Lessons Learned in Deploying the World's Largest Scale Lustre File System

Galen M. Shipman, David A. Dillow, Sarp Oral, Feiyi Wang, Douglas Fuller, Jason Hill, Zhe Zhang

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

Abstract

The Spider system at the Oak Ridge National Laboratory's Leadership Computing Facility (OLCF) is 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, the project had a number of ambitious goals. To support the workloads of the OLCF's diverse computational platforms, the aggregate performance and storage capacity of Spider exceed that of our previously deployed systems by a factor of 6x - 240 GB/sec, and 17x - 10 Petabytes, respectively. Furthermore, Spider supports over 26,000 clients concurrently accessing the file system, which exceeds our previously deployed systems by nearly 4x. In addition to these scalability challenges, moving to a center-wide shared file system required dramatically improved resiliency and fault-tolerance mechanisms. This paper details our efforts in designing, deploying, and operating Spider. Through a phased approach of research and development, prototyping, deployment, and transition to operations, this work has resulted in a number of insights into large-scale parallel file system architectures, from both the design and the operational perspectives. We present in this paper our solutions to issues such as network congestion, performance baselining and evaluation, file system journaling overheads, and high availability in a system with tens of thousands of components. We also discuss areas of continued challenges, such as stressed metadata performance and the need for file system quality of service alongside with our efforts to address them. Finally, operational aspects of managing a system of this scale are discussed along with real-world data and observations.

1 Introduction

The Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL) hosts the world's most powerful supercomputer, Jaguar [2, 14, 7], a 2.332 Petaflop/s Cray XT5 [5]. OLCF also hosts an array of other computational resources such as a 263 Teraflop/s Cray XT4 [1], visualization, and application development platforms. Each of these systems requires a reliable, high-performance and scalable file system for data storage.

Parallel file systems on leadership-class systems have traditionally been tightly coupled to single simulation platforms. This approach had resulted in the deployment of a dedicated file system for each computational platform at the OLCF, creating islands of data. These dedicated file systems impacted user productivity due to costly data transfers and added substantially to the total system deployment cost.

Recognizing the above problems, OLCF initiated the Spider project to deploy a center-wide parallel file system capable of serving the needs of its computational resources. The project had a number of ambitious goals:

1. To provide a single shared storage pool for all computational resources.

2. To meet the aggregate performance and scalability requirements of all OLCF platforms.

3. To provide resilience against system failures internal to the storage system as well as failures of any computational resources.

4. To allow growth of the storage pool independent of the computational platforms.

The OLCF began a feasibility evaluation in 2006, initially focusing on developing prototype systems and integrating these systems within the OLCF. While the OLCF was the primary driver of this initiative, partnerships with Cray, Sun Microsystems, and DataDirect Networks (DDN) were developed to achieve the project's ambitious technical goals. A Lustre Center of Excellence (LCE) at ORNL was established as a result of our partnership with Sun.

This paper presents our efforts in pushing Spider toward its pre-designated goals, and the remainder of it is organized as follows: Section 2 provides an overview of the Spider architecture. Key challenges and their resolutions are described in Section 3. Section 4 describes our approaches and challenges in terms of day-to-day operations since Spider was transitioned to full production in June 2009. Finally, conclusions are discussed in Section 5.

2 Architecture

2.1 Spider

Spider is a Lustre-based [21, 25] 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.

Unlike many existing storage systems based on local high-performance RAID sets for each computation platform, Spider is a large-scale shared storage cluster. 48 DDN S2A9900 [6] controller couplets provide storage which in aggregate delivers over 240 GB/s of bandwidth and over 10 Petabytes of RAID 6 formatted capacity from 13,440 1 Terabyte SATA drives.

The DDN S2A9900 couplet is composed of two singlets (independent RAID controllers). Coherency is loosely maintained over a dedicated Serial Attached Storage (SAS) link between the controllers. This is sufficient for insuring consistency for a system with write-back cache disabled. An XScale processor [9] manages the system but is not in the direct data path. In our configuration, host-side interfaces in each singlet are populated with two dual-port 4x Double Data Rate (DDR) InfiniBand (IB) Host Channel Adapters (HCAs). The back-end disks are connected via ten SAS links to each singlet. For Spider's SATA based system, these SAS links connect to expander modules within each disk shelf. The expanders then connect to SAS-to-SATA adapters on each drive. All components have redundant paths. Each singlet and disk tray has dual power-supplies, one of which is protected by an uninterrupted power supply (UPS). Figure 1 illustrates the internal architecture of a DDN S2A9900 couplet and Figure 2 shows the overall Spider architecture.

