Architecture and Performance of Perlmutter's 35 PB ClusterStor E1000 All-Flash File System

Glenn K. Lockwood, Alberto Chiusole, Lisa Gerhardt, Kirill Lozinskiy, David Paul, Nicholas J. Wright
Lawrence Berkeley National Laboratory
{glock, chiusole, lgerhardt, klozinskiy, dpaul, njwright}@lbl.gov

***Abstract*—NERSC's newest system, Perlmutter, features a 35 PB all-flash Lustre file system built on HPE Cray ClusterStor E1000. We present its architecture, early performance figures, and performance considerations unique to this architecture. We demonstrate the performance of E1000 OSSes through low-level Lustre tests that achieve over 90% of the theoretical bandwidth of the SSDs at the OST and LNet levels. We also show end-to-end performance for both traditional dimensions of I/O performance (peak bulk-synchronous bandwidth) and non-optimal workloads endemic to production computing (small, incoherent I/Os at random offsets) and compare them to NERSC's previous system, Cori, to illustrate that Perlmutter achieves the performance of a burst buffer and the resilience of a scratch file system. Finally, we discuss performance considerations unique to all-flash Lustre and present ways in which users and HPC facilities can adjust their I/O patterns and operations to make optimal use of such architectures.**

I. Introduction

The Perlmutter supercomputer deployed at NERSC is an HPE Cray EX system with performance that is 3-4× higher than NERSC's current-generation HPC system, Cori. In addition to the extreme computational performance afforded by over 6,000 NVIDIA A100 GPUs and 7,500 AMD Epyc 7763 CPUs, Perlmutter will also deliver extreme I/O performance through its 35 PB all-NVMe flash-based Lustre file system. This file system was designed to combine the performance of a burst buffer with the stability and ease-of-use of a traditional disk-based parallel file system. Once fully configured, it will be capable of delivering terabytes per second of bandwidth while providing sufficient capacity for users to retain their data for periods of weeks [1].

Although the Perlmutter file system delivers extreme I/O performance in the conventional senses of peak bandwidth, the all-NVMe design of the file system opens up new dimensions of performance as well. As is experienced with many HPC data centers, NERSC's I/O demands come from a combination of large I/Os and large files and small I/Os and small files [2], [3], [4], [1]. Designing a file system on all flash allows us to address the latter demands, which have historically performed poorly on all-hard disk drive (HDD) file systems, since solid-state drives (SSDs) can place data blocks in arbitrary locations with extremely low latency by virtue of having no moving parts. Thus, Perlmutter scratch represents one of the first extreme-scale parallel file systems that can effectively serve the I/O needs of both workloads optimized for traditional HPC and workloads that have historically performed poorly in HPC environments.

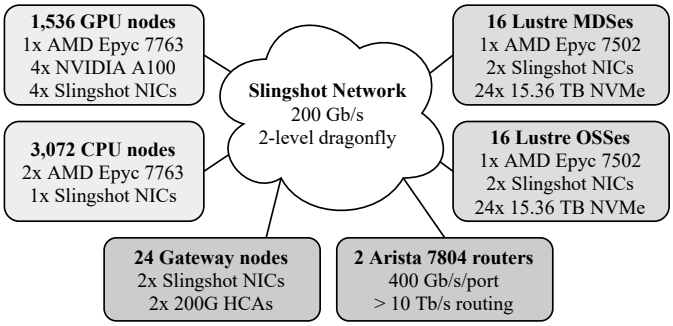

Fig. 1. Composition of the compute and I/O subsystems of the Perlmutter supercomputer.

This new dimension of performance afforded by all-flash also introduces new concerns around the efficiency of the hardware and software systems that reside between the raw solid-state disks (SSDs) and user applications. Any bottlenecks or inefficiencies in this data path can have dramatic impacts on the overall performance of the file system, since extreme-performance flash requires every other component to have commensurately extreme performance. As a result, the hardware architecture of the storage servers, their integration with the networks over which compute nodes and storage communicate, and the parallel file system software atop this hardware may require more consideration than previous HDD-based file systems.

In this work, we present the architecture of the Perlmutter file system at all of these levels and consider performance in terms of both bandwidth and IOPS. In addition, we examine the *performance efficiency* of the hardware and software of the file system to determine how much of the raw NVMe performance is lost at its different levels. To understand how much of an improvement in performance users may expect to experience in going from the previous-generation hybrid HDD+SSD storage architectures to all-flash, we also compare the performance of Perlmutter scratch servers to that of NERSC's previous system, Cori. Finally, we discuss several unexpected performance effects that are unique to all-flash file systems.

II. Architecture

The Perlmutter supercomputer is based on the Cray EX platform and is built around a 200 Gb/s Slingshot high-speed

network connected in a two-level dragonfly topology. Compute nodes and storage nodes are both directly integrated into this network as shown in Figure 1. This high-speed network is connected to NERSC's Ethernet networks through Arista 7804 routers that are directly connected to Slingshot switches; this enables high-bandwidth connectivity between compute nodes and external data sources such as NERSC's 200 PB HPSS tape archive, Internet-connected experimental facilities, and the cloud. In addition, dedicated gateway nodes enable Perlmutter to mount NERSC's 128 PB center-wide Community File System over InfiniBand. The remainder of this paper focuses on the high-performance file system integrated into Perlmutter.

