Blue Waters I/O Performance

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

Doug Petesch
Performance Group
Cray Inc.
Saint Paul, Minnesota, USA
dpetesch@cray.com

Abstract—**The Blue Waters system, installed at the National Center for Supercomputing Applications (NCSA), contains over 25000 compute nodes and hundreds of LNET routers and other service nodes spread across the largest Gemini network ever constructed. The external Lustre file systems are made up of 216 Lustre object storage servers (OSSs) and 1728 object storage targets (OSTs) and has produced more than 1 terabyte per second (TB/s) of sustained I/O bandwidth.**

This paper will describe, at a high level, the I/O infrastructure used on this system and present a variety of performance results. We will also talk about the challenges of benchmarking and understanding a system this large and complex.

I. Introduction

A. Brief Mainframe Inventory

The mainframe's compute infrastructure is composed of 25,712 compute nodes (22,640 XE and 3,072 XK) and 784 service nodes. Of the service nodes, there are 582 that act as LNET router nodes for the three Cray Sonexion-1600 file systems. The Gemini-based High Speed Network (HSN) is arranged in a 23x24x24 3-D torus.

B. Cray Sonexion-1600 File System Inventory

The Cray Sonexion-1600 Scalable Storage Units (SSUs) have bandwidth rates of 6GB/s peak and 5GB/s sustained. Each rack of the file system is composed of 6 SSUs. Each SSU is composed of two Object Storage Servers (OSSs) that act as each other's High Availability (HA) partner. Therefore, there are 12 OSSs in each rack. Each OSS controls 4 Object Storage Targets (OSTs). The Infiniband (IB) cabling within each rack is such that all OSSs on the left side are plugged into one Top-Of-Rack (TOR) IB switch while all OSSs on the right side are plugged into another TOR IB switch. See fig. 1.

Table I summarizes some of the characteristics of each file system.

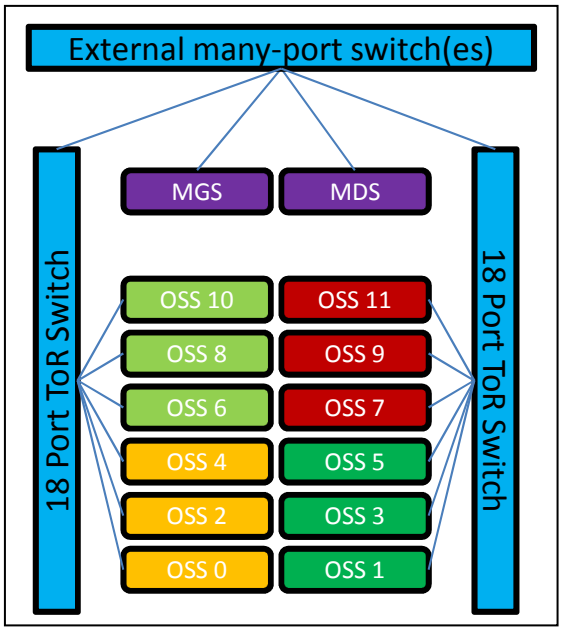

Fig. 1. Cray Sonexion-1600 Rack

TABLE I. Cray Sonexion-1600 Overview

	HOME	PROJECTS	SCRATCH	total
Racks	3	3	30	36
Metadata Serves	2	2	2	6
SSUs	18	18	180	216
OSSs	36	36	360	432
OSTs	144	144	1440	1728
Active Disks	1440	1440	14400	17280
Hot Spares	72	72	720	864
Peak Bandwidth	108	108	1080	1296
Sustained Bandwidth	90	90	900	1080

[a] Bandwidth values expressed in megabytes per second (MB/s)

C. *LNET Fine Grained Routing*

Fig. 2 shows a simple view of the LNET Fine Grained Routing (FGR) as implemented for the Blue Waters file systems. In order to provide the necessary LNET bandwidth to the OSSs, we created LNET groups containing four LNET router nodes and three OSSs. All four LNET router nodes reside on the same Cray XIO service blade. This is commonly referred to as a "4:3 FGR Ratio". This general method for creating LNET groups extends through all racks of all three file systems.

Table II shows the number of LNET routers assigned to the OSS LNET groups for the three file systems and their maximum bandwidth capabilities.

