Title

Designing an all-flash Lustre file system for the 2020 NERSC Perlmutter system

Authors

Lockwood, Glenn K

Lozinskiy, Kirill

Gerhardt, Lisa

et al.

Publication Date

2024-01-21

Peer reviewed

Designing an All-Flash Lustre File System for the 2020 NERSC Perlmutter System

Glenn K. Lockwood, Kirill Lozinskiy, Lisa Gerhardt, Ravi Cheema, Damian Hazen, Nicholas J. Wright
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
{glock, klozinskiy, lgerhardt, rcheema, dhazen, njwright}@lbl.gov

***Abstract*—New experimental and AI-driven workloads are moving into the realm of extreme-scale HPC systems at the same time that high-performance flash is becoming cost-effective to deploy at scale. This confluence poses a number of new technical and economic challenges and opportunities in designing the next generation of HPC storage and I/O subsystems to achieve the right balance of bandwidth, latency, endurance, and cost. In this paper, we present the quantitative approach to requirements definition that resulted in the 30 PB all-flash Lustre file system that will be deployed with NERSC's upcoming Perlmutter system in 2020. By integrating analysis of current workloads and projections of future performance and throughput, we were able to constrain many critical design space parameters and quantitatively demonstrate that Perlmutter will not only deliver high performance, but effectively balance cost with capacity, endurance, and modern features in Lustre.**

I. Introduction

Historically, the most demanding I/O workloads in high-performance computing (HPC) have come from modeling and simulation applications checkpointing and restarting their internal state [1]. As a result, a tremendous amount of research and development has been carried out to develop file systems [2] and I/O middleware [3], [4] that specifically optimize for these checkpoint-restart I/O patterns. These research efforts have resulted in applications being able to extract tremendous aggregate I/O bandwidth from large quantities of storage media based on magnetic disks, with the largest all-disk deployments surpassing 1 TB/sec of performance [5].

Two major factors are shifting this conventional wisdom of I/O performance requirements in HPC though. The emergence of applying artificial intelligence (AI) at scale is showing promise as a completely new means to extract new insights from huge bodies of scientific data [6], [7]. Concurrently, there has been an explosion of high resolution detectors available to experimental and observational scientific communities [8], [9] which can produce scientific data at unprecedented rates [10], [11]. These workloads do not simply perform I/O to save and load the state of their calculation synchronously; rather, they are often marked by having to process volumes of data that far exceed the amount of memory available on their processing elements. Furthermore, the amount of computations they perform are often dictated by the contents of the data they are processing and cannot be determined *a priori*. As a result, the I/O patterns of these emerging workloads differ from those of traditional checkpoint-restart and do not perform optimally on today's production, disk-based parallel file systems.

Fortunately, the cost of flash-based solid-state disks (SSDs) has reached a point where it is now cost-effective to integrate flash into large-scale HPC systems to work around many of the limitations intrinsic to disk-based high-performance storage systems [12]–[14]. The current state of the art is to integrate flash as a *burst buffer* which logically resides between user applications and lower-performing disk-based parallel file systems. Burst buffers have already been shown to benefit experimental and observational data analysis in a multitude of science domains including high-energy physics, astronomy, bioinformatics, and climate [6], [8], [15]–[17]. However such burst buffers often enable high performance by compromising data reliability. For example, virtually all systems that utilize node-local flash provide no pretense of data protection in the form of parity or replication, so if a job's compute node fails, it is generally assumed that the contents stored on that node are permanently lost. Similarly, Cray's DataWarp burst buffer eschews parity protection on writes to deliver the maximum number of I/O operations per second (IOPS) to applications. This tradeoff makes burst buffers valuable for storing very active data for hours to days but never as the sole copy of a valuable dataset.

This decision to favor performance over resilience results in an usability challenge that remains unresolved. It is ultimately the responsibility of users to track the tier (flash or disk) in which their most up-to-date data resides. This incentivizes a return to a single high-performance storage tier, and the dropping cost of SSDs [18] are expected to once again give rise to a single, high-performance storage tier that has the performance of a burst buffer but the capacity of a traditional disk-based parallel file system [19], [20].

In 2020, the National Energy Research Scientific Computing Center (NERSC) will be deploying the Perlmutter HPC system which has been specifically designed to address the needs of emerging data-driven workloads in addition to traditional modeling and simulation. A foundation of Perlmutter's data processing capabilities will be its 30 PB, all-flash Lustre file system. Unlike previous deployments of flash in HPC systems [12], [21], [22] that have co-deployed high-performance flash with high-capacity disk, the Perlmutter file system will use flash exclusively for both performance and capacity. As such, it is critical that it deliver extreme bandwidth, extreme

data and metadata IOPS, high resilience, and high capacity while maintaining cost-effectiveness.

In practice, balancing performance, capacity, resilience, and cost requires a system architecture driven by several goals:

- The capacity of the file system must be "just enough" for the aggregate workload to ensure that flash, which is still expensive on a cost-capacity basis, is not over- or underprovisioned for capacity
- The SSD media must be of sufficient endurance to meet the service requirements of the workload without being overprovisioned for unrealistically high endurance levels, as this adds to overall cost
- All available performance features for low latency I/O with Lustre must be effectively provisioned for and usable by the workload

Meeting these goals requires a quantitative understanding of the I/O workload that will run on the target storage system to ensure that the most critical portions of the system architecture receive the most investment.

In this work, we present a series of analytical methods by which the requirements of a future all-flash file system can be quantitatively defined. We then use telemetric data collected from a reference production storage system to inform the minimum and maximum values required to achieve an optimal balance of capacity and value on a future all-flash parallel file system. However, we do *not* address I/O performance in this work because we expect that the absolute throughput of first-generation all-flash parallel file systems will be limited by software, not flash media, when they are initially deployed. It follows that the absolute performance of these systems will steadily increase with successive software improvements over the service lives of these all-flash file systems, making performance prediction difficult and highly dependent on software architecture, not system architecture.

