Revisiting I/O Behavior in Large-Scale Storage Systems: The Expected and the Unexpected

Tirthak Patel
Northeastern University

Suren Byna
Lawrence Berkeley National Laboratory

Glenn K. Lockwood
Lawrence Berkeley National Laboratory

Devesh Tiwari
Northeastern University

Abstract

Large-scale applications typically spend a large fraction of their execution time performing I/O to a parallel storage system. However, with rapid progress in compute and storage system stack of large-scale systems, it is critical to investigate and update our understanding of the I/O behavior of large-scale applications. Toward that end, in this work, we monitor, collect and analyze a year worth of storage system data from a large-scale production parallel storage system. We perform temporal, spatial and correlative analysis of the system and uncover surprising patterns which defy existing assumptions and have important implications for future systems.

CCS Concepts

• Information systems → Distributed storage; **• Software and its engineering** → *Ultra-large-scale systems.*

ACM Reference Format:
Tirthak Patel, Suren Byna, Glenn K. Lockwood, and Devesh Tiwari. 2019. Revisiting I/O Behavior in Large-Scale Storage Systems: The Expected and the Unexpected. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3295500.3356183

1 Introduction

Large-scale applications typically spend a significant fraction of their execution time performing I/O (e.g., checkpointing and analysis output). While the compute characteristics of large-scale HPC applications are very well-studied, the I/O behavior of large-scale applications does not receive the same level of attention. In the past, some HPC facilities have attempted to address this problem by sharing best operational practices [10, 11, 19, 39], analyzing I/O workload characteristics [30, 33–35, 52, 59], and performing controlled experiments on a large-scale parallel storage system [36, 55, 56, 58]. However, a knowledge gap still exists in terms of understanding the most recent trends in I/O characteristics at the back end of the storage system and their implication for system design and operations. Bridging this gap is challenging because large-scale HPC storage systems are complex and difficult to manage, and hence, it is hard to monitor, collect, and accurately analyze I/O data to understand the characteristics of the system and applications - to identify and remove the sources of inefficiency [3, 8, 41].

To address this challenge and revisit the widely-held beliefs about I/O behavior of large-scale applications, this study performs measurement and systematic analysis of a year's worth of I/O activity data from National Energy Research Scientific Computing Center's (NERSC) HPC facility. This I/O activity data is collected during the year 2018 for all the Object Storage Servers (OSS), Object Storage Targets (OST), Meta Data Server (MDS), and Meta Data Target (MDT) for the Lustre parallel storage system at NERSC shared by Edison and Cori supercomputers. This rich data enables us to investigate patterns in time and space dimensions: from minute to month granularity and from one storage component in the hierarchy to all storage components concurrently. This helps us identify subtle relationships between different I/O activities and system components over time. Overall, we make the following contributions:

⋆ We develop an analysis pipeline to examine the I/O data collected from the shared parallel storage system at the NERSC HPC data center. Our pipeline includes a statistical characterization methodology to identify and analyze hidden trends in the I/O characteristics of a large-scale parallel storage system.

⋆ We investigate temporal, spatial, and correlative behavior of HPC I/O by analyzing different components of the storage system (e.g., OSTs, OSSes, and MDS). Our study uncovers surprising patterns which defy existing assumptions about HPC I/O and have important implications for future systems [3, 41].

⋆ Our analysis reveals that HPC storage systems may no longer be dominated by write I/O - challenging the long- and widely-held belief that HPC workloads are write-heavy. In fact, over the past few years, read I/O at NERSC has grown to surpass write I/O by a margin, even after accounting for burst-buffer writes!

⋆ We confirm the conventional wisdom that HPC I/O is usually bursty, but we show that write I/O is more bursty than read I/O. Moreover, while HPC I/O activity does not show diurnal patterns, write I/O shows high variance during evening hours and weekends – identifying such periods helps prevent scheduling of I/O-interference-sensitive jobs during these times.

⋆ Our study discovers that there is a huge load imbalance across OSTs in terms of I/O activity even at long time-scales, and hence, tools that perform intelligent file migration across OSTs are highly desirable for large-scale parallel storage systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC '19, November 17–22, 2019, Denver, CO, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356183

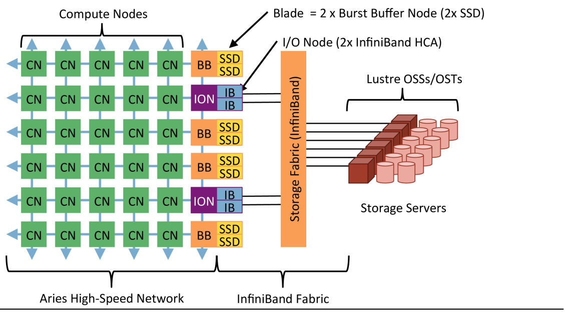


Figure 1: Cori supercomputing system overview with an SSD-based Cray DataWarp burst-buffer and a Lustre file system [5].

★ Our analysis uncovers that even large-scale HPC applications do not tend to take advantage of I/O parallelism, often using less than 10 OSTs concurrently even during high intensity I/O phases. This points toward the key reason for why HPC applications often observe very small fraction of the peak I/O bandwidth offered by large scale HPC storage systems.

★ We discover that the OSSes, which are typically as powerful as the compute nodes, are often idle and have very low CPU utilization. Carefully designed analytic tools can opportunistically steal these idle cycles to perform in-situ data analysis and file system verification tasks without interrupting I/O activities.

★ Our extensive temporal and spatial correlation analysis identifies correlated storage components and I/O activities (e.g., statistical characteristics of I/O activity periods, correlation between read and write I/O at the OST and the system level, etc.). These findings can be leveraged for designing intelligent I/O scheduling techniques that can predict and mitigate I/O contention in HPC storage systems by coordinating the I/O intensive phases of different applications at both, the system and the OST level.

2 Background

In this paper, we analyze the logs of a Lustre parallel file system which is accessed by two supercomputing systems deployed at NERSC, for the entire year of 2018. In this section, we briefly discuss the configurations of these systems, the Lustre file system, and the data monitoring and collection methodology.

2.1 System Architecture

The current flagship supercomputer at NERSC, named Cori, is a Cray XC40 system with two computation partitions: the first consists of energy efficient 9,688 68-core Intel Xeon Phi (Knights Landing or KNL) processors and the second consists of 2,388 16-core Intel Xeon (Haswell) processors. As shown in Fig. 1, an SSD-based Cray DataWarp burst-buffer storage layer is available between the compute nodes and the disk-based Lustre file system. The Lustre file system is composed of ≈10,000 disks organized as 248 Lustre Object Storage Targets (OST). Each OST has a corresponding Object Storage Server (OSS) which manages the I/O requests. The file system also has a Meta Data Server (MDS) and a Meta Data Target (MDT) to perform I/O metadata operations. The total size of the file system is ≈30 PB with an aggregate peak I/O bandwidth of 700 GB/s. This Lustre file system is also shared by another system, a Cray XC30 system named Edison, which has 12-core processors on each of its 5,586 nodes. Edison has a local Lustre file system as well, but in this paper, we study the logs of the shared Lustre file system.

We note this study does not specifically focus on burst-buffer I/O activities since the burst-buffer read and write activities are still quite limited (5-15%) and the shared scratch space continues to observe almost all the I/O traffic. However, for careful and complete analysis, we include burst-buffer I/O activity at certain places where our findings may have interactions with the burst-buffer activities.

2.2 Data Monitoring and Collection

The log data of Lustre is obtained by the Lustre Monitoring Tool (LMT) [53], which is a distributed system to provide Lustre server-side activity on various server nodes, similar to the Unix "top" command. LMT monitors the I/O activity of the OSSes, OSTs, MDS, and MDT and retains these data for the preceding 24 hours in a MySQL database. A separate service queries this MySQL database and archives the data from the previous day into an HDF5 file before it is expired from the MySQL database. Each HDF5 file consists of datasets which contain performance statistics such as CPU utilization of OSSes, file operations of MDSes, and the read/write transfer rates on the OSTs. Note that the LMT does not report the number and size of read/write I/O requests served by the OSSes.

The logger generates one HDF5 file for every day of the year; each file consists of 17280 entries as performance statistics are captured every 5 seconds. Note that each entry is not expected to accurately represent the state of the system during the previous 5-second interval. In cases of network congestion, the User Datagram Protocol (UDP) packets used to communicate information about the individual LMT devices to the LMT "watcher" often get accumulated in the entry when they reach the LMT watcher. This can cause several consecutive entries to report a value of zero, followed by an entry with large value. In order to filter out the noise generated by this phenomenon, we use a 1-minute interval which consists of 12 entries of 5-second intervals. We found that using a 1-minute interval substantially equalizes entries which are unexpectedly large (greater than the system's peak limit). Using a 1-minute interval is suitable for our analysis as we focus on larger trends which are independent of performance statistics at per-second granularity.