A. Storage Subsystem Architecture

The Perlmutter file system, hereafter referred to as *Perlmutter scratch*, provides over 35 PB (35×10^{15} bytes) of usable capacity and is composed entirely of NVMe SSDs; it contains zero HDDs. It is a scratch file system designed to deliver the highest performance for I/O-intensive parallel applications running on the Perlmutter system and it is *not* intended to be a center-wide resource for long-lived data. To this end, NERSC opted to deploy an all-NVMe file system instead of a hybrid flash-HDD since the added cost of HDDs would reduce the amount of SSDs (and therefore total performance) the file system could deliver for a fixed cost.

Perlmutter scratch uses the Lustre file system [5] and is comprised of 16 metadata servers (MDSes), 274 object storage servers (OSSes), and a total of 3,480 NVMe drives. The file system is directly integrated on to the same Slingshot network as the compute nodes so there are no LNet routers in the entire Perlmutter system. However, all Lustre servers are segregated into distinct *I/O groups* on the dragonfly network which allows the file system to remain up and accessible to users even when the compute node subsystem is powered off for maintenance. This network design also allows the I/O subsystem to achieve the full cross-sectional bandwidth and network path diversity of the dragonfly network between compute nodes and Lustre servers, providing optimal performance, resilience, and availability.

Perlmutter has a total of four I/O groups on the dragonfly network, and each I/O group is directly connected to:

- each compute group via 4× 200 Gb/s global links;
- each other I/O group via 6× 200 Gb/s global links;
- the Perlmutter's service group via 6× 200 Gb/s global links.

The Perlmutter service group contains user login nodes, system management nodes, and the Arista routers.

Each of the four I/O groups is comprised of four racks, and each rack is comprised of approximately ten HPE Cray ClusterStor E1000 enclosures, two gateway nodes, and four 64-port Slingshot switches as shown in Figure 2.

Lustre OSS and MDS nodes are hosted within ClusterStor E1000 enclosures which are optimized for reliability and contain no single points of failure. Each E1000 enclosure contains two server canisters, two power supplies, redundant fans and fan controllers, and the necessary infrastructure to support heartbeating and failover between its two servers. Each E1000 also contains 24 front-loaded, hot-swappable U.2 NVMe SSDs which are dual-ported such that both servers in the E1000 are connected to all 24 NVMe drives to further facilitate failover. All E1000 enclosures in Perlmutter scratch use 15.36 TB Samsung PM1733 NVMe drives.

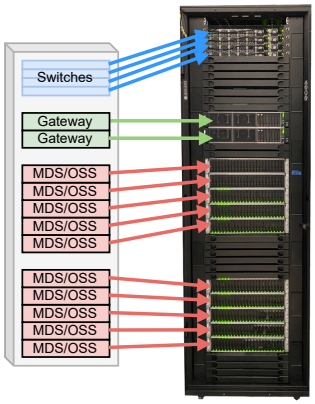

Fig. 2. Composition of a typical rack of Perlmutter scratch. The number of nodes in each rack differs slightly.

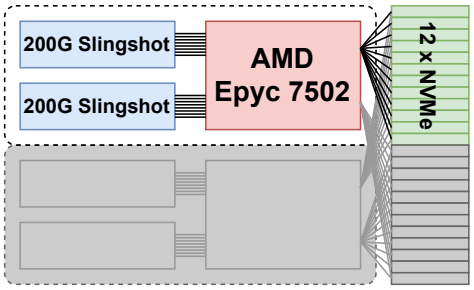

Fig. 3. Composition of a single HPE Cray ClusterStor E1000 enclosure with one of two servers highlighted. Servers act as active/active high-availability pairs, each responsible for half of the enclosure's NVMe drives during normal operation.

B. Node Architecture

Whereas the E1000 enclosures that comprise Perlmutter scratch are designed for resilience, the two servers within each enclosure are architected for performance. As depicted in Figure 3, each server is built around a single-socket AMD Epyc 7502 CPU with 128 lanes of PCIe Gen4 and eight channels of DDR4-3200 DRAM. 48 PCIe lanes are used to connect each of the enclosure's 24 NVMe drives via Gen4 x2 links, and 32 lanes are used to connect two 200 Gb/s Slingshot network adapters. By using AMD-based servers with PCIe Gen4, Perlmutter's Lustre OSSes and MDSes were able to employ a switchless PCIe topology and non-blocking PCIe bandwidth between NVMe drives and NICs. In addition, the single-socket configuration reduces the complexity of the NUMA topology within each server.