II. Methods

The goal of this work is to define models through which the design space surrounding several key dimensions of parallel file system architecture can be quantified. These models project the requirements of a notional future parallel file system by combining data from an existing reference parallel file system with parameters that describe the future system. For simplicity, we refer to the model inputs as coming from the *reference* system, and the model outputs as describing the requirements for a *new* system. To illustrate the efficacy of these models, we then apply data collected from the I/O subsystem of Cori, a Cray XC-40 system deployed at NERSC, to derive the requirements for a notional all-flash Lustre file system that will be deployed with Perlmutter, NERSC's next-generation system.

A. Reference System

The reference system is Cori, a Cray XC-40 system comprised of 9,688 compute nodes with Intel Xeon Phi 7250 processors and 2,388 compute nodes with Intel Xeon E5-2698 v3 processors. Cori's I/O subsystem has two tiers: a disk-based Lustre file system and an all-flash DataWarp burst buffer. The precise configurations of these two storage tiers are described in Table I.

We rely on the Lustre Monitoring Tool (LMT) [23] to quantify the I/O requirements imposed on the reference file system by NERSC's production workload. LMT reports the total number of bytes read and written to each Lustre object storage target (OST) since the time each object storage server (OSS) was last rebooted on a five-second interval. We use the pytokio archival service [24] to persist this LMT data over the entire service life of Cori, allowing us to calculate the total number of bytes read and written to the file system over any arbitrary time interval. To understand how the file system's fullness increases, we run the standard Lustre `lfs df` command every five minutes to archive a consistent view each OST's fullness.

To characterize the utilization of the burst buffer tier on Cori, we use the Intel Data Center SSD Tool [25] to collect device-level data including the total bytes read and written to each SSD by the host DataWarp server and the total bytes read and written to the underlying NAND. The device-level read and write activity from the host is a close approximation of the aggregate user workload because user I/O is simply striped across devices without additional parity in DataWarp. We do expect the host writes to be slightly overstated due to the added device-level I/O caused by internal file system activity such as superblock updates, but we expect this effect to be minimal relative to volume of data written by user applications. Comparing the host- and NAND-level I/O volumes also allows us to explicitly calculate the aggregate write amplification factor (WAF $=$ NAND bytes written/host bytes written) of each SSD over its service life.

We obtain the distribution of file and inode sizes from a MySQL database populated using the Robinhood policy engine [26] version 3.1.4. This database contains the results of scanning the Lustre namespace from a Lustre client and inserting records that catalog the POSIX metadata fields intrinsic to each inode. For the purposes of this study, we extract the inode type and inode size for each record from this Robinhood database's `ENTRIES` table. For each type of inode, we then build histograms of inode sizes with exponentially increasing bins such that bin i contains the number of inodes with size S such that $2^{i-1} < S \leq 2^i$.

B. New System

Although the precise architecture of the new system, Perlmutter, was not defined at the time of writing, the models developed in this work are not designed to produce exact architectures and are therefore only dependent on high-level target capabilities. For the purposes of this work, it is sufficient to state that the Perlmutter system will have between $3\times$ and $4\times$ the capability of the reference system, Cori. Its overall scientific workload is expected to be similar to that of the reference system in terms of job mix, although the precise mechanisms by which jobs achieve high computational performance will be different by virtue of Perlmutter's substantial

TABLE I
DESCRIPTION OF REFERENCE SYSTEM

Tier (File System)	Capacity	Peak Bandwidth	# Data Servers	# Drives
Burst Buffer (DataWarp)	1.84 PB	1,740 GB/s	288	1,152
Scratch (Lustre)	30.5 PB	717 GB/s	248	10,168

GPU capability. NERSC also anticipates a new workload component coming from the need to perform large-scale analysis of experimental and observational scientific data. Given that the reference system's workload is largely dominated by traditional modeling and simulation workloads, the effects that these new analysis workloads will have on the overall I/O requirements of Perlmutter are not well understood.

III. File System Capacity

To determine the minimum required capacity, C^{new}, for the storage subsystem of a new HPC system, we use a simple growth model that uses empirical measurements from the reference HPC system's compute and storage subsystems. This model is expressed as

$$C^{\text{new}} = \text{SSI} \cdot \left(\frac{\lambda_{\text{purge}}}{\text{PF}}\right) \cdot \left(\frac{\partial C^{\text{ref}}}{\partial t}\right) \tag{1}$$

where

1) SSI is the Sustained System Improvement [27], a metric incorporating both performance and throughput improvement of the new system relative to the reference system
2) $(\lambda_{\text{purge}}/\text{PF})$ encapsulates the numerical description of the anticipated data retention policy of the new file system
3) $\left(\partial C^{\text{ref}}/\partial t\right)$ is daily growth rate observed on the reference file system

We use SSI to account for the fact that a system with a higher capability or throughput will be able to consume and generate data at a proportionally higher rate. For example, a workflow that can execute $3\times$ faster on the new system will be able to produce three times as much useful output in a fixed amount of time relative to the reference system assuming no other changes.

The $(\lambda_{\text{purge}}/\text{PF})$ term represents a data management policy whose terms can be interpreted in several different ways. λ_{purge} is a measure of time that reflects either the periodicity of purge cycles or the time after which files are eligible to be purged. PF is the fraction of total file system capacity to be reclaimed after each purge or the fraction fullness of the file system above which files become eligible for purging. These two terms provide enough flexibility to capture the most common approaches to purging. For example, files not touched in more than λ_{purge} days will be purged if doing so will aid in driving down file system fullness below $(100 \times \text{PF})\,\%$. As with any numerical expression of a data retention policy, it cannot capture the effects of ill-intentioned users who touch files to make them ineligible for purge.

The rate at which the reference file system grows, $\left(\partial C^{\text{ref}}/\partial t\right)$, is the most challenging term to calculate rigorously. In practice, the growth rate of file systems is a function of many variables including user diversity (some scientific workflows must retain more data than others [28]), time of year (conference deadlines and allocation expiration dates are often preceded by high utilization), and system age (improvements to system stability and usability encourage longer-term data retention). As such, correctly parameterizing $\left(\partial C^{\text{ref}}/\partial t\right)$ requires institutional knowledge of both the technological and sociological factors intrinsic to the reference system.

