Output Performance Study on a Production Petascale Filesystem

Bing Xie[1], Jeffrey S. Chase[1], David Dillow[2], Scott Klasky[3], Jay Lofstead[4], Sarp Oral[3], and Norbert Podhorszki[3]

[1] Department of Computer Science, Duke University, Durham, USA
[2] dave@thedillows.org
[3] Oak Ridge National Laboratory, Oak Ridge, USA
[4] Center for Computing Research, Sandia National Laboratories, Albuquerque, USA

Abstract. This paper reports our observations from a top-tier supercomputer Titan and its Lustre parallel file stores under production load. In summary, we find that supercomputer file systems are highly variable across the machine at fine time scales. This variability has two major implications. First, stragglers lessen the benefit of coupled I/O parallelism (striping). Peak median output bandwidths are obtained with parallel writes to many independent files, with no striping or write-sharing of files across clients (compute nodes). I/O parallelism is most effective when the application—or its I/O middleware system—distributes the I/O load so that each client writes separate files on multiple targets, and each target stores files for multiple clients, in a balanced way. Second, our results suggest that the potential benefit of dynamic adaptation is limited. In particular, it is not fruitful to attempt to identify "good spots" in the machine or in the file system: component performance is driven by transient load conditions, and past performance is not a useful predictor of future performance. For example, we do not observe regular diurnal load patterns.

Keywords: Parallel I/O, Petascale filesystem, Output performance

1 Introduction

Output bandwidth is a precious resource in supercomputers. Trends suggest that this limitation is not likely to change. Therefore it is crucial for software to make efficient use of the bandwidth. In principle, large write bursts can stream effectively and achieve full bandwidth. In practice, delivered bandwidth is highly sensitive to the application's use of storage APIs and its data layout, placing an unwelcome burden on domain scientists to manage I/O performance tradeoffs at the application level.

In this paper, we summarize results from systematic I/O benchmarking—focusing on output bandwidth—of the production supercomputer Titan, the 4th fastest supercomputer in the world. We extended the methodology used to study the Jaguar supercomputer in [21], and designed a set of experiments to stress load on individual stages of Titan's multi-stage write path. These experiments yield distributions of performance behaviors on Titan over time and across the machine, enabling us to assess the impact of key configuration parameters and choices. By studying the results through sequences of such experiments, we can characterize the behaviors of individual stages in the write path over time.

The key contribution of our study is to enhance understanding of performance behaviors for a state-of-the-art parallel filesystem as currently deployed in a leadership-class production facility. The study is useful to understand the current Titan deployment and also to build models that predict output absorption time as a function of various parameter settings [22]. Although some factors may be unique to Lustre and/or Titan, we expect that many of our observations are representative of large-scale computing systems and their I/O performance behaviors. Here is a summary of the primary conclusions:

- We find that a small proportion of storage targets ($< 20\%$) are straggling at any given interval, but that stragglers are transient: over time, any target may appear as a straggler for some intervals. Stragglers throttle the write pipelines, limiting striping bandwidth and reducing the benefits of parallelism.
- As configured on Titan, the Lustre write pipelines do not allow a single client to obtain the full bandwidth of a storage target. The results suggest that in the ideal case each client writes to multiple files spread across multiple targets, with multiple clients per target.
- The I/O performance delivered on Titan is highly variable. Our study suggests that historical performance data and monitoring do not enable adaptive middleware to locate "good spots" in the supercomputer or in the file system. Local performance behavior is transient and unpredictable.
- Delivered aggregate output bandwidth is sensitive to location (density) of a job's compute nodes for large bursts, under a static node-to-router mapping policy adopted by Titan in its internal network configuration.

Our study offers insights that can inform design and deployment choices for exascale facilities and also technical choices for the ongoing development of integrated software stacks for parallel storage including parallel file systems and I/O middleware systems such as ADIOS [12]. ADIOS implements a variety of techniques to improve output performance, and many applications now use ADIOS, e.g., S3D [3], XGC [10] fusion codes, and M8 earthquake simulations [5]. For example, ADIOS enables applications to configure their output buffer size. It can issue writes to multiple independent files to avoid performance problems associated with write-shared files and striping, and it reorganizes output data for better read performance. The results in this study provide a foundation to understand and quantify the impacts of these techniques, and may expose new opportunities to manage I/O performance.

This paper summarizes some key aspects of our methodology and results. We are preparing a full-length paper to present the results in more detail.

