I/O Router Placement and Fine-Grained Routing on Titan to Support Spider II

Matt Ezell, Sarp Oral, Feiyi Wang, Devesh Tiwari,
Don Maxwell, Dustin Leverman, and Jason Hill
Oak Ridge National Laboratory; Oak Ridge, TN
{*ezellma,oralhs,fwang2,tiwari,maxwellde,leverman,hilljj*}*@ornl.gov*

David Dillow*
dave@thedillows.org

I. Abstract

The Oak Ridge Leadership Computing Facility (OLCF) introduced the concept of Fine-Grained Routing in 2008 to improve I/O performance between the Jaguar supercomputer and Spider, OLCF's center-wide Lustre file system. Fine-grained routing organizes I/O paths to minimize congestion. Jaguar has since been upgraded to Titan, providing more than a ten-fold improvement in peak performance. To support the center's increased computational capacity and I/O demand, the Spider file system has been replaced with Spider II. Building on the lessons learned from Spider, an improved method for placing LNET routers was developed and implemented for Spider II. The fine-grained routing scripts and configuration have been updated to provide additional optimizations and better match the system setup. This paper presents a brief history of fine-grained routing at OLCF, an introduction to the architectures of Titan and Spider II, methods for placing routers in Titan, and details about the fine-grained routing configuration.

***Keywords*-Lustre; Titan; Spider II; Fine-Grained Routing; Router Placement; I/O; ORNL; OLCF**

II. Background

The Spider file system was designed as a center-wide shared resource to service all Oak Ridge Leadership Computing Facility (OLCF) resources, in 2008. The design was targeted to eliminate data islands, to reduce deployment costs, and to increase data availability. The system was connected to Jaguar and other OLCF resources through an InfiniBand (IB) DDR network network, named Scalable I/O network (SION). Each storage server was connected to a leaf switch that was then connected to two 108 port IB core switches. An aggregration switch then connected the core switches. Network translation services from Cray SeaStar to InfiniBand was provided by Lustre Networking (LNET) routers. These routers were also directly connected to the same two aggregation switches.

After deployment, it was discovered that network congestion both at the Cray SeaStar and InfiniBand networks were severely limiting aggregate I/O performance. To solve this problem, OLCF developed and implemented a congestion avoidance method named Fine-Grained Routing (FGR) [1] [2]. FGR had two components. First, it paired clients to specific I/O servers that are topologically close to each other, reducing the load on the common SeaStar torus links and avoiding SeaStar link saturation. Second, FGR introduced a new LNET routing configuration. This new configuration assigned varying weights to LNET routes based on client I/O server pairings. Tests showed that with FGR, aggregate performance was boosted by 30%. Other solutions have since adopted the FGR techniques, including Cray's Sonexion product [3].

Jaguar was upgraded to Titan, a Cray XK7 system, in 2012. Like Jaguar, Titan has 18,688 clients. However, each Titan node is augmented with one NVIDIA Kepler GPGPU which increased the aggregate installed computational power by more than an order of magnitude. This also increased the I/O requirement. To address this need, a new file system called Spider II was deployed in 2013. Spider II provides a 4x boost in aggregate I/O performance and a 3x increase in data storage capacity compared to Spider I.

Spider II was designed with a similar architecture to its predecessor, Spider I. 20,160 2 TB Near-Line SAS disks are organized in 8+2 RAID 6 sets controlled by 36 DataDirect Network (DDN) SFA-12K couplets. These are physically arranged into four rows in the data center. The storage system is split into two distinct, non-overlapping sections, and each is formatted as a separate name space (atlas1 and atlas2). Each file system has 144 Lustre Object Storage Servers (OSSs) and 1,008 Object Storage Targets (OSTs). As of publication, a patched version of Lustre 2.4.3 is running on the I/O servers. Each OSS is connected to one InfiniBand FDR top-of-the-rack (TOR) switch and two DDN controllers, for reliability. Each row has nine TOR switches (36 total). On Titan, 440 XK7 service nodes are configured as Lustre LNET routers. Of these, 432 are used for file I/O and 8 are for metadata communication. The Titan LNET routers are directly connected to the Spider II TOR switches. Table I shows the quantity of each component. More details on Spider II have been published previously [4].

