Best Practices and Lessons Learned from Deploying and Operating Large-Scale Data-Centric Parallel File Systems

Sarp Oral, James Simmons, Jason Hill, Dustin Leverman, Feiyi Wang, Matt Ezell,
Ross Miller, Douglas Fuller, Raghul Gunasekaran, Youngjae Kim, Saurabh Gupta, Devesh Tiwari
Sudharshan S. Vazhkudai, James H. Rogers, David Dillow, Galen M. Shipman, Arthur S. Bland

Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory, USA

***Abstract*—The Oak Ridge Leadership Computing Facility (OLCF) has deployed multiple large-scale parallel file systems (PFS) to support its operations. During this process, OLCF acquired significant expertise in large-scale storage system design, file system software development, technology evaluation, benchmarking, procurement, deployment, and operational practices. Based on the lessons learned from each new PFS deployment, OLCF improved its operating procedures, and strategies. This paper provides an account of our experience and lessons learned in acquiring, deploying, and operating large-scale parallel file systems. We believe that these lessons will be useful to the wider HPC community.**

I. Introduction

The Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory (ORNL) has a long history of deploying and operating some of the world's fastest supercomputers. OLCF is home to the Titan [3] supercomputer that is currently ranked No. 2 on the Top500 list [7] of supercomputers, with a peak performance of 27 Petaflops (PF). Titan is a Cray XK7, a hybrid architecture with 18,688 compute nodes, with each node pairing a 16-core AMD Opteron processor with an NVIDIA GK110 GPU. Titan contains more than 710TB of aggregate memory (600 TB DDR3-1600 SDRAM and 110 TB GDDR5 SDRAM).

The compute power and scale of OLCF systems are key to achieving the U.S. Department of Energy's (DOE) mission for faster scientific discoveries and insights via computational simulations. Researchers from scientific domains including climate, combustion, fusion, astrophysics, and material science run massively parallel scientific simulations on Titan. Many scientific phenomena are better understood when the corresponding scientific simulation is run at a higher resolution (e.g. in time or space), or with more representative physics, at a higher computational complexity. Consequently, these simulations frequently run at very large-scales, concurrently using tens or hundreds of thousands of cores. The scientific workflow then contains a series of data analysis and visualization jobs, run on smaller clusters, to glean insights from the vast amounts of data produced by the simulation [20].

This coupling and the pipelined process of scientific discovery requires data sharing among the compute resources. To adequately meet the users' data-sharing requirements, we must understand the tradeoffs among available parallel file systems. A commonly used design paradigm is the machine-exclusive PFS model, where the PFS is tightly-coupled with a single compute resource. This approach is easier to deploy, but incurs excessive data movement costs to support the scientific workflow. Alternatively, a *data-centric* model is loosely coupled with multiple systems accessing the PFS. This eliminates the need for excessive data movement and promotes efficient data sharing. However, a data-centric PFS design and deployment is more complex, requiring careful consideration of the competing workloads.

Recognizing the long-term benefits of a data-centric system, the OLCF made a strategic decision in 2005 to move away from the traditional machine-exclusive PFS model to a center-wide, data-centric model. The challenge is to retain the flexibility of data sharing that comes with the data-centric model, while still delivering the desirable performance characteristics of a dedicated file system. In the data-centric model, the PFS is shared by all of the compute resources at the same time, requiring it to meet the data characteristics of different workloads. The design, procurement, deployment, testing, performance tuning, and operating of such a system is significantly more challenging than the machine-exclusive PFS systems.

In the past decade, the OLCF has engaged in two large-scale storage system procurements (*Spider I [28] and Spider II [27]*), independent of the computing resource acquisitions for the Jaguar [2] and Titan [3] supercomputers. Spider I and II are data-centric, center-wide shared parallel file systems. Both systems were designed to meet the needs of a wide variety of OLCF user workloads, incorporating supercomputers, data analysis clusters, and remote data transfer systems. Spider I provided 240 GB/s peak I/O bandwidth and 10 PB of data storage; Spider II provides more than 1 TB/s of peak I/O bandwidth and 32 PB of storage. Both systems were built around the Lustre parallel file system [34], [30], an open-source, community supported file system that is in use on more than 70% of the Top500 systems [7].

The entire process, from design to operations presents many challenging opportunities to HPC technology integrators and administrators, who face numerous decisions in the large-scale acquisition and deployment process. Storage system performance and capacity estimations must meet the needs of

978-1-4799-5500-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SC.2014.23

future scientific application workloads, which share the PFS.

This paper summarizes our best practices and lessons learned during designing, procuring, deploying, and operating large-scale PFS during the last decade at OLCF. We believe that our experiences can be applied by the HPC community to design and operate scalable file systems that meet the needs of their users.

II. Design Principles

The traditional model of designing a PFS centers around a single computational resource, which is tightly coupled with the scratch PFS and has exclusive access to it. Frequently, the scratch PFS is procured through the same contract as the computational resource. In designing such a machine-exclusive scratch file system, one needs to only consider the current and anticipated I/O requirements of a single system. Consequently, the design is straightforward. However, this model does not fit well with the I/O requirements of a collection of computing resources that must operate together and share data in a data-centric environment.

To accomplish data sharing in a tightly-coupled machine-exclusive design, one must link together the various machine specific PFS instances via a data movement cluster, which would orchestrate data migration jobs upon user requests. This scenario is not transparent to the user. In the absence of a smart data staging utility, the user is explicitly responsible for data migration before performing analysis or visualization. Further, the multiple namespaces can be confusing to the user and hinder productivity. Additionally, this scenario can be very costly, as the PFS can easily exceed 10% of the total acquisition cost. One must also consider the cost of additional infrastructure and interconnect hardware for transferring the data.

In the data-centric approach, data comes first. The data-centric model focuses on sharing data among multiple computational resources by eliminating unnecessary and expensive data movements. At the same time, it strives to serve diverse I/O requirements and workloads generated on disparate computing resources.

In very simple terms, HPC centers are service providers, where multiple specialized computational resources provide different services. Some of these resources are designed to run simulations and generate data. Others provide post-processing or analysis of the generated data, while some are responsible for visualization of the raw or reduced data. The common denominator in all of these scenarios is data. Data must be shared among these systems in an efficient manner. Moreover, the users should not face unnecessary challenges in accessing their data from various systems. Motivated by these principles, in 2005 the OLCF moved to a data-centric PFS model. Key design principles in this effort include *eliminating data islands, reducing acquisition and deployment costs, and improving data availability and system reliability*.

- **Eliminate data islands.** Users working on one resource, such as a visualization system, should be able to directly access data generated from a large-scale simulation on the supercomputer and must not resort to transferring data. Maintaining the data consistency and integrity across multiple file systems is a distraction for the users.
- **Reduce cost.** In addition to poor usability, machine-exclusive file systems can easily exceed 10% of the total acquisition cost.
- **Improve data availability and reliability.** A scheduled or an unscheduled downtime on a supercomputer can render all data on a localized file system unavailable to the users under the machine-exclusive PFS model.