2 Output Behavior on Titan

File Systems	Service Time	Partitions	Routing Policy	I/O Nodes	OSSes	OSTs
Spider	Jan.2008–Dec.2013	4	fine-grained	192	192	336 × 4
Spider 2	Nov. 2013–present	2	fine-grained	432	288	1008 × 2

Table 1. File systems on Titan. A Lustre client (compute node) issues I/O operations to RAID targets (OSTs) attached to Object Storage Servers (OSSes). The I/O path traverses the internal interconnect to a selected I/O node, which acts as a router to forward I/O traffic between the internal interconnect and an external storage network. In Titan the mapping of compute nodes to I/O nodes is static ("fine-grained") when all I/O nodes are functioning normally [7].

This section summarizes selected aspects of the burst absorption behavior of Titan and two of its Lustre file systems (see Table 1): Spider (Widow1) and Spider 2 (Atlas2). We design a sequence of experiments to stress the components and stages of the write pipeline using the methodology in [21]. Each experiment is a set of identical *runs*; each run varies one or more parameters across a sequence of values in each *round*. The experiments yield one instance (a sample point) in each round for each value of a varying parameter. Each instance reports output bandwidth delivered to a group of nodes writing a synchronized output burst from an IOR benchmark program. The runs occur at regular intervals over the measurement period. In this way, we profile Titan's write path statistically with multiple samples spread over time. We use several measures of output bandwidth:

- *Bandwidth* is measured as MB/s per client node.
- *Aggregate Bandwidth*, measured in MB/s, is bandwidth summed across all client nodes in an instance.
- *Effective Aggregate Bandwidth (EAB)* is aggregate bandwidth normalized to the peak bandwidth achievable from the number of targets written in an instance under a given set of parameters.

2.1 Pipeline Efficiency

We evaluated the efficiency of the write pipeline from a single client: a single process running on a single core to a single target (OST), as a function of burst size. The data is based on the measurements taken from March to July 2013 on Spider/Widow1. This experiment has 200 runs with 3 rounds each.

Figure 1 gives the results. Each boxplot displays a quantile distribution of samples for the corresponding parameter value on the x-axis, with "whiskers

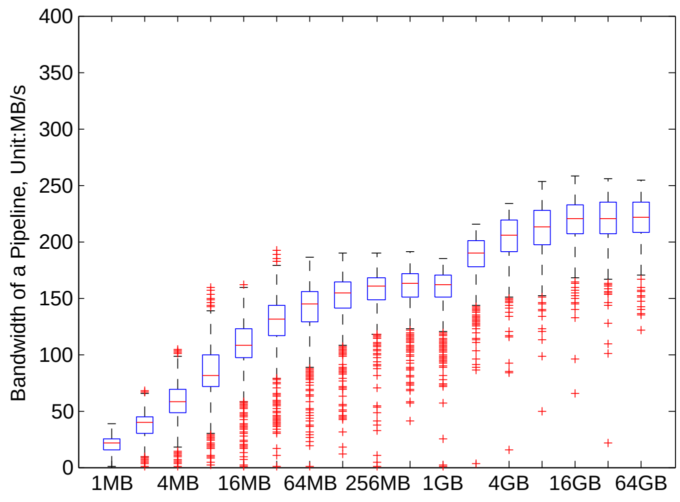

Fig. 1. Bandwidth of a single pipeline as a function of burst size. This graph shows results for a single process on a single core writing a single file on a single target. Other results (not shown) indicate that more client processes do not help: the configured write pipeline is not deep enough for one client to obtain full bandwidth from a target.

and dots" for the outliers. Each boxplot contains one point from each of the rounds—the result of the instance for the corresponding parameter value from that round. The upper and lower borders of each box are the 25th and 75th percentile values (lower quartile Q1 and upper quartile Q3). The band within each box denotes the median value. The value Q3-Q1 is the interquartile range or IQR; thus 50% of the y-values reside within the box, and the IQR is the height of the box. The upper and lower whiskers cover the points outside of the box, except that the upper and lower bounds of the whisker do not extend beyond $Q3 + 1.5 * IQR$ and $Q1 - 1.5 * IQR$ respectively. All y-values outside of this whisker range are outliers and are plotted as individual points.

