New Challenges of Benchmarking All-Flash Storage for HPC

Glenn K. Lockwood, Alberto Chiusole, Nicholas J. Wright
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
{glock,chiusole,njwright}@lbl.gov

***Abstract*—The proliferation of extreme-scale analytics and AI have motivated new parallel storage systems that use flash and storage class memory (SCM) exclusively to achieve the best attributes of both media, and such storage systems have clear relevance in HPC. We present a preliminary evaluation of one such exemplar system from VAST Data, which incorporates SCM and QLC flash in a single namespace, to quantify the benefit of such architectures. We show that traditional I/O performance measurement techniques struggle to properly characterize all-flash storage systems because they were designed to test much simpler storage systems, and we propose alternative methods to better reflect the performance that real workflows can expect.**

I. Introduction

The inclusion of solid-state storage media technologies–including NAND flash and 3D XPoint–in HPC storage subsystems has been underway for the last decade. The earliest flash-enabled HPC systems incorporated hundreds of terabytes of enterprise SATA SSDs in 2011 [1] to provide higher IOPS and bandwidth per-device over hard disk drive (HDD) storage. Petabyte burst buffers built on NVMe SSDs followed in 2015 [2]–[4], delivering increased performance owing to the significant performance gain from attaching SSDs directly to hosts' PCIe buses. Burst buffers also included advances in software design; the relative scarcity of flash brought dynamically schedulable and user-defined high-performance storage into HPC [5], [6]. Falling flash prices are leading the HPC industry to begin deploying all-flash storage which provides users with the performance of a burst buffer and the usability of a single namespace–with no requirement to explicitly stage data to account for tiered storage [7].

A new generation of flash-native storage systems is now being developed, including Intel DAOS [8] and VAST Universal Storage [9]. Both DAOS and VAST rely on combining 3D XPoint storage class memory (SCM) and low-endurance, high-capacity quad-level cell (QLC) flash to deliver the low write latency and high endurance of SCM and the low read latency and economical capacity of QLC as a single namespace. This tiered SCM/QLC architecture is showing great promise to date; for example, DAOS has shown significantly higher performance on metadata operations and misaligned I/Os than traditional parallel file systems [10].

Large HPC systems have historically defined I/O performance along only two dimensions: peak bulk-synchronous I/O bandwidth and metadata rates. One might assume that since SSDs can also deliver exceptional IOPS relative to HDDs, peak bulk-synchronous random I/O rates should also be a new dimension of performance. However, this is not an actual workload relevant to HPC, since few applications issue random synchronous I/O to achieve scientific ends [11]. Bulk-synchronous I/O rates are also not representative of what most users observe because they emulate a full-system compute job performing a bulk-synchronous checkpoint or restart in the absence of interfering I/O workloads, and they almost always test the read performance of recently written data.

Actual I/O performance is often lower for many reasons. For example, most HPC systems run many jobs concurrently which individually occupy only a subset of all compute nodes [12]. Many emerging workloads in artificial intelligence [13], [14], genomics [15], and experimental high-energy physics [16], [17] also access data in a way that does not map to the large, well-aligned stripes in which parallel file systems arrange data. Only recently has the parallel I/O community (notably, the IO-500 effort [18]) acknowledged that high throughput with such non-contiguous I/Os is important to scientific computing. User expectations are also evolving with the increase of interactivity arising from new user interfaces like Jupyter [19] and data sources such as experimental detectors [16], [20]–[24], resulting in tail latency and responsiveness becoming key measures of I/O performance.

Thus, the I/O performance analysis community must evolve to meet the challenges of new technologies and new usage modalities. In this work, we explore these challenges by demonstrating how traditional I/O benchmarking approaches misrepresent the performance users should actually expect from such next-generation SCM/QLC storage systems. In doing so, we present several approaches to develop a more comprehensive understanding of HPC storage system performance that better reflects both the strengths of next-generation all-flash file systems and what user workloads should expect.

II. Test Methodology

We chose to use VAST Data's storage system as an exemplar of next-generation all-flash storage systems. Fig. 1 illustrates the architecture of this system which is composed of two types of servers, "DNodes" and "CNodes." DNodes host the file system's SCM and SSDs through directly attached PCIe but otherwise do little beyond projecting bare drives over the

NVMe fabric using NVMe over Fabrics (NVMeoF). CNodes bridge the client-facing network and VAST's internal NVMe fabric, and most of VAST's data path logic is implemented within these servers.

CNodes are stateless and rely exclusively on transactionally consistent data structures on remote SCM and flash to store user data and metadata. The data path is designed such that all incoming writes are synchronously replicated to SCM and organized into multi-gigabyte stripes based on their similarity to existing data and expected lifetime. Asynchronously, when a stripe has completely filled, it is compressed and migrated to QLC drives by CNodes. The details of this data path are further described elsewhere [9], but suffice it to say that this hybrid SCM/QLC data path complicates performance analysis, and in this work we specifically examine the effects of the following advanced architectural features on performance.

Network erasure coding – Extreme-scale storage systems are turning to synchronous replication and erasure coding (EC) between servers to improve resilience over larger failure domains. Replication and EC use distributed transactions though, and these use extra network hops and bandwidth to ensure synchronous data consistency. It is important to evaluate whether these extra hops and data placement logic impact random I/O performance.