These monitoring data can contain corrupted or anomalous data points. Such undesirable data points are identified via performing simple sanity checks that verify that physical system constraints are not violated at each sample (e.g., theoretical data transfer bandwidth at each OST level, at the system-level, CPU utilization, etc.) We have ensured that our analysis is not affected by such data points. We also ensure that data points corresponding to system shutdown periods are appropriately handled and do not bias the analysis.

Finally, we note that server-side I/O logs, by design, do not contain client-side information such as application name, job id, or user information. Therefore, it limits our study to perform correlation between server-side and corresponding user-level information. Performing such a correlation while useful is quite challenging as it would require instrumenting the applications and then doing accurate correlation between records with potentially different time-stamps and without exact spatial information of I/O activities.

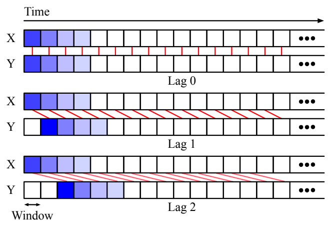


Figure 2: Capturing "lag" in correlation between two variables: Illustration of lag and window when calculating correlation between two random variables X and Y. "Lag" refers to the amount by which Y is shifted in time when correlating with X and "window" refers to the length of the considered time window (a single time window may have one or more samples).

2.3 Statistical Methods

This study derives insights by observing trends and substantiates these insights with statistical methods. In particular, we use the following statistical methods to support our findings: (1) Probability (PDF) and Cumulative (CDF) Density Function, (2) Coefficient of Variance (CoV), and (3) Coefficient of Correlation (CC).

Probability Density Function (PDF) and Cumulative Density Function (CDF): The PDF and CDF statistically represent the distribution of the variable under study (e.g., amount of data read) over a period of time. PDF is represented as a histogram and CDF is presented as a curve. The sum of all bars in the histogram is 100% and the ceiling of the CDF curve is 100%.

Coefficient of Variance (CoV): PDF and CDF are useful in representing the distribution, but they do not directly quantify the variance among the collected samples (e.g., data transferred at 1-minute granularity). While standard deviation can be used to characterize the spread among the collected samples, it is biased by the value of the mean. This is why we use the Coefficient of Variance (CoV) metric. CoV is the measure of the amount of variance in the random variable distribution normalized by the mean of the distribution. If the mean of the variable is μ and its variance is σ^2 (standard deviation is σ), then CoV (in %) = $\frac{\sigma}{\mu} \times 100$. A high CoV value indicates high variance.

Coefficient of Correlation between different variables: While the characteristics of individual variables can be captured by statistical metrics such as mean, standard deviation, and CoV, we need statistical methods to capture relationships between different variables. In this paper, we analyze different types of I/O activities (e.g., read and write), which may be correlated. Therefore, we use the Coefficient of Correlation (CC) metric to determine if there is a linear relationship between two random variables. For two random variables, X and Y sampled n times with means μ_X and μ_Y and standard deviations σ_X and σ_Y, respectively, the CC is defined as:

$$\frac{\sum_{i=1}^{n}(X_i - \mu_X)(Y_i - \mu_Y)}{\sigma_X * \sigma_Y}$$

The value of CC can range between -1 and 1. A value of 1 indicates high positive correlation (value of i^{th} sample of Y increases/decreases in perfect proportion to increase/decrease in value of i^{th} sample of X), a value of -1 indicates high negative correlation (value of i^{th} sample of Y decreases/increases in perfect proportion to increase/decrease in value of i^{th} sample of X), and a value of 0 indicates no correlation.

The previous method is effective in capturing the correlation between two variables at every time step. However, it is possible that two variables are correlated with some lag. For example, read activity on a particular OST at a particular instance might increase the read activity on a nearby OST in future but not at the same instance. We use the concept of "lag" to capture these potential delayed correlations. The "lag" refers to the number of samples by which the second random variable is shifted when calculating the CC. For a lag of k, the CC of X and Y is defined as:

$$\frac{\sum_{i=1}^{n-k}(X_i - \mu_X(1:n-k))(Y_{i+k} - \mu_Y(1+k:n))}{\sigma_X(1:n-k) * \sigma_Y(1+k:n)}$$

We note that μ and σ are calculated only for the indicated sample range. The lag helps detect if there is a temporal relationship between the two random variables, i.e., does an increase/decrease in i^{th} sample of X lead to an increase/decrease in the $(i+k)^{th}$ sample of Y (as opposed to comparing with the i^{th} sample of Y).

Fig. 2 provides a visual representation of the concept of "lag" and how it can be used to detect delayed temporal correlations. The "window" refers to the granularity of the considered time window. For example, to observe trends over larger time-scales, read activity can be accumulated at a 5-minute granularity, even though the samples are collected every minute, and then correlated with write activity at a 5-minute granularity. If the "window" is 5 minutes, and the variables are highly correlated with a lag of two, this implies the correlation is lagging by two windows (i.e., 10 minutes).

Auto-correlation: When a random variable X is correlated with itself (i.e. $Y = X$), it is referred to as autocorrelation. Autocorrelation helps detect temporal relationship of the random variable with itself i.e., does an increase/decrease in i^{th} sample of X lead to an increase/decrease in the $(i+k)^{th}$ sample of X. Note that autocorrelation of a random variable with a lag of 0 is always 1.

3 Overall System-Level I/O Behavior

In this section, we study the I/O behavior at the system-level by analyzing the trends in read and write I/O. First, we compare the total *amount of data transferred* for read and write I/O activities. Then, we compare the *rate of data transfer* of read and write activities.

Total Amount of Data Transferred: Our analysis reveals that read activity is significantly higher than write activity on the large-scale storage system under study (Fig. 3). In one full year, more than 400 PiB of data is read, while the amount of data written is less than 230 PiB. Cumulative read activity is 1.75× the cumulative write activity. We note that this trend is consistent across the whole

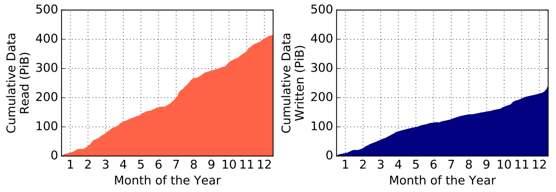


Figure 3: Total data read is 1.75× the total data written in 2018.

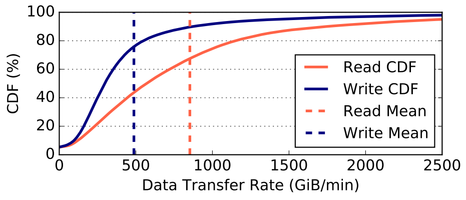


Figure 4: Read and write data transfer rates vary significantly.

period and is not an artifact of dramatic rise in read activity during a short period of time.

Traditionally HPC storage systems are designed and built for write-intensive activity (such as checkpoint and analysis output) [38, 39]. Recently published papers have shown that HPC systems are mostly dominated by write-intensive workloads. Even though read activity can be substantial, it is still lower compared to write activity (less than 45%) [4, 19, 23, 38, 58]. Darshan-tool based I/O monitoring and characterization had revealed that some Darshan-instrumented applications can be significantly read-intensive [10], but previous works have not shown that at the aggregate-level, read activity is likely be to be higher than the write activity by such a significant factor (1.75×). *In 2014, the average read and write volumes on Edison (NERSC's then largest supercomputer) were 139 TB/day and 303 TB/day, respectively [4]. The write volume was over 2× the read volume then. In comparison, we have shown that average read and write volumes in 2018 (at Cori and Edison) were 1200 TB/day (8.5× increase since 2014) and 685 TB/day (only 2.3× increase since 2014), respectively. In this context, our finding is critical for future storage system design as it shows that HPC workloads are no longer dominated by write activity; in fact, read activity is significantly higher than write activity, at least for this particular large-scale HPC storage system.*

Data Transfer Rate: In Fig. 4, we show the CDF of data transfer rate per minute for both read and write activities. Note that the data transfer rate is reported as GiB/minute instead of the traditional GiB/second metric because our sample granularity is 1 minute. Hence, this result is not termed as read/write bandwidth, but instead referred as data transfer rate, although the data transfer rate may indeed be impacted by the observed I/O bandwidth from the system.

We first observe that the average amount of data transferred during read activity is higher than the write activity (853 GiB/min vs. 487 GiB/min). This is consistent with our previous observation (Fig. 3), where the read activity is indeed 1.75× the write activity. As noted by other studies [4, 35], the absolute data transfer rate for this system also appears to be well below the peak I/O bandwidth (700GB/second). *One may argue that this observation is a side-effect of having burst-buffers in the storage hierarchy which arrest most of the writes and perhaps, not all writes are drained to the scratch space.* However, we found that this is not the case for the current NERSC system. Over the full year 2018, the burst-buffers observed less than 30 PiB read data and less than 40 PiB write data overall. Even when the read and write traffic to the burst-buffer is considered in the calculation, the overall read activity still remains significantly higher than the write activity (≈1.9×).

