I/O Performance on Cray XC30

Zhengji Zhao[1)], Doug Petesch[2)], David Knaak[2)], and Tina Declerck[1)]
[1)] National Energy Research Scientific Center, Berkeley, CA
[2)] Cray, Inc., St. Paul, MN

Email: {zzhao, tmdeclerck}@lbl.gov; {dpetesch, knaak}@cray.com

***Abstract*—Edison is NERSC's newest petascale Cray XC30 system. Edison has three Lustre file systems deploying the Cray Sonexion storage systems. During the Edison acceptance test period, we measured the system I/O performance on a dedicated system with the IOR benchmark code from the NERSC-6 benchmark suite. After the system entered production, we observed a significant I/O performance degradation for some tests even on a dedicated system. While some performance change is expected due to file system fragmentation and system software and hardware changes, some of the performance degradation was more than expected. In this paper, we analyze the I/O performance we observed on Edison, focusing on understanding the performance change over time. We will also present what we have done to resolve the major performance issue. Ultimately, we want to detect and monitor the I/O performance issues proactively, to effectively mitigate I/O performance variance on a production system.**

I/O performance; IOR; Lustre; Sonexion; OSTs; I/O bandwidth; read rate

I. INTRODUCTION

Edison [1], a Cray XC30 system, is NERSC's newest supercomputer, with a peak performance of 2.57 peta-flops. It features Intel's dual-socket 12-core Ivy Bridge processors; Cray's Aries interconnect with Dragonfly topology, and the Cray Sonexion storage system. Edison has three Lustre file systems providing 7.5 PB of online disk space with 168 GB/s peak IO bandwidth with a total of 336 OSTs.

As part of the acceptance tests, at the end of August 2013, we performed Lustre file system performance tests on a dedicated system using the IOR benchmark code from the NERSC-6 benchmark suite [2]. The IOR benchmark tests included the Posix file per process and MPI-IO shared file I/O tests with different access patterns that were designed to represent the NERSC I/O workload. About four months later (mid December, 2013), we ran the exact same set of I/O tests in dedicated mode again after the system went through several major hardware and software upgrades. We observed significant I/O performance changes. While most of the performance changes were under +/- 20% relative to the August results, some of the tests, especially MPI-IO shared file read tests, were more than 70% slower or faster than the August results. We noticed that the MPI-IO tests with larger transfer sizes ran about 70% faster, but that MPI-IO with a small transfer size was slower by more than 70%. While the increased performance was not a concern, the 70% decrease in performance of the MPI-IO with a transfer size of 10k (denoted as MPI-IO 10k hereafter; this is one of the tests included in the NERSC-6 benchmark suite) caught our attention. While some performance variation, perhaps as much as +/- 20%, may be expected due to system hardware and software changes and from file system fragmentation due the production I/O load, the 70% performance degradation was considered to be very serious and in need of further investigation.

From the Lustre file system point of view, the MPI-IO 10k test is not optimal because of the relatively small transfer size; in general, users should avoid such sizes. However, as shown by data collected by Darshan, an I/O characterization tool [3], about 50% of all I/O operations on Hopper, NERSC's large Cray XE6 system were unaligned, or small I/O operations with transfer sizes that are much smaller than the Lustre block size [4]. Understanding the performance variance of the MPI-IO 10k test after system changes, therefore, has practical meaning to NERSC users. In this paper we will present what we have done to understand this performance issue, and what we have learned from our debugging efforts.

Ultimately, we want to detect and monitor I/O performance issues proactively to effectively mitigate I/O performance variance on a production system, which can be very disruptive to user production workflows. While I/O performance variation from contention for the limited I/O resources is not easily avoidable under the current system settings and configuration, there are still some steps that can be taken to mitigate unnecessary contention by promoting good I/O practices. Moreover, we observed that when users report huge I/O time variations at NERSC, it is often when the file systems misbehave or perform poorly for a reason that requires further investigation. Therefore, it is critical to be able to monitor the file system health and performance. We will discuss what monitoring tools are in place on Edison, and will describe several efforts to detect and mitigate I/O performance variation.

II. THREE LUSTRE FILE SYSTEMS ON EDISON

Edison has three Lustre scratch file systems built on the Cray Sonexion 1600 storage system [5], configured in the ratio 2:2:3 for capacity and bandwidth. Table 1 shows the configuration of the three Lustre file systems. The first two file systems (FS1 and FS2) have 2.16 PB disk space and 48 GB/s aggregate peak I/O bandwidth with 12 Scalable

Table 1. The configuration of the three Lustre file systems on Edison

	Size (PB)	Aggr. Peak I/O Bandwidth (GB/s)	# of SSUs	# of OSSs	# of OSTs
FS1 or /scratch1	2.1	48	12	24	96
FS2 or /scratch2	2.1	48	12	24	96
FS3 or /scratch3	3.2	72	18	36	144

Storage Units (SSUs) and 96 OSTs each; the third file system (FS3) has 3.2 PB disk space and 72 GB/s peak I/O bandwidth with 18 SSUs and 144 OSTs. Each OST contains 8 data disks and 2 parity disks (dual-ported 3.5 inch 3TB NL-SAS 7,200 RPM disk drives) configured as a RAID 6 array; two dual-ported 3.5 inch 100GB SSDs drives, which are configured as a shared RAID 1 array, and are partitioned and used for the MDRAID and the file system journals; and two spare 3TB NL-SAS disk drives.

The default Lustre stripe size is 1MB and the default stripe counts are 2, 2, and 8 on the three file systems, respectively. Users are distributed to the first two file systems evenly in a round-robin fashion. The third file system is reserved for users who need large I/O bandwidth, and access is granted by request. Therefore, one may expect a different I/O usage pattern on FS3 while a similar I/O load and usage pattern may be expected on FS1 and FS2. These file systems are subject to purging. Files that are older than 12 weeks (defined by last access time) are removed.

III. BENCHMARK CODES AND TESTS

A. IOR

IOR [6] is a commonly used I/O benchmark program for testing performance of parallel file system. It provides multiple interfaces and options for access patterns that can be used to produce a wide variety of I/O workloads. IOR was one of the codes included in the NERSC-6 benchmark suite that resulted in the Hopper acquisition and was also used for Edison procurement that is internally coded as NERSC-7 project. The IOR version we used was 2.10.0.

B. IOBUF Library

IOBUF [7] library is an I/O buffering library provided by Cray. It can reduce the I/O wait time for programs that read or write large files sequentially. IOBUF intercepts I/O system calls such as READ and OPEN, and adds a layer of buffering, thus improving program performance by enabling asynchronous prefetching and caching of file data. IOBUF was used in some of our IOR benchmark tests. The behavior of IOBUF can be controlled by a number of environment variables. The environment variable, IOBUF_PARAMS, can be used to control the buffer sizes and files to selectively apply buffering. The IOBUF versions we used were 2.04 and 2.05.

C. Instrumented IOR

The standard IOR benchmark reports net read and write rates at the end of the runs. To debug the I/O performance issue, we used an instrumented version of IOR [8] in our debugging runs, which reports the "instantaneous" bandwidth during a benchmark run. The instantaneous I/O bandwidth was obtained by summing up the data moved per second by each processor core. The instrumented IOR was based on IOR 2.10.3.