Hybrid SCM and QLC flash – The use of small quantities of high-durability SCM backed by large quantities of low-endurance QLC flash promises the random write and metadata performance of SCM at the economics of high-capacity flash. This tiered media scheme complicates performance analysis because the client has no indication whether data accesses are being served by SCM or QLC, and benchmarks may follow a data path that real user data and applications do not.

High-performance data reduction – AI is breathing new life into old scientific datasets and motivates the need for high-performance storage at capacities that exceed what has been driven solely by advances in processing performance [21]–[24]. This makes high-performance data reduction (e.g., fast or hardware-accelerated compression algorithms, deduplication, or similarity-based reduction) integral to delivering affordable, extreme-scale all-flash storage. That said, data reduction is orthogonal, if not opposite, to high-performance I/O, and we must examine the implications of sophisticated similarity-based reduction and compression on performance and benchmarking methodologies.

A. Test Platform

In this study, we used a VAST system comprised of eight CNodes and two DNodes as depicted in Fig. 1. Each CNode was connected to the client-facing network using one 100G EDR InfiniBand connection, and the two DNodes provided 44× Intel D5-P4326 15.36 TB QLC SSDs and 12× Intel DC P4800X 1.5 TB Optane SSDs total. We used up to eight clients each with two Intel Xeon Gold 6148 CPUs, 384 GiB of DRAM, and four 100G EDR InfiniBand HCAs. Clients connected to VAST CNodes using VAST's implementation of NFS over RDMA with `nconnect` and multipathing enabled.

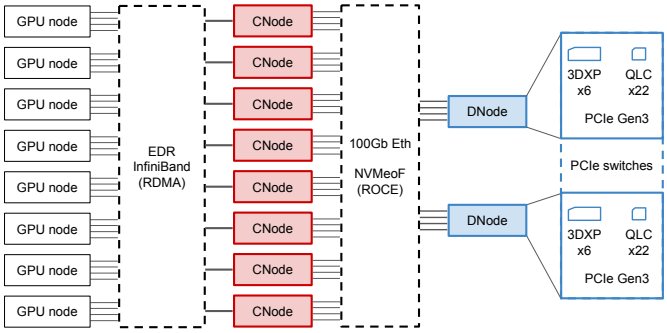

Fig. 1. High-level architecture of VAST system used in this study. GPU nodes were used as clients, and more detail can be found in Sec. II-A. Each DNode serves 6 SCM and 22 QLC drives each, and DNode pair supports failover.

B. Performance Datasets

To measure I/O bandwidth we used the IOR benchmark to generate parallel sequential read and write I/O workloads. Writes ran for at least 45 seconds using stonewalling with wear-out to ensure that all processes performed the same amount of I/O, and the benchmark walltime included the slowest process as would happen with a synchronous checkpoint/restart operation. We used 1, 2, 4 and 8 clients ($N = \{1, 2, 4, 8\}$) and 1, 2, 4, 8 and 16 processes per node ($p = \{1, 2, 4, 8, 16\}$ for both reads and writes. In addition, we ran tests for multiple transfer sizes $t = \{4\text{ KiB}, 512\text{ KiB}, 1\text{ MiB}, 4\text{ MiB}, 8\text{ MiB}, 32\text{ MiB}\}$. Each combination of ($N$, p, t, and access–read or write) was run five times for a total of 1,200 individual bandwidth tests.

We also used IOR to measure random access rates. As with bandwidth tests, all I/O was file-per-process and stonewalling was used to ensure write test ran for at least 45 seconds, but wear-out was not used since our goal was to test system-level performance rather than emulate a checkpoint/restart operation. Random read tests were performed against a large, pre-generated 27 TiB, 1024-file dataset that was specifically created such that read IOPS tests would run for at least 45 seconds without reaching end of file. We ran tests using the same values of N and p used for bandwidth tests, but $t = 4$ KiB and accesses were at random, 4 KiB-aligned, non-repeating offsets. Each combination of N and p was run five times for reads and writes for a total of 200 measurements.

In all read tests, all files were read from a different client than the one that generated it to avoid client read caches, and write tests were followed by an explicit `fsync(2)` which was included in the I/O time.

III. Sequential I/O Performance

A. Measuring Bandwidth Naïvely

We initially measured bandwidth using the standard approach of writing a large dataset, shifting MPI ranks by p to ensure reads did not hit client cache, and immediately reading it back. Fig. 2 shows the read and write bandwidths as a function of concurrency, averaged over all combinations of $N \times p$ and t.

The file system shows read bandwidths increasing without saturation with increasing concurrency as one would expect

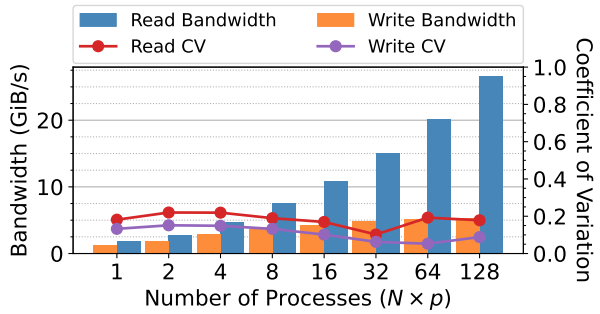

Fig. 2. Mean bandwidths of all bandwidth tests and accompanying coefficients of variation. Each x value contains all measurements that were taken using any values of (N, p, t) that produced the given total process count x.

from a parallel file system. Notably though, this scalability is achieved using an NFS client which has historically delivered very poor I/O performance due to its inability to perform parallel I/O. This data demonstrates that advancements in NFS such as multipathing (which allow the use of multiple network links between client and server), nconnect (which allow parallel data transfers between client and server), and NFS over RDMA allow NFS to achieve high bandwidth. Furthermore, the stateless nature of VAST CNodes allows clients to perform I/O to multiple CNodes concurrently without having to request file layout information as would be required by pNFS [25].

