MPI-I/O on Franklin XT4 System at NERSC

Katie Antypas and Andrew Uselton
NERSC, Lawrence Berkeley National Laboratory Berkeley, CA 94720
{kantypas, acuselton@lbl.gov}

Abstract

Prior to a software upgrade and hardware maintenance on March 17th 2009 on the Frankin Cray XT4 machine at the National Energy Research Scientific Computing (NERSC) Center, MPI-IO shared file performance saw only a small percentage of file-per-processor performance POSIX performance. The March 17th upgrade unintentionally increased I/O performance significantly for a number of applications. This paper shows the performance differences after the maintenance and explores some of the possible explanations for the dramatic improvements.

1 Introduction

The National Energy Research Scientific Computing (NERSC) Center regularly runs benchmark codes on the Franklin Cray XT4 machine to assess the system's health. After a system upgrade and maintenance on March 17th 2009, the I/O performance increased substantially and unexpectedly. A number of applications showed increased I/O performance after the maintenance, but MPI-IO codes showed the largest gains. The reason for the performance gains remain unknown. This report discusses the MPI-IO performance on the system before the maintenance and shows the increase in performance afterwards with three user applications. Finally, the report suggests some of the possible explanations for the improved I/O performance.

2 Overview of the Franklin Cray XT4

The Franklin Cray XT4 consists of 9660 compute nodes with quad-core Opteron processors connected as a 3D torus with the Cray SeaStar-2 interconnect. Each compute node has 8 GB of memory for a total of 77.3 TB of system memory. Figure 1 shows the torus connecting service nodes to external RAID-based disk storage units. Franklin has 21 OSSs and 5 raid storage units. The service nodes and RAID units support the *scratch* file system where users perform high-performance I/O. Each RAID unit houses two controllers, eight 4 GB/s *fibre-channel* (*fc4*) connections and sixteen 4 TB *logical disk units* (LUNs). Each service node mounts four of the LUNs as SCSI devices and hosts two *fc4* connections. The service nodes and RAID units all connect via a pair of Cisco 6500 *fibre-channel* switches. The */scratch* file system has 346 TB of disk, and had a peak write and read bandwidth of about 11 GB/s.

The scratch space on Franklin is configured to use the Lustre [2] parallel file system. Lustre uses the service nodes as *Object Storage Servers* (OSSs) for handling bulk data objects. An additional service node acts as the *Meta Data Server* (MDS) to organize the data objects and manage the *name space* as a POSIX-compliant file system. Lustre mediates the access to each SCSI device (LUN) via an *Object Storage Target* (OST), which is a service running on the OSS in kernel-space. The *Meta Data Target* (MDT) is the name of the equivalent service running on the MDS. With four OSTs on each OSS there are a total of 80 OSTs available to Franklin and collectively mounted as the */scratch* file system.

3 MPI-IO Performance Before March 17th Maintenance

The MPI-IO shared file performance on Franklin showed a low percentage of the performance of equivalent tests run without MPI-IO using a POSIX file-per-processor test. The tests were conducted with the IOR [6] benchmark. The IOR test reports the amount of data read and written and the time spent writing and reading as well as the calculated rates in MB/sec. Each processor writes and reads a set amount of data, known as the *block size* and transfers that data to disk in *transfer size* chunks. The number of *block size* transfers written or read from disk can be adjusted to increase the total file size. Figure 2 shows the results of IOR tests run before the March 17th