D. IOR Benchmark Tests

We used the IOR tests with different interfaces and I/O access patterns that were designed to represent the NERSC I/O workload. The tests include Posix file per process and MPI-IO shared file I/Os with transfer sizes of 10,000, 1,000,000 and 1,048,576 bytes using a number of processor cores that are proportional to the number of available OSTs in each file system. Table 2 shows these tests in more detail. For simplicity, we will use the short names, PosixFpP 10k, PosixFpP 1m1, PosixFpP 1m2, MPI-IO 10k, MPI-IO 1m1, and MPI-IO 1m2 to denote these benchmark tests hereafter. In all these tests, each node writes or reads 96 GB data (4 GB per core) either from a single file or multiple files, which is 1.5 times the available per-node memory, 64 GB. This is to eliminate cache effects. As one of the official run rules defined in the NERSC-6 benchmark suite (also by IOR default), in all of the IOR benchmark tests mentioned above, the IOR program does the write and the read tests within a single aprun invocation. It opens the file(s), writes, then closes the file(s); and reopens the file(s), reads, and closes the file(s) again. To defeat buffer caching for read after write, we made each task read its neighbor's data from different nodes (reorderTasks=1). The IOR program reports the net write and read rates at the end of the run. We will often refer to the read test in this standard benchmark run as the "read-after-write" test. There are many flexible ways to run IOR. For example, in some of the investigations, we made the IOR program read existing files that were written by the previous standard benchmark runs. We will refer the read test in this case (read alone) as the "re-read" test throughout the paper.

To improve I/O performance, we used the IOBUF libraries with some of the tests (See Table 2), especially with the MPI-IO 10k test. We disabled the collective buffering (CB) in the MPI-IO 10k test to use the IOBUF library, which improves this specific test performance significantly. Since the IOBUF library provides a buffer to aggregate small transfers to a bigger size before the file system actually "sees" any of these file operations, the MPI-IO 10k test + IOBUF with 1,000,000 byte buffer is equivalent to the MPI-IO 1m1 test with collective buffering (CB) disabled. Therefore, we ran the MPI-IO 1m1 test without CB in some cases when investigating the MPI-IO 10k performance issue for simplicity.

IV. I/O PERFORMANCE ON EDISON

Edison was delivered to NERSC at the end of June 2013. About one month after installation, site integration, configuration and staff tests, we enabled the first batch of early users. At the end of August we enabled all (approximately 2,000) NERSC users. We carried out the official I/O acceptance tests on Aug. 23, 2013 using a

dedicated system. Fig. 1 shows a part of the IOR performance results from the acceptance tests. Each test was run two or three times. Among the three file systems, FS2 (green symbols in Fig. 1) and FS3 (orange) were almost clean (1% full), while FS1 (blue) was about 30% full. There were 72, 72, and 144 OSTs in the three file systems, respectively, at that time. (As we will describe in the next section, three more SSUs were added to each of the first two file systems after the August acceptance tests, so that today Edison has 96, 96, 144 OSTs in its three file systems, respectively). Fig. 2 (a) shows three representative IOR tests selected from Fig. 1, but instead of showing the bandwidth for the whole file system, we show bandwidth per SSU. Fig. 2 (b) shows the coefficient of variation (COV) of the I/O rates for the three selected tests.

We can see that with the two clean file systems, FS2 and FS3, IOR achieved aggregate bandwidths of about 36 GB/s and 72 GB/s, respectively, which are 100% of the theoretical peaks. With 30% full FS1, IOR achieved around 80% of the peak bandwidth. We can see that the I/O bandwidth scales

Table 2. IOR benchmark tests run on Edison. Note the number of cores used and the size of the files are for the upgraded file systems, which have 96, 96 and 144 OSTs in FS1, FS2, and FS3, respectively. FS1 and FS2 had 72 OSTs each before the upgrade. The number of cores and the size of the files used in the acceptance tests were 75% of the values shown in this table for FS1 and FS2.

Transfer Size		10,000 bytes	1,000,000 bytes	1,048,576 bytes
Posix File Per Processor	Test Name	PosixFpP 10k	PosixFpP 1m1	PosixFpP 1m2
	# of Nodes/Cores Used	FS1: 32/768; FS2: 32/768; FS3: 48/1152		
	Aggregate File Size	FS1: 3.1TB; FS2: 3.1TB; FS3: 4.6TB		
	# of Files (4GB each)	FS1: 768; FS2: 768; FS3: 1152		
	IOBUF_PARAMS	count=2:size=32m:direct		
	Lustre Striping	lfs setstripe -s 1m -c 1		
	Other IOR options	useO_DIRECT=0 reorderTasks=1 fsync=1 intraTestBarriers=1		
MPI-IO Shared File	Test Name	MPI-IO 10k	MPI-IO 1m1	MPI-IO 1m2
	# of Nodes/Cores Used	FS1: 96/2304; FS2: 96/2304; FS3: 144/4608		
	Aggregate File Size	FS1: 9.2TB; FS2: 9.2TB; FS3: 13.8 TB		
	# of Files	1		
	IOBUF_PARAMS	For MPI-IO 10k: count=1:size=1000000: prefetch=0 For MPI-IO 1m1 and 1m2: IOBUF was not used		
	MPI-IO Hints	For MPI-IO 10k: cb_romio_read=disable cb_romio_write=disable For MPI-IO 1m1 and 1m2: cb_romio_read=enable cb_romio_write=enable		
	Lustre Striping	For MPI-IO 10k: lfs setstripe -s 1m -c -1 For MPI-IO 1m1 and 1m2: lfs setstripe -s 4m -c -1		
	Other IOR options	collective=1 reorderTasks=1 fsync=1 intraTestBarriers=1		

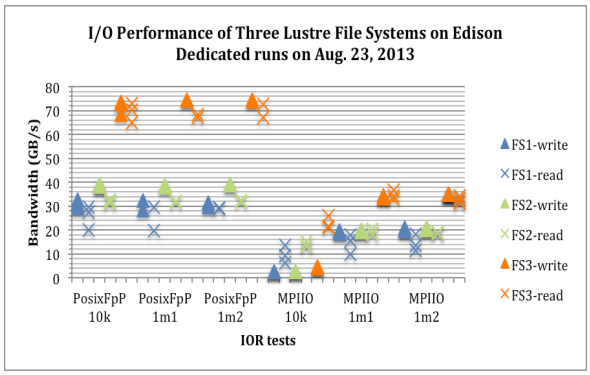


Figure 1. The IOR performance results observed in August 2013 on Edison's three file systems. Solid triangles and x's denote the write and read rates. The blue, green and orange symbols are for FS1, FS2, and FS3, respectively.

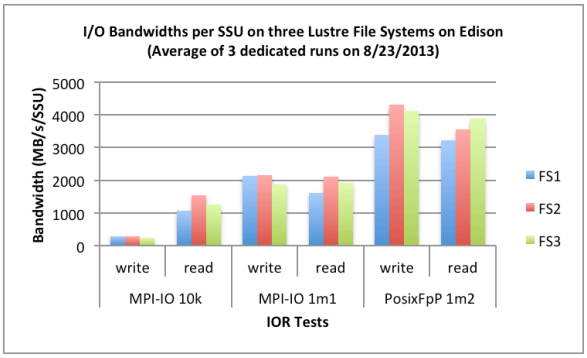


Figure 2. (a) The bandwidth per SSU obtained with three selected IOR benchmark tests with both Posix file per process and MPI-IO shared file tests on the three Lustre file systems on Edison.

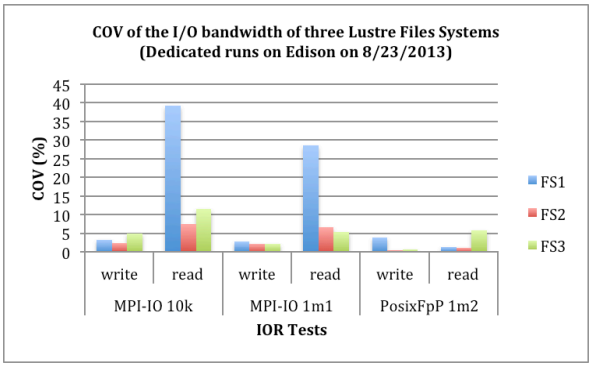


Figure 2. (b) The Coefficient of Variation (COV) of the read and write rates for the three selected IOR benchmark tests.

almost linearly up to 144 OSTs, which is the largest number of OSTs available in a single Lustre file system on Edison. The maximum bandwidths per SSU were about 4000 MB/s for write and slightly less for read (see Fig. 2 (a) PosixFpP 1m2 tests on FS2 and FS3). It should be noted that all our tests are "fixed data" IOR runs, which usually report lower bandwidths than the "fixed time" IOR tests [8].