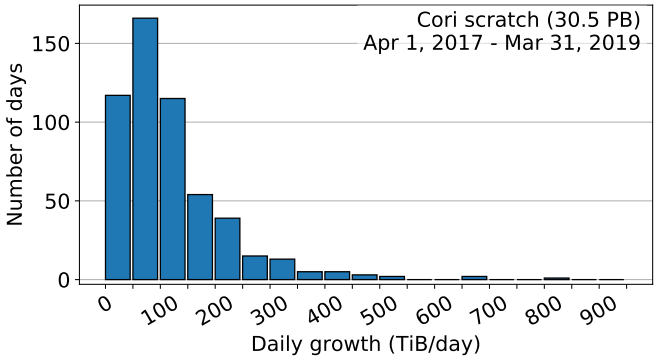

Fig. 1. Distribution of daily growth of Cori's scratch file system.

To determine $\left(\partial C^{\text{ref}}/\partial t\right)$ for our reference system, we first define the *daily growth* for day d as the difference between the capacity used on day d and $d-1$. We further constrain this daily growth metric by stating that it is undefined for days when there was a net reduction in file system fullness. In doing so, we minimize the effects of center-wide policies on daily growth by disregarding days during which significant amounts of user data were being purged.

To avoid biasing our analysis with the low growth rates often experienced during the earlier months of a system's service life, we also define an arbitrary cutoff date before which all daily growth measurements are discarded. For this study, we chose the cutoff to be exactly two years before the day on which the daily growth data were collected for this study to ensure that we captured the growth contributions of a broad range of projects that run at NERSC. In addition, our choice to align the sample period with a calendar year ensures that we capture the full range of sociological effects (such as conference deadlines) that may cause users to behave differently over the course of their year-long allocations.

Figure 1 shows the resulting distribution of this daily growth metric and reflects a median growth of 104 TB/day and a mean daily growth of 133 TB/day. The long tail of this distribution indicates that there are periods throughout the year when significant amounts of data are either generated within or imported to NERSC, and these outlying days should not be ignored when projecting future requirements. As a result, we choose to use the mean daily growth rather than the

median as the basis for our projected capacity requirements for Perlmutter and define $\left(\partial C^{\text{ref}}/\partial t\right) = 133$ TB/day.

We define the new system's data retention policy such that data older than 28 days is subject to purge, and each purge interval aims to remove or migrate 50% of the total file system capacity. Furthermore, the SSI for our new system is anticipated be between $3\times$ and $4\times$ that of the reference system. Given this range of anticipated SSI, Equation 1 gives the minimum required usable capacity C^{new} as being between 22 PB and 30 PB.

Although this is a wide range, Equation 1 provides a means to understand how tradeoffs can be made between user convenience (via a more generous data retention policy) and usable capacity. Similarly, the flexibility of the SSI metric also defines how changes to system capability, throughput, and application optimizations will affect storage system capacity requirements. Thus, it is possible to decide where in this range the target storage system capacity should be based on how a facility weighs each of these factors given a fixed budget.

IV. Drive Endurance

The flash cells within SSDs can be rewritten a finite number of times before they are no longer able to reliably store data, and as a result, SSDs are only warranted for a finite number of drive writes per day (DWPD) over their service life. Since HPC file systems have historically been subject to write-intensive workloads [29], [30], the endurance requirements of SSDs in HPC environments have been a cause of concern. To date, most large-scale flash deployments in HPC have resorted to using extreme-endurance SSDs (5-10 DWPD for a five-year period) to ensure that the SSDs do not fail before the end of the overall system's service life [12], [14].

This comes at a steep cost, though; for example, the Trinity supercomputer at Los Alamos National Laboratory employs a burst buffer comprised of drives configured to endure 10 DWPD instead of the factory default of 3 DWPD [14]. This is achieved by reserving 20% of each SSD's usable capacity for wear leveling, reducing the usable capacity of each SSD card from 4 TB to 3.2 TB. If this extreme level of endurance is not truly required though, reducing the drives' endurance from 10 DWPD to 3 DWPD would provide an additional 25% usable capacity at no added cost. To determine the optimal balance of cost and endurance for HPC workloads, we use an analytical model (Equation 2) that uses file system-level load data and sources of write amplification to demonstrate that 1 DWPD is sufficient for the anticipated Perlmutter workload.

$$\text{DWPD}^{\text{new}} = \text{SSI} \cdot \text{FSWPD}^{\text{ref}} \cdot \left(\frac{D+P}{D}\right) \cdot \text{WAF} \qquad (2)$$

where

1) SSI is the sustained system improvement as defined in Section III
2) $\text{FSWPD}^{\text{ref}}$ is the reference file system's total write volume expressed in units of file system writes per day

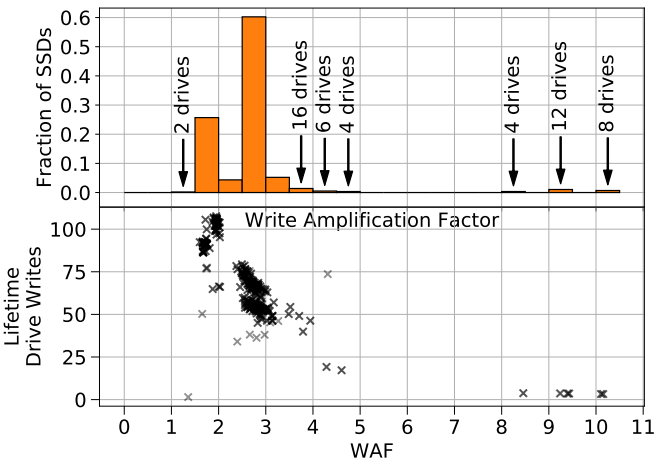

Fig. 2. Distribution of SSD WAFs on the Cori Burst buffer after approximately 3.4 years in service (top) and total lifetime write volumes, normalized to formatted drive capacity, for the WAF distribution (bottom)

3) D and P are the number of data and parity blocks, respectively, over which data are striped in the new system
4) WAF is the write amplification factor that results from factors intrinsic to the application workload

Because this model is independent of file system capacity, it can be applied to a new system of arbitrary size using data from a reference system of any size as long as the input parameters remain representative.

