Cray's Implementation of LNET Fine Grained Routing

Mark Swan
Performance Group
Cray Inc.
Saint Paul, Minnesota, USA
mswan@cray.com

Nic Henke
ClusterStor
Xyratex International Inc.
Saint Paul, Minnesota, USA
nic_henke@xyratex.com

Abstract—**External Lustre file systems, as deployed on Cray XE and XC30 systems, must coordinate the connectivity of compute and service nodes on the Cray high speed network, LNET router nodes that move data between the high speed network and the external Infiniband network, and meta-data storage servers (MDS) and object storage servers (OSS) on the Infiniband network. While simple configuration schemes exist for getting the file system mounted and functional, much more complicated schemes must sometimes be used to obtain the greatest performance from that file system.**

Work by the Oak Ridge National Laboratory (ORNL) introduced the concept of Fine Grained Routing (FGR), where optimal pathways are used to transport data between compute nodes and OSSs. This scheme minimizes the number of chips, backplane hops, and Inter-Switch-Links along that path all of which can function as bottlenecks and have a negative effect on performance. Understanding the concepts of how to apply FGR is relatively straightforward. Even implementing FGR on a small external Lustre file system is manageable by a human. However, very large installations that employ FGR can contain thousands of compute and service nodes, hundreds of LNET router nodes, and hundreds of Lustre servers. A human can easily be lost in a sea of IP addresses and NID numbers.

Cray has developed tools to aid in configuring FGR installations of all sizes, doing what computers do very well: converting human readable and understandable configuration information into a sea of numbers. The tool also performs live validation of the resulting configuration, providing a straightforward diagnosis of the inevitable mistakes during installation and failing hardware during the lifecycle of the machine.

This paper will provide a brief history of FGR, why Cray uses FGR, tools Cray has developed to aid in FGR configurations, analysis of FGR schemes, performance characteristics, and some interesting visual analysis.

I. LNET Primer

A. Background

LNET, by definition, is the Lustre Networking layer. It is responsible for transporting data between Lustre clients and Lustre servers. LNET router nodes, when used in Cray systems, are responsible for routing Lustre messages between the Cray High Speed Network (HSN) (where the compute node Lustre clients exist) and the Infiniband (IB) network (where the Lustre servers exist). LNET router nodes are then a pool of available resources that provide data connectivity between the two different physical networks.

B. Flat LNET

In a traditional implementation of an external Lustre file system, all LNET router nodes have connectivity to every Lustre server. Therefore, when a compute node needs to communicate with a server, the LNET layer simply chooses an LNET router to use from the entire pool based on a simple round-robin based algorithm. The same type of choice is made when a server needs to communicate with a compute node. We have been using the term "flat LNET" to describe this generic approach since all LNET routers have equal priority for each LNET message.

Fig. 1 is a simple depiction of the concept of a "flat LNET". There are many LNET router nodes that can transport data between many compute nodes and many file system servers.

Fig. 2 is a simple depiction of one possible path that a compute node might use when communicating with an OSS. In this case, the router node's IB connection is in the same IB switch as the OSS and, therefore, has the least amount of latency.

Fig. 3 is a simple depiction of another possible path that a compute node might use when communicating with an OSS. In this case, the router node's IB connection is not on the same IB switch as the OSS and, therefore, has a much greater amount of latency and will be restricted to the available bandwidth on the small number of links between switches. In fact, as the system size grows, this suboptimal path will be chosen more often than not which prevents the applications from realizing the full performance available.