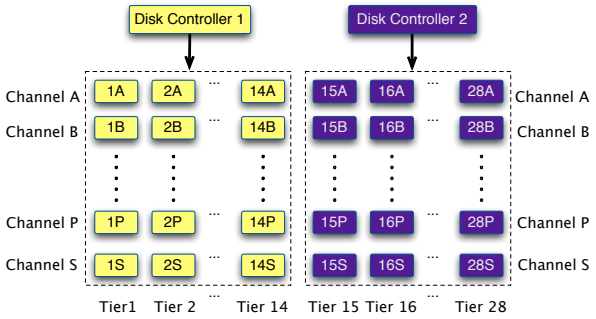

Figure 1: Internal architecture of a S2A9900 couplet.

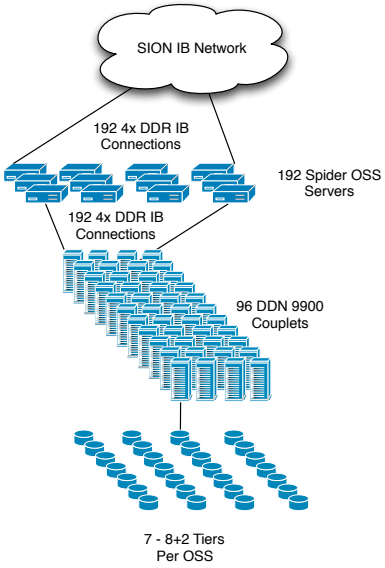

Figure 2: Overall Spider architecture.

The DDN storage is accessed through 192 Dell dual-socket quad-core Lustre OSS (object storage servers) 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) [11] and is served by 3 Dell quad-socket quad-core systems.

Each DDN S2A9900 couplet is configured with 28 RAID 6 8+2 tiers. Four OSSs provide access to these 28 tiers (7 each). OLCF compute platforms are configured with Lustre routers to access Spider's OSSs.

All DDN controllers, OSSs, and Lustre routers are

configured in partnered failover pairs to ensure filesystem availability in the event of any component failure. Connectivity between Spider and OLCF platforms is provided by our large-scale 4x DDR IB network, dubbed the Scalable I/O network (SION).

Moving toward 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 failover 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 XT5, 192 Cray Service I/O (SIO) nodes are configured as Lustre routers. Each SIO node has 8 AMD Opteron cores and 8 GBytes of RAM and are connected to Cray's SeaStar2+ [3, 4] network. Each SIO is connected to SION using Mellanox ConnectX [22] host channel adapters (HCAs). These Lustre routers allow compute nodes within the SeaStar2+ torus to access the Spider file system at speeds in excess of 1.25 GB/s per compute node. The Jaguar XT4 partition is similarly configured with 48 Cray SIO nodes acting as Lustre routers. In aggregate, the XT5 partition has over 240 GB/s of storage throughput while XT4 has over 60 GB/s. Other OLCF platforms are similarly configured.

2.2 Scalable I/O Network (SION)

In order to provide true integration among all systems hosted by OLCF, a high-performance, large-scale IB [8] network, dubbed SION, has been 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. Figure 3 illustrates the SION architecture.

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.

SION is a high-performance multi-stage DDR IB network providing over 889 GB/s of bisection bandwidth. The core network infrastructure is based on four 288-port Cisco 7024D IB switches. One switch provides an aggregation link, two provide connectivity between the two Jaguar segments and the Spider file system, and the

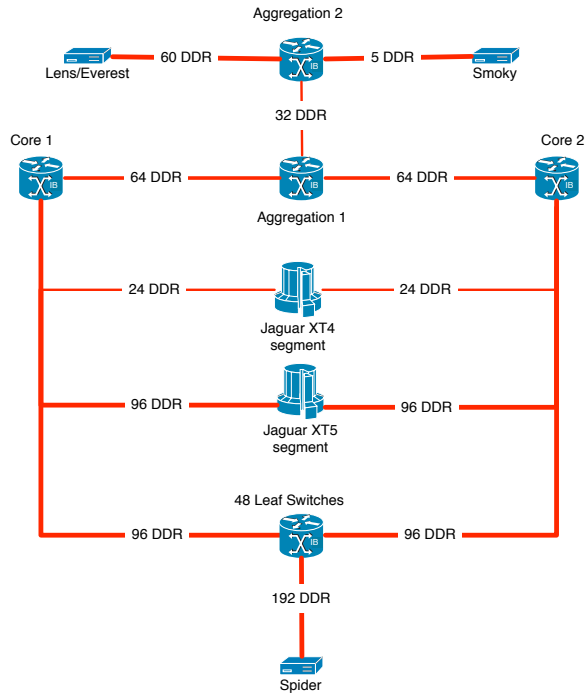

Figure 3: Scalable I/O Network (SION) architecture.

fourth provides connectivity to all other OLCF via the aggregation switch. The Spider is connected to the core switches via 48 24-port IB switches allowing storage to be accessed directly from SION. Additional switches provide connectivity for the remaining OLCF platforms.

