



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

Partner Guide

High Performance Data Division

Software Version: 2.4.0.0 or later

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

Disclaimer and legal information

Copyright 2016 Intel® Corporation. All Rights Reserved.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel® Corporation or its suppliers or licensors. Title to the Material remains with Intel® Corporation or its suppliers and licensors. The Material contains trade secrets and proprietary and confidential information of Intel® or its suppliers and licensors. The Material is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel® in writing.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL® ASSUMES NO LIABILITY WHATSOEVER AND INTEL® DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel® Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL® AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL® OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL® PRODUCT OR ANY OF ITS PARTS.

Intel® may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel® reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Before using any third party software referenced herein, please refer to the third party software provider's website for more information, including without limitation, information regarding the mitigation of potential security vulnerabilities in the third party software.

Contact your local Intel® sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel® literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>.

Intel® and the Intel® logo are trademarks of Intel® Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)

Contents

About this Document	ii
Conventions Used	ii
Related Documentation	ii
An Overview of LNet	1
Configuring LNet.....	2
Advanced Configuration	3
Fine-Grained Routing.....	4
Advanced Parameters.....	5
Troubleshooting.....	7
LNet Tuning.....	9
Designing LNet Routers to Connect Intel® OPA and InfiniBand*	12
Hardware Design and Tuning	12
CPU Selection.....	13
Memory Considerations.....	14
Software Compatibility.....	15

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® 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*
- *Creating a Monitored Lustre® Storage Solution over a ZFS File System*
- *Creating a High-Availability Lustre® Storage Solution over a ZFS File System*
- *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.