A. Parity and write amplification

The D and P parameters are required to account for the fact that a single user write is accompanied by additional parity blocks when written to the physical media. Similarly, WAF is required to account for the fact that writes that are smaller than a full RAID stripe require the RAID subsystem to (1) read the stripe blocks that will be modified, (2) make the modification to those blocks, (3) recalculate parity on P blocks, and (4) write a minimum of $P+1$ blocks back to the underlying media. This read-modify-write penalty is a function of the anticipated workload; if all applications buffer their writes such that only full-stripe writes are issued to the SSDs, this term is effectively 1.0. Alternatively, if 1,000 bytes are synchronously written to the file system one byte at a time, the effective write amplification factor (WAF) due to RAID would be $1000\times$ the RAID block size.

We do not directly monitor I/O transfer sizes on Cori's Lustre file system which prevents us from quantifying WAF for the reference system's workload. However, such transfer size distributions have been published for other large-scale Lustre file systems [29], [31], and the majority of transfers are either 4 KiB (the minimum Lustre transfer size) or 1 MiB (the maximum RPC size) as a result of Lustre's effective client-side write-back caching. Given that 4 KiB is the most frequent access size for SSDs and the size for which such devices are optimized [32], we do not anticipate an abnormally high WAF.

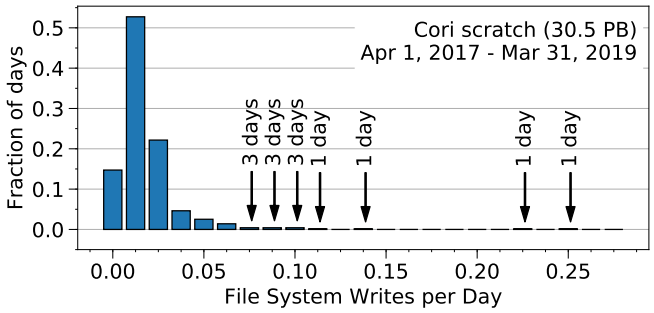

Fig. 3. Distribution of file system writes per day to the Cori scratch file system. 1 FSWPD = 30.5 PB of writes per day. Nonzero fractions in the tail are annotated in absolute days.

Furthermore, we can estimate the upper bound for WAF using actual WAF measurements taken from the reference system's burst buffer. As shown in Figure 2, the NERSC burst buffer workload results in WAFs ranging from 1.35 to 10.15, and the median and 95th percentile are $\text{WAF}_{50} = 2.68$ and $\text{WAF}_{95} = 3.17$, respectively. Although the drives showing extremely high WAFs (> 5) are cause for concern, these outliers are actually a result of drives that see extremely low use. Because SSDs must periodically rewrite pages internally regardless of if data is written to them from the host, there is a constant internal write load on SSDs which can become dominant in the presence of very light host write loads. In the case of Figure 2, all SSDs with WAF > 5 belong to a development partition of the burst buffer which is not accessible to production jobs.

B. Anticipated write load

The FSWPD parameter can be derived directly from a reference storage system by either direct measurement from file system telemetry or indirectly from device-level counters. We chose to define this parameter using data measured at the Lustre file system level since it is an unambiguous reflection of the user workload that excludes the effects of device-level buffering or amplification specific to the RAID implementation that resides between Lustre and the block devices.

Figure 3 shows the distribution of daily write workloads on the reference file system over a period of two years measured using LMT. As with the growth rates presented in Section III, there is a long tail of days that experience abnormally high write volumes which reflect the use of the file system as a data processing capability and should not be discarded. We therefore choose to use the mean, not median, FSWPD value of 0.024 FSWPD.

C. Endurance requirements

Because the SSI and WAF terms in Equation 2 have been defined as ranges, we apply Equation 2 to calculate the upper and lower bounds for the required drive endurance. We choose optimistic values of SSI, P/D, and WAF to determine the minimum required DWPD (DWPD_{min}) and pessimistic values to determine DWPD_{max}:

$$\begin{aligned} \text{DWPD}_{min} &= 3.0 \cdot \left(\frac{12}{10}\right) \cdot 2.68 \cdot 0.024 = 0.23 \\ \text{DWPD}_{max} &= 4.0 \cdot \left(\frac{10}{8}\right) \cdot 3.17 \cdot 0.024 = 0.38 \end{aligned} \tag{3}$$

From this, it becomes very clear that extreme-endurance SSDs are unnecessary for HPC workloads that resemble those of the reference system, and even 1 DWPD leaves significant headroom for increased wear from new workloads. Furthermore, advances in SSD controller technology are anticipated to reduce the overall WAF in the coming generation of SSDs and further reduce the need for high-endurance drives. For example, the Streams directive defined in the NVMe 1.3 standard [33] has been shown to reduce the WAF induced by mixed HPC workloads significantly [34].

Finally, an additional practical consideration when using Equation 2 to forecast the needs of a new storage system is the effect of increasing drive capacities. The bit density of NAND continually increasing [18], and as a result, the absolute number of bytes that comprises a single drive write per day is also increasing. Thus, using larger drives in a new storage system comes with the added benefit of increasing the absolute endurance of the aggregate file system and yields Equation 4.

$$\text{DWPD}^{\text{new}} = \text{DWPD}^{\text{ref}} \cdot \left(\frac{c^{\text{ref}}}{c^{\text{new}}}\right) \tag{4}$$

where DWPD^{ref} is given as defined in Equation 2, and c^{ref} and c^{new} are the per-drive capacities of the reference and new systems, respectively. Of course, this effectively trades performance for endurance; since per-SSD performance does not scale with per-SSD capacity, using fewer but larger drives results in less aggregate performance overall. Whether this tradeoff is appropriate is dependent on the workload, file system performance efficiency, OSS node architecture, and other factors beyond the scope of this discussion.

