Title

A Quantitative Approach to Architecting All-Flash Lustre File Systems

ISBN

978-3-030-34355-2

Authors

Lockwood, Glenn K

Lozinskiy, Kirill

Gerhardt, Lisa

et al.

Publication Date

2019

DOI

10.1007/978-3-030-34356-9_16

Peer reviewed

A quantitative approach to architecting all-flash Lustre file systems

Glenn K. Lockwood, Kirill Lozinskiy, Lisa Gerhardt, Ravi Cheema, Damian Hazen, and Nicholas J. Wright

Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
{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 work, we present quantitative models that use workload data from existing, disk-based file systems to project the architectural requirements of all-flash Lustre file systems. Using data from NERSC's Cori I/O subsystem, we then demonstrate the minimum required capacity for data, capacity for metadata and data-on-MDT, and SSD endurance for a future all-flash Lustre file system.

Keywords: architecture · Lustre · flash

1 Introduction

The conventional wisdom of I/O subsystem design in high-performance computing (HPC) are rapidly changing as a result of the broadening scope of HPC beyond traditional modeling and simulation. 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 [13, 14]. Concurrently, there has been an explosion of high resolution detectors available to experimental and observational scientific communities [5, 18] which can produce scientific data at unprecedented rates [3, 20]. 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 [6, 11, 19]. 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 [5, 7, 14, 17, 24].

However such burst buffers often enable high performance by exposing additional complexity to users. For example, DataWarp burst buffers offer a separate ephemeral namespace into which data must be staged in or out using non-standard APIs, while Infinite Memory Engine offers only eventual consistency between data written to flash and data on the backing file system. 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 [9] 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 [12, 15].

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.

2 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.

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 1.

Table 1. 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

We rely on the Lustre Monitoring Tool (LMT) [21] 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. 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 [2] 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. 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 [8] 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 each type of inode, we build histograms of inode size with exponentially increasing bins such that bin i contains the number of inodes with size S such that $2^{i-1} < S \leq 2^i$.

3 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 [4], 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× 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 though, it cannot capture the effects of ill-intentioned users who touch files to make them ineligible for purge, the effects of excluding certain projects from purge, or other purge exceptions that manifest in real production systems. As such, this may be a liberal estimate what the true production data retention policy may be and should be specified with these uncertainties in mind.

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 [1]), 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.

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.

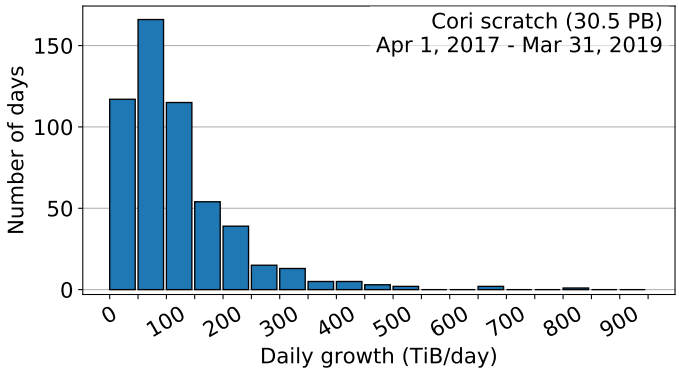

Fig. 1. Distribution of daily growth of Cori's scratch file system.

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 $\left(\partial C^{\text{ref}}/\partial t\right)$.

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× and 4× 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.

4 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 [10, 22], 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 [11, 19].

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 [11]. This is achieved by reserving 20% of each SSD's usable capacity for wear leveling. 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 define the minimum required DWPD for an all-flash file system.

$$\text{DWPD}^{\text{new}} = \text{SSI} \cdot \text{FSWPD}^{\text{ref}} \cdot \text{WAF} \cdot \left(\frac{1}{\chi}\right)\left(\frac{N^{\text{ref}}}{N^{\text{new}}}\right)\left(\frac{c^{\text{ref}}}{c^{\text{new}}}\right)\left(\frac{R^{\text{ref}}}{R^{\text{new}}}\right) \quad (2)$$

where

1. SSI is the sustained system improvement as defined in Section 3
2. $\text{FSWPD}^{\text{ref}}$ is the reference file system's total write volume expressed in units of file system writes per day
3. WAF is the write amplification factor that results from factors intrinsic to the application workload and file system implementation

4. χ is the fraction of Lustre capacity available after formatting, typically ranging from 0.95 to 0.97
5. N^{ref} and N^{new} are the number of drives in the reference and new systems
6. c^{ref} and c^{new} are the per-drive capacities in the reference and new systems
7. R^{ref} and R^{new} are the code rates of the reference and new systems

Figure 2 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 3, 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.