Second, the difference in distribution shape between read and write activities shows that the writes have more variance. While more than 75% of write samples have data transfer rates less than 487 GiB/min, the rest have much higher data transfer rates, which

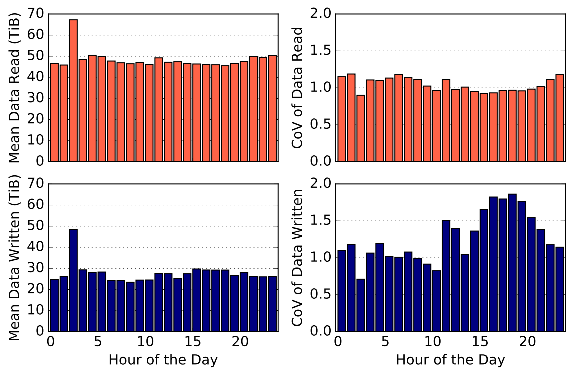


Figure 5: Hour-of-the-day behavior of reads and writes.

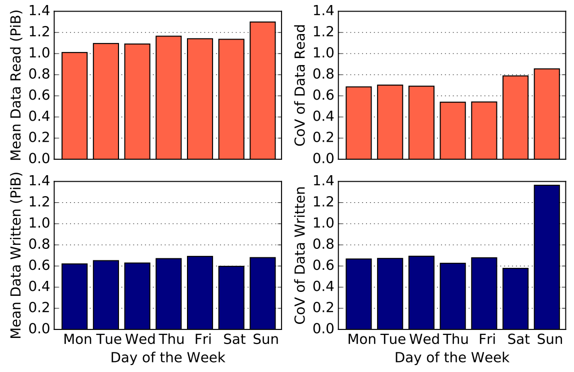


Figure 6: Day-of-the-week behavior of reads and writes.

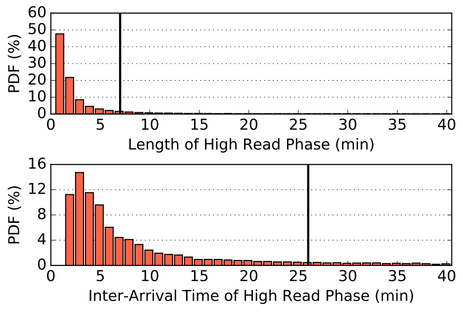
causes the high variation. To substantiate this further, we calculated the CoV. The normalized variation of write activity is indeed higher. The read activity CoV is 172% while write activity CoV is 245%; there is a 73% point difference. In other words, most of the time, the write activity is quite low compared to read activity; however, rest of the time, the write activity is uncharacteristically high – potentially exhibiting a bursty behavior. Prior works have shown that HPC I/O workloads are bursty [19, 28, 29, 39], our analysis confirms this and provide new evidence that writes have much higher variance and "burstiness" than reads.

Observation: *Production HPC storage systems may no longer be dominated by write-intensive workloads. Surprisingly, read I/O activity is approx. 1.75× the write I/O activity. While both read and write I/O activity on parallel HPC storage systems continue to be bursty [28, 39], our new evidence suggests that writes have much higher variance and burstiness than reads.*

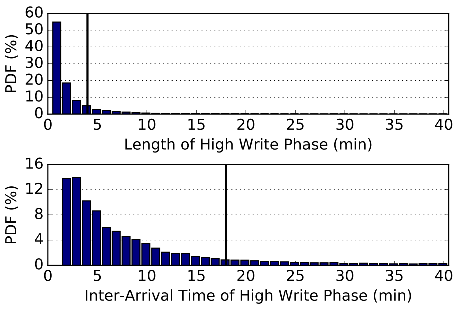
4 Temporal I/O Characteristics

Next, we characterize and analyze the temporal behavior of I/O activity at the system level. First, we characterize the I/O behavior at the hour-of-day and day-of-week granularity to discover diurnal and other related I/O patterns. Then, we classify the I/O activity in terms of intensity and quantify the I/O characteristics during I/O periods of different intensity.

The left column in Fig. 5 shows the mean amount of data read and written for each hour-of-the-day (accumulated over the entire year). We observe that both read and write activity are fairly evenly



(a) High Read Activity

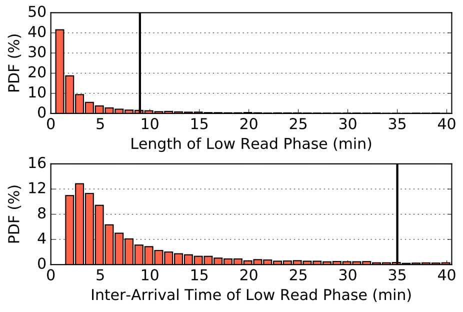


(b) High Write Activity

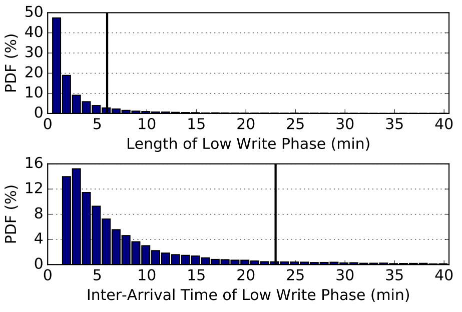
Mean Length of High Read Phase	6.62 minutes
Mean Inter-Arrival Time of High Read Phase	26.46 minutes
Mean Length of High Write Phase	4.38 minutes
Mean Inter-Arrival Time of High Write Phase	17.55 minutes
Std. Dev. of Length of High Read Phase	33.11 minutes
Std. Dev. of Inter-Arrival Time of High Read Phase	150.60 minutes
Std. Dev. of Length of High Write Phase	23.08 minutes
Std. Dev. of Inter-Arrival Time of High Write Phase	118.49 minutes

(c) Statistics of High I/O Phases

Figure 7: Characteristics of high read and write I/O phases. An I/O phase is considered to have "high" intensity if it falls in the top 25% of all I/O activity during the year. High read phases last longer than high write phases, and are less frequent than high write phases.



(a) Low Read Phases



(b) Low Write Phases

Mean Length of Low Read Phase	8.71 minutes
Mean Inter-Arrival Time of Low Read Phase	34.84 minutes
Mean Length of Low Write Phase	5.73 minutes
Mean Inter-Arrival Time of Low Write Phase	22.92 minutes
Std. Dev. of Length of Low Read Phase	153.81 minutes
Std. Dev. of Inter-Arrival Time of Low Read Phase	191.73 minutes
Std. Dev. of Length of Low Write Phase	125.03 minutes
Std. Dev. of Inter-Arrival Time of Low Write Phase	147.07 minutes

(c) Statistics of Low I/O Phases

Figure 8: Characteristics of low read and write I/O phases. An I/O phase is considered to have "low" intensity if it falls in the bottom 25% of all I/O activity during the year. Low read phases last longer than low write phases, but are less frequent than low write phases.

distributed over all hours-of-the-day and there are no significant diurnal patterns. This is in contrast to a previous work which has noted that read and write I/O activities change considerably during the day [23]. However, leadership-scale HPC systems like Cori and Edison have high resource utilization at all times and hence, are not likely to show idle periods. The only exception is the spike at 2am in both read and write activities due to system jobs that run in the background at night-time every day.

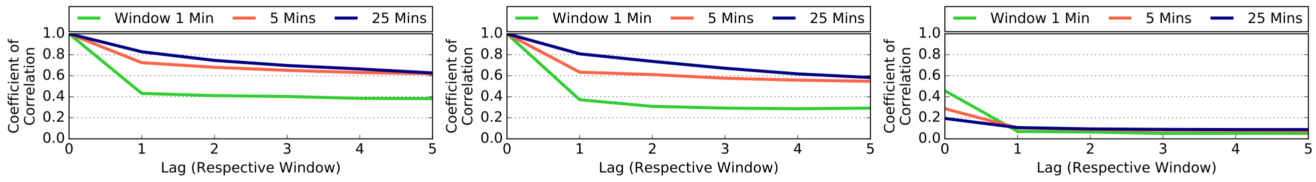
While the cumulative data written and read do not show diurnal patterns, interestingly, the variability in write I/O activity does. The right column in Fig. 5 shows that the CoV is roughly the same for read activity over hours-of-the-day, but the CoV for the write activity is much higher during the evening hours. This indicates that although the total amount of data written during evening hours is the same as the morning hours, the variability is much higher (e.g., some evenings might observe much lower write I/O than others).

Day-of-the-week characteristics (Fig. 6) reveal that both read and write I/O activities are roughly consistent across days. This behavior is expected since HPC systems have more than 90% utilization most of the time and are not idle during the weekends. However, we note that the variability in the total amount of data written is much higher on Sundays compared to other days of the week.

