U.S. Department of Energy Best Practices Workshop on
File Systems & Archives
San Francisco, CA
September 26-27, 2011
Position Paper

Kevin Harms
LCF/ANL
harms@alcf.anl.gov

ABSTRACT / SUMMARY

This paper addresses the "Usability of Storage Systems" and the "Administration of Storage Systems" topics. Given the small staff assigned to the LCF Intrepid storage resources we have searched for methods to optimize the use of our storage resources. We present these methods for proactively finding opportunities to tune application I/O and finding degraded hardware that is reducing overall I/O throughput.

INTRODUCTION

The LCF is a relatively new facility and is in the process of developing its storage practices and procedures. One item that has become clear is the need to be proactive about storage usage and administration. We have a limited staff dedicated to the storage system and waiting until an issue turns into a real problem leaves us in an awkward position. In order to address this, we are working on some methods to proactively find issues and start working to solve them before they become worse.

The first method was to install a tool, Darshan, to profile user I/O so that users and staff could have a basic tool to help tune I/O for Intrepid and best utilize the storage resources LCF provides.

The second method is to begin looking at the overall performance of the storage hardware to find and fix marginal hardware without the need to wait for it to degrade to the point of outright failure.

System Description

Here is an overview of the core Intrepid storage system. Intrepid has two main storage systems. The home file system is GPFS based and uses 4 DDN9550 SANs that are directly attached via DDR IB to 8 xSeries file servers. The scratch storage has two different file systems running on it, GPFS (intrepid-fs0) and PVFS, (intrepid-fs1) which utilize the same hardware. The scratch area uses 16 DDN9900 SANs, which are directly attached via DDR IB to 128 xSeries file servers. (8 servers per DDN) File system clients are connected over a 10 GB Myrinet fabric.

THE USABILITY OF STORAGE SYSTEMS

Darshan

Darshan [1] was a tool developed by the MCS department in ANL and deployed on the LCF Intrepid Blue Gene machine. Darshan captures information about each file opened by an application. Rather than trace all operational parameters, however, Darshan captures key characteristics that can be processed and stored in a compact format. Darshan instruments POSIX, MPI-IO, Parallel netCDF, and HDF5 functions in order to collect a variety of information. Examples include access patterns, access sizes, time spent performing I/O operations, operation counters, alignment, and datatype usage. Note that Darshan performs explicit capture of all I/O functions rather than periodic sampling in order to ensure that all data is accounted for.

The data that Darshan collects is recorded in a bounded (approximately 2 MiB maximum) amount of memory on each MPI process. If this memory is exhausted, then Darshan falls back to recording coarser-grained information, but we have yet to observe this corner case in practice. Darshan performs no communication or I/O while the job is executing. This is an important design decision because it ensures that Darshan introduces no additional communication synchronization or I/O delays that would perturb application performance or limit scalability. Darshan delays all communication and I/O activity until the job is shutting down. At that time Darshan performs three steps. First it identifies files that were shared across processes and reduces the data for those files into an aggregate record using scalable MPI collective operations. Each process then compresses the remaining data in parallel using Zlib. The compressed data is written in parallel to a single binary data file.

Darshan was deployed on Intrepid by creating a modified set of mpiXXX compiler wrappers which link in the darshan library code. These modified compiler wrappers are part of the users default path which means many applications link in Darshan with no extra work by the user. These applications put logfiles into a common area and are setup so only the user who produced the logs can read them. Later we change the group permission to a special 'darshan' group and then add group read permission. These logs then become accessible by the LCF staff and a few selected MCS research staff.

User Analysis

The first capability this provides is for users to look at some information about their jobs I/O profile and compare it to common suggestions available via our wiki documentation. If the user feels their I/O performance is not as good as it should be, when contacting the LCF staff, we already have some basic information about the I/O patterns they are using which might give some initial starting suggestions for the user to try for improving I/O performance on Intrepid. This also addresses a common issue where users are not familiar with how their application does I/O, perhaps because they are using some large application where someone other person or group implemented the IO code. Figure 1 shows an example from the *darshan-job-summary.pl* output.