Each OSS and MDS in Perlmutter scratch nominally hosts a single Lustre Object Storage Target (OST) or Metadata Target (MDT) comprised of 12 NVMe drives. Both OSTs

and MDTs use ldiskfs instead of ZFS in Perlmutter because ldiskfs delivers significantly higher bandwidth and IOPS over ZFS and Perlmutter scratch was designed principally to deliver the highest possible performance. OSTs use parity-declustered RAID6 implemented through GridRAID [6] in an 8+2 configuration with one distributed spare to best balance NVMe capacity, bandwidth, and fault tolerance. MDTs are configured as 11-way RAID10 arrays to best balance NVMe IOPS and fault tolerance. All E1000 enclosures act as active/active failover pairs so that if one of the OSSes or MDSes within an E1000 enclosure fails, its failover partner will serve both OSTs or MDTs. As a result, both servers in a single E1000 must fulfill the same role as either OSSes or MDSes.

III. Methods

We used the IOR benchmark [7] version 3.3.0 to measure the bandwidth and IOPS of Perlmutter scratch for reads and writes. The tests were configured as follows; in all cases, the files read and written used a stripe width of 1 and stripe size of 1 MiB, and all files were restricted to a single OST. All I/O was also buffered; the `O_DIRECT` option was not used in any tests. In the remainder of this paper, we define 1 GB as 10^9 bytes, 1 GiB as 2^{30} bytes, and 8 Gb as 1 GB.

Write bandwidth was measured by creating and writing one file per MPI process using contiguous 64 MiB (64×2^{20} bytes) transfers for 45 seconds. After 45 seconds of I/O, the largest amount of I/O written by a single rank was determined (B_{max}), and all other ranks were forced to "catch up" so that they all wrote the same number of bytes to their respective files, resulting in a total I/O volume of $N_{\text{ranks}} \times B_{\text{max}}$. IOR refers to this form of test as "stonewalling with wear-out." The total bandwidth is calculated as $N_{\text{ranks}} \times B_{\text{max}}/t_{\text{max}}$, where t_{max} is the time taken by the slowest rank to write and fsync B_{max} bytes; t_{max} does not include the time required to create, open, or close files. This test was constructed to reflect the I/O of a coherent checkpoint operation where a parallel application cannot proceed until the slowest rank completes all of its I/O.

Read bandwidth was measured by first generating a multi-file dataset and using stonewalling with wear-out to ensure that all files created had identical sizes. We ensured that this dataset was at least twice as large as the total DRAM on the Lustre OSSes to which the data was written to avoid the effects of server-side read caching, and all client caches were flushed after this dataset generation phase. This dataset was then read using one file per process using contiguous 1 MiB transfers for at least 45 seconds. Stonewalling wear-out was not used for this test, as our goal was to determine the peak *aggregate* read bandwidth of a single OSS and OST rather than try to emulate any specific application workload.

Write IOPS were measured by creating and writing one file per MPI process using 4 KiB transfers at random offsets for 45 seconds. Stonewalling wear-out was not applied for this test since a bulk-synchronous yet completely random write workload does not represent a real workload of interest to NERSC; rather, we aimed to test the peak capability of the OSS and OST under the intense workload of hundreds of users attempting to write to different locations throughout the file system at once. As with the write bandwidth test though, we use the time taken by the slowest rank to pass the 45 second limit and fsync its file (t_{max}) as the denominator in our IOPS calculation.

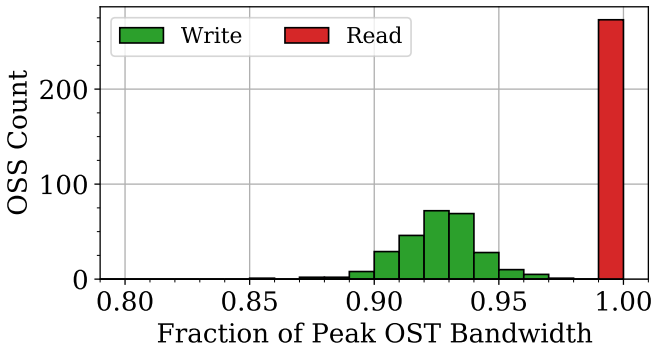

Fig. 4. Distribution of bandwidths measured using obdfilter-survey for 273 Perlmutter OSTs. One OST was undergoing hardware maintenance at the time of testing and is not represented.

Read IOPS were measured by first generating a dataset as was done for the read bandwidth test using one file per process. We then wrote arbitrary data to the same OSTs as our dataset for 45 seconds to ensure that none of the target dataset remained resident in OSS DRAM cache, and confirmed that the data volume written during this cache-flush stage was greater than twice the DRAM of the OSSes. We then read our dataset using one file per process and 4 KiB transfers at random offsets for 300 seconds. As with the write IOPS test, we did not use stonewalling wear-out because our goal was to emulate the behavior of hundreds of users reading files located randomly across the file system rather than a single bulk-synchronous random read workload.

