume must be an `ext3` volume using the rest of the space on the disks (599GB unformatted).

The LMT GUI requires the Java runtime environment (JRE) and works best with IBM JRE. This is available on the CentOS media for IMPS DAL.

Failover

The failover MGS can be used as the LMT server as long as all LMT agents (Lustre servers) are configured to send Cerebro messages to both the primary and the failover MGSs. There, Cerebro daemon (`cerebrod`) will be running on the MGS and its failover partner all the time since its failover partner is the metadata server (MDS). However, listening on the failover MGS (the MDS) can be turned off until the MGS failover occurs. The disks used for the MySQL database must be accessible to the primary and failover MGS. The nodes must be prevented from accessing the disks at the same time using STONITH.

If any object storage server (OSS) or MDS fails over, start `cerebrod` on its failover partner when failover has completed.

8.1 View and Aggregate LMT Data

View Data

There are two ways to view data provided by Lustre monitoring tool (LMT). Data can be viewed live with `ltop`. Historical data can be viewed from the MySQL database with `lmtsh`. These utilities are available only on the LMT server. For CLE with direct attached Lustre (DAL), the LMT server is the management server (MGS).

For help using `ltop` or `lmtsh`, see `man` page, or view usage information using the `--help` option.

Because the data is held in a MySQL database on the LMT server, the MySQL database can be directly accessed using MySQL commands if more control is needed over how the data is presented.

Aggregate Data

DAL configuration of the data aggregation cron job is handled through the IMPS configurator. LMT provides scripts which aggregate data into the MySQL database aggregate tables. To run the aggregation scripts, type the following:

```
mgs# /usr/share/lmt/cron/lmt_agg.cron
```

The first time the command is run will take longer than subsequent executions. Use `lmtsh` to see the tables populated by the aggregation scripts. The aggregation script can be set up to run as a cron job.

To set up the cron job:

As `root`, type `crontab -e` and then enter:

```
0 * * * * /usr/share/lmt/cron/lmt_agg.cron
```

This configures the `lmt_agg.cron` job to run every hour, on the hour.

8.2 Remove LMT Data

The Lustre monitoring tool (LMT) does not provide a way to clear old data from the MySQL database. The following `mysql` commands, run on the LMT server (the MGS in CLE systems using DAL), clear all data from the MDS_OPS_DATA table which is older than October 4th at 15:00:00. *`fsname`* is the name of the file system being cleared.

As `root`, access the MySQL database:

```
mgs# mysql -p -e "use filesystem_fsname;
delete MDS_OPS_DATA from
MDS_OPS_DATA inner join TIMESTAMP_INFO
on MDS_OPS_DATA.TS_ID=TIMESTAMP_INFO.TS_ID
where TIMESTAMP < '2013-10-04 15:00:00';"
```

Use the `lmtsh` shell to completely clear a table:

As `root`, type `lmtsh`, to start the `lmtsh` shell, the following at the `lmtsh` prompt where *`TABLE_NAME`* is the name of the table to clear.

1. Launch LMT shell.

   ```
   mgs# lmtsh
   ```

2. Clear LMT data.

   ```
   lmtsh> clr TABLE_NAME
   ```

To clear all aggregated tables:

```
lmtsh> clr agg
```

See the `lmtsh man` page for more information.

8.3 Stop Cerebro and LMT

To stop Cerebro from feeding data into Lustre monitoring tool (LMT), stop the Cerebro daemon (`cerebrod`) from running on all Lustre servers and the LMT server as follows.

1. Stop `cerebrod` on selected nodes.

   ```
   mgs# pdsh -w node-list "/sbin/service cerebrod stop"
   ```

2. Stop `cerebrod` on MGS.

```
mgs# /sbin/service cerebrod stop
```

This will stop the Lustre servers from sending file system data to the Cray management system (CMS). It will also stop Cerebro from listening for data on the CMS. If required, the MySQL database can be deleted as described in this publication.

If `cerebrod` has been turned on with `chkconfig`, it can also be turned off with `chkconfig` so that it won't start every time the system is booted. To turn off `cerebrod`, use the same command as for turning it on, but replace on with `off`. (This does not stop `cerebrod` immediately. Use the `service` command to do that, as shown above.)

```
mgs# chkconfig --level 235 cerebrod off
```

8.4 Delete the LMT MySQL Database

Prerequisites

There must be data stored in the Lustre monitoring tool (LMT) MySQL database to delete.

About this task

This procedure deletes all LMT data.

Procedure

1. Log into the LMT server (the management server (MGS) node in direct-attached Lustre (DAL) systems).
2. Delete the LMT MySQL database where *`fsname`* is the name of the file system to be removed.

   ```
   mgs# lmtinit -d fsname
   ```

3. Optional: Remove the MySQL users added by LMT.

   ```
   mgs# mysql -u root -p -e "drop user 'lwatchclient'@'localhost'; \
   drop user 'lwatchadmin'@'localhost';"
   ```

8.5 LMT Database Recovery Process

The Lustre monitoring tool (LMT) database can be corrupted when the management server (MGS)/primary metadata server (MDS) crashes in a direct-attach Lustre (DAL) file system. The corruption can be repaired by running `mysqlcheck` on the MGS/primary MDS.

Run `mysqlcheck` just after the primary MDS is rebooted. LMT will work as soon as the primary MDS is rebooted so long as the database is usable. If `mysqlcheck` is run after reboot, performance numbers are generated from LMT even when using the secondary MDS.

```
nid00325# mysqlcheck -r -A -p
Enter password:
filesystem_dal.EVENT_DATA                         OK
filesystem_dal.EVENT_INFO                         OK
filesystem_dal.FILESYSTEM_AGGREGATE_DAY           OK
filesystem_dal.FILESYSTEM_AGGREGATE_HOUR          OK
```