Figure 1 shows that single-pipeline bandwidth is sensitive to burst size, and that the write pipeline obtains its maximum overall bandwidth with a write burst of 2 GB or more. With these burst sizes the pipeline runs at full bandwidth for long enough to dominate the time to fill and drain the pipeline.

The results suggest that the conservative flow control configuration for output pipelines in Lustre (e.g., at most eight outstanding RPCs per client-target pair) prevents a single client from obtaining the full bandwidth of any target. This was true in the Jaguar study as well, and has continued to be true on the Titan Lustre deployment. One possible cause is that enhancements for asynchronous journaling (see [16]) may delay the RPC replies from the targets, requiring a larger number of outstanding RPCs for effective write streaming.

We determined from the multi-core experiment (not shown) that using multiple cores on a client does not help. In the multi-core experiment, each client runs multiple single-threaded IOR processes, each issuing a single output burst to a separate file on the single target, synchronized with MPI barriers. Using multiple cores from a client improves the delivered bandwidth by at most 5%.

Figure 1 also shows that many of the trials deliver low bandwidths. The results show substantial outliers on the low side (3% to 5% of all samples). Other experiments suggest that these are due to intermittent contention on the internal Titan interconnect.

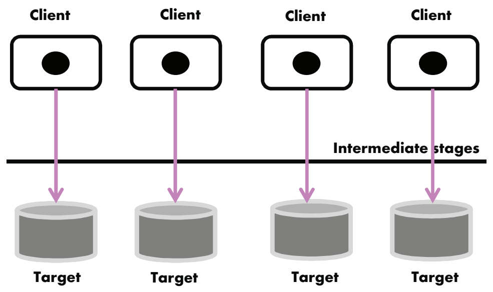

Fig. 2. Template for the many-pairs experiment. Each client node runs a single process that issues a 64 MB burst to an unstriped file on a selected target. Each client selects a different target. The bursts are synchronized. We vary the number of client-target pairs and measure aggregate bandwidth and the bandwidth (or completion time) for each client-target pair.

2.2 Many-Pairs Bandwidth and Stragglers

The "many pairs" experiment probe the aggregate I/O bandwidths achievable on Titan and the consistency of performance in different parts of the machine. The runs use equal numbers of clients and targets grouped in client-target pairs: each client runs a single process that writes a single file on a single target, following the template of Figure 2. We ran this experiment from February to July 2013 on Spider/Widow1 and produced 200 runs with 3 rounds each. At the largest scale we use 336 compute nodes to write to all 336 targets in the Widow1 storage system on Titan. The results reported here use a fixed burst size of 64 MB for each node-target pair.

A key factor in this experiment is the variance in completion times for the pairs in each instance. The bursts for all pairs are synchronized, and the aggregate bandwidth (or EAB) is determined by the completion time of the slowest pair. Some pairs in each instance complete quickly while others are "stragglers" that limit the aggregate bandwidth.

To quantify the impact of stragglers, Figure 3 plots the cumulative distributions of completion times across all client-target pairs for each instance of the experiment. In all cases, more than 95% of the synchronized bursts complete within 2 seconds, but almost every trial has a tail of stragglers, which cause other pairs in the instance to idle while waiting for the stragglers to finish. The

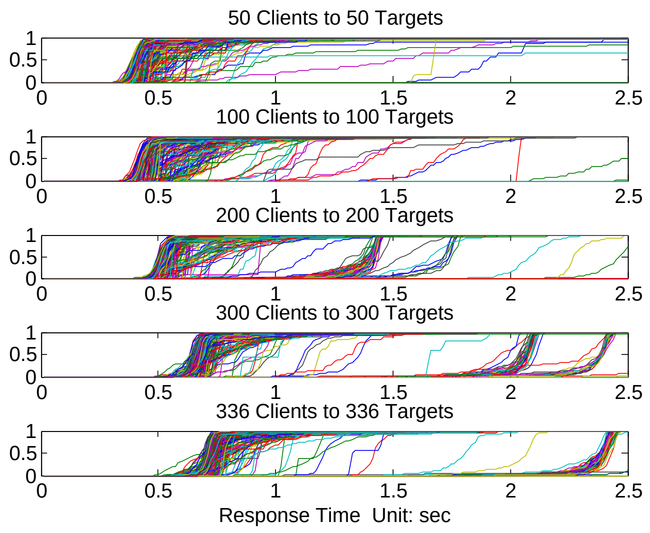

Fig. 3. CDFs of completion times for the instances of the many-pairs experiment. Each subgraph has 600 CDF lines, one for a trial of an instance. Each line shows the distribution of completion times for the pairs of one trial. Each line of the five types of instances has 50, 100, 200, 300 and 336 points (pairs) respectively. It is easy to see that almost every trial has good performance in some parts of the machine, as well as stragglers that limit the aggregate bandwidth.

impact of stragglers grows as we increase the number of pairs: both the number of stragglers and their completion times increase substantially.