The data-centric, center-wide file system design provides seamless access to data from a variety of computational resources within our center. Users need not concern themselves with data transfers between distinct file systems within this environment, rather their data is directly accessible from globally accessible namespaces. User feedback describes data-centric capability as critical to their productivity.

However, there are several challenges in designing a data-centric PFS. Data consumers and producers on multiple compute platforms compete for the same shared resource. Each compute resource, based on its primary use case, generates different I/O workloads at different rates. As an example, large-scale simulations running on Titan often consume a large percentage of the available I/O bandwidth and/or IOPs during application checkpoints. These write-heavy checkpoint/restart workloads can create tens or even hundreds of thousands of files and generate many terabytes of data in a single checkpoint. These workloads are bandwidth constrained. In contrast, the data analytics I/O workloads, such as visualization and analysis, are latency constrained and read-heavy.

We note that these different I/O patterns, in isolation, are harmless and can be serviced more easily by a machine-exclusive scratch file system. However, in a data-centric model, they pose a problem as a mixed workload, competing for the same shared resource. A shared scratch file system experiences these I/O workloads as a mix, not as independent streams. Our analysis of the I/O workloads on Spider I PFS [14] demonstrated a mix of 60% write and 40% read I/O requests. In some cases, competing workloads can significantly impact application runtime of simulations or the responsiveness of interactive analysis workloads. Write and read streams from different computing systems often interfere because of the difference in data production/consumption rates. The I/O streams may not be easily synchronized between different computing systems because the resources are shared among multiple users and applications. One must pay attention to overall mixed workload, not to individual machine specific I/O patterns, when designing a data-centric PFS. Our analysis [14] showed that a majority of I/O requests are either small (under 16 KB) or large (multiples of 1 MB), where the inter-arrival time and idle time distributions both follow a long-tail distribution that can be modeled as a Pareto distribution. We utilized these and other metadata specific characteristics to optimize Spider metadata servers that are exercised more heavily under varying I/O workload patterns.

***Lesson Learned** 1: Data-centric center-wide file systems provide significant benefit to the users in terms of ease of access to their datasets across multiple platforms, but at the*

cost of increased contention for a single shared resource. System architects must weigh the tradeoffs between ease of data access and the ability to isolate compute platforms from competing I/O workloads. Weighing these tradeoffs effectively requires a good understanding of the full breadth of application workloads (both simulation and data analytics) and user requirements for data accessibility.

***Lesson Learned** 2: While designing a data-centric shared file system, one must account for detailed I/O workload characteristics. Peak read/write performance cannot be used as a simple proxy for designing a scratch file system, because it may result in either over-provisioning the resources or suboptimal performance due to a mix of I/O patterns. Good random performance translates to better operational conditions across a wide variety of application workloads.*

III. Procurement

A. RFP Considerations and Development

The acquisition phase is a critical point in any deployment. Spider II's Statement of Work (SOW) was assembled with a few key principles in mind. It was targeted to elicit a diverse set of proposals that could be evaluated within a budget range. It requested multiple file system and storage technologies to be offered in a best value, total cost of ownership environment.

The primary challenge in structuring the request for proposal (RFP) for a parallel file system is how to accurately define the system characteristics that a successful PFS must present. What system(s) will be served by the file system? What system(s) will be used as storage targets for file system performance? What is the individual machine-specific I/O workload? What is the mixed workload? These questions will define the required system architecture and features, including network technology and topology, and lower and upper system performances. Combining this information with capacity requirements and budget constraints leads to a target system size and system architecture.

For the Spider II deployment, the main OLCF compute platform Titan was used as the reference target for file system performance. Interconnectivity for other compute platforms was required through Scalable I/O Network (SION) [27], the existing OLCF InfiniBand storage area network (SAN).

Given the HPC storage market during the Spider II procurement, OLCF considered two RFP response models for Spider II: the "block storage" model and the "appliance" model. In the block storage model, OLCF would procure block storage devices, file system servers (i.e. Lustre), and networking equipment separately and integrate them internally. In the appliance model, OLCF would procure an integrated solution using vendor-supplied storage, server, and networking equipment, along with the Lustre software and services in a single package. OLCF invited vendors to supply proposals implementing one or both models in order to assess cost, facility and system administrator impact, and risk.

In crafting the technical requirements, several considerations were identified:

- **Hardware landscape:** During the period between the Spider I and Spider II deployments, OLCF conducted significant technology evaluation efforts in partnership with storage vendors. This equipped OLCF with information concerning performance targets, desired features, product quality, and vendor roadmaps. This activity was extremely beneficial to crafting an RFP that was realistic both technically and financially given the knowledge of vendor product roadmaps, features, and limitations.
- **Lustre roadmap:** OLCF is heavily involved in Lustre development and used Lustre's roadmap projections to determine feature availability in the project timeline. Requirements surrounding Lustre were incorporated using this information.
- **Business considerations:** Total project budget, budget uncertainty and variability, and market conditions can impact the procurement schedule and vendor response. The OLCF designed options in the RFP to increase budget flexibility. Market conditions can have unintended consequences. For example, OLCF was forced to delay the release of the Spider II RFP due to the flooding in Thailand in 2011 that significantly impacted the disk availability and cost [31]. Given that over 20,000 disks were required to meet the expected capacity and performance targets, such factors must be accounted for.
- **Performance:** OLCF had specific requirements and limitations on total system performance based on its system architecture, network architecture, and program-specific requirements. Based on these parameters, OLCF set specific performance targets in the RFP. As stated, Titan has 600 TB of main memory. One key design principle was to checkpoint 75% of Titan's memory in 6 minutes. This drove the requirement for 1 TB/s as the peak sequential I/O bandwidth at the file system level. Our earlier tests showed that a single SATA or near line SAS hard disk drive can achieve 20-25% of its peak performance under random I/O workloads (with 1 MB I/O block sizes). This drove the requirement for random I/O workloads of 240 GB/s at the file system level.
- **Scale:** To accommodate the anticipated budget range, offered responses must demonstrate scalable units at specific performance and price points. This structure provides the flexibility to grow the PFS in the future as needed.

To accommodate issues of performance and scale, the procurement focused on the *Scalable System Unit* (SSU), a storage building block composed of a vendor-defined set of storage devices suitable for integration as an independent storage system. The SOW defined the SSU as the unit of configuration, pricing, benchmarking, and integration. Flexibility in SSU configuration provided for price and performance to be specifically tuned for the full solution.

Because the RFP allowed for separate procurement of

storage hardware, servers, and the networking equipment, specific performance requirements were engineered to guarantee that the installed system could meet overall performance goals. This was necessary because multiple vendors could be responsible for overall system performance.

