A Next-Generation Parallel File System Environment for the OLCF

Galen M. Shipman, David A. Dillow, Douglas Fuller, Raghul Gunasekaran, Jason Hill,
Youngjae Kim, Sarp Oral, Doug Reitz, James Simmons, Feiyi Wang

Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
{gshipman,dillowda,fullerdj,gunasekaran,hilljj,kimy1,oralhs,reitzdm,simmonsja,fwang2}@ornl.gov

Abstract

When deployed in 2008/2009 the Spider system at the Oak Ridge National Laboratory's Leadership Computing Facility (OLCF) was the world's largest scale Lustre parallel file system. Envisioned as a shared parallel file system capable of delivering both the bandwidth and capacity requirements of the OLCF's diverse computational environment, Spider has since become a blueprint for shared Lustre environments deployed worldwide. Designed to support the parallel I/O requirements of the Jaguar XT5 system and other smaller-scale platforms at the OLCF, the upgrade to the Titan XK6 heterogeneous system will begin to push the limits of Spider's original design by mid 2013. With a doubling in total system memory and a 10*x increase in FLOPS, Titan will require both higher bandwidth and larger total capacity. Our goal is to provide a* 4*x increase in total I/O bandwidth from over* $240GB/sec$ *today to* $1TB/sec$ *and a doubling in total capacity. While aggregate bandwidth and total capacity remain important capabilities, an equally important goal in our efforts is dramatically increasing metadata performance, currently the Achilles heel of parallel file systems at leadership. We present in this paper an analysis of our current I/O workloads, our operational experiences with the Spider parallel file systems, the high-level design of our Spider upgrade, and our efforts in developing benchmarks that synthesize our performance requirements based on our workload characterization studies.*

1 Introduction

The Spider file system departed from the traditional approach of tightly coupling the parallel file systems to a single simulation platform. Our decoupled approach has allowed the OLCF to utilize Spider as the primary parallel file system for all major compute resources at the OLCF providing users with a common scratch and project space across all platforms. This approach has reduced operational costs and simplified management of our storage environment. The upgrade to the Spider system will continue this decoupled approach with the deployment of a new parallel file system environment that will be run concurrently with the existing systems. This approach will allow a smooth transition of our users from the existing storage environment to the new storage environment, leaving our users a place to stand throughout the upgrade and transition to operations. A tightly coupled file system environment wouldn't allow this flexibility.

The primary platform served by Spider is one of the world's most powerful supercomputers, Jaguar [1, 12, 5], a 3.3 Petaflop/s Cray XK6 [2]. In the Fall of 2012 Jaguar will be upgraded to a hybrid-architecture coupling the AMD 16-core Opteron 6274 processor running at 2.4 GHz with an NVIDIA "Kepler" GPU. The upgraded system will be named "Titan". The OLCF also hosts an array of other computational resources such as visualization, end-to-end, and application development platforms. Each of these systems requires a reliable, high-performance and scalable file system for data storage.

This paper presents our plans for the second generation of the Spider system. Much of our planning has been based on both I/O workload analysis of our current system, our operational experiences, and projections of required capabilities for the Titan system. The remainder of this paper is organized as follows: Section 2 provides an overview of the I/O workloads on the current Spider system. Operational experiences with the Spider system are presented in Section 3. A high-level system

architecture is presented in Section 4 followed by our planned Lustre architecture in Section 5. Benchmarking efforts that synthesize our workload requirements are then presented in Section 6. Finally, conclusions are discussed in Section 7.

2 Workload Characterization

For characterizing workloads we collect I/O statistics from the DDN S2A9000 RAID controllers. The controllers have a custom API for querying performance and status information over the network. A custom daemon utility [11] periodically polls the controllers for data and stores the results in a MySQL database. We collect bandwidth and input/output operations per second (IOPS) for both read and write operations at 2 second intervals. We measure the actual I/O workload in terms of the number of read/write request with the size of the requests. The request size information is captured in 16KB intervals, with the smallest request less than 16KB and the maximum being 4MB. The request size information is sampled approximately every 60 seconds from the controller. The controller maintains an aggregate count of the requests serviced with respect to size from last system boot, and the difference between two consecutive sampled values will be the number of requests serviced during the time period.

We studied the workloads of our storage cluster using the data collected from 48 DDN "Couplets" (96 RAID controllers) over a period of thirteen months from September 2010 to September 2011. Our storage cluster is composed of three filesystem partitions, called *widow1, widow2, and widow3*. Widow1 encompasses half of the 48 DDN "Couplets" and provides approximately 5 PB of capacity and 120GB/s of bandwidth. Widow2 and Widow3 each encompass $1/4$ of the 48 DDN "Couplets" and provide 60GB/s and 2.5 PB of capacity each. The maximum aggregate bandwidth over all partitions is approximately 240GB/s. We characterize the data in terms of the following system metrics:

- *I/O bandwidth distribution*, helps understand the I/O utilization and requirements of our scientific workloads. Understanding workload patterns will help in architecting and designing storage clusters as required by scientific applications.
- *Read to write ratio* is a measure of the read to write requests observed in our storage cluster. This information can be used to determine the amount of partitioned area required for read caching or write buffering in a shared file cache design.
- *Request size distribution*, which is essential in understanding and optimizing device performance, and the overall filesystem performance. The underlying device performance is highly dependent on the size of read and write requests, and correlating request size with bandwidth utilization will help understand performance implications of device characteristics.

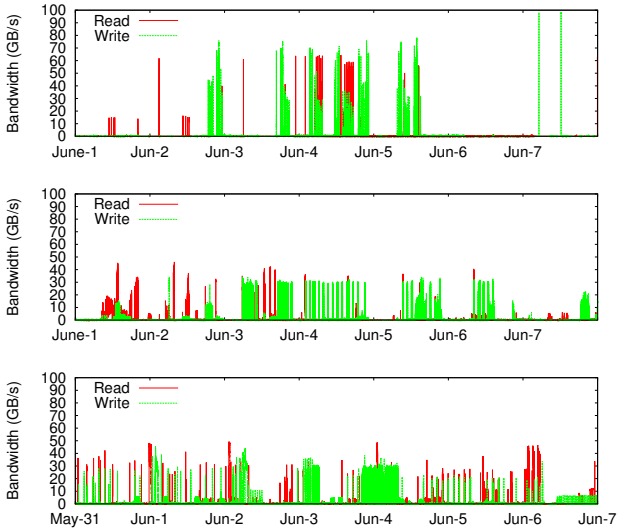

Figure 1: Observed I/O bandwidth usage for a week in June 2011.