Stragglers may be caused by bottlenecks in the interconnect, and not necessarily in the targets themselves. Using all 336 targets, the completion times of even the fastest pairs are noticeably higher, indicating that the run has triggered congestion in intermediate stages, uniformly affecting all pairs.

These stragglers are significant in part because of their impact on performance with striping. We found and reported in [21] that straggling targets gate the bandwidth of striped write operations. This situation improved when the Lustre client software was upgraded to improve internal concurrency using a pool of threads to handle RPC load in the client [19]. But the average write bandwidth with striping is still substantially lower than the bandwidth achievable using independent writes.

2.3 Performance Variability of Individual Components

To probe the stability of stragglers and further explore the opportunity to locate and avoid stragglers with adaptive I/O tools (e.g., ADIOS), we design a new experiment template to quantify the persistence of stragglers. It follows the template of the many-pairs experiment (Figure 2): in each instance N synchronous processes from N clients write to a sequence of N OSTs. However, in this exper-

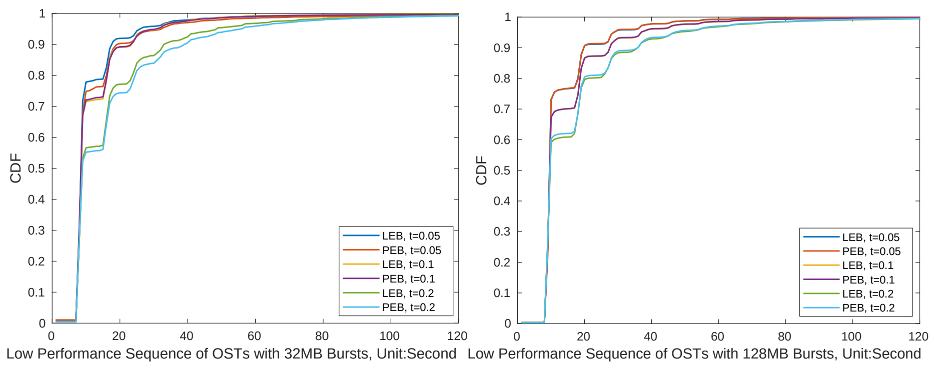

Fig. 4. CDFs of low behavior sequences of OSTs. From left to right, each subfigure shows the CDF of the time durations of the low-performance periods with 32MB and 128MB bursts respectively; in each subfigure, a line shows the CDF of the low-performance sequences determined by the quantile threshold t (defined in §2.3).

iment each client is paired to one of the N-length OST sequence according to a round-robin policy across consecutive instances in a run. In each run, the group of clients, the candidate OSTs, and the burst size are all fixed. We conducted 8 such experiments on Titan/Atlas2 with 1008 OSTs (see Table 1) from January to February 2017: in each experiment 126 coordinated clients focus bursts on a different sequence of 126 OSTs with 32MB and 128MB bursts respectively. In this set of experiments, the time duration of a run ranges from 0.5—1.7 hours; the time interval between two consecutive measures in a run ranges from 7—15 seconds.

To quantify the performance of individual components, we assign each node-target pair in an instance two relative measures: LEB (*Lag Effective Bandwidth*) and PEB (*Pair Effective Bandwidth*).

– The LEB score is the pair's bandwidth normalized to the fastest pair in its instance. The fastest pair ($LEB = 1$) gives a rough measure (a lower bound) of the performance achievable under the parameters and general system conditions for that instance.
– The PEB score is the pair's bandwidth normalized to the fastest pair measure within similar instances, which share identical parameter settings but run at different times. Such similar instances form *an equivalent instance set*. The fastest pair in a set ($PEB = 1$) gives a rough measure of the performance achievable for the set under ideal conditions: it is the best observed performance for any pair using those parameter settings.

