The Evolution of Lustre Networking at Cray

Chris Horn
Cray Inc.
Saint Paul, MN USA
hornc@cray.com

***Abstract*—Lustre Network (LNet) routers with more than one InfiniBand Host Channel Adapter (HCA) have been in use at Cray for some time. This type of LNet router configuration is necessary on Cray Supercomputers in order to extract maximum performance out of a single LNet router node. This paper provides a look at the state of the art in this dual-HCA router configuration. Topics include avoiding ARP flux with proper subnet configuration, flat vs. fine-grained routing, and configuration emplacement. We will also provide a look at how LNet will provide compatibility with InfiniBand HCAs requiring the latest mlx5 drivers, and what is needed to support a mix of mlx4 and mlx5 on the same fabric.**

***Keywords*-Lustre; LNet;**

I. Introduction

The goal in the design of Lustre Network (LNet) routers is to provide sufficient bandwidth to meet the capabilities of the back-end storage. Thus, as the performance of Lustre Object Storage Servers (OSS) has increased the design of LNet routers has had to adapt to meet new performance targets.

One way that LNet router design has evolved at Cray is in the adoption of multiple HCAs on a single LNet router node. Multiple HCAs can be used in multi-rail networking to alleviate issues of limited-bandwidth as well as to increase fault tolerance[3]. At Cray, multiple HCAs are employed to enable an LNet router node to realize the full bandwidth capabilities of the Aries High-Speed Network (HSN), and, as a result, meet the bandwidth requirements of back-end storage with fewer dedicated LNet routers.

This paper provides a look at how the design of LNet routers has evolved to include the use of multiple InfiniBand Host Channel adapters (IB HCAs). We'll discuss how LNet routers are configured by Cray to meet performance and resiliency goals, and we'll also provide a look at how LNet will provide compatibility with IB HCAs requiring the latest mlx5 drivers, and what is needed to support a mix of mlx4 and mlx5 on the same fabric.

II. LNet Overview

The exchange of requests and replies between hosts forms the basis of the Lustre protocol. One host will send a message containing a request to another and await a reply from that other host. The underlying network protocols used to exchange messages are abstracted away via the Lustre Network (LNet) software layer. LNet is largely independent from Lustre and is also used by Cray's Data Virtualization Service (DVS).

The LNet layer is itself network-type agnostic. It utilizes Lustre Network Drivers (LND) to interface with the driver for a specific network type. LNet supports Cray's Gemini and Aries HSNs with the gnilnd, as well as other networking technologies such as InfiniBand, via the o2iblnd, and ethernet, via the socklnd. A single Lustre Network can encompass multiple sub-networks through the use of LNet routers.

LNet router nodes in Cray systems provide store and forward services to bridge the Cray High Speed Network (HSN), where we have compute nodes and Lustre clients, and the IB fabric of Lustre servers. More generally, LNet routers can be used to bridge different network segments or other network technologies such as ethernet and Intel's OmniPath Architecture.

A Lustre Network containing routers is typically configured to be either flat, where all routers can forward requests to or from any Lustre server, or fine-grained, where some subset of the routers are configured to communicate with specific Lustre servers.

A flat routing configuration is the simplest to define, and its performance can be optimal at small scale. However, as the number of servers and routers increases performance can suffer dramatically. This is due to the increasing likelihood of taking non-optimal paths through the network. In particular, the latency introduced by traversing multiple Inter-Switch Links (ISLs) results in degraded performance[11].

Fine-grained routing (FGR) schemes were devised to solve the problem of traversing ISLs. Unlike flat routing, where every router can communicate with every server, in FGR groups of routers are defined to communicate with a particular group of servers. At Cray, the groups are defined such that the routers in a group have a direct connection to the same top-of-rack (TOR) IB switch that is used by the servers in the group. A fine-grained routing configuration can be much more complex to define, but it provides better performance at scale than flat routing[12].