SION is routed using OpenSM [24] with its min-hop routing algorithm with an added list of IB GUIDs that have their forwarding entries placed in order before the general population. This enhancement avoided a pathological routing case that resulted in a 33% performance degradation.

3 Integration

To provide a scalable, high-performance, and reliable parallel file system at OLCF, the authors have evaluated various technologies since 2007. IB has gained considerable traction in the HPC community and has been demonstrated at scales exceeding our requirements [18]. The availability of DDR IB, high port-count switches and long-reach (up to 100 meters) optical cabling provided a plausible solution to our system area network requirements. Much of the early work on IB evaluation focused on optical cable testing [12] and porting the OpenFabrics OFED stack to the Cray SIO node.

OLCF began evaluation of the DDN S2A9900 storage system in 2008. OLCF fielded one of the earliest examples of the S2A9900 platform for evaluation and worked with DDN to address a number of performance, stability and reliability issues. During this period, a number of firmware and software level changes substantially improved the S2A9900 storage platform in order to meet the requirements of the Spider file system.

3.1 Establishing a Baseline of Performance

In order to obtain a baseline of performance on the DDN S2A9900 the XDD benchmark [17] utility was used. Initial experiments focused on aggregate performance for sequential read or write workloads. Performance results using XDD from 4 hosts connected to the DDN via DDR IB are summarized in Table 4. The results presented show performance of 5 runs each of sequential read, sequential write, random read, and random write operations using 1MB and 4MB transfers. These tests were run using a single host with a single LUN and 4 hosts each with 7 LUNs which is labeled "multi" in the Tiers column. Of particular interest is the dramatically improved performance of random read and random write operations when transfer sizes are increased from 1MB to 4MB as the cost of the head seek is amortized over a larger write.

Sum - Disk MB/s			IO Type		Pattern	
			read		write	
Req Size	Tiers	Run	random	seq	random	seq
1mb	multi	1	2630.94	5907.87	2541.79	5422.21
		2	2629.95	5918.09	2539.40	5403.04
		3	2630.69	5901.75	2539.11	5379.23
		4	2630.81	5894.38	2538.80	5430.05
		5	2628.30	5916.40	2540.39	5413.06
	single	1	96.44	468.49	94.63	264.43
		2	96.34	471.66	94.41	272.06
		3	96.44	484.79	93.92	284.03
		4	95.96	478.78	94.13	261.35
		5	95.85	476.94	94.40	267.35
4mb	multi	1	4342.12	5421.92	5476.54	5490.39
		2	4337.55	5386.17	5483.57	5480.20
		3	4343.48	5338.70	5490.62	5496.76
		4	4339.00	5391.05	5486.23	5494.29
		5	4341.55	5352.51	5490.71	5477.88
	single	1	254.16	483.54	242.34	376.91
		2	252.69	509.96	242.14	386.55
		3	253.27	411.54	241.47	399.96
		4	256.78	498.00	241.44	377.63
		5	258.24	585.97	241.08	392.12

Figure 4: XDD performance results.

3.2 Improve Filesystem Journaling

After establishing a performance baseline using XDD, Lustre level performance was examined using the IOR benchmark [19]. IOR testing with Spider's DDN S2A9900s and SATA drives on Jaguar showed that Lustre level write performance was 24.9% of baseline performance with a 1 MB transfer size. Profiling the I/O stream using the DDN utilities revealed a large number of 4 KB writes in addition to the expected 1 MB writes. These small writes were traced to *ldiskfs* journal updates[1]. This information allowed identification of bottlenecks in the way Lustre was using the journal – each batch of write requests blocked on the commit of a journal transaction, which added serialization to the request stream and incurred the latency of a disk head seek for each write.

To address these issues a hardware-based solution as well as a software-based one was developed. External journals on solid state devices were used to eliminate head seeks for the journal, which allowed us to achieve 3,292.6 MB/sec or 58.7% of the baseline performance per DDN S2A9900 couplet. The solid-state device used for this testing was a Texas Memory Systems' RamSan-400 device [23] (on loan from ViON Corp.). On the software side, by removing the requirement for a synchronous journal commit for each batch of writes, dramatically fewer 4 KB journal updates (up to 37%) and associated head seeks were observed. This asynchronous journaling method substantially improved the block I/O performance to over 5,222.95 MB/s or 93% of the baseline performance per DDN S2A9900 couplet.

Overall, asynchronous journaling has proven to be a highly efficient solution to the performance problem in terms of performance as well as cost-effectiveness. Findings suggest that sub-optimal object storage file system journaling performance significantly hurts the overall parallel file system performance. A bottleneck from the critical write path was removed by providing an asynchronous write/commit mechanism for the Lustre file system and our approach is likely not specific to our DDN hardware. This solution has been previously proposed for NFSv3 and other file systems, and OLCF was able to implement it in an efficient manner to significantly boost the write performance in a very large scale production storage deployment.