Observation: *While, average I/O activity does not have diurnal or weekly patterns, write activity is much more variable during evening hours and on Sundays. Therefore, interference-sensitive, QoS-aware and reproducibility-aware jobs [22, 37] should not be scheduled during evening hours and on weekends when the variability is higher.*

Now, we investigate the temporal characteristics of I/O by analyzing the I/O behavior at finer time granularity. To study this systematically, we classify the I/O activity in terms of intensity and quantify the I/O characteristics during I/O periods of different intensities. We analyze the behavior of system I/O activity at 1-minute granularity and differentiate between when the intensity of the system I/O activity is "high" and "low". We categorize samples which fall in the third quartile (75th percentile and above) as high I/O activity and samples which are in the first quartile (below 25th percentile) as low I/O activity. We focus on only these two extreme periods because they are most interesting ones in terms of job scheduling, contention mitigation, and provisioning decisions. Fig. 7 and Fig. 8 show the I/O behavior during high and low I/O activity phases, respectively. Each figure shows the behavior for both types of I/O. We focus on two characteristics: (1) mean length of the high and low I/O phases, and (2) time between two such phases (i.e., inter-arrival time between two I/O phases of the same type). We make several interesting observations from these plots.

High intensity I/O activity phase: First, we observe that the mean length of high read I/O phase is 50% longer than the mean length of high write I/O phase (6.62 minutes vs. 4.38 minutes) (Fig. 7). This indicates that when the system goes in the high read I/O phase, it stays in that phase longer than when it enters the high write I/O



(a) Read I/O Autocorrelation **(b)** Write I/O Autocorrelation **(c)** Read and Write I/O Correlation

Figure 9: At the aggregate system level, reads and writes exhibit a high degree of auto-correlation with 5 and 25 minute time windows, but not at very short (1 minute) time windows. Reads and writes are only weakly correlated to each other within the same minute of each other and much weakly correlated at longer time windows.

phase. The mean inter-arrival time of high read I/O phase is also about 50% longer than the mean inter-arrival time of high write I/O phase (26.46 minutes vs. 17.55 minutes).

Comparing mean inter-arrival time and mean length of high I/O activity phase shows that, interestingly, writes are more bursty than reads. Write activity tends to enter the high activity phase more frequently but when write activity enters the high activity phase, it stays in that phase for a shorter time. This is in agreement with our observation in Sec. 3 about writes having larger variation in data transfer rates and being more bursty.

Low intensity I/O activity phase: Fig. 8 shows that the mean length of low I/O activity phase for reads is longer than writes – which resembles our observation for the high I/O activity phase (Fig. 7). Similarly, the mean inter-arrival time of low read I/O phase is higher than the mean inter-arrival time of low write I/O phase. *These two findings, when combined, indicate that the write activity enters the low activity phase more frequently, but stays in that phase for shorter time compared to read activity.*

Next, comparing Fig. 7 and Fig. 8, we find that (1) the mean inter-arrival time of low (read and write) I/O phase is higher than the mean inter-arrival time of high (read and write) I/O phase, and therefore (2) the mean length of low (read and write) I/O phase is typically more than the mean length of high (read and write) I/O phase. This implies that the system goes in the high I/O activity phase more often and stays in that phase for shorter times as compared to the low I/O activity phase.

Observation: *The system enters the high I/O activity phase more often than the low I/O activity phase. But, the system stays in the low I/O activity phase for longer duration than the high I/O activity phase. In other words, the system is less likely to enter dull phase, but when it does, it stays in that phase for longer duration. The distribution and characteristics of these phases can guide intelligent interference-aware scheduling of "data movement" to/from the scratch file system and job scheduling in a multi-tier storage system [21, 46, 49].*

5 Correlation Between Read and Write I/O Activity at the System Level

As shown in Fig. 7, periods of high intensity I/O can last up to multiple minutes. This naturally leads us to investigate if read and write activities have predictability in the short-term. For example, if one observes read activity for a few minutes, can it be used as a predictor to indicate that read activity will last for the next several minutes. This can be quantified by answering the following question: *"is the read activity auto-correlated with itself?"*. Therefore, we perform three tests: (1) read I/O auto-correlation, (2) write I/O auto-correlation, and (3) correlation between reads and writes.

As discussed earlier, to perform correlation analysis, we use the concepts of *window* and *lag*. A window refers to the time period during which an activity is observed and lag is used to capture correlation between two variables that might be correlated with some lag. By definition, a variable is perfectly auto-correlated with itself at zero lag (i.e., coefficient of correlation (CC) is equal to 1.0). If two variables are correlated but with some lag (say, 20 minutes), then the CC will be high at lag = 4, if the chosen window length is equal to 5 minutes. Alternatively, coefficient of correlation will be high at lag = 2, if the chosen window length is equal to 10 minutes.

We make several interesting observations from Fig. 9. First, Fig. 9(a) and (b) show that read and write I/O are not highly auto-correlated at the system level when the time window is 1 minute, but the auto-correlation is significantly high when the time-window is 5 minutes or more. In other words, simply observing read/write activity at 1-minute granularity is not a good indicator of subsequent read/write activity over next several minutes. However, read/write activity at a longer time period granularity (5 minutes or more) is a good indicator, that is the system is likely to see similar activity trends of the same type (read/write) over the next 30 minutes or longer. This finding can be further reasoned by observing the PDF of length of high and low I/O activity phases in Fig. 7 and Fig. 8. Length of most of high/low I/O activities phases are often only 1 or 2 minutes long. Therefore, only observing the I/O activity for a few minutes alone cannot act as a good predictor for future activities.

Next, Fig. 9(c) reveals a new insight: read and write I/O are very weakly correlated with each other for all time windows and lag units. That is, if the system is observing the read activity during a certain period, then, it cannot act as a good indicator for write activity in the same or subsequent windows. Note that individual applications can have this behavior where reads and writes are temporally interdependent, but at system-scale, read and write activities cannot be used to predict the behavior of one another.

Observation: *Observing read or write activities at the system-level at a very small time window is not useful for predicting storage system load/contention in the near future, but observing the I/O activity at the system-level for longer time windows (5 minutes or more) can act as an effective predictor for storage load/contention. This can be exploited for coordinating the schedule of I/O intensive phases from different applications; a system-level service that advises applications on scheduling I/O based on probing/coordination (such as [17, 18, 45]) should consider 5 minutes or longer for history to make decisions*

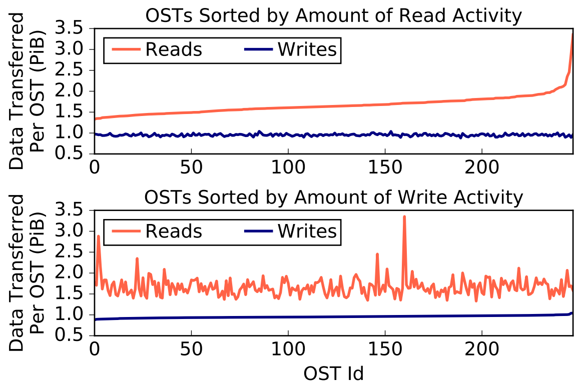


Figure 10: Load imbalance across OSTs: the total amount of data read and written per OST during the year is 1.67 PiB and 0.95 PiB on average, respectively. The amount of data read by an OST is not proportional to the amount of data written to it.

instead of traditional shorter time windows. However, read and write activities cannot act as a good predictor of future load for each other. That is, the system may be observing high read activity during a certain period, but, it cannot act as a good indicator for high anticipated write activity in the same or subsequent windows (and vice versa).

6 Spatial I/O Characteristics

In Sec. 4, we characterized and analyzed the temporal I/O behavior of the storage system. Now, we investigate the spatial I/O characteristics. In particular, we seek answers to the following three questions: ***(1)** What is the distribution of total amount of data written and read across all OSTs? **(2)** What is the parallelism during read and write activity phases? **(3)** Are read and write I/O correlated at the OST-level (i.e., given a time window, can current read or write activity on a given OST be a good indicator for future I/O on the same OST)?*

Load imbalance across OSTs: In Fig. 10, we show the total amount of data read (top) and written (bottom) per OST, sorted by the amount of I/O activity. This result reveals the following interesting trends: (1) There is much higher imbalance across OSTs in terms of data read than data written. The most read-intensive OST transferred over 2.5× more data than the least read-intensive OST. On the other hand, this difference is relatively less but significant for writes too (up to 15%). (2) Interestingly, the OSTs which are read intensively are not necessarily written intensively, and vice versa.

These findings have important implications for load-balancing I/O workload across OSTs. Note that existing MDS dispatcher capacity-balances files across OSTs. However, there is no direct support for distributing files in a manner which makes sure that each OST is load-balanced in terms of the number of I/O requests or the amount of data transferred. We also note that a write-intensive OST does not necessarily experience high amount of data read traffic and vice-versa. Therefore, load imbalance distribution across one type of I/O cannot inform us about the load imbalance distribution across the other type. Additionally, previous works have found that load imbalance across OSTs is a short-term phenomenon, and that the load balances out over longer periods [23]. However, our year-long analysis has found that even at longer time-scales, load imbalance across OSTs persists.