III. Cray Sonexion Overview

The Cray Sonexion has long been composed of two basic building blocks: a single Metadata Management Unit (MMU) and one or more Scalable Storage Units (SSUs).

Another building block is available for the Sonexion 2000 in the Additional DNE Unit (ADU). The MMU consists of two Lustre Management Servers (MGS) and two Lustre MDSs along with 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 solutions provide 5 GB/s per SSU sustained, and 6 GB/s per SSU peak. The Sonexion-2000 provides 7.5 GB/s per SSU sustained, and 9 GB/s per SSU peak[4][5]. Ensuring sufficient network bandwidth to each OSS is a key requirement in the design of Lustre Networks.

IV. Network Speeds and Feeds

On an XC system an I/O blade has one Aries Network Interface Controller (NIC) that feeds an I/O module (IBB) with 17 GB/s. That IBB has two nodes (LNet routers), so each node can achieve 8.5 GB/s. Each LNet router node can have up to two dual-port IB HCAs. Each active HCA port (ib0 and ib2, for instance) needs to be assigned to a different LNet (cannot bond them to a single LNet), so a single LNet router node will service two different LNets. A single FDR (Fourteen Data Rate) IB HCA is capable of 5.5 GB/s of LNet traffic. Therefore, if busy, the two IB HCAs on a single LNet router would split the 8.5 GB/s coming into the node and achieve 4.25 GB/s per IB port.

V. Bandwidth Matching for Fine Grained Routing

The throughput capabilities of the back-end storage and the capabilities of the LNet routers allow us to come up with ratios of routers to servers that ensure we can provide sufficient bandwidth for FGR schemes. We need to ensure that there are sufficient links between Lustre servers and the supercomputer such that each configured LNet FGR group has enough bandwidth to sate the back-end storage in that LNet FGR group.

In addition, if we want to apply the same ratio of IB links to servers across all servers in a filesystem we need to ensure that number of servers in an FGR group is evenly divisible by the number of servers in each TOR switch, otherwise the FGR groups can span more than one TOR switch which defeats the purpose of FGR. Alternatively, FGR groups can be defined with different ratios.

Table I shows the respective capabilities. Since a single FDR IB link provides sufficient bandwidth for a single Sonexion-1600 or Sonexion-2000 OSS we could simply use a ratio of n IB links to n servers. However, in the case of the Sonexion-1600, this results in wasted bandwidth.

We can see from table I that 6 Sonexion-1600 OSSs are capable of delivering 18 GB/s peak I/O bandwidth. If we were to assign 6 single HCA LNet router links to service the 6 servers then the fabric would be capable of 33 GB/s. This is nearly double what the 6 OSSs are capable of. If IB links from LNet routers with dual HCAs were assigned we would only be providing 23% more bandwidth than what is needed.

Table I
Cray Sonexion and XC40 LNet Router Single IB Link Bandwidth Capabilities

	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
Cray XC40 LNet Router, single HCA	5.50	11.00	16.50	22.00	27.50	33.00
Cray XC40 LNet Router, dual HCA	4.20	8.40	12.60	16.80	21.00	25.20

Assigning two single HCA LNet router links, or three IB links from dual HCA routers, to every 4 Sonexion-1600 servers would be ideal from the perspective of minimizing any wasted bandwidth, however this would either result in FGR groups that span more than one TOR switch or in the definition of additional sub-optimal FGR groups containing just two servers. If we only consider FGR groups where the number of servers in each FGR group is evenly divisibly by the number of servers in each TOR switch. This constraint limits us to considering ratios with one, two, three or six servers.