Current understanding and testing shows that OLCF's approach does not change the guarantees of file system consistency at the local OST level, as the modifications only affect how Lustre uses the journal, and not the operation of the journal itself. However, this approach comes with a temporary increase of memory consumption on clients while waiting for the server to commit the transactions. This a fair exchange for the substantial performance enhancement it provides on the very-large scale production parallel file system. More

[1] *Ldiskfs* is the back-end local file system for I/O server in Lustre version 1.6 and is a heavily patched and enhanced version of the Linux *ext3* file system.

details about this work can be found in [15].

3.3 Network Congestion Control

After improving Lustre level performance to over 5,222.95 MB/s or 93% of the baseline performance on a single DDN S2A9900, efforts were focused on examining performance at scale utilizing half of the available Spider storage and Jaguar XT5.

Jaguar XT5's SIO nodes were configured as Lustre routers, forwarding Jaguar XT5's Lustre IO traffic to Spider's OSSs. Decoupling individual system IO nodes from the Lustre OSSs allows each system an independent direct access path to the file systems. Reconfiguring the Lustre routing tables also allows systems' IO network access patterns to be tuned.

To optimize performance and avoid congestion on Jaguar XT5's SeaStar2+ 3D torus network, a mechanism was devised that allowed clients to allocate objects on OSTs that are topologically near the client. Performance was then measured using IOR with each client writing to a single file on an OST that was topologically nearby. Performance was improved substantially as illustrated in Figure 5. In Figure 5 "default read" and "default write" performance was obtained using the IOR benchmark using Lustre's default object allocation policy. The performance results of both "placed read" and "placed write" were obtained using IOR and preallocating files on OSTs topologically near the client writing to this file.

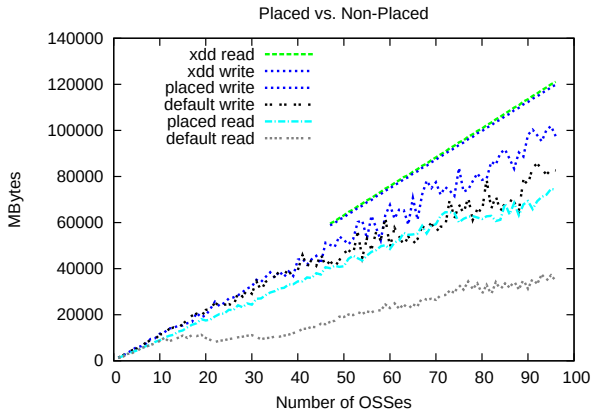

Figure 5: Performance on Jaguar XT5.

Having demonstrated that network congestion can severely impact aggregate file system performance, this problem was tackled in the context of the Spider routing configuration. Lustre clients and servers spread load amongst routers based on queue depths on the router with no other mechanism to detect congestion between a client and a router or a server and a router and prefer routers that minimize network congestion. Recognizing this as a severe architectural limitation, OLCF worked with Sun to provide a mechanism for clients and servers to prefer specific routers. This mechanism allows pairing of clients with routers within the SeaStar2+ torus in order to minimize congestion. In essence, a set of 32 routers can be viewed as a replicated resource providing access to every OST in the file system such that congestion in the IB network is minimized. With 192 total routers and 32 routers in each set, 6 replicated resources were replicated within the SeaStar2+ torus. By grouping these 32 routers appropriately clients were then assigned to these routers such that communication is localized to a 3-D sub-mesh of the torus. This strategy reduces contention in the SeaStar2+ torus based on previous results on OST placement. Figure 6 illustrates this concept using two routing groups on the SeaStar2+ network, each with two routers. Lustre clients in Group B (dark) will prefer the routers indicated by diamonds. The diamond-shaped routers can access all storage via an IB cross bar without resorting to traversing the IB fat-tree. In a similar fashion, Group A clients (light) can utilize the square-shaped (dark) routers to access any storage. Contention on both the SeaStar2+ network and the IB network is therefore minimized.

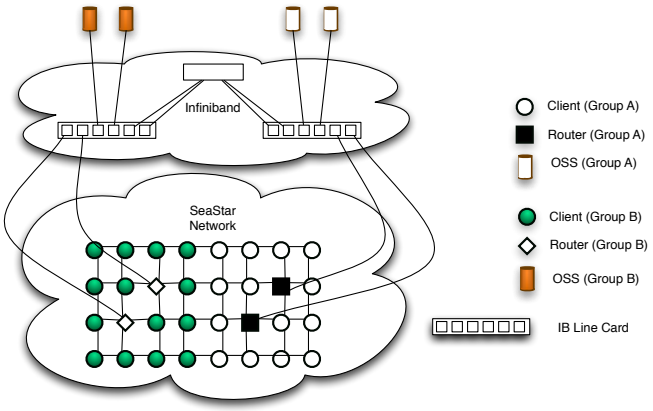

Figure 6: Illustration of Lustre fine-grain routing.

