Nine Months in the Life of an All-flash File System

Lisa Gerhardt*, Stephen Simms*, David Fox*, Kirill Lozinskiy*,
Wahid Bhimji*, Ershaad Basheer*, and Michael Moore†
*Lawrence Berkeley National Laboratory/National Energy Research Scientific Computing Center, CA, US;
and †Hewlett Packard Enterprise, US;
Email: * {lgerhardt, ssimms, dfox, klozinskiy, wbhimji, ebasheer}@lbl.gov, and † michael.moore@hpe.com

***Abstract*—NERSC's Perlmutter scratch file system, an all-flash Lustre storage system running on HPE (Cray) ClusterStor E1000 Storage Systems, has a capacity of 36 PetaBytes and a theoretical peak performance exceeding 7 TeraBytes per second across HPE's Slingshot network fabric. Deploying an all-flash Lustre file system was a leap forward in an attempt to meet the diverse I/O needs of NERSC. With over 10,000 users representing over 1,000 different projects that span multiple disciplines, a file system that could overcome the performance limitations of spinning disk and reduce performance variation was very desirable. While solid state provided excellent performance gains, there were still challenges that required observation and tuning. Working with HPE's storage team, NERSC staff engaged in an iterative process that increased performance and provided more predictable outcomes. Through the use of IOR and OBDfilter tests, NERSC staff were able to closely monitor the performance of the file system at regular intervals to inform the process and chart progress. This paper will document the results of and report insights derived from over 9 months of NERSC's continuous performance testing, and provide a comprehensive discussion of the tuning and adjustments that were made to improve performance.**

***Index Terms*—File systems, Lustre, Flash, NVMe, I/O performance.**

I. Introduction

Perlmutter [1], National Energy Research Scientific Computing Center's (NERSC) [2] flagship High-performance computing system, uses an all-flash Lustre file system [3] for high-speed I/O. Built on HPE (Cray) ClusterStor E1000 Storage System [4], with a size exceeding 36 PB, the file system is capable of delivering a theoretical peak performance in excess of 7 TB/s across the HPE Slingshot network fabric [5].

Because I/O is such a critical component of the work done at NERSC, optimizing performance and reducing performance variation are very important to ensuring efficient usage of resources. This has been a challenge, in part because NERSC serves more than 10,000 users from 1,000 different projects and the resulting workload is diverse, representing multiple disciplines. To enable Perlmutter to deliver performance to I/O patterns ranging from traditional checkpoint / restart to file-per-process random access data analytics, we turned to an all-NVMe [6] solution.

While designing the all-flash solution, we based our decisions on a quantitative analysis [7]. Being able to effectively balance capacity and cost was one of the crucial pieces of this analysis, which coupled with the latest market offerings, could allow us to offer NERSC users the low latency and high bandwidth that solid state storage promises. We determined that all-flash was becoming more affordable and therefore the perfect solution for Perlmutter. In the analysis we identified the optimal file system capacity, purge policy, required SSD endurance, and desired modern Lustre features. Years of historic workloads were analyzed to identify projections for performance and throughput needs. The outcome was that we were able to quantitatively demonstrate that it was economical and advantageous to deliver Perlmutter with an all-flash file system, which was one of the first of its kind.

While NVMe devices offered excellent performance compared with hard disks by eliminating slow downs from seeks and performance variation across tracks on the platter, they were not free from performance challenges. To achieve optimal and consistent performance with an NVMe file system required a lot of specific tuning.

Working with HPE's storage team, NERSC staff engaged in an iterative process that through time, increased performance and provided more predictable outcomes. Through the use of IOR [8] and obdfilter-survey [9], NERSC staff were able to closely monitor the performance of the file system at regular intervals to inform the process and chart progress. This paper will document the results of and report insights derived from over 9 months of NERSC's continuous performance testing, and provide a comprehensive discussion of the tuning and adjustments that were made to improve performance.

