I/O Congestion Avoidance via Routing and Object Placement

David A. Dillow, Galen M. Shipman, Sarp Oral
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
{dillowda,gshipman,oralhs}@ornl.gov

Zhe Zhang*
T.J. Watson Research Center
International Business Machines
zhezhang@us.ibm.com

ABSTRACT: As storage systems get larger to meet the the demands of petascale systems, careful planning must be applied to avoid congestion points and extract the maximum performance. In addition, the large size of the data sets generated by such systems makes it desirable for all compute resources in a center to have common access to this data without needing to copy it to each machine. This paper describes a method of placing I/O close to the storage nodes to minimize contention on Cray's SeaStar2+ network, and extends it to a routed Lustre configuration to gain the same benefits when running against a center-wide file system. Our experiments show performance improvements for both direct attached and routed file systems.

1 Introduction

The Oak Ridge Leadership Computing Facility (OLCF), located at Oak Ridge National Laboratory, houses Jaguar, a 200 cabinet Cray XT5. Jaguar [5] offers 18,688 compute nodes, each with two hex-core AMD Opterons, providing 224,256 cores and 2.3 petaflops of compute performance and nearly 300 terabytes of system memory. The OLCF also hosts a number of smaller scale systems used for analysis and visualization work, software development, and integration of evolving storage and compute technologies such as accelerator-enhanced systems. Supporting the I/O demands of these systems falls to Spider [9], our Lustre based center-wide file system, designed to provide reliable global accessibility and high performance.

Time on Jaguar is a limited resource. The majority of compute time on Jaguar is managed as part of the US Department of Energy's Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. For 2011, the average allocation for each accepted project was 27 million CPU-hours [1]. While this may seem to be all the time in the world, it represents just over 120 hours (5 days) at full scale on Jaguar. In these situations, it is desirable to achieve as much efficiency as possible to conserve resources. Much of this efficiency work is focused on improving the computational performance of the scientific codes, as that is where they spend the bulk of their time. However, attention should also be paid to the time spent reading the input deck, writing out results, and performing defensive I/O to protect against system failure during long-running simulations. Time spent performing these operations also represents time that could be used to more quickly perform the science and/or improve the resolution and detail of the study.

Our activities during the acceptance of Jaguar and Spider revealed a substantial digression between the expected and actual performance of the systems. While we had made provisions during the design of the system to spread the I/O load throughout the network in a manner inspired by Azeez et al [4], congestion and the associated load imbalance degraded our aggregate performance. This was clearly visible when watching the bandwidth statistics on the back-end storage. We would see impressive sustained performance at

*This work was performed while Zhe Zhang was a staff member at Oak Ridge National Laboratory.

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.

the expected peak bandwidth numbers, only to see the average bandwidth numbers plummet as the bulk of the writes completed their task, leaving several stragglers writing their share of the work at a small fraction of the available aggregate speed.

We identified the source of these losses and developed techniques to avoid or reduce their impact, realizing the performance of the storage system and allowing for reduced time-to-solution for a suite of scientific applications.

2 Spider and SION

Spider is one of the world's fastest and largest POSIX complaint parallel file systems. Designed for high performance in a small footprint, the systems is built from scalable building blocks. Each block is comprised of a DataDirect Networks (DDN) S2A9900 storage system, driven by four Lustre Object Storage Servers (OSS). Each DDN is configured with five high density drive trays, with 300 drives housed in 20U of rack space. For Spider, we populate those trays with 280 SATA drives, each with a capacity of 1 TB. Write-back caching is disabled to prevent unrecoverable data loss in the event of a controller failure. These drives are grouped into 28 DirectRAID tiers (RAID3 with two parity drives), with seven 7.2 TB LUNs exposed over DDR Infiniband for each OSS. Each OSS is a Dell PowerEdge 1950, with 16 GB of memory and two quad core Xeon E5410 running at 2.3GHz. Each OSS serves seven Object Storage Targets (OST). This building block is capable of delivering over 5.5 GB per second of raw block storage.

There are 48 of these building blocks in the Spider system, giving it an aggregate of 13,440 TB raw storage, or over 10 PB of capacity after accounting for the parity overhead of DirectRAID. There are 192 OSS servers, providing 14 teraflops of compute and 3 TB of memory dedicated to our Lustre file systems. This aggregate capability is broken up into three disjoint chunks to spread the metadata load. Widow1 has half the storage – 672 OSTs – while widow2 and widow3 equally and disjointly split the other half, each with 336 OSTs.

Metadata services are provided by three identical MetaData Servers (MDS). Each MDS is a Dell R900 with 64 GB of memory and four quad core Xeon E7330 running at 2.6 GHz. The MetaData Target (MDT) for each file system is stored on a shared Engineo 7900 (XBB2) storage system, connected via four 4Gbps Fibre Channel connections to each MDS. Each MDT is configured as a RAID10 volume on the XBB2 with 80 SATA 1 TB drives, short stroked to provide an 8 TB LUN.