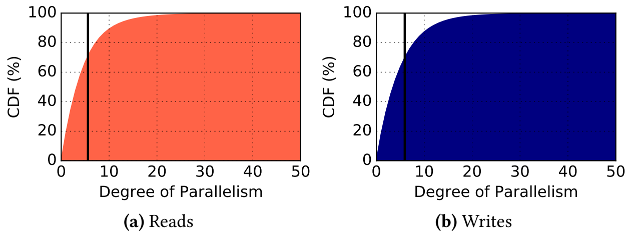
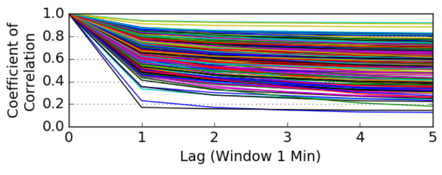


Figure 11: The degree of I/O parallelism (number of OSTs with similar read/write activity at any 1 minute time interval) is limited. The mean degree of parallelism of reads is 5.58 OSTs and that of writes is 5.93 OSTs.

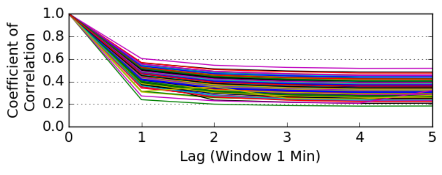
Observation: *Surprisingly, OSTs experience significant load imbalance for both read and write activities. While the MDS attempts to capacity-balance at file creation time, it does not mitigate the load imbalance of data access and write. This finding emphasizes the need to develop new techniques and production-level tools that periodically change the placement of files to mitigate the side-effects of load imbalance issues such as concentrated contention on some OSTs, disk failure, data corruption, write endurance issues [20, 27, 50, 57].*

Degree of I/O parallelism: Next, we investigate the second question: the degree of read and write parallelism during I/O activity. The degree of parallelism is defined as the number of OSTs which transfer similar amount of data during a given time interval. We use the following methodology to calculate the degree of parallelism: at each interval, the 248 OSTs are clustered based on the similarity of their read (write) activity. Clusters are formed such that within each cluster, the difference in I/O activity between the OST with the maximum read (write) activity and the minimum read (write) activity is less than 5% of the minimum read (write) activity. The size of each cluster is referred to as its *degree of parallelism.* For example, if a cluster has 5 OSTs, its degree of parallelism is 5 as it implies that there are 5 OSTs with similar I/O behavior at the same time. It is important to note that using our methodology, we are able to estimate the degree of parallelism in parallel storage systems I/O using only server-side logs, without instrumenting the applications or using client-side information.

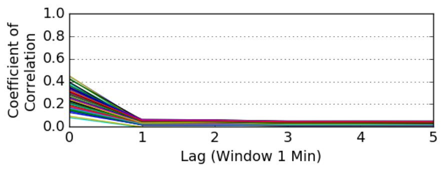
Fig. 11 plots the CDF of the degree of parallelism of all such clusters formed at all 1-minute time intervals during the year. We observe that the average degree of parallelism is fairly low for both read and write I/O. On average, less than 6 OSTs are in action simultaneously performing similar amount of data transfer. In fact, in more than 85% of the cases, less than 10 OSTs and in more than 98% of the cases, less than 20 OSTs appear to be acting in coordination. This trend is true for both reads and writes. This could be due to several reasons: (1) at NERSC, the default striping is one, and many applications might neglect to change the striping preference, (2) many applications might not use enough nodes to take advantage of the parallelism offered by the storage system, and (3) Users might be concerned that higher parallelism makes an application's I/O phase slower due to the higher statistical likelihood of encountering a straggling OST. Given that over 98% of NERSC workloads run on more than 1 core and more than 75% of them run on more than a thousand cores [4], this finding reveals a relatively less-investigated



(a) Per-OST Read Activity Autocorrelation



(b) Per-OST Write Activity Autocorrelation



(c) Per-OST Read and Write Activity Correlation

Figure 12: I/O activity correlation on individual OSTs (one line per OST). The read activity is autocorrelated on most OSTs for up to 5 minutes, while the write activity is not autocorrelated on most OSTs. Reads and writes are not correlated with each other on the same OST.

Table 1: Number of 1-minute occurrences of different degrees of I/O parallelism of reads and writes during their respective high I/O phases. Write I/O is more parallel than reads in general.

Degree of Parallelism	Number of Occurrences		
	Read I/O	Write I/O	% Difference
≥ 10	882911	1022264	16%
≥ 25	83248	105268	27%
≥ 50	5005	11138	123%
≥ 75	872	2594	198%
≥ 100	260	884	240%

shortcoming of parallel HPC applications: "HPC applications perform computation in parallel, but still do not take advantage of parallelism provided by the storage systems despite of the growing gap between the speed of compute and storage systems".

Note that our definition of degree of parallelism is optimistic since it does not require all OSTs to perform the exact same amount of data transfer in a given time interval. In fact, we found that our results and insights are not very sensitive to this range. For example, we obtained similar results even when we increased the range from 5% to 10%. We also found that increasing the time granularity to longer than 1 minute can slightly decrease the degree of parallelism since the intensity of I/O activity may not sustain over a long time duration. Note that we use a time interval of 1 minute for the reasons outlined in Sec. 2, but, it might be argued that higher degree of parallelism could be observed at smaller sampling granularity. However, prior works [36] have shown that most applications who take significant advantage of I/O parallelism either perform I/O for longer than 1 minute or transfer ample amount of data within 1 minute for the burst to be detected, and hence, have shown that a 1-minute interval is sufficient to correlate application I/O activity with system-wide I/O activity to detect I/O parallelism bursts.

We also performed deeper analysis to understand the distribution of the degree of parallelism when the system is in high I/O phase, as defined in Sec. 4, for both reads and writes during their respective high I/O phases. We found that the average degree of parallelism remains the same for read I/O (5.58), and is only slightly higher for write I/O (6.22 as compared to 5.93). Table 1 shows the number of 1-minute occurrences where the degree of parallelism is higher than 10, 25, 50, OSTs etc. We observe that during high I/O phases, writes display a higher degree of parallelism than reads. For example, we found that write I/O has 123% more occurrences of samples where the degree of parallelism is greater than 50 as compared to read I/O during high I/O phase for both reads and writes. The higher degree of parallelism for write I/O can also help explain why high write I/O phases last for shorter durations than the high read I/O phases

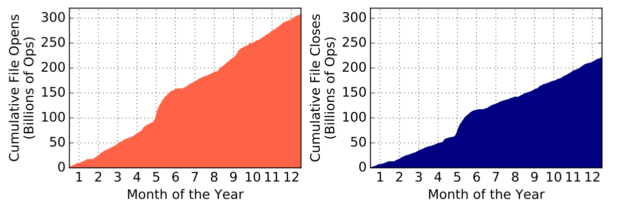


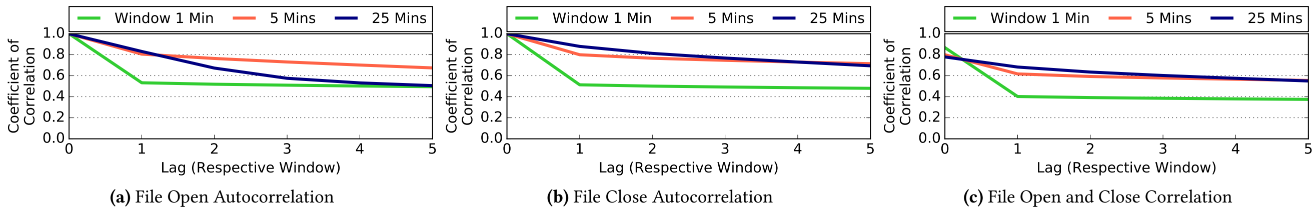
Figure 13: Cumulative number of file opens and closes during 2018. Almost 20% of opened files are not closed.

(Sec. 4). Applications are more likely to use the parallel power of storage systems when writing, but less likely to do so when reading, possibly because they read only a subset of files (that they have written to) at a time for analysis.

Observation: *Parallel HPC applications are still far from taking advantage of the parallelism provided by the parallel storage systems. The degree of I/O parallelism is very limited, less than 20 OSTs in more than 98% cases. Even during high intensity I/O phases, the degree of I/O parallelism is small, although writes appear to achieve higher parallelism than reads. This finding points to the key reason for why HPC applications often observe very small fraction of the peak I/O bandwidth offered by large scale HPC storage systems [4, 6, 7, 35, 48]. It also emphasizes the need for scaling out the I/O phase of parallel applications so that compute node cycles can be better utilized.*

Spatial autocorrelation at the OST granularity: Next, we present the spatial autocorrelation analysis for read and write I/O at per-OST level (recall that a similar analysis was performed in Sec. 5 at the whole system level, but not at per-OST granularity). Fig. 12(a) and (b) show the autocorrelation curves per OST for read and write I/O for window length of 1 minute for 5 lag units (total 248 curves on each figures).

