A Year in the Life of a Parallel File System

Glenn K. Lockwood*, Shane Snyder†, Teng Wang*, Suren Byna*, Philip Carns†, Nicholas J. Wright*
*Lawrence Berkeley National Laboratory, Berkeley, CA, USA
{glock, tengwang, sbyna, njwright}@lbl.gov
†Argonne National Laboratory, Lemont, IL, USA
{ssnyder, carns}@mcs.anl.gov

***Abstract*—I/O performance is a critical aspect of data-intensive scientific computing. We seek to advance the state of the practice in understanding and diagnosing I/O performance issues through investigation of a comprehensive I/O performance data set that captures a full year of production storage activity at two leadership-scale computing facilities. We demonstrate techniques to identify regions of interest, perform focused investigations of both long-term trends and transient anomalies, and uncover the contributing factors that lead to performance fluctuation.**

We find that a year in the life of a parallel file system is comprised of distinct regions of long-term performance variation in addition to short-term performance transients. We demonstrate how systematic identification of these performance regions, combined with comprehensive analysis, allows us to isolate the factors contributing to different performance maladies at different time scales. From this, we present specific lessons learned and important considerations for HPC storage practitioners.

I. Introduction

I/O performance variation has been studied extensively, and various conditions have been identified as factors contributing to poor I/O performance. Most studies have focused on enumerating the sources of I/O performance loss at a single point in time, assuming that performance loss is a transient effect due to contention from other jobs. However, recent work [1] has shown that performance variation can occur over periods of days as a result of systematic, longer-term conditions of a storage system. Overall performance (and thus scientific productivity) can be improved for a wide range of users if these deviations can be identified and attributed quickly in production. Additionally, the determination of the reasons for the performance slowdowns can be fed into the design of future file systems.

A number of recent efforts [1], [2], [3], [4] have advanced the state of the art in scalable data collection, making it possible to observe production systems at unprecedented scales. It remains an open problem, however, how to best *interpret* this data quickly for maximum production impact, and which telemetry sources have the greatest return on investment are not always clear. Aligning these broad and diverse data sets and contextualizing and interpreting their contents require both a deep understanding of I/O performance and a broad understanding of statistical analysis techniques.

We investigate these issues by studying performance data collected from large parallel file systems at two leadership-class high-performance computing (HPC) centers over the course of a year of production use. In addition to passive instrumentation, such as system monitoring and application profiling, we use active probing of I/O performance to record user-perceived performance over time. The breadth of the data set enables investigation over multiple time scales, ranging from days to months, to identify trends in both absolute performance and variability. The depth of the data set enables correlation analysis to identify subtle relationships between performance and a variety of systemwide metrics.

The primary contributions of this work are as follows.

- We have collected, and will make publicly available, an unprecedented year-long, multifacility I/O performance data set. It includes the results of daily I/O performance probes and their associated telemetry data.
- We show that baseline performance and variability change over time. Factors such as system software updates and sustained I/O-intensive workloads contribute to long-term variations.
- We show that the nature of correlations between I/O performance and system metrics also change over time. For example, we demonstrate that high CPU load can correlate with favorable performance under healthy file system conditions, and it can coincide with unfavorable performance when non-I/O workloads are impacting storage servers.
- We show that contention for bandwidth, input/output operations per second (IOPS), and metadata resources can be confidently determined to be the sources of transient I/O performance problems.
- We develop methods for identifying underlying trends within noisy telemetry data, and we show how analysis focused on these trends improves the quality of the analysis results.

This paper is organized as follows. Section II contains related work, Section III describes the data collection framework, the benchmarks used, and the HPC platforms studied, Section IV describes our methodology for determining regions of similar I/O performance, Section V contains the results of our statistical analysis of the data, Section VI discusses the broader implications of our work for the state of the practice, and Section VII presents our conclusions.

II. Related Work

Several recent studies have contributed techniques to combine systemwide HPC instrumentation data for integrated analysis [1], [2], [3], [4], [5]. Vazhkudai et al. notably used a Splunk data warehouse to analyze system logs and perform operational analytics on a large-scale storage system [2]. The

SC18, November 11-16, 2018, Dallas, Texas, USA
978-1-5386-8384-2/18/$31.00

cloud computing community has also identified the need to unify analysis capabilities across diverse environments. The Dapper system developed by Sigelman et al. combines comprehensive tracing with a novel sampling method to observe critical paths in large cloud environments in great detail [6].

Luu et al. [7] studied application I/O logs from multiple platforms to derive conclusions on I/O system utilization and library usage. Di et al. [8] and Park et al. [9] have investigated performing correlation analysis on HPC log data once it has been collected. Their analyses included compute and network resources, with a particular emphasis on event logs. Inacio et al. analyzed performance variability using statistical analysis of file system read/write operations and concluded that both the experimental environment and Lustre stripe settings impact performance applications. Others have documented I/O performance variability anecdotes on leadership-scale systems [10], [11], [12] and proposed methods to combat it.

However, these studies of application I/O and parallel file system performance and variability are based either on a small set of applications or on observations over a short duration. Furthermore, examining how performance and variation may change over time remains relatively unexplored, with the existing body of work being largely anecdotal [13]. In this work, we build upon best-in-class previous efforts by combining system monitoring, application monitoring, and active performance probing to quantify holistically how I/O performance variation manifests across many dimensions over a year-long period. To this end, we also introduce systematic methods to help automate the task of deriving actionable insight from these data sources over multiple time scales. We have analyzed these holistic data sets for an entire year on multiple parallel file systems and present a broad statistical analysis that provides an unprecedented advancement in our understanding of HPC I/O performance variability.

III. Methodology

Previous studies have demonstrated the feasibility of holistic HPC I/O analysis and used it to observe anomalous I/O behavior and contributing factors over short time scales [1]. This methodology was later formalized in the specification of the TOKIO (Total Knowledge of I/O) framework [14]. In this work, we use the TOKIO framework to collect a complete year of I/O metrics on multiple production platforms. We analyze the collected data to observe transient and long-term trends in I/O performance variability. This section summarizes the TOKIO framework, describes the production HPC platforms that we have used in this study, and introduces our methodology for active probing of storage system performance.