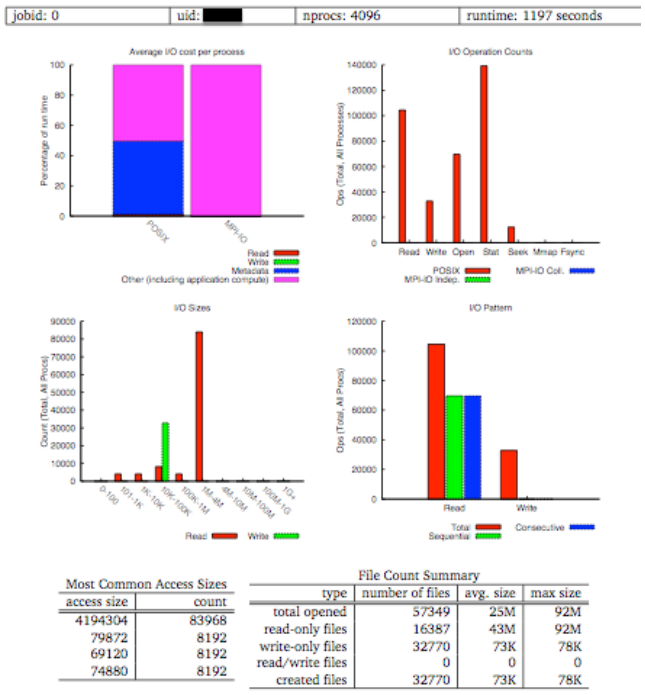
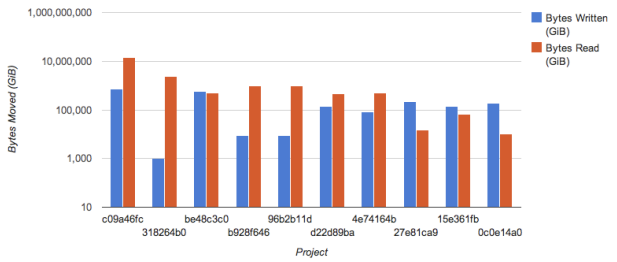


Figure 1 – Darshan Job Summary Example

This summary information can provide a useful starting point for I/O analysis. We are aware of a few applications that have used this output to successfully improve their application I/O for Intrepid.

Project Analysis

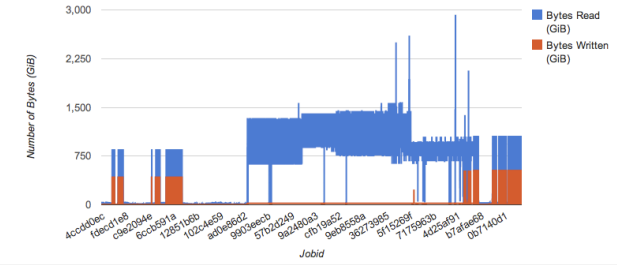
The second capability is to proactively analyze darshan logs to see how users are utilizing the storage system and if they are being effective. We are developing a basic web interface around aggregated log files that can be examined on a per-project basis to find who the major users of the storage system are and how are they using the system. We explored this idea in reference [2]. Figure 2 shows the top 10 projects from 1/1/2011 to 6/30/2011. We can look at these projects individually to see how they are using the I/O system.



Project	Major Project	Bytes Written (GiB)	Bytes Read (GiB)	Bytes Moved (GiB)
c09a46fc	incite	717960.320269	14378691.7527	15096652.073
318264b0	incite	1028.50599501	2415161.5417	2416190.0477
be48c3c0	incite	583163.035626	500354.756459	1083517.79209
b928f646	alcc	8999.55220313	970914.933426	979914.485629
96b2b11d	incite	8912.51224884	940718.474662	949630.986911
d22d89ba	incite	141647.703103	447793.368016	589441.071119
4e74164b	incite	81074.8510083	489839.286861	570914.137869
27e81ca9	incite	222602.486601	14497.4064404	237099.893041
15e361fb	incite	144678.540998	64092.520676	208771.061674
0c0e14a0	incite	181726.374965	10113.3177587	191839.692723

Figure 2 – Top 10 Projects by bytes moved

Once we identify the top I/O users we can examine their I/O usage in more detail. Figure 3 shows an example of aggregate information about a single project. We can look at the percent time spent in I/O and see if we should consider talking with a project about their I/O usage if it looks subpar and thereby improve their utilization of the core-hours they have been granted.