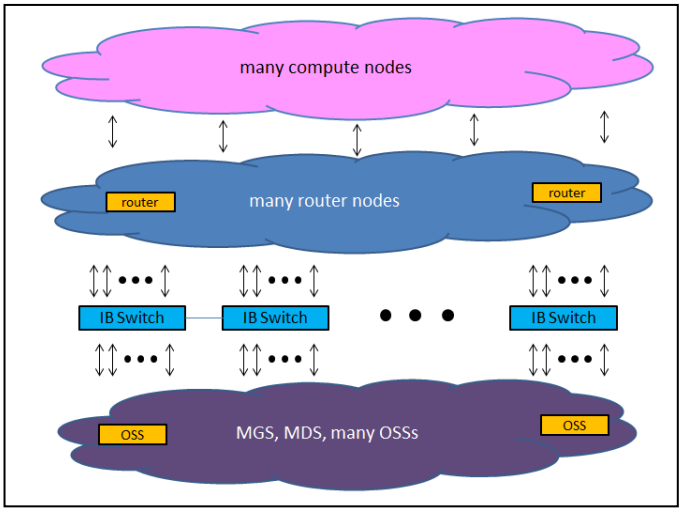

Fig. 1. Flat LNET layout

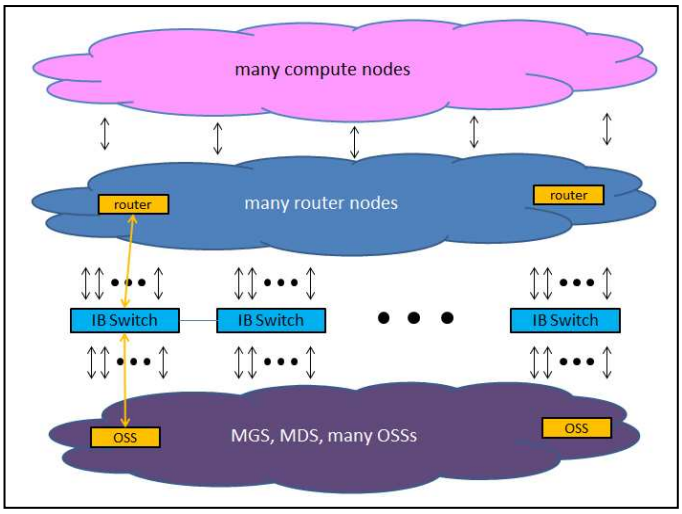

Fig. 2. Flat LNET optimal communication path

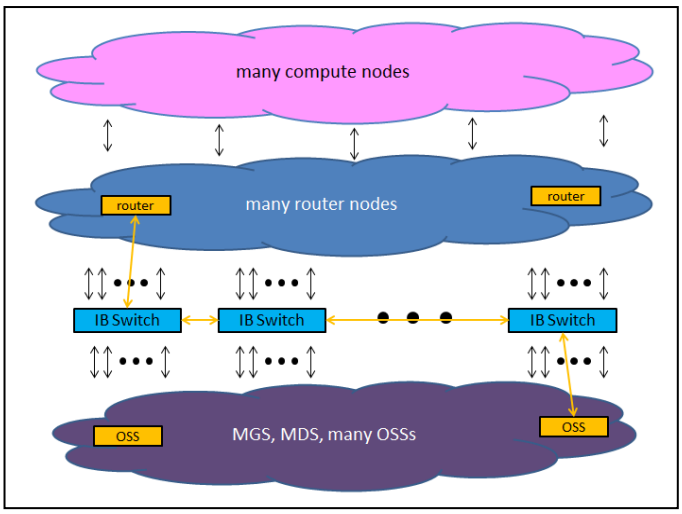

Fig. 3. Flat LNET nonoptimal communication path

C. *LNET Fine Grained Routing (FGR)*

This added [and unpredictable] latency as communication hops through the IB fabric was the focus of ORNL's research papers on LNET FGR [1][2]. The goal of FGR is to reduce this latency and remove the unpredictability of data movement between compute nodes and Lustre servers. Rather than every LNET router node being able to communicate with every Lustre server (as in a flat LNET), FGR defines a subset of LNET router nodes that can communicate only with a subset of Lustre servers.