We can also see that the performance variation from run to run was up to 40% for the read tests on FS1 even under dedicated conditions, while it was under 10% on the other two clean file systems. Since FS1 was 30% full, the large performance variation may be related to file system fragmentation and physical positions of the files relative to the slower or faster end of the disk drive. File fragmentation usually affects reads more than writes. As noted in [8], the physical positions of a file could result in 15-30% of performance variation even on a dedicated file system. However, it was not proven that fragmentation and physical positions of files alone had contributed for the 40% variation in our case. We do not exclude the possibility of FS1 being affected by some undetected undergoing file system events at that time as well. Unfortunately, we did not save the original files (multi-TB in size) to investigate this further.

V. I/O Performance Changes over Time

After the acceptance tests, Edison went through several major hardware and software upgrades. See Table 3 for details. Both FS1 and FS2 were expanded with three more SSUs (24 OSTs). There were multiple upgrades of the Cray Linux Environment (CLE) and Lustre client, as well as the Cray Developer Toolkit (CDT), which contains the MPI and IOBUF libraries, and compilers used by IOR. In addition, we opened up all the three file systems to users, so that FS2 and FS3 became loaded with user production I/O files. Since we enforce purging, the file system usage was under 30% on all the three file systems [See Fig. 3]. There were about 1,000 active users on each of the first two file systems and fewer than 40 active users (non-support staff) on FS3 as of now.

Table 3. Edison file system expansions and CLE /Lustre upgrades.

Date	FS1 (# of OSTs)	FS2 (# of OSTs)	FS3 (# of OSTs)	CLE/Lustre Versions
Aug 1, 2013	72	72	144	5.0.UP03/2.3.0
Dec 6, 2013				5.1.UP00/2.4.0
Dec 16,2013		96		
Jan 17, 2014	96			
Mar 11, 2014				5.1.UP01/2.4.1

Due to all the changes mentioned above, we ran the same set of the IOR tests again last December in dedicated mode. Fig. 4 shows the I/O performance change we observed in December relative to the August results. We can see that most of the performance changes were under +/- 20% relative to the August results. However, some of the tests, especially MPI-IO shared file read tests, were more than 70% slower or faster than the August results. The two MPI-IO read tests with relatively large transfer sizes, MPI-IO 1m1 and MPI-IO 1m2, were faster by 70%, while the MPI-IO 10k read test, which has a relatively small transfer size, was slower by more than 70% across all three file systems. In what follows, we attempt to uncover the basis for this change.

A. *Debugging the MPI-IO 10k performance slowdown*

To understand this performance degradation, we took a few dedicated system times between December 2013 and

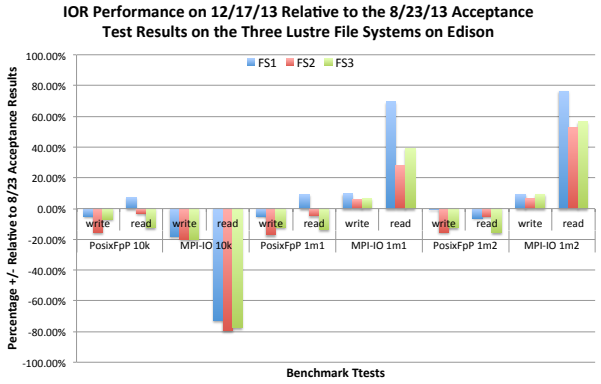


Figure 4. The IOR performance tests on December 17, 2013 on dedicated Edison system. The performance data was shown is relative to the last August results.

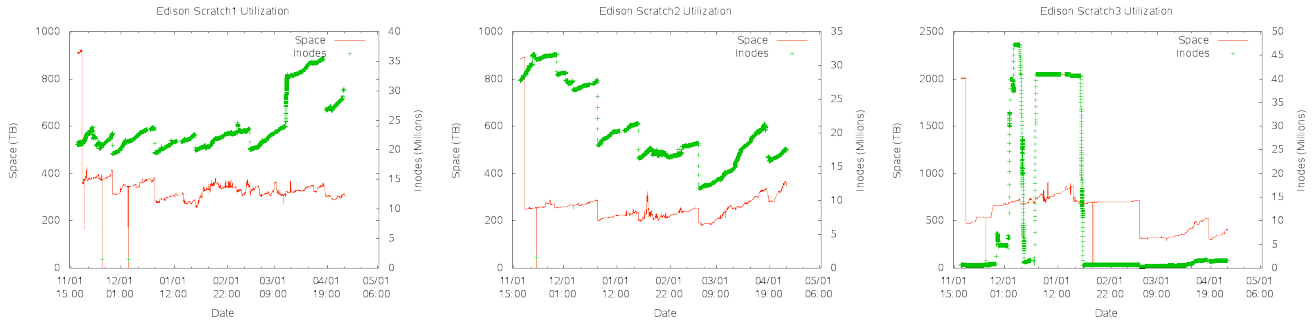


Figure 3. This figure shows the Lustre file system usage on Edison. The third file system has a larger average file size compared to the first two file systems.

March 2014. We ran the MPI-IO 10k tests on the dedicated system a few more times, and conducted a series of debugging runs. We started by confirming that this performance slowdown is indeed a persistent issue. (See the READ tests in Fig. 5). We looked at a number of issues; six of them are described below.

1) File Fragmentation and physical positions on the disk drive: As we have mentioned earlier, file fragmentation and physical positions of the files on the disk drives could result in significant performance variation, especially for read tests. However, these don't seem to account for the persistent 70% performance slowdown. As mentioned earlier, the write and read rates in a standard IOR benchmark run (read-after-write) were obtained in a single aprun invocation. We discovered that when running the read test alone (re-read), i.e., making the IOR program read an existing file that was generated by a previous MPI-IO 10k standard run, the read rates were consistently much improved compared to the read-after-write test, and sometimes they were even comparable to the August results. See Fig. 5 for the RE-READ tests. This rules out the file fragmentation and physical position on the disk drive as the cause of the 70% slowdown, since the read-after-write and the re-read tests both read the same file and so have the same file fragmentation and physical position.

2) Programming environment changes: When we run benchmark tests, we usually compile codes with the current default software versions. However, since we fortuitously retained some previous binaries, we had the opportunity to compare performance from tests compiled under CDT 1.10 and CDT 1.06. Fig. 6 shows the read rates of the read-after-write tests for the MPI-IO 10k test that were run on different dates. The August acceptance tests used the binary built on 7/19 under CDT 1.06. Note that the last two runs in the figure (purple and orange), were the runs on 12/30 with the two binaries built on 12/15 and 7/19, respectively. They were very similar, which suggests that none of the Intel compiler, MPICH, and IOBUF library changes were the cause of the 70% MPI-IO 10k read slowdown.

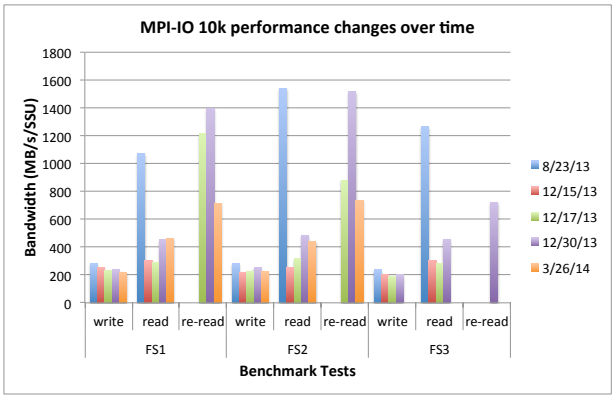


Figure 5. The MPI-IO 10k dedicated performance change over time.