Spider was intended to be used as a center-wide file system, and we designed our Scalable I/O Network (SION) to accommodate its performance goals. SION is deployed as a multi-stage DDR Infiniband fabric, and provides over 889 GB/s of bi-sectional bandwidth. The network infrastructure is based on three 288-port Cisco 7024D DDR Infiniband switches. Two core switches are dedicated to providing connectivity between Jaguar and Spider, while a third is used to tie the core switches together and provide links to the MDS and management services. That aggregation switch also has connectivity to the rest of the center via a fourth 7024D. Spider is connected to the core switches via 48 24-port Flextronic "Reindeer" switches, which allows the DDN 9900 storage systems to also be accessed over SION for test purposes. In total, SION contains over 3,000 Infiniband ports and over three miles of optical cables.

The 7024D switches each provide 288 user-facing ports via 24 leaf modules with 12 external ports and 12 uplinks into the internal fat-tree topology. Of the combined 48 modules in the two core switches, 32 are used to connect Jaguar to Spider – 16 in each core switch. In each of those modules, 6 ports are connected to service nodes (SIO) on Jaguar, and 6 are used to connect to an OSS or storage in Spider.

3 The Direct-attached Lustre File System

During the initial delivery and acceptance of the XT5 segment of Jaguar, we deployed a direct-attached Lustre file system to address any issues in the back-end storage and allow early users a more stable platform while we proceeded to test Lustre's routing capabilities at an unprecedented scale. To this end, 96 SIO nodes on Jaguar were configured to be OSSes. Each OSS was configured virtually identically to the Spider system – 7 OSTs, one per exported LUN, with one LUN per DDN 9900 tier. Using the properties of SION noted above, we carefully paired each SIO node with storage that would be directly accessible from the same leaf module in the 7024D switch as the SIO node. This allowed us to avoid any potential source of congestion on the InfiniBand network as all network traffic from the OSS to the backend storage was isolated to the crossbar and did not traverse the fat-tree fabric.

To establish a baseline for potential performance, we used XDD on the XT5 SIO nodes to exercise the storage system. XDD was configured to exercise all seven LUNs on each OSS simultaneously. Each LUN received a steady workload of sequential 1 MB requests, with 4 requests were kept active at all times. In this manner, each OSS had a total of 28 I/O requests outstanding at any given point during the testing. XDD was configured to run each test for 60 seconds, and the maximum bandwidth achieved from three trials is presented in Figure 1. The aggregate performance of the system scales nearly linearly as each additional OSS is added to the test. We reach our peak bandwidth of 120 GB/s running with 96 servers, with each LUN contributing nearly 180 MB/s. The slight ripple in the XDD results as

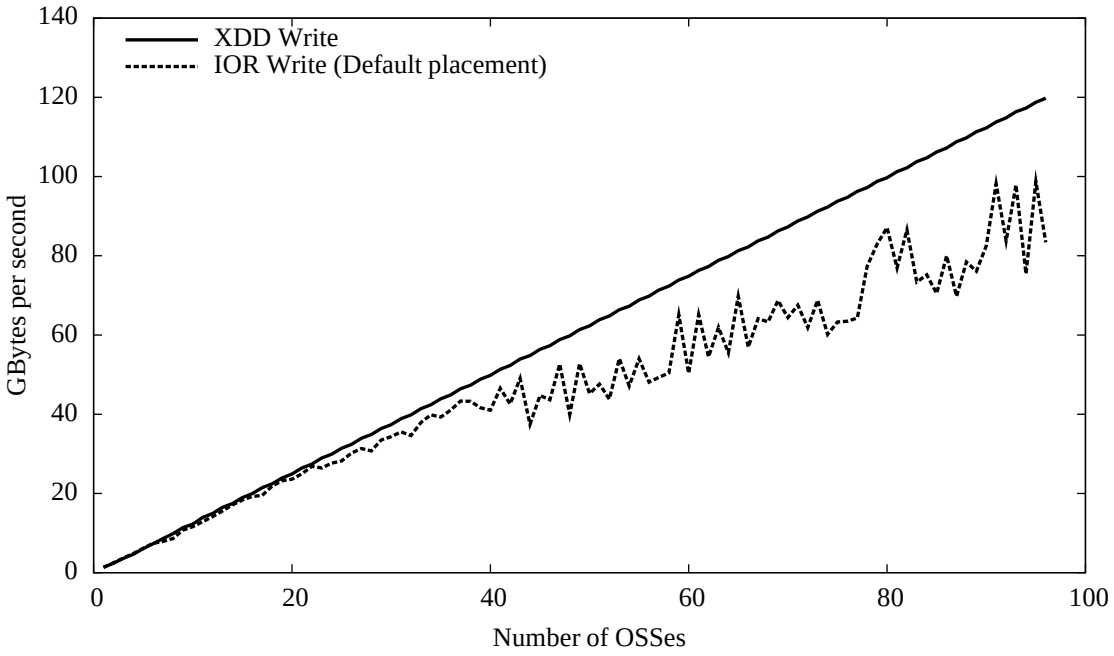

Figure 1: Baseline write performance for direct-attached Lustre filesystem

the number of OSSes increases is due to internal limits of the DDN 9900 architecture. While each port on the couplet is capable of 1,500 MB/s when tested in isolation, the singlet reaches a maximum bandwidth of approximately 2,500 MB/s. More recent versions of the DDN firmware lift the per-singlet bandwidth to slightly more than 2800 MBbyte/s for our configuration.