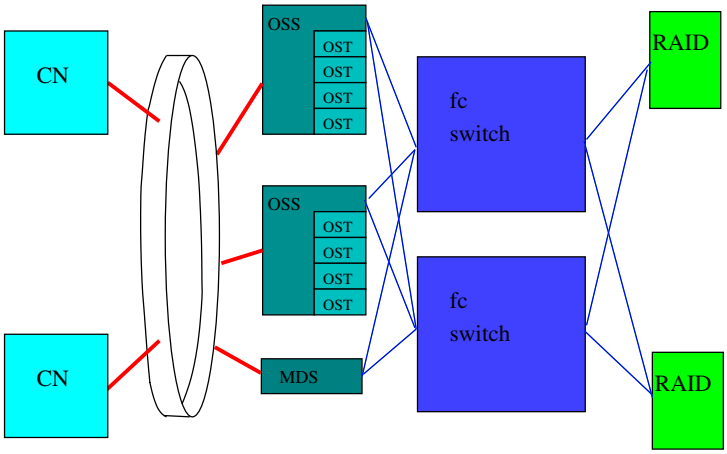


Figure 1. The Franklin Cray XT4 System

maintenance period. The figure shows a read and write test where each of 64 processors writes or reads a total of 2GB of data. We had seen I/O performance on Lustre vary greatly depending on whether the I/O transactions aligned to the Lustre block boundaries. To test this, two different *transfer sizes* were used, 1,048,576 bytes (1MiB) and 1,000,000 bytes (1 MB). The power-of-two based transfer size 1,048,576 bytes (1MiB) align to Lustre block boundaries while the 1,000,000 bytes (1MB) transfer would not. Additionally, each test was performed using two different application APIs. The first is a file-per-processor test, (shown in dark blue in Figure 2). The second, (shown in light blue) is a shared file MPI-IO test. Except for the 1 MiB read test, the file-per-processor tests perform significantly better than the MPI-IO cases. Some overhead from MPI-IO is expected and the 1 MiB write test performs at about 40% the rate of the file-per-processor test. The differences between the power-of-ten, MB case and the power-of-two, MiB cases are dramatic. The power-of-ten, 1MB read and write cases see only 1% and 7% of the file-per-processor performance. The difference between the power of two transfer sizes and the power of ten transfer sizes is attributed to efficiencies when I/O transactions are aligned to the Lustre block boundaries. Real user applications don't necessarily write and read perfect power-of-two sized bytes and so user applications often saw low MPI-IO performance. The 1MiB MPI-IO read test actually slightly outperforms the file-per-processor test. This was a surprising, but repeatable result and indicates there is little overhead from reading power of two sized data. (In this test, the file-per-processor test does not achieve the peak performance of the file system likely due to the overhead of writing smaller *block sizes* 2000 times.)

4 March 17th Maintenance

On March 17th, 2009 Franklin had a scheduled maintenance which included both hardware repairs and software upgrades. The hardware maintenance put four Seastar-2 interconnect links back into the system. Previously these down links had been mapped out of routing tables so that traffic was routed around them. The software upgrades included a major patch to the Cray OS software called Cray Linux Environment (CLE) officially called an upgrade to CLE2.1UP01 with patch sets 01, 01A and 02. The CLE upgrade included hundreds of bug fixes. NERSC and Cray staff were not aware that any of the intended changes would have a performance impact on I/O.

5 User Applications Showing Performance Increase after March 17th Maintenance

A couple days after the March 17th maintenance we noticed a significant I/O performance improvement for the MADBench2 [7] code. Two other other codes FLASH [4], S3D [1] as well as the IOR benchmark were used to confirm the reported I/O improvements.

5.1 MADBench2

MADBench2 is an I/O kernel extracted directly out of an application analyzing massive Cosmic Microwave Background datasets from satellites. Because large datasets and memory footprints are required, MADBench2 uses and out-of-core algorithm. Data is written to disk and then read back in from disk as the calculation progresses. MadBench2 can use one of two APIs, file-per-processor POSIX or MPI-IO. Figure 3 shows the write and read patterns of an MPI-IO MADBench2 application profiled with the IPM-IO [9] library. The application is run with 256 processors and the I/O profile is created by