With the above constraint in mind we can find suitable ratios:

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

VI. ARP Flux

The Address Resolution Protocol (ARP) protocol is used to map network layer addresses (e.g. an IPv4 address) to link-layer addresses (e.g. a MAC address). For example, when a host wants to send a message to another host over Ethernet it needs both the IP address and MAC address of the intended recipient. Once the sender has the receiver's IP address, through, say, a Domain Name Service (DNS) lookup, the sender will first check a local cached ARP table to see if it can find the MAC address for the receiver's IP. If a MAC address is found it can proceed with sending the message. If the MAC address is not found then it will broadcast an ARP "who-has" request. The host with the IP address contained in the ARP request will then respond with its MAC address and its IP address. The sender will cache the response into its ARP table and proceed with sending the message.

```
iosr60-3:~ # ip addr show dev ib0
6: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc pfifo_fast state UP group default qlen 256
    link/infiniband 80:00:00:48:fe:80:00:00:00:00:00:00:00:00:02:c9:03:00:f9:f1:61 brd
    inet 10.149.0.4/24 brd 10.149.0.255 scope global ib0
       valid_lft forever preferred_lft forever
    inet6 fe80::202:c903:f9:f161/64 scope link
       valid_lft forever preferred_lft forever
iosr60-3:~ # ip addr show dev ib2
8: ib2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc pfifo_fast state UP group default qlen 256
    link/infiniband 80:00:00:25:fe:80:00:00:00:00:00:00:7c:fe:90:03:00:62:ce:7a brd
    inet 10.149.0.5/24 brd 10.149.0.255 scope global ib2
       valid_lft forever preferred_lft forever
    inet6 fe80::7efe:9003:62:ce7a/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 1. Two IB interfaces on the iosr60-3 host.

When a host has multiple interfaces on the same subnet both interfaces may respond to ARP "who-has" requests. This results in non-deterministic population of the requesting host's ARP table. This issue has been termed ARP flux[9].

At Cray we've seen a variation of the ARP flux issue with dual IB HCAs and IPoIB. What we've observed is that only one interface will receive the ARP "who-has" requests, and that interface will respond to both the "who-has" requests that are intended for it and the requests intended for the other interface. Figures 1-5 illustrate this issue using the *rping* command to establish an RDMA connection and issue a single RDMA transfer between two hosts, iosr60-2 and iosr60-3.

In this example, iosr60-3 will act as the server, passively waiting for the connection request, and iosr60-2 will act as the client, actively initiating the connection and RDMA transfer. Figure 1 shows the IP and link-layer addresses, bolded, for two IB interfaces, on iosr60-3, each on a separate HCA[1]. Figure 2 shows the command used to invoke *rping* in server-mode on iosr60-3. Figure 3 shows the result of executing the *rping* command on iosr60-2. We can see that while we were able to resolve the address and route to the destination address the connection attempt failed with "error 8". We can look at the `enum ib_cm_rej_reason` defined in `rdma/ib_cm.h` to see that "error 8" is `IB_CM_REJ_INVALID_SERVICE_ID`. The reason we see this error is that the link-layer address for IPoIB interfaces contains the Queue Pair Number (QPN) and one of the GIDs of the port associated with the IPoIB interface, but, as shown in figure 4, the link-layer address for `10.149.0.5`, the ib2 interface on iosr60-3, in iosr60-2's ARP table is wrong. The link-layer address in the iosr60-2's ARP table is actually the link-layer address for `10.149.0.4`, the ib0 interface on iosr60-3. Finally, figure 5 shows output from the *tcpdump* command showing the ib0 interface on iosr60-3 responding to the "who-has" request from iosr60-2. Running *tcpdump* on the ib2 interface shows

[1]Please note we've removed some of the normal output from the *ip* command in order to better display the relevant information