V. Metadata Configuration

Lustre's Data-on-MDT (DOM) feature allows the first S_0 bytes of every file to be stored on the same storage devices as their file metadata. This introduces several major benefits for small-file access:

1) Lock traffic is reduced since data and metadata are colocated
2) File size can be determined without sending RPCs to OSSes
3) Small file I/O interferes much less with large-file I/O on OSTs

However, DOM adds additional complexity to system design because MDT capacity must now account for both the capacity required to store inodes and the capacity required to store small files' contents. The precise definition of what constitutes a "small" file is also site-configurable, meaning that system architects must define both the required MDT capacity, C^{MDT}, and the threshold for storing small files exclusively on the MDT, S_0.

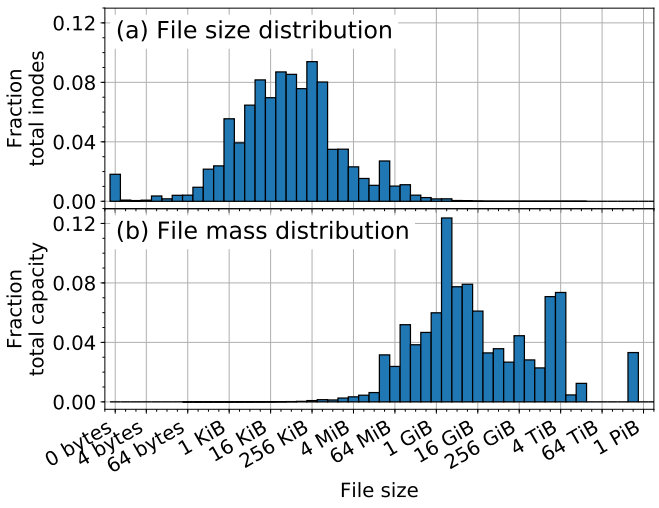

Fig. 4. (a) Probability distribution of file size, and (b) probability distribution of file mass (P_i^{file}) on NERSC's Cori file system in January 2019.

Thus, we define a model for the required MDT capacity as the sum of the capacity required by DOM to store the first S_0 bytes of every file (C^{DOM}) and the capacity required to store inodes (C^{inode}) in Equation 5.

$$C^{\text{MDT}} = C^{\text{DOM}} + C^{\text{inode}} \tag{5}$$

The required MDT capacity for a new system is invariably a function of the expected file size distribution on that new system. It is not sufficient to parameterize such a model on the average file size alone because file size distributions on HPC systems are almost always skewed towards small files [30], [35], [36], and small changes to the mean file size could represent a significant change to where the optimal DOM size threshold should be. As shown in Figure 4a, this is true of the reference system where 95% of the files comprise only 5% of the capacity used. As a result, both C^{DOM} and C^{inode} are a function of the probability distribution of file size, P_i^{file}, shown in Figure 4a.

A. MDT capacity required by DOM

To calculate C^{DOM}, we first convert the probability distribution of file sizes P_i^{file} into a mass distribution of data M_i^{data} for the new file system using Equation 6.

$$M_i^{\text{data}} = P_i^{\text{file}} \cdot C^{\text{new}} \tag{6}$$

Because P_i^{file} is expressed as a discrete histogram rather than a density function, Equation 6 requires that we assume all files in each bin i have an average mass that lies between the minimum and maximum size of the bin, $S_{i,\text{min}}$ and $S_{i,\text{max}}$. For example, if the bin bounded by $(1024 \text{ bytes}, 2048 \text{ bytes}]$ contains 512 files, we can only say that the total mass lies between 512.5 KiB (if all 512 files are of size $S_{i,\text{min}} = 1025$ bytes) and 1 MiB (if all 512 files are of size $S_{i,\text{max}} = 2048$ bytes). Thus, M_i^{data} is actually a set of mass distributions that result from assuming different average file sizes for each bin when applying Equation 6. Hereafter, we acknowledge this by referring to the set of mass distributions as $\boldsymbol{M}_i^{\text{data}}$. We use this set of distributions to attribute uncertainty to all subsequent calculations derived from M_i^{data} and explicitly calculate

- $M_i^{\text{data,min}}$ which assumes all files in i have size $S_{i,\text{min}}$
- $M_i^{\text{data,max}}$ which assumes all files in i have size $S_{i,\text{max}}$
- $M_i^{\text{data,avg}} = 1/2 \cdot (M_i^{\text{data,min}} + M_i^{\text{data,max}})$

From $\boldsymbol{M}_i^{\text{data}}$, we can then estimate file count distributions of the new file system, $\boldsymbol{N}_i^{\text{file}}$, using Equation 7.

$$\boldsymbol{N}_i^{\text{file}} = \boldsymbol{M}_i^{\text{data}} / \boldsymbol{S}_i \tag{7}$$

$\boldsymbol{N}_i^{\text{file}}$ is a set of distributions due to the dependence of Equation 7 on $\boldsymbol{M}_i^{\text{data}}$ and $\boldsymbol{S}_i$, both of which are themselves sets of distributions. Thus, as with $\boldsymbol{M}_i^{\text{data}}$, we carry forward the minimum, maximum, and average file count distribution using Equation 8.

$$\begin{aligned} N_i^{\text{file,min}} &= M_i^{\text{data,min}} / S_i^{\text{max}} \\ N_i^{\text{file,max}} &= M_i^{\text{data,max}} / S_i^{\text{min}} \\ N_i^{\text{file,avg}} &= M_i^{\text{data,avg}} / S_i^{\text{avg}} \end{aligned} \tag{8}$$

With an estimate of the number of files and their sizes for the new file system, we can now calculate the range of capacities required for DOM, $\boldsymbol{C}^{\text{DOM}}$, with Equation 9.

$$\boldsymbol{C}^{\text{DOM}} = \sum_i^{S_i \le S_0} \left(\boldsymbol{N}_i^{\text{file}} \cdot \boldsymbol{S}_i \right) + \sum_i^{S_i > S_0} \left(\boldsymbol{N}_i^{\text{file}} \cdot S_0 \right) \tag{9}$$

Equation 9 expresses the required MDT capacity for DOM in two components: (1) the mass of small files whose entire contents fit within the DOM threshold S_0 and (2) the mass of large files whose first stripe is stored using DOM. Thus, this gives us a way to determine the capacity required for DOM as a function of the DOM threshold S_0 that carries forward ranges of uncertainty intrinsic to our dependence on sets of discrete distributions.