Figure 1 shows the filesystem usage in terms of bandwidth for a week in the month of June 2011. This is representative of our normal usage patterns for a mix of scientific applications on our compute clients. We have the following observations from the figure:

- The Widow1 filesystem partition shows much higher bandwidths of reads and writes than those in other filesystem partitions (widow2 and 3). Note that widow1 partition is composed of 48 RAID controllers (24 "Couplets") whereas other partitions are composed of 24 RAID controllers (12 "Couplets"), widow1 is designed to offer higher aggregate I/O bandwidths than the other file systems.
- Regardless of filesystem partitions, utilized bandwidth observed is very low and only several high spikes of bandwidths could be sparsely observed. For example, we can observe high I/O demands, which can be over 60GB/s on June 2, June 3, June 4, however, other days show lower I/O demands. We can infer from the data that the arrival patterns of I/O requests are bursty and the I/O demands can be tremendously high for short periods of time but overall utilization can be dramatically lower than peak usage. This is consistent with application

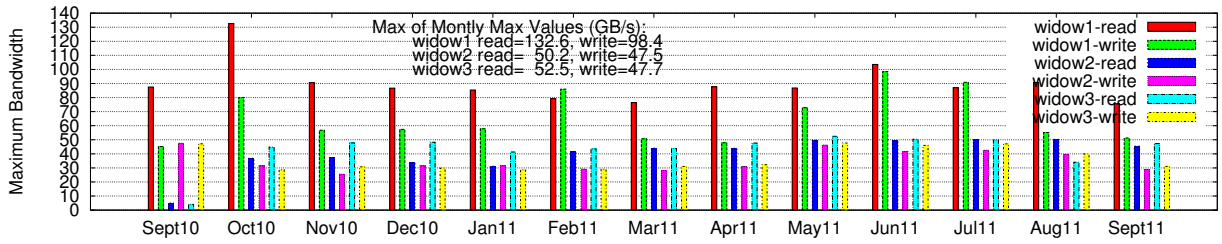

Figure 2: Aggregate read and write maximum bandwidths observed from widow1, widow2, and widow3 partitions.

Checkpoint/Restart workloads. Whereas peaks in excess of $60GB/s$ are common, average utilization is only 1.22GB/s and 1.95GB/s for widow2 and widow3 respectively. These results highly motivate a tiering strategy for next-generation systems with higher bandwidth media such as NVRAM with smaller capacity backed by larger capacity and relatively lower performance hard disks.

Figure 2 shows monthly maximum bandwidths for reads and writes. Overall it is observed from all widow filesystem partitions that max read bandwidth is higher than max write bandwidth. For example, in widow1, the max read bandwidth is about $132GB/s$ where as the max write bandwidth is about $99GB/s$. in widow2, max read bandwidth is $50.2GB/s$ whereas max write bandwidth is $47.5GB/s$. This asymmetry in performance is common in storage media.

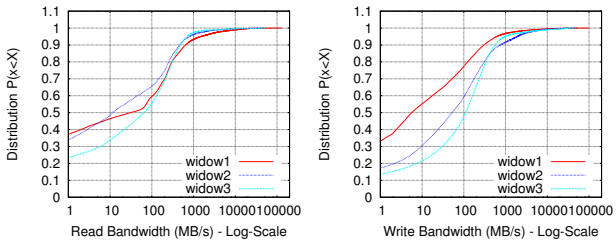

Figure 3: Cumulative distribution of I/O bandwidths for reads and writes for every widow partition.

Our observations further revealed bursty properties of I/O bandwidths from Figure 1. One way of analyzing I/O bandwidth demands is through the use of CDF (Cumulative Distribution Function) plots. In Figure 3, we show the CDF plots of reads and writes for all widow partitions. Similar to our observations that we made [9], the bandwidth distributions for reads and writes follow heavy long-tail distributions, and these trends are observed across all widow filesystem partitions.

For example, in Figure 3(a), we see that read bandwidth for widow1 exceed $100GB/s$ whereas it becomes lower than $10GB/s$ at 90th percentile. It becomes even lower than $100MB/s$ at the 50th percentile. Similar observations can be found in widow2 and 3. The bandwidth can exceed $10GB/s$ at the 99 percentile of the bandwidths, however, it becomes lower than $100MB/s$ at around the 65th and 55th percentiles for widow2 and widow3 respectively. Figure 3(b) illustrates the CDF plot of write bandwidth for widow1, 2, and 3. Similar observations can be found in Figure 3(a). However, it is observed that the max write bandwidths are lower than the read bandwidths, and at the 90th percentile, write bandwidth for widow1 becomes much lower than the read bandwidth.

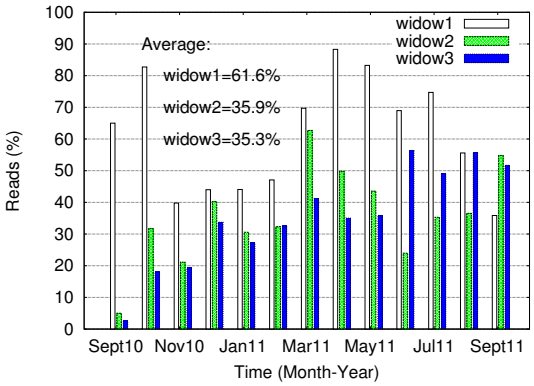

Figure 4: Percentage of read requests observed every month for every widow partition.

Typically scientific storage systems are thought to be write dominant; this is generally attributed to the large number of checkpoints written for increased fault tolerance. However in our observation we see a significantly high percentage of read requests.