Having demonstrated that the back-end storage is capable of delivering a raw performance of over 120 GB/s, we pressed forward with a Lustre configuration on the XT5. Each of the 18,688 compute nodes was configured to be a client of the direct-attached file system, communicating directly over Cray's proprietary network to the OSS servers. We used IOR in file-per-process mode to load the file system for the Lustre level testing. We configured the job launcher to only place one process per node to avoid any host-level bottlenecks in the client code, allowing us to focus on the network and storage performance. We used the Lustre utility `lfs setstripe` to pre-populate the output files for the IOR runs, which avoided file creation overheads and ensured that we did not have ranks/files contending for an OST. IOR was configured to insert a barrier between each phase of its operation to ensure that all ranks had their respective file open prior to testing bulk I/O performance. Each rank transferred 3 GB of data with a transfer size of 8 MB. The maximum bandwidth achieved from five trials is presented in Figure 1.

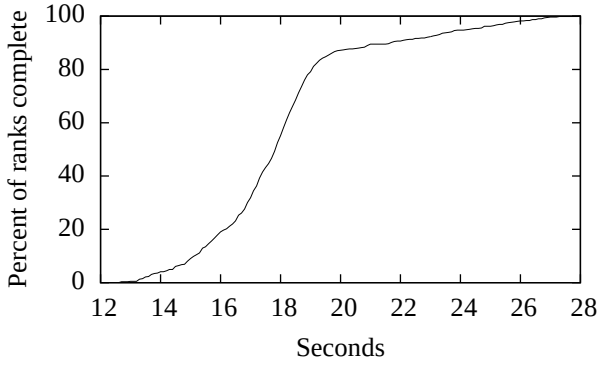

Figure 2: Cumulative Distribution Function plot of per-writer completion time from best run with default placement.

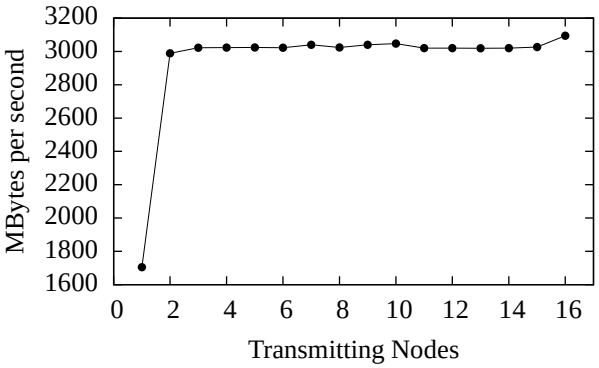

Figure 3: Unidirectional link bandwidth on SeaStar.

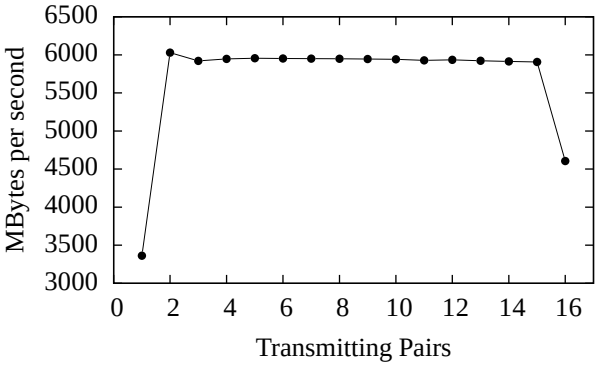

Figure 4: Bidirectional link bandwidth on SeaStar.

Testing at small OSS counts demonstrated performance that was in line with our expectations of approximately 1250 MBbyte/s per OSS. This performance scaled linearly up to 6 OSSes and then fell below our expectations of near-linear scaling. Beyond 40 servers in this scaling study, performance became highly erratic, achieving at best 86% of the raw baseline, and 61% in the worst case. As illustrated by the "Default" series in the cumulative distribution function (CDF) presented in Figure 2, a large difference in completion times persists between the fastest and slowest ranks. A large number of writers complete near the 18 second mark, but after 19 seconds a number of writes straggle in over the course of 10 seconds. Investigating the cause of this behavior led us to network-congestion on Jaguar's SeaStar2+ network as a primary cause.