In addition to these IOR tests, we also performed tests using the obdfilter-survey [8] and LNet Self-Test [9]. obdfilter-survey was run on each OST using 1 object, 1024 threads, 4 MiB records, and 512 MiB total size. LNet Self-Test was run against each OSS tested as a sink from two sources using eight concurrent requests and 1 MiB I/O sizes in bulk read-write mode. These tests were used to qualify the RAID subsystem and network performance of each Lustre OSS, respectively.

IV. Performance

At the time of writing, Perlmutter scratch was still being tuned for full scale and not yet in a production configuration. Rather than publish full-scale performance measurements that we know to be suboptimal, we choose to defer publication of Perlmutter's scale performance and instead focus on single-server performance in this paper. Thus, the findings presented in this section reflect the capabilities of the all-NVMe building blocks from which Perlmutter scratch is built.

A. Intra-OSS Performance Efficiency

The obdfilter-survey tool is routinely used to identify faulty storage hardware since it tests the performance of storage

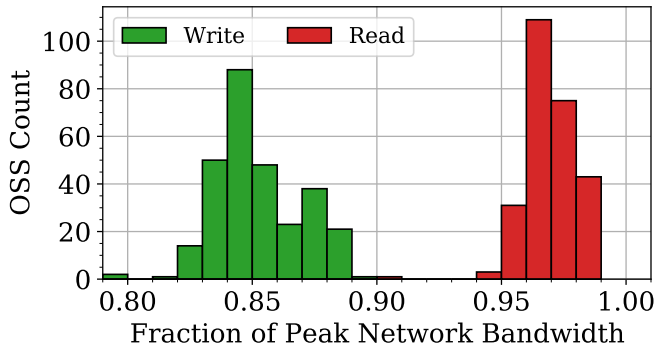

Fig. 5. Distribution of bandwidths measured using LNet Self-Test for 271 Perlmutter OSSes. Three OSSes undergoing maintenance were not included in this test.

volume underlying the OST in the absence of effects caused by the Lustre client or network stack. Because it tests the storage media in relative isolation, we can use it to also measure the *performance efficiency* of each OST by normalizing the obdfilter-survey bandwidth to the aggregate performance of the SSDs; this effectively quantifies the overheads of the RAID subsystem and the parts of Lustre which interact directly with it. The Samsung PM1733 NVMe SSDs used in Perlmutter scratch are connected to their host OSSes using two PCIe Gen4 lanes, and each SSD is specified to deliver up to 3.5 GB/s for reads and 3.2 GB/s for writes[1]. Given that there are twelve SSDs per OST, the peak read performance of an OST should be 42 GB/s. Because two parity blocks must be synchronously written for every eight data blocks written, the peak write performance of an OST should be 80% of 38.4 GB/s, or 30.7 GB/s.

Figure 4 shows the distribution of read and write bandwidth efficiencies measured using obdfilter-survey and demonstrates that Perlmutter's OSTs are extremely efficient at delivering bandwidth. Virtually all OSTs are capable of delivering over 90% of the bandwidth advertised by the underlying PM1733 SSDs; the median efficiency at the OST level is 92.6% for writes and 99.9% for reads. While this latter figure seems implausibly high, we have found that individual Samsung PM1733 SSDs are able to exceed their read bandwidth specification which suggests that the advertised device peak is an underestimate of the true hardware capability.

These results demonstrate that the combined overheads of ldiskfs-based OSTs and declustered RAID6 are not significant and still allow the Perlmutter OSTs to deliver a high fraction of the raw SSD bandwidth. In addition, this shows that the AMD 7502 processor in each OSS is sufficiently capable of driving the GridRAID data protection scheme such that it is able to distribute data across SSDs, compute parity, and update file system and RAID data structures without greatly impacting streaming bandwidth.

[1]Because Perlmutter uses only two PCIe Gen4 lanes to connect each SSD to its active OSS, each SSD's performance is effectively equivalent to the PM1733 specification for four lanes of PCIe Gen3.

TABLE I
CLIENT PERFORMANCE MEASURED WITH IOR

Metric	Write	Read
Bandwidth	27.1 GB/s	40.8 GB/s
IOPS	28.8 KIOPS	1452.1 KIOPS
Bandwidth efficiency	88.4%	97.2%
IOPS efficiency	5.33%	15.1%

We also examine the efficiency of the Lustre networking layer, LNet, using the LNet Self-Test tool provided with Lustre. The Slingshot NICs used in Perlmutter scratch are each capable of 200 Gb/s for a total of 400 Gb/s (50 GB/s) of peak network bandwidth per OSS using Lustre's multi-rail capabilities. As shown in Figure 5, the NICs, LNet, and multi-rail are also very efficient; the median efficiency across all OSSes was 97.0% for reads (egress) and 84.8% for writes (ingress). We also note that these LNet Self-Tests used the same two source OSSes to test each sink OSS. As such, the global links between the four I/O groups in Perlmutter were exercised for ¾ of the OSSes tested, and we attribute the broader distribution of bandwidths in Figure 5 (relative to Figure 4) to this.