Figure 4 presents the percentage of read requests with respect to the total number of I/O requests in the system. The plot is derived by calculating the total read and write requests observed during the 13 month period. On average, widow1 is read-dominant with 61.6% of total requests being reads. However, it's observed that read percentage can exceed 80% of reads. (referring to the percentages of reads in October 2010, April and May

2011 in Figure 4). Compared to widow1, widow2 and widow3 show lower read percentages. Average read percentage is around 35% in both widow2 and 3 partitions. However, we can also observe that the read percentage can exceed 50%. And we also observe that the read percentage increases; For example, average read is 41.1% in 2011 whereas it is 24% in 2010.

Conventional wisdom is that HPC I/O workloads are write dominant due to a lot of checkpointing operations, Spider I/O workloads do not follow this convention. This could be attributed to the center-wide shared file system architecture of Spider, hosting an array of computational resources such as Jaguar, visualization systems, end-to-end systems, and application development systems.

As we studied in our previous work [9], we observe that I/O request sizes from 4-16KB, 512KB and 1MB account for more than 95% of total requests. This is because the request sizes cluster near 512KB boundaries imposed by the Linux block layer. Spider's Luster OSSes use *DM-Multipath*, a virtual device driver that provides fault tolerance. To ensure it never sends requests that are too large for the underlying devices, it uses the smallest maximum request size. If all of those devices support requests larger than 512KB, it uses that size as its maximum request size. The lower lever devices are free to merge the 512KB into larger requests. Lustre also tries to send 1MB requests to storage when possible, thus providing frequent merge opportunities under load.

3 Operational Experiences

In this section we'll cover the reliability of the components of the storage system to date, cover our experience with using an advanced Lustre Networking (LNET) routing scheme to increase performance, and finally discuss our experiences with the Cray Gemini interconnect.

3.1 Hardware Reliability

3.1.1 DDN S2A 9900

The main focus of any storage system's reliability and performace is the disk subsystem. Over the course of 4 years in operation the DDN S2A9900 has been a very stable and productive platform at the OLCF. The performace has met our needs throughout its lifetime so far, and has resulted in very little unscheduled downtime for the filesystems. Of paramount concern as the storage system ages is component failure rates. Based on our current data from Crayport we have experienced an average of 4 disk failures per month since the storage system was brought online in 2009. Figure **??** shows disk failures in Spider over time.

Overall the S2A9900 has been fairly stable and has not required large quantities of component replacement. Additionally the architecture of Spider allows us to have portions of the 9900 fail and not cause an outage. Figure 5 shows a chart of failures by component, and a comparison of failures that result in FRU replacement. The majority of disk failures resulted in FRU replacement. Most of the other compponent failure types had a lower replacement to failure ration.

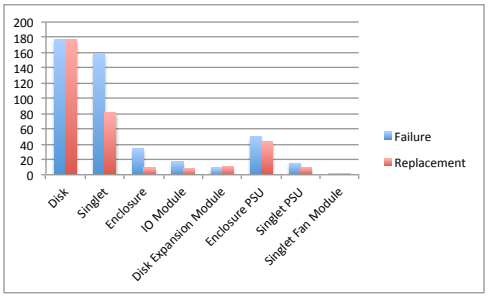

Figure 5: FRU failures in Spider from January 2009 through March 2012.

3.1.2 Dell OSS/MDS nodes

The Lustre servers for Spider are the Dell PowerEdge 1950 for OSS and MGS servers; R900 for the MDS servers. These platforms have been extremely reliable through the life of Spider to date. Our on-site hardware support team has been able to get nodes replaced quickly to re-enable the full features and performance of Spider. Node swaps are less common than disk failures, but on average since January 2009 we replace an OSS node ever 3 months. We take periodic maintenance to upgrade firmware and BIOS on machines. The most common reason to replace a node is bad memory - and it's far easier to work with Dell to get the problem resolved with the node in the Spare pool and not serving production data. The possibility exists to just swap the memory from a spare node but that makes the case tracking much harder when interacting with the vendor.

3.1.3 Cisco and Mellanox IB switch gear

The Spider Scalable IO network (SION) consists of approximately 2000 ports (both host and switch HCA's), and over 3 miles of optical infiniband cabling. The network was designed for fault tolerance – allowing access to the storage resources even if on the order of 50% of the networking gear is down. Since 2009 when Spider

was placed in production there are only 2 service interruptions that were based on issues with SION; one was related to hardware, the other related to the OFED software stack.

3.2 Software Reliability

As the Spider filesystem has transitioned from Lustre version 1.6.5.1 with approximately 100 patches through several versions of 1.8.X; stability has remained very good. That code base is mature and it has resulted an a very reliable and productive platform for the science objectives at the Oak Ridge Leadership Computing Facility. For calendar year 2011 (see Table 1 below), the largest filesystem in the OLCF had 100% scheduled availibility in 8/12 months. Overall for the year it had scheduled availibility of 99.26% – something that was almost unheard of in the version 1.4 days of Lustre. Those numbers are even more impressive when you consider the scale of the system in relation to the size of a Lustre 1.4 installation. The partnership between DDN, Dell, Mellanox, and the Lustre players at Sun Microsystems, Cray, and Whamcloud has provided a roadmap for other centers to remove islands of data inside compute platforms, lower interconnection costs between compute resources, and decouple storage procurements from compute procurements.

Table 1: Spider Availibility for 2011

Filesystem	Scheduled Availibility	Overall Availibility
widow1	99.26%	97.95%
widow2	99.93%	99.34%
widow3	99.95%	99.36%

3.3 Advanced LNET Routing for Performance

In May of 2011 we applied an advanced Lustre Networking (LNET) routing techinque we called Fine-Grained Routing (FGR) that allowed us to achieve much higherbandwidth utilization on the backend storage without any modifications to the user application codes [4]. As can be observed in Figure 6, the maximum aggregate performance graph below there is a considerable dropoff in aggregate performance in August 2011 – when we had to remove the FGR configuration in preparation for the progressive upgrade for Jaguar from XT5 to XK6. In August we saw a 20% performance decrease in the maximum aggregate bandwidth performance for Spider.