4 SeaStar Link Congestion

Every node in the Cray XT5 is connected into a 3D torus via the SeaStar 2+ interconnect chip (or NIC). Much has been written about the XT network since its introduction in Sandia's Red Storm machine ([7], [10], [6] for example), so we will only briefly touch upon the details here. Each SeaStar NIC also acts as a router for the network, and has six independent, full duplex links to the rest of the system. Each input port has a set of lookup tables that determine the egress port of a packet based on the destination address of the packet. These tables are initialized to implement dimension-order routing during the process of booting the machine, and remain static until the next boot. Barring accommodations for failed or missing components that prevent a uniform torus topology, packets are routed in a fix dimension order (X+, Y+, Z+, X-, Y-, Z-) [2].

We verified the bandwidth properties of the SeaStar network using a locally-developed Portals tool. This tool divides the nodes it is launched on into contiguous segments along each axis, and then generates traffic between pairs of nodes chosen to ensure that a single link is stressed in the axis under test. The offered load is 30 seconds of a streaming series of 1 MB transmit requests; we allow up to 256 requests to be in flight for each pair of nodes and the receiving node returns credits to the transmitting node every 32 messages. The tool varies the number of node pairings from one to the maximum number possible in each segment and tests each axis independently. We present the results from the Y axis of Jaguar in Figures 3 and 4. This axis contains the maximums of both cable length and dimension diameter for the machine. The data for other dimensions is nearly identical.

As shown in Figure 3, we measured 1,705 Mbytes/s of injection bandwidth from user space on a single node. Link bandwidth remains nearly constant at approximately 3,020 MB/s from 2 to 15 transmitting nodes, with a small increase to 3,094 MB/s at 16 transmitting nodes. When running the link in a full duplex mode (Figure 4) bandwidth nearly doubles as expected. Our initial pair achieves 3,363 MB/s, somewhat less than the perfect 3,410 MB/s. This is likely due to measurement noise and other small inefficiencies from the HyperTransport (HT) and processor memory buses. The link saturates at approximately 5,950 MB/s, which is slightly less than the expected 6,040 MB/s. We believe this loss to also be a result of measurement noise. The down-tick at 16 pairs may be due to credit starvation, but this was not investigated.

Our IOR testing (§ 3) showed that we can expect each OST to contribute 180 MB/s to the aggregate bandwidth under sustained loading of the entire Spider system. Given our measured SeaStar link bandwidth, each link can support 17 client-OST pairs before reaching saturation. By examining the routes programmed into the torus at the time of the test, we can calculate the number of client-OST pairs communicating over each link. We found that 70% of the test runs using the default placement had at least one link

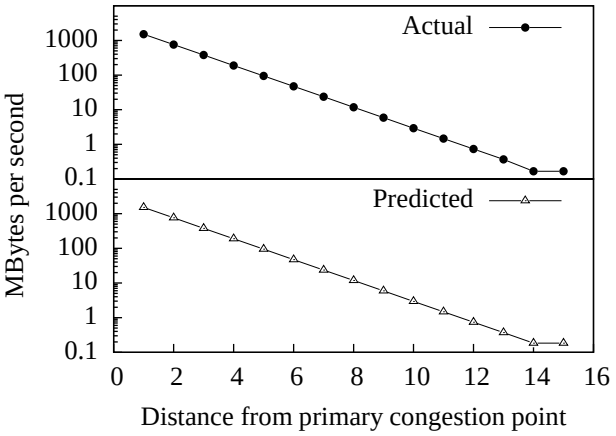

Figure 5: SeaStar link balance under congestion.

with 18 or more pairs communicating over it, 42% had at least one link with over 34 pairs, 21% with over 60 pairs, and 3% had over 70. This magnitude of bandwidth sharing is a significant contributing factor to the poor scaling results initially demonstrated.

In their discussion of age-based packet arbitration, Abts and Weisser [3] describe the behavior of the SeaStar network when packet age is not taken into account. Traffic traveling over more hops receives geometrically less bandwidth. The bandwidth each node receives in the test described above is governed by the equation

$$B_{node} = \frac{B_{link}}{(2n)^H}$$

where n is the number merging streams (1 for the farthest node from the congestion point, 2 for all other nodes) and H is the number of hops to the network segment under test. The transmitting node adjacent to the tested link has $n = 2$, $H = 1$, and is expected to receive half of the available bandwidth; the next node from the test link has $n = 2$, $D = 2$ and receives a quarter of the available bandwidth, and so on. Figure 5 shows that the the first node achieved 1,513 MB/s, which compares favorably to the expected value of 1,510 MB/s. For the nodes farthest from the test link, the nodes receive approximately 166 KB/s, or three orders of magnitude less bandwidth than the node nearest the link. This behavior further distorts the balance of the I/O times of client-OST pairs sharing a saturated link.

5 Controlling SeaStar Congestion By Placing I/O

To avoid congestion on the SeaStar links, we must carefully control which client talks to a particular OSS for the direct-attached file system. We achieve this by drawing from the well known pattern of nearest-neighbor communication often found in HPC applications. Client nodes are chosen such that each client is a minimum number of hops (distance) from the OSS responsible for the file it is writing. Given a sufficient set of nodes from which to select active clients, it is possible to avoid saturating a link as there are only seven OSTs served by each OSS, and a single client/file per OST for our testing.