Figure 2 also shows the coefficient of variation (CV), and bearing in mind that each value of x ($= N \times p$) is the average over six different transfer sizes t and multiple combinations of N, p, we see that the storage system delivers very consistent bandwidth regardless of I/O size. For writes, we attribute this to NFS's ability to aggressively coalesce and asynchronously issue I/O. All data layout decisions are delegated to CNodes because VAST clients use NFS, and the extent to which writes can be coalesced into an RPC is not constrained by how a file's contents are mapped to remote storage servers. Relatedly, NFS's read-ahead is efficient since any VAST CNode can service any read request without locality effects. Consistent, high performance for many values of p and t is also a characteristic the Optane SCM to which all writes are issued in VAST [26].

Write bandwidth is lower than read bandwidth though. This is unsurprising since incoming writes are synchronously replicated to two SCM drives, a common technique that avoids the read-modify-write penalties intrinsic to synchronous erasure coding but consumes twice as much write bandwidth [27]. We also expect that the metadata associated with similarity-based data placement and reduction specific to VAST incurs overheads on write bandwidth that do not apply to reads, and other SCM/QLC storage systems that synchronously inject logic to facilitate intelligent tiering will show similar performance asymmetry. Reducing write bandwidth to increase IOPS complements AI workloads which are biased towards performing random reads [13], but this tradeoff may not suit all HPC workload mixes. For example, NERSC workloads average a 1.75:1 read:write ratio to its scratch file system [28] which is lower than the 5:1 ratio demonstrated in Fig. 2.

This gap between write and read performance could be narrowed toward this 1.75:1 ratio in principle. Adding more SCM would amortize the replication overhead, and adding more CNodes would provide additional resources to address the metadata overheads of incoming writes. Naïvely, adding 3× more SCM drives but reducing the per-drive capacity could reduce the read:write performance ratio to as low as 1.67:1 without adding unnecessary SCM capacity and cost. This is very speculative, but VAST's disaggregated architecture is conceptually amenable to such changes.

B. *Effects of Hybrid SCM/QLC*

Like hybrid storage systems that combine SSD and HDD, hybrid SCM/QLC storage systems asynchronously migrate data from high-performance SCM to high-capacity flash over time. Unlike SSD/HDD systems though, accessing long-lived data from the QLC layer is *faster* since both SCM and QLC deliver comparable read bandwidth but QLC provides the majority of the capacity.[1] For example, the VAST system used in this work reserved ≈ 3 TB of SCM across 12 drives for incoming writes, providing 30 GB/s of theoretical aggregate read bandwidth for newly written data. Its QLC layer (to which user data is never directly written) provides more capacity (608 TB) and theoretical read bandwidth (140 GB/s) owing to using almost 4 × more drives (44 SSDs). It follows that these hybrid SCM/flash architectures complicate benchmarking because newly generated data resides on the lower-bandwidth SCM tier, and read performance actually *increases* as data ages and is migrated to QLC.

As a result of this, the read bandwidth presented in Fig. 2 is misleading since it was measured using data that had been written seconds before. This left the majority of the data being read in SCM, not QLC, and read performance was limited by the bandwidth of 12 SCM drives. To make our read tests more realistic, we re-ran all 1,200 write and read tests by first performing all 600 write tests and preserving their outputs, resulting in 275 TiB of data being written and retained on the file system. We then artificially aged this dataset by writing 8 TiB of throw-away data to the file system to ensure that any remnants of our 275 TiB dataset still resident in the SCM write buffer would be migrated to QLC. After this artificial aging of our dataset, we ran the 600 read tests against it, and Fig. 3 demonstrates the resulting performance increase observed.

By simply reordering our synthetic benchmarks to avoid unrealistic streaming read-after-write I/O, we see over 50% higher read bandwidth when accessing our aged dataset at higher concurrencies (e.g., $N \times p = 32$). This is significant because we estimate that virtually all data read from large-scale HPC environments will target data that qualifies as being aged and therefore be read from QLC flash, not SCM. For example, the 35 PB all-NVMe Lustre file system on NERSC's Perlmutter supercomputer was designed to sustain a production workload of 2.2 PB of writes per day [29]. Given VAST's ratio of SCM write buffer to QLC capacity of ≈

[1]Intel data sheets rate the read performance of DC P4800X Optane SSDs at 2,500 MB/s and 3,200 MB/s for D5-P4326 QLC SSDs.