Lesson Learned *3: Actively evaluate emerging file system and storage technologies. Consider vendor hardware and software roadmaps. Consider acquisition/budget strategy as applicable to your organization. Identify technical, financial, and supply chain risks. Align RFP release with favorable product development cycles.*

B. Benchmark Suite

To define the performance targets in the SOW, OLCF developed and released a benchmark suite based on its experience evaluating storage systems and earlier workload characterization efforts [14]. The benchmark suite tests several key low-level performance metrics indicative of overall system performance. The benchmark tool is synthetic, performing a parameter space exploration over several variables, including I/O request size, queue depth, read to write ratio, I/O duration, and I/O mode (i.e. sequential or random). It includes block-level and file system-level benchmark components. The block-level performance represents the raw performance of the storage systems. The file-system performance also accounts for the software overhead on top of the block level performance. By comparing these two benchmark results, we can measure the file system overhead. Specific parameters mimic real I/O workload patterns.

The block-level benchmark tool, *fair-lio* [6], was developed by OLCF and uses the Linux AIO library (*libaio*). It can generate multiple in-flight I/O requests on disks at specific locations, bypassing the file system cache. The file system benchmark tool is based on *obdfilter-survey* [25], a widely-used Lustre benchmark tool, benchmarking the *obdfilter* layer in the Lustre I/O stack to measure object read, write, and re-write performance. Vendors were provided both the benchmarks and instructions for executing them [23]. Results provided significant insight into the offered hardware for selection purposes.

Lesson Learned *4: Identify the I/O workloads expected for the PFS. Include sequential and random I/O characterization that mimics real workload patterns. Use benchmarks that can comprehensively describe the performance of this mixed workload.*

C. Evaluation

Offerors were encouraged to propose multiple configuration options to maximize flexibility in the design of the full system. This significantly increased the complexity in evaluating the responses, but was widely viewed as worth the additional effort.

The OLCF evaluated proposal responses based on their technical elements (including performance, capacity, reliability, power, footprint, maintenance, and service), delivery schedule, past performance, corporate capability, and total cost of ownership for the lifetime of the offered solution. During the evaluation process, each option provided by each vendor was considered. Ultimately, OLCF chose to purchase a block storage model. This decision resulted in significant design flexibility and cost savings while delivering a system that provided adequate delivered performance for users. However, this decision placed the integration and file system performance risk on the OLCF. The OLCF accepted this risk since the team possesses substantial experience deploying, managing, operating, and optimizing large scale file systems. The resulting contract for the Spider II file system included 32 PB capacity and more than 1 TB/s performance in 36 SSUs.

Lesson Learned *5: The evaluation criteria must structure the evaluation of all SOW requirements in a weighted manner such that every element of the vendor proposal is correctly considered in the context of the entire solution. Technical elements, performance, schedule, and cost each play an integrated role in choosing the correct solution. The decision to pursue different technologies must adequately address the inherent risk of each solution meeting the overall center objectives.*

IV. Integration

A center-wide PFS requires significant integration effort with sustained reliance on operational infrastructure and expertise. Figure 1 shows the Spider II architecture and its integration into the OLCF infrastructure. In this section we discuss the key lessons learned from our integration efforts.

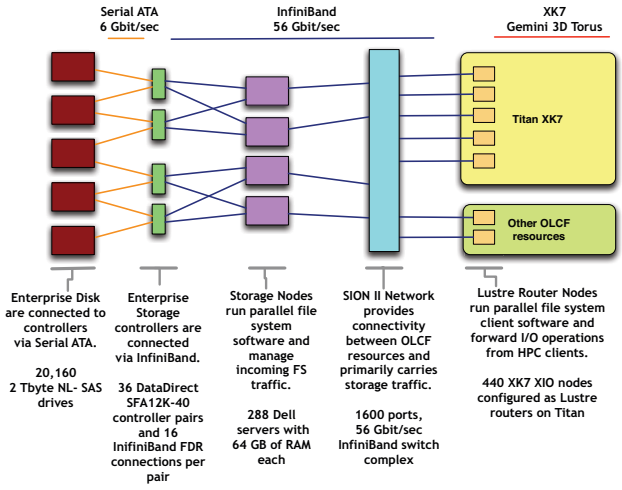

Fig. 1. Integration of Spider PFS and OLCF infrastructure.

A. Infrastructure integration

Many HPC centers deploy standard services for managing infrastructure; sufficient attention must be paid to the integration of the storage resources into the operational infrastructure. In the last decade OLCF has worked to deeply integrate their storage resources into the operational infrastructure of the center and has worked with the vendor community to push new features (e.g. parity de-clustering for faster disk rebuilds

and improved reliability characteristics) into their products that support these functions.

In 2008, the OLCF initially built out the infrastructure required for the Spider I file system as a separate instance to provide greater control and isolation from the rest of the operational infrastructure. The replication of services and efforts was resource intensive, managed separately, and failed to leverage the expertise built within the teams that provided services for the rest of the OLCF.

Recognizing that the benefits to a separate instance were outweighed by the additional effort, OLCF revisited its approach in 2010, and integrated the services that provide centralized authentication and authorization, log aggregation, configuration management, and other services for the PFS. The new structure does not relax any control mechanisms or security requirements.

This approach has yielded simple, robust infrastructure and has allowed the automation of some routine tasks that previously required system administrator attention. Further automation efforts continue.

Lesson Learned *6: The benefits of a single, integrated instance for the infrastructure of the center-wide systems outweigh the tighter control mechanisms offered by a disparate or separate file system instance. Where practical, centralize infrastructure services among disparate systems, center-wide, to defray expenses, harden those services, reduce administrative burden, share best practices, reduce inconsistencies, while retaining coordinated security and control.*

Below is a description of key features that were integral to the Spider I and Spider II deployments.

Cluster Management and Deployment

The OLCF has deployed cluster resources (both file system and compute) using the open-source Generic Diskless Installer (GeDI) [19] since 2007. This mechanism allows the nodes to boot over the control network, `tftp` an initial `initrd`, and then mount the root file system in a read-only fashion. Using diskless servers for the Lustre Object Storage Servers (OSS) and Metadata Servers (MDS) reduces the overall cost of the storage system because these nodes do not require RAID controllers, disk backplanes, cabling, disk carriers, or the physical hard drives. This materially reduces the acquisition and maintenance costs.

Custom scripts have been developed to create host-specific configuration files for boot services and the Lustre file system for the boot process so that the services can be started appropriately by the Linux `init` process. OLCF added a new feature to GeDI to support Spider II deployment where configuration files are built as the node boots, but before the service that needs the configuration file is started by the OS. This feature mimics the Linux model of `rc.local.d` for scripts that need to be part of the runtime. Scripts in `/etc/gedi.d` are run in integer order to build configuration files for network configuration, the InfiniBand srp_daemon configuration, and the InfiniBand Subnet Manager. These scripts also create RAM disks for the node-specific files in `/etc`, `/var`, and `/opt`.

This robust and repeatable image build process allows for rapid changes to both the operating system and the Lustre software base. The process is integrated with the center's change and configuration management system, BCFG2 [22], so that the effects of specific changes are easily determined. The change management process is consistent with the site cyber security policies. OLCF modifications to BCFG2 support diskless clients allowing for fast convergence to a node's configuration. This structure can positively impact mean time to repair (MTTR).