For a 3D torus, given the coordinates of two nodes n_1, n_2, and the length of each axis L_{axis}, the hop count h (distance) is given by:

$$h(n_1,n_2) = d_x(n_1,n_2) + d_y(n_1,n_2) + d_z(n_1,n_2)$$

where

$$d_{axis}(n_1,n_2) = \min \begin{cases} (n_{1,axis} - n_{2,axis}) \bmod L_{axis} \\ (n_{2,axis} - n_{1,axis}) \bmod L_{axis} \end{cases}$$

The best client(s) to use for a specified OSS n_{OSS} is given by calculating $h(n, n_{OSS})$ for each n in the set of compute nodes available. Choose the N nodes with minimum distance to n_{OSS}, and remove them from the set of available node. These are the best clients to use for the OSS. Repeat these steps for each OSS you wish to involve in the test.

Using the above algorithm and placing all 18,688 compute nodes into the available set, we ran another scaling study of the direct attached Lustre file system. We used `lfs setstripe` to place the file for each rank on a known OST – and thereby known OSS – and used a feature of the job launch facility to place the ranks of the IOR job onto the compute nodes such that the clients were paired up with OSTs on the nearest OSS to the client. The results were dramatic; as seen in Figure 6, aggregate bandwidth was increased an average of 18% when more than 40 OSSes were involved in the test. We see a minimum gain of 5%, and realize increases of over 45% in the best cases. Using placement of the I/O to avoid saturating links of the torus, IOR is able to achieve 87 to 92% of the aggregate raw performance of the back-end storage system as measured by XDD. In these tests, no more than 7 client-OST pairs shared a common link in the torus for their communication. This compares favorably to the default placement, where over 70% of the links had 18 or more client-OST pairs, overwhelming the links with offered load.

The reduction in the difference between the fastest and slowest writer when using placement is shown by the "Placed" series in Figure 7. All writes complete within a

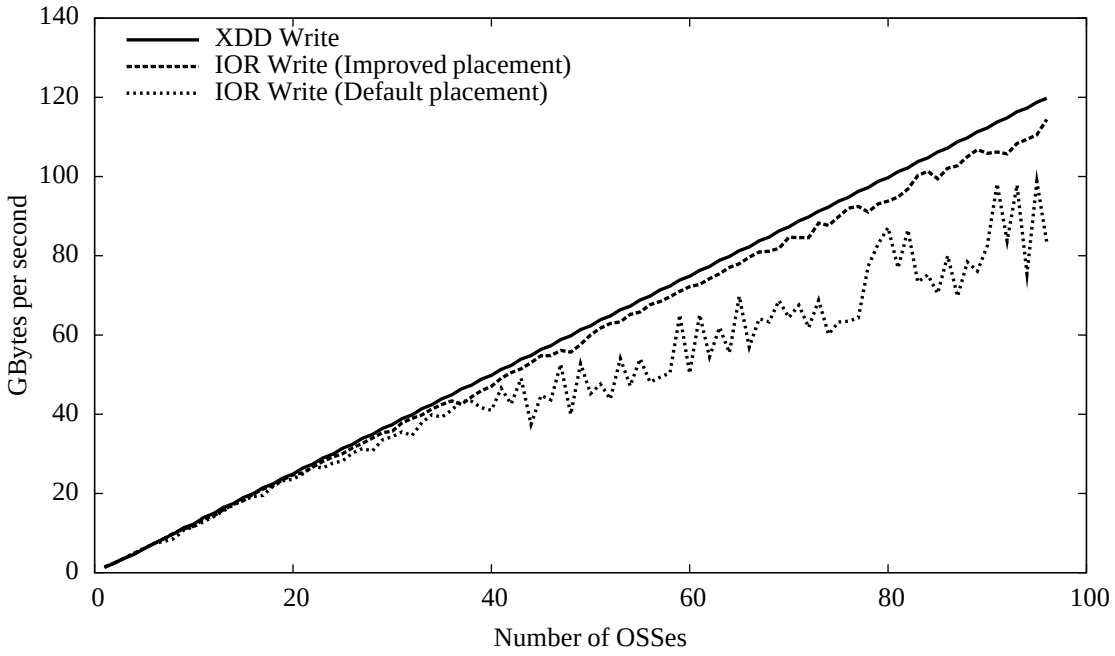

Figure 6: Write performance for direct-attached Lustre filesystem

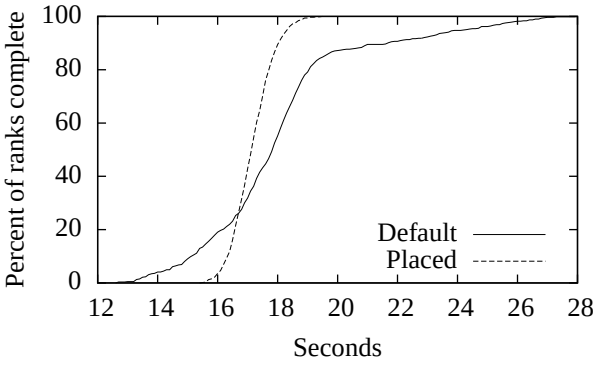

Figure 7: Cumulative Distribution Function plot of per-writer completion time. "Default" series is from best run without placement, "Placed" series is from worst run with placement.

span of less than 5 seconds, while the "Default" placement complete over a span of approximately 15.5 seconds. With the elimination of congestion in the torus, we achieve a better balance between writers and we leave less of the storage system idle while waiting for the stragglers.