These results show interesting and contrasting trends, read I/O at the OST level are highly autocorrelated at some OSTs but not at other OSTs. While, write I/O is typically not autocorrelated at the OST level. In other words, read I/O activity on some OSTs can be a good indicator for more read I/O requests over the next several minutes on the same OST, but this is not true for write I/O requests at the OST granularity. This is particularly noteworthy because at the whole system-level (Sec. 5) such autocorrelations at smaller intervals were very weak (Fig. 9(a) and (b)), but at the OST-level (fine-grained spatial location) there is some predictability for read I/Os. Interestingly, with longer window of 5 and 25 minutes (not shown for brevity), both reads and writes show strong correlation across OSTs for lag of 1. Contrasting trends between



(a) File Open Autocorrelation **(b)** File Close Autocorrelation **(c)** File Open and Close Correlation

Figure 14: File open and close activities are highly autocorrelated for longer windows for lags of up to 5 units. File open and close activities are correlated with each other at all window lengths for lag of 0.

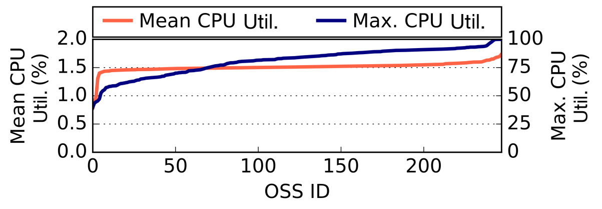


Figure 15: The mean and maximum CPU utilization of each OSS during 2018 in sorted order. Not only is the CPU utilization imbalanced across OSSes, but CPU cycles are also highly underutilized as the mean CPU utilization is less than 2% across all OSSes.

the whole system-level and OST-level can be explained when we carefully consider the degree of I/O parallelism observations. Since I/O activities are often happening at a very small number of OSTs concurrently, the overall effect when aggregated over all OSTs is less pronounced for identifying correlations at the macro-level and hence, the predictability is limited, esp. at small time windows. But, since HPC applications often perform more than just a few seconds of I/O, albeit only at a few OSTs at a time, observing these activities at the fine-grained (OST) level where the I/O is taking place can serve as a good indicator for the near-future at that particular OST.

Finally, Fig. 12(c) shows that reads and writes are not correlated at the OST-level for window length of 1 minute. These results echo the system-level results (Fig. 9(c)) which showed equally weak correlation among reads and writes. We found similarly weak correlation with longer time windows and lag units (not shown).

Observation: *As discussed earlier, read and write activities at the whole system granularity are not effective in predicting the near-term load (less than 5 minutes), but our results show and explain that at the individual OST granularity, read I/O activities in short-term can be used effectively for both near-term and slightly long-term future load prediction at a particular OST. This observation should be leveraged to mitigate and manage contention at the OST-level in both short-term and long-term (e.g., identifying and moving data at contended OSTs).*

7 MDS and OSS Utilization Analysis

Meta Data Server (MDS) and Object Storage Servers (OSS) are relatively less analyzed components in parallel HPC storage systems. In this work, we analyze the MDS load and OSS utilization behavior seeking answers to the following two specific questions: ***(1)*** *What are the characteristics of the file open and close operations on the MDS and is there any correlation among those operations?* ***(2)*** *What are the characteristics of the CPU utilization on each of the 248 OSSes?*

MDS Operations: Fig. 13 shows that users perform file open and close operations throughout the year as expected (cumulative number of files opened is more than 300 billion). However, not all files that are opened are closed. As shown in Fig. 13, roughly 20% of the files opened are not closed. While, the average number of file opens and closes is 586 thousand opens/min and 420 thousand closes/min, respectively, this number can be as high as 327 million opens/min and 235 million closes/min, respectively, during some intervals. This indicates that file opens and closes are bursty in behavior. The CoV of file opens and closes among 1-minute intervals during the year is 153% and 150%, respectively.

Next, Fig. 14 shows that, similar to read and write I/O at the system level, file open and close operations are autocorrelated at longer time windows, but not at small intervals (e.g., 1 minute). High autocorrelation in file open and close activities is expected since users typically open (and close) multiple files during an I/O phase. Also, as shown in Fig. 14 (c) file open and close activities are correlated with each other across all window lengths for lag of 0. This is expected as well, since users are likely to open and close the same files during an I/O phase. As was explored in Sec. 4 (Fig. 7), over 80% of high I/O phases last 5 minutes or less. This is reflected in the high correlation between file open and close activities at 1 minute and 5 minute windows for lag of 0.

OSS CPU utilization characteristics: Next, we investigate the second question: what are the characteristics of the OSS CPU utilization? We discovered that on average, OSSes have very light CPU utilization (Fig. 15). The average CPU utilization is less than 2% for all 248 OSSes. This is expected since the the system is not performing I/O at all times. However, even the max CPU utilization is lower than 75% for more than 60% of the OSSes. This finding highlights the opportunity to opportunistically but carefully exploit idle cycles on OSSes to perform in-situ data analysis, data integration and verification. However, we caution that leveraging this finding requires careful and efficient implementation. Some analytic tools can behave erratically by exhausting the memory resources on these OSSes or delaying the critical I/O requests from compute nodes. Such cases need to be handled carefully by designing strategies that mitigate such side-effects including instability of OSS nodes. A few strategies include preemption-capable analytic tools and compute resource partitioning for isolation. First, executing preemption-capable analytic tools on OSS enables us to prioritize scheduling I/O requests and avoid starvation of I/O requests. This may also require system-level capability for efficient checkpoint-and-restart capability on OSSes. Second, explicit and deterministic resource isolation can ensure that OSSes do not become unstable due to stealing of idle cycles. Execution of analytic tools

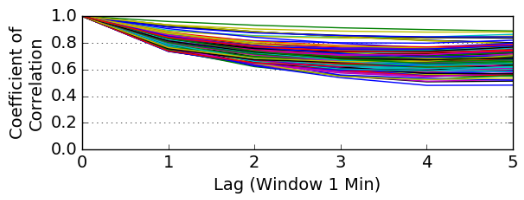


Figure 16: Correlation between OSS CPU utilization: the CPU utilization is highly autocorrelated across OSSes for 1-minute window.

can be limited in scope by assigning them to specific cores, setting the upper-limit on memory consumption, cache partitioning (e.g., Intel's Cache Allocation Technology).

Our results also reveal that OSS CPU utilization autocorrelates with itself for all OSSes for window length of 1 minute (Fig. 16). This is interesting because we found that read I/O activity and write I/O activity do not autocorrelate on many OSTs (Fig. 12(a) and (b)). At NERSC, different factors can affect OSS CPU utilization: (1) the number and type of I/O requests being served has a large impact on CPU utilization as it is proportional to the number of bookkeeping operations, (2) the number of "stat" calls on files being served has a small but statistically significant signature on OSS CPU utilization as every "stat" call triggers a read I/O, (3) the use of software RAID increases CPU utilization on writes (but not reads), etc. Therefore, in terms of I/O, OSS CPU utilization is impacted by the number and type of I/O requests and not the amount of data transferred. Therefore, the high autocorrelation of OSS CPU utilization shows that an OSS is likely to observe similar quantity of I/O requests for a few minutes on most OSSes. Autocorrelation analysis for wider window lengths (not shown for brevity) demonstrates that the correlation weakens across OSSes as the window gets wider, thus showing that OSS CPU utilization remains similar for up to 4-5 minutes (lag 4-5 for 1-minute window).

Observation: *MDS activities such as file open and close, as expected, are auto-correlated. MDS activities are bursty in nature, performing up to 327 million opens/min and 235 million closes/min during peak periods. However, the CPU utilization on OSSes are relatively modest even during high usage period (maximum CPU utilization is lower than 75% for more than 60% of the OSSes). This finding can open-up the opportunity to steal abundant idle cycles on OSSes to perform other works such as in-situ data analysis, file system verification [9, 43, 47]. However, as discussed above it requires careful and efficient implementation to avoid OSSes becoming unstable.*

8 Discussion

In this section, we discuss the scope of our findings and analysis, and identify the threats to the validity of our findings as other HPC centers learn from our experience.

Effects of I/O Workloads and NERSC-specific environment: As expected, our findings are directly affected by the nature of workloads being executing at NERSC and the NERSC environment. We emphasize that our findings cannot be generalized without appropriately factoring into NERSC environment.

Our finding that production HPC storage systems may no longer be dominated by write-intensive workloads only, clearly indicates the rise in the read-heavy workloads. We anticipate that increase in machine learning and analytic workload may attribute toward this observed trend. Machine learning workloads are often read-heavy, reading in the input data iteratively to converge to an accurate, stable and refined model. Wide increase of such workloads on leading HPC centers has been observed in recent years [1, 12, 13] and hence, this observation may become stronger over time and at other HPC centers in future too.