We use these relative measures of component performance to compensate for the effect of general contention (e.g., in the interconnect) that affects a large share of the machine. We find that over 99.5% of the LEB/PEB scores of individual compute nodes and storage targets are within the range in 0.4—0.9

across experiments and burst sizes. These measures allowed us to identify targets that were persistent stragglers due to a load imbalance in an early Titan configuration; this problem has since been fixed.

We take a two-step approach to quantify the stability of performance behavior for individual components over time:

1. Label a LEB/PEB score of the component as *low* or *normal* performance according to a threshold: if the score of the component is below the threshold, it is considered a low performance measure; otherwise, it is a normal performance measure. We determine a threshold according to a chosen quantile (t) of LEB/PEB scores obtained from each equivalent instance set, i.e., of all instances with the same parameter setting.
2. Measure the lengths of consecutive sequences of low/normal performance measures for the component across its time series. Long sequences indicate that performance states are stable over time; short sequences suggest that they are not.

We focus on three quantiles: t=0.05, =0.1, =0.2. Figure 4 shows the summary of low performance periods for storage targets. It suggests that for 32MB (or 128MB) bursts more than 96% (or 100%) of storage targets showing low performance return to normal within a minute (or 2 minutes). Similar analyses suggest that a node showing low performance tends to return to normal within 2 minutes, and any component showing normal performance tends to switch to low performance within 10 minutes.

Based on these system-wide measurements at small time scales, we conclude that local performance in Titan's I/O system is highly variable over time. This high variability suggests that it is not fruitful to identify "good spots" in the machine or in the file system for the purpose of improving I/O performance.

2.4 Performance Variability and Node Locality

This section probes performance variation across compute node locations. To this end, we examine the many-pairs experiment again (the template in Figure 2) with 16MB and 256MB bursts from each of 1008 compute nodes to a different storage target. We also extend the methodology to group the runs into *sets* each comprising multiple identical runs with the same group of compute nodes, closely spaced in time. Different sets executed on different groups of compute nodes and at different times.

Our analysis is based on measurements taken from May to June 2015. We collected 95 sets with a total of 103 runs; a few sets have multiple runs. Each run comprises 15 rounds of instances with 16MB and 256MB bursts respectively.

To explore the set behaviors for different burst sizes, we estimate the node distribution of each set by measuring the average path length (L) between the nodes in all pairs of nodes drawn from the set. A smaller L indicates a more tightly packed (denser) node set; a larger L indicates a more widely scattered node set. To measure the distance for each node pair in a set, we choose a

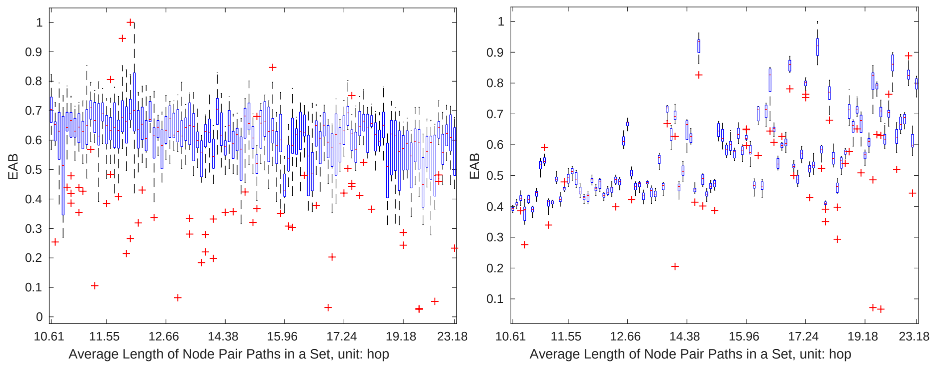

Fig. 5. EABs of the 95 sets with 16MB and 256MB bursts. From left to right, a subfigure reports the boxplots of the EABs of the 103 runs from the 95 sets with 16MB and 256MB bursts respectively. In each subfigure, the x-axis represents the 95 node sets sorted by L (defined in §2.4); the corresponding y values summarize the distribution of measured bandwidths (EABs) for that node set.

common metric, L_1 *routing distance*: the length of a path between two points in Titan's 3d torus. For a node pair at positions (x_1, y_1, z_1) and (x_2, y_2, z_2), the distance (d) of the pair is given by:

$$d = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \tag{1}$$

The L values of the 95 sets vary from 10.61 to 23.18 hops. To explore the correlation between output performance and set density, Figure 5 plots their EABs ranked by density, for 16MB and 256MB bursts.