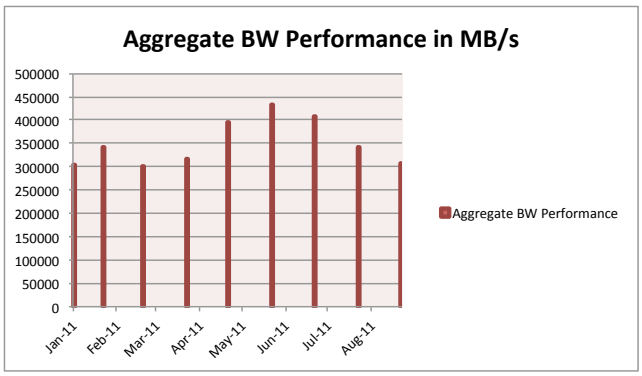

Figure 6: Aggregate bandwitdth to all of Spider for January 2011 through August 2011.

3.4 Gemini Experiences

3.4.1 Filesystem Interaction with Gemini Re-route

Our largest issue related to Gemini in production is documented in Cray bug 780996. Under certian circumstances the L0 component has extremely high CPU load and does not answer a routing request from the SMW, causing the entire machine to destabalize. In all cases where this bug was encountered, a reboot was required to rectify the situation. Work is ongoing with this bug with a near term resolution proposed. Under normal operating conditions we see this bug on the order of once per month. The dynamimc routing feature of the Gemini interconnect has been our biggest win for availibility of the compute resource in moving from the XT5 to the XK6.

3.4.2 LNET Self Test performance

As part of our acceptance suite for the XK6 we ran the LNET Self Test application from the Lustre testing suite. With our stated requirements of 1TB/s of sequential IO performance for the next generation filesystem we needed to verify that the Gemini interconnect could pass enough traffic based on the approximate number of LNET routers we would have in the machine. The metric was at least 500GB/s of performance, and we achieved 793GB/s using 370 XIO nodes; a 2.14GB/s average per node. Additional performance will be required to attain the 1TB/s performance as there are only 512 available service nodes in the machine and they cannot all be LNET routers. Areas where we think performance could be gained would be in making the checksumming routines multithreaded, and possibly changing the algorithm used for computing the checksums.

3.4.3 kgnilnd

At OLCF there is a large knowledge base of tunable parameters for the LNET modules for Seastar (kptllnd), and for tuning on Seastar as well. Transition to Gemini provided us the opportunity and challenge of learning the parameters for an entirely different network technology. Documentation for kgnilnd circa October 2011 was very sparse. There was no baseline configuration that could be referenced, and no official document from Cray that pointed to the module parameters and recommended as well as default values. This made bringing up Spider on portions of the XK6 more difficult as we would make progress and have to send an e-mail to a developer at Cray and wait for a response. Luckily we had planned the schedule to allow for things like this and we were able to go on working on other hardware while we waited for responses.

Interaction between the HSN and SION changes a little on Gemini also. With Seastar the HSN was the bottleneck – with Gemini the IB network is the bottleneck. Care must be taken to allow for router buffers and for monitoring outstanding messages on the router to insure that things are not backing up and causing congestion.

4 System Architecture

OLCF's current center-wide file system, Spider (§4.1), represented a leap forward in I/O capability for Jaguar and the other compute platforms within the facility. Lessons learned from the Spider deployment and operations will greatly enhance the design of the next-generation file system. In addition to a new storage infrastructure, the OLCF's Scalable I/O Network, SION (§4.2) will be rearchitected to support the new storage platform.

4.1 Spider

Spider is the current storage platform supporting OLCF. It was architected to support Jaguar XT5 along with the other computing and analysis systems within the center. The current Spider system is expected to remain at OLCF through 2013/2014.

Spider is a Lustre-based [13][15] center-wide file system replacing multiple file systems within the OLCF. It provides centralized access to petascale data sets from all OLCF platforms, eliminating islands of data.

Spider is a large-scale shared storage cluster. 48 DDN S2A9900 [3] controller couplets provide storage which in aggregate delivers over $240 GB/s$ of bandwidth and over 10 petabytes of formatted capacity from 13,440 1 terabyte SATA drives.

The storage is accessed through 192 Dell dual-socket quad-core Lustre OSS (object storage servers) nodes providing over 14 Teraflop/s in performance and 3 Terabytes of memory in aggregate. Each OSS can provide in excess of 1.25 GB/s of file system level performance. Metadata is stored on 2 LSI Engino 7900s (XBB2) [10] and is served by 3 Dell quad-socket quad-core systems.

A centralized file system requires increased redundancy and fault tolerance. Spider is designed to eliminate single points of failure and thereby maximize availability. By using fail-over pairs, multiple networking paths, and the resiliency features of the Lustre file system, Spider provides a reliable high-performance centralized storage solution greatly enhancing our capability to deliver scientific insight.

On Jaguar XK6, 192 Cray Service I/O (SIO) nodes are configured as Lustre routers. Each SIO is connected to SION using Mellanox ConnectX [14] host channel adapters (HCAs). These Lustre routers allow compute nodes within the Gemini torus network on the XK6 to access the Spider file system at speeds in excess of 1.25 GB/s per compute node. In aggregate, the XK6 system has over 240 GB/s of storage bandwidth.

4.2 Scalable I/O Network (SION)

In order to provide true integration among all systems hosted by OLCF, a high-performance, large-scale InfiniBand [8] network, dubbed SION, was deployed. SION provides advanced capabilities including resource sharing and communication between the two segments of Jaguar and Spider, and real time visualization, streaming data from the simulation platform to the visualization platform at extremely high data rates.

SION will be rearchitected to support the installation of the new file system while retaining connectivity for the existing storage and the other platforms currently connected to it. As new platforms are deployed at OLCF, SION will continue to scale out, providing an integrated backplane of services. Rather than replicating infrastructure services for each new deployment, SION permits centralized, center-wide services, thereby reducing total costs, enhancing usability, and decreasing the time from initial acquisition to production readiness.