Project	Major Project	Bytes Written (GiB)	Bytes Read (GiB)	Bytes Moved (GiB)
c09a46fc	incite	822788.152176	14693539.2635	15516327.4157

Project	Min (MiB/s)	First Quartile (MiB/s)	Second Quartile (MiB/s)	Max (MiB/s)
c09a46fc	0	42.270659	4827.955307	11199.311494

Project	Job Coverage	Job Coverage Ratio
c09a46fc	1799/3315	0.542684766214

Project	Percent Time in I/O
c09a46fc	0.289274515813

Figure 3 – Darshan Project Information

We had identified a project in 2010 that was a top I/O consumer but had a high percentage of time spent in I/O. We were successful in working with this project to change the method for writing of files, which gave them a 30% improvement in write throughput.

System Planning

The third capability we get is the ability to look at what I/O patterns users want to use and what they want to do. This information can be used to target how we allocate our resources for next generation systems. Examples from above show users are still obsessed with generating O(1000), O(100000), O(1000000) files. The file per process model tends to break down at the 8192 node level (or 32768 processes) on Intrepid. For our next generation system, we have planned to split data and metadata and use separate SSD based storage for the metadata in hopes of boosting metadata performance, which would serve as a band-aid for the file-per-process users.

Another point is that we see about 60% of the jobs at large scale go to either shared or partially shared files and fewer use the file-per-process model. Figure 4 shows this distribution. However, in this same time period we saw remarkably few people using high-level libraries such as HDF5 or PnetCDF. This might indicate we need to spend time educating the userbase about these libraries or find out why our users would rather create their own shared file format rather than leverage existing ones.

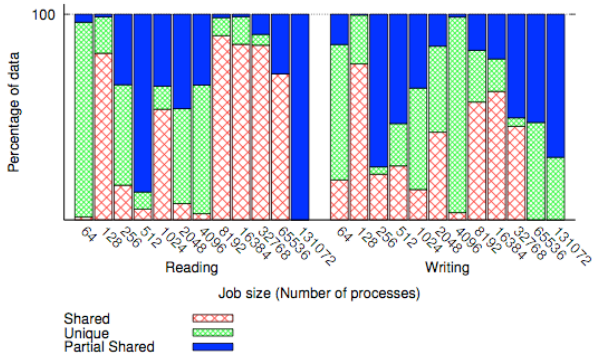


Figure 4 – File usage (1/2010 – 3/2010)

THE ADMINISTRATION OF STORAGE SYSTEMS

Another aspect of storage system efficiency is ensuring the current hardware is delivering performance up to its useable peak. Anecdotally we have observed a single failing SATA disk can produce a global slowdown of the scratch file

system. During our early test stages when we were tuning the /intrepid-fs1 file system, we would often find a marginal drive would cause a significant slow down in an IOR test case. As an example, we would see something on the order of losing 50% of total throughput. After failing 1 (or more) drives, the system would return to its optimal performance level.

The work we have done in this area is still very preliminary and we have not validated any of our suppositions.

Log Analysis

The DDN9900 will report many errors and statistics but it also logs informational events in the system log. These are generally not reflected directly in any of the system statistics. We developed a trivial monitoring tool to check the event logs of each DDN approximately once per day and send and alert if there were a large number of new messages in the log. Here is a short snippet from the monitoring tool, which emails its results.

```
       INFO  INT_GH    8-29  12:50:31  Recovered:
Unit Attention Disk 9G GTF000PAH51JNF

       INFO  INT_GH    8-29  12:59:29  Recovered:
Unit Attention Disk 22G GTF002PAHHKXRF

       INFO  INT_GH    8-29  13:02:43  Recovered:
Subordinate errors detected.

       INFO  INT_GH    8-29  13:05:18  Recovered:
Unit Attention Disk 2G GTF100PAHW59BF

       INFO  INT_GH    8-29  12:49:44  Recovered:
Unit Attention Disk 13G GTF002PAHWD21F

       INFO  INT_GH    8-29  13:00:31  Recovered:
Subordinate errors detected.

       New Log Messages: 2650
```

Example 1 – DDN Log monitoring output

In Example 1, we see that this DDN had 2600 new log messages and many of those messages are related to problems with disk access on channel ‘G’. In this case, we could have opened a support request with DDN to determine which component was really at fault. In this particular case, disk 7G failed 5 days later. We could have failed disk 7G earlier and presumably not lost any performance during that time period.

Visualization

Another method to monitor the storage infrastructure for marginal components is via visualization of I/O metrics. We setup a utility to pull the ‘tierdelay’ metric from all tiers of each of the 32 DDN controllers associated with the scratch file system. We then ran the IOR benchmark with a write workload while we collected samples every 10 seconds. The data was loaded into ParaView and we began looking for patterns.