```
iosr60-3:~ # rping -s -p 9999 -d -a 10.149.0.5
created cm_id 0x165bc10
rdma_bind_addr successful
rdma_listen
```

Figure 2. Starting rping in server mode on iosr60-3.

that the ib2 interface never receives the ARP "who-has" request.

To resolve this problem Cray recommends placing each active interface on a single host on a separate subnet. This ensures only the appropriate interface will receive and respond to ARP requests.

Subnet selection is necessarily made on a per-site basis, but we do recommend using a convention to simplify configuration. For example, if you have two subnets, say `10.150.0.x/16` and `10.151.0.x/16`, then placing every ib0 interface across all routers on the `10.150.0.x/16` subnet, and every ib2 interface across all routers on the `10.151.0.x/16` subnet can make it easier to spot mistakes in the IP configuration or otherwise validate that the correct configuration has been emplaced.

The "CLE XC System Administration" book details how to configure IPoIB on LNet router nodes. At the time of this writing the existing documentation does not reference the ARP flux issue, but the steps outlined to configure IPoIB can be easily modified to specify IP addresses on different subnets. Note, at the time of this writing this book has not yet been updated for the new configuration management paradigm introduced in CLE 6.0.

In order to maintain IP connectivity Lustre servers will need appropriate IP addresses on the subnets of routers they need to communicate with. A script can be utilized to add appropriate IP aliases to the ib0 interface of Lustre servers when the LNet module is loaded. An example script is included in figure 6. The script at figure 7 could then be used to remove the appropriate aliases when the LNet module is unloaded. Please note the IP networks shown in the script would need to be customized on a per-site basis.

```
iosr60-2:~ # rping -c -Vv -C 1 -d -p 9999 -a 10.149.0.5
port 9999
created cm_id 0x826c10
cma_event type RDMA_CM_EVENT_ADDR_RESOLVED cma_id 0x826c10 (parent)
cma_event type RDMA_CM_EVENT_ROUTE_RESOLVED cma_id 0x826c10 (parent)
rdma_resolve_addr - rdma_resolve_route successful
created pd 0x826f90
created channel 0x826fb0
created cq 0x826fd0
created qp 0x8273e0
rping_setup_buffers called on cb 0x823010
allocated & registered buffers...
cq_thread started.
cma_event type RDMA_CM_EVENT_REJECTED cma_id 0x826c10 (parent)
cma event RDMA_CM_EVENT_REJECTED, error 8
```

Figure 3. Instantiate RDMA ping-pong test from iosr60-2.

```
iosr60-2:~ # ip neigh show dev ib0 10.149.0.5
10.149.0.5 lladdr 80:00:00:48:fe:80:00:00:00:00:00:00:00:02:c9:03:00:f9:f1:61 STALE
iosr60-2:~ #
```

Figure 4. ARP Table of iosr60-2 showing incorrect link-layer address for 10.149.0.5.

```
iosr60-3:~ # tcpdump -enni ib0 arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ib0, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes
14:49:26.369468   B ethertype ARP (0x0806), length 72: Request who-has 10.149.0.5 tell 10.149.0.2, length 56
14:49:26.369508 Out ethertype ARP (0x0806), length 72: Reply 10.149.0.5 is-at
                    80:00:00:48:fe:80:00:00:00:00:00:00:00:02:c9:03:00:f9:f1:61, length 56
```

Figure 5. tcpdump output from iosr60-3 shows ib0 interface responding to ARP "who-has" request intended for ib2.

```
## Add IP aliases to ib0 making ib0:1
## when lnet module is loaded
/sbin/ip -o -4 a show ib0 | \
/usr/bin/awk '/inet/{
    s=$4;
    sub("10\\.156\\.8\\.", "10.156.12.", s);
    print "/sbin/ip address add dev ib0 label ib0:1", s};' | /bin/sh
/sbin/modprobe --ignore-install lnet $CMDLINE_OPTS
```

Figure 6. Sample script for adding IP aliases.

VII. LNet Configuration

Cray provides the *clcvt* command line utility to ease the configuration of Lustre Networking. This tool takes human-readable input files and generates the, sometimes complex, ip2nets and routes entries needed to configure LNet. Typically the ip2nets and routes entries generated with *clcvt* are stored in `ip2nets.dat` and `routes.dat` files, respectively. These files are then referenced in a modprobe configuration file on appropriate nodes. See figures 8 and 9 for sample output generated by clcvt.

In versions of CLE prior to CLE 6.0 the typical procedure was to create an "lnet" node class in the sharedroot, use the *xtopview* command on the boot node to access the sharedroot for the lnet node class, and modify the a `modprobe.conf.local` file there to specify the appropriate ip2nets and routes lnet module options.

Starting with CLE 6.0 this type of system configuration is performed on the System Management Workstation (SMW) via the Image Management and Provisioning System (IMPS) and its *cfgset* command.

VIII. LNet and MLX5

There is some work needed in Lustre to robustly support IB HCAs requiring MLX5 drivers. Some of the work has been completed for the community Lustre 2.8 release, and some work is targeted for the community Lustre 2.9 release. This section outlines some of the issues.

A. Memory Management

IB clients must register memory associated with a data transfer prior to that transfer taking place. Memory locations that have been registered are referred to as Memory Regions (MR). There are also APIs that allow Fast Memory Registration (FMR). FMR allows memory regions to be re-