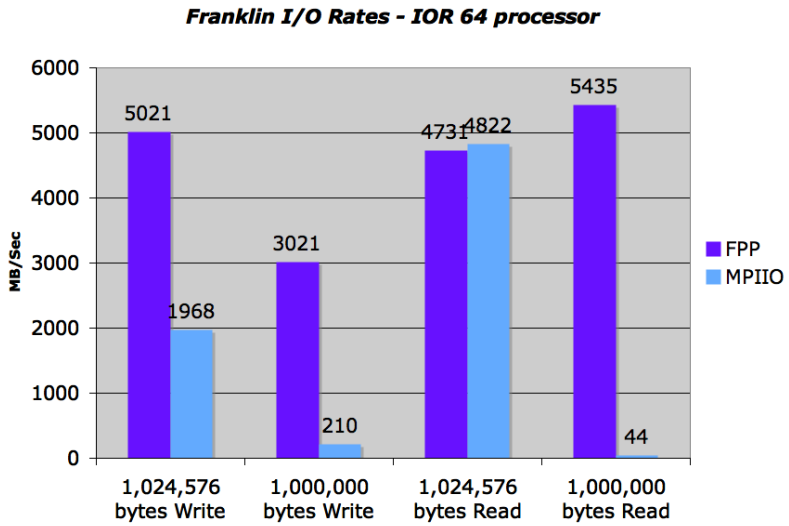


Figure 2. MPI-IO write and read performance as compared to file-per-processor (FPP) POSIX performance on Franklin before the March 17th 2009 upgrade. The non power-of-two transfer size write and read MPI-IO performance is particularly low.

intercepting libc open, close, read and write calls. Figure 3(a) shows the profile of the MadBench2 application before the March 17th, 2009 upgrade. The x-axis is time while the y-axis is processors. The color blue indicates writing while red indicates reading. Between each read and write call an MPI barrier is placed for synchronization. The white space then indicates time that a processor was waiting for others to finish. What is striking about the MADBench2 application before the March 17th upgrade is the large amount of time spent waiting for a few processors to finish a read. The blue write calls do not show the same degree of outlying processors.

In contrast, Figure 3(b) shows the read calls very tightly aligned with very little white space. The figure is shown to scale and thus shows the MADBench2 simulation running roughly 4 times faster after the March 17th maintenance. It appears that some change dramatically improved the performance of MPI-IO reads. (The MADBench2 runs were performed multiple times on different days before and after the maintenance.)

Figure 4 shows the performance of MADBench2 with the POSIX API. Notice there is little difference before and after the March 17th maintenance. Whatever caused performance the MPI-IO performance to increase does not appear to have the same effect on POSIX performance.

5.2 FLASH

The FLASH [4] code is a modular adaptive mesh code used for simulating compressible reactive flows in astrophysical environments, primarily focused on the deflagration and detonation of type Ia supernovae. The FLASH code's I/O pattern and performance is often used as a benchmark to gage system performance. [3, 5, 8]), While various physics applications may be run with the FLASH code, the I/O pattern for these applications is largely the same and consists of writing grid variables for checkpoint/restarting and smaller single precision plotfiles for visualization and analysis. In this particular study 2048 cores were used. Checkpoint files use the HDF5 API to write 10GB to a shared file. Five smaller analysis files written during each run also using HDF5 and are roughly 2.5 GB in size. At the beginning of a simulation a checkpoint file is read from disk. Figure 5 shows the read performance and the large increase seen after the March 17th. Before the March 17th maintenance, read performance was less than 100 MB/sec. After March 17th the performance jumped to 1,400 MB/sec, an increase of over 14 times.

Figure 6 shows the increase in write performance after March 17th. Although not as dramatic as the increase in read performance, write performance still increased 2-3 times after the maintenance. It is also interesting to note that code was not recompiled after March 17th.

5.3 S3D

S3D is a code used to solve the fully compressible Navier-Stokes equations to study the interactions of turbulence and chemistry reactions in combustion. The I/O in S3D uses a file-per-processor model, rather than MPI-IO to output restart/checkpoint files which are also used

(a) Before: slow reads by a few processors force all processors to wait increasing the runtime of the application (b) After: slow outliers gone

Figure 3. Madbench2 MPI-IO performance before and after the March 17th maintenance. Writes are shown in blue and reads are shown in red. The number and degree of slow read processors all but disappears after the March 17th maintenance.