*David Dillow was previously associated with Oak Ridge National Laboratory

Table I
SPIDER II COMPONENT COUNTS

Count per	Total	FS	Row	SSU	OSS	OST
Disks	20,160	10,080	5,040	560	70	10
OSTs	2016	1008	504	56	7	
OSSs	288	144	72	8		
I/O Routers	432	216	108	12		
IB Switches	36	18	9	1*		
Rows	4	2				
File Systems	2					

*Note: A given switch supports half of each of two SSUs

III. PLACEMENT

The placement of the I/O routers in a large 3D torus can have an enormous impact on the traffic patterns and congestion characteristics present in the system. This is important for maximizing I/O performance as well as minimizing the interaction between application communication and I/O. Building on the lessons learned from OLCF's Spider I implementation of fine-grained routing in 2008, an improved method for placing LNET routers on Titan was developed and implemented for Spider II.

A. Topological Concerns

The router placement layout used for Spider I was designed to distribute the routers topologically through the machine while also minimizing the number of cabinets that contained routers (see Figure 1). This resulted in a very regular I/O pattern that was prone to congestion if I/O traffic was not properly kept localized.

Jaguar's upgrade from Cray's SeaStar interconnect to Gemini significantly changed the network's characteristics. Details about Gemini's architecture and design are available in other literature [5]. Additional details and performance characteristics are also available [6]. Each Gemini supports two nodes, which effectively halved the Y-dimension length. Additionally, Y-dimension connections are comprised of only half the links of X- and Z-dimension connections. Thus, I/O traffic should be limited in the Y-dimension due to its reduced relative bandwidth. This suggests that routing zones should be "flattened" into "rectangular prisms" instead of the more traditional cubic zones as was implemented in Spider I.

Since Gemini uses dimension-ordered-routing, I/O tends to converge to a single path as it nears the destination. Thus, it is important to avoid putting routers in the same plane. This can help avoid congestion, but many-to-one communication patterns in a 3D torus will always suffer from some amount of congestion.

Minimizing the hop count between clients and routers is essential for providing high-bandwidth communications with the storage servers. While Gemini routing arbitration is locally fair, it is globally unfair. Packet age and hop-count are not taken into account when the router selects the next packet to forward. Figure 2 shows an example of this issue.

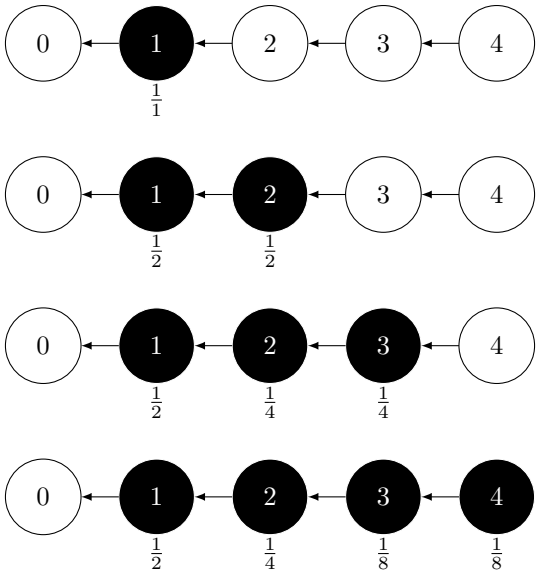

Figure 2. Geometric Bandwidth Reductions

Node 0 can be considered an I/O router while the others are acting as clients attempting to send data to the router. When only node 1 is communicating, it is able to achieve 100% of the bandwidth across the link. Once node 2 starts communicating, the router attached to node 1 accepts half the bandwidth from node 1 and half from node 2. Effectively, the bandwidth is shared between the nodes. When node 3 begins communicating, the router attached to node 2 fairly arbitrates traffic between nodes 2 and 3. Since that router only has half of the global bandwidth, nodes 2 and 3 each only get one quarter of the total bandwidth to the router. When node 4 begins communicating, the problem becomes even more obvious. The router attached to node 3 fairly arbitrates traffic between nodes 3 and 4, but it can only grant one eighth of the total bandwidth to each.