While the results when allowing the placement algorithm free reign of the machine to optimize the node placement are very encouraging, many applications desire the improved performance even when not using every core possible. To determine the potential improvement when using only 672 nodes of the machine – one for each OST in the test file system – we ran an experiment using the same nodes that the job launcher gave us for the un-optimized placement. Additionally, to investigate the impact of location in the torus on performance, we launched a "place-holder" job that consumed a specified number of nodes and kept them idle, displacing the location from which our test job would execute. The parameters of the test are as before, with the number of OSTs fixed at 672 and the variables being the offset within the machine and placement of the processes performing I/O. Each test was run five times, and the best bandwidth is reported in Figure 8. Optimizing which processing element performs I/O to a particular OST improved performance from 38 to 58%. In all cases throughput within 5% of the maximum observed was achieved when allowing unconfined placement of processes performing I/O. This demonstrates that substantial improvements are possible us-

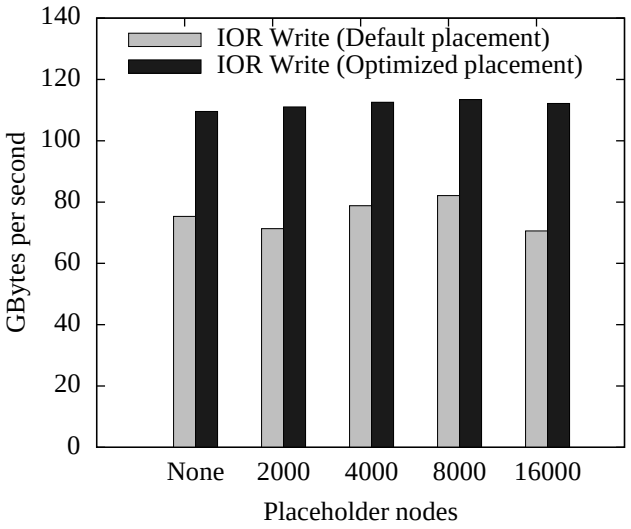

Figure 8: Write performance for direct-attached Lustre filesystem with confinement of available compute nodes.

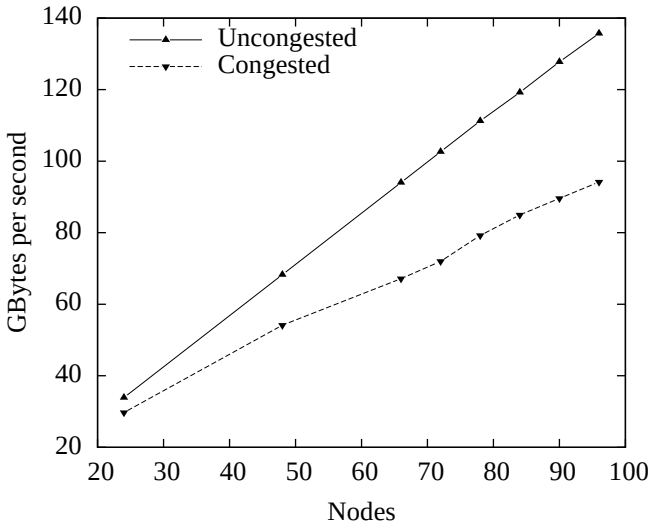

Figure 9: Aggregate block-level bandwidth of SION to 24 DDN 9900 Couplets. This test was performed with updated firmware compared to Figures 1 and 8.

ing our approach even for smaller scale application runs.

6 LNET Routing and the Baseline Routed Configuration

For the initial configuration of the routed file system, the XT5 was configured in the 'ptl' LNET network and our servers (OSS and MDS) where configured in the 'o2ib' LNET network. We converted the existing OSS servers (XT5 SIO nodes) from the direct-attached file system into LNET routers. We then added another 96 XT5 SIO nodes as LNET routers, bringing the total number of routers on the Jaguar system to 196. While this configuration had the benefit of a shared center-wide file system configuration, we quickly found that there were obstacles to achieving the performance potential we had demonstrated during our direct-attached testing. The LNET algorithm for choosing a router prohibited the use of the placement technique to avoid congestion on the torus. Furthermore, I/O traffic was no longer isolated to traversal of a an InfiniBand crossbar, causing congestion within the InfiniBand fat-tree network.