To contextualize these LNet performance measurements to peak SSD bandwidth, this demonstrates that OSSes can deliver 38.7% and 15.6% greater LNet bandwidth than peak SSD bandwidth on average. Thus, the bandwidth of these all-NVMe OSTs are limited by OST performance rather than network performance, and OST performance is over 90% that of the raw SSD performance. This observation confirms the design goal that Perlmutter's OSSes maximize performance; there are no bottlenecks within the servers that severely constrain the bandwidth of the NVMe drives.

B. End-to-end Performance

Section IV-A demonstrated that individual OSSes are performance-efficient, and it follows that a single OSS should be able to deliver a significant fraction of this performance to Lustre clients through the other components of the Lustre data path. Table I shows the peak performance measured from Lustre clients on Perlmutter to a single OST using the IOR tests described in Section III. These performance metrics were all measured from the same single OST, and the specific OST tested was chosen at random; as a result, these measurements most likely correspond to the the median performance measurements shown in Figures 4 and 5.

Following our definition of performance efficiency from Section IV-A, Table I shows that Lustre clients are able to draw over 85% of the bandwidth of the SSDs which is a testament to Lustre's capabilities as an all-flash file system. Conversely, Lustre is not able to deliver similarly high fractions of peak SSD IOPS capability. Intuitively, we attribute this to the fact that the SSD drive specifications are measured at the raw block device level using the fio benchmark[2] but IOR tests at the file level. The file system introduces significant latency since it requires copying data to and from the kernel on the client side,

[2]*Flexible I/O Tester*. https://github.com/axboe/fio. Accessed June 10, 2021.

TABLE II
CORI STORAGE SYSTEMS

	Scratch	Burst Buffer
File System	Lustre	DataWarp
Platform	Cray ClusterStor 9000	Cray XC-40
Data servers	248	288
Drive count	41 HDDs/server	4 SSDs/server
Drive model	Seagate ST4000NM0034	Intel P3608

and file I/O must also transit the OST subsystem (including ldiskfs and the GridRAID software stack) before reaching the block level at which the SSD drive specification was measured. The Lustre client and server are also separated by the Slingshot network which requires traversing one global link between compute group and I/O groups, but this 2 μs latency [10] is at least an order of magnitude smaller than the random access latency of Perlmutter's PM1733 SSDs. In addition, the IOR IOPS tests described in Section III are designed to saturate the OSS by employing the optimal number of clients and IOR processes, masking the effects of network latency on any given client.

The 3:1 asymmetry between read and write IOPS efficiency is the result of our choice to use RAID6. While our 8+2 configuration results in a 20% penalty to streaming bandwidth as discussed in Section IV-A, this data protection scheme incurs a 3x penalty for write IOPS at minimum. Every 4 KiB write requires its 128 KiB RAID block to be read, modified, and synchronously written back along with two 128 KiB parity blocks. This effectively triples the number of write operations that must complete synchronously for each random write generated by IOR. By comparison, there is no analogous I/O amplification penalty for random reads since Perlmutter scratch is not configured to check parity on reads.

C. Performance vs. Cori

NERSC chose to pursue an all-flash file system to reduce the complexity of having separate scratch and burst buffer tiers. As such, the Perlmutter file system was designed to deliver performance commensurate to a burst buffer and provide capacity commensurate to a disk-based performance tier. To assess the veracity of this goal, we compare the performance of a single OSS presented in Section IV-B to the performance of NERSC's previous-generation system, Cori.

The Cori file systems targeted by this comparison are summarized in Table II, and their respective architectures are detailed in previous works [11], [12]. In addition, the performance figures we cite for Cori come from full-system I/O tests performed when these storage systems were being first deployed in 2015 and 2016. Those IOR tests were configured to deliver the peak performance of their respective storage systems and did not use the parameters described in Section III. We consider this comparison reasonable despite the difference in test configuration since both Cori and Perlmutter IOR tests were designed to deliver optimal performance on their respective platforms. That said, it is important to recognize that the test conditions between Cori and Perlmutter are not identical in the following discussion.

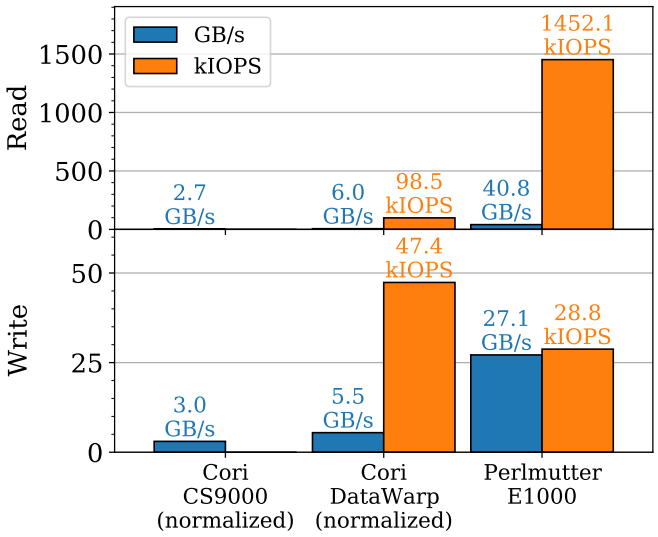

Fig. 6. Single-server performance of a Perlmutter E1000 OSS, Cori ClusterStor 9000 OSSes, and Cori DataWarp burst buffer nodes. Perlmutter performance was directly measured from a single server, but Cori performance is normalized to a single server from full-system runs (248 ClusterStor 9000 OSSes and 288 DataWarp nodes).