As these chains get longer and longer, the bandwidth available to the "last" node can become abysmal.

B. Physical Constraints

The following physical constraints and goals were kept in mind while determining an optimal placement algorithm:

Topological Concerns
: Routers must be placed to optimize for the topological concerns mentioned in Section III-A.

Partitionability
: Occasionally, Titan's 3D torus must be partitioned to facilitate extended maintenance activities or testing. During this situation, it is important to ensure full routability to boot from both "ends" of the machine. A boot node is located in physical columns 0 and 24 (topological columns 0 and 13). Thus, routers should be located in a way

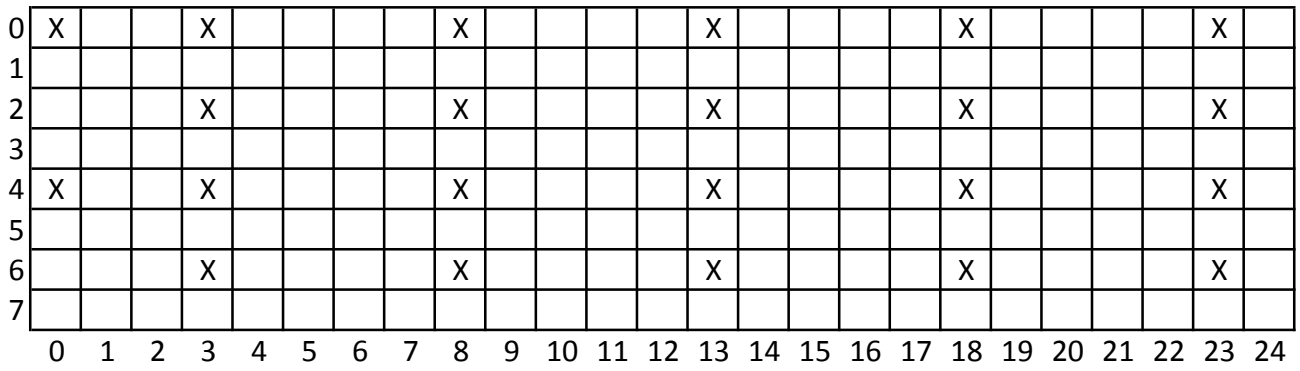

Figure 1. Jaguar Router Layout

that minimizes the number of columns required to access the entire file system.

Minimize Cabinet Count
: Each cabinet that contains a router module requires a hole to be drilled in the floor to accommodate the cables.

Cable Length
: Shorter cables are cheaper and easier to manage in the data center.

C. Placement Choices

The challenge is to choose the optimal placement for the 432 I/O routers. In the Cray XK architecture, each service module contains four service nodes. Since each service module displaces a compute module, it is important to utilize each node. Ideally, the LNET routers would exist on the same module as less bandwidth-intensive nodes (such as a login nodes) to dedicate more bandwidth to routing. This is unfortunately impractical due to the sheer number of routers required to achieve sufficient bandwidth.

Spider II has 36 InfiniBand switches that are named based on row and position: *atlas-ibsw{1-4}{a-i}*. Titan's 432 I/O routers are divided equally among the switches, with each switch providing 12 connections to Titan. Titan's routers are named based on the switch to which they connect and their index: *rtr{1-4}{a-i}-{1-12}*.

Having 4 routers on a module corresponds nicely with the 4 rows of Spider II. The four nodes on a given module will connect to the same "letter" switch in each of the 4 rows of Spider II. Initially, this was designed such that nodes 0, 1, 2, and 3 connected to rows 1, 2, 3, and 4, respectively. Before deployment, OLCF staff realized that this connection might be suboptimal. Router 0 and 1 share a Gemini, so in the typical use case of a job writing to a single file system, contention may be present on the links leading to that Gemini. The placement was modified so nodes 0 and 2 connect to rows 1 and 2 while nodes 1 and 3 connect to rows 3 and 4. This "swizzle" is designed to spread the load for a single file system to two different Gemini chips within a module.