In particular, NERSC has been observing increasing in data and learning workloads in NERSC Exascale Science Applications Program (NESAP) [15]. Data and learning applications such as PCA and BD-CATS are quite I/O-intensive (more than 40% of time performing I/O) [14]. At the same time, interestingly, the applications that generate the large read workloads observed in this study are the same applications that do not necessarily fall into the machine learning domain and have run at NERSC for many years (QCD and quantum modeling of materials). Therefore, increase in read-intensive workloads may not be limited to data and learning workloads only.

Correlating server-side logs with application I/O patterns is a worthy goal and may provide further insights. However, it is very difficult because Lustre implements a client-side page cache. This transparently restructures very small but contiguous I/Os into large sequential writes during write-back, and this process is transparent to both application-level profiling and the back-end file system monitoring (which only sees the write-back traffic). NERSC supports over 7,000 users and 700 applications, and there are typically several hundred jobs executing concurrently at any given moment on the Cori system. In aggregate, the I/O patterns can be complex as a result of this broad workload mix, but large bursts of I/O are still observed with moderately sized jobs issuing parallel I/O.

Effects of Burst-Buffer. Burst-buffer usage at NERSC is still at infancy stage; only a handful of users take advantage of burst-buffers and rest directly use the scratch space. As discussed earlier in Sec. 3, current level of activity at the burst-buffer does not significantly influence the findings in this study. A future useful deeper exploration would be to analyze the burst-buffer traffic in a mature stage. It would require distinguishing the stage-in and stage-out data. However, current monitoring facility does not support this capability and reports only aggregate data read and written to the burst-buffer. For example, currently, one cannot distinguish whether the data is read by the compute nodes or staged-in the burst-buffer. As the burst-buffer usage matures and its monitoring capability develops, we plan to perform similar analysis of the burst-buffer I/O and how it affects I/O to the scratch space.

Effects of Lustre File System: NERSC uses Cray-maintained Lustre which is based on an old Lustre version (2.5) but has a lot of back-ported features and patches. The Lustre clients and servers are also different versions, and the clients get updated every time we patch the system. Note that the patch version of the Cray Lustre client is useful for Cray maintenance only and does not correspond to Lustre's public versions. The current server version is 2.5.1 (`jenkins-Changeling_Lustre-361-361-gbb51c1c-CHANGED-2.6.32-431.17.1.x2.0.90.x86_64`) and the client version is 2.7.5.13. We have ensured that our findings are not a side-effect of Lustre bugs.

We note that our findings cannot be generalized or trivially extended to other parallel file systems. For example, GPFS is a fundamentally different file system architecture that is block-based rather than object-based, and the definition of a stripe is fundamentally different. We also note that NERSC's default policy is no-striping because it was observed to work better with Lustre file system and useful for file-per-process I/O pattern. Typically, HPC centers set the default striping factor between 1 and 4 which may not significantly affect our observations about the low degree of I/O parallelism at NERSC, but setting the striping to higher counts can potentially decrease the load imbalance across OSTs.

For OST file distribution, NERSC's Lustre uses both round robin and weighted allocator continuously based on the weighting factor. The qos_prio_free paramter is tuned to 91% currently. The choice of the allocator is a function of this weight and the level of OST fullness of all OSTs at the moment every file is created.

9 Related Work

Tools for Monitoring I/O: Over the last two decades, there has been a large interest in trying to examine, model, and predict the I/O behavior of HPC applications. To this end, several software implementations have been proposed to monitor I/O at job-level [10, 11, 24, 44, 54] and at storage-system-level [2, 25, 26, 49, 58]. More recent attempts aim to develop an all-encompassing and cohesive monitoring system which can monitor end-to-end I/O behavior of jobs at each step along their I/O path [16, 31, 32, 40, 42, 58]. The purpose of these works is to introduce the tools and demonstrate how they can be used. Therefore, most of these works only provide a few examples as case studies and do not provide an in-depth analysis of either job-level or storage-system-level I/O behaviors.

An exception to this is a recent work by Yang et al. [58], which studies a few job-level characteristics such as write I/O's proportion of the job's total I/O, and system level characteristics such as MDS utilization by login and I/O nodes. However, these are limited to surface-level characteristics shown to demonstrate the tracing ability of their tool, Beacon. Lockwood et al.'s work [31] provides analysis based on profiling a few applications, but does not discover temporal and spatial correlations at the system-level.

I/O Characterization and Analysis: Works which provide an analysis of I/O can be broadly classified into two categories: (1) ones which study the I/O behaviors of individual applications, and (2) ones which study the I/O behavior of the storage system as a whole. The former type of works study and model I/O behaviors such as data transfer rate, I/O periodicity and repetition, and I/O variability of individual jobs [30, 33–35, 52, 55, 56, 59]. These works do not necessarily only rely on job-level monitoring tools. For instance, works by Liu et al. [28, 29] look at server-side logs to identify patterns in I/O of individual jobs, and Madireddy et al. [36] attempt to correlate the monitoring data of job-level logs with storage-system-level logs. These works are orthogonal to our work, and the findings from these works can be used in conjunction with ours to develop a holistic job-level and storage-system-level view.

On the other hand, there are also works which analyze the storage system [19, 23, 39, 49, 51]. For example, Oral et al. [39] use benchmark suites to analyze different types and configurations of file and storage systems, and provide recommendations. But it does not analyze traces of existing activity on existing storage systems. These works do not provide an in-depth analysis at MDS, OSS and OST level. They only provide the high-level characteristics at entire storage-system level, but miss on discovering interesting insights found in this work due to lack of detailed temporal and spatial analysis at different levels in space and time.

10 Conclusion

In this paper, we have drawn new insights about the temporal, spacial, and correlative behavior of HPC I/O by analyzing server-side I/O logs. We have shown that read I/O activity dominates the system's I/O utilization in terms of the amount of data transferred. In spite of this, write I/O is generally more bursty than read I/O and it displays a higher degree of parallelism. We have also shown that larger amount of reads and writes generally signal large amount of reads and writes in the near future. We also found that the OSTs are not load-balanced in terms of the amount of data transferred, and that the OSSes go largely underutilized in terms of their CPU cycles. We discussed how these findings can help improve the state of practice by enhancing the scheduling and load-balancing decisions for parallel storage systems.

Acknowledgment We are thankful to anonymous reviewers for their constructive feedback. This work is supported by NSF Award 1910601 and 1753840, and the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under contract numbers DE-AC02-05CH11231. This research used 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.

References

[1] NERSC 2017 Annual Report. https://www.nersc.gov/assets/Uploads/2017NERSC-AnnualReport.pdf, 2017.
[2] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, et al. The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications. In *SC'14*, pages 154–165. IEEE, 2014.
[3] G. Amvrosiadis, A. R. Butt, V. Tarasov, E. Zadok, and M. Zhao. Data storage research vision 2025 report. *Technical Report*, 2019.
[4] B. Austin. NERSC 2014 Workload Analysis. http://portal.nersc.gov/project/mpccc/baustin/nersc_2014_workload_analysis_v1.1.pdf, 2014.
[5] W. Bhimji, D. Bard, D. Paul, M. Romanus, et al. Accelerating science with the NERSC Burst Buffer Early User Program. In *Cray User Group (CUG)*, 2016.
[6] J. Borrill, L. Oliker, J. Shalf, and H. Shan. Investigation of leading hpc i/o performance using a scientific-application derived benchmark. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, page 10. ACM, 2007.
[7] J. Borrill, L. Oliker, J. Shalf, H. Shan, and A. Uselton. Hpc global file system performance analysis using a scientific-application derived benchmark. *Parallel Computing*, 35(6):358–373, 2009.
[8] A. Brinkmann, K. Mohror, and W. Yu. Challenges and opportunities of user-level file systemsfor hpc. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2017.
[9] J. Cao, O. R. Gatla, M. Zheng, D. Dai, V. Eswarappa, Y. Mu, and Y. Chen. Pfault: A general framework for analyzing the reliability of high-performance parallel file systems. In *Proceedings of the 2018 International Conference on Supercomputing*, pages 1–11. ACM, 2018.
[10] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross. Understanding and improving computational science storage access through continuous characterization. *ACM Transactions on Storage (TOS)*, 7(3):8, 2011.
[11] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley. 24/7 characterization of petascale i/o workloads. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–10. IEEE, 2009.
[12] S. W. Chien, S. Markidis, V. Olshevsky, Y. Bulatov, E. Laure, and J. S. Vetter. TensorFlow Doing HPC. *arXiv preprint arXiv:1903.04364*, 2019.
[13] S. W. Chien, S. Markidis, C. P. Sishtla, L. Santos, P. Herman, S. Narasimhamurthy, and E. Laure. Characterizing Deep-Learning I/O Workloads in TensorFlow. In *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*, pages 54–63. IEEE, 2018.
[14] 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*, 2017.
[15] J. Deslippe, D. Doerfler, B. Friesen, Y. H. He, T. Koskela, M. Lobet, T. Malas, L. Oliker, A. Ovsyannikov, S. Williams, et al. Analyzing Performance of Selected NESAP Applications on the Cori HPC System. In *High Performance Computing: ISC High Performance 2017 International Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, P^3MA, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers*, volume 10524, page 334. Springer, 2017.

