Configuring LNet Routers for File Systems based on Intel* EE for Lustre* Software

Partner Guide

High Performance Data Division

November 11, 2016
Software Version: 2.4.n.n or later
World Wide Web: http://www.intel.com

Disclaimer and legal information

Contents

About this Document

Conventions Used

Conventions used in this document include:

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

Related Documentation

- *Intel® Enterprise Edition for Lustre* Software Installation Guide*
- *Intel® Manager for Lustre* Software User Guide*
- *Installing Intel® EE for Lustre* Software on Intel® Xeon Phi™ Coprocessors*
- *Hierarchical Storage Management Configuration Guide*
- *Installing Hadoop, the Hadoop Adapter for Intel® EE for Lustre*, and the Job Scheduler Integration*
- *Creating an HBase Cluster and Integrating Hive on an Intel® EE for Lustre*® File System*
- *Lustre* Installation and Configuration using Intel® EE for Lustre* Software and OpenZFS*
- *Upgrading a Lustre file system to Intel® Enterprise Edition for Lustre* Software (Lustre only)*
- *Creating a Scalable File Service for Windows Networks using Intel® EE for Lustre* Software*
- *Intel® EE for Lustre* Hierarchical Storage Management Framework White Paper*
- *Architecting a High-Performance Storage System White Paper*

An Overview of LNet

Lustre* file systems have the unique capability to run the same global namespace across several different network topologies. The LNet components of Lustre provide this abstraction layer. LNet is an independent project from Lustre and is used for other projects beyond the Lustre file system. LNet was originally based on the Sandia Portals project.

LNet can support Ethernet, InfiniBand*, legacy fabrics (ELAN and MyriNet) and specific compute fabrics as Cray* Gemini, Aries, and Cascade.

LNet is part of the Linux kernel space and allows for full RDMA throughput and zero copy communications when available. Lustre can initiate a multi-OST read or write using a single Remote Procedure Call (RPC), which allows the client to access data using RDMA, regardless of the amount of data being transmitted.

LNet was developed to provide the maximum flexibility for connecting different network topologies using LNet routing. LNet's routing capabilities provide an efficient protocol to enable bridging between different networks, e.g., from Ethernet-to-InfiniBand, or the use of different fabric technologies such as Intel® Omni-Path Architecture (OPA) and InfiniBand.

Figure 1 shows an example of how to connect an existing Infiniband network (storage and compute nodes) to new Intel® OPA compute nodes for more information on Intel® OPA routing options please consult the Intel® Omni Path Storage Router Design Guide.

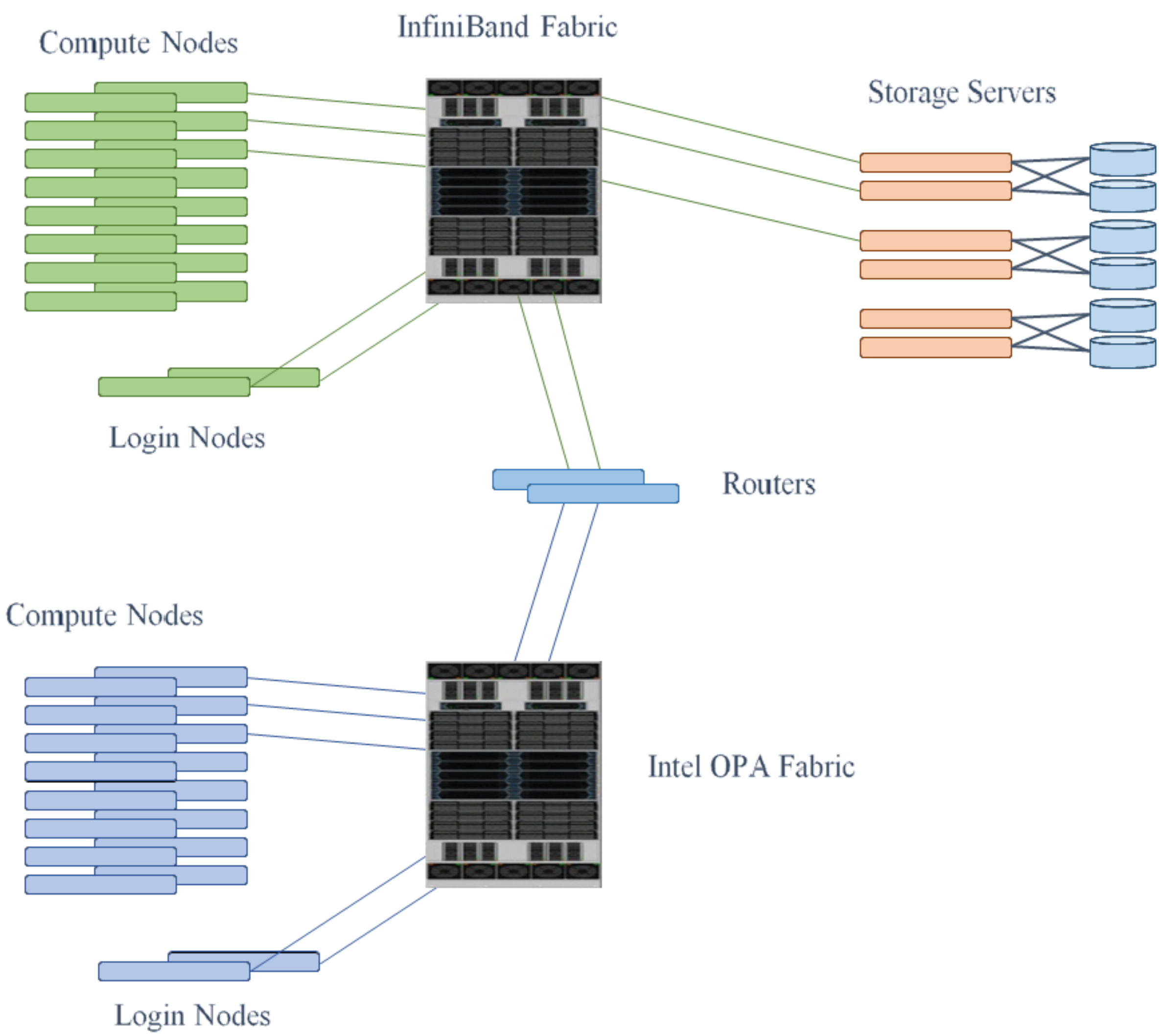

Figure 1. Heterogeneous topology

Configuring LNet

An LNet router is a specialized Lustre client where only the LNet is running. An industry-standard, Intel® based server equipped with two sockets is appropriate for this role.