II. Perlmutter's Scratch File System

Perlmutter Scratch is a Lustre file system comprised of 16 metadata servers (MDSes), an initial 274 object storage servers (OSSes), and a total of 3,480 NVMe drives. Lustre OSS and MDS pairs are contained within ClusterStor E1000 enclosures and use the kfabric Lustre network driver (kfilnd) to communicate with clients across Perlmutter's Slingshot network. LDISKFS (based on EXT4) is used for the back-end file system. A full description of Perlmutter's scratch file system can be found in "Architecture and Performance of Perlmutter's 35 PB ClusterStor E1000 All-Flash File System" [10].

At deployment, the system provided 33 PB of usable space from 274 OSTs, but the capacity was increased to 36 PB from 298 OSTs in December of 2023. Each of NERSC's 10,000 users receives their own directory with default quotas of 20TB and 20M inodes. Directories are spread across the 16 metadata servers and user directory assignment is determined by the

TABLE I
TIMELINE OF MAJOR ACTIVITIES ON THE PERLMUTTER SCRATCH FILE SYSTEM

Date	Configuration Change
September 2023	SafeRet BIOS Update Applied [11]
October 2023	Trim Frequency Increased
November 2023	Lustre Checksumming Turned On [12]
December 2023	File System Capacity Increase
	Weighted Free Space Allocator Turned On
	Automatically Disable Writes for OSTs >75% Full
February 2024	User Data Purging Began

first letter of the username. The above mentioned capacity increase is only one change that the system has undergone since Perlmutter was accepted in July of 2023; a list of notable events is shown in Table I.

III. DATA COLLECTION

After acceptance, Perlmutter's high-performance scratch file system displayed intermittent, unexpectedly low write performance which prompted an investigation. The initial inquiry began with the hourly collection of data from IOR to help identify and evaluate the source of this occasional drop in write performance. These IOR runs were performed on 32 GPU nodes, writing and reading back approximately 250TB of data. The output files were striped so each run would interact with every OST on the system. These regularly run IOR tests gave us the ability to track changes in overall performance and eventually helped provide a metric for the file system's quality of performance.

At the beginning of the investigation, ad-hoc testing with obdfilter-survey revealed several OSTs intermittently, without obvious cause, reporting very slow write rates between 25% and 50% lower than expected. Due to the intermittent appearance of the problem, a daily, off-hours obdfilter-survey test was configured by NERSC staff in August of 2023 for further data collection. Over the remaining course of the investigation these reference obdfilter-survey data helped us to better understand the storage server behavior, identify OSTs that required further scrutiny, and verify the positive impact of remediation actions. Figure 1 shows the measured obdfilter results since the daily testing began.

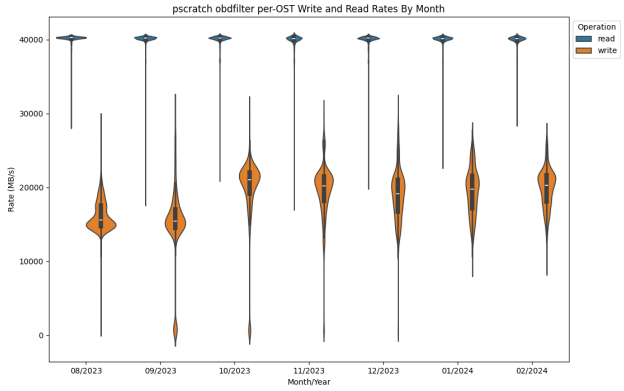

Fig. 1. Perlmutter Scratch obdfilter-survey OST Read and Write Rates

IV. STORAGE SERVER CHALLENGES

The move to a Lustre system based on solid state media presented challenges related to both, the solid state drives and the operating system's interaction with them as well as the Lustre file system's ldiskfs block allocation mechanism.