Lesson Learned *7: Build PFS clusters using diskless nodes to increase reliability and reduce complexity and cost. Build repeatable, reliable processes that rely on configuration and change management.*

Monitoring

A center-wide file system resource must be continuously available to its clients. OLCF has developed mechanisms for providing better reporting about the health of the file system [29] through the OLCF's monitoring framework provided by Nagios [21]. Vendor tools frequently exhibit limited capabilities, high overheads, and inability to scale. To address these weaknesses, we developed light-weight bandwidth monitoring and performance quality assurance tools. Where possible, these tools are open-sourced to the community to support their efforts.

During the Spider I operational period, OLCF developed a utility called Lustre Health Checker that provided visibility into internal Lustre health events, giving system administrators a coherent collection of associated errors from a Lustre failure condition. Additional utilities were extended to coalesce physical hardware events on the Lustre servers to help identify and debug failure conditions. These two features allowed system administrators to discriminate between hardware events and Lustre software issues [12].

To monitor the InfiniBand adapter and network, custom checks were written around the standard OFED [32] tools for HCA errors and network errors. These alerts provide detection of problems between the server and the InfiniBand connected storage or problems in the network between the Lustre server and clients. Single cable failures can cause performance degradation in accessing the file system. OLCF has developed procedures for diagnosing a cable in-place and provided these procedures to the manufacturer.

DDN Tool [18] was developed to monitor the DDN S2A and SFA storage system RAID controllers. This tool polls each controller for various pieces of information (e.g. I/O request sizes, write and read bandwidths) at regular rates and stores this information in a MySQL database. Standardized queries and reports support the efforts of the system administrators.

Lesson Learned *8: A robust monitoring/alerting platform coupled with analysis tools reduces cluster and file system administration complexity for large parallel file systems. Leverage open-source and vendor supported tools and utilities to overcome inherent weaknesses of basic tools.*

B. Compute-side integration

The Lustre client and server software stacks must remain compatible with each other over the life of the file and storage system. Titan is a unique resource that supports testing at extreme scale. To benefit the OLCF and the community at large, the OLCF allocates the Titan and the Spider PFS for full scale tests of candidate Lustre releases. These tests identify edge cases and problems that would not manifest themselves otherwise.

Lesson Learned *9: Develop a center roadmap for features that are important; align it with vendor and developer roadmaps and rigorously test against production workloads. Leverage the benefit of external test resources that can reveal problems at scale.*

C. Capacity Planning, Provisioning, and Management

The OLCF examined two strategies for deploying a parallel file system: a single large capacity namespace and multiple smaller namespaces.

A single namespace has several benefits. It is easier for users, provides the most bandwidth to applications, and provides the most flexibility in managing the capacity allotted to each project. While providing a single namespace to users is a straightforward way to deploy a data-centric resource, it introduces key disadvantages that can make it impractical to large-scale centers, such as the OLCF. Lustre supports a single metadata server per namespace. This limitation cannot sustain the necessary rate of concurrent file system metadata operations for the OLCF user workloads. With a single namespace, any problem has the potential to propagate to all other OLCF resources. These technical challenges are significant enough to divide the capacity space into smaller file system namespaces. The authors acknowledge that the Lustre 2.4 version introduced the Distributed Namespace (DNE) feature. Currently, some legacy Lustre clients block implementation of this feature at OLCF. We recommend using both DNE and multiple namespaces, concurrently.

Splitting the namespace across the storage hardware will reduce the delivered bandwidth to any one specific application. However, our experience has shown that other factors will limit the users' observed performance before the reduction in hardware will. Users are not confined to a single namespace at the OLCF. They may request access to all namespaces and use them as they see fit. This strategy can provide improved performance for their application workloads.

Next, OLCF developed a model that classifies projects based on their capacity and bandwidth requirements. The projects were then distributed among the namespaces. This model allowed the OLCF to manage the capacity and bandwidth more evenly across the namespaces for Spider I (four namespaces) and Spider II (two namespaces) file systems.

The Spider file systems are scratch. To maintain these volumes, the OLCF employs an automatic purging mechanism. Files that are not created, modified, or accessed within a contiguous 14 day range are deleted by an automated process. This mechanism allows for automatic capacity trimming. The OLCF as well as many other HPC centers that use Lustre note a severe performance degradation after the resource is 70% or more full.

Lesson Learned *10: Create multiple namespaces and use the operational expertise to segregate intense workloads and manage overall capacity; use an automatic purging mechanism to control capacity. Ensure that the acquisition strategy provides sufficient total storage such that performance is maintained up to typical performance degradation points. This may require capacity targets 30% or more above aggregate user workload estimates.*

D. Product Extensions to Satisfy Operational Needs

Commercial product offerings may not fully meet the requirements of the HPC centers. OLCF identified functionality, reliability, and performance gaps in Lustre. To close these gaps, OLCF direct-funded developments efforts through multiple providers to produce features including asymmetric router notification, high-performance Lustre journaling, and imperative recovery, all benefiting the Lustre community at large. OLCF continues to contribute to Lustre development efforts through direct funding and community consortium participation (i.e., OpenSFS).

E. Human Errors

Despite precautions and prescriptive policy, human error may affect the operation of HPC centers in a significant and undesirable manner. An incident in 2010 demonstrates the need to anticipate mistakes, planning for their occurrence, but limiting their effect through careful design, effective policy.

A disk was replaced in a storage enclosure. In the Spider I file system design, 10 disks in a RAID 6 set were evenly distributed across five disk enclosures. During the disk rebuild, the connection between the storage controller to the disk enclosure was interrupted due to a hardware failure and failed over to the other storage controller as designed. This unit was returned to production while still in a rebuild state, which meets the design specification. However, eighteen hours later, the affected storage array was taken offline, while still in the rebuild mode, losing journal data for more than a million files managed by that controller pair. Recovery of the lost files took more than two weeks, with 95% successful recovery rate. This was a drawback in the design of Spider I; architected design used 5 disk enclosures per storage controller pair instead of 10. A design using 10 enclosures per storage controller pair would have tolerated this failure scenario.

Lesson Learned *11: Failures due to human error will occur. To the greatest extent possible, the design, process, procedures, and policy should be structured to minimize the impact of these errors. Recognize that individual failures of hardware or software can be compounded by human error to produce unanticipated error conditions with significant impact.*

V. End-to-end Performance Tuning

Large-scale supercomputing storage infrastructure is a complex distributed system that needs to be tuned at several levels to achieve good end-to-end I/O performance. Several factors should be considered while tuning the performance including the storage hardware subsystem, the file system, user workloads, the clients, and the routing infrastructure. Vendor advertised peak sequential I/O performance numbers will realistically not be achievable on systems under real-life workloads.

