A More Realistic Way of Stressing the End-to-end I/O System

Verónica G. Vergara Larrea*, Sarp Oral*, Dustin B. Leverman*,
Hai Ah Nam†, Feiyi Wang* and James Simmons*
*National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, USA
Email: {vergaravg,oralhs,leverman,fwang2,simmonsja} at ornl.gov
†Computing and Computational Sciences Division
Los Alamos National Laboratory[1]
Los Alamos, NM, USA
Email: hnam at lanl.gov

***Abstract*—Synthetic I/O benchmarks and tests are often insufficient in realistically stressing a complex end-to-end I/O path. Evaluations built solely around these benchmarks can help establish a high-level understanding of the system and save resources and time. However, they fail to identify subtle bugs and error conditions that can occur only when running at large-scale. The Oak Ridge Leadership Computing Facility (OLCF) recently started an effort to assess the I/O path more realistically and improve the evaluation methodology used for major and minor file system software upgrades. To this end, an I/O test harness was built using a combination of real-world scientific applications and synthetic benchmarks. The experience with the harness and the testing methodology introduced are presented in this paper. The more systematic testing performed with the harness resulted in successful upgrades of Lustre on OLCF systems and a more stable computational and analysis environment.**

***Keywords*-Parallel I/O; Lustre; testing; supercomputers**

I. Introduction

Synthetic I/O benchmarks such as IOR and mdtest are often insufficient in realistically stressing large-scale and complex end-to-end I/O paths. Evaluations built around these benchmarks can help establish a high-level understanding of the system performance while saving resources and time. However, synthetic benchmarks are mostly run with predefined configurations for predefined test scenarios. Oftentimes, they fail to reveal subtle bugs, edge-case race conditions and errors that can occur only when running at large-scale. The entries in the Lustre bug tracking system [1] can testify to the limited effectiveness of synthetic benchmarks in identifying extreme error cases at very-large scales.

In early 2014, an attempt to upgrade the Lustre clients and servers at the Oak Ridge Leadership Computing Facility (OLCF) caused several unexpected problems due to a series of previously unidentified issues that surfaced on upgraded production systems. Although these newer versions of the Lustre software were tested on smaller scale testbed systems prior to deployment, production jobs exposed severe issues that affected the functionality and stability of the Lustre file system.

Learning from this experience, the OLCF started an effort to assess the end-to-end I/O path more realistically, aiming to improve the evaluation of all major, as well as minor file system software upgrades. In order to comprehensively test new Lustre software versions, the OLCF staff built an I/O test harness using a combination of real-world scientific applications and synthetic benchmarks. This paper describes the new testing methodology and process established and presents experiences gained from the utilization of the I/O test harness.

The I/O test harness pulls in scientific applications from the OLCF workload including combustion, fusion, and climate. These applications, as a mix, were identified as being representative of the type of workloads OLCF users run on Titan. Harness tests were chosen to exercise various middle-layer I/O libraries, methods, and I/O patterns commonly seen on OLCF systems. The test harness also includes common user operations that can generate a high load on the file system.

The new testing procedure incorporates multiple test cases of these I/O harness applications running concurrently to better simulate a production workload. As part of the new test methodology, tests are performed at different scales starting with testbed systems of a few nodes, moving on to medium-size systems, and finally testing at large-scale on Titan. In addition, at each stage, several configurations are used to more accurately replicate the workload executed on Titan by OLCF users.

The addition of a more realistic workload utilizing the I/O test harness has allowed the OLCF to perform systematic testing to identify and isolate issues, which has resulted in a successful major version upgrade of the Lustre software on OLCF systems. The methodology introduced also provides a more stable computational and analysis environment for

[1]LA-UR-15-22500

OLCF users. The experiences with the I/O test harness shared in this work could be valuable to other members of the Lustre community.

II. Background

The OLCF at Oak Ridge National Laboratory (ORNL) has a long track record of hosting some of the world's fastest supercomputers. It is now the home to Titan [2], the second fastest supercomputer in the world [3] Titan is a hybrid system and is the largest GPU supercomputer in the world [4]. Titan is a Cray XK7 system and has 18,688 compute nodes. Each node has a 16-core 2.2 GHz AMD Interlagos processor and an NVIDIA Kepler K20X GPU accelerator. Users of OLCF resources span a wide range of scientific domains [5], [6] utilizing a diverse programming models and I/O patterns [7], [8].

Titan has access to Spider 2, OLCF's 1 TB/s, 32 PB center-wide Lustre file system [9]. Lustre is a parallel file system and very popular in large-scale supercomputers. As of 2013, it was deployed in 70% of Top10 and 60% of Top100 systems in the Top500 list [10]. More details on Lustre can be found in [10], [11].