A. NVMe Related Issues

1) Garbage Collection: Both spinning disks and solid state drives map logical block addresses (LBAs) to physical block addresses (PBAs). In the case of spinning disks the LBAs have a one to one correspondence with their PBAs, which represent a physical location. An NVMe drive's LBAs are more akin to symbolic links which point to the correct PBAs, which represent memory locations. So, while spinning disk systems will support a physical overwrite of a disk location, solid state storage systems will not overwrite data. Instead, data may only be written into an empty space that is either unused or has been erased. While SSDs will only permit erasure at the granularity of a block, writes can be performed at the smaller granularity of a page. In the case of a read-modify-write, the page that has been read cannot be overwritten, so the modified page is written into a blank page while the unmodified page is marked as invalid. Garbage Collection (GC) in this context can be seen as a drive level read, write, erase that allows valid pages to be read and relocated into a new block, while pages marked invalid are left behind and reclaimed through erasure. GC occurs in the background and helps keep the NVMe drive's list of blocks available for new writes current.

2) Trimming: While GC is the mechanism that allows space occupied by invalid data to be reclaimed, the means by which the operating system communicates acts of deletion to drives is called trimming. While the drive knows to allocate a block for data via a write there is no implicit way for the drive to know a file has been removed and its associated data can again be usable capacity. Trimming is, at a high level, the process of communicating to the underlying NVMe devices which blocks are no longer in use by the file system. On many file systems, including ClusterStor, this is done through the `fstrim` command. As free blocks are communicated to the underlying NVMe devices the devices in turn erase the blocks so they are available for new writes. As the full capacity of the drive is written, even if data is removed, write performance begins to slowly decrease as the available pool of erased blocks is consumed. Performance can be restored by executing a trim on the drives. Without timely trimming, drives don't receive the necessary information to remove deleted data and GC will treat those data as valid, to be migrated. This problem of performance loss from the migration of deleted data is called write amplification.

3) Drive Utilization: As utilization on Perlmutter Scratch has increased, the pool of free blocks on the NVMe devices has decreased. Blocks holding valid data can neither be trimmed

nor take part in GC. So, when the file system has available capacity but the drives do not yet know through a trim, only a re-use of LBAs will cause writes to new memory addresses and therefore trigger GC. Additionally, if a trim has not been performed in a timely fashion, the re-use of LBAs could result in additional GC work from write amplification.

In general, a drive aims to perform GC at an opportune time of low use. However, if there are a large amount of incoming writes, as happens during an I/O test, the drive may need to perform GC during a write workload. Through the collection and analysis of NVMe drive telemetry logs by the drive vendor it was verified that individual devices were entering GC during a write workload which can increase write latencies by orders of magnitude.

4) Writeback Throttling: The final drive related piece of the investigation pertains to the kernel's writeback throttling mechanism. The mechanism helps maintain device responsiveness by limiting incoming writes as drive latency increases. This is a generally beneficial behavior because an increase in drive latency is often correlated with device load, and continuing to fill the drive queue will likely exacerbate the issue. However, a corner case emerged on ClusterStor with NVMe because of GC on a Redundant Array of Independent Disks (RAID) device. As previously observed, high latency can occur when a device enters GC and a ClusterStor E1000-based NVMe flash Object Storage Target (OST) is comprised of 12 NVMe devices. Any of those single NVMe devices entering GC will cause I/O at the RAID level to suffer increased latency because the I/O won't complete until all devices have completed the write. Further, since all devices are managing their internal space with slight variations, the devices will go into GC due to the incoming writes within the window of test but likely not for the same I/Os. In the case of Perlmutter's scratch it was observed that a relatively small number of requests were seeing extraordinarily high latency, occurring at slightly different times across the 12 devices. The high latency of a few outstanding commands would trigger the kernel's writeback throttling mechanism which would inadvertently throttle requests to the drives not in GC. While this process was occurring on the order of milliseconds, the observed impact on write performance was largely consistent when drives were in this state. By simply disabling the kernel's writeback throttling mechanism we were able to recover 10% of write performance when GC during writes was the limiting factor.