3.4 Scalability

In order to verify the stability of the Spider file system at full scale, testing was conducted using 4 major systems at the OLCF including the Jaguar XT5 and

Jaguar XT4 partition. All systems were configured to mount Spider concurrently totaling over 26,000 Lustre clients and over 180,000 processing cores. In conducting this testing a number of issues were revealed. As the number of clients mounting the file system increased, the memory footprint of Lustre grew at an unsustainable rate. As the memory footprint on the OSSes grew past 11 GB out-of-memory errors (OOMs) on the OSS nodes were observed. By analyzing the memory allocations from Lustre, it was discovered that an extremely large number of 64 KB buffers were being allocated. Reviewing the Lustre code base revealed that 40 KB memory allocations were made (1 for each client-OST connection) which resulted in 64 KB memory allocations within the Linux kernel. With 7 OSTs per OSS and over 26,000 clients this equalled $26000 * 7 * 64\text{KB} = 11.1\text{GB}$ of memory per OSS for server side client statistics alone. As client statistics are also stored on the client, a much more scalable solution as each client would only store $7 * 64\text{KB} = 448\text{KB}$ in the configuration, the server side statistics were entirely removed.

3.5 Fault Tolerance

As Spider is a center-wide resource, much of the testing at full scale centered on surviving component failures. One of the many fault-tolerance techniques used is the Linux kernel dm-multipath [16]. This allows Spider to sustain I/O and provide filesystem resources to the compute platforms during a failure of one or many components. DDN controller failure, Infiniband cable failure, or Infiniband HCA failures are all examples of modes where performance will be degraded, but the filesystem will continue to function. In other implementations without dm-multipath, any one of these failures will cause an outage for all compute platforms that use the filesystem. This architecture has prevented 13 unscheduled outages since June 2009.

A major concern of the center wide file system approach was the impact of an unscheduled outage of a major computational resource utilizing the Spider file system. To test Spider's fault-tolerance at scale, the file system was mounted on all the compute nodes spanning the Jaguar XT4 partition and two other compute platforms. With an I/O workload active on Jaguar XT4, and two other compute platforms, the Jaguar XT4 system was rebooted. Shortly thereafter the file system became unresponsive. Postmortem analysis of this event showed that the OSSes spent a substantial amount of time processing client evictions. Using the DDN S2A9900 performance analysis tools, a large number of small writes to each OST was observed during this eviction processing with very little other I/O progressing on the system. This was later tracked down to a synchronous write to each OST for each evicted client resulting in a backlog of client I/O from Lens and Smoky. Changing the client eviction code to use an asynchronous write resolved this problem and in later testing allowed us to demonstrate the file system's ability to withstand a reboot of either Jaguar XT4 or Jaguar XT5 with minimal impact to other systems with active I/O. Figure 7 illustrates the impact of a reboot of Jaguar XT4 with active I/O on Jaguar XT4, and two other compute platforms. The y-axis shows the percentage of peak aggregate performance throughout the experiment. After 206 seconds, Jaguar XT4 is rebooted. RPCs timeout and performance of the Spider file system degrades substantially for approximately 290 seconds. Aggregate bandwidth improves at 435 seconds and steadily increases until we hit the new steady state performance at 524 seconds. Aggregate performance does not return to 100% due to the mixed workload on the systems and the absence of Jaguar XT4 I/O load.

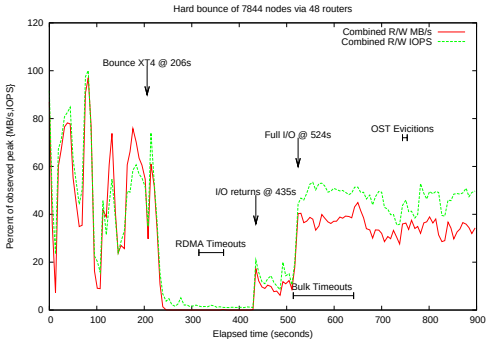

Figure 7: Impact of Jaguar XT4 reboot on aggregate Spider I/O performance.

4 Day-to-day Operations

4.1 Monitoring Tools

Monitoring the components of a file system at this scale is a daunting task, as the health of a single component can affect the performance seen by users across all OLCF compute platforms. Currently the health of the OSS nodes (Infiniband connectivity, OST mounts, multipath health, ping, ssh, load, and physical server information –power supply status and voltage, fan speed, and ambient and CPU temperatures–) is monitored via Nagios [13]. Additionally, centralized *syslog* is parsed for messages that are known to be indicative of problems

with the file system. Also the *ddntool* application is being used to get near-time information about the impact of user jobs on the file system, as well as performance at the DDN controller level. Finally a periodic scan of the processed RPC's at the MDS generates information that can correlate MDS load to specific jobs.