The Lustre file system is not mounted on the router, and a single LNet router can serve different file systems. In the context of LNet routing between two RDMA enabled networks, in-memory zero copy capability is used in order to optimize latency and performance.

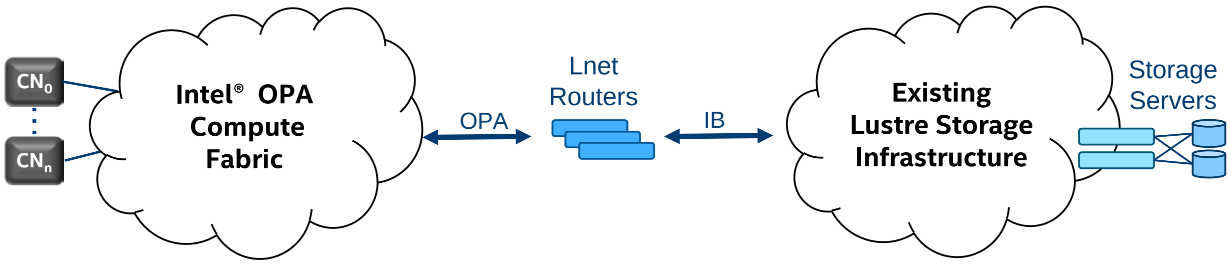

Figure 2. LNet router

Consider the simple example shown in Figure 2 above, where:

- Storage servers are on LAN1, a Mellanox based InfiniBand network – 10.10.0.0/24
- Clients are on LAN2, an Intel® OPA network – 10.20.0.0/24
- The router is between LAN1 and LAN2 at 10.10.0.20 and 10.20.0.29

The network configuration on the servers (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks="o2ib1(ib0)" routes="o2ib2 10.10.0.20@o2ib1"
```

The network configuration on the LNet router (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks="o2ib1(ib0),o2ib2(ib1)" "forwarding=enabled"
```

The network configuration on the clients (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks="o2ib2(ib0)" routes="o2ib1 10.20.0.29@o2ib2"
```

Restarting LNet is necessary to apply the new configuration. Clients will mount the Lustre file system using the usual command line (assuming `mgs1` and `mgs2` are the ip addresses of the two Lustre servers hosting the MGS service on the LAN1 network):

```
# mount -t lustre mgs1@o2ib1:mgs2@o2ib1:/<file system name> /<mount
point>
```

Advanced Configuration

Lustre is designed to avoid any single point of failure and to scale as much as possible. The implementation of LNet routers should also follow this philosophy. We can organize a pool of routers to provide load balancing and high availability.

LNet routers are designed to discover each other and function as a pool (cluster); they monitor peer health and communicate state. In the case of a router failure, they will route traffic in order to avoid the failed peer router.

Routers return state information to clients and clients process the state of each router in the pool. This information is used to load-balance traffic across the entire pool of routers, and a routing table and routing status is maintained on each client.

Referring again to Figure 2, consider this example using pools, where:

- Servers are on LAN1, a Mellanox based InfiniBand network – 10.10.0.0/24
- Clients are LAN2, an Intel® OPA network – 10.20.0.0/24
- Routers on LAN1 and LAN2 at 10.10.0.20-29 and 10.20.0.20-29

The network configuration on the servers (typically created in /etc/modprobe.d/lustre.conf) will be:

```
options lnet networks="o2ib1(ib0)" routes="o2ib2 10.10.0.[20-
29]@o2ib1"
```

The network configuration on the LNet routers (typically created in /etc/modprobe.d/lustre.conf) will be:

```
options lnet networks="o2ib1(ib0),o2ib2(ib1)" "forwarding=enabled"
```

The network configuration on the clients (typically created in /etc/modprobe.d/lustre.conf) will be:

```
options lnet networks="o2ib2(ib0)" routes="o2ib1 10.20.0.[20-
29]@o2ib2"
```

Restarting LNet is necessary to apply the new configuration. Clients will mount the Lustre file system using the usual command line (assuming `mgs1` and `mgs2` are the ip addresses of the two Lustre servers hosting the MGS service on the LAN1 network):

```
# mount –t lustre mgs1@o2ib1:mgs2@o2ib1:/<file system name> /<mount
point>
```

Fine-Grained Routing

The `routes` parameter is used to tell a node which route to use when forwarding traffic, by identifying LNet routers in a Lustre configuration. The `routes` parameter specifies a semi-colon-separated list of router definitions.

```
routes=dest_lnet [hop] [priority] router_NID@src_lnet; \
dest_lnet [hop] [priority] router_NID@src_lnet
```

An alternative syntax consists of a colon-separated list of router definitions:

```
routes=dest_lnet: [hop] [priority] router_NID@src_lnet \
```

```
[hop] [priority] router_NID@src_lnet
```

When there are two or more LNet routers, it is possible to give weighted priorities to each router using the `priority` parameter. Here are some possible reasons for using this parameter:

- One of the routers is more capable than the other.
- One router is a primary router and the other is a back-up.
- One router is for one section of clients and the other is for another section.
- Each router is moving traffic to a different physical location. The priority parameter is optional and need not be specified if no priority exists.

The hop parameter specifies the number of hops to the destination. When a node forwards traffic, the route with the least number of hops is used. If multiple routes to the same destination network have the same number of hops, the traffic is distributed between these routes in a round-robin fashion. To reach/transmit to the LNet `dest_lnet`, the next hop for a given node is the LNet router with the NID `router_NID` in the LNet `src_lnet`.

Given a sufficiently well-architected system, it is possible to map the flow to and from every client or server. This type of routing has also been called *fine-grained routing*.

Advanced Parameters

In a Lustre configuration where different types of LNet networks are connected by routers, several kernel module parameters can be set to monitor and improve routing performance.

The routing related parameters are:

- `auto_down` - Enable/disable (1/0) the automatic marking of router state as up or down. The default value is 1. To disable router marking, enter:

  ```
  options lnet auto_down=0
  ```

- `avoid_asym_router_failure` - Specifies that if even one interface of a router is down for some reason, the entire router is marked as down. This is important because if nodes are not aware that the interface on one side is down, they will still keep pushing data to the other side presuming that the router is healthy, when it really is not. To turn it on, enter:

  ```
  options lnet avoid_asym_router_failure=1
  ```

- `live_router_check_interval` - Specifies a time interval in seconds after which the router checker will ping the live routers. The default value is 60. To set the value to 50, enter:

```
options lnet live_router_check_interval=50
```

- `dead_router_check_interval` - Specifies a time interval in seconds after which the router checker will check the dead routers. The default value is 60. To set the value to 50, enter:

  ```
  options lnet dead_router_check_interval=50
  ```

- `router_ping_timeout` - Specifies a timeout for the router checker when it checks live or dead routers. The router checker sends a ping message to each dead or live router once every `dead_router_check_interval` or `live_router_check_interval` respectively. The default value is 50. To set the value to 60, enter:

  ```
  options lnet router_ping_timeout=60
  ```

- `check_routers_before_use` - Specifies that routers are to be checked before use. Set to off by default. If this parameter is set to on, the `dead_router_check_interval` parameter must be given a positive integer value.

  ```
  options lnet check_routers_before_use=on
  ```

The router_checker obtains the following information from each router:

- time the router was disabled
- elapsed disable time

If the router_checker does not get a reply message from the router within `router_ping_timeout` seconds, it considers the router to be down.

When a router in a priority class goes down, the traffic stops intermittently until LNet safely marks the router that is down as 'down', and then proceeds on again, depending either on other routers of the same class, or a different priority class. The time it takes for LNet to recover is roughly based on the values for the live/dead_router_checker parameters provided.

If a router that is marked 'up' responds to a ping, the timeout is reset. If 100 packets have been sent successfully through a router, the sent-packets counter for that router will have a value of 100. The ping response also provides the status of the NIDs of the node being pinged. In this way, the pinging node knows whether to keep using this node as a next-hop or not. If one of the NIDs of the router is down and the `avoid_asym_router_failure = 1` is set, then that router is no longer used.

LNet dynamic configuration

In lustre version 2.7 (Intel® Enterprise Edition for Lustre* software version 3.0 or later), the LNet can be configured dynamically using the `lnetctl` utility. The `lnetctl` utility, in fact, can be used to initialize LNET without bringing up any network interfaces. This gives flexibility to the user to add interfaces after LNET has been loaded.

In general the `lnetctl` format is as follows:

```
lnetctl cmd subcmd [options]
```

The following configuration items are managed by the tool:

- Configuring/unconfiguring LNET
- Adding/removing/showing Networks
- Adding/removing/showing Routes
- Enabling/Disabling routing
- Configuring Router Buffer Pools

After LNET has been loaded via modprobe (`modprobe lnet`), lnetctl utility can be used to configure LNET without bringing up networks which are specified in the module parameters. It can also be used to configure network interfaces specified in the module parameters by providing the --all option.

```
lnetctl lnet configure [--all]
```

Now LNet is ready to be configured. In the following example we are adding a o2ib1 LNet network using the ib1 InfiniBand card.

```
lnetctl net add --net o2ib1 --if ib1
```

Using the show subcommands, it is possible to review the configuration:

```
lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
```

```
    - net: o2ib1
      nid: 192.168.5.151@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```

To add a routing path is easy as:

```
lnetctl route add --net o2ib0 --gateway 192.168.5.152@o2ib1
```

It is possible to review the configuration using the following command:

```
lnetctl route show --verbose
route:
    - net: o2ib
      gateway: 192.168.5.152@o2ib1
      hop: 1
      priority: 0
      state: down
```

To make this configuration permanent it is necessary to create a YAML file under `/etc/sysconfig/lnet.conf`. We can then import/export the live configuration file:

```
lnetctl export > /etc/sysconfig/lnet.conf
```

The `lnet` script in `/etc/init.d/` is compatible with DLC and should be enabled to be started at boot time:

```
systemctl enable lnet
```

The routing configuration is also managed by the lnet startup script, but uses a different YAML configuration file: `/etc/sysconfig/lnet_routes.conf`.

Based on the previous example the configuration should be:

```
o2ib0: { gateway: 192.168.5.152@o2ib1 }
```

More information is available in the Lustre Operations Manual.

Troubleshooting

LNet provides a several metrics to troubleshoot a network. Referencing Figure 2 again, considering the following configuration:

- six Lustre servers are on LAN0 (o2ib0), a Mellanox based InfiniBand network – 192.168.3.[1-6]
- sixteen clients are LAN1 (o2ib1), an Intel® OPA network – 192.168.5.[100-254]
- two routers on LAN0 and LAN1 at 192.168.3.7-8 and 192.168.5.7-8

On each Lustre client we can see the status of the connections using the /proc/sys/lnet/peers metric file. This file shows all NIDs known to this node, and provides information on the queue state:

```
# cat /proc/sys/lnet/peers
nid                 refs state  last   max   rtr   min   tx  min queue
192.168.5.8@o2ib1     4    up    -1     8     8     8     8  -505 0
192.168.5.7@o2ib1     4    up    -1     8     8     8     8  -473 0
```

Here, “state” is the status of the routers. In the case of a failure of one path, I/O will be routed through the surviving path. When both paths are available, RPCs will use both paths in round-robin.

Here, “max” is the maximum number of concurrent sends from this peer and “tx” is the number of peer credits currently available for this peer.

Notice the negative number in the “min” column. This negative value means that the number of slots on the LNet was not sufficient and the queue was overloaded. This is an indication to increase the number of *peer credits* and *credits* (see LNet Tuning). Increasing the credits value has some drawbacks, including increased memory requirements and possible congestion in networks with a very large number of peers.