5) Problem Mitigation: The recommendations to address this group of issues are relatively straightforward and currently implemented on Perlmutter Scratch. First, disable write back throttling. This eliminated the previously described drive starvation resulting from to high latency seen from drives entering GC. The second recommendation is to perform a more frequent trim which will keep free blocks on the NVMe devices more closely aligned with current file system usage, and reduce reliance on LBA re-writes and GC to reclaim space. The application I/O performance impact during trim is minimal. Third, reset the ldiskfs allocation pointer to the beginning of the logical device through a /proc file system entry at a 15 minute interval i.e. `/proc/fs/ldiskfs/md0/mb_last_group`. This will encourage the re-use of LBAs and give drives more opportunity to perform GC before streaming writes force high-latency requests. The preference for sequentially writing all of the device's LBA space without reuse comes from its origins on rotational media where sequential writes were optimal for physical drive heads. As described above, using all LBAs presents a sub-optimal behavior for flash-based media where LBA and physical block mapping are not fixed. Finally, the most impactful change is to reduce OST utilization. Although initial expectations were that the media characteristics of NVMe drives would allow for full or near-full performance at high levels of utilization, we have seen that the necessary management of free blocks within the drive on live, production file systems does not match those expectations. The effects described in this section and inefficiencies in the current block allocator algorithm (described in the next section) suggest an optimal fullness value is currently around a surprising 75% of file system capacity.

B. Lustre Block Allocator Issues

Sitting above the NVMe and block-level RAID devices in the storage stack is the ldiskfs file system which is based on ext4 and runs on the Perlmutter Scratch OSTs. While a full discussion of ldiskfs block allocation is beyond the scope of this paper, what follows is a discussion of the investigation into and subsequent findings about ldiskfs block allocator performance.

When a freshly formatted Lustre file system begins allocating blocks on an OST for incoming data, it has a fairly trivial task. With many large, contiguous areas available to be allocated, the work required from the file system is minimal. As the file system ages, space is allocated and freed across the underlying OSTs, leaving a wide variety of contiguous blocks ranges. As both the count of ranges of contiguous blocks and contiguous blocks themselves decrease, the block allocator must perform more work, scanning for free blocks. As previously mentioned, the ldiskfs block allocator was developed in an era of rotational media and was not only optimized for that media but had performance requirements aligned with those type of OSTs. Although the size of NVMe based OSTs is smaller the higher supported throughput of NVMe-based OSTs has created new performance profiles for solid state block allocation. As one of the first large scale NVMe file systems, Perlmutter Scratch has helped illuminate these new demands as the file system has aged and utilization has increased.

1) Kernel Profiling and Findings: Linux kernel profiling on multiple OSSes via perf [13] was performed to further the investigation into the continued degraded write performance. A kernel module, available via a support request to HPE, is required in order to run perf on E1000 nodes. The profiling results showed both fast and slow write performance, and a closer evaluation identified time spent in block allocation

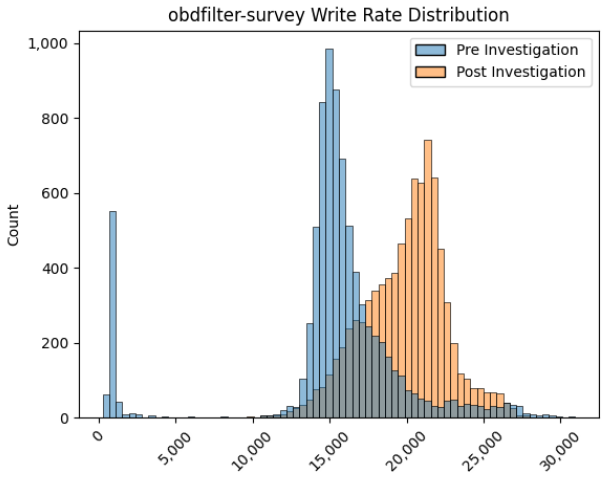

Fig. 2. Pre/Post Investigation obdfilter Monthly Write Rate Distribution

as a significant differentiator between the two. Higher CPU utilization was also visible on the OSSes with slower OSTs.