OLCF has extensive experience deploying and operating large-scale Lustre file systems. The first Lustre file system at OLCF was deployed in 2005 to serve the Jaguar XT3 supercomputer. Since then, OLCF has deployed two large scale Lustre file systems (i.e. Spider 1 and 2) and a multitude of smaller scale Lustre file systems. OLCF contributes to the Lustre community by providing large-scale test resources and significant staff resources for Lustre code development and bug fixes.

A. Testing at the extreme scale

A file system should be designed according to production environment needs. Scale, number and type of computational platforms to support directly influence design complexity. The number of concurrent applications running on these systems and variety of applications also increase the demands of a production file system. Therefore, large-scale file system design, acquisition, and deployment phases all include rigorous technology evaluation and testing efforts. Test methodologies vary across supercomputing facilities and institutions. Upgrading production file system also requires similar levels of testing and effort at the large-scale.

Traditional methods of testing large-scale production file systems mainly involve executing a set of predefined I/O scenarios and observing the response of the file system. The tests, whether conducted pre-production at the acquisition/deployment phase or in-production at the file system upgrade phase, are contending with the facility and system resources that were allocated for production cycles to users. Therefore, such tests are usually condensed in time, scope, and resources.

Furthermore, in the extreme scale (i.e. top tier of the Top500 systems), file system vendors are lacking the capabilities to adequately test features and fixes before public releases. Testing at these scales is currently assumed as the customer responsibility.

Popular tools for file system testing, especially for Lustre, are mainly centered around synthetic benchmarks, such as IOR and mdtest. While it is possible to mimic a vast majority of I/O workloads in isolation with such tools, they are not successful in identifying edge-cases or race-conditions that are often observed under real production workloads. A quick look at the Lustre bug tickets will confirm this.

It is because of these reasons, OLCF decided to augment existing test procedures and tools with more realistic ones. Similar efforts are now being undertaken by several other large-scale supercomputing facilities all around the world. Our idea for improvement was simple: to assess the end-to-end I/O path realistically, we need to stress our systems using better tools in more realistic ways. This does not exclude the use of synthetic benchmarks, rather it augments them. Sanity checks and small- or large-scale performance tests are still being conducted at OLCF using a combination of community or in-house developed synthetic benchmarks. However, we believe running various mixes of real-world scientific applications in a multitude of configurations and scales provides a clearer picture of the end-to-end system characteristics and responses.

III. End-to-end I/O Path

A. Spider 2

Spider 2 is the primary Lustre file system resource for the OLCF. It is broken up into two file systems. Figure 1 shows OLCF's end-to-end I/O path and the Spider 2 architecture. Each of the Spider 2 file systems are roughly 20 times larger than our largest test resource. This presents issues when testing the Lustre server or client side software because large-scale systems can behave differently under different workloads than small-scale systems. For example, a code that writes a single-shared file on a test system may only be able to write using 50 stripes, but on one of the Spider 2 file systems it can write using 1008 stripes. The number metadata operations involved in lock coordination for this file is much higher on the larger system and can expose bugs in the Lustre software that are not otherwise exposed on a test system.

To help address the scale issue on the test systems, each stripe on the Spider 2 file systems was divided into two pieces at the deployment phase: a large partition that will be used in production, and an small partition that will be used in testing. This allowed us to create another file system on the smaller partition that can be used to run the I/O harness and other tests at large-scale during a test without affecting the production partition or user data. The small test file systems