The status of the routing table can be obtained from the /proc/sys/lnet/routes file from a client:

```
Routing disabled
net      hops priority   state router
o2ib        1        0      up 192.168.5.8@o2ib1
o2ib        1        0      up 192.168.5.7@o2ib1
```

The status of the routers can be verified from the /proc/sys/lnet/routers file from a client:

```
ref rtr_ref alive_cnt state last_ping ping_sent deadline down_ni router
4      1        3     up        47         1      NA      0 192.168.5.7@o2ib1
4      1        1     up        47         1      NA      0 192.168.5.8@o2ib1
```

On each LNet router, the /proc/sys/lnet/peers metric shows all NIDs known to this node, and provides the following information (values are examples and not all information is shown):

```
nid                  refs state  last   max   rtr   min    tx   min
queue
192.168.3.4@o2ib    1   up    165     8     8    -8     8   -15    0
192.168.3.1@o2ib    1   up     47     8     8    -6     8    -8    0
192.168.3.6@o2ib    1   up    165     8     8    -8     8   -15    0
192.168.3.3@o2ib    1   down    115     8     8    -8     8   -12
0
192.168.3.5@o2ib    1   up    153     8     8    -8     8    -8    0
192.168.3.2@o2ib     1  up     83     8     8     8     8     7    0
192.168.5.134@o2ib1  1  up     65     8     8    -8     8    -6    0
192.168.3.104@o2ib   1  down  9999    8     8    -8     8    -1    0
192.168.5.139@o2ib1  1  up    127     8     8    -4     8   -13    0
192.168.5.131@o2ib1  1  up     67     8     8    -8     8   -26    0
192.168.5.144@o2ib1  1  up    170     8     8    -3     8   -12    0
192.168.5.136@o2ib1  1  up    151     8     8    -4     8    -7    0
192.168.3.106@o2ib   1  down  9999    8     8     4     8     4    0
192.168.5.141@o2ib1  1  up     58     8     8    -3     8    -9    0
192.168.5.133@o2ib1  1  up    178     8     8    -8     8   -14    0
192.168.5.146@o2ib1  1  up     63     8     8    -4     8   -18    0
.  .  .
```

In the output above, we can see some Lustre clients on LNet0 are down.

Credits are initialized to allow a certain number of operations. In the example in the above table, this value is 8 (eight), shown under the max column. LNet keeps track of the minimum number of credits ever seen over time showing the peak congestion that has occurred during the time monitored. Fewer available credits indicates a more congested resource.

The number of credits currently in flight (number of transmit credits) is shown in the “tx” column. The maximum number of send credits available is shown in the “max” column and that never changes. The number of router buffers available for consumption by a peer is shown in the “rtr” column.

Therefore, rtr – tx is the number of transmits in flight. Typically, rtr == max, although a configuration can be set such that max >= rtr. The ratio of routing buffer credits to send credits (rtr/tx) that is less than max indicates operations are in progress. If the ratio rtr/tx is greater than max, operations are blocking.

LNet also limits concurrent sends and number of router buffers allocated to a single peer, so that no peer can occupy all these resources.

Real time statistics of the LNet router can be obtained using the "routerstat" command. Routerstat watches LNet router statistics. If no interval is specified, stats are sampled and printed only once; otherwise, stats are sampled and printed every interval. Output includes the following fields:

- M - msgs_alloc(msgs_max)
- E - errors
- S - send_count/send_length
- R - recv_count/recv_length
- F - route_count/route_length
- D - drop_count/drop_length

LNet Tuning

LNet tuning is possible by passing parameters to the Lustre Network Driver (LND). The Lustre Network Driver for RDMA is the **ko2iblnd** kernel module. This driver is used both for Intel® OPA cards and InfiniBand cards.

Intel® Enterprise Edition for Lustre* Software (version 2.4 and later) will detect a network card and, using the `/usr/sbin/ko2iblnd-probe`, set tunable parameters for supported cards.

Intel® OPA cards are automatically detected and configured by the script to achieve optimal performance with Lustre. The script can be modified to detect other network cards and set optimal parameters.

The following example is a configuration file (`/etc/modprobe.d/ko2iblnd.conf`) for a Lustre peer with an older version of Lustre.

```
options ko2iblnd peer_credits=128 peer_credits_hiw=64 credits=1024
concurrent_sends=256 ntx=2048 map_on_demand=32 fmr_pool_size=2048
fmr_flush_trigger=512 fmr_cache=1
```

Starting from Intel® EE for Lustre* software version 3.0.1 or later, is possible to configure different LNet tunables for each card, previously all peers (compute nodes, LNet router, servers) on the network require identical tunable parameter for LNet to work independently from the hardware technology used (Intel® OPA or InfiniBand).

If you are routing into a fabric with older Lustre nodes, these must be updated to apply identical options to the ko2iblnd module.

LNet uses `peer credits` and a network interface `credits` to send data through the network with a fixed MTU size of 1MB.

The `peer_credits` tunable parameter manages the number of concurrent sends to a single peer and can be monitored using the `/proc/sys/lnet/peers` interface. The number for `peer_credits` can be increased using a module parameter for the specific Lustre Network Driver (LND):

```
ko2iblnd-opa peer_credits=128
default value is 8
```

It is not always mandatory to increase `peer_credits` to obtain good performance, because in a very large installation, an increased value can overload the network and increase the memory utilization of the OFED stack.

The tunable network interface credits (`credits`) limits the number of concurrent sends to a single network, and can be monitored using the `/proc/sys/lnet/nis` interface. The number of network interface credits can be increased using a module parameter for the specific Lustre Network Driver (LND):

```
ko2iblnd-opa credits=1024
```

The default value is 64 and it shared across all the CPU partitions (CPTs).

Fast Memory Registration (FMR) is a technique used to reduce memory allocation costs. In FMR, memory registration is divided in two phases: 1) allocating resources needed by the registration and then 2) registering using resources obtained from the first step. The resource allocation and de-allocation can be managed in batch mode, and as result, FMR can achieve a much faster memory registration. To enable FMR in LNet, the value for `map_on_demand` should be more than zero.

```
ko2iblnd-opa map_on_demand=32
```

The default value is 0.

Fast Memory Registration is supported by Intel® OPA and Mellanox FDR cards (based on the mlx4 driver), but it is not supported by Mellanox FDR/EDR cards (based on the mlx5 driver).

Table 1. Lustre suggested tunables for Intel® OPA

Tunable	Suggested Value	Default Value
peer_credits	128	8
peer_credits_hiw	64	0
Credits	1024	64
concurrent_sends	256	0
ntx	2048	512
map_on_demand	32	0

fmr_pool_size	2048	512
fmr_flush_trigger	512	384
fmr_cache	1	1

From the above table:

- `peer_credits_hiw` sets high water mark to start to retrieve credits
- `concurrent_sends` is the number of concurrent HW sends to a single peer
- `ntx` is the number of message descriptors allocated for each pool
- `fmr_pool_size` is the size of the FMR pool on each CPT
- `fmr_flush_trigger` is the number of dirty FMRs that triggers a pool flush
- `fmr_cache` should be set to non-zero to enable FMR caching.