Next, the ldiskfs block allocation statistics were compared between fast and slow OSTs which are located in the mb_alloc_stats file here: `/proc/fs/ldiskfs/md0/mb_alloc_stats` Comparing the statistics highlighted a large number of useless c1 loops. The word "useless" in this context refers to an allocation request that executed a given loop but found no blocks to allocate. The increased CPU demand and associated latency of these useless c1 loops were the source of the decreased write performance. Empirical tests showed that this process started to become really inefficient around a threshold of 75%. This discovery has been unexpected and the issue is currently being investigated further.

2) Problem Mitigation: Given the complexity of the block allocator and time to fully verify a solution across OST types and all workloads a workaround was implemented using ldiskfs loop thresholds to skip over the c1 loop. The recommendation was to set the value of`/sys/fs/ldiskfs/md0/mb_c2_threshold` to 25 from 15.

```
/sys/fs/ldiskfs/md0/mb_c1_threshold:25
/sys/fs/ldiskfs/md0/mb_c2_threshold:25
/sys/fs/ldiskfs/md0/mb_c3_threshold:5
```

After implementation of the new block allocator loop thresholds, the the long tails in the obdfilter-survey results were greatly diminished in January and February as shown in Figure 1. The net positive effect of the investigation into OST write performance can be seen in Figure 2. The distribution of obdfilter-survey write measured during the month our efforts began compared to the month the investigation concluded illustrates a significant shift. In specific terms, the investigation yielded an increase in mean write rate from 15.1 GB/s to 19.9 GB/s - a 30% improvement. More importantly for application workloads, the minimum observed value increased from 260 MB/s to 9,032 MB/s. These measurements were run on all OSTs regardless of their current failover status, fullness, or utilization by application workloads. The dramatically improved minimum observed rate, while not ideal, aligns with what one would expect in cases where individual OST utilization exceeds 75%.

V. Maintaining OST Free Space

Managing free space on a file system can be very challenging. A runaway multi-node job, for example, can rapidly consume a lot of file system space. What our file system investigation has shown is that while maintaining free space is a challenge, it is also a solution. To maintain acceptable performance, it is imperative to maintain a minimum of 25% free space on the file system OSTs. Toward that end NERSC has implemented different mechanisms to manage the free space on Perlmutter Scratch such that no single OST will exceed 75% of its capacity.

A. File System Policy and Purging

NERSC uses on-file system metadata indexing information generated by ClusterStor Data Services to automatically purge eligible files from the file system and maintain a desired overall file system utilization. The overall target threshold for Perlmutter Scratch can be adjusted to ensure the system doesn't exceed the 75% OST target fullness.

Per NERSC policy, all user generated files on Perlmutter Scratch are eligible for automated purging with few exceptions. Each user is provided with a weekly list of files that look likely to be purged in the coming week, so attentive users can archive important files at risk of being purged.

When file system utilization exceeds the target threshold, the file age and size information gathered from the metadata indexes are used to calculate a purge horizon such that if all files older than the horizon were deleted the file system utilization would drop below the desired utilization threshold. For the purposes of the purging process, NERSC uses the most recent of three parameters (access time, modification time, and inode change time) to determine the age of a file so that older files are mostly deleted first. While NERSC advises users that files on Perlmutter Scratch become eligible for purging at 8 weeks of age, the current calculated purge horizon is approximately 365 days. We expect this number to fluctuate as usage changes on the system.

B. Automated OST Monitoring and Management

In addition to managing overall file system utilization, mechanisms have been implemented to reduce the likelihood that individual OSTs will exceed the 75% threshold. NERSC enables Lustre's Weighted-Free-Space allocator with aggressive settings to encourage the file system to keep OST usage relatively balanced. Users can specify file layouts that can reduce opportunities for the file system to make placement decisions, but this is fairly rare (most users either keep the default striping of 1 file per OST or only specify the number