4.3 New System Architecture

OLCF's next-generation parallel file system will be required to support the I/O needs of Titan while succeeding the current storage infrastructure as the primary center-wide storage platform. While the final architecture will depend strongly on the storage solution selected and supporting hardware required to build the file sys-

tem, many necessary characteristics are known based on currently understood system requirements.

In its final configuration, Titan will contain between 384 and 420 service nodes for use as I/O routers. These nodes will connect to SION using QDR InfiniBand. Based on the final file system architecture, this QDR fabric will likely exist as a separate stage within SION with cross-connections to the Spider file system.

To fully support the I/O capabilities of Titan, the new file system will require up to 1TB/sec of sequential bandwidth. The system will employ Spider's scalable cluster philosophy to enable a flexible deployment capable of adjustment based on evolving system requirements and budget realities. Additionally, the decoupled nature of the SION network combined with the network agnosticism of LNET will permit OLCF to procure any back-end storage technology supported by Lustre. A conceptual diagram of the new system architecture is presented in Figure 7.

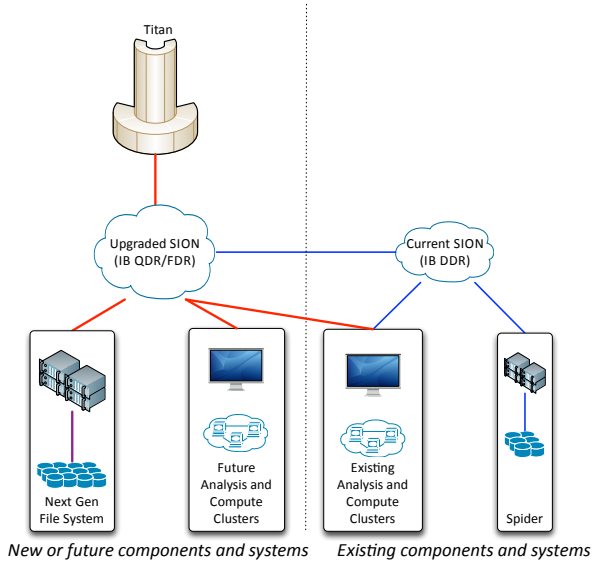

Figure 7: Conceptual diagram of the new system architecture.

5 Lustre Architecture

Spider is a Lustre-based [13, 15] center-wide file system.Implementing Spider drove many enhancements to the Lustre file system in the 1.6 and 1.8 release series, and we expect the next-generation OLCF file system to make use of both OLCF and OpenSFS driven features in Lustre 2.2 and beyond.

Lustre 2.2 saw many improvements in the code base. Many of these directly affect the performance of the file system, and help to improve the scientific productivity of the center:

- *Parallel directory operations(PDO)* improve a long standing performance issue when operating a system at large scale – each modification to a directory is serialized by a single lock, allowing a single operation to proceed at a time. File-per-process IO models combined with ever-increasing core counts significantly increases the impact of this serialization on creating a checkpoint or analysis data set. OpenSFS funded development to split the the directory lock into multiple locks, improving parallelism and reducing the time required to create thousands of files in a single directory.

- *Increased maximum stripe count* allows more OSTs to be used for a single file. Prior to Lustre 2.2, a file was limited to 160 OSTs. With ldiskfs's limit of 2 TB per object, a single file was limited to 320 TB. This ORNL funded development allows up to 2000 OSTs per file, allowing a single file to store 4 PB. Perhaps more important than the increase in file size is the potential performance improvement from spreading the load over more OSTs. Spider currently has over 672 OSTs in its largest file system; these improvements allow a theoretical 4x bandwidth improvement for a single shared file with well formed IO characteristics.

- *Imperative recovery* helps reduce the time required to recover from the inevitable hardware failures seen at scale. In the past, recovery is allowed to take up to three times the OBD timeout to complete. At small scale, this does not add significant delay to recovery, as the default timeout is 100 seconds and smaller client counts both increase the odds of each client performing IO at the time of the failure and decrease the chance of a client failure during the recovery window. However, the scale of OLCF has the opposite effect: it is unlikely that all of the center's resources are performing IO during a failure, and it is much more likely that a client may die during the recovery window. This, combined with operational parameters that required a 6x increase in OBD timeout, caused the recovery window to grow to over 30 minutes.

  Imperative recovery takes advantage of a shared lock on the MGS to note when an OST is restarted – either on the same OSS or another – and causes each client to notice the new location of the OST without waiting for a timeout to occur. While this alone improves the recovery time, the recovery window is also reduced to take advantage of this active notification – we notice dead clients

much sooner. In combination, these ORNL funded changes are on track to reduce the OLCF recovery time from over 30 minutes to well under 5 minutes.

- *Asynchronous glimpse locks* Lustre has over the last several years been improved with added features to deal with traversing directories. Prior to Asynchronous glimpse locks, the client would trigger many RPCs for each file or subdirectory for a specific parent directory. To improve this, statahead was added to Lustre so clients could prefetch file attributes from the MDS asynchronously. This initial design while improving the problem still experienced limitations. Each parallel statahead thread would require access to the VFS layer which introduce race conditions with other operations also needing to access the VFS layer as well. In addition, not all RPCs were processed asynchronously. The glimpse size RPCs used to pre-fetch file size where still synchronous. The solution was to make glimpse size RPCs asynchronous and to take information in the RPC reply to cache it in a statahead local dcache on the client.

- *PTLRPC thread pools* In Lustre 2.2 a generic ptlrpcd pool of threads was developed to handle all the asynchronous RPCs on the client. In the past lustre had 2 threads to handle the RPC load. Often one of those threads would be idle while the the CPU was pegged. As one can see this does not scale well to modern machines. So a pool of threads was created that would take advantage of each CPU core. These threads are then bound to each core to avoid cache misses which degrades overall perform. Even though this helps distribute the CPU load, it is still possible to have one thread experience a delay due to an over utilized CPU. To get the best of both worlds two pools are created. Some threads are always bound to a core and the other threads are free to migrate to the least loaded core. This can be controlled with binding polices set at module load time.