Note that file systems running Intel® EE for Lustre* Software achieved higher performance with Intel® OPA cards using the tuning parameters in Table 1, above.

The following string is automatically generated by the `/usr/sbin/ko2iblnd-probe script`:

```
options ko2iblnd-opa peer_credits=128 peer_credits_hiw=64
credits=1024 concurrent_sends=256 ntx=2048 map_on_demand=32
fmr_pool_size=2048 fmr_flush_trigger=512 fmr_cache=1
```

Designing LNet Routers to Connect Intel® OPA and InfiniBand*

The LNet router can be deployed using an industry standard server with enough network cards and the LNet software stack. Designing a complete solution for a production environment is not an easy task, but Intel® is providing tools (LNet Self Test) to test and validate the configuration and performance in advance.

The goal is to design LNet routers with enough bandwidth to satisfy the throughput requirements of the back-end storage. The number of compute nodes connected to an LNet router normally doesn't change the design of the solution.

The bandwidth available to an LNet router is limited by the slowest network technology connected to the router. Typically, we've observed a 10-15% decline in bandwidth from the nominal hardware bandwidth of the slowest card, due to the LNet router.

In every case, we encourage validating the implemented solution using tools provided by the network interface maker and/or the LNet Self Test utility, which is available with Lustre.

LNet routers can be congested if the number of credits (peer_credits and credits) are not set properly. For communication to routers, not only a credit and peer credit must be tuned, but a global router buffer and peer router buffer credit are needed.

To design an LNet router in this context, we need to consider the following topics**:**

- Hardware design and tuning
- Software compatibility

Hardware Design and Tuning

When designing an LNet router between two different network technologies such as Mellanox InfiniBand and Intel® OPA, one should consider that LNet was developed taking advantage of the RDMA zero copy capability. This makes the LNet router extremely efficient.

To achieve higher performance from Intel® OPA in a Lustre file system, one must tune the LNet stack as described in LNet Tuning and in the Intel® Omni-Path Performance Tuning user guide. However as shown here in the Software Compatibility, some higher-performing hardware combinations are not desirable with specific software versions because some Mellanox cards based on the mlx5 driver don't support the `map_on_demand` tuning parameter. This issue is addressed in the Intel® EE for Lustre* Software, version 3.0.1 or later.

CPU Selection

General speaking, the CPU performance is not critical for the LNet router code, and the recent SMP affinity implementation enables the LNet code to scale on NUMA servers. Figure 1 shows the CPU utilization of two LNet routers configured for load balancing and routing an Intel® OPA client network and a Mellanox FDR storage network during a large IOR test. The activity between the two routers is completely specular and balanced. The CPU utilization is below 10% to sustain a FDR card. The CPU activity is two times during WRITE compared to READ. Both routers were equipped with two Intel® Xeon® Processor E5-2697 v2 CPUs clocked at 2.7 GHz.

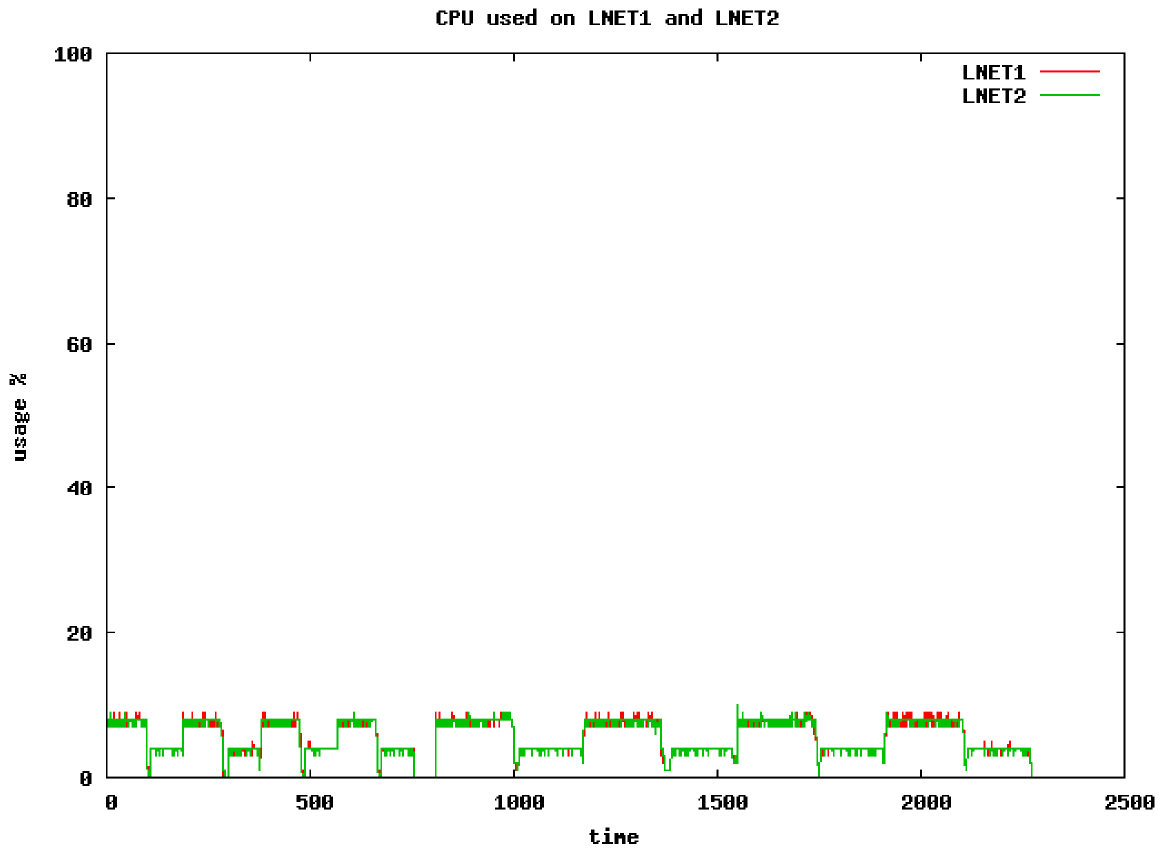

Figure 2. CPU utilization for two LNet routers routing between an Intel® OPA network an a Mellanox FDR network

To obtain higher performance, we suggest turning off the Hyper-Threading Technology and Frequency Scaling capabilities of the CPU (see below).