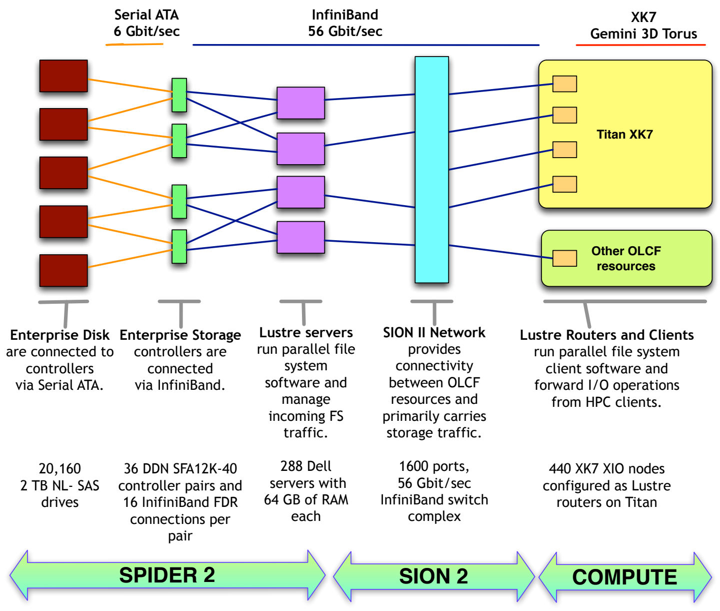
Serial ATA
6 Gbit/sec
InfiniBand
56 Gbit/sec
XK7
Gemini 3D Torus
Titan XK7
Other OLCF resources
Enterprise Disk are connected to controllers via Serial ATA.
Enterprise Storage controllers are connected via InfiniBand.
Lustre servers run parallel file system software and manage incoming FS traffic.
SION II Network provides connectivity between OLCF resources and primarily carries storage traffic.
Lustre Routers and Clients run parallel file system client software and forward I/O operations from HPC clients.
20,160 2 TB NL- SAS drives
36 DDN SFA12K-40 controller pairs and 16 InifiniBand FDR connections per pair
288 Dell servers with 64 GB of RAM each
1600 ports, 56 Gbit/sec InfiniBand switch complex
440 XK7 XIO nodes configured as Lustre routers on Titan
SPIDER 2
SION 2
COMPUTE

Figure 1. OLCF Spider 2 end-to-end I/O path.

have been critical in being able to stress the Lustre client and server codes using the I/O test harness at scale.

Complexity is an issue that comes with working on at-scale systems. Performance degradations can occur for a number of reasons, such as networking problems, software bugs, slow disks, and contention with other users that are running on the same systems.

B. *Lustre server and client software*

Testing in a complex environment introduces an additional challenge in that the vendors that support our software stack do not have test resources at the scale we operate. This results in the OLCF having to take system outages to test code changes.

In general, the Lustre server code (the code that handles reading and writing the data to/from a disk) does not change often on our file systems. The client code (the code that runs on the compute node and orchestrates reading and writing data to the Lustre servers) is where we encounter most bugs. However, rigorous testing for both server and client software updates are necessary to ensure new problems from software incompatibilities are not introduced leading to performance regressions.

IV. I/O HARNESS

User applications and workloads at the OLCF represent a diverse scientific community [5] [6] and utilize a broad range of programming models and I/O patterns [7] [8]. The OLCF developed an I/O harness to better evaluate the functionality, stability, and performance of the Spider 2 file systems when stressed by the typical OLCF production workload. To balance between testing-window time constraints and capturing real application behavior, the I/O harness test suite includes micro-benchmark tests, application I/O kernels, and real-world applications. Tests are run simultaneously to create contention, as seen during production, to identify adverse impacts from interactions between applications using shared system resources, such as the file system and network.

A. *I/O Harness Infrastructure*

In addition to the application workload, the I/O harness attempts to replicate typical user behavior exerted on the system. To simulate this, the I/O harness includes file system operations users frequently perform (e.g. ls, find, grep, etc.), as well as compiling complicated applications directly on the Lustre file system. Although OLCF users have 10 GB of home directory space on the NFS storage to store source files and build their applications, multiphysics application source and build files can often exceed this capacity, forcing users to build their applications directly on the Lustre file system. The entire production user workflow, including building, running, and analyzing data, is captured in the I/O harness tests and automated through a harness infrastructure written in Python. The harness infrastructure was created at the OLCF for use during acceptance testing to build applications, submit jobs to the queue, analyze correctness and performance requirements at the end of the job, and archive results. This workflow is common at the OLCF and stresses the file system beyond a single application run.

B. *I/O Harness Applications*

To mimic the real-user workload and behavior, we created the application-centric I/O test suite to exercise the file system on several aspects including: scale, diverse I/O patterns (e.g. single shared file, file per process, size of files, file striping, etc.), I/O library usage (e.g. HDF5, NetCDF, MPI-IO), and application workflow interaction with the file system. The current tests were chosen from the daily OLCF workload, which are representative of the I/O behavior of typical user applications. The current I/O harness test suite includes the following applications:

S3D*: Combustion modeling in flames:* S3D [12] is a combustion modeling application providing high-fidelity simulations using direct numerical simulation (DNS) methods in combustion regimes where turbulence-chemistry interactions are needed to describe the system. S3D uses explicit nearest-neighbor local communication with MPI and is primarily written in FORTRAN. Both CPU-only and GPU-ready via OpenACC versions of S3D are used for OLCF systems evaluation. Table I shows the S3D test sizes available in the I/O harness. S3D tests scaled to roughly 80% of Titan system resources.