TABLE II. LNET ROUTER ASSIGNMENTS FOR OSS LNET GROUPS

	HOME	PROJECTS	SCRATCH	total
LNET Router Nodes	48	48	480	576
LNET Bandwidth	124.8	124.8	1248	1497.6

[a.] Bandwidth values expressed in megabytes per second (MB/s)

II. PERFORMANCE RESULTS

A. *Linear Scaling*

One of the design goals of the Cray Sonexion, and the implementation of it for Blue Waters, is scaling. The architecture of the file systems is such that there are independent building blocks that make up the whole. For example, each OSS of a SSU is independent of the other, each SSU in a rack is independent of the others, and each LNET FGR group is independent of the others. All these building blocks came together to create linearly scaling file systems.

Figures 3, 4, and 5 show the actual and expected scaling of the three file systems. The IOR benchmarking application was used to exercise one-third, two-thirds, and the entirety of each of the file systems. For the HOME and PROJECTS file systems, this means 1 rack, 2 racks, and 3 racks. For the SCRATCH file system, this means 10 racks, 20 racks, and 30 racks.

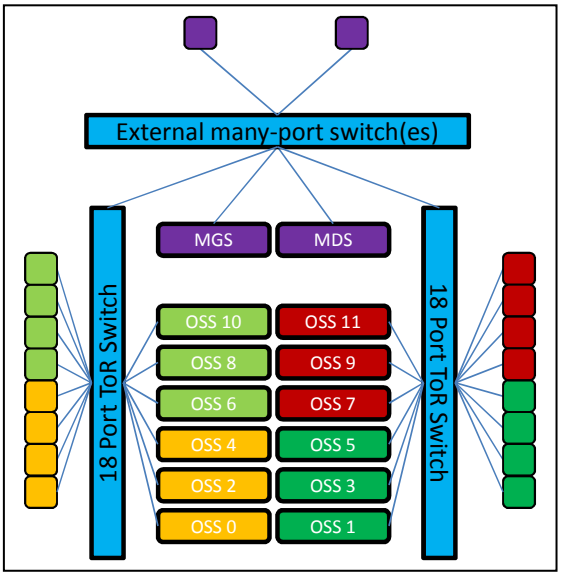

Fig. 2. Cray Sonexion-1600 Rack with FGR

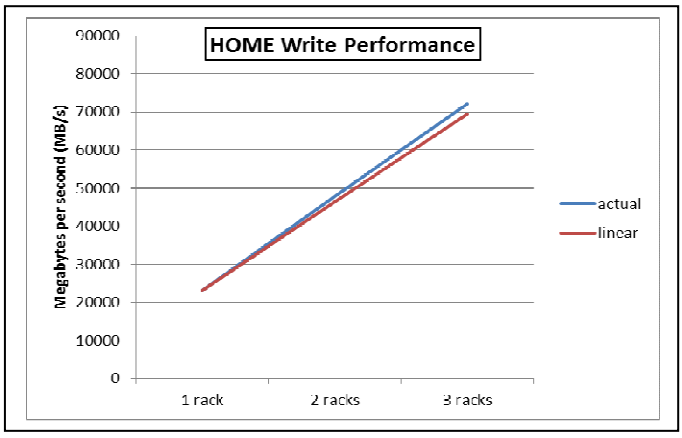

Fig. 3. Scaling of HOME Performance

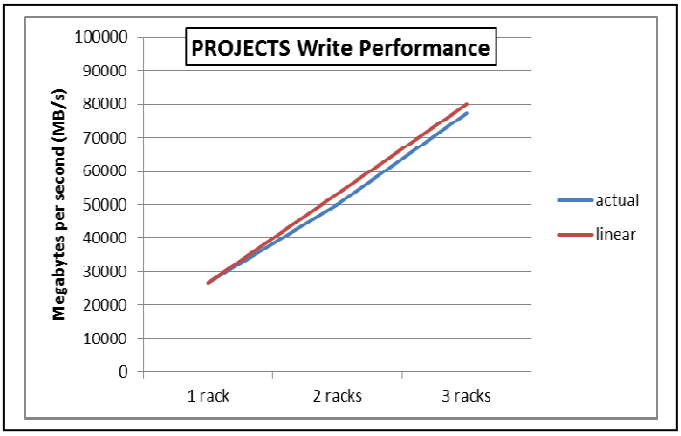

Fig. 4. Scaling of PROJECTS Performance

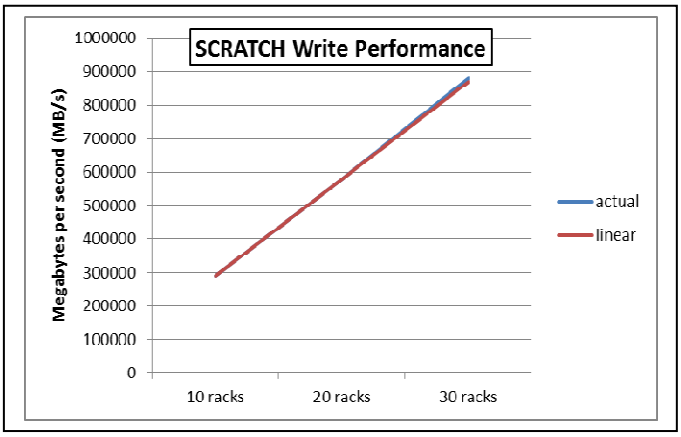

Fig. 5. Scaling of SCRATCH Performance

B. *Optimal Writing*

To verify maximum I/O bandwidth of the entire system, we constructed an IOR job that simultaneously wrote to all three file systems. The characteristics of this job are as follows and were based on our knowledge of how to optimally write to a Cray Sonexion-1600:

- POSIX file per process
- Buffered I/O
- 4 MB transfer size
- 12 files per OST for a total of 17,280 ranks
- 4 ranks per node for a total of 5,184 nodes
- Nodes randomly placed among all 25,712 nodes

Fig. 6 shows the results of running this job 10 times. The average across the 10 runs was 1.137 terabytes per second (TB/s).

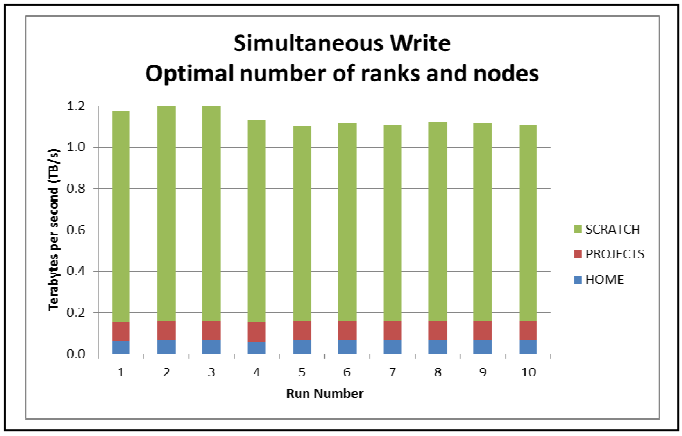

Fig. 6. Total Bandwidth, Optimal Writes

C. *Writing From All XE Nodes*

Another scenario of interest was the capability of every XE node in the system to write to all three file systems simultaneously. Of the 22,640 XE nodes, we chose to use 22,580 so the number of nodes assigned to each file system was proportional to the sizes of the file systems. The characteristics of this job were:

- POSIX file per process
- Direct I/O
- 16 MB transfer size
- 1 rank per node
- 1,880 nodes assigned to HOME
- 1,880 nodes assigned to PROJECTS
- 18,800 nodes assigned to SCRATCH

This arrangement resulted in 13.06 files per OST. Fig. 7 shows the results of running this job 10 times. The average across the 10 runs was 1.117 terabytes per second (TB/s).

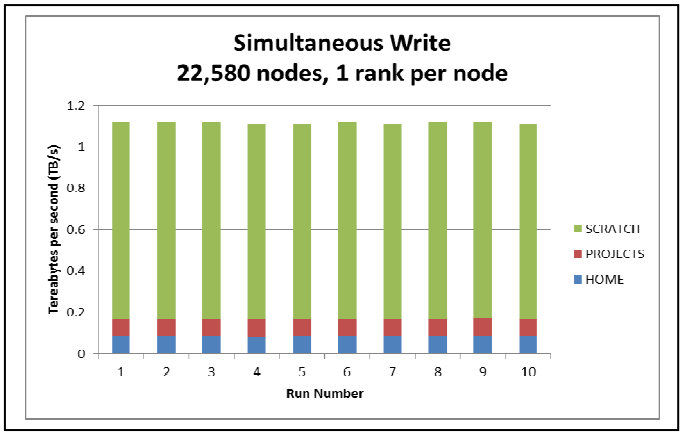

Fig. 7. Total Bandwidth, All XE Nodes Writing

D. Shared File MPI I/O

The Lustre version currently installed on the Blue Waters system is capable of striping files across a maximum of 160 OSTs. While this is not a limitation for the HOME and PROJECTS file systems (which have 144 OSTs), this does present an issue for the SCRATCH file system with its 1,440 OSTs. In order to still test the I/O capabilities of shared files, we created ten Lustre pools within the SCRATCH file system. Each pool was 144 OSTs (i.e., one-tenth of the size of the entire file system).