Lustre 2.3 is expected to bring further improvements in performance:

- *4+ MB requests* will help improve storage backend performance when driven under heavy workloads from thousands of clients. While well-formed, sequential IO is able to generate peak bandwidth from the storage system, the IO stream presented quickly becomes random at the block level. With 1 MB write request sizes, block-level performance on Spider's DDN 9900 controllers is reduced to between 35% and 40% of peak unless I/O requests are merged within the controller. Sending 4 MB write requests to the controllers returns performance to above 80% of peak irrespective of merging of requests and achieves above 90% in many cases.

  Lustre 2.3 is expected to increase the maximum RPC size to 4 MB, allowing the OSSes to send larger requests through the object storage file system to the block layer. This will recover much of the performance lost to the highly random IO patterns seen from thousands of clients simultaneously accessing the file system.

- *SMP scalability improvements,* funded by OpenSFS, will improve performance on today's multi-core systems often used for Lustre servers. Currently, Lustre has many points of contention at multiple layers of the stack. At these points, locks may be bounced from core to core, and from socket to socket, thrashing caches and dramatically increasing latency. Additionally, requests may be bounced from core to core while migrating from LNET to the RPC service that will ultimately process it, loosing the benefits of any warm cache it may have generated.

  The SMP scalability improvement work will split locks with high rates of contention and add per-socket and/or per-core queues for LNET and RPC services. This will improve the LNET small message throughput, RPC service rate, and overall responsiveness and throughput for the MDS server.

- *Online Object Index scrubbing* will help protect the file system from corrupted meta-data due to non-graceful shutdowns, as well as restore the ability to perform file-level MDT backups. Lustre 2.x uses an object index to map file ids (FIDs) to inodes in the backing store. If this index is corrupted or deleted due to a file-level backup or other storage issue, Lustre is unable to retrieve user data from storage without manual intervention. OpenSFS has funded work to perform an online scrub of this index in the background of user-initiated operations. This will verify that every FID in the index points to a correct and valid inode, and that every in-use inode has a FID and entry in the object index. This work will form the basis for follow-on work to allow for online integrity checks of the Lustre metadata between MDT(s) and the OSTs, avoiding lengthy down-times to detect and correct missing and/or orphaned data objects.

- *The Network Request Scheduler (NRS)* enhancement will allow the Lustre servers more control over the order in which they processes the RPCs

presented by the clients, leading to better IO patterns to the block scheduler and allowing for different quality of service levels for different users or clients.

Currently, Lustre processes requests from clients in a first-in, first-out (FIFO) manner. This allows a large system to starve out smaller ones due to the disparate request generation rate, or due to many requests for the same resource to eat up service threads waiting for a lock while requests to other resources sit idle in the network queue. NRS allows pluggable algorithms to specify the order in which requests are processed. These algorithms can implement policies that restrict the relative number of requests from a group of clients or could give preferential treatment for contiguous requests to the same data object. While the enabling technology will be introduced in Lustre 2.3, we expect to see community development of additional policy engines once the foundational mechanism is in place.

- *The Object Storage Device (OSD) rework* will allow Lustre to utilize different object storage systems. Currently, Lustre uses an ext4 derivative called ldiskfs to store user data in objects on disk. Lawrence Livermore National Laboratory has been working to allow Lustre to take advantage of the redundancy, management, and performance features of the ZFS filesystem. The OSD rework provides the flexibility to allow both ldiskfs and ZFS to be used as the underlying object store in the near future, and allows for future expansion to BTRFS and other new file systems longer-term.

As we look beyond the expected features in the Lustre 2.3 release, it becomes more difficult to determine when various improvements will become available to the Lustre community. However, there are several improvements being worked on as likely candidates for inclusion in the OLCF next-generation file system:

- *Distributed Namespaces (DNE)* will improve Lustre aggregate metadata performance by allowing multiple MDTs to be used in a single Lustre filesystem. This horizontal scaling will help isolate the metadata demands between groups of users while allowing management of the storage as one aggregated pool. The initial implementation of DNE will allow directories to be created on separate MDTs, spreading the load in different portions of the namespace. Follow-on work will allow a single directory to be striped across multiple MDTs, allowing improvements when thousands of clients are performing operations in that directory, such as during large-scale job startup and checkpointing.

- *Client IO improvements* to prioritize page write-back. It has been observed for some time that individual clients suffer from several IO bottle necks. Lustre currently uses a basic FIFO to cache dirty pages which ties performance to the behavior of the application. This is remedied by merging sequential pages on the client side during formation of requests. Further performance improvement will be gained by prioritizing the write-back of pages associated with a lock being canceled by the OST. This will reduce the time required to release the lock to other clients waiting to write their data. By moving the locking to a per-object basis, this will also reduce lock contention on the clients.

- *Improved performance for directory scans* will improve common use cases such as a user listing the names and sizes of files in a directory or applications scanning for their input deck. Currently, Lustre uses multiple requests to retrieve the names of the files in the directory, the size of the files, and the attributes of the files. While multiple names may be retrieved in one request, obtaining the size and attributes requires two requests per file. This load is reduced by sending all of the information about the file along with the name, reducing the number of requests (and LDLM locks) required from the MDT. While this improvement may not significantly impact application performance, it will increase responsiveness to interactive user commands.

While looking to the future of the OLCF file systems, it is important to not forget the past and to minimize user pain as much as possible during the transition. It is possible to upgrade the existing Spider file systems from 1.8 to 2.2 (and beyond) without requiring a reformat. This has the advantage of not requiring users to move their data to archival storage and restoring it to the new system, but trades off access to many of the new features supported by recent Lustre releases. There is work underway to convert the on-disk format of 1.8 file systems to 2.X, and OLCF will monitor the progress of this effort to provide the least interruption possible to our users.