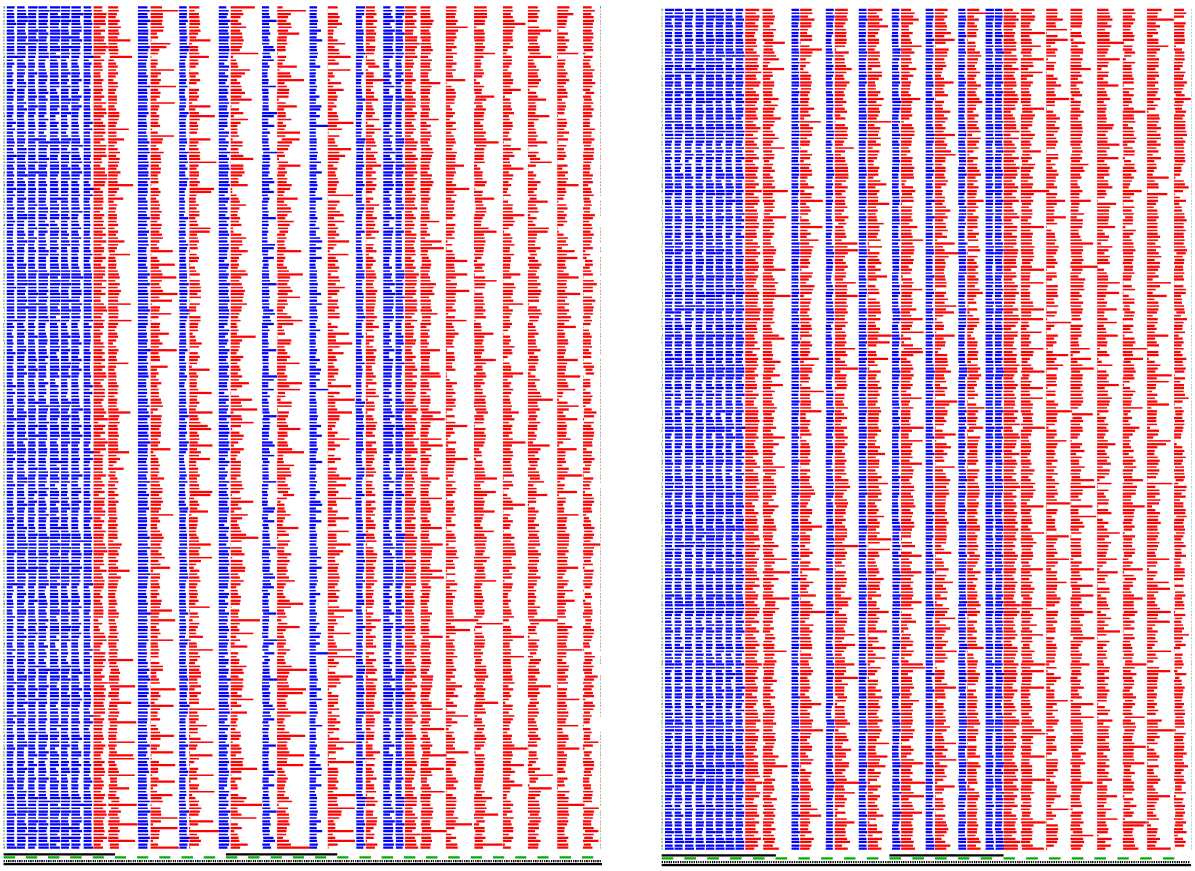
for analysis. As with the other applications, the number of processors and frequency of output can be adjusted. In this case, 512 cores were used and 5 checkpoint/restart files were written, one ever 50 timesteps. Each processor writes 5.8 MB to its own file for every checkpoint output. Figure 7 shows the performance of the S3D I/O rate before and after the March 17th maintenance. There is a high degree of variability in both cases, however using the harmonic mean, the average rate on February 27th was 955 MB/sec, while the average rate jumped to 4972 MB/sec after the upgrade, an increase of over five times. Unlike Madbench2 with the POSIX API, S3D does show a significant increase in performance after the maintenance.

5.4 IOR

The IOR benchmark was used again to gage the extent of the performance improvements on the system after the March 17th upgrade. Figure 8 shows the IOR MPI-IO test described at the beginning of the paper in Figure 2. The MPI-IO performance, particularly for the power-of-ten read tests show dramatic performance improvements. As is inline with the FLASH application, write performance increased by roughly a factor of three. In read performance, the power-of-two case performed about the same as before the upgrade, but the power-of-ten MPI-IO read performance increase by over fifty times. Clearly something was not performing correctly before the upgrade and MPI-IO performance is more inline with expectations now.