B. MDT capacity required for inodes

The MDT capacity required to store inodes, C^{inode}, follows a similar approach. By default, Lustre reserves 4 KiB of MDT capacity for every inode, nominally making the process of calculating the inode capacity requirement quite simple. However, there are some cases where inodes can be significantly larger than 4 KiB as a result of, for example, directories containing millions of files. Figure 5 shows the probability distribution of non-file inodes' sizes on the reference system and demonstrates this phenomenon. In the most extreme case, a single directory inode is nearly 1 GiB in size as a result of it containing over eight million child inodes.

To ensure that such extreme requirements are not lost when calculating the MDT inode capacity requirements, we explicitly calculate the inode size distribution for every inode type (directories, symbolic links, etc.) based on the file size distributions $\boldsymbol{N}_i^{\text{file}}$ derived from Equation 7. Equation 10

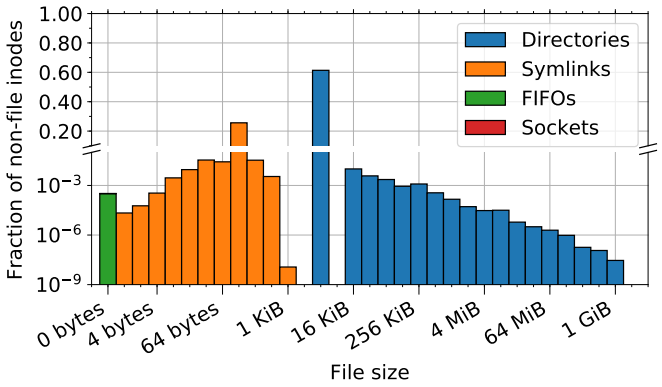

Fig. 5. Probability distribution of inode sizes on NERSC's Cori file system in January 2019. This diagram does not show block or character device inode types because none were present at the time of data collection. Break in y scale intended to contrast small numbers of large directory inodes with the predominant 4 KiB inode size.

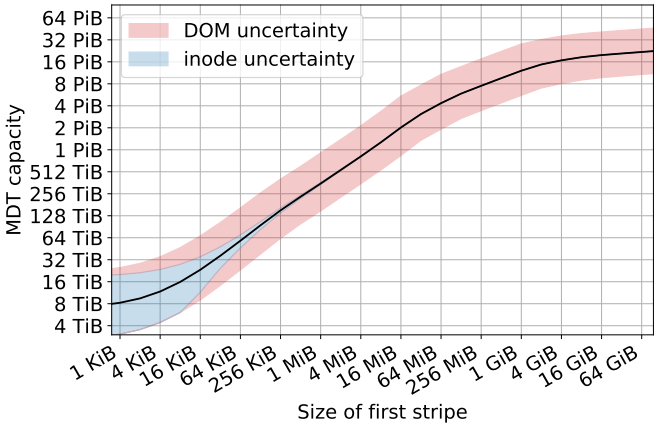

Fig. 6. Required MDT capacity as a function of S_0. Shaded area bounded by the minimum and maximum estimated requirements dictated by the DOM component and the inode capacity component of MDT capacity.

demonstrates this derivation for the directory size distribution; the process is the same for all non-file inode types.

$$\boldsymbol{N}_i^{\text{dir}} = \boldsymbol{N}_i^{\text{file}} \cdot \frac{N_i^{\text{ref,dir}}}{N_i^{\text{ref,file}}} \tag{10}$$

The inode size distribution reported by Robinhood can be misleading as a result of the difference between an inode's apparent size (as returned by `stat(2)`) and its block consumption. To ensure that inodes of small apparent size do not underrepresent the true inode capacity requirements, we assume that each inode whose apparent size is less than 4 KiB actually requires a full 4 KiB block. Thus, we calculate the total mass of these inodes using Equation 11.

$$\boldsymbol{C}^{\text{inode}} = \sum_i \left[\max\left(\boldsymbol{S}_i, 4096\right) \cdot \sum_j \boldsymbol{N}_i^j \right] \tag{11}$$

This equation gives the total mass of all bins i for all inodes of type j with the constraint that all inodes must consume at least one block and therefore be at least 4 KiB in size.

Although $\boldsymbol{N}_i^{\text{ref,j}}$ was extracted using Robinhood for this work, the identical analysis can be done using a reference system that is not Lustre. For example, $\boldsymbol{N}_i^{\text{ref,j}}$ can also be generated using IBM Spectrum Scale's Information Lifecycle Management policy manager or a simple parallel find tool. The only major practical consideration is that $N^{\text{ref,file}}$ must often be treated separately from the rest of the inode types since file size, not the file inode size, are reported by `stat(2)`.

C. Overall MDT capacity

Given Equations 9 and 11, we can now calculate total MDT capacity requirements using Equation 5. Taking the most conservative and optimistic values for the mass distribution (P_i^{file}) and inode size distributions ($\boldsymbol{N}_i^j$ and $\boldsymbol{N}_i^{\text{ref},j}$), we can evaluate $C^{\text{MDT,min}}$, $C^{\text{MDT,max}}$, and $C^{\text{MDT,avg}}$ as a function of the DOM threshold size S_0. Figure 6 shows the result of this model for a target capacity $C^{\text{new}} = 30$ PB from Section III. The shaded regions bound $C^{\text{MDT,min}}$ and $C^{\text{MDT,max}}$, and the black line is $C^{\text{MDT,avg}}$, a reasonable estimate of the true requirement. Furthermore, the components of this uncertainty attributed to $\boldsymbol{C}^{\text{DOM}}$ and $\boldsymbol{C}^{\text{inode}}$ are separated.