Table 2. LNet router CPU tuning

Hardware		Recommendation
CPU		E5-2640 v4
HT		OFF
CPU Frequency Scaling		DISABLED

It is important to select the right PCI-e slot in the server for the Intel® OPA and IB cards in order to avoid long distance paths in the NUMA architecture. See Figure 3.

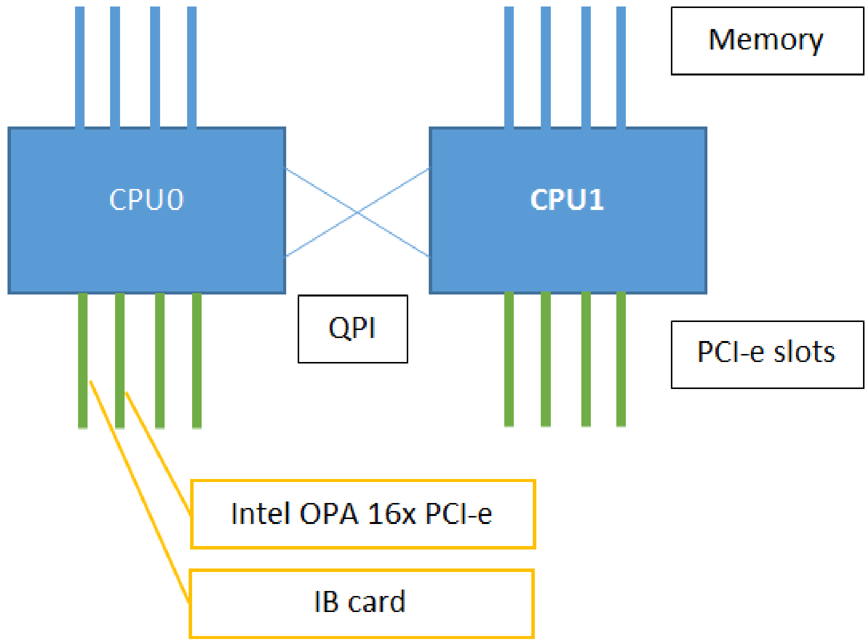

Figure 3. PCI-e slot allocation

Memory Considerations

An LNet router uses additional credit accounting when it needs to forward a packet for another peer:

- Peer Router Credit: This credit manages the number of concurrent receives from a single peer and prevent single peer from using all router buffer resources. By default this value should be 0. If this value is 0 LNet router uses peer_credits.
- Router Buffer Credit: This credit allows messages to be queued and select non data payload RPC versus data RPC to avoid congestion. In fact, an LNet Router has a limited number of buffers:
    - `tiny_router_buffers` – size of buffer for messages of <1 page size
    - `small_router_buffers` – size of buffer for messages of 1 page in size
    - `large_router_buffers` – size of buffer for messages >1 page in size

These LNet kernel module parameters can be monitored using the `/proc/sys/lnet/buffers` file and are available per CPT:

```
pages count credits     min
    0   512     512     503
```

```
  0   512     512     504
  0   512     512     497
  0   512     512     504
  1  4096    4096    4055
  1  4096    4096    4050
  1  4096    4096    4048
  1  4096    4096    4072
256   256     256     244
256   256     256     248
256   256     256     240
256   256     256     246
```

Negative numbers in the “min” column above indicate that the buffers have been oversubscribed; we can increase the number of router buffers for a particular size to avoid stalling.

The memory utilization of the LNet router stack is caused by the Peer Router Credit and Router Buffer Credit parameters. A LNet router with a RAM size of 32GB or more has enough memory to sustain very large configurations for these parameters. In every case, the memory consumption of the LNet stack can be measured using the `/proc/sys/LNet/lnet_memused` metrics file.

Table 3: Table LNet router memory

Hardware	Recommendation
RAM	32GB
Technology	DDR3 or DDR4 ECC

Software Compatibility

This section discusses compatibility considerations for the software stack to be used:

- The Intel® Fabric Suite (IFS) for Intel® OPA supports RHEL 7.2.
- The Mellanox OFED 3.x stack is supported by Intel® EE for Lustre* Software, version 2.4 or later.
- Intel® EE for Lustre* Software, version 2.4 supports RHEL 7.2 as Lustre client and LNet Router only.
- Intel® EE for Lustre* Software, version 3.0 supports RHEL 7.2 as Lustre client, LNet Router and Storage Server.

- Intel® EE for Lustre* Software, version 3.0.1 is supporting RHEL 7.2 as Lustre client, LNet Router and Storage Server, optimization for ConnectX-4/IB cards and per card LNet tunable.

Table 4 lists other possible/common configurations and following sections describe two common and practical implementations.

Table 4: Intel® EE for Lustre* Software version compatibility matrix

<table>
<tr><th>Use case</th><th>Compute Node</th><th>LNet Router</th><th>Storage Server</th></tr>
<tr><td>Legacy storage support or legacy lustre version (2.5).
See Use Case #1 note.</td><td>Intel® OPA and Intel® EE for Lustre* software version 2.4</td><td>Intel® OPA or Mellanox ConnectX-3 and Intel® EE for Lustre* software version 2.4</td><td>Mellanox ConnectX-3 and Intel® EE for Lustre* software version 2.4</td></tr>
<tr><td>Legacy storage support on new Lustre version (2.7)
See Use Case #2 note.</td><td>Intel® OPA and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Intel® OPA or Mellanox ConnectX-3 and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Mellanox ConnectX-3 and Intel® EE for Lustre* software version 3.0.1 or later</td></tr>
<tr><td>New storage support on new Lustre* version (2.7)
See Use Case #3 note.</td><td>Intel® OPA and
Intel® EE for Lustre* software version 3.0.1 or later</td><td>Intel® OPA or Mellanox ConnectX-4 or IB and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Mellanox ConnectX-4 or IB and Intel® EE for Lustre* software version 3.0.1 or later</td></tr>
<tr><td>Old cluster based on old lustre version (2.5) and new storage.
See Use Case #4 note</td><td>Mellanox ConnectX-3 and Intel® EE for Lustre* software version 2.4</td><td>Mellanox ConnectX-3 or Intel® OPA and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Intel® OPA and Intel® EE for Lustre* software version 3.0.1 or later</td></tr>
<tr><td>New cluster based on Infiniband and new storage based on OPA.
See Use Case #5 note.</td><td>Mellanox ConnectX-4 or IB and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Mellanox ConnectX-4 or IB or Intel® OPA and Intel® EE for Lustre* software version 3.0.1 or later</td><td>Intel® OPA and Intel® EE for Lustre* software version 3.0.1 or later</td></tr>
<tr><td colspan="4">Use Case #1: Compute nodes and LNet Routers on RHEL 7; Storage Server on RHEL 6. Support only for ConnectX-3 legacy cards. LNet tunables for OPA across all the cluster.
Use Case #2: Compute nodes and LNet Routers on RHEL 7; Storage Server on RHEL 7. Different LNet tunables for OPA and Infiniband cards.
Use Case #3: Compute nodes and LNet Routers on RHEL 7; Storage Server on RHEL 7. Different LNet tunables for OPA and Infiniband cards, optimizations for mlx5 drivers.
Use Case #4: Storage Servers and LNet Routers on RHEL 7 due the support of Intel® Fabric Suite for Intel® OPA and lustre 2.7. Lustre client on RHEL 6 and lustre 2.5.
Use Case #5: Storage Servers and LNet Routers on RHEL 7.2 due the support of Intel® Fabric Suite for Intel® OPA.</td></tr>
</table>