3) Characteristic performance profile of the MPI-IO 10k read test: Debugging an I/O performance problem observed in the dedicated tests is difficult on a production system because there is no way to separate performance variation due to contention from other users. However, since Edison is a petascale system that delivers 3.2 million core-hours to users daily, it was very difficult for us to obtain significant dedicated time. (Even on a dedicated system, I/O performance could easily vary by about 40% from run to run.) In general, one can run the same tests multiple times to mitigate variation effects; however, it was not very practical for us to do so with this specific test, as it takes a bit more than two hours to complete the standard MPI-IO 10k test (write and then read) during a limited dedicated system reservation hours. So it was highly desirable for us to be able to reproduce the performance issue on a smaller internal Cray R&D XC30 system.

Reproducing an I/O performance at scale on a smaller internal Cray R&D XC30 was not straightforward, as the exact benchmark test can not be run due to the smaller number of OSTs and compute nodes available (and may also be subject to other slight architectural differences). In addition, the net I/O bandwidths reported by IOR alone were not sufficient to tell if a problem at scale is reproduced at a smaller scale or not, because large I/O variation from run to run may occur even on dedicated system (we have seen up to 40% variation). A key advance in the investigation was made last March when we learned that the read-after-write and the re-read tests have very different performance profiles using an instrumented version of IOR. Fig. 7 shows the read profiles we observed on the Cray R&D XC30 system (with 32 OSTs) for the read-after-write and the re-read tests. As shown in Fig. 7, while the read rate steeply declines in the read-after-write test, it stays fairly constant after an initial drop in the re-read test. This profile was consistently seen later on various systems as long as the file size per node was at least as large as the memory per node and multiple OSTs were used. The read rate drops even more steeply when the larger file size per node and larger number of OSTs were used.

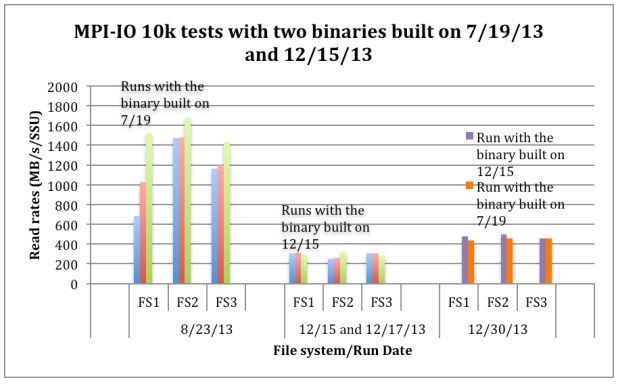


Figure 6. This figure shows the MPI-IO 10k read rates (read-after-write) measured by the two binaries that were built for the August acceptance tests (built on 7/19) and for the December retest (built on 12/15/13).

In March of 2014, we got dedicated system time on Edison and ran a number of tests. First, we needed to confirm if the same profiles also occurred on Edison. Fig. 8 shows the I/O profiles we obtained using the instrumented IOR with the MPI-IO 10k test on Edison FS2. It shows two plots, each with three curves: a write, a read, and a re-read. The first read is the distinctive read-after-write profile and the second read is the re-read profile. We can see that the read profiles on Edison are consistent with what we observed on the internal Cray R&D XC30 system (Fig. 7). With more OSTs and file size per node being used, a steeper read rate drop is observed. Note that Fig. 8 has the write, read, and re-read data all in sequence which makes the curves more compressed than in Fig. 7

Then we used the instrumented IOR for a number of scaling experiments on Edison FS3 using the MPI-IO 1m1 test with collective buffering disabled as seen in Fig. 9. Fig. 9 (a) has three curves. These three tests were all done on 64-GB nodes with file sizes equal to 16, 32, and 64 GB per node. All three tests were read-after-write, but only the read data are plotted for clarity. The first test completed in about 100 seconds, with performance fairly constant throughout. The second test, with a file twice the size as the first test, completed in about 200 seconds, and again, with performance fairly constant. The third test, with file size twice the size as the second test, but significantly, also the size of all the compute node memory, did not have a constant rate but declined dramatically throughout the test. This confirmed that the performance profile seen on the smaller Cray system was indeed happening on Edison. Fig. 9 (b) shows what happened when the file size, the number of OSTs, and the number of ranks were all doubled compared to the third test in Fig. 9(a). Performance was

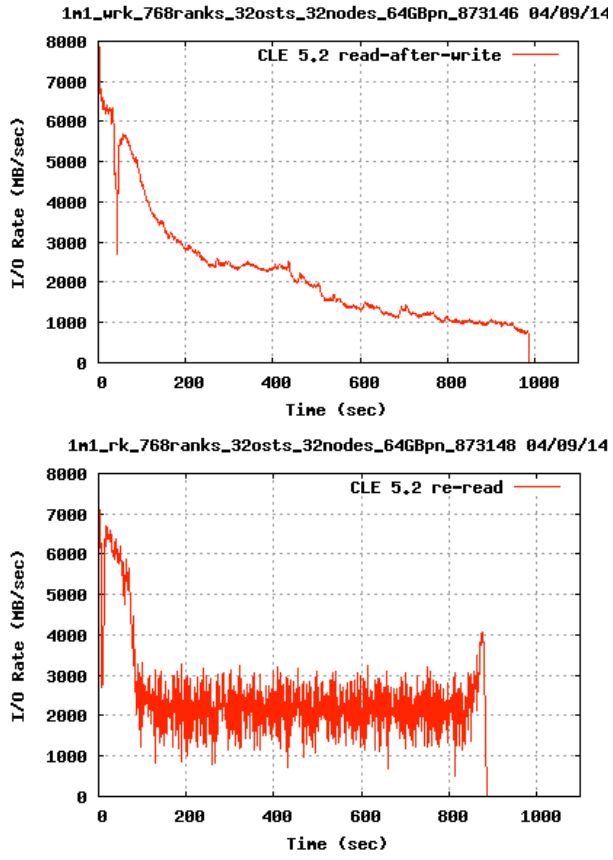


Figure 7. The distinctive read profiles for the read-after-write (upper plot, write rates are not shown) and the re-read test (lower plot) observed on the internal Cray R&D XC30 system with dual socket 12-core Ivy Bridge processors with 64 GB per node memory (dedicated). 32 nodes, 32 OSTs and 64GB/node file size were used in the tests. Instead of the MPI-IO 10k test, the equivalent MPI-IO 1m1 test without collective buffering was run in the tests.