OLCF staff wrote several *Perl* scripts to generate and analyze the placement for the I/O routers. First, the machine was broken up into 4 sections of two rows each. The intention here was to limit hops in the Y-direction due to its limited bandwidth. The 12 modules for a given group were assigned to the sections, three per section. The starting position for the first router was manually determined to aid in satisfying physical constraints. The subsequent routers in the subgroup were placed approximately $\frac{1}{3}$ around the X- and Z-dimensions. When two modules for different groups are placed in the same cabinet, they are offset by 12 in the Z-dimension ($\frac{1}{2}$ of the way). Between sections, the starting X- and Z-dimensions are offset. By modifying the starting positions and parameters, several iterations were created and evaluated before arriving at a final configuration. The final placement map is shown in Figure 3.

IV. Fine-Grained Routing

The first step towards achieving higher performance is to place LNET routers equidistant from each other as much as possible, subject to physical constraints and other boundary conditions. This ensures that LNET routers are distributed across the machine and are not segregated in a particular portion of the torus. The second key step is to pair clients with their closest possible LNET routers. This section discusses the algorithm and implementation that pairs a client with the optimal LNET routers to minimize end-to-end hop counts and congestion.

A. Router Selection Algorithm

Recall from previous sections that there are a total of 440 LNET routers where only 432 routers are used for file I/O; the remaining 8 are used for metadata operations. Four LNET routers together form a router module, resulting in a total of 108 router modules spread throughout the system. These 108 router modules are divided into 9 groups of 12 modules each, corresponding to the 9 InfiniBand switches

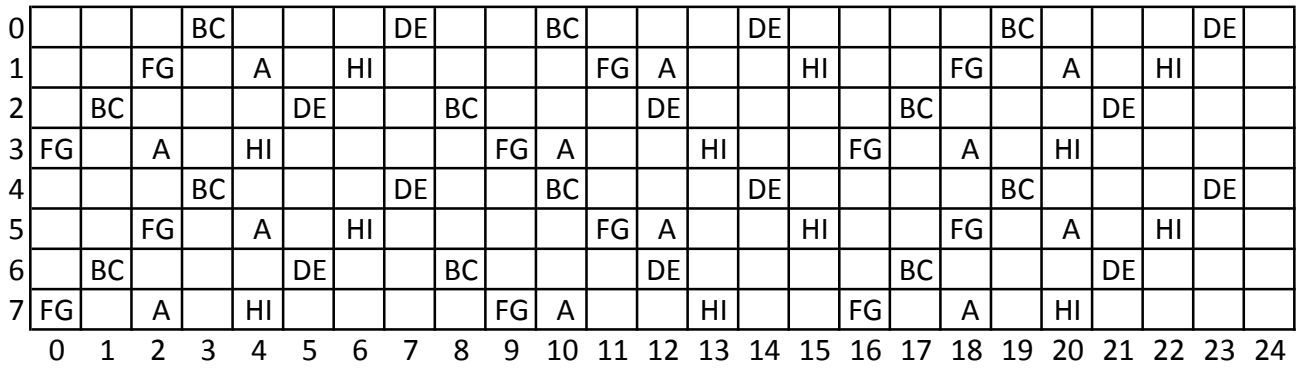

Figure 3. Titan Router Layout

present in each row of Spider II. Each group is further divided into 4 sub-groups that service two rows of Titan. Each sub-group consists of 3 router modules.

Algorithm 1 describes how a client chooses the optimal router module for a given group. The client-to-group pairing is decided using a fixed arrangement. Note that in the presented algorithm $R_i^G(S)$ denotes i^{th} router module in the G^{th} group and S^{th} sub-group. Based on the number of groups, sub-groups, and router modules in the system, G, S and i will have the following respective ranges: $(1, \ldots, 9)$, $(1, \ldots, 4)$, and $(1, \ldots, 3)$.

The algorithm takes two input parameters: coordinates of the client (C) and the destination router group (R^G). The algorithm returns the coordinates of three routers assigned to the input client (C): one primary and two backup routers.