6 Systematic Benchmarking

The benchmarking suite is a comprehensive suite providing Lustre file system-level and block-level performance metrics for a given file or storage system. The suite also provides tools for quick visualization of the results allowing head-to-head comparison and detailed analysis of system's response to exercised I/O scenarios. Using bash scripts as wrappers to control, coordinate, and synchronize pre-selected I/O workloads, the

suite uses the obdfilter-survey at the file system-level and the fair-lio at the block-level as workload generators. Obdfilter-survey [7] is widely used and exercises obdfilter layer in Lustre I/O stack for reading, writing and rewriting Lustre objects. Fair-lio [4] is in-house developed and libaio-based tool which generates parallel and concurrent block-level sequential and random, read and write asynchronous I/O to set of specified local block-level targets. The benchmark suite is developed as part of OLCF's efforts towards procuring and deploying the next generation center-wide shared Lustre file system for the Titan supercomputer and other OLCF computing, analysis, and visualization resources. The suite is publicly available and can be obtained at the OLCF website [6].

Based on the lessons learned from our Spider file system deployment and operations and also from our I/O workload characterization work outlined in Section 2, our benchmark suite evaluates the file and storage system I/O response in terms of various characteristics, such as, performance and scalability. Our I/O workload characteristics work and experiences with the Spider file system both pointed out that the aggregate I/O workload observed at the file and storage system level is highly bursty, random, and a heavy mix of small (less than 512 kB) and large (512 kB and above) read and write requests. Therefore, our benchmark suite tries to mimic this workload.

As stated, the benchmark suite has a block I/O section and a file system I/O section. The block I/O section consists 4 different benchmarks: single host scale up test (block-io-single-host-scale-up.sh), single host full scale test (block-io-single-host-full-run.sh), scalable storage unit scale up test (block-io-ssu-scale-up.sh), and the scalable storage unit degraded mode full scale test (block-io-ssu-degraded.sh). Of these, the first three are used for assessing the performance and scalability of a healthy system, while the fourth is used for degraded systems. All four benchmarks require pdsh and dshbak utilities to execute. In each of the four benchmarks, the storage system is exercised with random and sequential, small and large read and write I/O operations. For all tests, command line parameters including queue size, block size, and I/O test mode (sequential write, sequential read, random write random read) and the iteration number is generated before the actual execution and then randomized. The randomized list of tests then fed into the benchmark I/O engine and executed. Through this randomization of test parameters and sequence of I/O operations and modes, we eliminate the caching effects on test nodes and the storage system, therefore obtaining much more realistic readouts (based on our Spider experiences and results of our workload study, we know that the I/O traffic is heavily bursty and heavy mix of I/O modes and operations with varying block sizes). Each individual iteration of tests is run for 30 seconds to obtain statistically meaningful results.

Of these four benchmarks, the block-io-single-host-full-run test is used to characterize the performance and scalability of the underlying storage system for a single I/O server perspective. A single SCSI disk block device (sd device) configured on the target host is exercised in this test. There are 720 individual tests in this one benchmark and the total run time of the benchmark will be (720 tests * 45 seconds), or in other words 9 hours. The script will generate a summary file capturing the STDOUT of the script with some additional information and a .csv (comma separated values) results file capturing detailed results of the tests and derived statistics. The script will also create a subdirectory located at the parent directory where the script was launched and write individual raw test results in separate files in this new directory. This benchmark will use 4 kB, 8 kB, 16 kB, 32 kB, 64 kB, 128 kB, 256 kB, 512 kB, 1 MB, 2 MB, 4 MB, and 8 MB block I/O request sizes and sequential and random write and read I/O modes and operations and queue sizes of 4, 8, and 16. A permutation of all these variables is generated as command line arguments before the actual execution of the benchmark and then randomized and then fed into the actual I/O workload generator engine (i.e. fair-lio), as explained above.

The block-io-single-host-scale-up test runs a randomized set of sequential and random write and read I/O benchmarks using the fair-lio binary for various block and queue sizes for all SCSI disk block devices configured on one a single I/O server node for a given scalable storage unit for multiple iterations. Similar to the previous test, this benchmark also generates and randomizes test parameters and test modes and operations before the execution. Also, as outlined in our workload study, I/O block size PDF show three spikes at less than 16 kB, 512 kB, and 1 MB for both read and write operations. These three request sizes were observed to account for more than 95% of total requests. Therefore, to speed up the benchmarking process and to shorten the actual test time, we chose 4 kB, 8 kB, 512 kB, and 1 MB as block sizes. The total run time of the test will be at least log2(number of target test devices) * (144 tests * 45 seconds), or in other words at least 1.8 hours for (log2(number of target test devices)). As an example, if every test host has 5 target test devices, the total run time will be 7.2 hours.

The block-io-ssu-scale-up benchmark will exercise all configured SCSI block devices on all configured test hosts on the scalable storage unit to gather the maximum obtainable performance of the scalable test cluster under

various I/O test modes and operations. This test will again run a randomized set of sequential and random write and read I/O benchmarks using the fair-lio binary for various block and queue sizes for all SCSI disk block devices devices for multiple iterations on all I/O servers. The block and queue sizes selected for this benchmark is identical to those of the block-io-single-host-scale-up benchmark. The total run time of the benchmark will therefore be at least (log2(number of target test devices)) * (144 tests * 45 seconds), or in other words at least 1.8 hours for (log2(number of target test devices)). As an example, if every test host has 5 target test devices, the total run time will be 7.2 hours.

The degraded mode test, block-io-ssu-degraded, is similar to ssu scale-up test, and has an identical sets of block size, queue size, and I/O mode and operation parameters. This test will exercise all SCSI block devices (i.e. RAID arrays or LUNs) on all test hosts and provide the performance profile of the ssu when 10% of the SCSI block devices are being rebuilt. Before running this script it is expected that the tester makes sure that there are at least 10% of the SCSI block devices are in active rebuild state for the entire execution of the benchmark. This script will again run a randomized set of sequential and random write and read I/O benchmarks using the fair-lio binary for various block and queue sizes for all SCSI block devices. The total run time of the test will be (144 tests * 45 seconds), or in other words 1.8 hours.

Also included in the benchmark suite, are tools for parsing and plotting the obtained results for the block I/O benchmarks. These plotting tools require gnuplot and ps2pdf utilities to execute. These two utilities are quite common in HPC environments. A sample block I/O plot is presented in Figure 8.