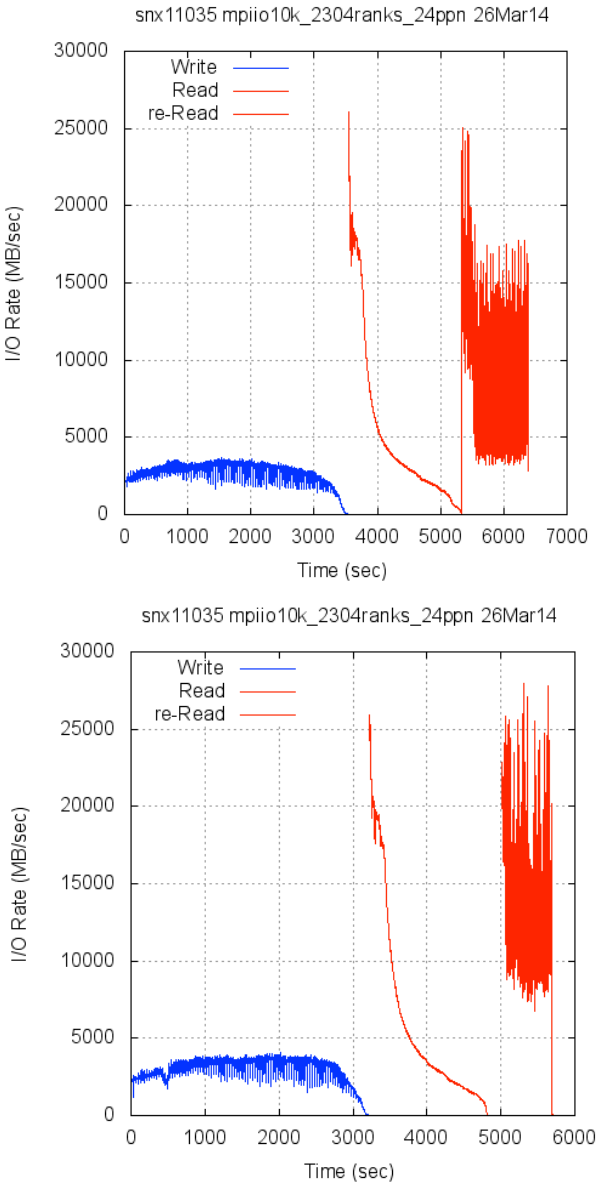
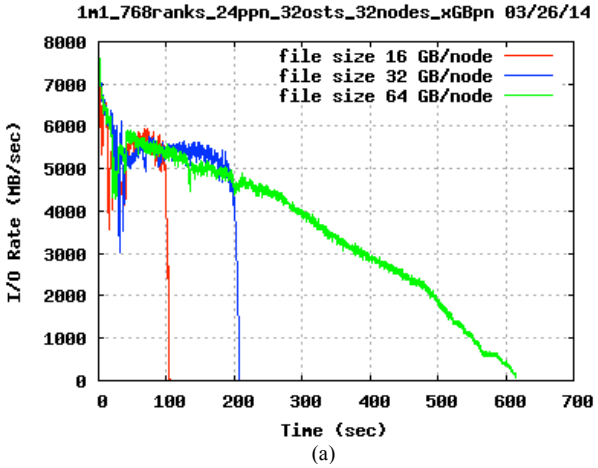


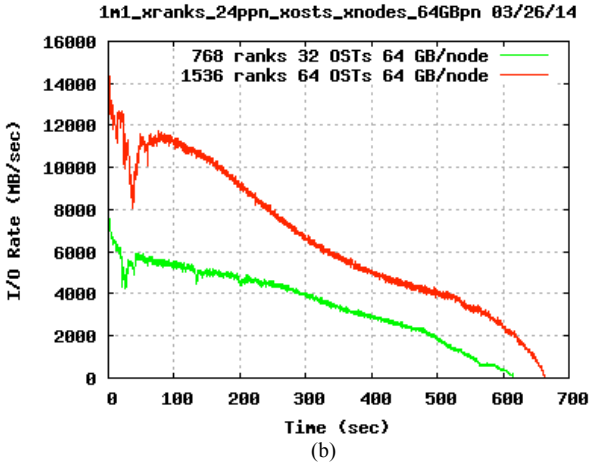
Figure 8. The I/O rates measured by the instrumented IOR for the two MPI-IO 10k jobs that were run on Edison FS2 in dedicated mode on 3/26/2014. The second read curve shows the read rate from the re-read job.

almost doubled, due to having twice as many OSTs, but the profile has the same overall shape.

These scaling tests strengthen the belief that the problem can be reproduced on a smaller system and that the profile, rather than the net performance, is the signature of the problem. Since this read profile as reported by instrumented IOR was to play a critical role, we compared that profile with the file system performance profile from the Lustre Monitoring Tool (LMT) [10]. Fig. 10 shows the LMT data for the same two jobs that were shown in Fig. 8 (MPI-IO 10k runs on FS2). They agree, except that the IOR plot shows a more zigzag shape, because it used one-second bins to collect data, while the LMT used five-second bins to calculate the rates over time. Otherwise, the match in the profiles confirms that instrumented IOR is reporting the true performance.

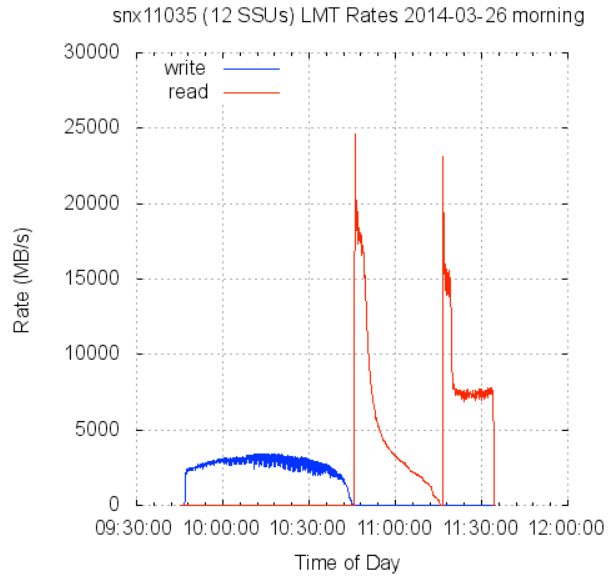


(a)



(b)

Figure 9. The scaling tests with the MPI-IO 1m1 test without CB on FS3 in dedicated mode. The MPI-IO 1m1 test with CB disabled is equivalent to the MPI-IO 10k test using IOBUF library with a buffer size of 1,000,000 bytes. The upper panel shows the read rates when increasing the file sizes. The number of cores and OSTs used were kept constant, 768 cores and 32 OSTs. The lower panel shows the read rates when increasing the number of OSTs used while keeping the file size per OST constant. Note, read rates shown here were from the read-after-write tests, not from the re-read tests.



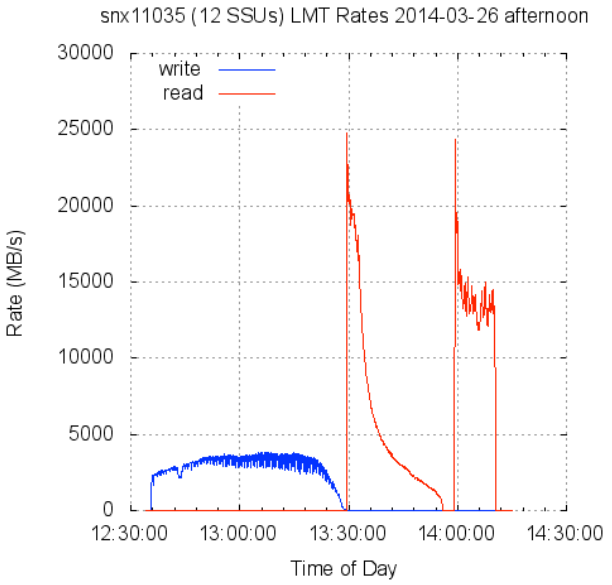
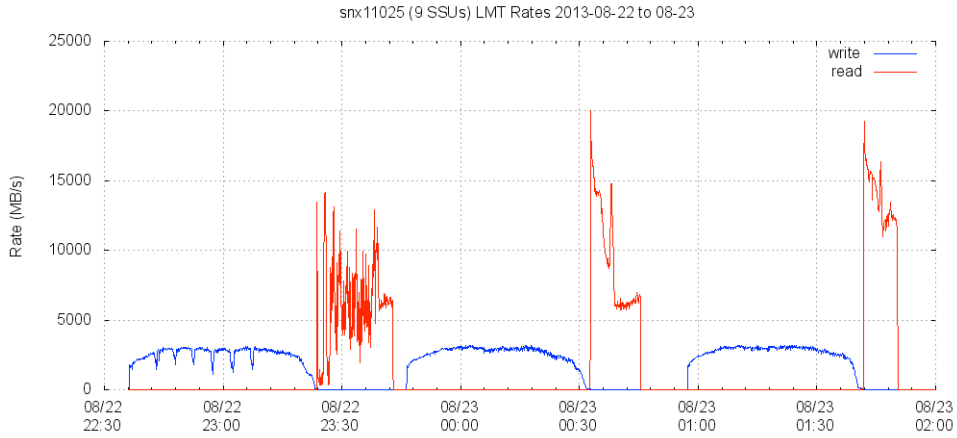