We constructed an IOR job that simultaneously wrote a single shared file to each of the ten Lustre pools defined within SCRATCH. Each shared file was striped as wide as the pool (i.e., 144 OSTs). Our desire was to engage as many of the XE nodes as possible so we assigned 2,260 nodes (with 1 rank per node) to each of the ten shared files.

Fig. 8 shows the results of 22,600 XE compute nodes writing to ten shared files in ten separate Lustre pools of SCRATCH. The average across three runs was 771 gigabytes per second (GB/s).

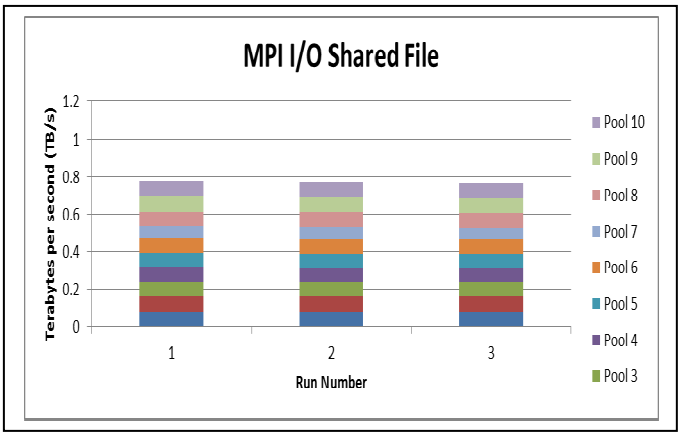

Fig. 8. MPI I/O Shared File Performance in SCRATCH Pools

III. Performance Challenges

Whenever a computing system is constructed that is as large and complex as Blue Waters, challenges will arise. Some challenges can be predicted while others cannot. As a member of the Department Of Bottleneck Discovery (i.e., the Cray Performance Team), it is our job to not only prove the performance capabilities of a system but, in the event that challenges arise, find explanations for, and, possibly, solutions to, those challenges.

A. Reading Striped Shared Files with MPI I/O

When writing our ten shared files in the ten Lustre pools within SCRATCH, we did not need to carefully match the number of nodes to some integral multiple of the stripe count in order to achieve acceptable performance. However, that relationship is important when trying to maximize writing and even more important when trying to maximize reading a shared file that is striped across multiple OSTs [1].

With our goal still being to use as many XE compute nodes as possible, we observed that using 22,600 for both writing and reading the ten shared files produced very poor read rates. Clearly, the biggest reason for this was our choice of the number of compute nodes compared to the stripe count of the shared files. Using a stripe count of 144 with 2,260 compute nodes means we did not evenly distribute the compute nodes across all OSTs (2,260 / 144 = 15.69). We decided to reduce the stripe count to 141 and use 2,256 compute nodes per shared file (2,256 / 141 = 16). This meant we would use 22,560 compute nodes which is 99.6% of all XE nodes.

The result of using 2,256 ranks per shared file and a stripe count of 141 rather than 2,260 ranks per shared file and a stripe count of 144 was the write rates improved by 12% and read rates more than doubled.

B. Fragmented File Systems

Issues with fragmented file systems have been around since the invention of the technology. Recognizing how the fragmentation affects I/O performance has also been around just as long. Many implementations of the Lustre file system suffer from these same issues. In particular, the highly used HOME file system exhibits these characteristics.

The HOME file system, as you can imagine, is a place where users store files of all sizes. Over a several month period of use, this file system sees many millions of files being created and deleted. As this happens, holes of various sizes are left behind and free space becomes scattered and non-contiguous. When writing and reading data from the HOME file system, it was quite apparent that we were experiencing decreased performance due to fragmentation. It is quite easy to analyze the files produced by IOR (or any other application) to see the number and size of pieces that make up an entire file.