Analyzing *syslog* messages is essential for debugging and monitoring purposes. However, in large-scale parallel systems, interpreting the *syslog* is a challenge because of the complex inter-dependencies among system events. To interprete these logs for system diagnostics, machine learning and data mining techniques are used. Most often, these error or event log messages are best understood when they are interpreted collectively rather than being read individually. The log messages are clustered in a sliding time widow by grouping them based on the commonalities associated among the log events. As an example, let us consider Lustre Cray Portals layer messages, which are common in the Jaguar console log messages. Within a certain time widow (one-minute, without loss of generality), identical error messages with a common source (generating node) or target type are identified. If all messages are occuring from a common source (e.g. compute nodes), then the relation between these source nodes (e.g. are they running a common application, physically located in the same column or cabinet, or are they one-hop neighbors in the 3D Torus interconnect?) are identified. If error messages have a common target (e.g. an OST) then the health and routes leading to that particular target are investigated. Although this work is still in development and testing phase, by intelligent grouping and correlation of events from multiple sources, failure events affecting the system performance are identified.

The DDN S2A9900 appliances have an API for monitoring a variety of parameters. In order to monitor all 48 couplets, a tool was developed to poll each couplet individually, aggregate the results and store them in a MySQL [20] database. Parameters being monitored include alarm data such as failed power supplies, excessive temperatures and failed disks, performance data such as RAID rebuild and verify status, current IOPS and bandwidth status and response times for data requests.

A number of separate programs have been written that query the hosting database for specific information. With the support from MySQL, these programs are simple and easy to maintain. For example, the tool to monitor for any temperature alarms requires a single SQL query and less than 50 lines of Python code.

To detect performance degradation of metadata operations, a rudimentary monitoring service is run at 5 minute intervals, which measures the time to execute *ls* in a directory containing a large number of files. When this service detects performance degradation, an alert is sent to the operations staff. The alert is then manually analyzed using the RPC statistics collected at the MDS. Correlations between application workloads and MDS RPC traces are then preformed to isolate which applications may be causing global performance degradation from pathological workloads.

4.2 Hardware Failures

For the seven month period of June 2009 through January 2010 hardware failure data was gathered and is shown in Figure 8. There were 82 hardware failures. However, none of these failures were catastrophic and none caused data loss. 63.4% of all these were failing disks as illustrated in Figure 8. Figure 9 presents the disk failure breakdown by month. Although disk failures are the dominant source of failures, Spider's RAID 6 configuration prevented data loss in all cases. Figure 8 also tells that power supply failures constitute 12.2% of all failure cases. This is important since the aforementioned failure model did not take into account power supply failures. There were also 15 S2A9900 singlet failures. Of these, 13 were fully transparent to the users, and did not caused data corruption or loss or unscheduled system downtimes. Of the remaining 2 cases, one was due to system operators' error and the other was traced to a bug in the dm-multipath code path. Both of these two cases caused unscheduled temporary system downtimes but did not results in data corruption or loss.

4.3 Metadata Scaling Problems

Center-wide file systems operating under production conditions must handle pathological behavior often exhibited by new applications and existing applications scaling to previously unprecedented sizes. Especially in the latter case, application authors (domain scientists) often lack tools and opportunities for evaluating I/O performance characteristics of their applications at scale. Center-wide file systems exacerbate these issues by transmitting the performance impact of pathological application I/O workloads to all file system users on all attached compute platforms. Therefore, pathological I/O workloads must be carefully isolated, profiled, corrected, and tested to maintain sufficient sustained performance for other applications.

Among such applications presenting pathological I/O behavior on OLCF platforms, application A is a Density