The sigmoidal shape of C^{MDT}'s dependence on the DOM threshold S_0 is a result of two competing factors. For very small S_0, the large number of small files simply does not consume a large amount of DOM capacity so there are only modest increases in C^{DOM} in this region. For very large S_0, the great majority of files are stored entirely within the MDT and only a small number of very large files are increasing the C^{DOM} requirements.

The magnitude of uncertainty is also affected by these properties of the file size distribution. Given our definition of C^{inode} in Equation 11, the capacity requirements to store inodes is independent of S_0 and therefore remains constant in x in Figure 6. However, the uncertainty contributed by C^{DOM} increases in proportion to the number of files entirely stored on the MDT. Thus, the predominant uncertainty at small DOM thresholds arises from our estimation of how many inodes will be required on the new file system. The DOM capacity is orders of magnitude smaller by comparison; only $C^{\text{DOM,max}}$ approaches the same order of magnitude as $\boldsymbol{C}^{\text{inode}}$ for small S_0. For large S_0, the reverse is true and the dependence of C^{DOM} on S_0 results in uncertainty increasing proportionally while the relative magnitude of C^{inode} becomes vanishingly small. For very large S_0, the majority of files reside entirely on the MDT, and the effects of increasing S_0 results in minimal increases in C^{MDT} overall.

The region between the two extremes of small and large DOM thresholds leaves considerable room for optimization though. For example, doubling the MDT capacity from 512 TiB to 1 PiB allows a fourfold increase in S_0. The cost-per-bit for an MDT is typically higher than that for an OST due to different parity configuration (e.g., 5+5 parity on an MDT vs. 8+2 on an OST), but this increased cost comes with better IOPS performance.

If one assumes that C^{DOM} is proportional to cost and S_0 is proportional to IOPS performance, Figure 6 becomes a price-performance curve as well. In this context, the behavior for $S_0 \rightarrow C^{\text{new}}$ suggests that the benefit of increasing S_0 above several GiB is not an optimal configuration for price/performance. Thus, while a Lustre file system entirely comprised of MDTs with DOM could be possible in principle, its performance improvements would likely not justify its cost when compared to a Lustre file system with a judiciously chosen S_0. Furthermore, S_0 is inversely proportional to bandwidth performance since DOM is not striped, so choosing a very large S_0 would adversely impact per-file bandwidth as well.

VI. Conclusion

We have presented methods by which workload data from a reference file system can be used to determine the best balance of cost, performance, and usability along several dimensions. We then quantified the relationship between factors including purge policy, growth rate, and file size distribution and design space parameters surrounding an all-flash file system such as data capacity, SSD endurance, and metadata configuration. As the economics of flash continue to displace hard disk drives from high-performance storage tiers, such analytical methods will become increasingly important for future system deployments.

Acknowledgments

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] B. Xie, J. Chase, D. A. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki, "Characterizing output bottlenecks in a supercomputer," in *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, nov 2012, pp. 1–11. [Online]. Available: http://ieeexplore.ieee.org/document/6468446/

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

[3] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate, "PLFS: A Checkpoint Filesystem for Parallel Applications," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis - SC '09*. New York, New York, USA: ACM Press, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1654059.1654081

[4] K. Sato, K. Mohror, A. Moody, T. Gamblin, B. R. de Supinski, N. Maruyama, and S. Matsuoka, "A User-Level InfiniBand-Based File System and Checkpoint Strategy for Burst Buffers," in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, may 2014, pp. 21–30. [Online]. Available: http://ieeexplore.ieee.org/document/6846437/

[5] S. S. Vazhkudai, B. R. de Supinski, A. S. Bland, A. Geist, J. Sexton, J. Kahle, C. J. Zimmer, S. Atchley, S. Oral, D. E. Maxwell, V. G. V. Larrea, A. Bertsch, R. Goldstone, W. Joubert, C. Chambreau, D. Appelhans, R. Blackmore, B. Casses, G. Chochia, G. Davison, M. A. Ezell, T. Gooding, E. Gonsiorowski, L. Grinberg, B. Hanson, B. Hartner, I. Karlin, M. L. Leininger, D. Leverman, C. Marroquin, A. Moody, M. Ohmacht, R. Pankajakshan, F. Pizzano, J. H. Rogers, B. Rosenburg, D. Schmidt, M. Shankar, F. Wang, P. Watson, B. Walkup, L. D. Weems, and J. Yin, "The design, deployment, and evaluation of the coral pre-exascale systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 52:1–52:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291726

[6] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, Prabhat, and M. Houston, "Exascale deep learning for climate analytics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 51:1–51:12, arXiv:1810.01993. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291724

[7] W. Joubert, D. Weighill, D. Kainer, S. Climer, A. Justice, K. Fagnan, and D. Jacobson, "Attacking the opioid epidemic: Determining the epistatic and pleiotropic genetic architectures for chronic pain and opioid addiction," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 57:1–57:14. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291732

[8] 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

[9] 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/s12859-015-0736-4 http://www.biomedcentral.com/1471-2105/16/304

[10] J. Thayer, D. Damiani, C. Ford, I. Gaponenko, W. Kroeger, C. O'Grady, J. Pines, T. Tookey, M. Weaver, and A. Perazzo, "Data systems for the Linac Coherent Light Source," *Journal of Applied Crystallography*, vol. 49, no. 4, pp. 1363–1369, aug 2016. [Online]. Available: http://scripts.iucr.org/cgi-bin/paper?S1600576716011055

[11] B. Alewijnse, A. W. Ashton, M. G. Chambers, S. Chen, A. Cheng, M. Ebrahim, E. T. Eng, W. J. Hagen, A. J. Koster, C. S. López, N. Lukoyanova, J. Ortega, L. Renault, S. Reyntjens, W. J. Rice, G. Scapin, R. Schrijver, A. Siebert, S. M. Stagg, V. Grum-Tokars, E. R. Wright, S. Wu, Z. Yu, Z. H. Zhou, B. Carragher, and C. S. Potter, "Best practices for managing large CryoEM facilities," *Journal of Structural Biology*, vol. 199, no. 3, pp. 225–236, sep 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1047847717301314

[12] S. M. Strande, P. Cicotti, R. S. Sinkovits, W. S. Young, R. Wagner, M. Tatineni, E. Hocks, A. Snavely, and M. Norman, "Gordon: Design, performance, and experiences deploying and supporting a data intensive supercomputer," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond*, ser. XSEDE '12. New York, NY, USA: ACM, 2012, pp. 3:1–3:8. [Online]. Available: http://doi.acm.org/10.1145/2335755.2335789

[13] 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

[14] 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.