A. Data collection framework

TOKIO is a framework facilitating holistic characterization and analysis of I/O workloads running on today's production HPC systems. Conceptually, it provides an abstraction layer between component-level monitoring tools already deployed on HPC platforms and higher-level I/O analysis tools that utilize this data, as illustrated in Figure 1. The fundamental

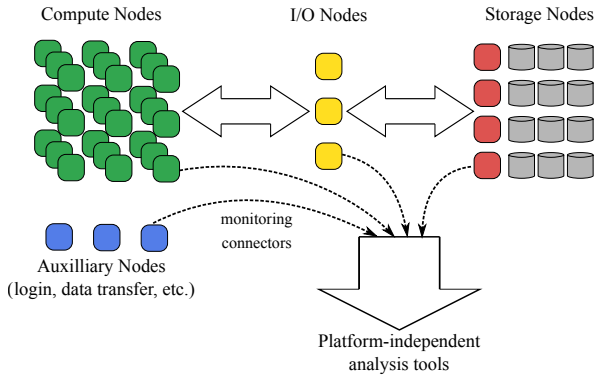

Fig. 1. Overview of the TOKIO framework, providing holistic analysis across the numerous components of the I/O subsystem on HPC platforms.

roles of the TOKIO framework are to collate monitoring data from distinct components, integrate and normalize the data from these components, and present coherent interfaces for indexing and accessing this data.

TOKIO uses a modularized software architecture and a generic data format specification for system monitoring data, simplifying portability to new HPC platforms. Software modularity is accomplished by defining abstract *connectors* to monitoring sources; these connectors then expose interfaces for extracting relevant data in a format suitable for TOKIO. A generic time series format is used for semantically consistent access to data originating from distinct monitoring tools, even though each tool uses its own underlying data format with its own scope and granularity. The time series data format designates a number of metrics that are common to classes of monitoring components. For example, file system monitoring tools often gather common metrics such as read/write bandwidths and operation counts. TOKIO connectors can convert the native data formats of their underlying tools to this generic format in-memory as part of on-demand analysis, or TOKIO-aware tools can archive directly into this format on-disk. These design decisions greatly simplify the process of integrating new monitoring sources as well the process of developing platform-independent I/O analysis tools. This study in particular relies on integrated analysis of the following monitoring connectors.

- **Application-level monitoring**: *Darshan* [15] is an application I/O characterization tool that is commonly deployed at production HPC facilities. It provides a condensed set of I/O counters, timers, and other statistics for each file accessed by a given application. Previous analysis [16] has demonstrated Darshan's negligible runtime overheads and log file sizes (ranging from tens of kilobytes to a few megabytes per job, depending on the workload).
- **File system workload monitoring**: *LMT* [17] and *ggiostat* [1] are examples of file system monitoring tools for Lustre and GPFS deployments, respectively. These tools each capture metrics quantifying file system workloads periodically over a time interval, with some of these metrics being common to both tools (e.g., observed read/write bandwidths, number of specific metadata operations issued)

and others being file system specific (e.g., CPU utilization on a given Lustre metadata server captured by LMT). File system workload data is collected asynchronously outside of the application data access path to minimize overhead.
- **File system capacity/health monitoring**: File-system-specific tools such as Lustre's *lfs* and *lctl* or GPFS's *mmdf* and *mmlsdisk* can be invoked periodically to capture current file system state, including the capacity and failover status of individual storage servers and/or LUNs in the system.
- **Resource manager monitoring**: Resource managers such as Slurm [18] often keep a detailed accounting of all jobs that execute on a particular system, including useful details like the size of the job and its placement across available compute nodes.

The telemetric data from these connectors is aggregated into scalar *attributes* associated with each job, and all the attributes associated with a job are represented as *feature vectors*. For monitoring sources that produce time-resolved data (e.g., LMT reports workload statistics at five-second intervals), we calculate the sum, minimum, maximum, and average value of the data generated over the duration of the job and define those four reduced scalars as attributes of the feature vector. Wherever possible, attributes are also expressed as *coverage factors* [1], which quantify the fraction of systemwide activity that can be attributed to the job of interest. For example, a job's *bandwidth coverage factor* is the number of bytes read and written by that job's application (measured by Darshan) divided by the number of bytes read and written across the entire parallel file system (measured by LMT or ggiostat) while that job was running. A coverage factor of 1.0 indicates that a job was the exclusive consumer of a resource, while a coverage factor of 0.4 indicates that other competing jobs consumed 60% of the total delivered resources.

B. Platforms

We applied the TOKIO framework on the Edison and Cori systems at NERSC and the Mira system at the ALCF. Each of these platforms, along with their corresponding file systems analyzed as part of this study, is briefly described in Table I.

Darshan is installed and automatically enabled on each of these systems, transparently characterizing the I/O workloads of a large portion of each system's job population. NERSC has deployed LMT for full-time monitoring of the Lustre scratch volumes on both Edison and Cori, while the ALCF has deployed ggiostat to do the same for the GPFS volumes on Mira. NERSC systems additionally utilize the Slurm resource manager, which allows for the capture of detailed metadata for every job executed on Edison or Cori. Similar job metadata is available from ALCF's Ni API. TOKIO can use these data sources in-place to provide a unified holistic view of I/O performance without copying data to a dedicated database.

TABLE I
DESCRIPTION OF NERSC AND ALCF TEST PLATFORMS.

	Platform	FS Name (Type) Servers (LUNs)	Size	Peak Rate
Edison (NERSC)	Cray XC30 5,586 CNs	scratch1 (Lustre) 24 (24)	2.2 PiB	48 GB/sec
		scratch2 (Lustre) 24 (24)	2.2 PiB	48 GB/sec
		scratch3 (Lustre) 36 (36)	3.3 PiB	72 GB/sec
Cori (NERSC)	Cray XC40 12,076 CNs	cscratch (Lustre) 248 (248)	28 PiB	744 GB/sec
Mira (ALCF)	IBM BG/Q 49,152 CNs	mira-fs1 (GPFS) 48 (336)	7.0 PiB	90 GB/sec

C. I/O Performance Probes

To measure the performance variation on the systems described in Section III-B, we ran four I/O-intensive application benchmarks on a daily basis from February 14, 2017, to February 15, 2018. We utilize these benchmarking runs as another monitoring source for this study, actively probing the user-perceived I/O performance of each analyzed file system on a daily basis across a range of representative I/O motifs. The four underlying benchmarks (VPIC [19], BDCATS [20], [19], HACC [21], and IOR [22]) were chosen to exercise a variety of I/O workloads and covered the four I/O motifs (listed in Table II) in both read and write modes. Each probe was configured to run identically to previous work [1] with the goal of using a substantial fraction of the I/O subsystem's peak bandwidth while using minimal production cycles.