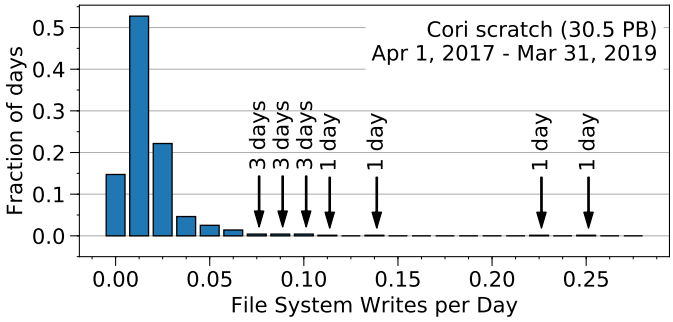

Fig. 2. Distribution of file system writes per day to the Cori scratch file system. 1 FSWPD = 30.5 PB of writes per day. Nonzero fractions in the tail are annotated in absolute days.

The WAF term accounts for the fact that writes 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.[1] 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.

We do not directly monitor I/O transfer sizes on the Cori file system which prevents us from quantifying WAF. However, we can estimate WAF from the SSDs in the reference system's burst buffer, shown in Figure 3 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) appear to be cause for concern, these outliers are actually a result of drives that see extremely low use. Because

[1] We do not consider write amplification caused by garbage collection internal to the SSDs since drive endurance is warranted on the basis of host-initiated write load, not total write load to NAND.

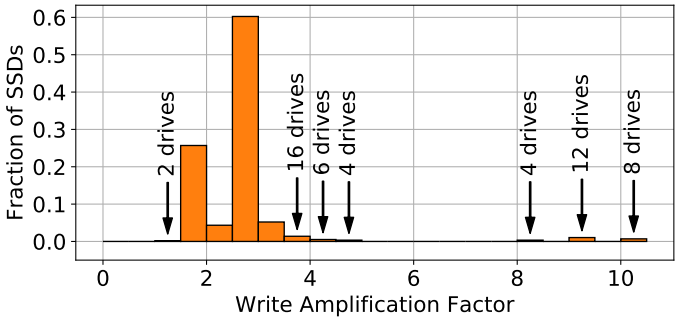

Fig. 3. Distribution of SSD WAFs on the Cori Burst buffer after approximately 3.4 years in service

SSDs must periodically rewrite pages internally regardless of weather data is written to them from the host, there is a constant internal write load on SSDs which becomes dominant in the presence of very light host usage. In the case of Figure 3, all SSDs with WAF > 5 belong to a development partition of the burst buffer that is not in production.

The χ and R terms in Equation 2 are required to account for the fact that the formatted capacity of a reliable Lustre file system is smaller than the aggregate storage device capacity. The coding rates R^{ref} and R^{new} capture differences in write amplification caused by different ratios of parity overheads in the systems' RAID schemes. χ accounts for additional minor capacity overheads such as the root file system of each Lustre OSS. The N and c terms are required to account for differences in the actual file system capacity of the reference file system and new file system, as a single file system write to a small reference file system represents only a fraction of a file system write to a larger new file system.

We choose values that are optimistic (SSI = 3.0, $R^{\text{new}} = {}^{10}/_{12}$, and WAF = 2.68) and pessimistic (SSI = 4.0, $R^{\text{new}} = {}^{8}/_{10}$, and WAF = 3.17) to determine the minimum and maximum required DWPD, respectively. Because the specific hardware geometry for a new file system (N^{new} and c^{new}) may not be defined at the time of requirements definition, we note that $\chi \cdot N^{\text{new}} \cdot c^{\text{new}} \cdot R^{\text{new}} \approx C^{\text{new}}$ and reduce Equation 2 to

$$\text{DWPD}^{\text{new}} \approx \text{SSI} \cdot \text{FSWPD}^{\text{ref}} \cdot \text{WAF} \cdot \frac{N^{\text{ref}} \cdot c^{\text{ref}} \cdot R^{\text{ref}}}{C^{\text{new}}} \tag{3}$$

Given that $R^{\text{ref}} = {}^{8}/_{10}$, $N^{\text{ref}} = 10168$, and $c^{\text{ref}} \approx 4$ TB, we find that $\text{DWPD}^{\text{new}}_{\text{min}} = 0.28$ and $\text{DWPD}^{\text{new}}_{\text{max}} = 0.33$. 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 unanticipated new workloads.