While there will be some differences between different deployments, the multiple layers of any large-scale storage system will generally comprise block devices, the interconnect between the block devices and the file system servers, file system servers and the file system itself, and the interconnect between the file system servers and file system clients. Some configurations will include I/O routers between the file system servers and the clients (Figure 1). Each layer must be separately benchmarked to ensure proper operation and optimal performance. Our testing methodology is bottom up. We start our evaluation from the block-level to establish a baseline performance and work our way up the stack. We establish an expected theoretical performance of a given layer and then compare that with the observed results. Each additional layer introduces new bottlenecks and re-defines the expected performance.

Our deployment-specific subsystems include the 36 block storage system units (SSUs), the Lustre file system, the scalable I/O network (SION), the Lustre router nodes, and the Lustre clients on the Titan compute nodes themselves. Other OLCF resources are connected to Spider over the same SION and each resource has its own set of Lustre routers. Each one of these layers is a large system with its own idiosyncrasies. For example, Spider's block storage subsystem comprises 20,160 2 TB near-line SAS disks that are organized into 2,016 object storage targets (OSTs). 288 storage nodes are configured as Lustre object storage servers (OSS). 440 Lustre I/O router nodes are integrated into the Titan interconnect fabric. Titan provides 18,688 clients, all performing I/O. Each of these layers must be tuned and optimized to derive optimal performance from the end-to-end I/O stack. The Spider storage system delivers a peak throughput of 1 TB/s by optimizing each of these subsystems and how they interact with each other.

Lesson Learned *12: Build the performance profile for each layer in the PFS, from the bottom up. Quantify and minimize the lost performance in traversing from one layer to the next along the I/O path. Systematically calibrate the performance baselines for each layer correctly and fix the bottlenecks at each point.*

A. Tuning the Block Storage Layer

The underlying block storage layer must be well-tuned and working as expected within a tight performance envelope to obtain consistent parallel I/O performance. Spider II disks are organized as RAID level 6 arrays (8 data and 2 parity disks). Each RAID group is then used as a Lustre Object Storage Target (OST). When a file is stripped over multiple OSTs to achieve higher performance, all underlying OSTs must perform as expected. During deployment, it was discovered that identifying and eliminating slow disks, even if they were functional with no errors, helps to significantly reduce the variance and tighten the performance envelope across the RAID groups. Block-level benchmarks were run to ensure that the slowest RAID group performance over a single SSU was within the 5% of the fastest and across the 2,016 RAID groups the performance varied no more than the 5% of the average. We conducted multiple rounds of these tests, eliminating the slowest performing disks at each round. We derived the 5% threshold empirically based on our experience with the Spider I deployment and operations. For all tests, the RAID groups were organized into performance bins and disk level statistics were gathered from the lowest performing set of groups. Disks accumulating higher I/O request service latencies were identified and replaced. Overall, during the deployment process we replaced around 1,500 of 20,160 fully functioning, but slower, disks. After deployment, the same process was repeated at the file system level and we eliminated approximately another 500 disks. This is a continuous effort for the lifetime of the PFS, but the initial effort helped in improving the per OST and aggregate I/O performance immensely. In production, the initial requirement for 5% variability among RAID groups was determined to be prohibitive and was contractually adjusted to 7.5%.

Lesson Learned *13: Variance caused by slow disks in a data-centric PFS will negatively impact overall performance. It is critical to identify and replace them. It is essential to repeat this process periodically for the lifetime of the PFS to ensure consistent sustainable performance. Seek disk performance gains through manufacturer software and firmware updates.*

B. Tuning the I/O Routing Layer

Large-scale distributed systems must take advantage of data locality to avoid the performance penalty of excessive data movement. Titan's network is based on the Cray Gemini interconnect that is configured as a 3D torus. I/O routers must be used to bridge Titan's Gemini interconnect to Spider's InfiniBand fabric. Spider II was designed with a decentralized InfiniBand fabric that consists of 36 leaf switches and multiple core switches. The I/O from Titan's compute clients are funneled through the I/O routers to the Spider storage system.

Considerable effort was directed towards calculating the router placement on Titan's 3D torus. This required significant understanding of the network topology. Several factors were taken into consideration such as network congestion (both on the Gemini and I/O InfiniBand networks), physical and logical dimensions of the Titan layout, number of clients and routers, and Gemini interconnect injection rates and link speeds [9]. Additional information about Gemini's performance characteristics and routing algorithm relevant to the placement decisions are available elsewhere [8]. The final I/O router placement in Titan's 3D torus network topology is shown in Figure 2. Each box in the figure represents a cabinet, where X and Y denotes the dimensions in Titan's 3D torus topology. Colored boxes

represent cabinets containing at least one I/O module. Similar colors correspond to identical "router groups." Router groups roughly correspond to SSU indices, with each group being assigned to four InfiniBand switches that provide connectivity between routers and the Lustre file servers. Each I/O module contains 4 I/O routers, with each router on the module connecting to a different InfiniBand switch.

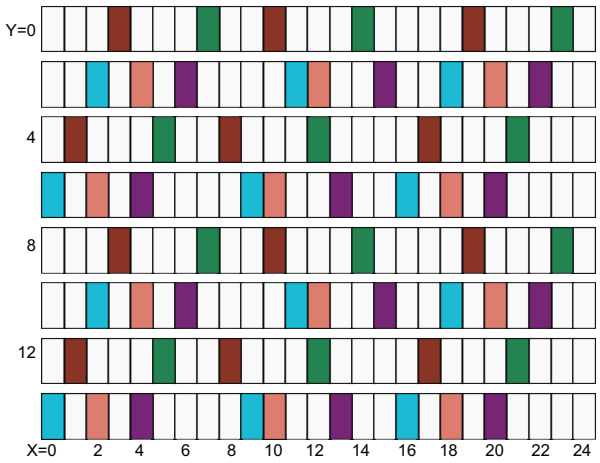

Fig. 2. Topological *XY* representation of Titan's Lustre routers.

For our mixed workloads, it was vital to physically distribute the routers in a way that ensures fair sharing for all of the compute clients. The average distance between any client and its closest I/O router should be optimized. OLCF devised a fine-grained routing (FGR) technique [5] to optimize the path that I/O must traverse to minimize congestion and latency. At the most basic level, FGR uses multiple Lustre LNET Network Interfaces (NIs) to expose physical or topological locality. Each router has an InfiniBand-side NI that corresponds to the leaf switch it is plugged into. Clients choose to use a topologically close router that uses the NI of the desired destination. Clients have a Gemini-side NI that corresponds to a topological "zone" in the torus. The Lustre servers will choose a router connected to the same InfiniBand leaf switch that is in the destination topological zone.

Lesson Learned *14: Network congestion will lead to sub-optimal I/O performance. Identifying hot spots and eliminating them is key to realizing better performance. Careful placements of I/O processes and routers and better routing algorithms, such as FGR, are necessary for mitigating congestion.*

C. Scaling Tests

Part of end-to-end performance tuning is scaling studies. While hero runs provide a good indicator of the peak throughput, scaling tests provide a more realistic picture of the I/O performance.