Practical implementations

This section covers two of the most common network interface configurations:

- Example 1: Legacy storage with Infiniband card ConnextX-3/IB/4 connected to new compute nodes using OPA.
- Example 2: New storage with OPA connected to legacy compute nodes on Infiniband card ConnextX-3/IB/4.

Note: Throughout these examples, IP addresses are examples only.
In these two example configurations, we made the assumption that all the components can be upgraded. Please consult an Intel® Lustre specialist for non-destructive methods to upgrade Lustre. We will use as much as possible the Dynamic LNet Configuration (DLC) technology.

Example 1: Legacy storage with Infiniband card connected to new compute nodes using OPA

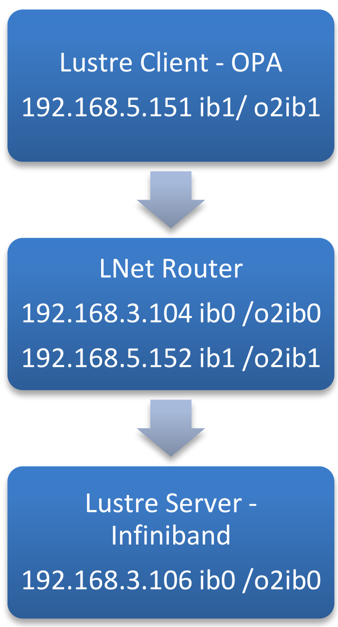

Figure 4 Network topology for Example 1 configuration

Figure 4 shows this simplified network topology:

- a Lustre client equipped with an Intel® Omni-Path fabric card
- an LNet router equipped with an OPA card and an Infiniband card
- a legacy Lustre server equipped with an Infiniband card

Perform the following procedures to achieve this configuration:

1. Upgrade all Lustre servers to Intel® EE for Lustre* software, version 3.0.1 or later. See the *Intel® Enterprise Edition for Lustre* Software Installation Guide* for instructions.
2. On all Lustre Clients and LNet Routers, install Intel® EE for Lustre* software version 3.0.1 or later. See the *Intel® Enterprise Edition for Lustre* Software Installation Guide* for instructions.
3. Perform the steps in the section Configure Lustre Clients (example 1).
4. Perform the steps in the section Configure LNet Routers (example 1).
5. Perform the steps in the section Configure Lustre Servers (example 1).

Configure Lustre Clients (example 1)

The following commands are based on the topology in Figure 4

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl route add --net o2ib0 --gateway 192.168.5.152@o2ib1
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.151@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
```

```
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"

#lnetctl route  show --verbose
route:
    - net: o2ib
      gateway: 192.168.5.152@o2ib1
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo “o2ib0: { gateway: 192.168.5.152@o2ib1 }” >
/etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```

Configure LNet Routers (example 1)

By default Intel® EE for Lustre* software will deploy the following ko2iblnd configuration (`/etc/modprobe.d/koblnd.conf`) in order to optimize any existing OPA card:

```
alias ko2iblnd-opa ko2iblnd
options ko2iblnd-opa peer_credits=128 peer_credits_hiw=64
credits=1024 concurrent_sends=256 ntx=2048 map_on_demand=32
fmr_pool_size=2048 fmr_flush_trigger=512 fmr_cache=1

install ko2iblnd /usr/sbin/ko2iblnd-probe
```

Some of the above parameters are not compatible with Infiniband cards, so we will use DLC to set per-card parameters using the following procedure:

```
#modprobe lnet

#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl net add --net o2ib0 --if ib0
#lnetctl set routing 1
#lnetctl net show --verbose
net:
```

```
- net: lo
  nid: 0@lo
  status: up
  tunables:
      peer_timeout: 0
      peer_credits: 0
      peer_buffer_credits: 0
      credits: 0
      CPT: "[0,0,0,0]"
- net: o2ib1
  nid: 192.168.5.152@o2ib1
  status: up
  interfaces:
      0: ib1
      lnd tunables:
          peercredits_hiw: 64
          map_on_demand: 32
          concurrent_sends: 256
          fmr_pool_size: 2048
          fmr_flush_trigger: 512
          fmr_cache: 1
  tunables:
      peer_timeout: 180
      peer_credits: 128
      peer_buffer_credits: 0
      credits: 1024
      CPT: "[0,0,0,0]"
- net: o2ib
  nid: 192.168.3.104@o2ib
  status: up
  interfaces:
      0: ib0
      lnd tunables:
          peercredits_hiw: 64
          map_on_demand: 32
          concurrent_sends: 256
          fmr_pool_size: 2048
          fmr_flush_trigger: 512
          fmr_cache: 1
  tunables:
      peer_timeout: 180
      peer_credits: 128
      peer_buffer_credits: 0
      credits: 1024
      CPT: "[0,0,0,0]"
```

To edit the configuration and to make it permanent, we will export it:

```
#lnetctl export > /etc/sysconfig/lnet.conf
```

Infiniband cards based on the mlx5 driver are not compatible with the `map_on_demand=32` and other parameters.