All probes ran using 256 nodes (4,096 processes) on Cori, 128 nodes (2,048 processes) on Edison, and 1,024 nodes (16,384 processes) on Mira. The shared-file probes on Lustre were configured to use maximum striping per file, while the file-per-process probes placed each file on a single off-line storage table. The objective in both cases was to (in aggregate) leverage all available storage devices. The default automatic striping policy was used on GPFS. In the case of Cori and Edison, each job had access to the full I/O bandwidth of its I/O nodes as well, but because of the way in which I/O nodes are allocated in a fixed ratio to job size on Blue Gene/Q systems [23], Mira jobs were restricted to the bandwidth provided by eight I/O nodes.

Of the intended probes, 81.9% successfully generated results, providing 11,986 performance observations across Mira, Cori, and Edison over the course of a year. The remaining 18.1% of potential probes failed because of system downtime, malfunctions of component-level monitoring tools or the automated test scheduling, queue wait times that exceeded 24 hours, or excessive walltime usage. These failed probe samples (which are not considered in this study) point out the need for additional analysis in future work that incorporates failure data as well as performance data.

TABLE II
I/O PERFORMANCE PROBE MOTIFS

	$O(10^1$ MiB) **Transfers**	$O(10^2$ MiB) **Transfers**
Shared File	IOR/shared	VPIC and BDCATS
File Per Process	IOR/fpp	HACC

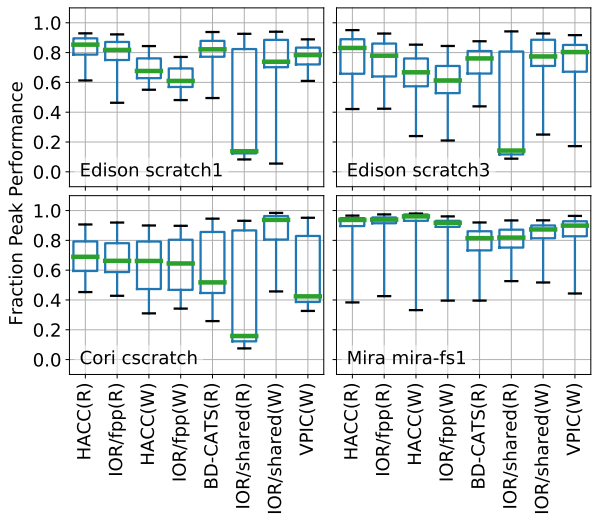

Fig. 2. I/O performance grouped by I/O motifs and read(R)/write(W) mode on different file systems. Whiskers represent the 5th and 95th percentiles. Edison scratch2 distributions are not shown but closely resemble scratch1.

IV. Identifying Trends

Continuous passive monitoring and active performance probing produce a large volume of data ($\approx$ 15 GiB) over the course of a year. We began our investigation by summarizing the dataset at a high level, establishing methods for organizing and navigating the data in a semantically meaningful way, and establishing methods for identifying regions of interest.

A. Dataset overview

Absolute I/O performance is influenced by many factors, most notably (a) application I/O pattern, (b) read/write ratio, and (c) I/O system architecture [1], [24]. These make it difficult to contextualize performance variation across benchmarks or platforms because each combination is capable of a different baseline performance level. We address this problem by normalizing the performance of each of the 11,986 observations in terms of its *fraction of peak performance*. The fraction of peak performance is the absolute performance (in bytes/sec) of an observation divided by the maximum absolute performance observed across all jobs with a common (a), (b), and (c) above. This approach allows us to focus on variability for different classes of I/O workloads rather than their relative performance.

The distribution of the fraction of peak performance measurements for four of the five systems tested is shown in Fig. 2. The figure illustrates that the performance of active I/O performance probes on production file systems is highly dynamic over the course of a year. It does not provide any insight into the temporal nature of the performance fluctuations, however.

Figure 3 visualizes the same data in the form of a heatmap over time. This shows that performance variation is not randomly distributed over the year; this key characteristic is not captured by time-independent performance distributions. Several archetypical forms of correlated performance degradation observed in Fig. 3 are highlighted in Fig. 4 and fall into three broad categories of variation.

1) Dark vertical bands, exemplified in the Mira data in Fig. 4a, represent transient systemwide issues that result in a uniform loss of performance for probes executed that day.
2) Dark horizontal bands, shown in the Cori data in Fig. 4b, indicate a long-term degradation in performance that disproportionately affects a specific I/O motif.
3) Isolated dark blocks represent individual probes where performance was poor for a very short period of time.

The preponderance of these time-dependent phenomena underscores the observation that baseline I/O performance and variability are not constant over time and that what may qualify as abnormally poor performance during one period of time may be the baseline performance expectation during another.

This observation has implications for both HPC facility operators and users. For facility operators, it indicates that performance anomalies and their root causes should not be assessed in isolation. By integrating broader spatial and temporal context into the analysis, facility operators can more accurately discriminate between application problems and environmental factors. For users and application developers, it follows that the accuracy of parameterized I/O performance models [24], [25] will degrade unless they are reparameterized as the I/O subsystem that they model evolves. Both these cases justify the need for a systematic approach for identifying different regions of I/O performance in order to differentiate long-term factors and phenomena from short-term transients.

B. Time-dependent analysis

Section IV-A clearly illustrates the presence of time-varying behavior, but quantitative methods are needed to extract actionable insight from these observations. The main challenge that these methods face is differentiating performance trends from individual short-term fluctuations in time series data. This problem is not unique to I/O performance or even computer science, however. Notably, financial market technical analysis techniques are routinely used for a similar purpose: attenuating day-to-day volatility in the price of assets and identifying price movement trends [26], [27].