We used IOR, a common synthetic I/O benchmark tool [26]. Our custom benchmark suite discussed in Section III-B was primarily designed for small-scale block-level performance testing of a storage system, suitable for vendor responses to an acquisition activity. IOR, on the other hand, provides a readily available mechanism for testing the file system-level performance at-scale.

The parameter space for the scaling tests for a file system-level performance study is huge. Tests require a systematic approach of conducting controlled experiments. As an example, Figures 3 and 4 show a sample scaling study conducted on a single Spider II file system namespace. Since each Spider II namespace spans half of the available storage hardware resources, the expected the top-level performance is also halved.

As an experiment, we first sought the optimal transfer size per I/O process. To do this, we fixed the client size, the total amount of data per I/O process and the test duration and varied the I/O transfer size per I/O process. We used IOR in the file-per-process mode. Figure 3 shows the results. We identified that the best performance for writes can be obtained by using a 1 MB transfer size. We then fixed the transfer size to 1 MB and scaled the number of clients (I/O writer processes). Figure 4 presents the results. While these two tests are not indicative of the best possible performance that a single Spider II namespace can provide (as further optimizations are underway), they showcase how large-scale file systems can be systematically probed for performance. As can be seen from Figure 4, given its current configuration, a single namespace can scale almost linearly up to 6,000 clients and then provide relatively steady performance with respect to increasing number of clients. For both tests, we used a quiet system (Titan and Spider II were idle) and we let the scheduler place test clients on Titan compute nodes (i.e. random placement). This placement is optimized for nearest-neighbor communication, not for I/O. Each test ran for multiple iterations and each iteration ran for 30 seconds (stone wall option) to eliminate stragglers.

It is also worth mentioning that, as of this writing, the Spider II storage controllers were recently upgraded with faster CPU and memory. This upgrade improved the overall performance characteristics of the Spider II system. As a result of this upgrade, we observed 510 GB/s of aggregate sequential write performance out of a single Spider II file system namespace, versus 320 GB/s before the upgrade. IOR was used for this test in the file-per-process mode with 1 MB I/O transfer sizes. The peak performance was obtained using only 1,008 clients against 1,008 OSTs. The clients were optimally placed on Titan's 3D torus such that it minimized network contention for I/O. Further testing is scheduled to obtain scaling performance numbers from each namespace.

Lesson Learned *15: Running scaling I/O performance tests for large number of variables at extreme-levels can consume significant operational time, but yields very important scaling data. It is critical to carefully reduce the parameter space to shorten the test window and minimize the cost and program impact. In the end, the data obtained should be used to further configure and optimize the end-to-end I/O stack. Same scaling data should be used to guide the expectations for operational performance.*

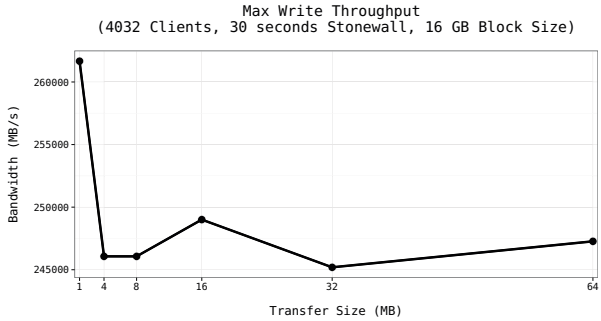

Fig. 3. Performance analysis as a function of I/O transfer size on a single Spider II namespace.

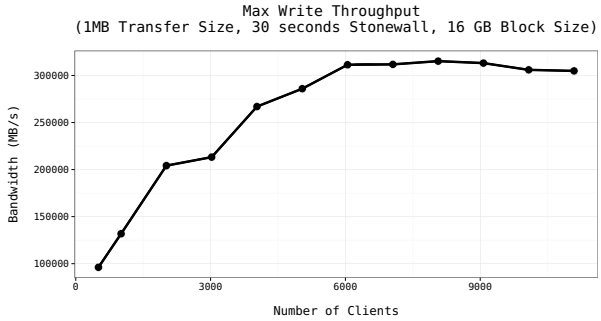

Fig. 4. Performance analysis as a function of number of clients on a single Spider II namespace.

D. Sustainable Lifetime Performance

Sustaining storage system performance through the lifetime of the system requires a *performance quality assurance (QA)* approach. Just as one would perform periodic QA of a data repository to ensure that the contents are valid, one needs to conduct performance QA to ensure that the storage system can continually deliver the desired performance metrics. During the initial deployment, the storage system is fresh, without any user data. This allows destructive tests that build and tear down file systems, while tuning and measuring performance. Performance testing will require writing and reading large volumes of data, which is not feasible on a live storage system. To this end, the Spider file systems were provisioned with a small part of each RAID volume reserved for long-term testing. While it only represents a small percentage of the total hardware capacity, it can be used to stress the entire system. This *"thin"* file system, which contains no user data, can be used to run destructive benchmarks even after Spider has been put into production. It also allows for performance comparisons between full file systems and those that are freshly formatted.

Lesson Learned *16: Plan and design for test resources for the lifetime of the PFS. Mechanisms such as a thin file system can accommodate the destructive nature of some of these tests. The capacity for these thin file systems are minimal, but should be accounted for in the acquisition process. Additional resources for testing and validating hardware and software upgrades before being deployed on production systems, is extremely beneficial over the life time of the PFS. A small-scale replica of the PFS building blocks can suffice for this testing and validation purposes.*

VI. Higher-level Services

As part of the ongoing operation and maintenance of the center-wide file systems at OLCF, a number of custom tools and utilities were written to assist both the users and the system administrators. These tools provide a mechanism for exposing the underlying capabilities and performance of the PFS to users. They also address scalability shortcomings of standard Linux file system utilities.

A. Balanced Data Placement

We have outlined the reasons for designing and operating a data-centric center-wide file system. While a data-centric file system has its merits, it has its own challenges. Due to the shared nature of file system, the underlying system is continuously under contention. Consequently, it becomes more difficult to expose the underlying raw capability directly to the scientific applications. Complex system architecture of the storage system and non-deterministic end-to-end I/O routing paths exacerbate this problem further, making it harder to deliver high performance at the client-side.

Therefore, to bridge the gap between the low-level file system capabilities and applications, we have designed an easy-to-use, portable runtime library [33]. Our placement library (`libPIO`) distributes the load on different storage components based on their utilization and reduces the load imbalance. In particular, it takes into account the load on clients, I/O routers, OSSes, and OSTs and encapsulates these low-level infrastructure details to provide I/O placement suggestions for user applications via a simple interface. Experimental results at-scale on Titan demonstrate that the I/O performance can be improved by more than 70% on a per-job basis using synthetic benchmarks.