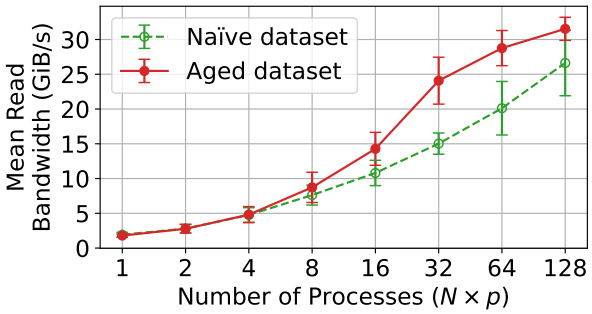

Fig. 3. Difference in read bandwidth measured from reading a newly generated dataset that resides on SCM (naïve; same data as Fig. 2) and an aged dataset that has been migrated to QLC media. Error bars signify one standard deviation.

1:200, a comparably sized VAST file system would turn over its $\approx$ 173 TB SCM write buffer every 113 minutes. Thus, we estimate that any data read more than two hours after it was first generated would qualify as aged.

Because VAST also employs data reduction based on global similarity in its SCM/QLC tiering, we had to extend IOR to generate fine-grained pseudorandom data using the golden ratio primes algorithm implemented in the elbencho benchmark[2]. Without high-entropy data, we observed written datasets compressing into narrow stripes that fit on a small subset of the QLC SSDs, limiting our read performance to much fewer than 44 QLC drives. Because scientific data is only modestly compressible [30], we expect real-world read performance to resemble our aged dataset in Fig. 3 rather than the highly compressible, lower-performance naïve dataset.

From this, we posit that most real-world read activity will target aged data, and most data will be only modestly compressible and therefore striped across many QLC drives. Thus, the naïve read bandwidth measurements shown in Figs 2 and 3, despite being measured in the de facto standard approach, understate the true read bandwidth that users will experience. This highlights the importance of increasing the sophistication of bandwidth testing on hybrid SCM/QLC storage systems to better match the access patterns, re-access times, and entropy that true scientific applications use.

IV. Random I/O Performance

A. Measuring IOPS Naïvely

Two extremes of I/O access patterns are fully sequential and fully random, and the performance at these extremes is measured in terms of bandwidth (as discussed above) and I/O operations per second (IOPS), respectively. Benchmarking random access performance on parallel file systems typically involves writing and reading[3] 4 KiB data at random offsets across many files to measure random access performance in the absence of complicating factors such as lock contention. Fig. 4 summarizes the results of such a test on our file system as measured using both the default client behavior (NFS client write-back caching and read-ahead enabled) and `O_DIRECT` to disable write-back caching.

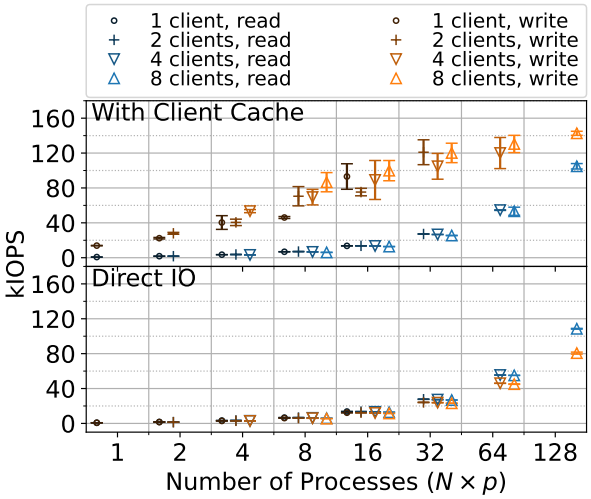

Fig. 4. Distribution of measured IOPS as a function of parallel I/O processes. Error bars signify one standard deviation.

Fig. 4 shows that random read rate is insensitive to different N for a given x on VAST; for example, the system will deliver similar aggregate performance to a single node randomly reading from eight cores and eight single-threaded random readers across eight nodes. However, the random read rate does not approach any obvious saturation point with increasing $N \times p$, indicating that we lacked enough concurrency to drive the full random read rate of the storage system.

Random write performance exhibits wider variation for different ratios of N and p (and even within the same N, p) when client cache is enabled. Also unlike random reads, random write performance also appears to saturate at the highest process counts which is a result of the entire benchmark being limited by write-back cache rate. Because VAST is able to achieve high write bandwidth independent of transfer size (as indicated by the low CV in Fig. 2) and its clients have no notion of data locality, this random write workload can be aggressively cached and coalesced into larger I/O operations.

Disabling write-back client caching using `O_DIRECT` significantly reduces the effective operation rate for random writes. The scaling of unbuffered random writes resembles random reads; it is insensitive to different combinations of N and p for any given $N \times p$ and never approaches saturation, indicating that the file system is capable of delivering more aggregate random writes than we could drive with eight clients. This disparity in apparent random write performance begs the question: which approach to benchmarking random writes results in a representative performance measurement?

B. Quantifying Nonsequential I/O Performance

Few HPC applications perform completely random writes [11], and this pattern is limited to few scientific use cases such as out-of-core sorting and data processing applications that update database-like files at unpredictable offsets. Because these random-write-heavy applications target write-only output files, the read-after-write consistency of

[2] *elbencho*. https://github.com/breuner/elbencho/. Accessed Aug. 20, 2021.

[3] Following the conclusions of Section III-B, random reads were measured using a dataset of randomized data that was artificially aged to ensure its residence on QLC.