The fine-grained routing algorithm consists of two steps. The first step (lines 4–14) is to choose the sub-group whose Y-coordinates fall within the close range of Y-coordinates of the input client, C. This is because the bandwidth in the Y-direction is limited compared to other directions, as discussed earlier. Therefore, it is desirable to minimize the traffic in that direction first.

Once the sub-group is selected, the second step (lines 16–31) is to return assigned routers from this sub-group. As mentioned earlier, each sub-group consists of 3 routers and the algorithm returns a vector of 3 routers. So, all three of them are returned, but the one with the lowest distance in X-dimension is assigned as the primary router to minimize the hops and avoid congestion in that direction. Note that the X-direction crosses cabinet boundaries, therefore it more desirable to minimize the hops in that direction compared to the Z-direction that run within a cabinet in vertical direction.

B. LNET Routing

Lustre uses LNET for all communications between clients, routers, and servers. Routing allows communication to span multiple network types, as long as one or more nodes exist that can "bridge" the disjoint networks. Each unique network is given an identifying name that consists of the

Algorithm 1 Fine-grained routing algorithm

1: **procedure** ROUTE SELECTION ALGORITHM (R^G, C)
2:
3: Divide R^G into 4 sub-groups: $R^G(1) \ldots R^G(4)$.
4: **for all** sub-groups $R^G(i)$ **do** ▷ i ranges 1 to 4
5: $C[y] \leftarrow$ y coordinate of C
6: $R_1^G(i) \leftarrow$ first router module in the i^{th} sub-group
7: $R_1^G(i)[y] \leftarrow$ y coordinate of $R_i^G(S)$
8: **if** $(C[y] == R_1^G(i)[y] - 1)$
9: or $(C[y] == R_1^G(i)[y])$
10: or $(C[y] == R_1^G(i)[y] + 1)$
11: or $(C[y] == R_1^G(i)[y] + 2)$ **then**
12: break with sub-group i selected
13: **end if**
14: **end for**
15:
16: $i \leftarrow$ index of selected sub-group
17: $r_1, r_2, r_3 \leftarrow$ first, second, and third router module
18: selected sub-group i
19: $d_{min} \leftarrow \infty$
20: $Index_{primary} \leftarrow \infty$
21:
22: **for** j in $1, \ldots, 3$ **do**
23: $d_{current} \leftarrow$ dist$(C[x], r_j[x])$ ▷ distance along X dimension
24: **if** $d_{current} < d_{min}$ **then**
25: $Index_{primary} \leftarrow j$
26: **end if**
27: **end for**
28: primary router module $\leftarrow R^G_{Index_{primary}}(i)$
29: backup router modules $\leftarrow$
30: two other modules in the i^{th} sub-group
return <primary and backup router modules>
31:
32: **end procedure**
33:

network type and an arbitrary integer (for example, *o2ib0* for the 0^{th} InfiniBand network or *gni101* for the 101^{st} Gemini network).

Each node in an LNET network has a unique Network Identifier (NID) that is in the form *identifier@network*. The InfiniBand Lustre Networking Driver (LND) uses the IP-over-IB address as its unique identifier (ex. *10.10.10.101@o2ib0*), while the Gemini LND uses its Gemini network ID (ex. *4044@gni101*). It is permissible for a network interface to have multiple NIDs assigned to it. For example, a node with a single InfiniBand interface may have NIDs *10.10.10.101@o2ib0*, *10.10.10.101@o2ib1*, and *10.10.10.101@o2ib2*. The Lustre Manual [7] describes how to specify network settings.

In mixed-network environments, system administrators setup the LNET routing tables on each node. For every remote network that the node should be able to communicate with, a list of routers should be provided. Each router is given a "hop count" that indicates the relative distance between the nodes.

When a packet is ready to leave a node, the destination network ID is compared to the list of local network IDs. If a match is found, then the message is sent directly to the destination. Otherwise, the routing table must be used to determine the next hop. Under normal circumstances, LNET will cycle through all the appropriate routers with the lowest hop count. Routers with higher hop counts will only be used if **all** routers of a lower hop count are unavailable. In all cases, LNET uses its source NID that matches the network of the next hop.