I/O Test Attributes:

- Fortran I/O, 1 process/core (16 cores/node on Titan)
- File per process writes/reads for checkpoint/restart files
- Each checkpoint file is small (3.3MB)
- Each checkpoint creates a new directory with a Lustre stripe count of 1 (each file only uses 1 OST)
- 3 checkpoints during the test

Table I
S3D I/O TEST SIZES

No. processes (writers)	No. Titan nodes	Total Memory\ checkpoint
600	38	2.025 GB
2400	150	7.92 GB
6,000	375	20.25 GB
30,000	1,875	101.25 GB
96,000	6,000	324 GB
150,000	9,375	495 GB
210,000	13,125	693 GB
240,000	15,000	810 GB

GTC*: Particle-in-cell simulations of plasma microturbulence in fusion:* The Gyrokinetic Toroidal Code (GTC) [13] uses the gyrokinetic particle-in-cell method in full toroidal geometry to study magnetically confined plasmas, with specific application to modeling turbulent transport in the burning plasma fusion device at the International Tokamak Experimental Reactor (ITER). GTC is written in FORTRAN90 and C/C++ with nearest-neighbor communication and parallelism from MPI and OpenMP. There are multiple versions of GTC using a variety of I/O and math libraries depending on platform and research group. The CPU-only version of GTC is used for OLCF system evaluation. This I/O pattern and compute node utilization is common for many scientific applications at the OLCF. Table II shows GTC test sizes available in the I/O harness. GTC tests scale to 82% of Titan resources.

I/O Test Attributes:

- Fortran I/O, MPI/OpenMP, 2 processes/node, 8 threads/process on Titan
- File per process writes a checkpoint/restart file to a single directory
- Each checkpoint file is mid-size (274 MB)
- The checkpoint directory uses the default Lustre stripe count of 4
- 1 checkpoint during the test

Table II
GTC I/O TEST SIZES

No. processes (writers)	No. Titan nodes	Total Memory\ checkpoint
64	32	17.5 GB
128	64	35.1 GB
256	128	70.1 GB
512	256	140.3 GB
960	480	280.6 GB
1,920	960	561.2 GB
3,840	1,920	1,122.3 GB
7,680	3,840	2,244.6 GB
15360	7,680	4,489.2 GB
30,720	15,360	8,417.3 GB

FLASH 4.2*: I/O Kernel with a single restart write and read :* FLASH [14] is a publicly available parallel multiphysics simulation code used to simulate many astrophysical environments. FLASH exercises the HDF5 library and performs a large number checkpoints during the course of the entire simulation. A single checkpoint operation, including the reading and writing of a checkpoint file, was used in the test suite to track I/O performance variability in read and write operations across multiple runs during a test period. This test also highlighted and isolated differences in read and write performance across file system partitions with varying utilization and fragmentation.

I/O Test Attributes:

- HDF5 library
- Read in and write out of 1 average checkpoint file (54 GB), default striping of 4
- 625 nodes, 5000 processes
- Runtime 5 minutes

MPI-IO Kernel: *Large single shared file with wide stripes:* Some I/O patterns, such as many writers to a large single shared file, were identified to be potential stressors of the system resources based on Titan's performance degradation during the 2014 Lustre upgrade attempt. Pin-pointing a specific application or I/O pattern to be the source of system stress is a complicated process in a shared environment. Often the offending application can cause other applications to struggle and generate system errors. To verify that the use of collective MPI-IO was the source of the system degradation, the pattern was captured in the MPI-IO kernel and verified through our testing procedures. This test only manifested system errors when using 450 clients or more and increased in system disruptions, such as hangs or error messages in system logs, with a greater number of clients. Table III shows the variations to the number of clients and file striping in the MPI-IO test suite.

I/O Test Attributes:

- Collective MPI-IO; Large file (512 GB)
- Tests varied striping and number of clients
- Each test performed 3 successive reads and then 3 successive writes to new files in a single directory
- The second and third reads show substantial performance improvements from caching

Table III
MPI-IO Test Sizes

No. processes (writers)	Stripe Size
450	32, 128
512	32
1024	4, 32, 128, 360, 720

***CESM 1.2**: Community Earth System Model:* CESM [15] is a fully-coupled, community, global climate model used extensively by users at the OLCF and across multiple HPC centers. Due to the high frequency of I/O operations performed in a CESM run, it is anecdotally the first application to manifest issues in the file system. Build times of CESM can range from 40 minutes to 1 hour, depending on the file system. During the CESM application run, there are a large number of frequent writes to capture the status and history of the simulation for each coupled model. These history files are small in size, but accessed frequently. At the end of the CESM run, their internal scripts compress and archive the results and all supporting files used in the run. This frequent interaction with the file system makes CESM highly dependent on responsiveness of the command line and overall file system performance.