Figure 5 shows a combined visualization of total operation count for each channel/tier combination for all DDNs at the last timestep.

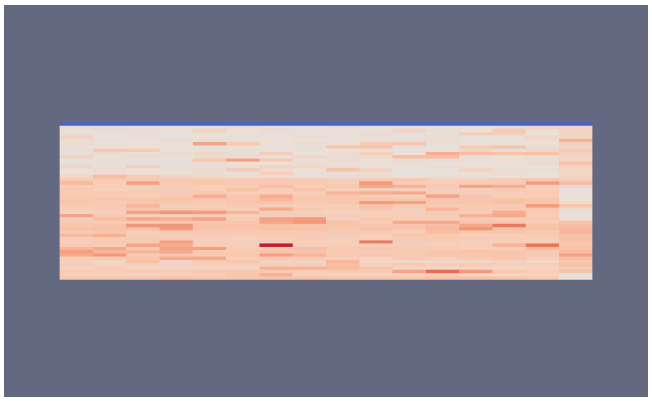


Figure 5 – Cumulative Operations

Since the IOR workload was evenly distributing data over all LUNs we should see similar operation counts, but instead we see one tier (dark red) that has significantly more operations than the rest of the tiers. In talking with DDN, the ‘tierdelay’ counter records all operations including internal retries. It would appear that there is some issue on this particular tier resulting in retries being generated.

Figure 6 shows the same metric again but now as a 3D volume.

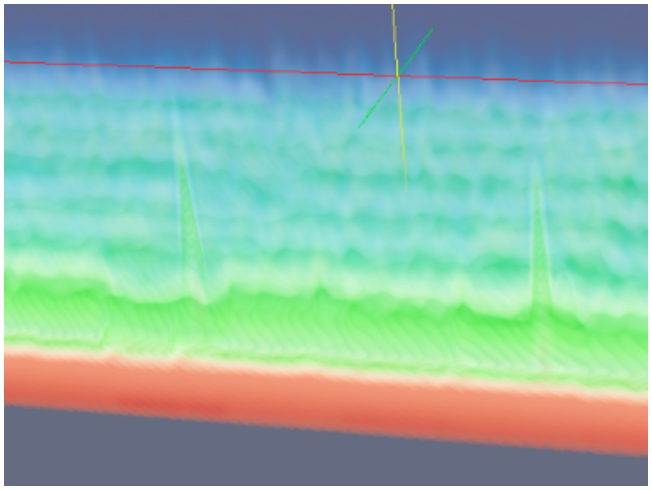


Figure 6 – Tier Delay as Volume

The volume shows a count for the number of operations, which occurred within a defined bucket. For example, 100 operations at 0.2 second delay. The bottom of the volume is the shortest delay and top of the volume is the highest delay. The dark red coloring are higher counts going to blue at the lowest counts. In general the image shows the lowest latency buckets have the highest counts, which is good and the highest latency buckets have the lowest counts, also good. However, you can see a spike on a couple of disks where the higher latency buckets have a much higher total count than most other disks. We don't have conclusive findings that those disk are causing system wide problems, but that is an example of what we hope to find.

CONCLUSIONS

The Darshan deployment has been successful on the LCF Intrepid system. A few projects have used it to tune I/O characteristics to optimize for Intrepid and seen improvements in throughput. We also identified a project that was significant storage user but also suffered from slow I/O performance. We worked with the members of this project to update their code with a slightly modified I/O model that used fewer files which resulted in a 30% improvement of their I/O write speed. We plan to continue to enhance our summarization web tools to provide easier access to the darshan data for the LCF staff.

Our progress on identifying faulty hardware prior to failure on the DDN SANs is still very preliminary and we have not validated any of the results. We hope to progress this further by being able to validate performance improvement after a hardware replacement. We would also hope to identify these patterns so that we could create statistical models that would work on the normal I/O load of Intrepid without the need for an invasive diagnostic run.

ACKNOWLEDGEMENTS

Phil Carns – for all that is Darshan

Neal Conrad – for Darshan web development

Justin Binns – for IO visualizations

REFERENCES

1. Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. 24/7 characterization of petascale I/O workloads. In *Proceedings of 2009 Workshop on Interfaces and Architectures for Scientific Data Storage*, September 2009.
2. Philip Carns, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham, and Robert Ross. Understanding and improving computational science storage access through continuous characterization. In *Proceedings of 27th IEEE Conference on Mass Storage Systems and Technologies (MSST 2011)*, 2011.