We have integrated libPIO with a large-scale scientific application, S3D, which is widely run on OLCF resources. S3D is a large-scale parallel direct numerical solver (DNS) that performs the direct numerical simulation of turbulent combustion [11]. S3D is I/O intensive and periodically outputs the state of the simulation to the scratch file system. This output is later used for both checkpoint and analysis purposes. The S3D I/O pattern and performance have been analyzed in [35]. We note that only 30 lines of code were needed to be added/modified in the application for integration with our placement library. To evaluate the effectiveness of our approach, we conducted tests in a production (noisy) environment. We observed substantial gains in S3D I/O performance, up to 24% improvement in POSIX file I/O bandwidth. Also, the ease of implementation with minimal code changes encourages us to believe that libPIO can be widely adopted by scientific users.

Lesson Learned *17: The real performance of a data-centric PFS is what users observe. A file system with a high theoretical peak bandwidth may not deliver that performance to users under contention. It is a good practice to expose low-level infrastructure details to users in an easy-to-use programmable fashion to achieve higher performance for advanced users.*

B. Dynamic Resource Allocator

As stated in Section IV, scientific applications are statically distributed over two Spider II namespaces for load balancing and capacity planning. However, our static distribution does not fully take into consideration the dynamic nature of I/O workloads. To address this issue, we developed a tool called I/O Signature Identifier (IOSI) [16]. IOSI characterizes per-application I/O behavior from the server-side I/O throughput logs. We determined application I/O signatures by observing multiple runs and identifying the common I/O pattern across those runs. Note that most scientific applications have a bursty and periodic I/O pattern with a repetitive behavior across runs. Unlike client side tracing which provides detailed I/O trace data, our approach provides an estimate of observed I/O access patterns at no cost to the user and without taxing the storage subsystem. IOSI can be used to dynamically detect I/O patterns and aid users and administrators to allocate resources in an efficient manner.

Lesson Learned *18: Smart I/O-aware tools can be built for load balancing, resource allocation, and scheduling.*

C. Scalable Linux Tools

Standard Linux tools do not work well at scale [17]. A good example is the standard Unix `du` command. `du` imposes a heavy load on the Lustre MDS when run at this scale. Therefore we developed the LustreDU tool, which gathers disk usage metadata from the Lustre servers once per day. This tool is one of the keys to maintaining our policies for file system usage. We have seen direct performance degradation when the utilization of the filesystem is greater than 50%.

There are other Linux tools inefficient at scale, such as copy (cp), archive (tar), and query (find). These are single threaded commands, designed to run on a single file system client. Despite several efforts made to address this issue in an effective way [13], [1], [17], [15], there is not a complete solution yet, accepted and adopted by the HPC community. A collaborative effort between OLCF, Lawrence Livermore National Laboratory (LLNL), Los Alamos National Laboratory (LANL), and DataDirect Networks (DDN) has been established to research and develop efficient, scalable common tools [10]. As a product of this collaboration several tools have already been developed, such as parallel copy (dcp), parallel tar (dtar), and parallel find (dfind). These tools and future developments are expected to merge upstream to benefit both our users and the community at large.

Lesson Learned *19: Standard Linux tools do not work efficiently on large-scale systems. Develop or tune some of the basic Linux utilities, such that they can perform well at scale.*

VII. Discussion

Our intent in writing this paper is to share a compilation of comprehensive lessons learned from deploying file systems at the largest scale. The community of practitioners that are stressing the technical boundaries of performance and capacity remains small. To our knowledge, no other paper attempts to share this information in a comprehensive manner.

Many of our lessons are outcomes of deploying new solutions to unique technical challenges at scale. These challenges include designing cost effective, complex large-scale systems, methods for careful placement of I/O routers, developing new techniques for load balancing among storage sub-components, and developing a benchmark suite that supports the acquisition process. In many cases, these lessons were learned the hard way, at the expense of longer integration periods, greater engineering effort, and delayed deployment.

Revisiting Tradeoffs Between Traditional and Data-Centric PFS: Understanding the tradeoffs in designing, deploying, and operating data-centric center-wide parallel file systems is important. These file systems must balance competing I/O workloads from multiple computer systems. While resource-exclusive file systems can be more easily optimized for specific workloads, it is at the expense of more complicated data movement.

Another tradeoff metric for the PFS models is the cost of adding an additional compute resource (e.g., another data analysis cluster) to an HPC center. We typically express a capacity target for a parallel file system of no less than 30x the aggregate system memory of all connected systems. This capacity target was recently used in the DOE/NNSA CORAL acquisition [4]. For the current OLCF systems, total memory of all connected systems—Titan plus other data analysis and visualization clusters—is approximately 770 TB. With more than 30 PB (formatted), the Spider II capacity not only exceeds this target, but provides some margin for accommodating new systems with minimal cost. Conversely, resource-specific dedicated file systems will require us to incur not only the acquisition cost, but the integration and deployment effort as well. The original Spider I filesystem met a similar capacity target and supported all compute systems in the facility without the need for an upgrade.

Lustre Best Practices: In order to maximize the scalability, efficiency, and performance of user codes running on OLCF systems, we provide guidelines and best practices to our users [24]. These best practices are not exhaustive but they cover many common I/O patterns and use cases. They are generically applicable to any large-scale Lustre file system. Some of these focus on user behavior. Examples include warning the users not to edit or build user codes on the Lustre scratch file systems and point them to the NFS mounted home file area whenever possible and to use `ls -l` only where absolutely necessary on the Lustre scratch file system, since both use cases could impose a heavy load on the Lustre metadata server, hurting the whole user population. These kinds of problems can be avoided easily by changing the user behavior. Other best practices focus on application

I/O patterns. Examples include reading small, shared files from a single process, and placing small files or directories containing many small files on a single OST by setting the striping count to 1. In both cases, localization on a single OST improves the `stat` performance since every `stat` operation must communicate with every OST which contains file or directory data. Other examples include employing large and stripe-aligned I/O requests whenever possible to improve the I/O performance and scalability.

VIII. Conclusions

The OLCF began reevaluating its parallel file system (PFS) architectures in 2005 to remove data islands, promote data sharing, limit file system deployment costs, and improve the user experience. A data-centric, center-wide PFS design was found to best satisfy these criteria. While it was viewed as risky back then, through two large-scale deployments, the data-centric PFS model is the current preferred architecture for the OLCF. OLCF has gained extensive expertise in designing, acquiring, deploying, performance tuning, and operating these large-scale and complex PFS resources. This paper is an attempt to organize and elucidate the valuable lessons learned during this process, so that it can benefit the HPC community.

We started out with the motivating factors that drove us to design a center-wide, data-centric PFS. The ease of data sharing was well-received by our user community. Yet the benefits are not without cost: the sharing nature puts more demands on file and storage system availability, stability, and performance under mixed workloads. Peak sequential I/O bandwidth advertised by vendors are often unattainable under mixed workloads and a data-centric file system should be designed around the peak random I/O performance.

Our acquisition process balances the desirable features and user needs, pushing the technology limits within reason and budget. A comprehensive benchmark suite was developed to provide for both conforming references and consistency, employed by both bidding vendors and our technology evaluation team. Special attention has been paid to integrating such a complex array of storage hardware and software into the compute infrastructure seamlessly, with a robust operational monitoring infrastructure.