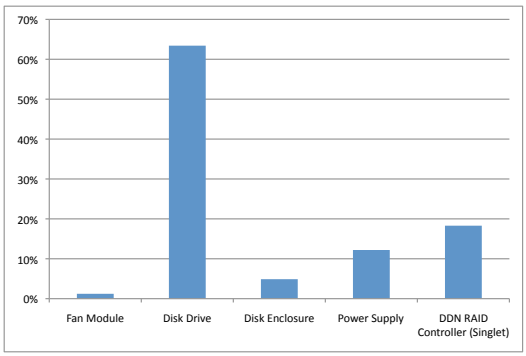

Figure 8: Spider hardware failures

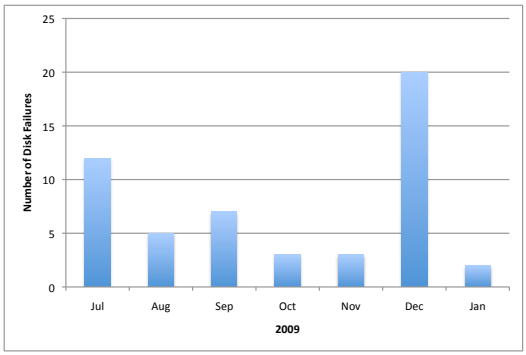

Figure 9: Spider disk failures

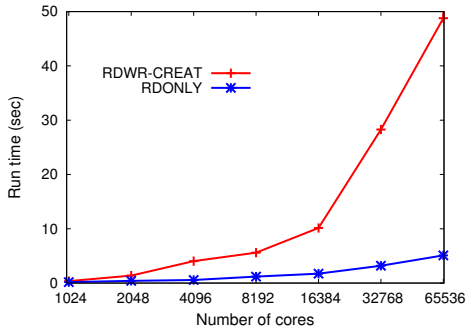

Figure 10: Performance of opening per-process files.

Functional Theory code used to determine the properties of atomic nuclei in detail. Given its inherent low I/O requirements, optimization and parallelization efforts were focused on compute performance. When run for the first time at scale (48,000 processes on Jaguar XT5/Spider), I/O performance issues limited its scalability and affected overall file system performance.

Application A's primary I/O workload consists of scanning a short ASCII parameter file by all compute threads. The serial implementation read variable values by opening, scanning, reading, then closing the input file for each variable. The parallel implementation used the same method, resulting in 48,000 processes rapidly and repeatedly opening and closing the same file and thus overloading the file system metadata server. As a result, overall center-wide file system performance was significantly reduced during this period. The concomitant degradation in interactive filesystem responsiveness also impacted center-wide throughput by hindering user operations.

Lustre file system performance analysis tools developed at the OLCF were used to isolate and diagnose the performance degradation. The serialized I/O code was then reworked to improve scalability. Replacing the input routine with a single reader followed by a network broadcast improved overall application performance by a factor of two. It also alleviated the metadata congestion and associated effects.

To better identify the metadata scaling problem and its effects, in-house benchmarks were developed. These benchmarks are executed on the separated file system; results are presented in figures 10 and 11.

Figure 10 illustrates the time it takes for $1,024 \sim 65,536$ per-process files to be opened in parallel, in the *RDWR-CREAT* (passing the *O_RDWR* and *O_CREAT* flags to the file system) and *RDONLY* modes, respectively. A synchronization barrier is enforced after all

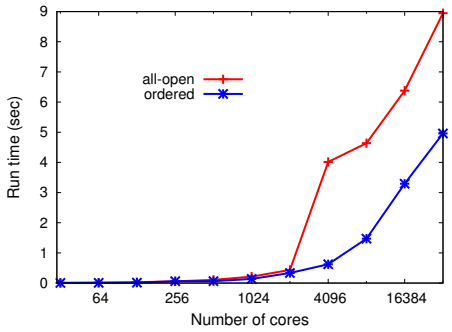

Figure 11: Performance of opening shared file.

files have been opened, and the cost of this entire phase is reported (this behavior is similar to the applications discussed above). The *RDONLY* mode keeps the execution time below 5 seconds. But in the *RDWR-CREAT* mode the time grows fast and reaches nearly 50 seconds with 65,536 processes, mainly because that each process needs to lock the containing directory. This causes a large number of Lustre RPCs waiting for the lock to be released, not only slowing down the application itself but also affecting other applications by exhausting the server's thread pool.

Besides file-per-process, another commonly used model of parallel I/O is for all processes to write to a single shared file. Figure 11 shows the cost of opening a single shared file from $32 \sim 32,768$ processes in parallel with two different strategies. In "all-open," each process passes the *O_RDWR* and *O_CREAT* flags to the file system and attempts to create the file if it doesn't exist. In the "ordered" method, the root process creates the file and all other processes use *WRONLY* mode to write to the file. Although file open time increases with the number of processes in either mode, "ordered" has a clear win with larger than 2,048 processes. The difference, again, is caused by different locking requirements – in the "all-open" mode, although only one process will successfully create the file, all other processes acquire the lock for the parent directory in order to check if the file exists or not. This further illustrates the severe impact of parent directory lock contention.

Based on the observations from applications and benchmarking results, the following optimization techniques will be investigated to enhance the scalability of metadata operations on the Spider system.

- Directory fanout: Create a directory hierarchy for file-per-process I/O. This strategy reduces lock contention, which should enable Lustre to handle the metadata volume more efficiently.

- I/O Aggregation: Collect bulk I/O at a number of delegated I/O worker processes that interact with the file system. This strategy reduces metadata load for the same amount of simulation data.