Fig. 4 is a simple depiction of LNET FGR. The key feature to notice is that when a compute node needs to communicate with the yellow OSS, it only has a narrow list of yellow LNET router nodes from which to choose. Likewise when needing to communicate with the green OSS. The yellow LNET router node does not even know how to communicate with the green OSS. Therefore, communication from compute node to OSS will always take the lowest latency path (assuming correct cabling, of course).

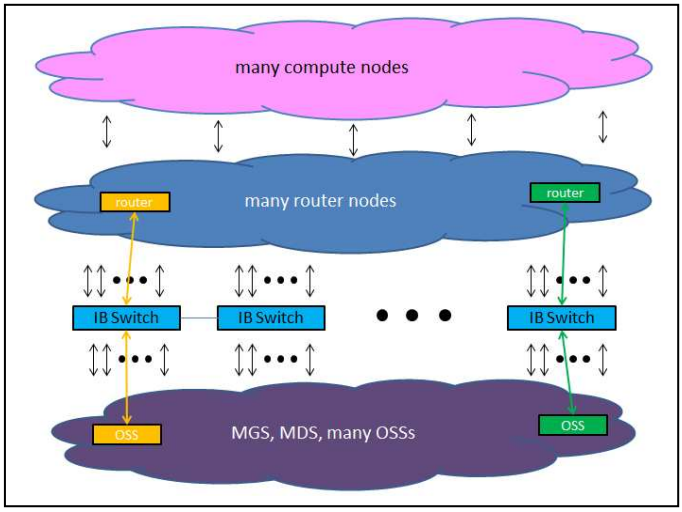

Fig. 4. LNET Fine Grained Routing

Figure 5 depicts a LNET FGR implementation for a Cray system connected to small Cray Sonexion-1600 with 2 Scalable Storage Units (SSUs). In this implementation, we form the following LNET groups:

- MGS and MDS (purple)
- OSS0 and OSS2 (green) which are connected to the "even" IB switch
- OSS1 and OSS3 (red) which are connected to the "odd" IB switch

The Cray system has a limited supply of LNET router nodes and we have assigned them as follows:

- All four will handle metadata traffic.
- Two will handle traffic to and from the "even" OSSs
- Two will handle traffic to and from the "odd" OSSs

The entries in the LNET router node's "/etc/modprobe.conf.local" file might look like this:

```
options lnet ip2nets="gni0 10.128.*.*;\
                      o2ib1000 10.10.100.[60,61,62,63];\
                      o2ib1002 10.10.100.[60,61];\
                      o2ib1003 10.10.100.[62,63]"
options lnet routes="o2ib1000 1 [68,69,90,91]@gni0;\
                      o2ib1002 1 [68,69]@gni0;\
                      o2ib1003 1 [90,91]@gni0"
```

As you can see, the entries are easy to understand and easy to get correct when entering the values with a text editor. However, let's look at a more complicated implementation.

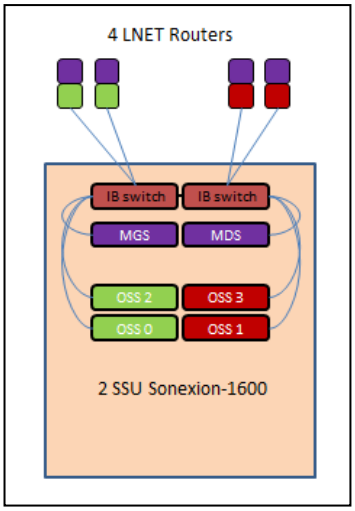

Fig. 5. LNET FGR for a small Cray Sonexion-1600

Figure 6 is a depiction of LNET FGR as implemented for the "home" and "projects" file systems on the Blue Waters system at NCSA. There are:

- 18 SSUs across 3 racks of Cray Sonexion-1600
- MGS and MDS in one LNET group
- Sets of 3 OSSs form an LNET group (12 LNET groups)
- 50 LNET router nodes
  - Two for the MGS/MDS LNET group
  - Four for every OSS LNET group

As you can imagine, entering the contents of the LNET router node's "/etc/modprobe.conf.local" file involves dozens of IB IP addresses and NID numbers. For the Blue Waters "scratch" file system, this example becomes 10 times more complex.