5 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.

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[2] (C^{inode}) in Equation 4.

$$C^{\mathrm{MDT}} = C^{\mathrm{DOM}} + C^{\mathrm{inode}} \tag{4}$$

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 [16,22,23], 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 4, 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 4.

5.1 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 5.

$$M_i^{\mathrm{data}} = P_i^{\mathrm{file}} \cdot C^{\mathrm{new}} \tag{5}$$

Because P_i^{file} is expressed as a discrete histogram rather than a density function, Equation 5 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, $S_{i,\min}$ and $S_{i,\max}$. For example, if the bin bounded by (1024 bytes, 2048 bytes] contains 512 files, we can only say that the total mass lies between 512.5 KiB (if all 512 files

[2] Strictly speaking, we define C^{inode} to include the MDT block allocated for inodes *and* additional data blocks that may be required to store, for example, large numbers of directory entries.

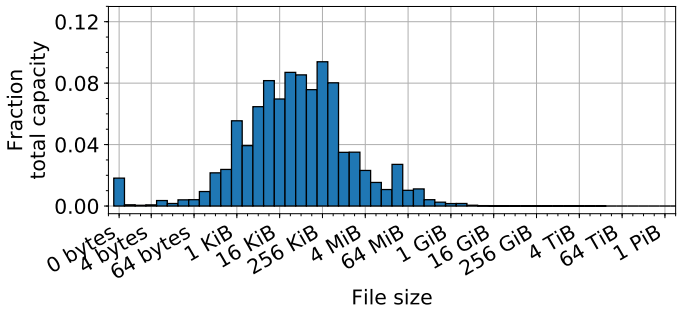

Fig. 4. Probability distribution of file size on the reference system in January 2019.

are of size $S_{i,\min} = 1025$ bytes) and 1 MiB (if all 512 files are of size $S_{i,\max} = 2048$ 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 5. 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 $S_{i,\min}$
- $M_i^{\text{data,max}}$ which assumes all files in i have size $S_{i,\max}$
- $M_i^{\text{data,avg}} = 1/2 \cdot (M_i^{\text{data,min}} + M_i^{\text{data,max}})$

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 6.

$$\boldsymbol{N}_i^{\text{file}} = \boldsymbol{M}_i^{\text{data}} / \boldsymbol{S}_i \tag{6}$$

$\boldsymbol{N}_i^{\text{file}}$ is a set of distributions due to the dependence of Equation 6 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 7.

$$\begin{aligned} N_i^{\text{file,min}} &= M_i^{\text{data,min}} / S_i^{\max} \\ N_i^{\text{file,max}} &= M_i^{\text{data,max}} / S_i^{\min} \\ N_i^{\text{file,avg}} &= M_i^{\text{data,avg}} / S_i^{\text{avg}} \end{aligned} \tag{7}$$

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 8.

$$\boldsymbol{C}^{\text{DOM}} = \sum_i^{S_i \le S_0} \left(\boldsymbol{N}_i^{\text{file}} \cdot \boldsymbol{S}_i \right) + \sum_i^{S_i > S_0} \left(\boldsymbol{N}_i^{\text{file}} \cdot S_0 \right) \tag{8}$$

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.

5.2 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, but there are cases where an inode can consume significantly more capacity. Figure 5 shows the probability distribution of non-file inodes' masses on the reference system and demonstrates this phenomenon. In the most extreme case, a single directory requires nearly 1 GiB of MDT capacity as a result of it containing over eight million child inodes.