I/O Test Attributes:

- pnetCDF library for checkpointing
- 3,567 Titan nodes, 57,072 cores
- Capture performance of build (40 minutes - 1 hours) and run of test (20 minutes)

V. Testing Methodology

In addition to developing the I/O test harness, the OLCF implemented a test procedure that is now regularly used before a Lustre software upgrade is deployed to user-facing systems. The test procedure is the product of a collaborative effort between several different groups within the OLCF including computational scientists, system engineers, system administrators, and user assistance specialists.

A. OLCF File Systems Testing Team

The OLCF file systems testing team was formed to ensure that both the tests chosen and the methodology used include a diverse set of perspectives, which in turn, allows us to perform a more comprehensive evaluation. The expertise from each group is needed in order to fully exercise the new version of the Lustre software.

System administrators are responsible for verifying the hardware, installing and validating the software, preparing testbed systems, and troubleshooting issues encountered during any test stage. We also leverage the storage system expertise from system engineers to validate the new version of the Lustre software, and to identify, characterize, and fix any bugs that are discovered while running tests. Computational scientists select, prepare, and execute application tests that are representative from current workloads run at the OLCF. User assistance specialists execute tests from the I/O test harness, and also identify, troubleshoot, and replicate user reported issues that could help detect edge-cases in future tests. Of course, given the collaborative nature of this effort, team members from different groups work together on most of these tasks. In particular, during a test, all-hands are on deck to monitor the file system at the server and client levels, as well as monitor any observable change or degradation on the file system or on application performance while tests are running. Having members from each of the different groups available during a test has proven to be extremely effective when identifying and troubleshooting issues on-the-fly.

B. Test Planning and Procedure

The OLCF file system test team established a weekly meeting to discuss future Lustre upgrades, performance issues reported by users, and any upcoming systems and features. This meeting allows us to start planning well in advance of a Lustre software upgrade. When a new version of Lustre, major or minor, is under consideration a test timeline is created to keep track of all test stages. The first test phase usually requires packaging the new software, building a new Lustre image for the testbed system, and deploying it. Phase 1 usually takes 7 days and is executed at small scale with dedicated test resources. These small scale systems include a vast majority of storage, networking, and computing technologies. This first step allows us to evaluate features and bug fixes more quickly and, since they are dedicated resources, do not requires us to compete with production priorities. Most obvious performance regression problems are identified and corrected at this first phase of testing.

After successful completion of the first phase of testing, we then move to mid-scale testing. This next phase usually involves testing on medium user-facing compute resources at the OLCF which adds an extra layer of coordination to the process. To minimize impact to users, tests are run only during the regularly scheduled maintenance windows and are announced with at least one week notice. Planning, coordination with all OLCF stakeholders, and resource allocation at this phase is moderate, but still significant compared to the first phase of testing.

If both test phases are successful, a large-scale system test is scheduled on Titan and is usually coupled with regular hardware maintenance schedules. Planning, coordination, and resource allocation at this phase is key and it requires successful completion of the first two testing phases. Any failure, restarts the testing effort from the previous phase.

*1) **Phase 1**: Small-scale testing:* Our testing procedure starts at small scale, using one node jobs on a dedicated XK7 test resource (Arthur) and running two microbenchmarks: simul and IOR. If the tests are successful (i.e. tests completed without errors and showed no major performance regressions), the application I/O harness is used. To run real-world applications we use a larger XK7 testbed system (Chester) that has the same configuration per node as Titan, and a test development file system (TDS) capable of stripe counts of up to 56. Because Chester is fairly small, at this stage we only run the first two tests from GTC, and only the first test from S3D. In addition, we execute only the build step from CESM, and run MPI-IO using a small 128MB input file.

*2) **Phase 2**: Medium-scale testing:* After we have verified the Lustre software is stable on testbed systems, we move to the medium-scale testing phase which is conducted on two OLCF compute resources: Rhea, a Red Hat Enterprise Linux cluster with 512 nodes, and Eos, a Cray XC30 supercomputer with 736 nodes. The larger number of nodes allows us to execute MPI-IO tests with up to 450 nodes, in addition to repeating the tests from Phase 1.

*3) **Phase 3**: Large-scale testing:* If all tests in the previous two phases are successful, a test on Titan is scheduled at the next planned maintenance window. The testing window begins after preventive hardware maintenance is completed and both Titan and Spider 2 are placed into the configuration being tested. When a reboot of Titan is required, this process can take several hours. As soon as system administrators verify the configuration, the large-scale evaluation of the Lustre software can begin.