Figure 6 shows the results of a single Perlmutter OSS as presented in Section IV-B compared to Cori's ClusterStor 9000-based scratch and DataWarp-based burst buffer. We normalize Cori's data to a single server to facilitate comparison with the Perlmutter data and demonstrate the capability of a single parallel building block of each storage system. Furthermore, since Perlmutter scratch has roughly the same total storage servers as Cori (274 Perlmutter OSSes vs. 248 Cori OSSes and 288 burst buffer nodes), this comparison also offers a rough qualitative comparison of the systems' overall capabilities.

As expected, the bandwidth of a single Perlmutter OSS is significantly higher than Cori scratch OSSes on the basis that 12 SSDs provide much higher bandwidth than 41 HDDs. Relative to Cori's burst buffer, the per-server bandwidth of Perlmutter is also significantly higher as a result of (1) each Perlmutter OSS having three times as many NVMe drives as a single Cori burst buffer server, and (2) Perlmutter's PM1733 drives using a newer and higher-performance NVMe controller and NAND configuration relative to Cori's Intel P3608 NVMe drives. While Perlmutter does use PCIe Gen4 for NVMe connectivity, it only uses two lanes to connect to its OSS whereas Cori's burst buffer uses four lanes of PCIe Gen3. Thus, the effective PCIe bandwidth per-drive on Perlmutter is the same as Cori's burst buffer, and Perlmutter's PCIe Gen4 does not give a significant performance uplift over Cori in this regard.

The peak IOPS of Cori scratch was never measured because HDDs are known to perform poorly for random workloads[3], so we do not compare the random I/O performance of Perlmutter

[3]The data sheet for Cori scratch's ST4000NM0034 HDDs does not include an IOPS specification, but we estimate a ceiling of 240 IOPS per drive (9.8 kIOPS/OST for reads) based on their 4.16 ms average latency specification.

to Cori scratch. Perlmutter's random read performance is significantly higher than the burst buffer though, and we attribute this to newer drive architecture and larger number of SSDs per server. Random read performance is especially crucial for the NERSC workload because it resembles the aggregate workload of many users performing many common operations including `ls -l` (gathering file sizes from OSTs) and reading many small files (such as configuration files and Python libraries). Many of these intense small-read workloads are performed interactively by users and result in perceptible latency or lag while operating in the terminal or running a Python script. In turn, this interactive lag adversely impacts users' perceptions of how "fast" the file system is. Since the Perlmutter OSSes are so much more capable at serving intensive bursts of small reads at random locations, we anticipate the file system to feel much more responsive to these users.

The random write performance of a Perlmutter OSS is not as high as a DataWarp server, but this is unsurprising given the steep random write performance penalty incurred by Perlmutter's RAID6 configuration discussed in Section IV-A. By comparison, Cori's burst buffer stripes data across all SSDs without data protection, allowing applications to access the full random write performance of its SSDs without the overheads of parity updates or read-modify-write. This represents a tradeoff of using an all-flash scratch versus a burst buffer: Perlmutter scratch delivers less write IOPS than Cori's burst buffer, but it is also resilient to drive failures and server crashes. Whereas single SSD or server failure on the burst buffer will cause data loss, downtime, and the failure of running jobs, Perlmutter can tolerate such failures while remaining fully available.

It is important to stress that the comparisons in Figure 6 have large margins because the Cori measurements include the effects of scaling while the Perlmutter measurements do not. Scaling is undeniably critical to parallel I/O performance, but Perlmutter clients were not appropriately tuned at the time of testing. As a result, this discussion is merely indicative of order-of-magnitude differences between Cori and Perlmutter, and the Perlmutter numbers reflect an optimistic upper limit on the performance that may be expected at scale.

V. Considerations for All-NVMe Lustre

In many regards, understanding the performance of all-flash Lustre is straightforward since the performance of SSDs is superior to HDDs in every dimension. However, several unique performance considerations have arisen over the course of deploying Perlmutter scratch and transitioning from HDD to all-flash.

A. Asymmetric I/O Variation

Figure 4 highlights that read and write bandwidth on flash OSTs are very asymmetric, and this is to be expected since (a) writes incur a 20% overhead due to RAID6 and (b) SSDs can deliver more read bandwidth than write bandwidth (see Section IV-A). However these obdfilter-survey results also show that the *variation* in bandwidth is higher for writes than for reads. This is an important consideration in the context of bulk-synchronous parallel I/O operations which are limited by the slowest OST.