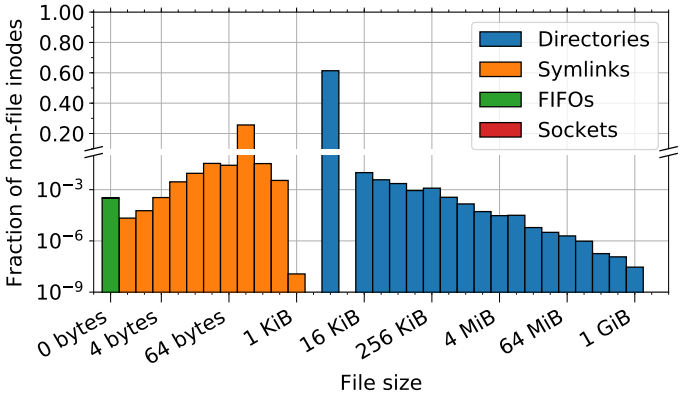

Fig. 5. Probability distribution of inode sizes on NERSC's Cori file system in January 2019. This diagram does not show block or character device inode types because none were present at the time of data collection. Break in y scale intended to contrast small numbers of large directory inodes with the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

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 6. Equation 9 demonstrates this derivation for the directory size distribution; the process is the same for all non-file inode types.

$$\boldsymbol{N}_i^{\text{dir}} = \boldsymbol{N}_i^{\text{file}} \cdot \frac{N_i^{\text{ref,dir}}}{N_i^{\text{ref,file}}} \tag{9}$$

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 10.

$$\boldsymbol{C}^{\text{inode}} = \sum_i \left[\max\left(\boldsymbol{S}_i, 4096\right) \cdot \sum_j \boldsymbol{N}_i^j \right] \tag{10}$$

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.

5.3 Overall MDT capacity

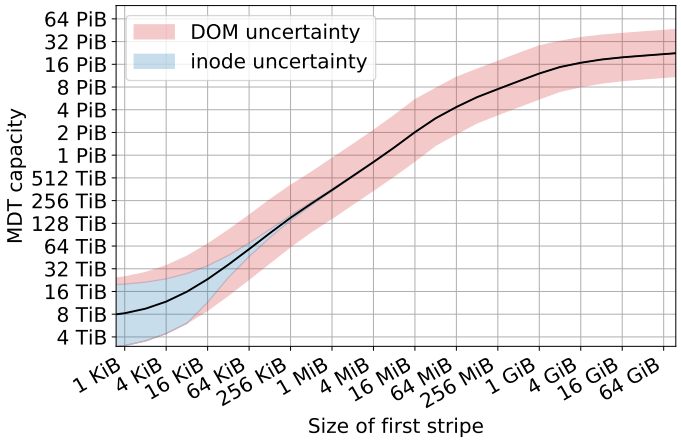

Fig. 6. Required MDT capacity as a function of S_0. Shaded area bounded by the minimum and maximum estimated requirements dictated by the DOM component and the inode capacity component of MDT capacity. Note that, in practice, the DOM component size cannot be smaller than the minimum Lustre stripe size. At the time of writing, the minimum DOM component size is 64 KiB.

We now evaluate $C_{\text{min}}^{\text{MDT}}$, $C_{\text{max}}^{\text{MDT}}$, and $C_{\text{avg}}^{\text{MDT}}$ as a function of the DOM threshold size S_0 using Equation 4. Figure 6 shows the result of this model for a target capacity $C^{\text{new}} = 30$ PB from Section 3. The shaded regions bound $C^{\text{MDT,min}}$ and $C^{\text{MDT,max}}$, and the black line is $C^{\text{MDT,avg}}$. 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 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 $S_0 \to C^{\mathrm{new}}$ 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.

6 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

The authors would like to thank John Bent, Andreas Dilger, and the anonymous reviewers for their valuable feedback on this work. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-05CH11231. This research used resources and data generated from resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

1. APEX Workflows Whitepaper. Tech. rep., Los Alamos National Laboratory, Lawrence Berkeley National Laboratory, and Sandia National Laboratories (2016), https://www.nersc.gov/assets/apex-workflows-v2.pdf
2. Intel SSD Data Center Tool (2017), https://www.intel.com/content/www/us/en/support/articles/000006289
3. Alewijnse, B., Ashton, A.W., Chambers, M.G., Chen, S., Cheng, A., Ebrahim, M., Eng, E.T., Hagen, W.J., Koster, A.J., López, C.S., Lukoyanova, N., Ortega, J., Renault, L., Reyntjens, S., Rice, W.J., Scapin, G., Schrijver, R., Siebert, A., Stagg, S.M., Grum-Tokars, V., Wright, E.R.,