C. A Test on Titan

First, an initial stability check is performed using synthetic benchmarks. For mid- to large-scale synthetic workloads, we aim to stress the file system as much as we can. This set of tests is designed to accomplish three objectives: (1) sanity check at scale; (2) basic scaling test; (3) hero runs. The size of the parameter space is potentially large, and we try to explore it judicially to make efficient use of resources. The following example makes use of IOR options to illustrate some of our choices:

- We run both single shared file as well as file-per-process runs for both read and write.
- We iterate from as few as 1 node on Titan all the way to 4,096 nodes to observe scalability characteristics for both read and write. The single client performance can be revealing at times.
- We usually set transfer size at 1 MB as it is stripe aligned with our backend RAID storage.
- We always make sure write sync happens.
- The block size is chosen to balance the need for mitigating cache effects and overall runtime with increasing number of clients.
- We conduct both POSIX and MPI-IO tests as these two are the prevalent choice of interfaces for user applications.
- To test wide striping, we usually drive two cases: one is to fix the number of clients, and scale from 4 stripes all the way up to the partition limit (1,008); the other case is to increase the number of clients and the stripe count at the same time and observe its scalability trends.

The permutations of above test cases can be huge. Depending on the window of time available for the test, we sometimes choose to break it up or cherry pick cases that we have identified to be more problematic than the others. It is worth noting that these tests serve as a basic sanity check as well as scalability check, and they are not designed to test edge cases. To test for the latter, we utilize a mixed application workload which is described next.

The hero runs are designed to extract the maximum performance out of the system under ideal circumstances. Under this assumption, we can carefully place client nodes evenly around I/O routers as well as considering other resource balance along the end-to-end I/O path. The exact setup and details are out of the scope of this paper.

The synthetic benchmark testing stage is followed by an application stability test that consists of a subset of test cases used to assess the stability of the Lustre software under real-world scenarios. This subset includes MPI-IO test cases with 450 and 1,024 clients and stripe counts of 32, 128, 360, and 720. In several occasions, this workload was sufficient to trigger Lustre errors that resulted in Lustre client evictions.

If no problems are detected during the MPI-IO runs, we launch the full-suite of applications described in Section IV. The goal at this stage is to simulate a realistic workload by launching jobs that use a diverse number of nodes and I/O patters, and that are representative of the workload on Titan when it is in production. Consecutive MPI-IO, S3D, and GTC jobs are submitted continuously to the batch system for the duration of the application testing window. Simultaneously, we launch the CESM test case which first builds the code and then submits a job. The idea here is to simulate an activity that users frequently perform on the Lustre file system. If successful, the CESM launch script submits a job to the queue that can be used to assess performance. In addition, the large FLASH test is launched to generate as much contention as possible. Throughout the period when jobs are running, system administrators and system engineers carefully monitor the different components of the end-to-end I/O path. If at any point Lustre errors are identified, the test is terminated so that Titan and Spider 2 can be returned to their production configuration and user jobs.

Since the harness generates a continuous workload, in the absence of errors and abnormal behavior, jobs are only terminated once the testing window ends. At this point, we gather performance data from all stages of the test and generate a report that is used to compare the performance of the Lustre software being tested against previously measured reference results. Usually the reference results are obtained from the Lustre software version currently in production.

In addition to exercising the Lustre software, the application I/O harness is used to check for correctness and to measure performance from a user's standpoint. The applications in the I/O harness were designed to include a built-in correctness script that ensures the results from each test are accurate, and that no data corruption occurred. Using reference values from tests run with the Lustre software version in production, we can quickly identify when a significant performance degradation has occurred.

To account for I/O operations that can exercise the file system in unexpected ways, we closely watch user reports after upgrades. Using the information from these reports we have augmented the I/O test harness to include metadata intensive operations: compiling large code bases, copying collections of small files, and performing sequential reads using single clients.

As part of our testing procedure, we also test the Lustre recovery feature by simulating failures that can occur in production. The failures are simulated by first rebooting an MDS, then an OSS, and finally both simultaneously. The feature is marked as verified only if the file system is able to recover quickly and cleanly. The functionality of the recovery feature is critical since it keeps users' data safe when a critical event occurs. Furthermore, during a test, we verify a set of files that were created using Titan's production configuration. This test is designed to test for file corruption that can occur when changing to a new version of the Lustre software. Once Titan is returned to its production configuration, the data verification test is repeated using files created during the test. The goal of this test is to identify scenarios that could cause data loss, in the event that the upgrade is performed and we need to rollback to a previous configuration.