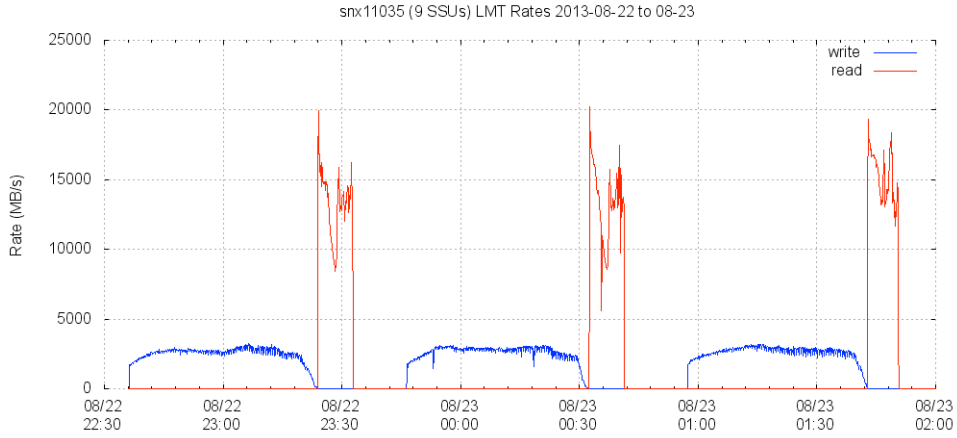
Figure 10. The LMT data for the two MPI-IO 10k jobs ran on 3/26/2014 (the same two jobs as in Fig. 8). The second read curve in each panel is the read rate from the re-read job.

Knowing that the read profile of the read-after-write is a characteristic of the performance slowdown in the MPI-IO 10k test, we then looked back the August LMT data. Fig. 11 (a) is for FS1 and Fig. 11 (b) is for FS2. We see that the read-afer-write profile in August was more like the current re-read profile. FS2 was less full and less fragmented than FS1 that time, which probably accounts for the slightly better performance of FS2.

We attempted to look further into the difference between the two read profiles. Fig. 12 shows the I/O time per compute node for one of the dedicated runs on FS2. Fig. 13 shows the I/O rates for the two selected nodes, 47 and 48. We can see that the I/O times of the compute nodes vary a



(a)



(b)

Figure 11. These are LMT data for the three dedicated MPI-IO 10k jobs on the FS1 (upper panel) and FS2 (lower panel) that were run on 8/23/2013.

lot in the read-after-write test but are very balanced in the re-read test. Since each compute node has perfectly balanced I/O workload, and does not do any computing, we could expect a very balanced I/O time among compute nodes in dedicated runs. The fact that the re-read job has a well balanced I/O time while the read-after-write does not suggest some kind of interaction between the write and read phases in the standard run. However, from how the IOR code was run, and from the application level, we do not expect any interaction between the write and read phases.

Whatever has caused the 70% slowdown between August and December, we have at least identified that something has changed that dramatically changed the performance profile and the net performance. This has made it practical to use internal Cray R&D systems to debug the problem and to confirm the fix. In addition, on the internal systems it is easy to switch back and forth between CLE and Lustre versions compared to a production system like Edison.

4) CLE and Lustre upgrades: The CLE and Lustre versions are not easily revertible, especially on a production

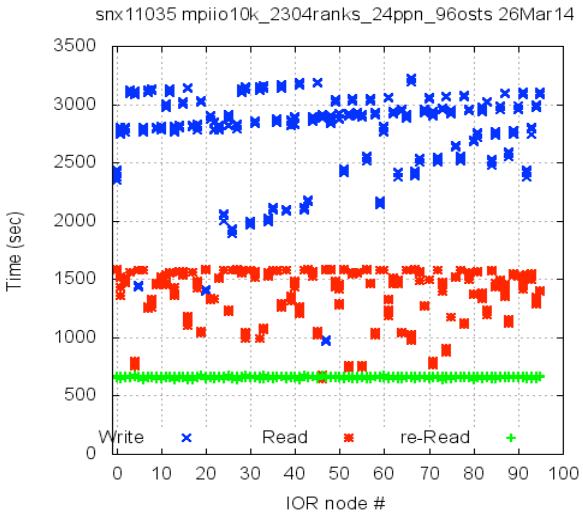


Figure 12. The write and read time per compute node. This was a dedicated run on FS2 on 3/26/2014.

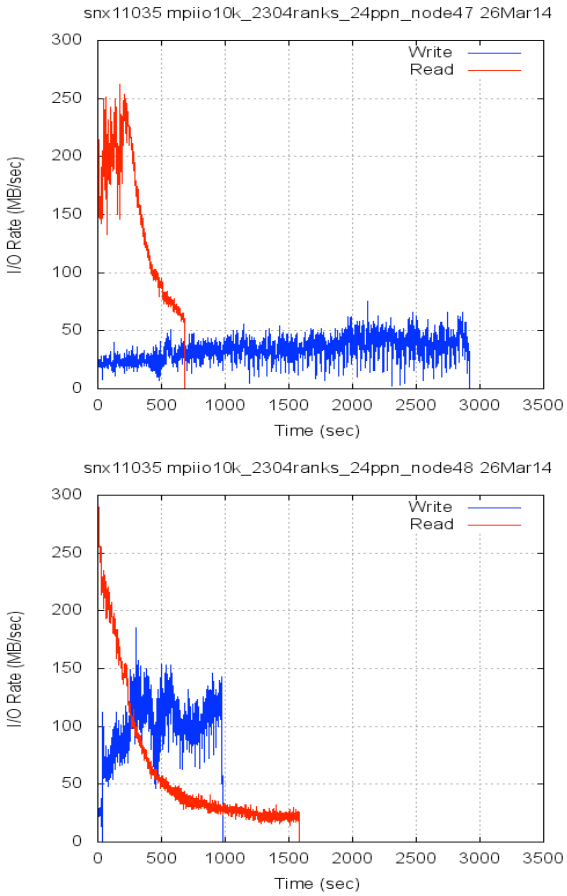


Figure 13. I/O rates for the nodes 47 (upper panel), and 48 (lower panel) in Fig. 12.

system. However, we had a chance to run a test with CLE 4.2 + Lustre 1.8.6 and with CLE 5.2 + Lustre 2.4 on the internal Cray R&D XC30 system. We observed that the read profile was fairly flat with CLE 2.4 + Lustre 1.8.6 (See Fig. 14) while it steeply declined with CLE 5.2 + Lustre 2.4 (Fig. 7 upper panel). This comparison is more evidence that something has changed in CLE and/or Lustre client that is the cause of the change in performance. By comparing the versions running on Edison in August and in December, it further narrows down the range of changes to somewhere between CLE 5.0.UP03 + Lustre 2.3.0 and CLE 5.1.UP00 + Lustre 2.4.0. However, since the Lustre client upgrades are always incorporated into specific CLE versions, it was difficult to determine if it was CLE or Lustre that was responsible for the performance slowdown.

5) A Sonexion parameter readcache_max_filesize: When looking into the file system configuration changes on Edison since last August, we noticed that the value of a Sonexion parameter, readcache_max_filesize, was changed from "infinite" to 1MB with one of the Lustre upgrades on Edison. This parameter controls the maximum size of a file that both the read and write-through caches will try to keep in memory. Files larger than readcache_max_filesize will not be kept in cache for either read or write. This parameter

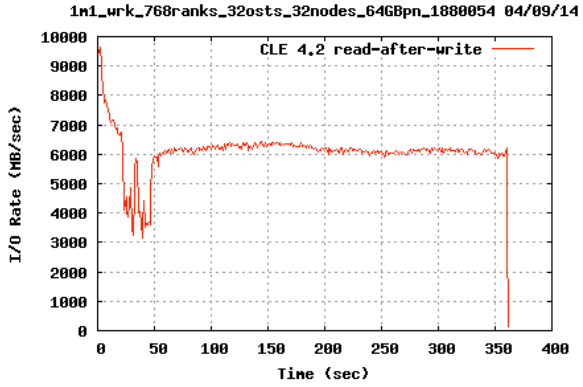


Figure 14. The read rates measured by the instrumented IOR for an MPI-IO 1m1 test without collective buffering under the CLE 4.2 and Lustre client 1.8.6 on the internal Cray R&D Cray XC30 system.

change appeared to be worth investigating. Fortunately, this parameter was resettable without needing to revert the Lustre version. So in one of our dedicated tests, we tested if this change was responsible for the MPI-IO 10k performance 70% slowdown. We observed that when set to "infinite" (the same as in last August) the MPI-IO 10k read rates improved, especially in the re-read tests. However, the read pattern of the MPI-IO 10k test was not changed, and the improvement of the read rate in the read-after-write test was not enough to restore the last August results. See Fig. 15.