Wu, S., Yu, Z., Zhou, Z.H., Carragher, B., Potter, C.S.: Best practices for managing large CryoEM facilities. Journal of Structural Biology **199**(3), 225–236 (sep 2017). https://doi.org/10.1016/j.jsb.2017.07.011, https://linkinghub.elsevier.com/retrieve/pii/S1047847717301314

4. Austin, B., Daley, C., Doerfler, D., Deslippe, J., Cook, B., Friesen, B., Kurth, T., Yang, C., Wright, N.J.: 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). pp. 63–71. IEEE (nov 2018). https://doi.org/10.1109/PMBS.2018.8641549, https://ieeexplore.ieee.org/document/8641549/
5. Bhimji, W., Bard, D., Burleigh, K., Daley, C.S., Farrell, S., Fasel, M., Friesen, B., Gerhardt, L., Liu, J., Nugent, P., Paul, D., Porter, J., Tsulaia, V.: Extreme I/O on HPC for HEP using the Burst Buffer at NERSC. Journal of Physics: Conference Series **898**, 082015 (oct 2017). https://doi.org/10.1088/1742-6596/898/8/082015, https://iopscience.iop.org/article/10.1088/1742-6596/898/8/082015
6. Bhimji, W., Bard, D., Romanus, M., Paul, D., Ovsyannikov, A., Friesen, B., Bryson, M., Correa, J., Lockwood, G.K., Tsulaia, V., Byna, S., Farrell, S., Gursoy, D., Daley, C.S., Beckner, V., Straalen, B.V., Trebotich, D., Tull, C., Weber, G., Wright, N.J., Antypas, K., Prabhat: Accelerating Science with the NERSC Burst Buffer Early User Program. In: Proceedings of the 2016 Cray User Group. London (2016), https://cug.org/proceedings/cug2016_proceedings/includes/files/pap162.pdf
7. Daley, C.S., Ghoshal, D., Lockwood, G.K., Dosanjh, S., Ramakrishnan, L., Wright, N.J.: Performance characterization of scientific workflows for the optimal use of Burst Buffers. Future Generation Computer Systems (dec 2017). https://doi.org/10.1016/j.future.2017.12.022, http://linkinghub.elsevier.com/retrieve/pii/S0167739X16308287
8. Declerck, T.M.: Using Robinhood to Purge Data from Lustre File Systems. In: Proceedings of the 2014 Cray User Group. Lugano, CH (2014), https://cug.org/proceedings/cug2014_proceedings/includes/files/pap157.pdf
9. Fontana, R.E., Decad, G.M.: Moore's law realities for recording systems and memory storage components: HDD, tape, NAND, and optical. AIP Advances **8**(5), 056506 (may 2018). https://doi.org/10.1063/1.5007621, http://aip.scitation.org/doi/10.1063/1.5007621
10. Gunasekaran, R., Oral, S., Hill, J., Miller, R., Wang, F., Leverman, D.: Comparative I/O workload characterization of two leadership class storage clusters. In: Proceedings of the 10th Parallel Data Storage Workshop (PDSW'15). pp. 31–36. ACM Press, New York, New York, USA (2015). https://doi.org/10.1145/2834976.2834985, http://dl.acm.org/citation.cfm?doid=2834976.2834985
11. Hemmert, K.S., Glass, M.W., Hammond, S.D., Hoekstra, R., Rajan, M., Vigil, M., Grunau, D., Lujan, J., Morton, D., Nam, H.A., Peltz, P., Torrez, A., Wright, C., Dawson, S.: Trinity: Architecture and Early Experience. In: Proceedings of the 2017 Cray User Group (2017)
12. John Bent, Brad Settlemyer, Gary Grider: Serving Data to the Lunatic Fringe: The Evolution of HPC Storage. ;login: **41**(2), 34–39 (2016), https://www.usenix.org/publications/login/summer2016/bent
13. Joubert, W., Weighill, D., Kainer, D., Climer, S., Justice, A., Fagnan, K., Jacobson, D.: 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. pp. 57:1–57:14. SC '18, IEEE Press, Piscataway, NJ, USA (2018), http://dl.acm.org/citation.cfm?id=3291656.3291732
14. Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., Mahesh, A., Matheson, M., Deslippe, J., Fatica, M., Prabhat, Houston, M.: Exascale deep learning for climate analytics. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis. pp. 51:1–51:12. SC '18, IEEE Press, Piscataway, NJ, USA (2018), http://dl.acm.org/citation.cfm?id=3291656.3291724, arXiv:1810.01993
15. Lockwood, G.K., Hazen, D., Koziol, Q., Canon, S., Antypas, K., Balewski, J., Bathaser, N., Bhimji, W., Botts, J., Broughton, J., Butler, T.L., Butler, G.F., Cheema, R., Daley, C.S., Declerck, T., Gerhardt, L., Hurlbert, W.E., Kallback-Rose, K.A., Leak, S., Lee, J., Lee, R., Liu, J., Lozinskiy, K., Paul, D., Prabhat, Snavely, C., Srinivasan, J., Stone Gibbins, T., Wright, N.J.: Storage 2020: A Vision for the Future of HPC Storage. Tech. rep., Lawrence Berkeley National Laboratory, Berkeley, CA (2017), https://escholarship.org/uc/item/744479dp
16. Lockwood, G.K., Wagner, R., Tatineni, M.: Storage utilization in the long tail of science. In: Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure (2015). https://doi.org/10.1145/2792745.2792777, http://dl.acm.org/citation.cfm?id=2792777
17. Regier, J., Pamnany, K., Fischer, K., Noack, A., Lam, M., Revels, J., Howard, S., Giordano, R., Schlegel, D., McAuliffe, J., Thomas, R., Prabhat, .: Cataloging the visible universe through bayesian inference at petascale. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 44–53 (May 2018). https://doi.org/10.1109/IPDPS.2018.00015
18. Standish, K.A., Carland, T.M., Lockwood, G.K., Pfeiffer, W., Tatineni, M., Huang, C.C., Lamberth, S., Cherkas, Y., Brodmerkel, C., Jaeger, E., Smith, L., Rajagopal, G., Curran, M.E., Schork, N.J.: Group-based variant calling leveraging next-generation supercomputing for large-scale whole-genome sequencing studies. BMC Bioinformatics **16**(1), 304 (dec 2015). https://doi.org/10.1186/s12859-015-0736-4, http://dx.doi.org/10.1186/s12859-015-0736-4 http://www.biomedcentral.com/1471-2105/16/304
19. Strande, S.M., Cicotti, P., Sinkovits, R.S., Young, W.S., Wagner, R., Tatineni, M., Hocks, E., Snavely, A., Norman, M.: 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. pp. 3:1–3:8. XSEDE '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2335755.2335789, http://doi.acm.org/10.1145/2335755.2335789
20. Thayer, J., Damiani, D., Ford, C., Gaponenko, I., Kroeger, W., O'Grady, C., Pines, J., Tookey, T., Weaver, M., Perazzo, A.: Data systems for the Linac Coherent Light Source. Journal of Applied Crystallography **49**(4), 1363–1369 (aug 2016). https://doi.org/10.1107/S1600576716011055, http://scripts.iucr.org/cgi-bin/paper?S1600576716011055
21. Uselton, A.: Deploying Server-side File System Monitoring at NERSC. In: Proceedings of the 2009 Cray User Group (2009)
22. Vazhkudai, S.S., Miller, R., Tiwari, D., Zimmer, C., Wang, F., Oral, S., Gunasekaran, R., Steinert, D.: 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). https://doi.org/10.1145/3126908.3126946, http://dl.acm.org/citation.cfm?doid=3126908.3126946

23. Wang, F., Sim, H., Harr, C., Oral, S.: 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. pp. 37–42. ACM Press, New York, New York, USA (2017). https://doi.org/10.1145/3149393.3149399, http://dl.acm.org/citation.cfm?doid=3149393.3149399
24. Weeks, N.T., Luecke, G.R.: Optimization of SAMtools sorting using OpenMP tasks. Cluster Computing **20**(3), 1869–1880 (2017). https://doi.org/10.1007/s10586-017-0874-8