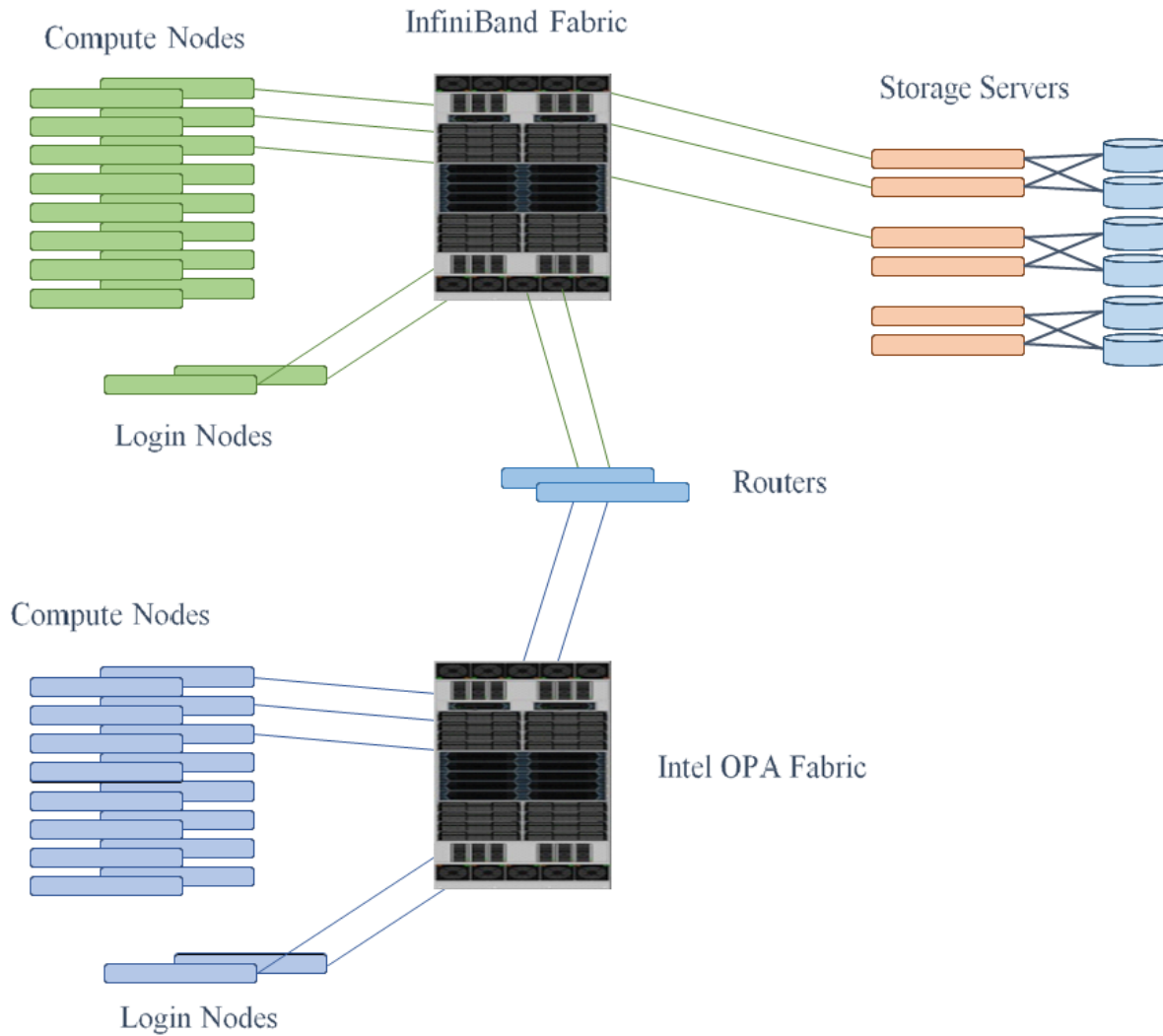


Figure 1. Heterogeneous topology

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.

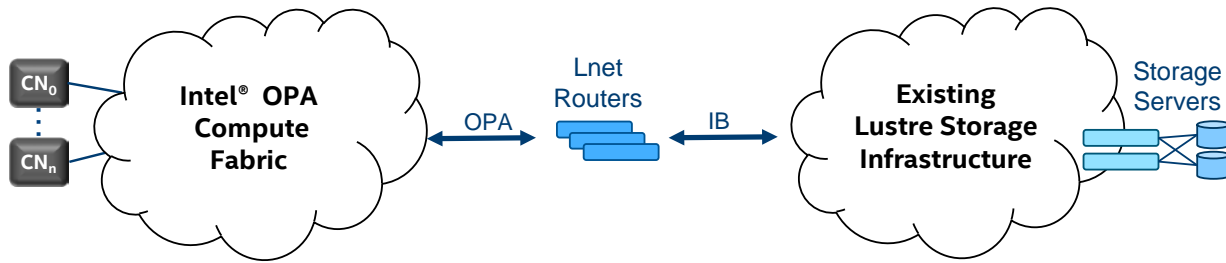


Figure 2. LNet router

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.con) 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.con) will be:

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

The network configuration on the clients (typically created in /etc/modprobe.d/lustre.con) 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.

LNNet 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.

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/fs/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/fs/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 $max \geq rtr$. 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.

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).

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 and Intel® True Scale 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.

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

The following example is a configuration file (`/etc/modprobe.d/ko2iblnd.conf`) for a Lustre peer with a Mellanox ConnectX-3 card on a machine running Intel® EE for Lustre® Software, version 2.4.

```
alias ko2iblnd-mlx4 ko2iblnd
options ko2iblnd-mlx4 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
```

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
```

Intel® EE for Lustre® Software, version 3.0 will support different optimizations based on the networking technology used, and it will also be possible to change these parameters online without unloading the LNet kernel modules.

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 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 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.

```
ko2ib1nd-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 tunable for Intel® OPA

Tunable	Suggested Value	Default Value
<code>peer_credits</code>	128	8
<code>peer_credits_hiw</code>	64	0
<code>Credits</code>	1024	64
<code>concurrent_sends</code>	256	0
<code>ntx</code>	2048	512
<code>map_on_demand</code>	32	0
<code>fmr_pool_size</code>	2048	512
<code>fmr_flush_trigger</code>	512	384
<code>fmr_cache</code>	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](#). However as shown in Table 2, some higher-performing hardware combinations are not desirable because certain Mellanox cards based on the `mlx5` driver don't support the `map_on_demand` tuning parameter. This issue will be addressed in the Intel® EE for Lustre® Software, version 3.0, however for the time being,

Table 2 identifies hardware combinations providing optimum performance.

Table 2: Expected performance based on the network topology

Compute Node	LNet Router	Storage Server	Notes
Intel® OPA	Intel® OPA/Mellanox mlx4	Mellanox mlx4	Optimized Performance
Intel® OPA	Intel® OPA/Mellanox mlx4	Mellanox mlx5	Un-optimized Performance
Mellanox mlx4	Mellanox mlx4/Intel® OPA	Intel® OPA	Optimized Performance
Mellanox mlx5	Mellanox mlx4/Intel® OPA	Intel® OPA	Un-optimized Performance

Also see [Software Compatibility](#) for additional considerations.

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.

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

Table 3: LNet router CPU tuning

Hardware	Recommendation
CPU	E5-2697 v3
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.