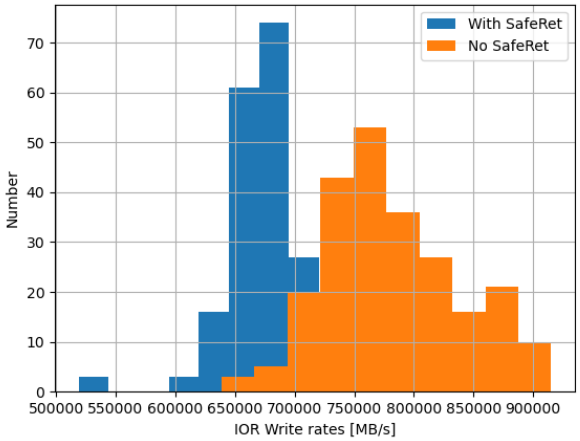

Fig. 3. Perlmutter Scratch IOR Write Rates [MB/s] before and after enabling SafeRet.

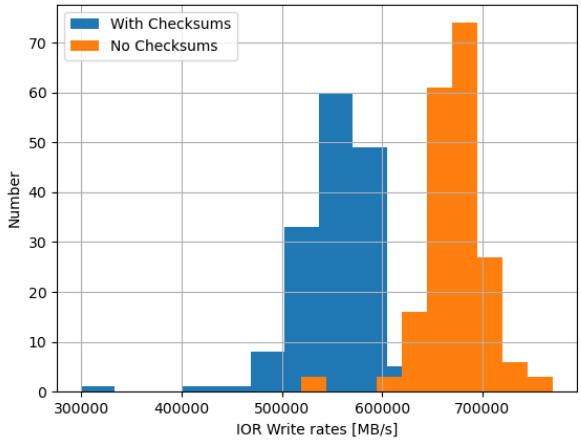

Fig. 4. Perlmutter Scratch IOR Write Rates [MB/s] before and after enabling checksums.

of OSTs they'd like file striped across). On the rare occasions when individual OSTs exceed 75% utilization, object creation on those OSTs is temporarily disabled, removing them from consideration by both the Lustre round-robin and weighted-free-space allocators. Natural attrition from file removal occurs through time and eventually lowers OST utilization. As a last resort, files that have large footprints on overly full OSTs can be identified using the metadata indexing information, and migrated using, e.g., lfs_migrate.

VI. Additional Performance Challenges

Over the 9 month period of Perlmutter Scratch operation there were several other unexpected issues that were presented. Because the nature of these problems involved security and data integrity, the proposed workarounds were not optional, yet have had a quantifiable impact on performance.

A. Security mitigation for microprocessors aka SafeRet

In August of 2023, AMD released a CVE warning of a side channel vulnerability on AMD CPUs that may result in speculative execution at an attacker-controlled address, potentially leading to information disclosure [14]. Dubbed "Inception", AMD released a BIOS patch and NERSC enabled a SafeRet kernel feature to further protect against this vulnerability in September of 2023. Because of the added overhead to each calculation, we measured a decrease in mean write bandwidth values of 14%. Figure 3 shows the write rates for IOR runs from two 240-hour periods before and after these BIOS updates were applied. This performance drop could be somewhat mitigated by disabling the SafeRet kernel feature, but this would be incompatible with NERSC's security posture. Also, the relative gain in performance would be negated by the effects of having to enable checksums in November.

B. Checksum Performance Impact

At the present time, HPE recommends that sites run Lustre with checksums disabled by default. NERSC followed this recommendation until November of 2023, when HPE issued a communique advising that all Lustre systems using kfilnd on SlingShot 11 activate checksums. This change would eliminate the possibility of data corruption that might result from a newly discovered bug involving the incorrect reuse of an RDMA memory key (RKEY). Before the communique and up to the change there had been no indication that any data corruption had occurred on Perlmutter scratch. However, after activation, NERSC observed two incidents where checksums had caught potential data corruption.

While NERSC currently has no plans to disable checksums, the assured data integrity provided does come with a performance penalty. Figure 4 shows write rates for IOR runs performed over two 240-hour periods, one with checksums enabled, and one without. Enabling checksums caused a 17% drop in mean write bandwidth values (from 673320 MB/s $\pm$ 31779 to 554012 MB/s $\pm$ 37846).