In order to chose the router to use for a given message to a remote LNET network, the client keeps each possible router for a given weight class on a list. For each message to be sent, the first alive router is selected from the list as the destination for this message, and that router entry is then placed at the tail of the list. In this manner, the client will distribute the load among all alive routers in a given weight class for a remote LNET. When there is more than one router for a remote network it is no longer possible for an application to predict which router will handle a given request, rendering any attempt to place clients topologically close to their destination ineffective.

In addition to prohibiting attempts to optimize congestion on the torus, this simple route configuration will inject traffic into the InfiniBand fabric from ports that are not on the appropriate leaf module. This forces traffic to traverse the internal fat-trees on the core switches, leading to congestion deep within the switches and reduced aggregate performance. While adaptive routing strategies [8] have shown significant performance improvements for long-lived communication patterns, their effectiveness in short-lived, highly-dynamic communication patterns has not been demonstrated.

Figure 9 shows an example of the impact of congestion in the InfiniBand fabric. This is a block level test, much like the XDD tests from the direct-attached file system. The "Uncongested" series shows the linear increase in speed expected when the SIO node is issuing block IO to a storage back-end on the same leaf module. The "Congested" series shows the aggregate bandwidth achieved when an SIO node is forced to traverse the fat-tree to communicate with an off-module storage module. At 96 SIO nodes – using half of the Spider hardware (24 couplets/48 singlets) – the observed performance is 135.7 GB/s without congestion and 94.1 GB/s with congestion. This represents a 30% performance degradation. These experiments were performed some time after the results from Figures 1 and 6. We had then transitioned much of our internal block level testing to use our newly developed synthetic benchmark, `fair-lio`. `fair-lio` has better sequential I/O characteristics com-

pared to XDD and some of the improved performance noted in the "Uncongested" series of Figure 9 is a result of that change.

7 Improved Routing Configurations

To minimize the congestion on the InfiniBand network and to allow application developers the ability to optimize their I/O using our placement strategy, the following routing configurations were considered:

1. *Nearest-neighbor.* In this configuration, the servers and routers are broken up into 32 sets. Membership in a set is governed by the module in the core switch the element is connected to. This configuration leads to 32 remote LNETs each accessible by 6 routers. Each client is configured to communicate with a remote LNET using the router for that set that is topologically nearest in the 3D torus. As the distribution of routers is not perfectly even throughout the torus, static load-balancing is performed to balance the number of clients serviced by each router. This configuration would increase variability in performance as the bandwidth available to a job will have a greater dependency on its location within the torus. Smaller jobs will see reduced bandwidth as the set of routers in active use grows proportionally to the job size.

2. *Round-robin.* In this configuration, the servers and routers are again broken up into 32 sets as with Nearest-neighbor. Instead of using the nearest router, one of the 6 optimal routers is chosen in a round-robin fashion for each remote LNET. While this configuration allows for placement and avoids congestion on the InfiniBand fabric, it unnecessarily distributes I/O traffic throughout the torus.

3. *Projection.* In this configuration, each server gets a unique remote LNET. There are 192 remote LNETs, and the clients are configured with a single router for each remote LNET. This is effectively a projection of the OSS servers into the torus. It allows placement of clients to avoid congestion on the torus and avoids congestion on the InfiniBand fabric. Each client may see significant variances in distance to particular servers, but this is no worse than encountered in traditional direct-attached storage.

In all cases, I/O for a particular OSS is directed to a router attached to a leaf module in the InfiniBand fabric common to that OSS. This ensures that there is no traversal of the switch's internal fat-tree thereby avoiding the issue of link saturation and head-of-line blocking in the fabric.

We tested the three configurations with IOR and compared them to our baseline routed configuration. We tested

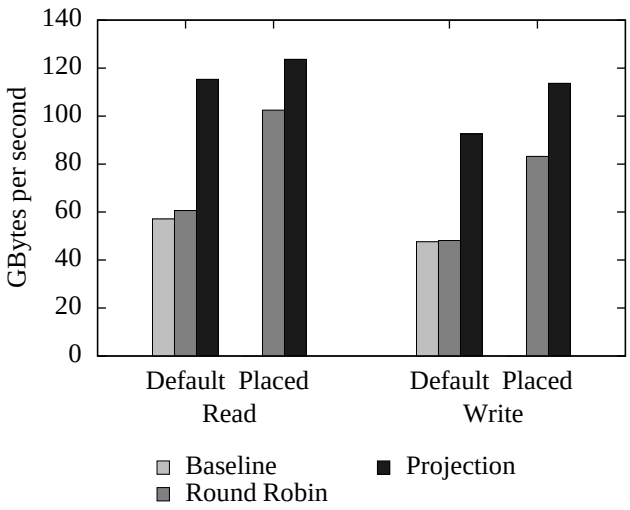

Figure 10: IOR write bandwidth on routed filesystem.