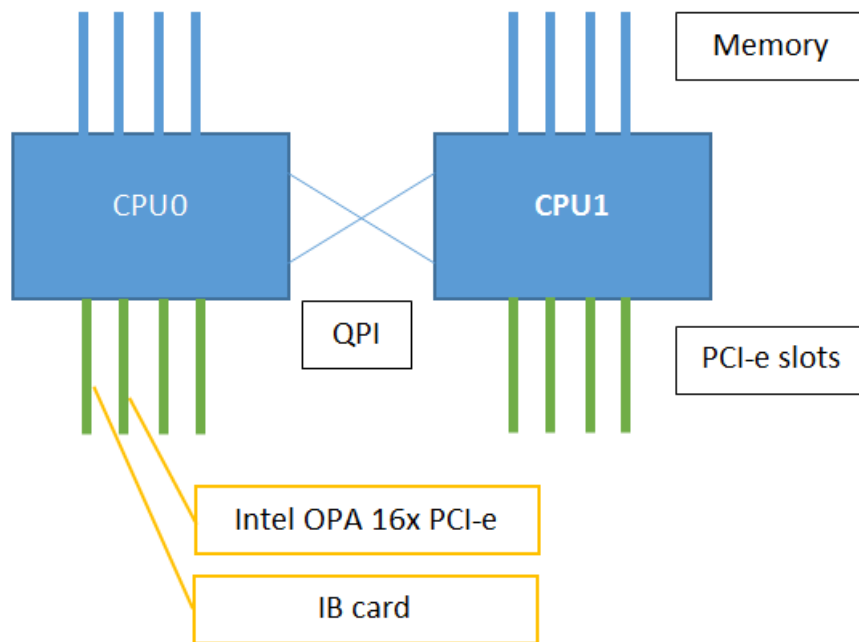


Figure 3: PCI-e slot allocation

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 64GB 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` metric's file.

Software Compatibility

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

- The Intel® Fabric Suite (IFS) for Intel® OPA supports RHEL 7.1+.
- The Mellanox OFED 3.1 stack is supported from Intel® EE for Lustre® Software, version 2.4 or later.
- Intel® EE for Lustre® Software, version 2.4 is supporting RHEL 7.1+ as Lustre client and LNet Router only.
- Intel® EE for Lustre® Software, version 3.0 will support RHEL 7.2+ as Lustre client, LNet Router and Storage Server.

Table 4: Table LNet router memory

Hardware	Recommendation
RAM	64GB
Technology	DDR3 or DDR4 ECC

Table 5: Intel® EE for Lustre® Software version compatibility matrix

Compute Node		LNet Router		Storage Server		Notes
Intel® OPA	2.4+	Intel® OPA/Mellanox ConnectX-3 or IB	2.4+	Mellanox ConnectX-3 or IB	2.4+	Compute nodes and LNet Routers on RHEL 7.1+; Storage Server on RHEL 6.7+
Intel® OPA	2.4+	Intel® OPA/Mellanox ConnectX-3 or IB	2.4+	Mellanox ConnectX-4	2.4+	Compute nodes and LNet Routers on RHEL 7.1+; Storage Server on RHEL 6.7+ due the support of Mellanox OFED 3.1
Mellanox ConnectX-3 or IB	2.4+	Mellanox ConnectX-3 or IB /Intel® OPA	2.4+	Intel® OPA	3.0+	Storage Servers and LNet Routers on RHEL 7.2 due the support of Intel® Fabric Suite for Intel® OPA
Mellanox ConnectX-4	2.4+	Mellanox ConnectX-3 or IB/Intel® OPA	2.4+	Intel® OPA	3.0+	Storage Servers and LNet Routers on RHEL 7.2 due the support of Intel® Fabric Suite for Intel® OPA

Table 6: Intel® Foundation Edition for Lustre® Software version compatibility matrix

Compute Node		LNet Router		Storage Server	
Intel® OPA	2.7.1	Intel® OPA/Mellanox ConnectX-3 or IB	2.7.1	Mellanox ConnectX-3 or IB	2.7.1
Intel® OPA	2.7.1	Intel® OPA/Mellanox ConnectX-3 or IB	2.7.1	Mellanox ConnectX-4	2.7.1
Mellanox ConnectX-3 or IB	2.7.1	Mellanox ConnectX-3 or IB /Intel® OPA	2.7.1	Intel® OPA	2.7.1
Mellanox ConnectX-4	2.7.1	Mellanox ConnectX-3 or IB/Intel® OPA	2.7.1	Intel® OPA	2.7.1

Table 7: Community edition (opensfs.org) compatibility matrix

Compute Node		LNet Router		Storage Server	
Intel® OPA	2.8	Intel® OPA/Mellanox ConnectX-3 or IB	2.8	Mellanox ConnectX-3 or IB	2.8
Intel® OPA	2.8	Intel® OPA/Mellanox ConnectX-3 or IB	2.8	Mellanox ConnectX-4	2.8
Mellanox ConnectX-3 or IB	2.8	Mellanox ConnectX-3 or IB /Intel® OPA	2.8	Intel® OPA	2.8
Mellanox ConnectX-4	2.8	Mellanox ConnectX-3 or IB/Intel® OPA	2.8	Intel® OPA	2.8