`O_DIRECT` is rarely required. We consider this I/O pattern realistic only where an application performs random writes that are not page-aligned and the close-to-open consistency of NFS could result in inconsistent data in the absence of `O_DIRECT`. Because unbuffered random writing is such an adversarial I/O pattern, applications that rely on it historically experienced poor performance which has disincentivized its use in practice. On this basis, we arrive at the following conclusions for measuring random *write* performance:

- To benchmark *application-level* random I/O performance, we should enable client write-back caching since most applications would not require `O_DIRECT` when output is written nonsequentially and benefit from write-back.
- To benchmark *system-level* random I/O performance, we should use `O_DIRECT` since it allows clients to force the file system to touch blocks in an unpredictable fashion. This approximates the aggregate workload of many users performing uncoordinated I/O across a large HPC system irrespective of their use of client write-back caching.

Random reads are more common among statistical analyses which sample large datasets such as AI training workloads [13], [31]. As demonstrated in Fig. 4 though, page cache has negligible effect on benchmarking random reads and the above corollaries need not be considered. However, hybrid SCM/QLC systems complicate performance analysis of real analysis workflows since the random read rate they deliver depends on the fraction of a dataset resides on the SCM layer.

Consider a workflow that copies a dataset from an external source to a high-performance file system before immediately reading it as part of an AI training step. Because the dataset is randomly read immediately after it has been copied to the file system, it would *not* have aged, and only a fraction of it will have been migrated to QLC and demonstrate the random read performance of that media. We approximate this workflow by generating a dataset through sequential writes, then reading that dataset randomly to quantify how the apparent random read performance varies with the size of the dataset. Because VAST does not provide any way to determine how files are split between SCM and QLC, we approximate different degrees of SCM/QLC occupancy by generating datasets with fixed sizes and assuming that larger fractions of them will be resident on QLC as they approach the 3 TB SCM buffer size. The results of this test are shown in Fig. 5.

Fig. 4 showed that this 64-process random read workload should achieve 52 KIOPS when 100% of the dataset resides on QLC. Fig. 5 shows that the actual read IOPS rate can be up to 3× faster if the dataset has not been fully migrated to QLC because SCM requires less concurrency to drive random read rates [26]. This highlights a fundamental challenge of setting expectations for workloads that perform random reads: the I/O performance an application should expect can vary by 3× based on the locality of the data being accessed. In turn, this locality can be a function of many complex factors including age, as discussed in Section III-B; in this specific case of an Optane and QLC hybrid storage solution, aged datasets deliver

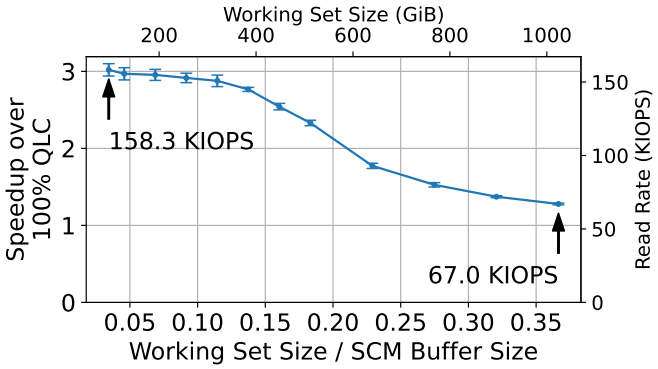

Fig. 5. Variation in observed random read IOPS as a function of the total dataset size being read. SCM buffer size is 3 TB, and a dataset that is 100% resident on QLC can achieve 52 KIOPS. Error bars signify one standard deviation calculated from five measurements.

higher read bandwidth at larger scales but lower read IOPS at smaller scales. Thus, as the data path includes more logic to facilitate transparent data placement, predicting performance becomes more difficult.

V. Conclusion

Hybrid SCM/QLC storage systems deliver new dimensions of performance beyond bandwidth-intensive checkpoint/restart workloads, but this flexibility comes with new challenges in performance analysis. Benchmarks must be designed to accurately reflect user behavior; for example, reading aged data results in higher sequential but lower random performance due to uneven SCM:QLC ratios. Combining SCM and QLC also provides more read than write bandwidth overall which suits analytics at scale, but traditional HPC workloads may require the write performance of a more SCM-rich design. We have also shown that VAST specifically delivers high performance despite its use of NFS, and NFS's lack of locality awareness actually allows applications to make better use write-back and readahead to deliver high bandwidth at all I/O sizes. Although these conclusions were drawn from VAST, it is likely that many also apply to similar hybrid SCM/QLC storage systems such as DAOS.

Acknowledgment

The authors would like to thank Sven Breuner, Jeff Denworth, and Howard Marks for key insights into the behavior of the VAST Universal Storage system and many staff at VAST Data for ongoing support and expertise. We also thank the anonymous reviewers for constructive feedback in improving the quality of this work. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-05CH11231. This research used resources and data generated from resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] M. L. Norman and A. Snavely, "Accelerating data-intensive science with Gordon and Dash," in *Proceedings of the 2010 TeraGrid Conference*, 2010, pp. 1–7. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1838574.1838588

[2] B. Hadri, S. Kortas, S. Feki, R. Khurram, and G. Newby, "Overview of the KAUST's Cray X40 System - Shaheen II," in *Proceedings of the 2015 Cray User Group*, 2015.