- I/O middleware: Utilize I/O middleware layers to parametrize I/O strategies. This approach would likely require major changes to the underlying code.

OLCF's existing toolset and test file system will allow for comprehensive evaluation of these techniques in the near future.

5 Conclusions and Future Work

In collaboration with Cray, SUN, and DDN, the Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL) has successfully architected and integrated a center-wide file system capable of supporting over 26,000 clients and delivering 240 GB/sec of I/O bandwidth. The OLCF then deployed this file system to production. To achieve this goal, a number of unique technical challenges were met. Designing a system of this magnitude required careful analysis of failure scenarios, fault tolerance mechanisms to deal with these failures, scalability of system software and hardware components, and overall system performance. Through a phased approach of deploying and evaluating prototype systems, deployment of a large scale dedicated file system followed by a transition to Spider, we have delivered one of the world's highest performance file systems. The Spider file system is now in full production use by all platforms and projects at the OLCF.

While a large number of technical challenges have been addressed during the Spider project, a number still remain. Performance degradation during an unscheduled system outage may last for up to 5 minutes as detailed in Section 3. The OLCF is working closely with file system engineers from Sun and Cray in order to minimize or eliminate this degradation entirely. Fine-grained routing in the LNET layer is currently accomplished by creating multiple distinct LNET networks. Testing of this approach is currently underway and some success is reported although the complexity of the configuration is daunting.

The OLCF's reliability model needs to be updated in light of the gathered hardware failure data. Future work also includes implementing more in-house tools for aggregating all Lustre monitoring data together, and deployment of the Lustre Monitoring Toolkit (LMT) [10].

The system log monitoring and data mining work is being extended to perform real-time log analysis, modeling failure patterns and, if possible, predicting failures. Work for increasing metadata scalability by collaborating with the OLCF's domain scientists for evaluation and implementation of various metadata optimization techniques such as directory fanout, I/O aggregation, and development and deployment of more efficient I/O middleware is continuing. The OLCF's existing benchmarking and analysis tool set and testbed capabilities will allow for comprehensive evaluation of these techniques in the near future.

References

[1] S. R. Alam, J. A. Kuehn, R. F. Barrett, J. M. Larkin, M. R. Fahey, R. Sankaran, and P. H. Worley. Cray xt4: an early evaluation for petascale scientific simulation. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[2] 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.

[3] R. Brightwell, K. Pedretti, and K. D. Underwood. Initial performance evaluation of the cray seastar interconnect. In *HOTI '05: Proceedings of the 13th Symposium on High Performance Interconnects*, pages 51–57, Washington, DC, USA, 2005. IEEE Computer Society.

[4] R. Brightwell, K. T. Pedretti, K. D. Underwood, and T. Hudson. Seastar interconnect: Balanced bandwidth for scalable performance. *IEEE Micro*, 26(3):41–57, 2006.

[5] Cray Inc. Cray XT5. `http://cray.com/Products/XT/Systems/XT5.aspx`.

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

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

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

[9] Intel Corporation. Intel XSacle Technology. `http://www.intel.com/design/intelxscale/`.

[10] J. Garlick. Lustre Monitoring Toolkit (LMT). `http://code.google.com/p/lmt`.

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

[12] M. Minich. Infiniband Based Cable Comparison. Technical report, Oak Ridge National Laboratory, June 2007.

[13] Nagios Enterprises. Nagios. `http://www.nagios.org`.

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

[15] S. Oral, F. Wang, D. Dillow, G. Shipman, R. Miller, and O. Drokin. Efficient object storage journaling in a distributed parallel file system. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST '10)*, 2010.

[16] Red Hat. DM-multipath. `http://www.redhat.com/docs/manuals/csgfs/browse/4.6/DM_Multipath/MPIO_description.html`.

[17] T. Ruwart. XDD. `http://www.ioperformance.com/`, 2009.

[18] Sandia National Laboratories Technical Report. Thunderbird Linux Cluster ranks 6th in Top500 supercomputing Race. `http://www.sandia.gov/news/resources/releases/2006/thunderbird.html`.

[19] H. Shan and J. Shalf. Using IOR to analyze the I/O performance of XT3. In *Proceedings of the 49th Cray User Group (CUG) Conference 2007*, Seattle, WA, 2007.

[20] Sun Microsystems. MySQL. `http://www.mysql.com`.

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

[22] 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.

[23] Texas Memory Systems Inc. Ramsan-400. `http://www.ramsan.com/products/ramsan-400.htm`.

[24] A. Vishnu, A. R. Mamidala, H.-W. Jin, and D. K. Panda. Performance modeling of subnet management on fat tree infiniband networks using opensm. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 18*, page 296.2, Washington, DC, USA, 2005. IEEE Computer Society.

[25] 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.