6) Compute node kernel and Lustre caches: As we have mentioned earlier, in the standard IOR benchmark runs the write and read tests were run in a single aprun invocation. The fact that the read-after-write performs poorly while the re-read test does not indicates that the write phase must have left some "residue" which affects the follow-on read test. The compute node kernel caches appear to be the first suspect. We did an MPI-IO 10k test (in production), in which we cleared the compute node kernel caches between the write and read phases by running the following command on the compute nodes,

echo 3 > /proc/sys/vm/drop_caches

This was possible because IOR has an option to add an arbitrary time delay between the write and the read phases (the –d option), so that we (root) could run the above command on the compute nodes. However, this did not result in any observable difference in the read profile of the MPI-IO 10k test. Next, we further cleaned up the Lustre caches by running the following command on the compute nodes,

echo 1 > /proc/fs/lustre/ldlm/drop_caches

and finally, we found that the read profile of the MPI-IO 10k test changed to the same as that of the re-read test. See Fig. 16. This points to issues in the Lustre implementation. We provided our investigation results to a Cray Lustre developer, who worked closely with us at the late stage of the debugging. With further analysis, he was able to identify

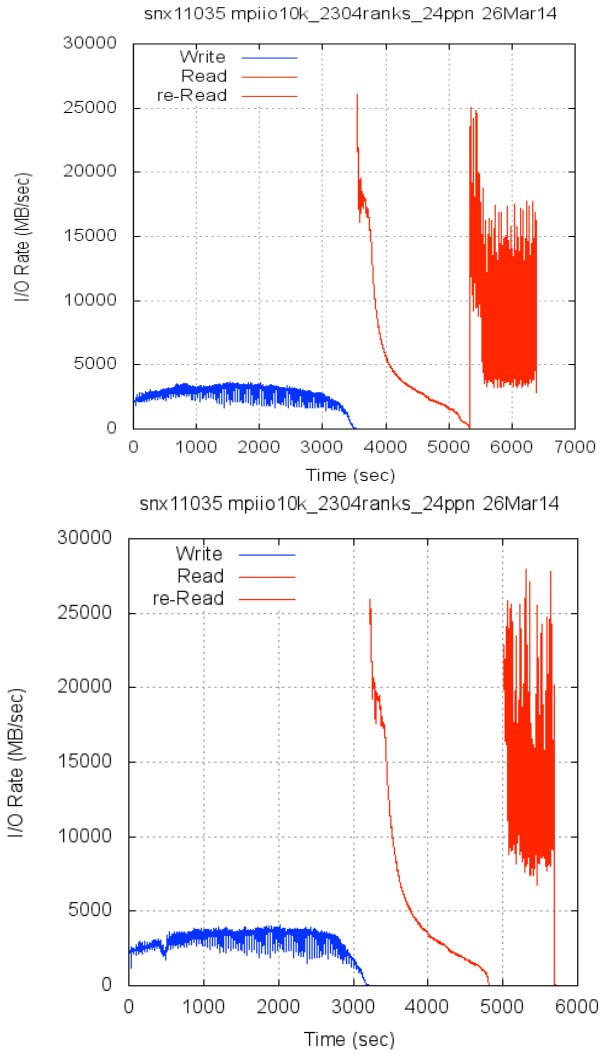


Figure 15. Dedicated MPI-IO 10k test runs on FS2 with the readcache_max_filesize of 1MB (upper panel, the current value) and "infinite" (lower panel, the value in August, 2013). The IOR rates were from instrumented IOR.

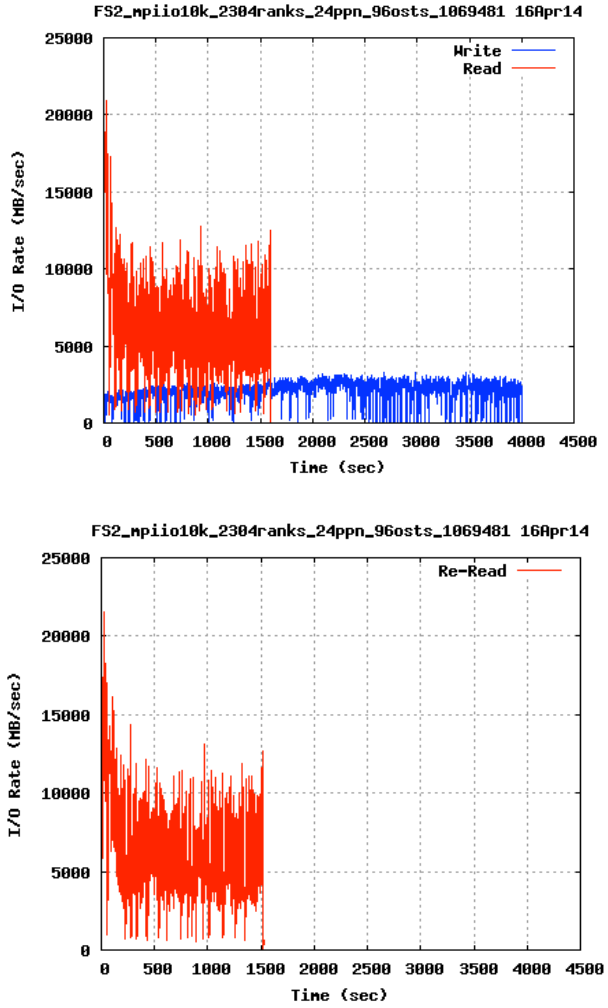


Figure 16. The figure shows the I/O rates for an MPI-IO 10k standard test (write and then read, upper panel), and a re-read test (lower panel) on FS2. These tests were run under the production environment. Between the write and read phases during the standard run, the compute node kernel and Lustre caches were cleaned up.

the specific Lustre patch which was responsible for this performance issue [9]. However, the patch was both too old and too central to be easily removed from Lustre 2.4/2.5/etc. Further investigation is still under way.

B. *Lessons learned from the MPI-IO 10k performance issue*

Through the debugging of this performance issue, we clearly see some room to improve the software release, installation and testing process. While we could not expect bug-free software in general, a better set of benchmark suites seems to be needed for both developers/vendors and sites, which would help in catching performance issues earlier. Yet while this may be obvious in principle, there are many challenges in I/O testing. For example, this specific MPI-IO 10k test takes more than two hours to complete, and that has made it an unfavorable test to run after every CLE and Lustre client upgrade in dedicated mode for a production system like Edison. However, running a shorter version of this test may not show the performance problem. It seems the instrumented IOR is something we could deploy in the future as it may provide a way to catch performance profiles without running lengthy benchmark runs. Or if we have to run lengthy tests, the instrumented IOR may catch distinctive performance profile changes without needing to run on dedicated systems.