The most straightforward initial technique to apply from this domain is the use of simple moving averages (SMAs) over the performance observations. Given a time window of width w, the SMA for performance at time t is the arithmetic mean of the fraction peak performance over $-0.5w <= t < +0.5w$. When chosen to be sufficiently short ($w_{short} \sim O(\text{days})$), the resulting SMA_{short} provides a rapid visual means to identify performance degradation or recovery that lasts for $O(\text{days})$. Multiple SMAs can be plotted simultaneously over the same dataset to differentiate short- and long-term trends and identify key crossover points. We further note that SMAs can be calculated across individual workloads or across a set of workloads by simply calculating the arithmetic mean of all relevant performance observations falling within w. The former is useful for identifying motif-specific or transient performance issues, while the latter is helpful for detecting systemic performance issues.

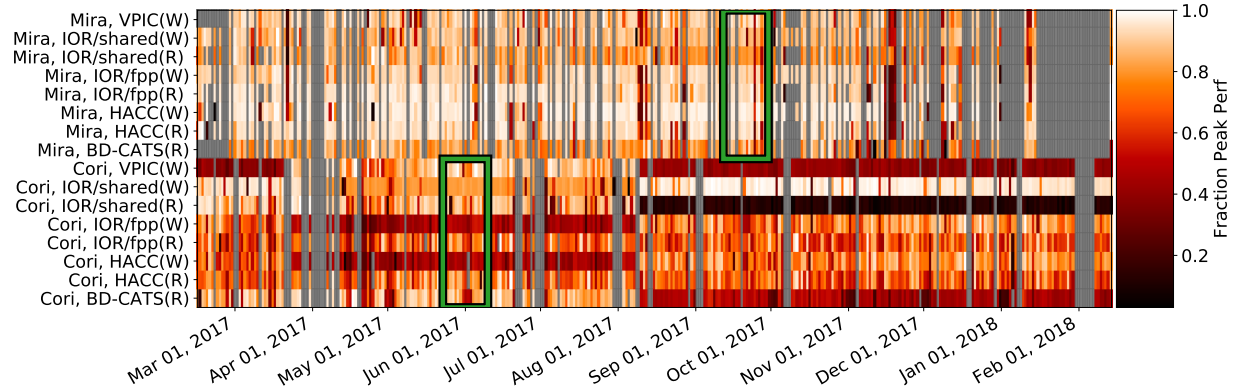

Fig. 3. Performance of daily probes normalized to the peak observed performance for each probe type (I/O motif and read/write mode combination) on the specified system. The y-axis labels show combinations of system, I/O motif, and mode (Read/Write). Grey represents days on which no observations were made. The two regions highlighted in green boxes are expanded upon in Figure 4.

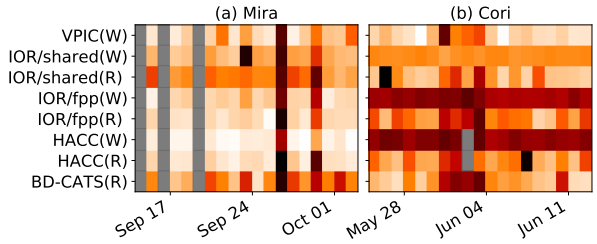

Fig. 4. Examples of structure in the fraction of peak performance observations. Color scale is the same as that in Fig. 3. In (a), the vertical band on Sept. 26 corresponds to a transient systemwide degradation on Mira. The horizontal bands for the IOR file-per-process write workload (IOR/fpp(W)) and HACC write workload (HACC(W)) in (b) show a motif-specific sustained performance problem for file-per-process write workloads on Cori.

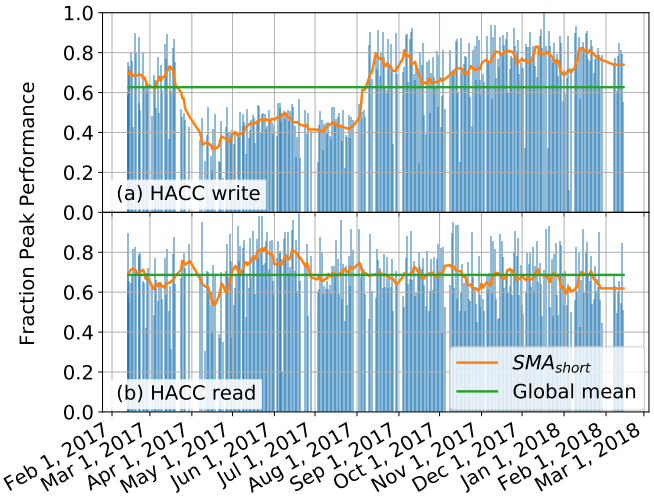

Fig. 5. Performance evolution of HACC file-per-process workload on Cori. Green line is the global mean (298 GiB/sec write, 204 GiB/sec read), and blue bars are raw performance measurements.

An example of an SMA (w_{short} = two weeks) applied to the performance data collected from the HACC workloads run on Cori is shown in Fig 5. When contrasted with a time-independent summary statistic such as the overall mean performance of the entire year, the SMA clearly identifies the long period of degraded HACC write performance on Cori that was qualitatively shown in the bottom half of Fig. 3. The points at which the SMA rises above or below the global mean performance also provide quantitative measurements of the region of time when an underlying issue manifested itself; in Fig. 5, these *crossover points* fall on March 24 and August 10. Cross-referencing these dates with the service history of Cori retrospectively revealed that the beginning and end of this long-term region of divergent performance coincided with major system software upgrades that also happened on March 24 and August 10.

Curiously, the performance of HACC read workload (Fig. 5b) was unaffected during this time, demonstrating that not all workloads are affected by long-term variation equally. This asymmetry, in combination with the bounding dates of this divergent region, allowed us to trace this specific issue to unintentional behavior introduced (and later fixed) in the Lustre software running on Cori between the system upgrades. Although this particular case of long-term performance divergence was caused by an unexpected bug in system software, the reality of most production storage systems is that they are regularly patched and upgraded. At a minimum, the security requirements of the centers that run these systems drive system updates; and as exemplified by the recent Spectre and Meltdown patches, such updates can have nontrivial effects on certain types of I/O [28]. Thus, administrative activities such as maintenance patches and software updates are a significant source of time-dependent, long-term performance variation. This must be accounted for in both retrospective performance analysis and forward-looking performance modeling parameterization. The ramifications for I/O research practitioners are that holistic I/O monitoring must incorporate environmental provenance information, such as kernel, operating system, and file system version, to aid in correlation.