[3] W. Bhimji, D. Bard, M. Romanus, D. Paul, A. Ovsyannikov, B. Friesen, M. Bryson, J. Correa, G. K. Lockwood, V. Tsulaia, S. Byna, S. Farrell, D. Gursoy, C. S. Daley, V. Beckner, B. V. Straalen, D. Trebotich, C. Tull, G. Weber, N. J. Wright, K. Antypas, and Prabhat, "Accelerating Science with the NERSC Burst Buffer Early User Program," in *Proceedings of the 2016 Cray User Group*, London, 2016. [Online]. Available: https://cug.org/proceedings/cug2016_proceedings/includes/files/pap162.pdf

[4] K. S. Hemmert, M. W. Glass, S. D. Hammond, R. Hoekstra, M. Rajan, M. Vigil, D. Grunau, J. Lujan, D. Morton, H. A. Nam, P. Peltz, A. Torrez, C. Wright, and S. Dawson, "Trinity: Architecture and Early Experience," in *Proceedings of the 2017 Cray User Group*, 2017.

[5] D. Henseler, B. Landsteiner, D. Petesch, C. Wright, and N. J. Wright, "Architecture and Design of Cray DataWarp," in *Proceedings of the 2016 Cray User Group*, London, 2016. [Online]. Available: https://cug.org/proceedings/cug2016_proceedings/includes/files/pap105.pdf

[6] M. Richerson, "DataWarp Transparent Cache: Data Path Implementation," in *Proceedings of the 2018 Cray User Group*, 2018. [Online]. Available: https://cug.org/proceedings/cug2018_proceedings/includes/files/pap156s2-file1.pdf

[7] G. K. Lockwood, A. Chiusole, L. Gerhardt, K. Lozinskiy, D. Paul, and N. J. Wright, "Architecture and Performance of Perlmutter's 35 PB ClusterStor E1000 All-Flash File System," in *Proceedings of the 2021 Cray User Group*, 2021.

[8] Z. Liang, J. Lombardi, M. Chaarawi, and M. Hennecke, "DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory," in *Supercomputing Frontiers*, D. K. Panda, Ed. Cham: Springer International Publishing, 2020, pp. 40–54.

[9] "Universal Storage Explained," 2021. [Online]. Available: https://vastdata.com/whitepaper

[10] A. Dilger, D. Hildebrand, J. Kunkel, J. Lofstead, and G. Markomanolis, "IO500 ISC21 Lists," Jul. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5171694

[11] S. Saini, J. Rappleye, J. Chang, D. Barker, P. Mehrotra, and R. Biswas, "I/O performance characterization of Lustre and NASA applications on Pleiades," in *2012 19th International Conference on High Performance Computing*, 2012, pp. 1–10.

[12] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu, "On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, may 2016, pp. 750–759. [Online]. Available: http://ieeexplore.ieee.org/document/7516071/

[13] S. W. D. Chien, S. Markidis, C. P. Sishtla, L. Santos, P. Herman, S. Narasimhamurthy, and E. Laure, "Characterizing Deep-Learning I/O Workloads in TensorFlow," in *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*. IEEE, nov 2018, pp. 54–63. [Online]. Available: https://ieeexplore.ieee.org/document/8638422/

[14] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, P. Prabhat, and M. Houston, "Exascale Deep Learning for Climate Analytics," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, nov 2018, pp. 649–660. [Online]. Available: https://ieeexplore.ieee.org/document/8665799/

[15] K. A. Standish, T. M. Carland, G. K. Lockwood, W. Pfeiffer, M. Tatineni, C. C. Huang, S. Lamberth, Y. Cherkas, C. Brodmerkel, E. Jaeger, L. Smith, G. Rajagopal, M. E. Curran, and N. J. Schork, "Group-based variant calling leveraging next-generation supercomputing for large-scale whole-genome sequencing studies," *BMC Bioinformatics*, vol. 16, no. 1, p. 304, dec 2015. [Online]. Available: http://dx.doi.org/10.1186/s12859-015-0736-4http://www.biomedcentral.com/1471-2105/16/304

[16] W. D. Pence, L. Chiappetti, C. G. Page, R. A. Shaw, and E. Stobie, "Definition of the Flexible Image Transport System (FITS), version 3.0," *Astronomy & Astrophysics*, vol. 524, no. 10, p. A42, dec 2010. [Online]. Available: http://www.aanda.org/10.1051/0004-6361/201015362

[17] W. Bhimji, D. Bard, K. Burleigh, C. S. Daley, S. Farrell, M. Fasel, B. Friesen, L. Gerhardt, J. Liu, P. Nugent, D. Paul, J. Porter, and V. Tsulaia, "Extreme I/O on HPC for HEP using the Burst Buffer at NERSC," *Journal of Physics: Conference Series*, vol. 898, p. 082015, oct 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/898/8/082015

[18] N. Monnier, J. F. Lofstead, M. Lawson, and M. Curry, "Profiling Platform Storage Using IO500 and Mistral," in *IEEE/ACM Fourth International Parallel Data Systems Workshop, PDSW@SC 2019, Denver, CO, USA, November 18, 2019*. IEEE, 2019, pp. 60–73. [Online]. Available: https://doi.org/10.1109/PDSW49588.2019.00011

[19] T. Robinson, "Challenges in Providing an Interactive Service with Jupyter on Large-Scale HPC Systems," in *Proceedings of the 2019 Cray User Group*, Montreal, 2019. [Online]. Available: https://cug.org/proceedings/protected/cug2019{_}proceedings/includes/files/pres104s1.pdf