For the Infiniband card, edit the `lnet.conf` file and add the following new parameters (bolded and underlined below).

```
- net: o2ib
   nid: 192.168.3.104@o2ib
   status: up
   interfaces:
       0: ib0
       lnd tunables:
           peercredits_hiw: 7
           map_on_demand: 0
           concurrent_sends: 8
           fmr_pool_size: 512
           fmr_flush_trigger: 384
           fmr_cache: 1
   tunables:
       peer_timeout: 180
       peer_credits: 128
       peer_buffer_credits: 0
       credits: 256
       CPT: "[0,0,0,0]"
```

To enable the configuration at startup:

```
#systemctl enable lnet
```

Configure Lustre Servers (example 1)

The Lustre servers should already be configured, however we need to change the configuration in order to add the routing path to the OPA network and enable the new OPA clients through the LNet routers.

Remove any LNet configuration normally in `/etc/modprobe.d/lustre.conf`

```
#modprobe lnet

#lnetctl lnet configure
#lnetctl net add --net o2ib0 --if ib0
#lnetctl route add --net o2ib1 --gateway 192.168.3.104@o2ib0
```

```
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib
      nid: 192.168.3.106@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 4
              map_on_demand: 0
              concurrent_sends: 8
              fmr_pool_size: 512
              fmr_flush_trigger: 384
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          CPT: "[0,0,0,0]"


#lnetctl route  show --verbose
route:
    - net: o2ib1
      gateway: 192.168.3.104@o2ib
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo “o2ib1: { gateway: 192.168.3.104@o2ib0 }” >
/etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```

Example 2: New storage with OPA connected to legacy compute nodes on Infiniband cards

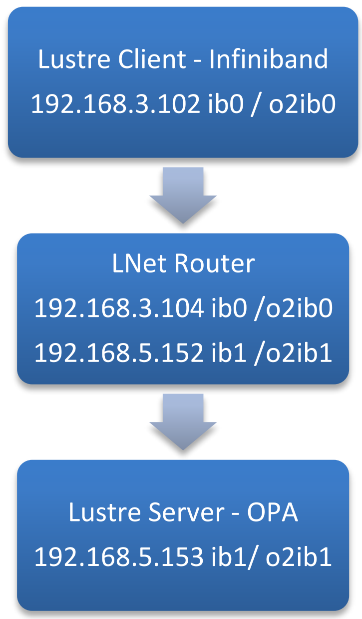

Figure 5 Network topology for adding new OPA-connected servers

Figure 5 shows another common example of a simplified network topology:

- a legacy Lustre client connected via an Infiniband card
- an LNet Router implementing OPA and Infiniband cards
- a Lustre server connected via an Intel® OPA card.

Perform the following procedures to achieve this configuration:

1. Upgrade all Lustre clients to Intel® Enterprise Edition for Lustre* software 3.0.1 or later. See the *Intel® Enterprise Edition for Lustre* Software Installation Guide* for instructions.
2. On all Lustre servers and routers, install Intel® EE for Lustre* software 3.0.1 or later. See the *Intel® Enterprise Edition for Lustre* Software Installation Guide* for instructions.
3. Configure Lustre Clients (example 2).
4. Configure LNet Routers (example 2).
5. Configure Lustre Servers (example 2).

Configure Lustre Clients (example 2)

The following commands are based on the topology in Figure 5. Reconfigure the clients after upgrade, removing the `/etc/modprobe.d/lustre.conf`

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib0 --if ib0
#lnetctl route add --net o2ib1 --gateway 192.168.3.104@o2ib0
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib
      nid: 192.168.3.102@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 4
              map_on_demand: 0
              concurrent_sends: 8
              fmr_pool_size: 512
              fmr_flush_trigger: 384
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          CPT: "[0,0,0,0]"


#lnetctl route  show --verbose
route:
    - net: o2ib1
      gateway: 192.168.3.104@o2ib
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo "o2ib1: { gateway: 192.168.3.104@o2ib0 }" >
/etc/sysconfig/lnet_routes.conf

#systemctl enable lnet
```

Configure LNet Routers (example 2)

By default Intel® EE for Lustre* software will deploy the following ko2iblnd configuration (`/etc/modprobe.d/koblnd.conf`) in order to optimize any existing OPA card:

```
alias ko2iblnd-opa ko2iblnd
options ko2iblnd-opa peer_credits=128 peer_credits_hiw=64
credits=1024 concurrent_sends=256 ntx=2048 map_on_demand=32
fmr_pool_size=2048 fmr_flush_trigger=512 fmr_cache=1

install ko2iblnd /usr/sbin/ko2iblnd-probe
```

Some of the tunables are not compatible with Infiniband cards, so we will use DLC to set per-card tunables using the following procedure:

```
#modprobe lnet

#lnetctl lnet configure

#lnetctl net add --net o2ib1 --if ib1
#lnetctl net add --net o2ib0 --if ib0
#lnetctl set routing 1

#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.152@o2ib1
      status: up
```

```
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
    - net: o2ib
      nid: 192.168.3.104@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```

To edit the configuration and to make the configuration permanent, we will export it:

```
#lnetctl export > /etc/sysconfig/lnet.conf
```

Infiniband cards based on the mlx5 driver are not compatible with the `map_on_demand=32` and other parameters.

For the Infiniband card, edit the `lnet.conf` file and add the following new parameters (bolded and underlined below):

```
    - net: o2ib
      nid: 192.168.3.104@o2ib
```

```
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 7
              map_on_demand: 0
              concurrent_sends: 8
              fmr_pool_size: 512
              fmr_flush_trigger: 384
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 256
          CPT: "[0,0,0,0]"
```

To enable the configuration at startup:

```
#systemctl enable lnet
```

Configure Lustre Servers (example 2)

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl route add --net o2ib0 --gateway 192.168.5.152@o2ib1
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.153@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
```

```
                map_on_demand: 32
                concurrent_sends: 256
                fmr_pool_size: 2048
                fmr_flush_trigger: 512
                fmr_cache: 1
        tunables:
            peer_timeout: 180
            peer_credits: 128
            peer_buffer_credits: 0
            credits: 1024
            CPT: "[0,0,0,0]"

#lnetctl route show --verbose
route:
    - net: o2ib
      gateway: 192.168.5.152@o2ib1
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo “o2ib0: { gateway: 192.168.5.152@o2ib1 }” >
/etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```