Figure 5 shows that for 16MB bursts the EABs of all sets are distributed in a wide range and are only weakly correlated with density. Small bursts are sensitive to transient contention on the shared intermediate stages or on the storage servers/targets. However, for the 256MB bursts, the sets with larger L are more likely to deliver higher aggregate bandwidths.

To quantify the behaviors of the sample-size bursts on various set densities, we further partition the 95 sets into three ranges of average hop distance: 10.61 — 15, 15.01 — 20 and 20.01 — 23.18. Figure 6 shows the output bandwidths (EABs) for the instances in each range. It suggests that, for 256MB bursts, above 80% of the instances in the L range 1, range 2 and range 3 report ∼0.4, ∼0.52 and ∼0.67 EABs respectively. For the larger bursts the sets with larger L tend to deliver higher aggregate bandwidths.

We conclude that while the bandwidth of small bursts is dominated by transient contention, the performance of large bursts is impaired by denser node sets. Of course, the job scheduler prefers dense node sets because a densely packed job experiences less cross-contention from other jobs on the internal interconnect. However, denser sets may experience self-contention on the internal interconnect and also are locked into using a smaller set of I/O routers, since the binding of

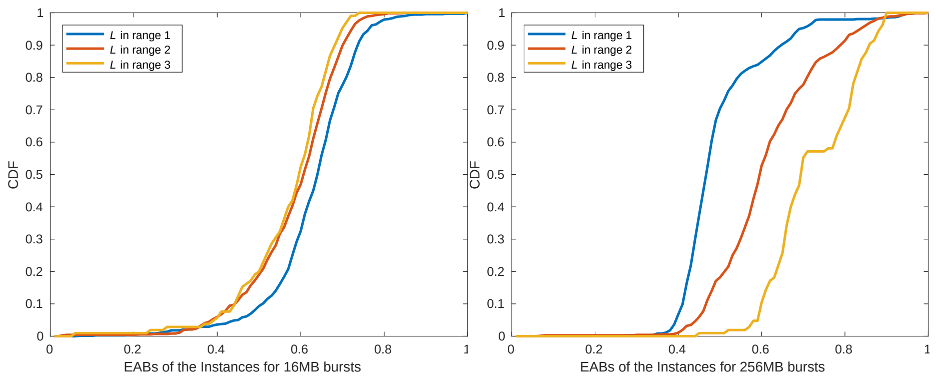

Fig. 6. CDFs of the instance EABs of the 95 sets with 16MB bursts (left) and 256MB bursts (right). In each subfigure, the lines (blue, red and yellow) depict the CDFs of the instances examined on the sets with L in three hop ranges respectively: 10.61 — 15 (720 instances), 15.01 — 20 (720 instances), and 20.01 — 23.18 (105 instances).

nodes to routers is static and determined by node location (Table 1). More dispersed sets spread their loads across a larger portion of the interconnect and a larger number of I/O routers, and tend to show higher bandwidth accordingly.

Moreover, it is worth noting that for 16MB bursts, the results suggest slightly better performance for denser node sets. Even dense sets are free of self-contention for small bursts, and may benefit from the lack of cross-contention in the interconnect and I/O routers from other jobs on the machine. Even so, in a busy and highly contended system like Titan, the transient system conditions on the storage servers/targets dominates this effect. For longer bursts the transient hot spots in the storage system tend to cancel out, and the results are dominated by persistent self-contention in the interconnect and I/O routers. This suggests that the I/O routing policy could be improved to spread load for large burst. Moreover, in this scenario I/O adaptive tools (e.g., ADIOS) might be helpful to move and redistribute the load across the machine.

3 Related Work

Several studies benchmark HPC file systems by measuring their performance under real application workloads. Several influential studies were published in the 1990s [14, 8, 15, 6, 4]. A significant recent study installs continuous monitoring software on compute nodes to characterize the I/O requests of real application workloads in real time, modulating the data collected to keep overhead within acceptable limits [2, 1, 13].

Uselton et al. [20] propose a statistical method to collect and analyze I/O events to more fully characterize the I/O behavior of ensembles. They also observe the straggler phenomenon, suggesting that the straggler problem is a general issue in supercomputers. Their work focuses on improving the I/O performance of a given application in a given supercomputer system. Our goal is to characterize the multi-stage write pipeline in a petascale file system, locate write absorption bottlenecks, and capture component performance variability that influence on the design and configuration choices for adaptive middleware and HPC applications.