6 Possible Explanations for Improvements

Since no I/O performance improvements were intended, NERSC and Cray staff have been hypothesizing various reasons for the improved performance seen after the March 17th upgrade. Besides the down links and the software upgrades, workload changes could also have been another coincidental factor.



(a) Before (b) After

Figure 4. Madbench2 POSIX performance before and after the March 17th, maintenance. There does not appear to be any significant performance increase for the POSIX API.

6.1 Down Links

On March 17th four SeaStar2 hardware interconnect links were put back into service. NERSC and Cray staff considered the effects down links could have if they were placed in crucial positions in the torus. Could four down links be the cause of poor MPI-IO performance? Subsequently Cray staff did an analysis on the SeaStar2 network congestion before and after March 17th. They found no change in network congestion before or after March 17th.

6.2 Software Upgrade

Another possibility is that the upgrade to CLE2.1 UP01 caused the I/O performance increase. The software upgrade contained hundreds of bug fixes. Currently, the leading suspect is an asynchronous journal commit modification included in CLE2.1 UP01. With the journaling modification the frequency at which journal entries were written was reduced by a factor of 8, lowering the number of I/O operations. Additionally, the implementation of writing to the journal was made more efficient. Examining the I/O operations sent to the DDN controllers after March 17th (Figure 9, shows that the number of operations was significantly lowered. It is not clear though, that the reduction in I/O operations wasn't due to workload changes on the system.

6.3 Workload Changes

The Franklin system has a dynamic and constantly changing workload. Could the performance improvement be caused by a lower I/O load on the system? Steve Luzmoor from Cray did a workload analysis before and after March 17th. One NERSC user, in particular, has an application which creates an enormous amount of traffic on the metadata server. The user launches many small jobs and scripts onto the compute nodes with the *aprun* command. (The *aprun* command is similar to the *mpirun* command for Linux clusters.) Figure 10 shows the number of *aprun* commands over 2009. (There was a drop at the end of February when the system undergoing mainte-

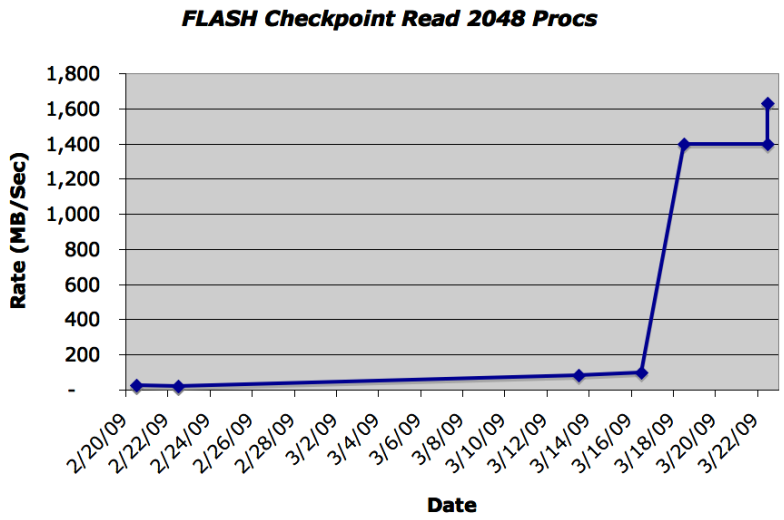


Figure 5. I/O read performance of the FLASH code for a 10GB checkpoint file. Performance increases substantially after March 17th.

nance for a couple of days.) Looking later into March, the number of jobs launched with *aprun* has remained low since March 17th. Not coincidentally, one particular user was responsible for 20,000-40,000 *apruns* until he slowed his usage after the 17th. While a change in workload is a possible explanation, it isn't clear why the workload change would affect MPI-IO performance and not POSIX performance. Recall with the MADBench2 analysis, POSIX performance showed little difference before and after March 17th while MPI-IO performance consistently showed improvements after the maintenance. S3D, a non-MPI-IO code, on the other hand, did show I/O improvements.