C. Regions of interest

We can generalize the analysis technique from Section IV-B by superimposing a second SMA (SMA$_{long}$) with a longer window ($w_{long} \sim O(\text{weeks})$) on top of SMA$_{short}$ (which captures variations $O(\text{days})$). Doing so allows us to examine short-term performance variations (e.g., a period of sustained bandwidth contention) in the context of longer-term trends (e.g., in the presence of a file system software regression). Once a region of interest has been defined, we can then

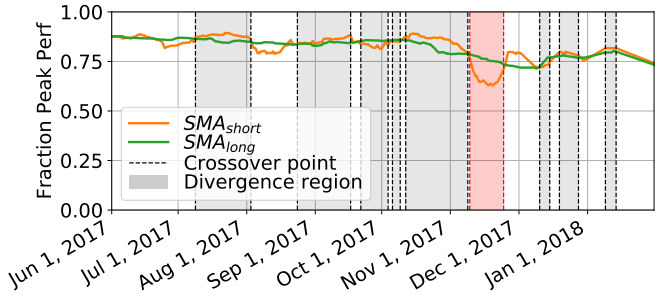

Fig. 6. Schematic depicting the relationship between SMAs, crossover points, and divergence regions on Mira. Both white and gray regions between crossovers are divergence regions, but only a subset of divergence regions are shaded for clarity. The significance of the divergence region highlighted in red is discussed in Section V-A.

constrain more detailed analysis techniques to that region to determine why it was defined. The crossover points at which SMA_{short} and SMA_{long} intersect establish the boundaries of regions where short-term performance has diverged from longer-term performance and anomalous performance is prevailing. Thus, we introduce the notion of *divergence regions*, which are the periods of time bounded by two crossover points and which capture correlated performance. Figure 6 illustrates how these concepts are applied to partition data.

For the remainder of this study, we apply the concept of *divergence regions*, bounded by the crossovers between SMA_{short} and SMA_{long}, to systematically identify and characterize periods in time where anomalous performance was observed by the active I/O performance probes running across the test systems. We define SMA_{short} to have $w_{short} = 14$ days as in Section IV-B and SMA_{long} to have $w_{long} = 49$ days. We chose w to be a multiple of seven days to align with one week and ensure that weekends and weekdays were equally represented in both SMA calculations. w_{long} was set to seven weeks to span multiple w_{short} regions and at least one month boundary. In our experience, the analysis presented in this work was insensitive to changes of ± 1 week.

In general, we find that financial market technical analysis techniques can be adapted to time series I/O performance data to attenuate noise and identify underlying trends. In the context of financial markets, SMAs (and other more sophisticated techniques) are used for predictive purposes to time the execution of market trades. In this study, we are not applying them to the task of predicting performance, but rather to identify regions of interest in recorded observations.

V. Investigating Trends

SMA crossover-based partitioning of performance observations provides a systematic method for grouping time-correlated performance events. Once a divergence region (i.e., a performance trend) has been identified, more focused statistical analyses can then be applied within the region to gain insight into the factors that contributed to that trend. Examples of contributing factors may include resource utilization, system health, and component performance. In the following analyses, we separate our 11,986 performance observations into sets

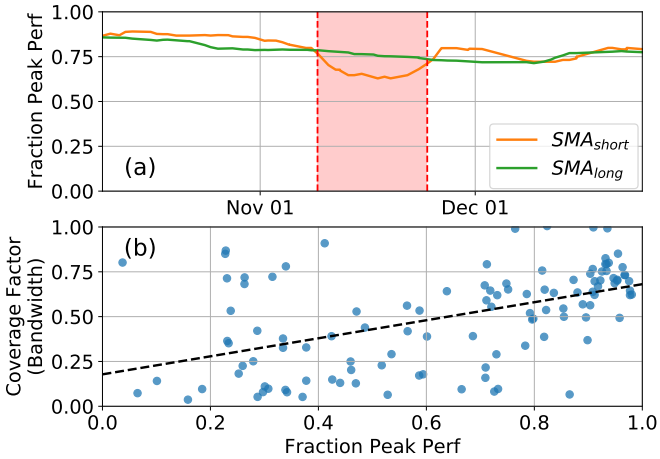

Fig. 7. Correlation between performance and bandwidth coverage factor in a divergence region on Mira for all I/O motifs combined. (a) Divergence region of interest, corresponding to the same highlighted region shown in Fig. 6. (b) Correlation between performance and bandwidth coverage factor in that region. Correlation coefficient is 0.507, and p-value is 8.71×10^{-9}; dashed line in (b) is a linear fit with slope 0.503 drawn for visual aid.

of observations that all ran on the same test platform (as described in Table I) to characterize the factors that contribute to time-dependent, systemic performance variation across different file systems and architectures.

A. Correlative analysis

We begin our correlative analysis by partitioning a year-long dataset into divergence regions using the method described in Section IV-C. Using the performance observations across all Mira mira-fs1 workloads as an example, we set w_{short} = two weeks and w_{long} = seven weeks to identify divergence regions and then discard any regions with fewer than three data points. Regions with few data points are discarded for two reasons: (a) intuitively, very short divergence regions occur when $\text{SMA}_{short} \approx \text{SMA}_{long}$ and there is minimal long-term variation, and (b) statistically, it is impossible to assert the statistical significance of a correlation with fewer than three data points. This yields 32 divergence regions.

We then apply Pearson correlation [29] to the feature vectors within each divergence region to identify the factors that correlate with its performance trend. The result of this process is a new feature vector for each divergence region that contains the correlation coefficients between the fraction of peak performance and every other attribute across all observations in that region. We then use the p-value of each correlation coefficient[1] to further down-select the total set of regions to those with extremely high significance (p-value $< 1.0 \times 10^{-5}$). This threshold yields a total of nine relevant divergence regions on Mira mira-fs1, which are all depicted as shaded extents in Fig. 6. Each of these nine regions exhibits at least moderate cor-

[1]The p-value of a correlation coefficient is the probability of observing data that would show the same correlation coefficient in the absence of any real relationship between the underlying metrics. A low p-value indicates that it is extremely unlikely that the calculated correlation coefficient would be observed if the metrics being compared had no real correlation. As such, p-values represent the statistical significance of a statistical measurement.