Other previous studies use an approach similar to ours: stress the file system with synthetic benchmarks. A number of HPC I/O benchmarks are designed to be sufficiently flexible to emulate the typical I/O behaviors in supercomputer environments, such as the FLASH I/O, IOR, and BTIO benchmarks. This flexibility enables users to configure the benchmark for a desired pattern approximating an observed application behavior. In our work, we take IOR as a generator and run different patterns and configurations to focus traffic on specific stages and elements of the write pipeline to gain a complete picture of output burst absorption in a production facility.

A recent study [11] uses a similar methodology to measure the performance of the Intrepid file system at the Argonne Leadership Computing Facility. The authors report the capacity of each I/O stage and measure the behavior of the entire subfile system for large-scale runs of a set of benchmarks. The measurements are taken on dedicated hardware before the supercomputer system was running in production mode. Our work explores the delivered bandwidth of the I/O stages in ongoing production use, reflects the impact of competing workloads under observed usage patterns in production, and shows how to filter noise from competing workloads to obtain insights into the behavior of the underlying hardware and software.

Earlier studies also use configurations of the IOR benchmark to analyze the behavior of HPC systems [18] [17]. Kim et al. [9] collect I/O performance data from Titan's predecessor Jaguar. That study is complementary to ours: it reports monitoring data from the storage servers showing the combined workload on the machine. We focus on the end-to-end behavior observed by jobs running on the compute nodes, and the impact of write patterns and I/O configuration choices.

4 Conclusion

I/O bandwidth is a scarce resource on supercomputers. Output burst absorption can have a substantial impact on delivered performance, as demonstrated by a simple performance model. Observed output bandwidth is sensitive to various uses of the storage system APIs and different supercomputer I/O system conditions.

We apply a statistical benchmarking approach to probe Lustre filesystem output performance in Titan and its Spider and Spider 2 file stores. The measured distributions quantify the frequency and severity of contention (stragglers) and other transient system conditions. These imbalances lessen the benefit of coupled I/O parallelism (striping). This effect motivates structuring choices to loosen the coupling of parallel I/O. For example, on Titan's I/O system under typical conditions, the peak median output bandwidths are obtained with parallel writes to many independent files, with no write-sharing or striping, and with each target storing files for multiple clients, and each client writing files on multiple OSTs.

The prevalence of these imbalances motivates adaptive responses in the I/O middleware layer. To evaluate the potential of adaptation, we studied the behavior of individual components to expose temporal usage patterns, slow components and system-level performance variability that can lead to imbalances in the write path. Our results show that these performance stutters are difficult to predict, and that system changes state quickly and frequently, suggesting that dynamic adaptation to congestion is not a fruitful approach.

Under a static node-to-router mapping policy adopted by Titan in its network configuration, for large bursts output performance is sensitive to the density of a job's compute nodes, as measured by the mean pairwise routing path distance within the node group.

Acknowledgment

We thank Chris Zimmer from OLCF for his detailed explanation on the network configuration of Titan.

The work was supported by the U.S. Department of Energy, under FWP 16-018666, program manager Lucy Nowell.

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

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, R. Ross: Understanding and improving computational science storage access through continuous characterization. ACM Trans. on Storage 7(3), 8–26 (2011)
2. P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, K. Riley: 24/7 characterization of petascale I/O workloads. In: Proceedings of the IEEE International Conference on Cluster Computing(CLUSTER'09). pp. 1–10. New Orleans, LA (2009)
3. L. Chacón: A non-staggered, conservative, finite-volume scheme for 3D implicit extended magnetohydrodynamics in curvilinear geometries. Computer Physics Communications 163(3), 143–171 (2004)
4. P.E. Crandall, R.A. Aydt, A.A. Chien, D.A. Reed: Input/output characteristics of scalable parallel applications. In: Proceedings of the ACM/IEEE Conference on Supercomputing(SC '95). pp. 59–89. San Diego, CA (1995)