with the full machine available for optimization using IOR in file-per-process mode against 672 OSTs. Each rank wrote 8 GB of data with a transfer size of 8 MB to a file configured with a single stripe. Five trials were run and the maximum aggregate bandwidth is reported in Figure 10. Our baseline configuration, with its congestion issues in both the torus and InfiniBand fabric, resulted in 57.1 GB/s for reads and 47.6 GB/s for writes using the default placement of ranks. Round-robin does slightly better when not using our placement strategy, at 60.6 GB/s for reads and 48.1 GB/s for writes. Round-robin performs well when placement is used, yielding 102 GB/s reads and 83 GB/s writes. Projection yields the best observed results with 115 GB/s reads and 92 GB/s writes without placement, and 123 GB/s reads and 113 GB/s writes with placement. Nearest-neighbor is not reported due to issues in the placement calculations and limited dedicated system time for testing these configuration. We leave this issue open for further analysis and future work.

Using the Projection routing configuration with the combined storage of Spider – 1,344 OSTs – we were able to demonstrate aggregate bandwidths of over 244 GB/s for both read and writes. This result was generated by IOR in file-per-process mode. Each rank wrote 8 GB of data with a transfer size of 8 MB.

8 Conclusions

During the deployment of Jaguar XT5 at the OLCF, the performance of the direct-attached Lustre file system did not meet our expectations. A detailed analysis revealed that congestion on the Cray SeaStar torus was a primary source

of this discrepancy. To address this issue, a mechanism was developed to allow clients to be paired to specific I/O servers – a technique we refer to as "placement" – reducing the load on common XT5 torus links to avoid link saturation. Using this approach, 92% of the raw back-end storage performance was achieved at the file system level using the IOR synthetic benchmark. Furthermore, this level of performance (within 5%) was achieved even when the choice of clients performance I/O was significantly limited. These results indicate that placement is a viable mechanism to increase aggregate I/O performance, not only for large-scale application invocations that span the entire Jaguar XT5 platform, but also for smaller scale applications that may use a much smaller fraction of the available system.

The performance benefits of placement did not automatically follow when the Lustre file system was transitioned from a direct-attached configuration to a routed configuration in support of center-wide access to Spider. The naive configuration in which all 192 routers were assigned the same weight coupled with LNET's per-message round-robin selection policy prohibited our optimization strategy. In addition to negating the benefits of reduced congestion on the SeaStar 2+ network, this configuration introduced substantial congestion within the SION InfiniBand fabric. To regain opportunities for optimization via placement and eliminate InfiniBand congestion, we developed and evaluated three additional LNET routing configurations.

After weighing the benefits and drawbacks of alternate routing configurations, a configuration dubbed "Projection" was ultimately selected for our production computing environment. This configuration yielded over 90% of the raw back-end storage performance which compares quite favorably to the 92% achieved on the direct-attached file system. In conjunction with the placement strategies outlined in this paper, this configuration demonstrated aggregate performance of 244 GB/s for both reads and writes when using the entire Spider storage system.

References

[1] Innovative and Novel Computational Impact on Theory and Experiment (INCITE) Awards Fact Sheet. `http://science.energy.gov/~/media/ascr/pdf/incite/docs/2011_incite_fact%sheets.pdf`.

[2] D. Abts. The Cray XT4 and Seastar 3-D Torus Interconnect. `http://research.google.com/pubs/archive/36896.pdf`, 2010.

[3] D. Abts and D. Weisser. Age-based packet arbitration in large-radix k-ary n-cubes. In *SC*, 2007.

[4] B. Azeez, H. Kim, Y. Jin, and E. Kim. I/O Node Placement for Performance and Reliability in Torus Networks. In *International Conference on Parallel and Distributed Computing and Systems (PCDS2006), IASTED*, 2006.

[5] A. Bland, R. Kendall, D. Kothe, J. Rogers, and G. Shipman. Jaguar: The world's most powerful computer. In *Proceedings of the Cray User Group Conference*, 2009.

[6] R. Brightwell, T. Hudson, K. Pedretti, R. Riesen, and K. D. Underwood. Portals 3.3 on the Sandia/Cray Red Storm System. In *Proceedings of the Cray User Group Conference*, 2005.

[7] R. Brightwell, K. Pedretti, and K. D. Underwood. Initial performance evaluation of the Cray SeaStar interconnect. In *Proceedings of the 13th IEEE Symposium on High-Performance Interconnects*, 2005.

[8] P. Geoffray and T. Hoefler. Adaptive routing strategies for modern high performance networks. *High-Performance Interconnects, Symposium on*, 0:165–172, 2008.

[9] G. Shipman, D. Dillow, S. Oral, and F. Wang. The Spider center wide file system: From concept to reality. In *Proceedings of the Cray User Group Conference*, 2009.

[10] J. S. Vetter, S. R. Alam, T. H. D. Jr., M. R. Fahey, P. C. Roth, and P. H. Worley. Early Evaluation of the Cray XT3. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium*, 2006.