[20] T. Declerck, K. Antypas, D. Bard, W. Bhimji, S. Canon, S. Cholia, and Y. H. He, "Cori - A System to Support Data-Intensive Computing," in *Proceedings of the 2016 Cray User Group*, London, 2016. [Online]. Available: https://cug.org/proceedings/cug2016{_}proceedings/includes/files/pap171s2-file2.pdf

[21] A. Piccione, J. Berkery, S. Sabbagh, and Y. Andreopoulos, "Physics-guided machine learning approaches to predict the ideal stability properties of fusion plasmas," *Nuclear Fusion*, vol. 60, no. 4, p. 046033, mar 2020. [Online]. Available: https://doi.org/10.1088/1741-4326/ab7597

[22] A. Pau, A. Fanni, S. Carcangiu, B. Cannas, G. Sias, A. Murari, and F. R. and, "A machine learning approach based on generative topographic mapping for disruption prevention and avoidance at JET," *Nuclear Fusion*, vol. 59, no. 10, p. 106017, aug 2019. [Online]. Available: https://doi.org/10.1088/1741-4326/ab2ea9

[23] X. Huang, C. Storfer, A. Gu, V. Ravi, A. Pilon, W. Sheu, R. Venguswamy, S. Banka, A. Dey, M. Landriau, D. Lang, A. Meisner, J. Moustakas, A. D. Myers, R. Sajith, E. F. Schlafly, and D. J. Schlegel, "Discovering New Strong Gravitational Lenses in the DESI Legacy Imaging Surveys," *The Astrophysical Journal*, vol. 909, no. 1, p. 27, mar 2021. [Online]. Available: https://doi.org/10.3847/1538-4357/abd62b

[24] M. Arratia, "Jet-based TMD measurements with H1 data," in *Proceedings of the XXVIII International Workshop on Deep-Inelastic Scattering and Related Subjects*, Stony Brook, NY, 2021. [Online]. Available: https://www-h1.desy.de/h1/www/publications/htmlsplit/H1prelim-21-031.long.html

[25] D. Hildebrand and P. Honeyman, "Exporting storage systems in a scalable manner with pNFS," in *22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*, 2005, pp. 18–27.

[26] J. Yang, B. Li, and D. J. Lilja, "Exploring Performance Characteristics of the Optane 3D Xpoint Storage Technology," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 5, no. 1, pp. 1–28, feb 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3372783

[27] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "DiskReduce: Replication as a Prelude to Erasure Coding in Data-Intensive Scalable Computing," 2011.

[28] T. Patel, S. Byna, G. K. Lockwood, and D. Tiwari, "Revisiting I/O behavior in large-scale storage systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. New York, NY, USA: ACM, nov 2019, pp. 1–13. [Online]. Available: http://dl.acm.org/doi/10.1145/3295500.3356183

[29] G. K. Lockwood, K. Lozinskiy, L. Gerhardt, R. Cheema, D. Hazen, and N. J. Wright, "A Quantitative Approach to Architecting All-Flash Lustre File Systems," ser. Lecture Notes in Computer Science, M. Weiland, G. Juckeland, S. Alam, and H. Jagode, Eds. Cham: Springer International Publishing, 2019, vol. 11887, pp. 183–197. [Online]. Available: http://link.springer.com/10.1007/978-3-030-34356-9_16

[30] J. M. Kunkel, "Analyzing Data Properties Using Statistical Sampling Techniques – Illustrated on Scientific File Formats and Compression Features," in *Supercomputing Frontiers and Innovations*, 2016, vol. 3, no. 3, pp. 130–141. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46079-6_10

[31] S. W. D. Chien, A. Podobas, I. B. Peng, and S. Markidis, "tf-Darshan: Understanding Fine-grained I/O Performance in Machine Learning Workloads," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 2020, pp. 359–370.

APPENDIX

A. Test System Description

The VAST system used in this study was a standard VAST LightSpeed appliance in a "2x1" configuration with two "CBoxes" (8 CNodes) and one "DBox" (2 DNodes). Each VAST CNode is configured with

- 2x Intel Xeon Silver 4216 CPUs
- 8x 32 GiB DDR4-2400
- 1x 100G EDR InfiniBand to client fabric
- 4x 50G Ethernet to NVMe fabric

Each VAST DNode is configured with

- 2x Intel Xeon E5-2630 v4
- 8x 8 GiB DDR4-2133
- 4x 100G Ethernet to NVMe fabric

CNodes and DNodes were connected via the NVMe fabric comprised of two Mellanox SN2100 100 GbE switches used exclusively by VAST for NVMe I/O. The VAST cluster itself was running version 3.4.0-sp6-367205 which dictates the versions and configurations of its operating system.

All clients were Cray CS-Storm 500NX nodes each with

- 2x Intel Xeon Gold 6148 (Skylake) CPUs
- 8x NVIDIA Tesla V100 GPUs (not used here)
- 12x 32 GiB DDR4-2666
- 4x 100G EDR InfiniBand ports to VAST

Each client mounted VAST using the VAST NFS-over-RDMA driver version 3.7.7 and connected to all 32 virtual IPs (`remoteports`) using all four 100G EDR HCAs (`localports`) of the VAST cluster using `nconnect=32`. Clients were configured such that maximum RPC size for both reads and writes was 1048576 bytes, and the threshold to begin writing back dirty pages was 10% of system memory (`dirty_background_ratio=10`) and writes blocked when dirty pages exceeded 20% of system memory (`dirty_ratio=20`).