5. Y. Cui, K. Olsen, T. Jordan, K. Lee, J. Zhou, P. Small, D. Roten, G. Ely, D. Panda, A. Chourasia, J. Levesque, S. Day, P. Maechling: Scalable earthquake simulation on petascale supercomputers. In: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis(SC'10). pp. 1–20. Washington, DC (2010)
6. R. Cypher, A. Ho, S. Konstantinidou, P. Messina: Architectural requirements of parallel scientific applications with explicit communication. In: Proceedings of the 20th Annual International Symposium on Computer Architecture(ISCA'93). pp. 2–13. San Diego, CA (1993)
7. M. Ezell, D. Dillow, S. Oral, F. Wang, D. Tiwari, D. Maxwell, D. Leverman, J. Hill: I/O router placement and fine-grained routing on Titan to support Spider II. In: Proceedings of the Cray User Group Conference(CUG'14). pp. 1–6. Lugano, Switzerland (2014)
8. G.R. Ganger: Generating representative synthetic workloads: an unsolved problem. In: Proceedings of the Computer Measurement Group Conference(CMG'95). pp. 1263–1269. Nashville, TN (1995)
9. Y. Kim, R. Gunasekaran, G.M. Shipman, D.A. Dillow, Z. Zhang, B.W. Settlemyer: Workload characterization of a leadership class storage cluster. In: Proceedings of the 5th Petascale Data Storage Workshop(PDSW'10). pp. 1–5. New Orleans, LA (2010)
10. S. Ku, C.S. Chang, M. Adams, J. Cummings, F. Hinton, D. Keyes, S. Klasky, W. Lee, Z. Lin, S. Parker, the CPES team: Gyrokinetic particle simulation of neoclassical transport in the pedestal/scrape-off region of a tokamak plasma. Journal of Physics 46(1), 87–91 (2006)
11. S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, W. Allcock: I/O performance challenges at leadership scale. In: Proceedings of the ACM/IEEE International Conference for High Performance Computing Networking, Storage and Analysis (SC'09). pp. 40–52. Portland, OR (2009)
12. J. Lofstead, F. Zheng, S. Klasky, K. Schwan: Adaptable, metadata-rich I/O methods for portable high performance I/O. In: Proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium(IPDPS'09). pp. 1–10. Rome, Italy (2009)
13. H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, M. Prabhat, S. Byna, Y. Yao: A multiplatform study of I/O behavior on petascale supercomputers. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing(HPDC'15). pp. 33–44. Portland, OR (2015)
14. A.L. Narasimha Reddy, P. Banerjee: A study of I/O behavior of perfect benchmarks on a multiprocessor. In: Proceedings of the 17th Annual International Symposium on Computer Architecture(ISCA'90). pp. 312–321. Seattle, WA (1990)
15. N. Nieuwejaar, D. Kotz, A. Purakayastha, C.S. Ellis, M.L. Best: File-access characteristics of parallel scientific workloads. IEEE Trans. on Parallel and Distributed Systems 7(10), 1075–1089 (1996)
16. S. Oral, F. Wang, D. Dillow, G. Shipman, R. Miller, O. Drokin: Efficient object storage journaling in a distributed parallel file system. In: Proceedings of the 8th USENIX Conference on File and Storage Technologies(FAST'10). pp. 143–154. San Jose, CA (2010)
17. H. Shan, K. Antypas, J. Shalf: Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis(SC '08). pp. 42–54. Austin, TX (2008)

18. H. Shan, J. Shalf: Using IOR to analyze the I/O performance for HPC platforms. In: Proceedings of the Cray User Group Meeting(CUG'07). pp. 1–15. Washington, DC (2007)
19. G. Shipman, D. Dillow, D. Fuller, R. Gunasekaran, J. Hill, Y. Kim, S. Oral, D. Reitz, J. Simmons, F. Wang: A next-generation parallel file system environment for the OLCF. In: Proceedings of the Cray User Group Conference(CUG'12). pp. 1–12. Stuttgart, Germany (2012)
20. A. Uselton, M. Howison, N.J. Wright, D. Skinner, N. Keen, J. Shalf, K.L. Karavanic, L. Oliker: Parallel I/O performance: from events to ensembles. In: Proceedings of the 24th IEEE International Parallel & Distributed Processing Symposium(IPDPS'10). pp. 1–11. Atlanta, GA (2010)
21. B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, N. Podhorszki: Characterizing output bottlenecks in a supercomputer. In: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12). pp. 1–11. Salt Lake City, UT (2012)
22. B. Xie, Y. Huang, J.S. Chase, J.Y. Choi, S. Klasky, J. Lofstead, S. Oral: Predicting output performance of a petascale supercomputer. In: Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing(HPDC'17). pp. 1–12. Washington D.C. (2017)