7 Conclusions

Since the March 17th 2009 maintenance the Franklin system has undergone further changes. NERSC doubled the number of I/O nodes and redistributed them more evenly across the system. The system also has become more stable and better performing since the March 17th maintenance and so changing the system back to the old configuration is infeasible. The priority now is to assure that performance on the system does not regress. However, the performance improvements were so significant that finding the root cause would be useful to both Cray and NERSC. Cray should be aware of which fix caused such a large I/O improvement so they can assure all XT4s and future systems are configured to have the best performance possible. This study also underscores the importance of performance monitoring over the lifetime of a system as performance can change dramatically, unintentionally.

8 Acknowledgments

Many thanks to Noel Keen for his work on I/O tracing with IPM. Harvey Wasserman provided the S3D data and analysis and Steve Luzmoor and Tom Davis provided the workload analysis data. John Shalf, David Skinner, Richard Gerber and Shane Cannon have been deeply involved I/O performance analyses on Franklin and have provided invaluable discussions and insights.

All LBNL authors were supported by the Office of Advanced Scientific Computing Research in the Department of Energy's Office of Science under contract number DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under contract No. DE-AC02-05CH11231.

References

[1] E. H. andR Sankaran, J. C. Sutherland, and J. H. Chen. Idirect numerical simiulation of turbulent combustion: Fundamental insights towards predictive models. *Journal of Physics: Conference Series*, pp. 65-79, 2005.

[2] P. Braam. File systems for clusters from a protocol perspective. In *Proceedings of the Second Extreme Linux Topics Workshop*, Monterey, CA, June 1999.

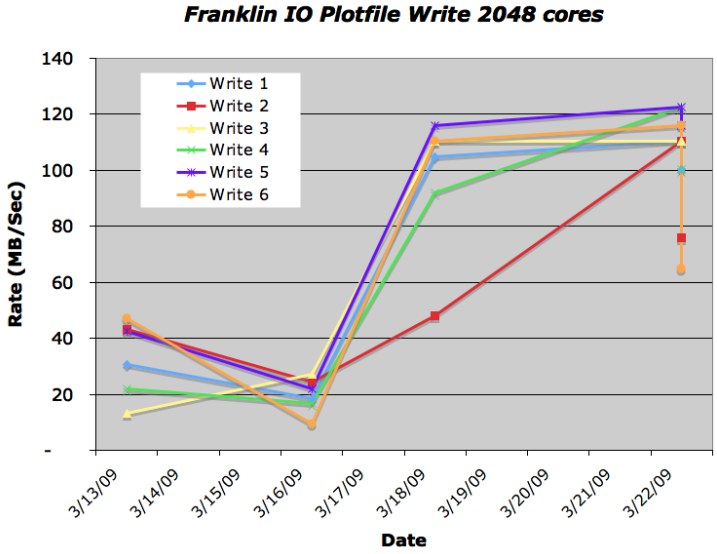


Figure 6. Performance of FLASH code using HDF5 to write 2.5GB of data.

[3] A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp. Efficient structured data access in parallel file systems. In *Cluster 2003 Conference*, Dec 4, 2003.

[4] Flash code. `http://www.flash.uchicago.edu/`.

[5] Flash io benchmark. `http://www-unix.mcs.anl.gov/pio-benchmark/`.

[6] The ASCI I/O stress benchmark. `http://sourceforge.net/projects/ior-sio/`.

[7] MADbench2: A Scientific-Application Derived I/O Benchmark. `http://outreach.scidac.gov/projects/madbench/`.

[8] H. Shan, K. Antypas, and J. Shalf. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In *Proc. SC2008: High performance computing, networking, and storage conference*, Austin, TX, Nov 15-21, 2008.

[9] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, volume to appear, Heidelberg, Germany, 2005.

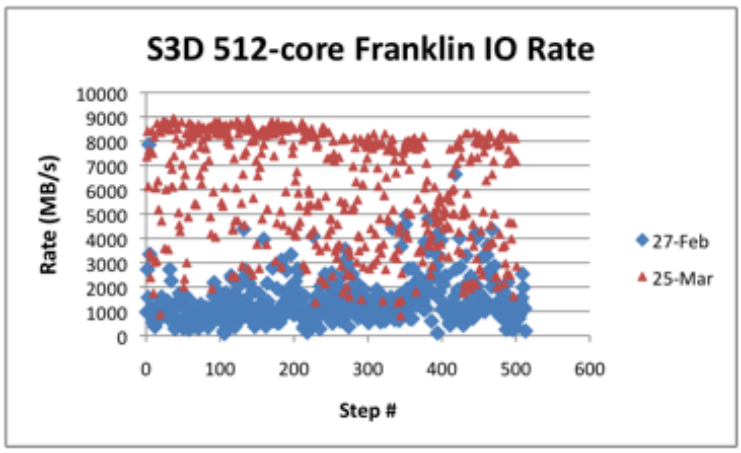


Figure 7. S3D performance before and after maintenance.