The real performance of a PFS is what users observe. End-to-end performance tuning is a critical issue. Performance of every layer in the I/O path should be carefully profiled, from bottom to top. It is advisable to systematically establish an expected performance characteristic of a given layer and then compare that with the observed experimental data. Every added layer introduces new bottlenecks and redefines the overall system performance.

Another key point is reducing the performance variance across similar components of a subsystem, such as disks. When exercised in parallel, variance caused by slower components hurts the overall performance. Identifying and replacing slower components is key to improving the aggregate performance. This is an ongoing process and should be repeated periodically for the lifetime of the PFS, to ensure a sustainable performance.

In summary, the lessons learned through nearly a decade of production at a leading HPC center are important for the community. Platform specific data resources may no longer remain the norm in HPC centers and our experiences can help the community design better PFS.

Acknowledgment

The authors would like to thank all past and present OLCF staff for their tremendous effort in making Spider systems a reality. The authors specifically acknowledge the efforts of Don Maxwell. They are also grateful to Dr. Suzanne Parete-Koon for her comments and valuable insights.

This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

References

[1] D. H. Ahn, M. J. Brim, B. R. de Supinski, T. Gamblin, G. L. Lee, M. P. Legendre, B. P. Miller, A. Moody, and M. Schulz, "Efficient and scalable retrieval techniques for global file properties," in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, ser. IPDPS '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 369–380. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2013.49

[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] A. S. Bland, J. C. Wells, O. E. Messer, O. R. Hernandez, and J. H. Rogers, "Titan: Early experience with the Cray XK6 at Oak Ridge National Laboratory," in *Proceedings of Cray User Group Conference (CUG 2012)*, May 2012.

[4] "Coral request for proposal b604142," https://asc.llnl.gov/CORAL/, accessed: 2014-07-29.

[5] D. Dillow, S. Oral, G. M. Shipman, Z. Zhang, and Y. Kim, "Enhancing I/O throughput via efficient routing and placement for large-scale parallel file systems," in *Proceedings of the 30th IEEE Int'l Performance Computing and Communications Conference (IPCCC 2011)*, November 2011.

[6] 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.

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

[8] M. Ezell, "Understanding the impact of interconnect failures on system operation," in *Proceedings of Cray User Group Conference (CUG 2013)*, May 2013.

[9] M. Ezell, D. A. Dillow, S. Oral, F. Wang, D. Tiwari, D. E. Maxwell, D. Leverman, and J. Hill, "I/O router placement and fine-grained routing on Titan to support Spider II," in *Proceedings of Cray User Group Conference (CUG 2014)*, May 2014.

[10] "File utilities for distributed systems," http://fileutils.io/, accessed: 2014-04-18.

[11] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen, "Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models," in *Journal of Physics: Conference Series*, vol. 16, no. 1. IOP Publishing, 2005, p. 65.

[12] J. J. Hill, D. B. Leverman, S. M. Koch, and D. A. Dillow, "Determining the health of Lustre filesystems at scale," in *Proceedings of Cray User Group Conference (CUG 2010)*, May 2010.

[13] G. M. S. Kenneth D. Matney, Sr, "Parallelism in system tools," in *Proceedings of the Cray User Goup conference (CUG 2011)*, 2011.

[14] 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.

[15] P. Z. Kolano and R. B. Ciotti, "High performance multi-node file copies and checksums for clustered file systems," in *Proceedings of the 24th international conference on Large installation system administration*. USENIX Association, 2010, pp. 1–8.

[16] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Automatic Identification of Applications I/O Signatures from Noisy Server-Side Traces," in *Proceedings of the USENIX Conference on File and Storage Technologies (FAST 2014)*, February 2014.

[17] K. Matney, S. Canon, , and S. Oral, "A first look at scalable i/o in linux commands," in *Proceedings of the 9th LCI International Conference on High-Performance Clustered Computing*, 2008.

[18] R. Miller, J. Hill, D. A. Dillow, R. Gunasekaran, G. M. Shipman, and D. Maxwell, "Monitoring tools for large scale systems," in *Proceedings of Cray User Group Conference (CUG 2010)*, May 2010.

[19] M. Minich, "GEneric Diskless Installer," http://sourceforge.net/projects/gedi-tools/, 2009.

[20] R. P. Mount, "The office of science data-management challenge," Stanford Linear Accelerator Center (SLAC), Tech. Rep., 2005.

[21] Nagios Enterprises, "Nagios," http://www.nagios.org.

[22] D. Narayan, R. Bradshaw, and J. Hagedorn, "System management methodologies with bcfg2," *Login*, vol. 3, pp. 11–18, 2006.

[23] Oak Ridge Leadership Computing Facility, "OLCF I/O evaluation benchmark suite," http://www.olcf.ornl.gov/wp-content/uploads/2010/03/olcf3-benchmark-suite.tar.gz.

[24] "OLCF Spider Best Practices," https://www.olcf.ornl.gov/kb_articles/spider-best-practices/, accessed: 2014-07-29.

[25] Oracle Inc., "Benchmarking lustre performance (lustre I/O kit)," http://wiki.lustre.org/manual/LustreManual20_HTML/BenchmarkingTests.html.

[26] 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.

[27] G. M. Shipman, D. A. Dillow, D. Fuller, R. Gunasekaran, J. Hill, Y. Kim, S. Oral, D. Reitz, J. Simmons, and F. Wang, "A Next-Generation Parallel File System Environment for the OLCF," in *Proceedings of Cray User Group Conference (CUG 2012)*, May 2012.

[28] G. M. Shipman, D. A. Dillow, S. Oral, and F. Wang, "The spider center wide file systems; from concept to reality," in *Proceedings of the Cray User Group Conference (CUG)*, 2009.

[29] G. M. Shipman, D. A. Dillow, S. Oral, F. Wang, D. Fuller, J. Hill, and Z. Zhang, "Lessons learned in deploying the worlds largest scale lustre file system," in *Proceedings of the Cray User Group Conference (CUG)*, May 2010.

[30] Sun Microsystems, "Luste Wiki," http://wiki.lustre.org, 2009.

[31] "Hard drive prices double after Thai floods," http://www.theinquirer.net/inquirer/news/2128270/hard-drive-prices-double-thai-floods, accessed: 2014-04-17.

[32] "The OpenFabrics Alliance," http://www.openfabrics.org/downloads, accessed: 2014-04-16.

[33] F. Wang, S. Oral, S. Gupta, D. Tiwari, and S. S. Vazhkudai, "Improving large-scale storage system performance via topology-aware and balanced data placement," Oak Ridge National Laboratory, National Center for Computational Sciences, Tech. Rep., 2014, technical Report ORNL/TM-2014/286.

[34] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang, "Understanding Lustre Filesystem Internals," Oak Ridge National Laboratory, National Center for Computational Sciences, Tech. Rep., 2009, technical Report ORNL/TM-2009/117.

[35] W. Yu, J. S. Vetter, and H. S. Oral, "Performance characterization and optimization of parallel i/o on the cray xt," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–11.