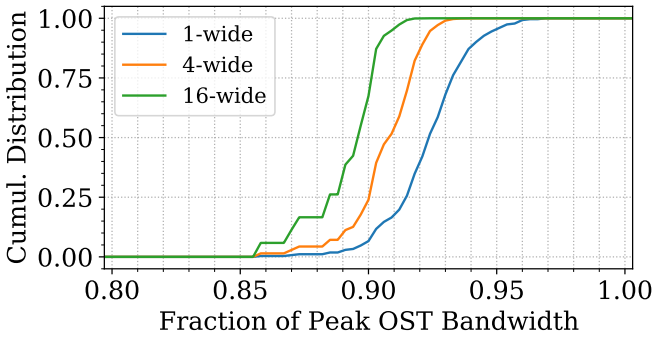

Fig. 7. Cumulative distribution function of probability that a file of given stripe width will be limited by the performance of a slow OST exhibiting the given fraction of peak write bandwidth.

If a file is striped over a single OST, the probability of its performance being any given fraction of peak is described by the distribution in Figure 4. However with wider stripes, the odds of a file stripe being placed on a "slow" OST increases as illustrated in Figure 7. For example, we observed eighteen OSTs whose write performance was 90% or worse than the peak observed value. If a file is striped over a single OST, the probability of that file being placed on such a slow OST is approximately 6.6%. However that probability increases to 24% for files striped across four OSTs and to 67% for files striped across sixteen. This suggests that there is a tradeoff for bulk-synchronous write-heavy workloads: users should stripe wide enough to achieve high peak bandwidth but not so wide as to unduly increase the likelihood that a file will be affected by a straggling OST.

The data shown in Figure 7 is derived from a snapshot in time of obdfilter-survey results and should not be generalized to suggest that files striped across four OSTs will always have a 24% chance of achieving below 90% of peak. Many factors contribute to straggling OSTs including hardware issues and contention between users. We present these data to highlight that performance variation intrinsic to all-flash OSTs are yet another factor to consider when users determine their optimal striping for write-intensive workloads. As shown in Figure 4, there is no analogous wide variation in read bandwidth intrinsic to flash; this is a favorable finding for Perlmutter since the majority of NERSC users' I/O to scratch is read-intensive [13].

B. Performance over Time

Storage drives have historically lost performance as more data is written to them. In the context of HDDs, this is largely due to disk fragmentation and the penalties associated with sequential writes and reads requiring an increasing number of physical seeks [14]. Defragmentation does remedy this age-related performance loss, but the high cost of periodic defragmentation on a large Lustre file system has rendered this process impractical, and NERSC has accepted this per-

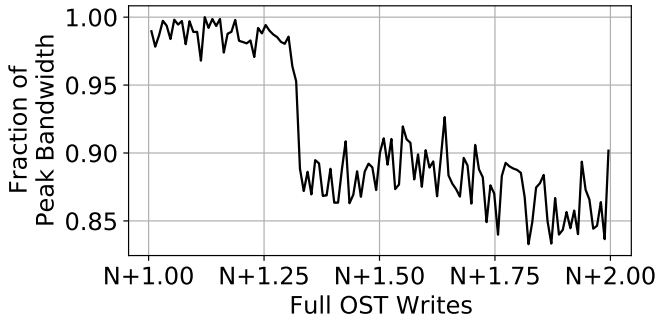

Fig. 8. Relative write bandwidth with increasing number of bytes written. One OST stores 133.3 TB.

formance loss as unavoidable on its Cori scratch file system. Although SSDs do not have such seek penalties, they do require periodic garbage collection as erase blocks become sparse, and this results in a distinct loss of bandwidth once an SSD becomes sufficiently "dirty" [15]. I/O separation has been shown to mitigate this age-related performance loss on parallel file systems [16], but such techniques require hardware support implemented in the file system, and Lustre does not yet support any such capabilities.

We have purposely triggered this age-related performance degradation on Perlmutter scratch to understand its impact. As shown in Figure 8, there is a $\approx$ 10% loss of write bandwidth once an OST experienced enough writes and deletes, and the transition to this age-induced degraded performance happens very quickly. N was not carefully documented for the experiment used to generate Figure 8 because we did not track the total bytes written to the OST prior to the test. However, other tests suggest N $\approx$ 4 for Perlmutter's E1000-based OSTs, and performance loss begins after approximately 5 full OSTs worth of data ($\approx$ 665 TB) have been written and erased.

Fortunately, Lustre can recover from this age-related performance loss by issuing TRIM commands to the underlying SSDs, and this has been shown to fully restore the performance of an OST once completed [17]. Unlike HDDs though, this trim process takes hours rather than days and effectively returns the drive to like-new performance, providing another $\approx$ 5 full OST writes before the next onset of performance loss. Given that Perlmutter is designed to support between 2,200 TB and 2,900 TB of writes distributed over 274 OSTs per day [1], we expect 6% - 8% of each OST's capacity to be written daily, and we expect to reach this 5 full OST writes watermark every 60 - 80 days. Assuming the aggregate user workload ages Perlmutter's OSTs at a rate faster than our synthetic testing, we plan to trim Perlmutter scratch every month to maintain optimal write bandwidth.