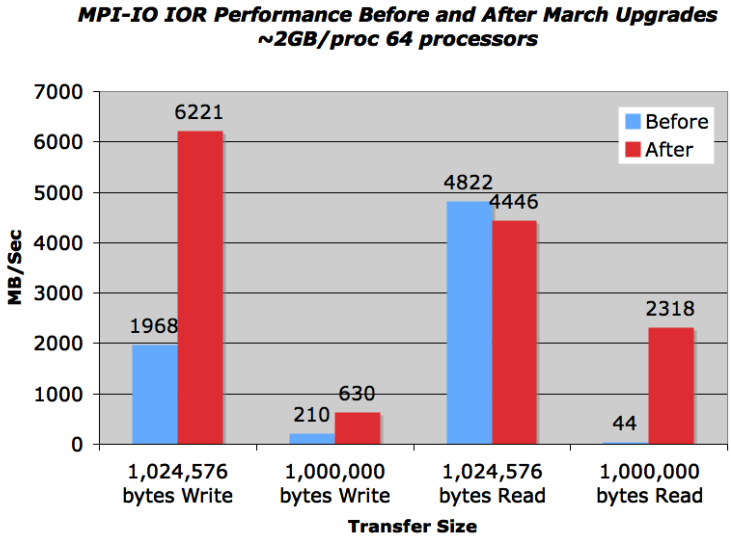


Figure 8. MPI-IO Performance before an after the March 17th maintenance. The non power-of-two MPI-IO read performance increases dramatically

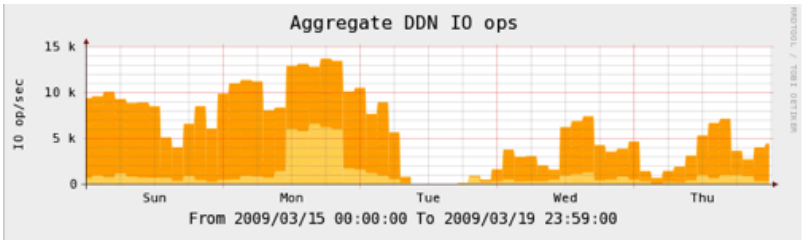


Figure 9. I/O operations to the DDNs. There does appear to be fewer operations after the March 17th maintenance (Tuesday), however it is too soon to tell if this is a real phenomenon or simply a changing workload pattern.

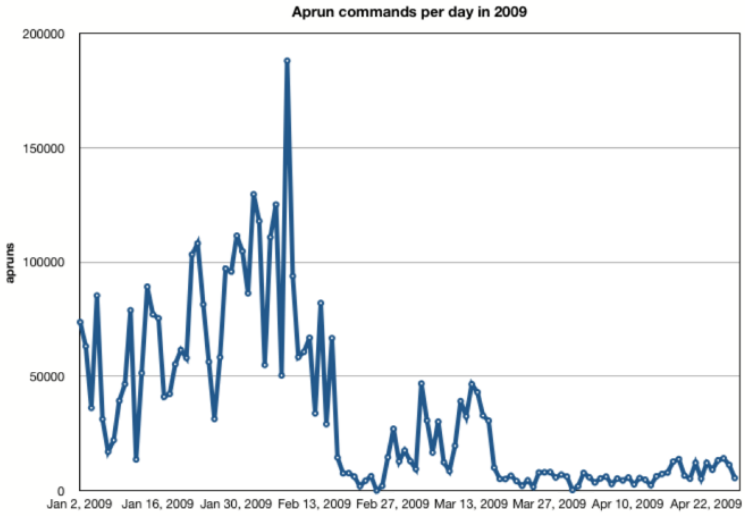


Figure 10. Number of 'aprun' calls launched on Franklin.