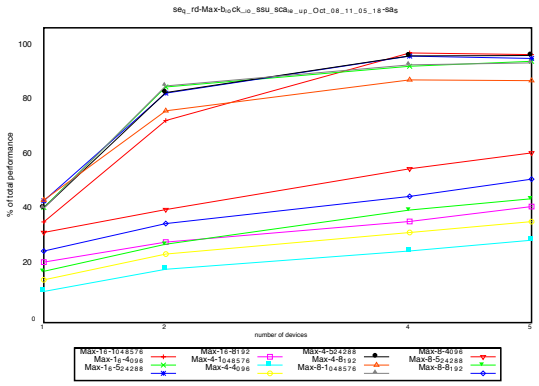

Figure 8: A sample block I/O benchmark plot

For assessing the Lustre-level performance and scalability of a file and storage system, we developed a set of tools built around the obdfilter-survey engine. Our tools generate a required set of parameters and variables and then feed them to the obdfilter-survey. Again, similar to our block I/O tests, these parameters are determined based on our experiences with Spider and our I/O workload characterization study. Our Lustre-level benchmark package includes a obdfilter-survey-olcf script. This script is different than the one that comes with Lustre distributions and includes the required set of I/O parameters and variables. The only modifiable parameters in this benchmark package is the list of OSTs to be tested. There are no other modifiable variables or parameters. This benchmark assumes a fully configured and functional Lustre file system already running on the test hardware. However, Lustre clients are NOT needed to run this benchmark suite. The benchmark is tested against Lustre version 1.8. The benchmark package requires passwordless ssh capability from the head node to the OSSes and between the OSSes, as well. A sample file-system-level benchmark plot is presented in Figure 9.

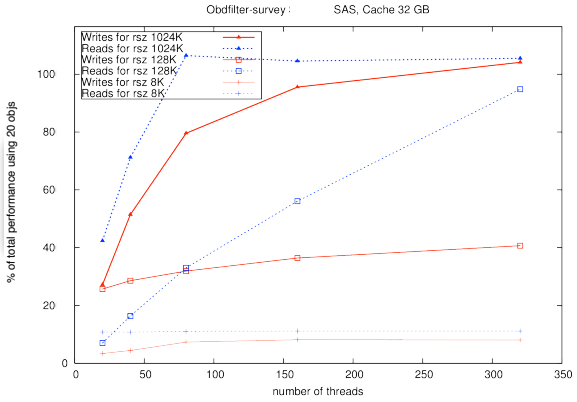

Figure 9: A sample file-system-level benchmark plot

OLCF's benchmark suite was shared with our partners and vendors in 2011 and is publicly available since early 2012. The initial feedback we have received is very encouraging.

7 Conclusions

To support the need of a scalable, high-performance, center-wide parallel file system environment, the OLCF architected, developed, and deployed the Spider system. In 2013, Spider will have been in operation for over 5 years having supported Jaguar XT4, Jaguar XT5, and Jaguar XT6 throughout this time. As the OLCF transitions to the next-generation hybrid Titan system, the Spider system will undergo a major upgrade to meet the performance and capacity requirements of Titan.

The design of our next-generation Spider system will draw upon our operational experiences, I/O workload characterizations, and the performance, capacity, and resiliency requirements of the OLCF. This system will incorporate a number of major changes in Lustre to improve resiliency, performance, and scalability. Similar advances in storage system technologies will be incorporated into this system.

Given the flexible architecture of the current Spider system, this upgrade will be brought online and will be operated concurrently with our current storage systems. This strategy will provide a smooth transition for our users allowing them access to our current storage systems as new storage is deployed and made accessible.

References

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

[2] Cray Inc. Cray XK6. `http://www.cray.com/Products/XK6/XK6.aspx`.

[3] Data Direct Networks. DDN S2A9900. `http://www.ddn.com/9900`.

[4] D. A. Dillow, G. M. Shipman, S. Oral, and Z. Zhang. I/o congestion avoidance via routing and object placement. In *Proceedings of Cray User Group Conference (CUG 2011)*, 2011.

[5] J. Dongarra, H. Meuer, and E. Strohmaier. Top500 supercomputing sites. `http://www.top500.org`, 2009.

[6] O. R. L. C. Facility. Olcf i/o evaluation benchmark suite. `http://www.olcf.ornl.gov/wp-content/uploads/2010/03/olcf3-benchmark-suite.tar.gz`.

[7] O. Inc. Benchmarking lustre performance (lustre i/o kit). `http://wiki.lustre.org/manual/LustreManual20_HTML/BenchmarkingTests.html`.

[8] Infiniband Trade Association. Infiniband Architecture Specification Vol 1. Release 1.2, 2004.

[9] Y. Kim, R. Gunasekaran, G. M. Shipman, D. Dillow, Z. Zhang, and B. W. Settlemyer. Workload characterization of a leadership class storage. In *Proceedings of the 5th Petascale Data Storage Workshop Supercomputing '10 (PDSW'10) held in conjunction with SC'10*, November 2010.

[10] LSI Corporation. 7900 HPC Storage System. `http://www.lsi.com/storage_home/high_performance_computing/7900_hpc_storage_system/index.html`.

[11] R. Miller, J. Hill, D. D. A., G. Raghul, G. M. Shipman, and D. Maxwell. Monitoring tools for large scale systems. In *Proceedings of Cray User Group Conference (CUG 2010)*, 2010.

[12] Oak Ridge National Laboratory, National Center for Computational Sciences. Jaguar. `http://www.nccs.gov/jaguar/`.

[13] Sun Microsystems Inc. Luste Wiki. `http://wiki.lustre.org`, 2009.

[14] S. Sur, M. J. Koop, L. Chai, and D. K. Panda. Performance analysis and evaluation of mellanox connectx infiniband architecture with multi-core platforms. In *HOTI '07: Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects*, pages 125–134, Washington, DC, USA, 2007. IEEE Computer Society.

[15] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang. Understanding lustre filesystem internals. Technical Report ORNL/TM-2009/117, Oak Ridge National Lab., National Center for Computational Sciences, 2009.