VI. Experiences upgrading from Lustre 1.8 to 2.5

Before the move to the new Spider 2 file system, the OLCF was using the Intel release of Lustre version 1.8 (the final Lustre release before moving to 2.x). After the Spider 2 deployment, the smaller production clusters began the process of migrating to the Lustre 2.4 client. The evaluation of the new client in our test and development system (TDS) using the synthetic benchmark workloads, failed to expose many issues that we encountered once Titan underwent the upgrade process. Once the Cray version of the Lustre 2.4 clients was rolled into production on Titan on January 28, 2014, user reports of slowness started to flood into our ticketing system. Besides slow metadata performance, Lustre clients would regularly get evicted from Lustre servers causing even greater user perceived performance degradations. After a two week period, Titan was rolled back to the previously deployed Lustre 1.8 client. A detailed timeline describing the events that transpired during the upgrade to Lustre 2.4 can be found in Table IV.

Table IV
Timeline of Events

Date	Description
1/28/14	Upgraded Titan to the Lustre 2.4 client
1/30/14	Users started reporting slowness issues
2/10/14	Titan rebooted to revert back to the Lustre 1.8 client. Spider 2 was rebooted to pick up LNET draining issue patch and a debugging patch for the MDS performance issue and Titan rebooted to revert back to the Lustre 1.8 client
2/26/14	LU-4008 signed off by Intel and put in production on Spider 2
10/7/14	Upgraded Titan to the Lustre 2.5.2 client
3/10/15	Upgraded Titan to CLE5.2UP02 and got the Lustre 2.5.1+ Cray client

While the rollback of the Lustre client to the previous version did resolve many of the issues that OLCF users were experiencing, a subset of the problems remained. This was a clear indication that the remaining issues originated from the server side. With help from Lustre engineers at Intel, we were able to track down the bugs responsible for the remaining issues. One of these problems would manifest when a user created a single shared file striped across the entire file system (1,008 stripes), causing the servers to hold a global lock while allocating a larger chunk of memory. This problem was only exposed on a file system with greater than 672 stripes. If a user was striping across less than 672 stripes they would never encounter this issue since a different allocation method was used for smaller kernel memory allocations. It was also discovered that the layout locks internal to Lustre that handle memory allocation always allocated for the maximum possible stripe count, making the servers always encounter the global kernel lock. The details for this particular issue are covered in the Intel bug tracker system under ticket LU-4008.

After the Lustre 1.8 Cray client and Lustre 2.4 server configuration was stabilized, the focus was to resolve the Lustre 2.4 client issues encountered after the attempted upgrade in January of 2014. Using our small-scale systems, we were able to replicate and track down bugs causing client evictions and other single shared file performance issues.

Some examples of the problems detected using our synthetic benchmarks were tracked under Intel tickets LU-5294 and LU-4829. See Table V for some examples of bugs that were identified since the new testing procedure was put in place.

During the same time frame, Intel moved the maintenance branch from Lustre 2.4 to Lustre 2.5, which contained many of the fixes to problems we had been tracking in our Lustre 2.4 clients. Over the next few months, the new Lustre 2.5 clients began passing all tests using our synthetic benchmarks and application I/O harness. Larger clusters were being put into production at this time, which gave us new opportunity to run our tests at an intermediate scale before running on Titan. At this larger scale of testing, again, all synthetic benchmarks passed.

From our previous large scale deployments it was evident that we needed to expand the scope of our testing to cover cases that cannot be reproduced on our smaller test systems. One case which is particularly difficult to test at small-scale is when large stripe counts are utilized. Besides the vmalloc global locking issues discussed earlier, another bug we could not have easily duplicated at small-scale was filed under Intel ticket LU-4719. This bug made it possible for a user to kernel panic the MDS when using a very large stripe count for a single shared file. What makes this problem so pronounced is the ease with which a user can set the stripe count to the maximum value. A simple `lfs setstripe -c -1` will communicate to the file system that all OSTs in the file system must be used.

Our team also understood the limitations of our synthetic benchmark test suite, making the addition of the new application harness a useful tool on our small and medium sized systems. In our initial runs of the new application suite, we could easily reproduce client evictions which were later addressed under the Intel ticket LU-2728. These evictions could not be reproduced with the synthetic benchmark test suite alone. On May 20, 2014 the next test was performed on Titan using both the traditional method of testing and the new application test harness so we could simulate the mixed I/O workload typically observed in production. More client evictions and performance issues were exposed using the application harness at this scale that would otherwise have remained hidden. Using the data logs collected, the issues found would typically be addressed by the time the next test on Titan was scheduled.