```
## Remove all ib0:1 aliases when lnet module is unloaded
/sbin/modprobe -r --ignore-remove lnet &&
/sbin/ip -o -4 a show label ib0:1 | \
/usr/bin/awk '{print "/sbin/ip addr del dev ib0 label ib0:1", $4}' | /bin/sh
```

Figure 7. Sample script for removing IP aliases.

```
o2ib4 10.100.[104,105].*
o2ib4000(ib0) 10.100.104.[4,254]
o2ib4000(ib0) 10.100.105.0
o2ib4001(ib0:1) 10.101.104.[5,255]
o2ib4001(ib0:1) 10.101.105.1
o2ib4002(ib0) 10.100.104.[6,8,10,12,14,16,18,20]
o2ib4003(ib0:1) 10.101.104.[7,9,11,13,15,17,19,21]
```

Figure 8. Sample ip2nets entries generated by clcvt to be placed in a ip2nets-loading file

```
o2ib4000 1 [2,262,642,2241]@gni1
o2ib4001 1 [2,262,642,2241]@gni1
o2ib4002 1 [70,261,710,845,1218,1413,1673,2054]@gni1
o2ib4003 1 [70,261,710,845,1218,1413,1673,2054]@gni1
o2ib4004 1 [130,322,578,906,1286,1474,1734,2053]@gni1
o2ib4005 1 [130,322,578,906,1286,1474,1734,2053]@gni1
o2ib4006 1 [6,321,577,905,1285,1473,1794,1930]@gni1
o2ib4007 1 [6,321,577,905,1285,1473,1794,1930]@gni1
o2ib4008 1 [66,386,646,966,1162,1538,1862,1929]@gni1
o2ib4009 1 [66,386,646,966,1162,1538,1862,1929]@gni1
```

Figure 9. Sample routes entries generated by clcvt to be placed in a routes-loading file

used which can reduce the overhead incurred through normal memory registration/deregistration.

FMR is enabled in the o2iblnd by setting the `map_on_demand` kernel module parameter to `0 < map_on_demand <= 256`. However, FMR is deprecated, and FMR is not supported in the mlx5 driver. Support for the IB Base Memory Management Extensions (BMME)[2][13] was made available in Linux kernel 2.6.27. This support includes APIs for creating "fast register memory regions"[13]. Support for these APIs was added to the o2iblnd in LU-7181[1]. At the time of this writing the current o2iblnd code prefers FMR when the HCA supports it. It will use the IB BMME APIs when an HCA supports them, and does not support FMR. The o2iblnd will place an entry in the kernel log announcing which memory registration is going to be used. Figure 10 shows an example of these log messages.

B. Mixed Fabric

The Lustre network drivers have historically only supported a single set of settings for each interface type on a single node. i.e. one set of settings for all gni interfaces on a node, one set of settings for all o2ib interfaces on a node, etc. Different types of HCAs may need different settings to work optimally, so this limitation can lead to situations where we need to compromise the performance of an interface if there are different types of HCAs in a single node.