VI. I/O Performance Variance and Monitoring In Production Enviornment

I/O performance variation is very disruptive to user workflows. The two major causes of I/O time variation are the contention for the I/O resource among users and the degraded file system performance. While contention is unavoidable under our current configuration (because I/O resources are not a "schedulable" resource), we can still take

some steps to mitigate the situation by promoting good I/O practices where applicable. During one of our debugging processes we located a user job that was stressing the file system with the Posix file per process I/O on FS3. The user bundled 11 job instances into one large job. Each job instance used 1024 cores; each core read a 50 MB file. So the job ran with a total of 11,264 cores, reading 500 GB in total. Looking into the past Darshan [3] profiling data (unfortunately, Darshan is disabled on Edison now because it does not work with the current craype version, 2.0 and up, therefore no current data) it appeared that the user code may do small transfer size I/O operations. In addition, the user was using the default Lustre stripe count on FS3 (8), which is not optimal in general for Posix file per process I/O. So we suggested that the user try the IOBUF library with stripe count 1. The user was able to get at least 100 % I/O time improvement and reduced I/O time variation. See Fig. 17.

We notice that when users report huge I/O variation, usually it is when some components of the file systems are underperforming or misbehaving. Last March, many users reported that their file I/O was slow by more than 10 times. With dedicated debugging from the onsite Cray staff (the debugging was not trivial), we were able to locate a slow disk drive, and resolved the problem by replacing the slow drive with a spare one. During this process, we ran the Posix FpP 1m2 IOR test multiple times to aid the debugging. This test assigns one file (or several files) with stripe count of 1 to each OST, and has exactly one rank writing to and reading from each file. This way, the performance of each specific OST can be measured with only about 5 minutes of test time. Any OST that is having performance problems can then be easily identified, even without dedicated system time. Determining the cause is another challenge, but at least it is known that there is a problem and it has been narrowed down to a specific OST. It has proven to be an easy tool to use to help debugging and confirm the fix. Therefore, we are now running IOR tests regularly to proactively detect performance issues.

Being able to monitor and detect file system health and performance is crucial to delivering quality I/O service to users. Currently we have the Lustre Monitoring Tool (LMT) [10], which monitors Lustre file system servers, and the Simple Event Correlator (SEC) [12], which monitors system events, in place on Edison. LMT data is very useful to monitor the file system activity and performance. Currently, the LMT data is not available for general users, though. It requires extra efforts to be able to make them available to users. NERSC is working on it now. Edison uses the Cray provided Simple Event Correlator (SEC) tool to monitor file system events, which can alert system administrators when there are system changes that are predefined in the SEC rule file. For example, SEC monitors the following file system related events:

Boot, disk in and out,
Various failovers, e.g., mds, OST, etc.,
Slow or hung threads on OSS nodes
Failed to connect to database
Lock timed out
Fan enclosure error

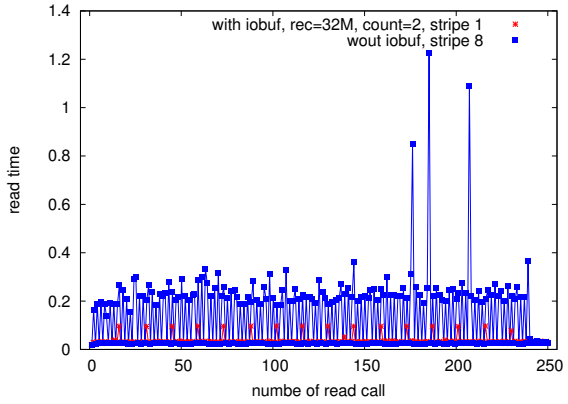


Figure 17. File read time comparison of a user code, QLUA [11], with and without using the IOBUF library. The stripe count was also changed from the default 8 to 1. This figure was provided by a NERSC user.

We often receive SEC reports about slow or hung threads on the OSS nodes, but it is usually difficult to determine if it is something that can wait until the system recovers by itself (e.g., if they are just from user contention), or it is serious enough and is in need of investigation right away. In addition, it is usually difficult to correlate the slow threads with the affected user jobs without non-trivial manual interaction.

We do not use the Cray Sonexion System Manager (CSSM) [13] on Edison. It has a web based GUI that provides status and control of all system components, including storage hardware, RAID, operating system and the Lustre file system. It seems to be considered more as a debugging tool instead of a monitoring one among the onsite Cray staff. In their last attempt to run CSSM on Edison, the GUI failed to display LMT data because the available browser version was too old on the system management server. It seems some work is needed for CSSM to fully function on Edison at this point. However, in the long run if we could incorporate some of the monitoring capability of CSSM in to the Nagios [14] framework, it should be very helpful.

VII. Conclusion and Future Work

In this paper we provided a detailed investigation into many potential causes of I/O performance variation on NERSC's Cray XC30 system. Through an extensive series of experiments on Edison and on an internal Cray system we ruled out programing environment changes, file fragmentation and physical positions, a Sonexion caching parameter, and CLE upgrades. We were able to narrow the cause to a range of Lustre releases and eventually to a specific Lustre patch. A further investigation to fix the problem is still under way.

The key progress made in this investigation was identifying the distinctive read profiles of the MPI-IO 10k test with the instrumented IOR benchmark code, which made it possible to reproduce the dedicated performance issue of large file systems on a small internal Cray system, and to investigate the problem in a production environment.

Catching the distinctive performance profiles using the instrumented IOR could be a general approach that helps debugging elusive IO performance issues. Because the performance profile is more sensitive to the changes compared to the net I/O rates, especially a large I/O rate variation from run to run may occur.

The I/O run time variation can be very disruptive to user workflows. Promoting good user I/O practices may mitigate the variation from user contention. NERSC uses the SEC and LMT to monitor the file system health and performance, and is running IOR benchmarks regularly to help monitor the file system performance. NERSC is looking into making LMT data available to users, and is also looking into other benchmark options to test the I/O performance with small transfer sizes.

ACKNOWLEDGMENT

The authors would like to thank Mark Swan at Cray, Inc., for extracting LMT data on Edison, and would also like to thank Steve Luzmoor, Patrick Farrell at Cray, Inc., for their work and help to resolve the bug 809189 [the bug we opened for this MPI-IO 10k performance issue]. The authors also thank a NERSC user, Marcus Petschlies, who provided the IOBUF tests results with a QCD code, QLUA. They also thank Nathan Wichmann at Cray, Inc., who conducted the IOR acceptance tests. They also thank Shane Canon at NERSC for providing Lustre file system usage on Edison, and Harvey Wasserman at NERSC for valuable discussion and help. In addition, most of the debugging jobs were required to run on the dedicated Edison system, which was not possible without the support from Jeff Broughton, the NERSC-7 project manager, and the help from other Cray onsite and NERSC system staff. This work was supported by the ASCR Office in the DOE, Office of Science, under contract number DE-AC02-05CH11231. It used the resources of the National Energy Research Scientific Computing Center (NERSC).

REFERENCES

[1] NERSC Edison: http://www.nersc.gov/users/computational-systems/edison/

[2] NERSC-6 benchmark suite, a set of benchmark codes and tests resulted in the Hopper system, a Cray XE6, acquisition. Edison procurement, coded as NERSC-7 internally, used the same set of benchmark suite.

[3] Darshan, a light weight I/O profiler, http://www.mcs.anl.gov/research/projects/darshan/

[4] NERSC workload analysis used for NERSC-8 procurement [internal communication]

[5] Cray Sonexion Brochure, is available at http://www.cray.com/Assets/PDF/products/sonexion/SonexionBrochure.pdf

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

[7] Cray IOBUF: module load iobuf, man iobuf

[8] D. Petesch, M. Swan, "Instrumenting IOR to Diagnose Performance Issues on Lustre File Systems", Proc. Cray User Group, May 2013.

[9] LU-744 osc: add lru pages management - new RPC, http://review.whamcloud.com/#/c/2514/

[10] LMT: Lustre File System Operations Manual - Version 1.8 S-6540

[11] QLUA, https://usqcd.lns.mit.edu/redmine/projects/qlua_code

[12] SEC: http://simple-evcorr.sourceforge.net/

[13] CSSM: Cray® Sonexion® Administrator's Guide HR5-6093-B

[14] Nagios: http://www.nagios.org