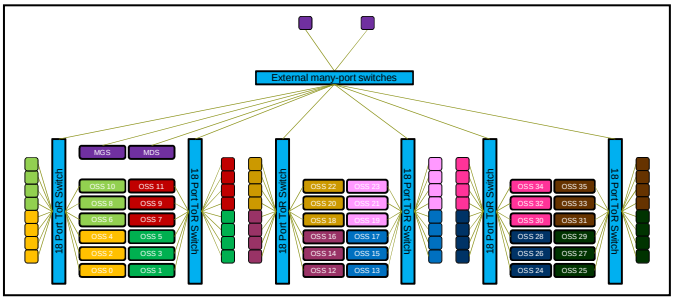

Fig. 6. LNET FGR representation for 18 SSU Cray Sonexion

D. Bandwidth matching

How does one decide how many LNET router nodes and OSSs should be in each LNET group? The answer is "bandwidth matching".

Table 1 shows the maximum bandwidth capabilities of the Cray Sonexion-1600 OSS and the various Cray LNET router nodes.

Either because of luck or good, clean living, a full Cray Sonexion-1600 rack consists of 6 SSUs and Cray XE LNET router nodes are four per blade. Therefore, assigning four LNET router nodes for every three Sonexion-1600 OSSs gives us slightly more LNET bandwidth and the peak bandwidth needs of those OSSs. In general, it is simply a matter of assigning enough LNET router nodes to a group of OSSs such that you have more LNET bandwidth than the peak amount of disk I/O for those OSSs.

TABLE I. BANDWIDTH CAPABILITIES IN GIGABYTES PER SECOND

	1	2	3	4	5	6
Sonexion-1600 OSS	3.0	6.0	9.0	12.0	15.0	18.0
Cray XE LNET Router	2.6	5.2	7.8	10.4	13.0	15.6
Cray XC30 LNET Router, single HCA	5.5	11.0	16.5	22.0	27.5	33.0
Cray XC30 LNET Router, dual HCA	4.2	8.4	12.6	16.8	21.0	25.2

II. THE COMPLEXITY ISSUE

> "Obscurity is the realm of error" - Luc de Clapiers

As we can see from the description of Figure 6, as the numbers and sizes of the file systems and clients grow, manually configuring a robust FGR implementation can tax even the most patient system administrator. There can be dozens to hundreds of LNET router NID numbers, LNET router IB IP addresses, and Lustre server IB IP addresses, all combined with hundreds to thousands of network cables.

Most often, systems are described in terms of hostname or physical location, not the IP address or NIDs that LNET requires. The need to translate from this upper level nomenclature to the addresses needed in LNET is a tricky and time consuming process. Moreover, the nature of IP addresses and NID numbers is that they are sequences of similar integer numbers provides ample room for blurred vision and mistaken typing to inject errors that are easily missed by the human eye.

Once an error is introduced into the configuration, large configurations are equally complicated to analyze and debug.

- Must cross-reference NID values and IB IP addresses with hostnames.
- Must trace the endpoints of every IB cable in every IB switch.
- Must visually parse output from utilities such as "ibnetdiscover".
- Must cross-check what was intended with what is actual.

All of this discovery must be done within multiple sets of disjoint text files that often reach to kilobyte sizes, and are all of course formatted and described differently. Often the tools used to debug physical network hardware talk in realms of Globally Unique Identifiers (GUIDs) or World Wide Names (WWNs), providing another step of required transmogrification before becoming usable to resolve issues best described in the normal naming of hosts and nodes. Once all of these opportunities for error injection are combined, it is difficult to envision even moderately sized FGR LNET configurations being implemented without significant difficulty.