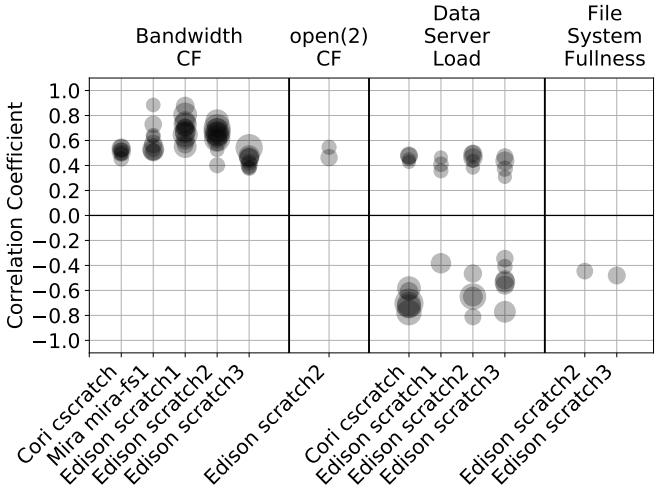

Fig. 8. Correlations discovered between fraction peak performance and all other attributes measured during job execution. Each circle represents the correlation coefficient over a single trend region, and its diameter is proportional to $-\log_{10}$(p-value). CF denotes coverage factor.

relation (R) ranging from 0.507 to 0.884 with the bandwidth coverage factor feature.

Figure 7 illustrates the correlation between bandwidth coverage factor and fraction peak performance for a particular divergence region in November 2017 on Mira mira-fs1. This example shows the lowest correlation ($R = 0.507$) with bandwidth coverage factor of any of the selected regions, and the scatter plot of performance results shows why: This region contains a cluster of poorly performing probes ($0.2 <$ fraction peak perf < 0.4) that ran with a relatively high bandwidth coverage factor. This region is also the single largest divergence region observed on Mira; a difference of over 20% between SMA_{short} and SMA_{long} was observed during this time. This divergence region example highlights the importance not only of identifying regions and calculating correlations but also of identifying cases in which the analysis indicates the presence of an unknown factor that is not adequately captured by the instrumentation framework.

Despite the unusual region shown in Fig. 7, however, the data indicates that the time-dependent performance divergences observed on Mira show either moderate or strong correlation with bandwidth contention. Additionally, this correlation with performance degradation occurs across *all* I/O motifs (similar to Fig. 4a), which distinguishes it from the motif-specific case discussed in Section IV-B.

When the same Pearson correlation is calculated across the entire collection of Mira mira-fs1 data in the absence of partitioning, the correlation with the bandwidth coverage factor yields an overall result of $R = 0.483$ and p-value $= 2.25 \times 10^{-88}$. Thus, significantly stronger correlations can be found by focusing analysis on algorithmically identified regions of interest in the data.

B. Survey of divergence regions

Next we apply the same correlation analysis to the other test platforms in our study, again keeping only those correlations

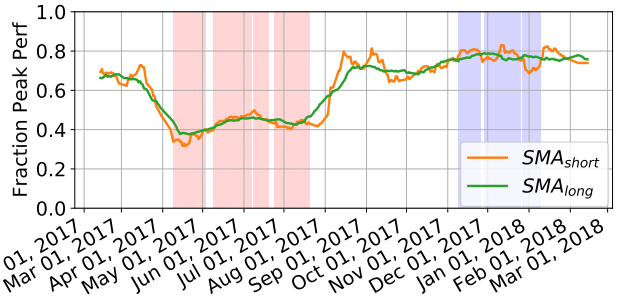

Fig. 9. Regions of negative correlation (red) and positive correlation (blue) between fraction peak performance and data server load during divergence regions identified on Cori. SMAs for Cori's HACC write workload are also shown to illustrate the coincidence of a long-term performance issue with the direction of correlation.

with an extremely high significance (p-value $< 1.0 \times 10^{-5}$). The results of this analysis are shown in Fig. 8. As was found with Mira in Section V-A, a moderate to strong correlation exists between I/O performance and the bandwidth coverage factor on the Lustre file systems of Cori and Edison. Although bandwidth contention resulting in performance loss is intuitive at the scale of a single performance transient, the fact that these correlations were found over longer-term divergence regions indicates that bandwidth contention from sustained, I/O-intensive workloads often accompanies sustained performance losses. This fact is particularly relevant to the increasing fraction of experimental and observational data that is being processed on modern HPC platforms; as the volume of data being continually streamed from large-scale scientific instruments increases, the effects of sustained bandwidth contention are likely to become increasingly prominent.

Another noteworthy feature that this method reveals is the bimodality of correlation between performance and the CPU load of the file system data servers ("Data Server Load" in Fig. 8) on the Lustre-based test platforms. A time-resolved view of the regions where performance correlates with data server CPU loads (Fig. 9) reveals that the bimodality of the correlation matches the biomodality observed in the HACC write workload on the affected storage systems. During the long-term performance regression discussed in Section IV-B, high CPU load on the Lustre Object Storage Services coincided with low performance of the I/O performance probes. As soon as performance was restored on August 10, the relationship reversed, and high CPU load was observed favorably with respect to performance.