[16] S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello. Logaider: A tool for mining potential correlations of hpc log events. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 442–451. IEEE, 2017.

[17] M. Dorier, G. Antoniu, R. Ross, D. Kimpe, and S. Ibrahim. Calciom: Mitigating i/o interference in hpc systems through cross-application coordination. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 155–164. IEEE, 2014.

[18] A. Gainaru, G. Aupy, A. Benoit, F. Cappello, Y. Robert, and M. Snir. Scheduling the i/o of hpc applications under congestion. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 1013–1022. IEEE, 2015.

[19] 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*, pages 31–36. ACM, 2015.

[20] H. S. Gunawi, R. O. Suminto, R. Sears, C. Golliher, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey, et al. Fail-slow at scale: Evidence of hardware performance faults in large production systems. *ACM Transactions on Storage (TOS)*, 14(3):23, 2018.

[21] S. Herbein, D. H. Ahn, D. Lipari, T. R. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer. Scalable i/o-aware job scheduling for burst buffer enabled hpc clusters. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 69–80. ACM, 2016.

[22] D. Huang, Q. Liu, J. Choi, N. Podhorszki, S. Klasky, J. Logan, G. Ostrouchov, X. He, and M. Wolf. Can i/o variability be reduced on qos-less hpc storage systems? *IEEE Transactions on Computers*, 68(5):631–645, 2018.

[23] Y. Kim and R. Gunasekaran. Understanding i/o workload characteristics of a peta-scale storage system. *The Journal of Supercomputing*, 71(3):761–780, 2015.

[24] M. Koo, W. Yoo, and A. Sim. I/o performance analysis framework on measurement data from scientific clusters. 2015.

[25] J. M. Kunkel, E. Betke, M. Bryson, P. Carns, R. Francis, W. Frings, R. Laifer, and S. Mendez. Tools for analyzing parallel i/o. In *International Conference on High Performance Computing*, pages 49–70. Springer, 2018.

[26] J. M. Kunkel, M. Zimmer, N. Hübbe, A. Aguilera, H. Mickler, X. Wang, A. Chut, T. Bönisch, J. Lüttgau, R. Michel, et al. The siox architecture–coupling automatic monitoring and optimization of parallel i/o. In *International Supercomputing Conference*, pages 245–260. Springer, 2014.

[27] S. Liang, Z. Qiao, J. Hochstetler, S. Huang, S. Fu, W. Shi, D. Tiwari, H.-B. Chen, B. Settlemyer, and D. Montoya. Reliability characterization of solid state drives in a scalable production datacenter. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3341–3349. IEEE, 2018.

[28] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai. Automatic Identification of Application I/O Signatures from Noisy Server-Side Traces. In *FAST*, volume 14, pages 213–228, 2014.

[29] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai. Server-side log data analytics for i/o workload characterization and coordination on large shared storage systems. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pages 819–829. IEEE, 2016.

[30] G. K. Lockwood, S. Snyder, T. Wang, S. Byna, P. Carns, and N. J. Wright. A year in the life of a parallel file system. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, page 74. IEEE Press, 2018.

[31] 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. 2018.

[32] G. K. Lockwood, W. Yoo, S. Byna, N. J. Wright, S. Snyder, K. Harms, Z. Nault, and P. Carns. Umami: A recipe for generating meaningful metrics through holistic i/o performance analysis. In *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems*, pages 55–60. ACM, 2017.

[33] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[34] J. Lüttgau, S. Snyder, P. Carns, J. M. Wozniak, J. Kunkel, and T. Ludwig. Toward understanding i/o behavior in hpc workflows. In *Proc. of Workshop in conjunction with ACM/IEEE Supercomputing Conference, Dallas, TX, USA*, 2018.

[35] H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, M. Prabhat, S. Byna, and Y. Yao. A multiplatform study of i/o behavior on petascale supercomputers. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, pages 33–44. ACM, 2015.

[36] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. M. Wild. Analysis and Correlation of Application I/O Performance and System-Wide I/O Activity. In *Networking, Architecture, and Storage (NAS), 2017 International Conference on*, pages 1–10. IEEE, 2017.

[37] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, and R. Ricci. Taming performance variability. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 409–425, 2018.

[38] S. Oral, D. A. Dillow, D. Fuller, J. Hill, D. Leverman, S. S. Vazhkudai, F. Wang, Y. Kim, J. Rogers, J. Simmons, et al. Olcfs 1 tb/s, next-generation lustre file system. In *Proceedings of Cray User Group Conference (CUG 2013)*, pages 1–12, 2013.

[39] S. Oral et al. Best practices and lessons learned from deploying and operating large-scale data-centric parallel file systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 217–228. IEEE, 2014.

[40] B. H. Park, S. Hukerikar, R. Adamson, and C. Engelmann. Big data meets hpc log analytics: Scalable approach to understanding systems at extreme scale. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 758–765. IEEE, 2017.

[41] R. Ross, L. Ward, P. Carns, G. Grider, S. Klasky, Q. Koziol, G. K. Lockwood, K. Mohror, B. Settlemyer, and M. Wolf. Storage systems and i/o: Organizing, storing, and accessing data for scientific discovery. Technical report, USDOE Office of Science (SC)(United States), 2019.

[42] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. 2010.

[43] H. Sim, Y. Kim, S. S. Vazhkudai, D. Tiwari, A. Anwar, A. R. Butt, and L. Ramakrishnan. Analyzethis: an analysis workflow-aware storage system. In *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2015.

[44] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright. Modular hpc i/o characterization with darshan. In *2016 5th Workshop on Extreme-Scale Programming Tools (ESPT)*, pages 9–17. IEEE, 2016.

[45] H. Song, Y. Yin, X.-H. Sun, R. Thakur, and S. Lang. Server-side i/o coordination for parallel file systems. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 17. ACM, 2011.

[46] K. Tang, P. Huang, X. He, T. Lu, S. S. Vazhkudai, and D. Tiwari. Toward managing hpc burst buffers effectively: Draining strategy to regulate bursty i/o behavior. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 87–98. IEEE, 2017.

[47] D. Tiwari, S. Boboila, S. Vazhkudai, Y. Kim, X. Ma, P. Desnoyers, and Y. Solihin. Active flash: Towards energy-efficient, in-situ data analytics on extreme-scale machines. In *Presented as part of the 11th {USENIX} Conference on File and Storage Technologies ({FAST} 13)*, pages 119–132, 2013.

[48] A. Uselton and N. Wright. A file system utilization metric for i/o characterization. In *Proc. of the Cray User Group conference*, 2013.

[49] 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. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 45. ACM, 2017.

[50] L. Wan, F. Wang, S. Oral, D. Tiwari, S. S. Vazhkudai, and Q. Cao. A practical approach to reconciling availability, performance, and capacity in provisioning extreme-scale storage systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 75. ACM, 2015.

[51] F. Wang, S. Oral, S. Gupta, D. Tiwari, and S. S. Vazhkudai. Improving large-scale storage system performance via topology-aware and balanced data placement. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 656–663. IEEE, 2014.

[52] T. Wang, S. Snyder, G. Lockwood, P. Carns, N. Wright, and S. Byna. Iominer: Large-scale analytics framework for gaining knowledge from i/o logs. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 466–476. IEEE, 2018.

[53] C. H. Wartens and J. Garlick. Lmt-the lustre monitoring tool, 2010.

[54] S. A. Wright, S. D. Hammond, S. J. Pennycook, R. F. Bird, J. Herdman, I. Miller, A. Vadgama, A. Bhalerao, and S. A. Jarvis. Parallel file system analysis through application i/o tracing. *The Computer Journal*, 56(2):141–155, 2012.

[55] B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki. Characterizing output bottlenecks in a supercomputer. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

[56] B. Xie, Y. Huang, J. S. Chase, J. Y. Choi, S. Klasky, J. Lofstead, and S. Oral. Predicting output performance of a petascale supercomputer. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, pages 181–192. ACM, 2017.

[57] E. Xu, M. Zheng, F. Qin, Y. Xu, and J. Wu. Lessons and actions: What we learned from 10k ssd-related storage system failures. In *2019 USENIX Annual Technical Conference*.

[58] B. Yang, X. Ji, X. Ma, X. Wang, T. Zhang, X. Zhu, N. El-Sayed, H. Lan, Y. Yang, J. Zhai, et al. End-to-end i/o monitoring on a leading supercomputer. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 379–394, 2019.

[59] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu. On the root causes of cross-application i/o interference in hpc storage systems. In *2016 IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, pages 750–759, 2016.