Next in line with the HOME file system is SCRATCH. This file system had not been in use as long as HOME and typically does not get exposed to millions of small file creations and deletions. The effects of fragmentation on SCRATCH were not as noticeable as HOME.

Finally, the PROJECTS file system was the least used and, consequently, the least fragmented file system. Our testing showed that this file system consistently performed better (per SSU) than the other two file systems.

C. *Maximizing Read Performance On A Large Gemini*

Recall that the Gemini network of the Blue Waters system is a very large three dimensional torus. Data moving from point A to point B in this torus first travels along the X dimension, then along the Y dimension, then along the Z dimension. This is called "dimension ordered routing". Each Gemini has six directions in which to send data (X+, X-, Y+, Y-, Z+, Z-) in order to move the data toward the destination.

Also, recall that the LNET FGR implementation is such that there are three OSSs and four LNET router nodes in each LNET group. Therefore, there are 12 OSTs in each LNET group. Also, there are two Gemini serving the four LNET router nodes.

When performing IOR write testing to the SCRATCH file system, we would commonly create 12 files per OST. This meant that we would employ several thousand ranks. Depending on how many ranks per node we ran with, this could equate to hundreds or thousands of nodes. In the vast majority of our writing tests, it did not seem to matter how we allowed the nodes to be placed in the system because we often achieved good write rates. However, this was not true of node placement when reading the file system.

Fig. 9 depicts the compute node and LNET router node placement for a portion of a job that is reading files from the SCRATCH file system. What is being shown are the Geminis for the LNET router nodes for LNET group o2ib3037 (one of the 120 LNET groups for the file system) and the Geminis for the 48 compute nodes that had files on the OSTs of the three OSSs in this LNET group. This job had 4 files per OST and there are 12 OSTs in this LNET group and, therefore, 48 compute nodes (1 rank per node). The compute nodes were chosen to be randomly distributed among all XE compute nodes.

Fig. 10 depicts the Gemini-to-Gemini segments that were utilized by compute nodes and LNET router nodes in LNET group o2ib3037 when reading data from the file system. Data comes into the LNET router nodes from the IB connections to the OSSs and the LNET router nodes must then send data to the compute nodes. Data movement is X, then Y, then Z. The coloring indicates loading of the segment bandwidth. Green indicates that the amount of data attempting to be transported across that segment from all activity in the job is less than the maximum bandwidth of the segment. Red indicates that more bandwidth is attempting to be transported across the segment than the segment is capable of.

Notice, for this particular LNET group, there is only one compute node whose data does not have to first travel along the X dimension.

Fig. 11 depicts the same LNET group (o2ib3037) but with data being written to the file system rather than being read. Data is still routed X, then Y, then Z. Even though most of the data ends up moving along the Z dimension, that dimension is the fastest in the system and there is very little contention.

What lessons can be learned from these observations? An important lesson is that compute node placement matters. In order to further investigate the effects of compute node placement, we experimented with performing I/O to smaller subsets of the file system and requesting specific compute nodes that were near the LNET router nodes while taking advantage of the six data directions that the LNET router node's Geminis could use. Utilizing this kind of placement, read performance dramatically increased to the point where we were achieving full bandwidth rates from the OSSs in that LNET group.