The positive correlation between performance and CPU load is consistent with the data servers using CPUs primarily to service incoming I/O requests, whereas the negative correlation indicates that another CPU load (as may be caused by an algorithmic bug) was present and competed with the data servers' ability to use CPU to service those same requests. From these results we conclude that not only does baseline I/O performance vary with time but also the nature and magnitude of how different attributes correlate with I/O performance change over time. Had this correlation analysis been performed without partitioning over divergence regions, the regions of

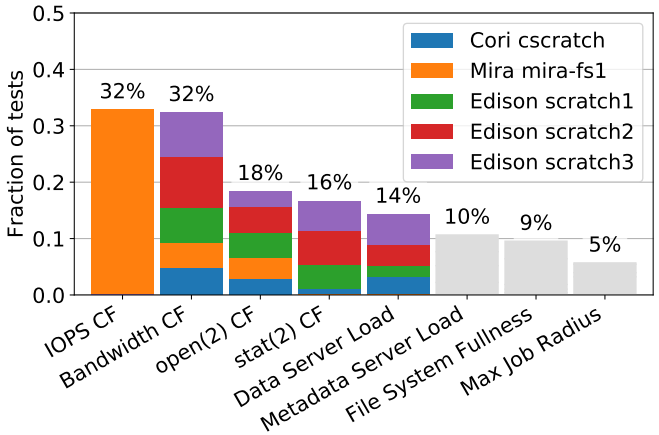

Fig. 10. Attributes that correlated with poor I/O performance across all file systems and benchmarks tested, normalized to the number of probes during which each attribute was measured. Mira was the only system for which IOPS coverage factor was measured. Gray bars are attributes whose rate of classification were not statistically significant. Percentages do not add up to 100% because multiple attributes are often classified as contributors for a single anomalous observation.

positive and negative correlation would have obfuscated each other in the net result.

The remaining two attributes that were found to correlate with performance are file system fullness and `open(2)` coverage factor (a measure of metadata resource contention). The two instances of file system fullness correlating negatively with performance are clear and were corroborated with independent observations from facilities staff. These two regions encapsulate periods when their respective file systems approached 90% fullness for a period of several days. The observed loss of performance is consistent with Lustre's known susceptibility to significant performance degradation as OSTs approach 90% fullness [30], [1].

The correlation with `open(2)` coverage factor is intuitive because this metric acts as a proxy for metadata contention. One of these divergence regions was found to overlap with an unusually extensive, long-running multiday purge of the Edison scratch2 file system. However, at this time the cause of the other correlated divergence region is unclear. (We will continue to investigate.)

C. Transient performance loss

In addition to characterizing long-term performance issues, it is advantageous to determine the reasons why I/O performance is severely degraded for one and only one day in an otherwise unremarkable period of time. Such performance losses, indicated by isolated dark blocks in Fig. 4, may be observed in only one of the I/O performance probes issued on a given day and suggest a very short-lived issue that disappeared over the course of one or two of the eight daily probes. The lack of a consistent performance trend surrounding these transients makes them difficult to correlate with other metrics as was done in the preceding section, necessitating a different approach to characterizing them.

To address this need, we apply the same strategy of partitioning performance observations into divergence regions and then performing statistical analysis within each region. To identify individual performance anomalies in a divergence region and classify the factors which contributed to them, we apply the following binary classification method.

1) We examine the feature vector for every observation in the divergence region, and for each attribute a we determine the observation where that attribute's measured value was at its lowest, $\min(a)$.
2) We then define the *anomalous observation* for the divergence region as the observation whose feature vector contains $\min(a)$ for fraction peak performance. Since divergence regions contain observations of similar performance by definition, this anomalous observation is truly anomalous—it is differentiated from the long-term performance trends described in Sec. IV as well as the ones within its own divergence region.
3) We then determine which other $\min(a)$ values also fall in this anomalous observation's feature vector, and we classify those attributes as contributors to the anomalous observation's performance.

Qualitatively, this process codifies the conjecture that if I/O performance is anomalous on a specific day, the other measured attributes that were also anomalous at that time are what contributed to that behavior. Quantitatively, this simple classification scheme allows us to identify relationships and define the statistical significance of each classification for individual anomalous observations using p-values.[2]

The result of this process is zero or more attributes being positively classified as contributors to anomalous performance. For example, if the lowest values of the bandwidth coverage factor and IOPS coverage factor attributes occur in the same feature vector as the lowest value for performance, we classify both bandwidth and IOPS coverage factors as contributors to that anomalous observation's performance.

To apply this method, we first group observations into sets of data by the test platform on which they ran. These sets are then further subdivided according to I/O motif and read/write mode of the probe to classify anomalies at full temporal resolution and motif-level granularity. The net result are $5 \times 4 \times 2 = 40$ sets of data, each representing a unique combination of test platform ($5\times$, as listed in Table I), I/O motif ($4\times$, per Table II), and whether the probe was reading or writing ($2\times$). Schematically, each horizontal row in Fig. 3 represents a single set. For each set of observations, SMAs are calculated, crossover points are defined, and the set is partitioned into a set of divergence regions.

This partitioning results in 1,146 divergence regions across 40 sets of observations. For each such region, we then apply the aforementioned binary classification to identify anomalous observations and classify attributes that affected performance. All statistically insignificant classifications (p-value > 0.10) are discarded, in order to eliminate divergence regions that

[2]The p-value is the probability of making a positive classification in the absence of an underlying relationship or, equivalently, the probability of positively classifying a random value. Since there is only one $\min(a)$ in each region of N observations, the p-value for our classifications is thus $\frac{1}{N}$.

are too small to make any meaningful classifications. The final products are 490 anomalous observations, 410 (84%) of which have at least one positively classified attribute.

Not every observation's feature vector has every feature defined as a result of some monitoring components being offline at the time of a test. To avoid biasing our results away from those attributes that were measured for only a subset of observations, we express the importance of each attribute as the number anomalous observations where it was positively classified divided by the total number of observations where it was measured. Figure 10 enumerates the attributes that were positively classified the most times.

As with the longer-term performance divergence characterized in Section V-B, high contention for bandwidth also coincides with short-term performance transients. In contrast, contention for IOPS is positively classified in a significant fraction of anomalous observations despite its not arising as a factor in longer-term performance divergence. This contrast suggests that IOPS contention becomes a significant contributor to performance degradation only in short-term transients. Conversely, we conclude that periods of high IOPS contention do not last long enough to be implicated in long-term performance degradation on production file systems, whereas the same is not true of bandwidth contention. These are two distinct forms of performance variation that require unique investigation techniques to uncover.

We also observe the coincidence of anomalous observations and anomalous metadata coverage factors and CPU load on data servers to a less significant degree. This is consistent with the moderate correlations shown in Figure 8. Unlike the correlative analysis, however, this transient analysis shows that metadata contention impacts individual jobs on every test platform, and it is observed at much higher frequency on short time scales. This indicates that, like IOPS contention, metadata contention is much more likely to coincide with transient performance loss than a long-term performance divergence.