```
LNet: Using FMR for registration
LNet: Added LNI 10.149.0.3@o2ib1000 [8/1024/0/30]
LNet: Using FastReg for registration
LNet: Added LNI 10.149.1.3@o2ib2000 [8/1024/0/30]
```

Figure 10. Log messages indicating memory registration API support.

For example, as mentioned in the previous section, HCAs requiring the mlx5 drivers cannot use FMR. However, some HCAs, such as Intel OPA[8], gain increased performance from FMR. Thus, with the limitation of having a single set of settings for all o2ib interfaces, one could not optimize a node, say an LNet router, that had both an OPA HCA and an IB HCA requiring the mlx5 drivers. This limitation has recently been addressed by the work in LU-7101[8].

With the patch for LU-7101 it is now possible to set certain o2iblnd module parameters on a per interface basis. The supported parameters are `map_on_demand`, `peer_credits`, `peercredits_hiw`, and `concurrent_sends`. Also supported are the FMR tuning parameters `fmr_pool_size`, `fmr_flush_trigger`, and `fmr_cache`.

With this new functionality we're able to configure `0 < map_on_demand <= 256` for OPA interfaces and also configure `map_on_demand = 0` for mlx5 interfaces. The configuration is emplaced using the Dynamic LNet Configuration feature first introduced in Lustre 2.7[10]. The *lnetctl* command can be used to configure interfaces at runtime, however it doesn't currently support setting all of the per-interface parameters. To configure the per-interface parameters one must use the *lnetctl* command's ability to input configuration data in the YAML format. Figure 11 shows configuration info in YAML syntax defining two o2ib interfaces with different parameters, and figure 12 shows a sequence of commands to import the YAML configuration data.

With the ability to set per-interface parameters there is one additional issue that must be addressed. The o2iblnd has long had the requirement that all peers on an o2ib network have identical values for `peer_credits` and `map_on_demand`. Since it is necessary to define different `map_on_demand` parameters for certain types of HCAs the Lustre community has needed to address this historical shortcoming. That work was accomplished as part of LU-3322[7].

In the o2iblnd there is an active and passive side to connection requests. The active side initiates the connection request, and the passive side receives and responds

```
net:
    - net: o2ib1
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 8
              map_on_demand: 256
              concurrent_sends: 32
              fmr_pool_size: 1280
              fmr_flush_trigger: 1024
              fmr_cache: 1
      tunables:
          peer_timeout: 100
          peer_credits: 16
          peer_buffer_credits: 0
          credits: 2560
    - net: o2ib2
      interfaces:
          0: ib2
          lnd tunables:
              peercredits_hiw: 8
              map_on_demand: 0
              concurrent_sends: 16
      tunables:
          peer_timeout: 180
          peer_credits: 16
          credits: 2560
```

Figure 11. Sample YAML syntax defining two o2ib interfaces with different parameters.