VI. Conclusion

We have presented the architecture of the Perlmutter scratch all-flash Lustre file system and how this system has been architected for extreme performance through direct integration on to Perlmutter's Slingshot fabric and a well-balanced data path within each Lustre OSS. We demonstrated that its HPE Cray ClusterStor E1000 building blocks can deliver over 85% of the NVMe bandwidth and 5% - 15% of the NVMe IOPS to Lustre clients. As a result, we have successfully positioned Perlmutter scratch as having burst buffer-level performance but file system-like resilience, effectively collapsing the two tiers into one while keeping the best qualities of each. Finally, we have identified that all-flash OSTs are susceptible to wider variation in write performance and performance loss due to age, but judicious striping and periodic trimming are sufficient to mitigate these peculiarities.

Acknowledgment

The authors would like to thank John Fragalla, Jeff Hudson, Peter Bojanic, Cory Spitz, and the HPE Cray ClusterStor engineering team for their insights in designing, evaluating, and deploying this platform.

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] 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," in *High Performance Computing*, M. Weiland, G. Juckeland, S. Alam, and H. Jagode, Eds. Cham: Springer International Publishing, 2019, pp. 183–197.

[2] A. Uselton and N. J. Wright, "A File System Utilization Metric for I / O Characterization," in *Proceedings of the 2013 Cray User Group*, Napa, CA, 2013. [Online]. Available: https://cug.org/proceedings/cug2013_proceedings/by_auth.html

[3] F. Wang, H. Sim, C. Harr, and S. Oral, "Diving into petascale production file systems through large scale profiling and analysis," in *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems - PDSW-DISCS '17*. New York, New York, USA: ACM Press, 2017, pp. 37–42. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3149393.3149399

[4] S. S. Vazhkudai, R. Miller, D. Tiwari, C. Zimmer, F. Wang, S. Oral, R. Gunasekaran, and D. Steinert, "GUIDE: A Scalable Information Directory Service to Collect, Federate, and Analyze Logs for Operational Insights into a Leadership HPC Facility," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '17*, pp. 1–12, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3126908.3126946

[5] P. Schwan, "Lustre: Building a File System for 1,000-node Clusters," *Proceedings of the Linux Symposium*, pp. 401–409, 2003.

[6] M. Swan, "Sonexion GridRAID Characteristics," in *Proceedings of the 2014 Cray User Group*, Lugano, may 2014.

[7] J. Kunkel, G. K. Lockwood, C. J. Morrone, M. Chaarawi, J.-Y. Vet, S. Snyder, R. Latham, A. Jackson, J. Inman, B. Kettering, E. Zickler, A. Huebl, M. Nelson, N. Hjelm, J. Schwartz, O. Tatebe, V. Leung, B. Crossman, A. Dilger, S. Didelot, A. Moody, A. Torrez, J. Bent, O. Steffen, P. Koutoupis, S. Breuner, V. Hapla, A. Hück, F. Gadban, and G. Zheng, "hpc/ior: IOR version 3.3.0," dec 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4391430

[8] "Chapter 33. Benchmarking Lustre File System Performance (Lustre IO Kit)," in *Lustre Software Release 2.x Operations Manual*. [Online]. Available: https://doc.lustre.org/lustre_manual.xhtml#benchmark.ost_perf

[9] "Chapter 32. Testing Lustre Network Performance (LNet SelfTest)," in *Lustre Software Release 2.x Operations Manual*. [Online]. Available: https://doc.lustre.org/lustre_manual.xhtml#lnetselftest

[10] D. De Sensi, S. Di Girolamo, K. H. McMahon, D. Roweth, and T. Hoefler, "An in-depth analysis of the slingshot interconnect," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20. IEEE Press, 2020.

[11] 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

[12] 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

[13] 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

[14] J. Kaitschuck, "The Effects of Fragmentation and Capacity on Lustre File System Performance," in *Proceedings of the 2017 Lustre User Group*, Bloomington, IN, 2017.

[15] J. Han, D. Koo, G. K. Lockwood, J. Lee, H. Eom, and S. Hwang, "Accelerating a Burst Buffer via User-Level I/O Isolation," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, pp. 245–255.

[16] D. Koo, J. Lee, J. Liu, E.-K. Byun, J.-H. Kwak, G. K. Lockwood, S. Hwang, K. Antypas, K. Wu, and H. Eom, "An empirical study of I/O separation for burst buffers in HPC systems," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 96–108, feb 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0743731520303907

[17] S. Ihara, "Lustre Optimizations and Improvements for Flash," in *Proceedings of the 2019 Lustre User Group*, Houston, TX, 2019. [Online]. Available: https://conference.cacds.uh.edu/wp-content/uploads/2019/05/LUG2019-Shuhchiihara-v3.pdf