Table V
LUSTRE BUGS IDENTIFIED

Date	BugID	Identified by	Bug Description
02/26/14	LU-4008	Application I/O harness	vmalloc contention on MDS
07/04/14	LU-5294	Synthetic Benchmarks	Cannot unlink or rm
05/08/14	LU-4578	Application I/O harness	Adaptive timeout bug causing MDS to reboot
05/08/14	LU-4584	Application I/O harness	Client evictions in lustre-2.4. Same issue as LU-2728 in lustre-2.5
03/20/14	LU-4719	Synthetic Benchmarks	Kernel panic with large stripe files
08/07/14	LU-4829	Synthetic Benchmarks	Crash on mount
10/16/14	LU-5724	Application I/O harness	Imperative Recovery Issues
10/14/14	LU-5803	Application I/O harness	Recovery Issues, server not able to keep up with requests

CONCLUSIONS

The Oak Ridge Leadership Computing Facility (OLCF) has recently changed its methods and tools for conducting large-scale file system tests. A file system should be tested for functionality, correctness, and performance at every step throughout its operational life. It was recently found that testing using synthetic benchmarks cannot adequately stress the end-to-end I/O path and our test cases and tools have been augmented with real-world scientific applications as a result. The OLCF's new test procedure includes testing in increasing scales with automated and comprehensive test patterns, and includes the execution of a mixed workload of applications and synthetic benchmarks. Our experience over the last year and a half has proven that our new method and tools can better detect edge-cases, race-conditions, and bugs at large-scales. With this new method we identified more bugs before releasing the file system after an upgrade, therefore minimizing service interruptions due to file system issues.

Our future work includes periodic investigation of the aggregate I/O workload on the system, allowing us to reevaluate our choices of applications in the I/O harness to ensure that the mix of applications and synthetic benchmarks as needed to fit observed workloads.

ACKNOWLEDGMENT

The authors would like to thank the entire OLCF file systems testing team that also includes Don Maxwell, Matt Ezell, John Lewis, and Phil Curtis. The authors would also like to thank Duane Rosenberg, Ramanan Sankaran, Wayne Joubert, Matt Norman, and Bronson Messer for providing applications for the test suite; and Arnold Tharrington for use of his harness scripts to automate the build, launch and analysis of tests. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility.

References

[1] Intel, "Lustre (lu) bug tracking and new feature development," https://jira.hpdd.intel.com/browse/LU, 2015.

[2] 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," *Cray User Group*, 2012.

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

[4] J. Wells, "Flash i/o benchmark routine – parallel hdf 5," https://www.olcf.ornl.gov/2014/05/06/everybodys-talking-about-titan/, 2014.

[5] W. Joubert, D. Kothe, and H. A. Nam, "Preparing for exascale: Ornl leadership computing facility application requirements and strategy," *ORNL Technical Report ORNL/TM–2009/308. December*, 2009.

[6] V. Anantharaj, F. Foertter, W. Joubert, and J. Wells, "Approaching exascale: application requirements for olcf leadership computing," Technical report, Tech. Rep., 2013.

[7] Y. Kim, R. Gunasekaran, G. M. Shipman, D. A. Dillow, Z. Zhang, and B. W. Settlemyer, "Workload characterization of a leadership class storage cluster," in *Petascale Data Storage Workshop (PDSW), 2010 5th*. IEEE, 2010, pp. 1–5.

[8] Y. Kim and R. Gunasekaran, "Understanding i/o workload characteristics of a peta-scale storage system," *The Journal of Supercomputing*, pp. 1–20, 2014.

[9] S. Oral, D. A. Dillow, D. Fuller, J. Hill, D. Leverman, S. S. Vazhkudai, F. Wang, Y. Kim, J. Rogers, J. Simmons *et al.*, "Olcfs 1 tb/s, next-generation lustre file system," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[10] Prabhat and Q. K. *(ed)*, *High Performance Parallel I/O*. CRC Press, 2014, vol. 22.

[11] P. J. Braam *et al.*, "The lustre storage architecture. cluster file systems," *Inc. http://www. clusterfs. com*, vol. 8, p. 29, 2003.

[12] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo, "Terascale direct numerical simulations of turbulent combustion using s3d," *Computational Science & Discovery*, vol. 2, no. 1, p. 015001, 2009. [Online]. Available: http://stacks.iop.org/1749-4699/2/i=1/a=015001

[13] S. Ethier, W. M. Tang, and Z. Lin, "Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms," *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 1, 2005. [Online]. Available: http://stacks.iop.org/1742-6596/16/i=1/a=001

[14] "Flash center for computational science, university of chicago," http://flash.uchicago.edu/site/flashcode/, 2015.

[15] "Ncar ucar community earth system model," http://www2.cesm.ucar.edu/, 2015.