```
# modprobe lnet
# lnetctl lnet configure
# lnetctl import < /tmp/lnet_conf.yaml
```

Figure 12. Sequence of commands to import LNet configuration in YAML format.

to the connection request. Part of the data contained in the connection request are the values of the active side's `peer_credits` and `map_on_demand`. If the values in the connection request did not match the passive's values then the passive side would reject the connection request. A typical log message for such a connection rejection is shown in figure 13.

With the change from LU-3322 the passive side can generally accept connections where the desired `peer_credits` are less than or equal to the `peer_credits` defined for the passive peer. If the desired `peer_credits` are larger than the `peer_credits` defined for the passive peer then the connection request will still be rejected, but the rejection message will contain additional information about what value of `peer_credits` the passive side can accept. This allows the active side the opportunity to issue a new connection request with a lower `peer_credits` value[2]

Figure 14 shows a message logged by the passive side of a connection request when the desired credits is greater than the credits available.

IX. Future Work

We have plans to issue a tech note on the topic of ARP flux to better educate our users and site administrators on the specifics of the ARP flux issue.

Lustre 2.7 introduced the Dynamic LNet Configuration feature. Cray is currently investigating how we can best utilize this new feature to further enhance our configuration and administrative capabilities. More information on Dynamic LNet Configuration can be found at[10].

Multi-rail LNet is a new feature targeted for the Lustre 2.10 release. This feature should provide much more robust support for utilizing multiple interfaces to communicate between two nodes. More information on this feature is available at[6].

X. Conclusion

We've shown how the design of Cray's LNet routers is influenced by the capabilities of back-end storage. We've also shown how the design and configuration of LNet networks has evolved to include the use of multiple HCAs in our LNet routers. The dual-HCA configuration has allowed us to extract additional bandwidth out of a single LNet router node at the cost of configuration complexity.

Acknowledgment

We would like to acknowledge the hard work of Cray support staff who were instrumental in debugging the ARP flux issue. We would like to acknowledge the work done by Oak Ridge in their research on Fine-grained routing.

References

[1] Lu-7181. https://jira.hpdd.intel.com/browse/LU-7181, 2016.

[2] InfiniBand Trade Association. Infiniband architecture specification volume 1. http://www.infinibandta.org/index.php, 2015.

[3] Salvador Coll, Eitan Frachtenberg, Fabrizio Petrini, Adolfy Hoisie, and Leonid Gurvits. Using multirail networks in high-performance clusters. In *Proceedings. 2001 IEEE International Conference on Cluster Computing*. IEEE, 2001.

[4] Hussein N. El-Harake and Colin McMurtrie. Evaluation of the cray sonexion 2000 storage system. Technical report, CSCS – Swiss National Supercomputing Centre, 2014.

[5] Cray Inc. Cray sonexion specifications. http://www.cray.com/sites/default/files/resources/cray_sonexion_specification

[2] A note here for those interested in reviewing the code. The `struct kib_connparams` contains the actual values sent back and forth in the connect messages. The variables of interest are `ibcp_queue_depth` (peer_credits), `ibcp_max_frags` (map_on_demand), and `ibcp_max_msg_size` (IBLND_MSG_SIZE).

```
LNetError: 4:0:(o2iblnd_cb.c:2301:kiblnd_passive_connect()) Can't accept 10.149.1.14@o2ib: incompatible queue depth 63 (126 wanted)
```

Figure 13. Log message from incompatible peer_credits rejected connection request.

```
Can't accept conn from 10.0.51.1@o2ib, queue depth too large: 128 (<=8 wanted)
```

Figure 14. Log message from incompatiable peer_credits rejected connection request. Shows desired number of peer_credits ¡= 8.

[6] Intel and SGI. Multi-rail lnet. http://wiki.lustre.org/Multi-Rail_LNet, April 2016.

[7] jira.hpdd.intel.com. Lu-3322. https://jira.hpdd.intel.com/browse/LU-3322.

[8] jira.hpdd.intel.com. Lu-7101. https://jira.hpdd.intel.com/browse/LU-7101.

[9] linux ip.net. 2.1. address resolution protocol (arp). http://linux-ip.net/html/ether-arp.html, 2016.

[10] lustre.org. http://lustre.org/documentation/, April 2016.

[11] Galen M. Shipman, David A. Dillow, Sarp Oral, Feiyi Wang, Douglas Fuller, Jason Hill, and Zhe Zhang. Lessons learned in deploying the world's largest scale lustre file system. In *Proceedings of the 52nd Cray User Group Conference*, 2010.

[12] Mark Swan and Nic Henke. Cray's implementation of lnet fine grained routing. In *Proceedings of the 55th Cray User Group Conference*, 2013.

[13] Bob Woodruff, Roland Dreier, Sean Hefty, and Hal Rosenstock. Introduction to the infiniband core software. https://www.kernel.org/doc/ols/2005/ols2005v2-pages-279-290.pdf, 2005.