B. IOR Test Configurations

All experiments were carried out using two different versions of IOR. All bandwidth tests used a derivative of the IOR's development branch forked from commit `a436395` and extended to include a new data packet type with randomized (nominally incompressible) contents (hereafter "ior-incompress").[4] All IOPS tests used a derivative of IOR version 3.3.0 that incorporated one minor error-handling bug patch (hereafter "ior-3.3.0+6356464").[5]

Table I describes the arguments used for each test presented. Of note, we universally used

- `-C` - Ensure that each process does not read the same file that it wrote. Effectively pointless for these runs since we read and write in separate IOR jobs and dropped client caches at the start of each IOR invocation through Slurm.
- `-D 45` - Use stonewalling and stop issuing reads or writes I/O after 45 seconds.
- `-F` - Write and read from one file for each MPI process.
- `-e` - Call `fsync(2)` after the write phase to flush dirty pages and include this time in the I/O time. The combination of `-D 45` and `-e` results in actual write times taking longer due to the time required to flush dirty pages.
- `-g` - Use barriers between open, write, read, and close phases. This has no effect on performance measurements but was included to avoid potential issues that may have arisen from the loose metadata consistency semantics of NFS clients.

All bandwidth tests also used `-O stoneWallingWearOut=1` to require all read and write processes to move equal numbers of bytes and "catch up" to the fastest writer or reader before the bandwidth timer was stopped. `blockSize` (`-b`) and `transferSize` (`-t`) were varied, and `segmentCount` (`-s`) was defined so each process would attempt to read/write the lesser of 16 TiB or 2,147,483,647 segments in the absence of stonewalling (`-D`), and the exact transfer sizes (t) are described in Sec. II.

All non-naïve read tests were performed against pre-generated datasets generated using the parameters described in the "Generate" step in Table I. These datasets were generated by writing datasets using `-O stoneWallingWearOut=1` to limit the computational cost of this process while ensuring all files comprising each dataset had equal size and could be read without unexpected end-of-file errors during the subsequent read tests. Following the generate step, SCM was flushed during a "Pre-Read" step by writing and deleting 1 TiB of "random" data eight times in sequence using 1 MiB sequential transfers in 1 MiB blocks. Following generate and pre-read steps, read tests were performed on the generated dataset.

C. Performance Summaries

As described in Sec. II-B, each unique combination of N, p, t, and read/write was run five times to establish a basis for statistical significance. We report the standard deviation for these five measurements to this end, but we acknowledge that not all measurements to which this technique was applied are expected to be normally distributed.

These tests were all run on a non-production file system of which we were the exclusive user, and we observed that these measurements typically approached the "speed of light" of the underlying clients and/or storage system. This behavior, absent of the interference present on production systems, would be better modeled as a right-skewed gamma distribution than a normal distribution. However we lacked the resources to rigorously model the asymmetry of these performance distributions in this preliminary study and instead used the mean and standard deviation of all samples presented to approximate a 68% confidence interval. Thus, reproducing this study precisely requires that this approximation (that performance measurements are normally distributed in all samples) be made.

[4]Available online: https://github.com/glennklockwood/ior/commit/e1208a6f52e9946a073a1583154ca29dd1621903. Accessed April 9, 2021.

[5]Available online: https://github.com/glennklockwood/ior/commit/635464630c232ff17afa1fc04d47a88ec070a19c. Accessed May 17, 2021.

TABLE I
IOR ARGUMENTS USED

Dataset	IOR version	Step	Arguments
Bandwidth, Naïve	ior-incompress	Write	–stoneWallingWearOut=1 -C -D=45 -F -e -g -k -vv -w
Bandwidth, Naïve	ior-incompress	Read	–stoneWallingWearOut=1 -C -D=45 -F -e -g -r -vv
Bandwidth, Aged	ior-incompress	Write	–stoneWallingWearOut=1 -C -D=45 -F -e -g -k -l=random -vv -w
Bandwidth, Aged	ior-incompress	Read	–stoneWallingWearOut=1 -C -D=45 -F -e -g -l=random -r -vv
IOPS, Buffered I/O	ior-3.3.0+6356464	Write	-C -D=45 -F -e -g -vv -w -z
IOPS, Buffered I/O	ior-incompress	Generate	–stoneWallingWearOut=1 -D=600 -F -k -l=random -vv -w
IOPS, Buffered I/O	ior-3.3.0+6356464	Read	-C -D=45 -F -e -g -k -r -vv -z
IOPS, Direct I/O	ior-3.3.0+6356464	Write	–posix.odirect -C -D=45 -F -e -g -vv -w -z
IOPS, Direct I/O	ior-incompress	Generate	–stoneWallingWearOut=1 -D=600 -F -k -l=random -vv -w
IOPS, Direct I/O	ior-3.3.0+6356464	Read	–posix.odirect -C -D=45 -F -e -g -k -r -vv -z
IOPS vs. Size	ior-incompress	Generate	-C -D=300 -F -k -l=random -vv -w
IOPS vs. Size	ior-3.3.0+6356464	Read	-C -D=45 -F -g -r -vv -z
All Non-Naïve	ior-incompress	Pre-Read	-C -F -e -g -l=random -vv -w