III. COMPLEXITY REDUCTION

By applying FGR to an LNET configuration we add a nontrivial amount of difficulty. In order to productize these schemes and make them usable without domain expertise, we must find ways to reduce the complexity to the point where a human can look at a simple description of the system and understand it while utilizing software to remove human error. As we looked at how to implement FGR on the Blue Waters system, we saw the necessary building blocks in the Cray Sonexion.

As you can see in figure 6, each Cray Sonexion-1600 rack is composed of the following elements.

- There are 6 SSUs per rack.

- An SSU is made up of two OSSs.
    - One on the left side.
    - One on the right side.
- All OSSs on the left side are given even numbered node names.
- All OSSs on the right side are given odd numbered node names.
- There is a separate IB switch for the left and right sides.
- All OSSs on the left are plugged into the IB switch on the left.
- All OSSs on the right are plugged into the IB switch on the right.

Furthermore, due to bandwidth matching, we can create other building blocks using LNET router groups.

- The MGS and MDS are placed in their own LNET group.
- OSS LNET groups are formed on the left and right sides.
- Two groups of three OSSs on the left.
- Two groups of three OSSs on the right.
- Four LNET routers assigned to each group.

To complete our consistent approach to LNET groups, we employed a simple and consistent naming scheme.

- The MGS and MDS are in a group with suffix “000”.
- The OSS LNET groups are given suffixes “002”, “003”, and so on.
- Even numbered LNET groups contain even numbered OSS hostnames.
- Odd numbered LNET groups contain odd numbered OSS hostnames.

Referring back to figure 6 once again, you will notice that there are two LNET groups associated with every Cray Sonexion top-of-rack (TOR) IB switch. Therefore, it is easy to see that the LNET router nodes that are assigned to one of those LNET groups has access to the other LNET group’s OSSs while still avoiding any inter-switch-links (ISLs). In addition to the high availability features of the SSU (i.e., Lustre level failover between the two OSSs), this IB connectivity feature allows us to implement another level of resiliency. By configuring the eight LNET routers (that are plugged into the same IB TOR switch) to be each other’s secondary paths, we could lose up to seven of the eight LNET routers and still have connectivity to the LNET groups on that side of the rack.

All of these independently operating building blocks not only create building blocks of performance, but also create building blocks of problem isolation. Failed IB cables, failed LNET routers, failed IB switch ports, failed XIO blades in the mainframe, and even failed OSSs in the file system merely create local (not global) issues.

In a flat LNET implementation, for instance, losing a LNET router connection would mean that there is some percentage of bandwidth no longer available to the entire file system. I/O jobs that measure performance would potentially notice a slight decrease in overall performance but there would be no obvious indication of where the problem exists. Any overall LNET bandwidth decrease would likewise decrease each OSSs perceived performance. However, an implementation of LNET FGR, the decrease of LNET bandwidth caused by the loss of a LNET router is confined to the LNET group to which it belongs. No other LNET group would be affected. Tools that measure I/O performance would clearly see a performance decrease in that LNET group alone and this greatly aids in problem isolation.

The LNET naming scheme also tricks Lustre into helping us through its logging abilities. When connectivity problems occur, Lustre will output a volume of information, including the LNET NID of the problematic Lustre client or server. With the addition of distinct LNET number for each group, this information can be quickly parsed out to find the small group of hardware that needs further investigation.

Figure 8 shows an interpretation of per-OSS write and read completion times. OSSs 6, 8, and 10 are part of the same LNET group and we can see that their read times are all abnormally high. This was due to the loss of one of the four LNET router nodes for that group.