C. FGR in Practice

Thirty-six IB LNET network identifiers (*o2ib201* to *o2ib236*) exist that correspond to the 36 IB leaf switches. Each LNET router has exactly one IB NI that corresponds to the switch to which it connects. The service and compute nodes are broken up into twelve Gemini regions (*gni101* to *gni112*) based on their topological location. Each router configures 3 *gni* interfaces corresponding to the three indices in the topological section.

Upon boot, each client applies Algorithm 1 for all groups A to I. The node will create a *gni* interface corresponding to each primary router. The primary router is added to the routing table with hop count 1 while the secondaries are added with hop count 10. In the end, each client will have 36 primary routes (one for each IB switch) and 72 secondary routes.

Additionally, *gni100* is configured on all clients for metadata and Cray DVS traffic. Lustre metadata traffic uses all 8 metadata routers; it does not use fine-grained routing.

The Lustre servers each configure one *o2ib* network identifier that corresponds to its IB leaf switch. Routes for all 12 *gni* networks are configured through the 12 routers also connected to the same switch.

V. Issues and Future Work

The I/O router placement attempts to address the issue of traffic routing and imbalance at the system level. The route selection algorithm aims to minimize hops and mitigate contention between the compute clients and I/O routers. However, several issues remain.

Titan's scheduler is completely oblivious to the placement of I/O routers; jobs are placed based on node availability. No mechanism exists for nodes to request locality to or distance from I/O routers. A job placed entirely within one section (two rows) of the machine, for example, will never be able to achieve greater than $\frac{1}{4}$ of the total file system bandwidth. Identical jobs placed in different sections of the machine may have widely varying locality to routers. To users, this manifests as I/O variability.

Benchmark tests against Spider II have been run using both optimally placed and randomly placed clients. On a quiet system the difference between the two modes is minimal. However, on a busy system the difference can be more substantial. Arbitrary users jobs have no insight into which ranks are closest to I/O routers. To overcome this limitation, OLCF is designing an end-to-end balanced data placement strategy to complement the backend fine grained routing algorithm. The primary goal is to extend the balancing from the clients, through the I/O routers, and into the final destination. This work is ongoing.

While the concepts and algorithms behind fine-grained routing are straightforward, the actual implementation is quite complex. When issues arise, it can be difficult narrow down to find the root cause. Over time, various scripts have been developed to ensure all nodes are cabled correctly and that they can communicate properly. Additional work improving these scripts will aid in timely debugging.

VI. Conclusion

The evolution of compute platforms at Oak Ridge National Laboratory has necessitated unique designs for the storage systems that support them. Lessons learned from the deployment and operation of the Spider I file system led to the development of fine-grained routing to improve throughput and minimize congestion. The Spider II system was designed and built with fine-grained routing as a core component. To support this new file system, the quantity of I/O routers in Titan was increased and they were carefully placed to minimize congestion. A new fine-grained routing mechanism was created that was tailored specifically to the two systems.

VII. Acknowledgement

This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

Notice: This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

[1] D. A. Dillow, G. M. Shipman, S. Oral, and Z. Zhang, "I/O congestion avoidance via routing and object placement," in *Proceedings of Cray User Group Conference (CUG 2011)*, 2011.

[2] D. Dillow, G. Shipman, S. Oral, Z. Zhang, and Y. Kim, "Enhancing I/O throughput via efficient routing and placement for large-scale parallel file systems," in *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, Nov 2011, pp. 1–9.

[3] M. Swan and N. Henke, "Cray's implementation of LNET fine grained routing," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[4] S. Oral, D. A. Dillow, D. Fuller, J. Hill, D. Leverman, S. S. Vazhkudai, F. Wang, Y. Kim, J. Rogers, J. Simmons, and R. Miller, "OLCF's 1 TB/s, next-generation Lustre file system," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[5] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini system interconnect," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, Aug., pp. 83–87.

[6] M. Ezell, "Understanding the impact of interconnect failures on system operation," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[7] Lustre Developers and Contributors, "Lustre file system documentation." [Online]. Available: http://lustre.opensfs.org/documentation/