[15] C. S. Daley, D. Ghoshal, G. K. Lockwood, S. Dosanjh, L. Ramakrishnan, and N. J. Wright, "Performance characterization of scientific workflows for the optimal use of Burst Buffers," *Future Generation Computer Systems*, dec 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X16308287

[16] N. T. Weeks and G. R. Luecke, "Optimization of SAMtools sorting using OpenMP tasks," *Cluster Computing*, vol. 20, no. 3, pp. 1869–1880, 2017.

[17] J. Regier, K. Pamnany, K. Fischer, A. Noack, M. Lam, J. Revels, S. Howard, R. Giordano, D. Schlegel, J. McAuliffe, R. Thomas, and

. Prabhat, "Cataloging the visible universe through bayesian inference at petascale," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2018, pp. 44–53.

[18] R. E. Fontana and G. M. Decad, "Moore's law realities for recording systems and memory storage components: HDD, tape, NAND, and optical," *AIP Advances*, vol. 8, no. 5, p. 056506, may 2018. [Online]. Available: http://aip.scitation.org/doi/10.1063/1.5007621

[19] John Bent, Brad Settlemyer, and Gary Grider, "Serving Data to the Lunatic: Fringe The Evolution of HPC Storage," *;login:*, vol. 41, no. 2, pp. 34–39, 2016. [Online]. Available: https://www.usenix.org/publications/login/summer2016/bent

[20] G. K. Lockwood, D. Hazen, Q. Koziol, S. Canon, K. Antypas, J. Balewski, N. Bathaser, W. Bhimji, J. Botts, J. Broughton, T. L. Butler, G. F. Butler, R. Cheema, C. S. Daley, T. Declerck, L. Gerhardt, W. E. Hurlbert, K. A. Kallback-Rose, S. Leak, J. Lee, R. Lee, J. Liu, K. Lozinskiy, D. Paul, Prabhat, C. Snavely, J. Srinivasan, T. Stone Gibbins, and N. J. Wright, "Storage 2020: A Vision for the Future of HPC Storage," Lawrence Berkeley National Laboratory, Berkeley, CA, Tech. Rep., 2017. [Online]. Available: https://escholarship.org/uc/item/744479dp

[21] N. Gaffney, C. Jordan, T. Minyard, and D. Stanzione, "Building Wrangler: A transformational data intensive resource for the open science community," in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, oct 2014, pp. 20–22. [Online]. Available: http://ieeexplore.ieee.org/document/7004480/

[22] T. K. Petersen and B. Loewe, "The Role of SSD Block Caches in a World of Networked Burst Buffers," in *Proceedings of the 2018 Cray User Group*, Stockholm, 2018.

[23] A. Uselton, "Deploying Server-side File System Monitoring at NERSC," in *Proceedings of the 2009 Cray User Group*, 2009.

[24] G. K. Lockwood, N. J. Wright, S. Snyder, P. Carns, G. Brown, and K. Harms, "TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis," in *Proceedings of the 2018 Cray User Group*, 2018.

[25] "Intel SSD Data Center Tool," 2017. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000006289

[26] T. M. Declerck, "Using Robinhood to Purge Data from Lustre File Systems," in *Proceedings of the 2014 Cray User Group*, Lugano, CH, 2014. [Online]. Available: https://cug.org/proceedings/cug2014_proceedings/includes/files/pap157.pdf

[27] B. Austin, C. Daley, D. Doerfler, J. Deslippe, B. Cook, B. Friesen, T. Kurth, C. Yang, and N. J. Wright, "A Metric for Evaluating Supercomputer Performance in the Era of Extreme Heterogeneity," in *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, nov 2018, pp. 63–71. [Online]. Available: https://ieeexplore.ieee.org/document/8641549/

[28] "APEX Workflows Whitepaper," Los Alamos National Laboratory, Lawrence Berkeley National Laboratory, and Sandia National Laboratories, Tech. Rep., 2016. [Online]. Available: https://www.nersc.gov/assets/apex-workflows-v2.pdf

[29] R. Gunasekaran, S. Oral, J. Hill, R. Miller, F. Wang, and D. Leverman, "Comparative I/O workload characterization of two leadership class storage clusters," in *Proceedings of the 10th Parallel Data Storage Workshop (PDSW'15)*. New York, New York, USA: ACM Press, 2015, pp. 31–36. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2834976.2834985

[30] 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

[31] 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

[32] W. Cheong, C. Yoon, S. Woo, K. Han, D. Kim, C. Lee, Y. Choi, S. Kim, D. Kang, G. Yu, J. Kim, J. Park, K.-W. Song, K.-T. Park, S. Cho, H. Oh, D. D. G. Lee, J.-H. Choi, and J. Jeong, "A flash memory controller for 15μs ultra-low-latency SSD using high-speed 3D NAND flash with 3μs read time," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, vol. 61. IEEE, feb 2018, pp. 338–340. [Online]. Available: http://ieeexplore.ieee.org/document/8310322/

[33] "NVM Express Revision 1.3," NVM Express, Inc., Tech. Rep., 2017. [Online]. Available: https://nvmexpress.org/wp-content/uploads/NVM_Express_Revision_1.3.pdf

[34] 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.

[35] G. K. Lockwood, R. Wagner, and M. Tatineni, "Storage utilization in the long tail of science," in *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2792777

[36] 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