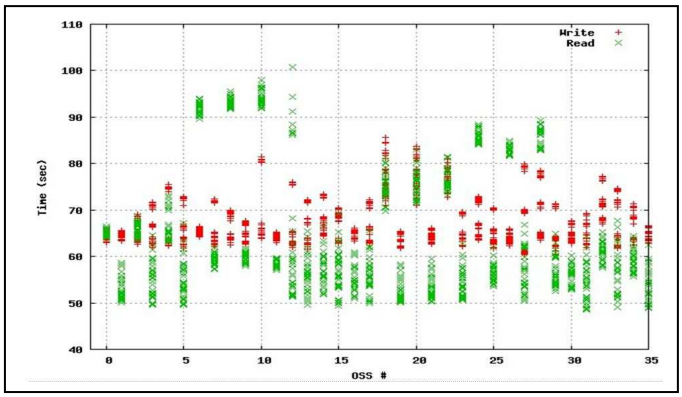

Fig. 7. Localized performance effects

A. *Cray LNET Configuration and Validation Tool (CLCVT)*

When we set out to design CLCVT, we had the following goals in mind.

- Simple and descriptive input file format
- Understands Cray Sonexion IB switch configuration
- Handles multiple Cray clients connected to multiple Sonexion file systems
- Generates secondary routes for LNET groups
- Generates appropriate "lnet.conf" file contents for each system
- Generates a cable table
- Performs live validation of IB connectivity
- Performs live validation of LNET group membership
- Performs live validation of LNET destinations (i.e., cable check)

As an example, the following is the CLCVT input file for a Cray XE connected to a 6 SSU Cray Sonexion-1600.

```
[info]
clustername = snx11029n
SSU_count = 6
clients = hera


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


o2ib6000: c0-0c2s2n0, c0-0c2s2n2 ; MGS and MDS
o2ib6002: c1-0c0s7n0, c1-0c0s7n1, c1-0c0s7n2, c1-0c0s7n3 ; OSSs 2/4/6
o2ib6003: c3-0c1s5n0, c3-0c1s5n1, c3-0c1s5n2, c3-0c1s5n3 ; OSSs 3/5/7
o2ib6004: c3-0c1s0n0, c3-0c1s0n1, c3-0c1s0n2, c3-0c1s0n3 ; OSSs 8/10/12
o2ib6005: c3-0c2s4n0, c3-0c2s4n1, c3-0c2s4n2, c3-0c2s4n3 ; OSSs 9/11/13


[snx11029n]
lnet_network_wildcard = o2ib6:10.10.100.*


o2ib6000: snx11029n002, snx11029n003 ; MGS and MDS
o2ib6002: snx11029n004, snx11029n006, snx11029n008 ; OSSs 2/4/6
o2ib6003: snx11029n005, snx11029n007, snx11029n009 ; OSSs 3/5/7
o2ib6004: snx11029n010, snx11029n012, snx11029n014 ; OSSs 8/10/12
o2ib6005: snx11029n011, snx11029n013, snx11029n015 ; OSSs 9/11/13
```

One of the inherent beauties of this configuration file format is that the majority of its contents can actually be software generated by simply knowing some key aspects of the file system, such as:

- Number of SSUs
- File system "cluster name"
- Intended "o2ib" base name

The resulting "/etc/modprobe.conf.local" file information that is generated for LNET router nodes.

```
options lnet ip2nets="gni1 10.128.*.*;\
                o2ib6000
10.10.100.[101,102,103,104,105,106,107,108,109,110,111,112,113,114,11
5,116,117,118];\
                o2ib6002 10.10.100.[103,104,105,106,107,108,109,110];\
                o2ib6003 10.10.100.[111,112,113,114,115,116,117,118];\
                o2ib6004 10.10.100.[103,104,105,106,107,108,109,110];\
                o2ib6005 10.10.100.[111,112,113,114,115,116,117,118]"
options lnet routes="o2ib6000 1 [68,90]@gni1;\
                o2ib6002 1 [750,751,752,753]@gni1;\
                o2ib6003 1 [618,619,628,629]@gni1;\
                o2ib6004 1 [608,609,638,639]@gni1;\
                o2ib6005 1 [648,649,662,663]@gni1;\
                o2ib6000 2
[608,609,618,619,628,629,638,639,648,649,662,663,750,751,752,753]@gn
i1;\
                o2ib6002 2 [608,609,638,639]@gni1;\
                o2ib6003 2 [648,649,662,663]@gni1;\
                o2ib6004 2 [750,751,752,753]@gni1;\
                o2ib6005 2 [618,619,628,629]@gni1"
```