VII. Measuring Progress and Characterizing "Noise"

When navigating the investigation of and solutions for multiple problems on a production system, it can be a challenge to establish a reasonable way to measure progress and assess what success looks like.

The intermittent problem of low OST performance could be seen in the regular IOR runs. We quantify the performance variability in our IOR data using the coefficient of variation (COV, defined as the ratio of the standard deviation to the mean) over a 48-hour period. Since the longest wall time allowed for jobs on Perlmutter is 24 hours, choosing a 48 hour averaging period for this metric limited the impact of any one particular workload. For a target value, NERSC chose a COV

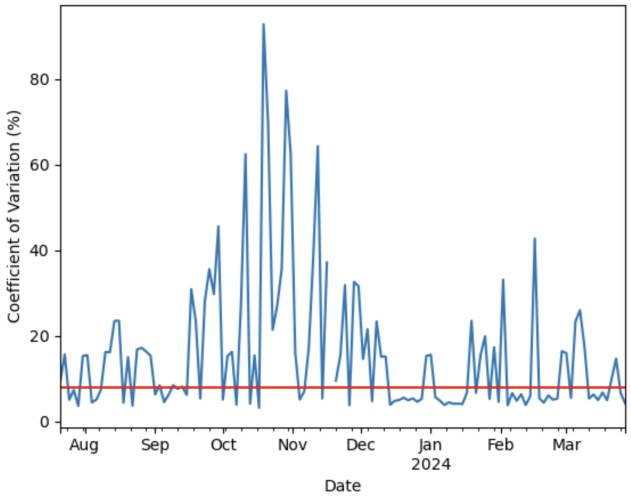

Fig. 5. COV values for Perlmutter Scratch IOR Write Rates [MB/s] over time. The red horizontal line denotes the desired value of 8%.

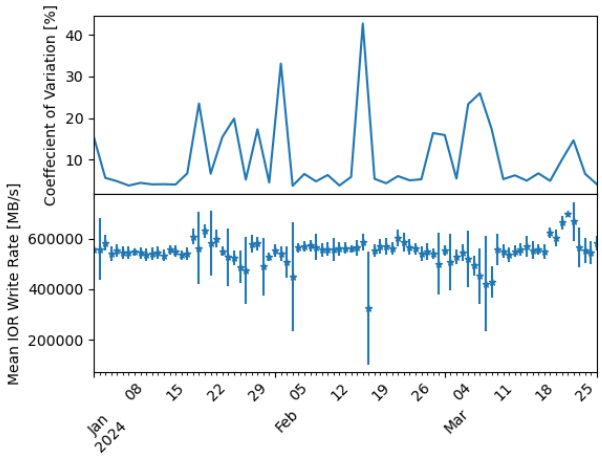

Fig. 6. Mean Perlmutter Scratch IOR Write Rates [MB/s] (the error bars are the standard deviation for the day) and the COV values over the same period.

of 8% or below, which aligns with the expected variation for Perlmutter's application benchmarks.

Fig. 5 shows the COV as a function of time for Perlmutter Scratch. By the start of 2024 most of the configurations designed to reduce performance variation had been applied and a corresponding drop in the frequency of large COV values was observed.

Taking a closer look at the 2024 figures, one can see that Perlmutter Scratch performance has largely stabilized since the beginning of the year. Fig 6 shows the mean write bandwidth values in MB/s from hourly 32-node IORs along with the corresponding COV values. Instances where an IOR run did not finish in its allotted time window are recorded as zeros. There are 13 days when the COV rose above the target value of 8%. For five of these days, we could not find a discernible cause for the elevated COV (2024-01-01, 2024-01-23, 2024-01-25, 2024-02-28, 2024-03-23). Four instances of an elevated COV were related to the file system being too full (2024-01-29, 2024-02-16, 2024-03-01, 2024-03-05). One instance was due to a single OST filling up very quickly. The other three stemmed from our OST load balancing mechanism, described in Section V-B, marking a large number of OSTs as read only. This meant only about 50 OSTs were available, resulting in slower write rates. Finally, the last two were a result of system failures, one within the Lustre file system where a single OST failed silently and another when Perlmutter experienced a system-wide issue with the proper resolution of DNS addresses.