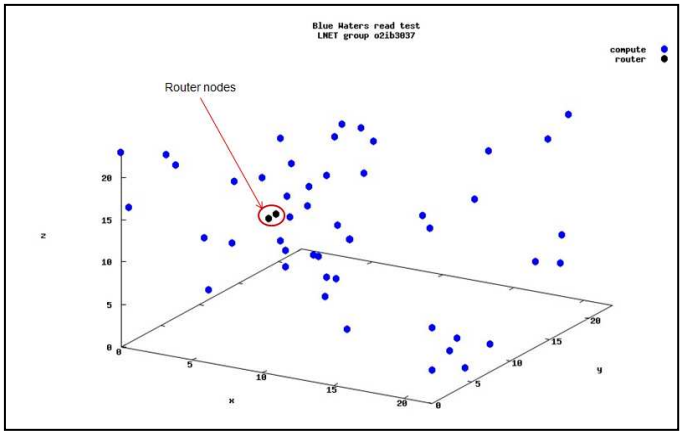

Fig. 9. LNET Group o2ib3037

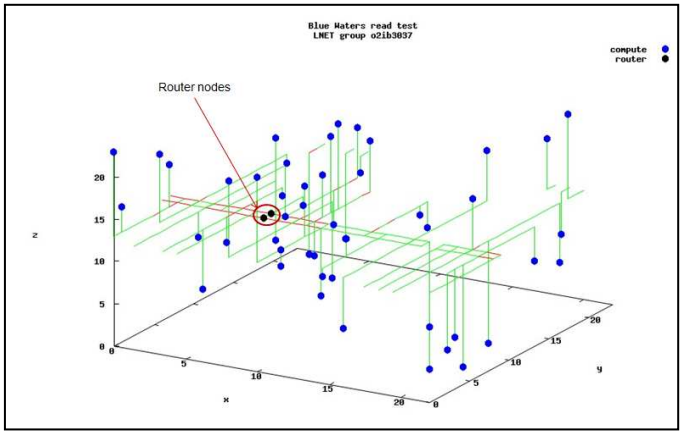

Fig. 10. LNET Group o2ib3037, Reading

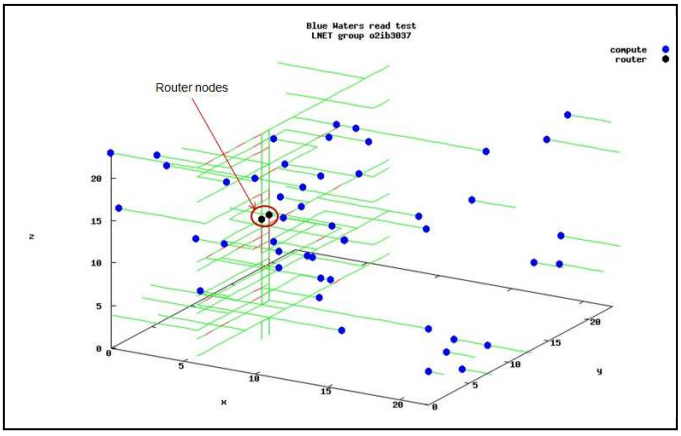

Fig. 11. LNET Group o2ib3037, Writing

IV. Future Research Opportunities

A. Compute Node Placement for Improved I/O

One of the larger challenges described previously in this paper was that of maximizing the Gemini bandwidth of the LNET router nodes. Due to the dimension ordered routing of Gemini, there are many cases where most of the data movement at the LNET router nodes is concentrated in only two of the six channels. When writing data to the file systems, the Z+ and Z- channels are more highly used than X+, X-, Y+, or Y-. When reading data from the file systems, the X+ and X- channels are more highly used than Y+, Y-, Z+, or Z-.

If an application is constructed such that only certain ranks are performing I/O, it would be advantageous to place these ranks in a three dimensional halo around the LNET router nodes so that all six channels of the LNET router node's Geminis are utilized.

Furthermore, there are bandwidth disparities between links in the X, Y, and Z directions. These bandwidth capability differences could also be accounted for when creating the three dimensional halo of I/O ranks around the LNET router nodes. For example, more of these ranks could be in the same Z planes of the LNET routers since the Z dimensions have more bandwidth capability than the X or Y.

If we're going to attempt to place applications (or parts of applications) near LNET router nodes, the placement algorithm also needs to know where the files exist that the application needs. For example, suppose an application is placed near LNET router node X and so writing checkpoint files to the OSTs assigned to X is optimized. If that application is restarted and placed near LNET router node Y, the checkpoint files that the application wants to read are still associated with LNET router node X and might not be optimally accessed. However, reads of those checkpoint files would be optimized if the job could inform the placement algorithm that its files are associated with LNET router node X.

This type of file access information simply does not exist on the mainframe. Along with making the placement algorithm aware of LNET router locations, it would also be necessary to define the association between OSTs (where files exist) and their LNET routers.

B. OST Fragmentation and Physical Positioning

Not only is fragmentation of the data on OSTs a factor in achieving good I/O rates, so is the physical placement on the spinning media. It has been observed that I/O rates on the outer edges of the disks (where there is more data per track) are significantly faster than rates on the inner edges (where there is less data per track). Investigating methods of coalescing fragmented blocks of files could be combined with methods of placing seldom used files (or less performance sensitive files) toward the slower regions of the OSTs.

V. References

[1] Getting Started on MPI I/O, Cray publication S-2490-40.

The source for the IOR benchmark application, version 2.10.3, is available at "http://sourceforge.net/projects/ior-sio".

The source for the IOR benchmark application, version 3.0.0, is available at "https://github.com/chaos/ior".