B. *Other Live Validation*

The LNET Selftest (LST) tool can be used to perform LNET bandwidth testing of the resulting LNET groups. LST allows testing of each OSS independently, all OSSs simultaneously, each LNET group independently, and all LNET groups simultaneously. Due to the building block nature of FGR, this is yet another tool to help easily spot performance issues that are localized to a LNET group.

Figure 7 shows the LST results when running two LNET groups simultaneously on a Cray XE connected to a 2 SSU Sonexion-1600.

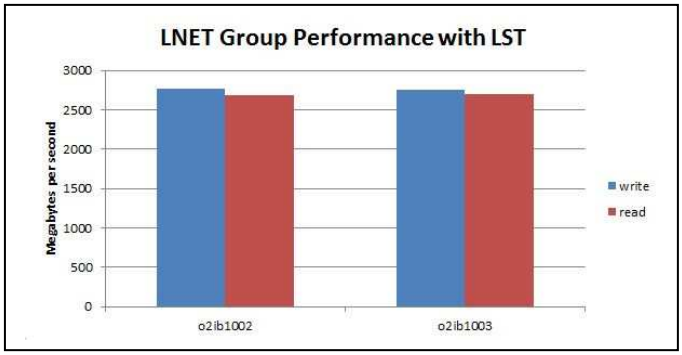

Fig. 8. LNET Selftest Results

IV. Future Work

A. Helping an application understand where LNET routers are

There can be significant I/O performance improvements by placing certain ranks of an application (e.g., those performing I/O) near the LNET routers they will be using. This could potentially help take advantage of the six links into and out of each Gemini associated with the LNET routers. Areas of exploration are:

- Providing FGR information to the mainframe.
- Which OSTs belong to which LNET groups
- Which LNET routers service which LNET groups
- Providing directives to the application placement tool for how to place ranks in the system.

B. CLCVT becoming topology aware

Clearly, not every customer has the same IB connectivity as imposed by an implementation such as that in Figure 6. The use of large external IB switches is not detected and could be handled more gracefully. In general, making CLCVT aware of the IB topology is desirable.

C. CLCVT validation on the server side

In the same manner that CLCVT performs live validation of LNET group membership and IB connectivity on the Cray mainframe, it could also validate the Luster server side.

D. CLCVT understanding other external Lustre implementations

CLCVT contains built-in knowledge of the architectural nuances of the Cray Sonexion-1600. Other external Lustre implementations, such as esFS, will have their own unique characteristics. While some support for alternative implementations exist in CLCVT, including flat LNET, enhancements to the tool would make these alternatives easier to configure and validate.

V. Conclusion

For a large LNET configuration, there is a fixed amount of complexity; fixed number of LNET routers, OSSs, IB cables, switches, etc. A flat LNET configuration is easy to configure but leaves the difficult tasks of optimization and debugging to run-time. LNET FGR, on the other hand, increases the level of difficulty related to how all these components are configured and used but greatly reduces the run-time challenges associated with debugging.

CLCVT, in concert with LNET FGR, gives us the best of both worlds. Ease of configuration and with all the benefits of reduced run-time complexity.

VI. References

[1] D. Dillow, J. Hill, D. Leverman, D. Maxwell, R. Miller, S. Oral, G. Shipman, J. Simmons, and F. Wang, “Lessons Learned in Deploying the World’s Largest Scale Lustre File System,” Proceedings of the Cray User Group Conference, May 2010.

[2] G. Shipman, D. Dillow, S. Oral, and F. Wang, “The Spider Center Wide File System; From Concept to Reality,” Proceedings of the Cray User Group Conference, May 2009.