VIII. Conclusion

Over the course of nine months NERSC and HPE staff worked together to better understand performance problems and provide more predictable file system performance. To achieve progress, we set up processes to collect data on a live production system which not only helped us address our primary performance issue, but helped NERSC to more easily assess the performance impact of necessary system changes and establish a metric to assess the relative performance health of the Perlmutter Scratch file system.

Our experiences during this span have underscored two things. First, this investigation has made clear the difficulty of diagnosing performance issues on a heavily used file system while in production. While the COV of our IOR tests has proven informative and very useful, there were still five performance events in Section VII for which we could find no obvious cause. NERSC will continue to work with HPE to better understand and mitigate these events.

Second, the value of an organized and performant purging system cannot be understated. In addition to ensuring free space for users, it is also necessary to keep solid state OSTs at less than 75% of capacity to maintain good performance. Once NERSC's purging efforts catch up from the current backlog, we expect the perturbations we see from load-balancing OSTs to cease.

The scale at which Perlmutter Scratch operates has helped illuminate some of the challenges of using Lustre on solid state media. Ideally some of the challenges around block allocation can be improved in the future for solid state media. Additionally, factors such as fragmentation and configured NVMe drive write per day could further improve the fullness threshold required for optimal performance. As others plan for their future file system, we hope our experiences will help provide them with relevant data. For those that may not have used Lustre with NVMe, we have tried to show some ways that spinning disk and solid state media differ in this context. For those that have, we hope that our experiences will inspire them to share theirs to help build a wider base of knowledge in this area.

Acknowledgment

The authors would like to thank John Fragalla, Jeff Hudson, Peter Bojanic, Cory Spitz, Alexander Zarochentsev, Chris

Walker, and Jeremy Higdon 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] "Perlmutter (NERSC-9) System," 2024. [Online]. Available: https://docs.nersc.gov/systems/perlmutter/

[2] "National Energy Research Scientific Computing Center," 2024. [Online]. Available: https://www.nersc.gov

[3] "Lustre file system," 2024. [Online]. Available: https://www.lustre.org/

[4] "Cray ClusterStor E1000 Storage Systems," 2024. [Online]. Available: https://buy.hpe.com/us/en/enterprise-solutions/storage-solutions/cray-clusterstor-storage-systems/cray-clusterstor-e1000-storage-systems/cray-clusterstor-e1000-storage-systems/p/1012842049

[5] "HPE Slingshot interconnect," 2024. [Online]. Available: https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html

[6] "NVM Express specifications," NVM Express, Inc., Tech. Rep., 2024. [Online]. Available: https://nvmexpress.org/specifications/

[7] 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.

[8] The Regents of the University of California, "IOR." [Online]. Available: https://github.com/hpc/ior

[9] P. Schwan, E. Barton, J. McIntyre, M. MacDonald, and C. White, "obdfilter-survey." [Online]. Available: https://github.com/lustre/lustre-release/tree/master/lustre-iokit/obdfilter-survey

[10] 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 *Cray Users Group*, 2021. [Online]. Available: https://cug.org/proceedings/cug2021_proceedings

[11] "Speculative Return Stack Overflow (SRSO)," 2024. [Online]. Available: https://docs.kernel.org/admin-guide/hw-vuln/srso.html

[12] "Chapter 23.5.1 Managing the File System and I/O Other I/O Options (Lustre Checksums)," in *Lustre Software Release 2.x Operations Manual*. [Online]. Available: https://doc.lustre.org/lustre_manual.xhtml#idm140334719896864

[13] "perf: Linux profiling with performance counters." [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page

[14] "Return Address Security Bulletin "INCEPTION"," Advanced Micro Devices, Inc., Tech. Rep., 2023. [Online]. Available: https://www.amd.com/en/resources/product-security/bulletin/amd-sb-7005.html