Metadata server CPU load, file system fullness, and maximum job radius were also classified in some anomalous observations. However, the low number of anomalous observations in which they appeared calls into question the statistical significance of these three findings. To address this result, we calculate the p-values (the probability of observing the same number of classifications in the absence of a true underlying relationship) for each metric using binomial tests.[3]

These significance tests reveal that the number of times metadata server load, file system fullness, and maximum job radius were classified is statistically insignificant. Whereas the leftmost five attributes in Fig. 10 all have p-values of 5×10^{-5} or lower, the insignificant metrics are 0.15 or higher. Qualitatively, these negative findings are not unreasonable; for example, file system fullness is most often a degenerative, long-term health problem, as was identified in Section V-B and prior work [30], [1]. Thus, while it may coincide with transient anomalies, it is unlikely to be the sole contributor to poor performance for a transient anomaly.

The classification process and analysis described here demonstrate that one can apply statistical analysis to fine-grained divergence regions and still obtain statistically significant insight into the causes of transient performance degradation. While we chose a simple binary classification criterion based on $\min(a)$, this process could be applied using any classification methodology that identifies relationships between feature vectors and quantifies the associated statistical significance.

[3]We use one-tailed binomial tests with the number of positive classifications (k) and total observations (n) for each metric. The fact that our dataset was filtered to include only regions with p-value < 0.10 allows us to use 0.10 as a conservative value for the probability of success (p).

VI. Findings and Implications for State of the Practice

We found that the combination of systemwide telemetry, active performance probes, modular data integration, and generalized analysis tools was highly effective in deriving insight from otherwise opaque large-scale storage systems. Over the course of this study we codified successful techniques, such as adaptations of financial analysis strategies, into our open source TOKIO framework (see Appendix A) and contribute to the state of the practice by making this framework readily available so that the methodology is repeatable on other platforms. Our framework notably decouples analysis techniques from data integration so that the the same analysis tools can be reused at any facility once modular connectors have been added to normalize that facility's telemetry data.

We also contribute to the state of the practice by uncovering novel insights into the nature of I/O performance in production storage systems. In the remainder of this section, we revisit the most significant outcomes and provide corollaries that improve the ability for HPC storage practitioners to contextualize, quantify, and analyze I/O performance variability on large-scale storage systems.

A. Understanding large-scale storage system behavior

The following findings refine our understanding of how large-scale storage systems behave at a high level.

- **Baseline I/O performance and variability are not constant over time:** This observation has direct implications for our specifications and expectations of system performance. It is unrealistic to assume that benchmarks performed upon system delivery will accurately represent performance over time, and performance expectations must be recalibrated as storage systems age. We also note that our analysis was not able to account for all variability; additional work is needed in both analysis techniques and monitoring.
- **Attributes that correlate with transient performance problems often differ from those that correlate with long-term performance problems:** I/O-intensive workloads in HPC are widely known to be unusually bursty compared with other I/O-intensive computing workloads, but this study demonstrates that some aspects of HPC I/O workloads (such as IOPS and metadata operations) are more bursty than others (such as bandwidth utilization). Performance

degradation that results from IOPS or metadata contention is unlikely to persist for days, whereas bandwidth contention can result in performance degradation at all time scales.

B. Improving monitoring and telemetric coverage

The following findings motivate further advances in how large-scale storage systems are monitored and what data sources are required.

- **Administrative activities such as system patches and updates are a significant source of time-dependent, long-term performance variation**: HPC systems are complex, and their upgrades may be dictated by external factors including maintenance schedules, vendor release cycles, and security disclosures. Thus, capturing explicit measurements before and after the upgrade process may not be possible. Continuous monitoring and active probing of performance mitigate this problem by making such measurements a routine procedure regardless of upgrade schedule, much in the same way that continuous integration testing automatically monitors software development processes. Every large-scale facility incorporates testing procedures into its upgrade strategy, but this study highlights the need for breadth of performance testing across workload motifs.
- **Holistic I/O monitoring should incorporate environmental provenance information such as kernel, OS, and file system versions**: This is an obvious finding in retrospect but is not widely taken into account in current instrumentation tools. Tools such as Darshan and LMT should capture sufficient environmental information alongside conventional performance measurements to aid in correlation between performance losses and environmental changes.
- **Bandwidth contention from sustained, I/O-intensive workloads often accompanies sustained performance losses**: The impact of bandwidth contention on I/O performance is widely supported in the literature, but this study demonstrates time-dependent behavior not previously measured. Detrimental contention can occur over time spans lasting several weeks (e.g., aggregate workload due to project allocation timing) and may be driven by factors not captured by conventional HPC monitoring (e.g., wide area transfers or archival traffic). This calls for a broadening of the definition of "holistic I/O characterization" to include not just the full HPC I/O stack but also the auxiliary resources that utilize the storage system.

C. Analyzing evolving storage systems

The following findings highlight ways in which the accuracy and significance of I/O analysis and modeling methodologies can be improved.

- **Significantly stronger correlations can be found by focusing analysis on algorithmically identified regions of interest in the data**: With the proliferation of machine learning, it is tempting to apply unguided techniques to a large data set in a bid to extract meaning from the data set. However, we have demonstrated that a systems practitioner will gain more confident insights from targeted analysis of relevant regions of interest. The region identification method need not depend on SMAs, and alternative approaches for both partitioning time series data and classifying the measurements within regions can be easily replaced. The simple technique we applied here still identified causes of poor performance in 84% of the jobs that experienced transient performance loss. Applying more sophisticated classification methods is likely to improve upon this.
- **The nature and magnitude of how different attributes correlate with I/O performance also change over time:** This observation has critical ramifications for developing I/O performance models. Models developed from a training set without temporal context will produce incorrect predictions, even on the same target system, if external factors have caused the performance of that system to evolve. For example, high CPU load was found to correlate with favorable performance under healthy file system conditions, yet it also coincides with unfavorable performance when non-I/O workloads are impacting storage servers.
- **Financial market analysis techniques can be adapted to I/O performance time series data to attenuate noise and identify underlying trends**: Processing a large, continuously expanding, and noisy time series data set is not unique to I/O performance characterization. We found similarities between our data set and financial market data that enabled straightforward adaptation of known techniques and terminology. More advanced market analysis strategies or strategies from other fields would likely be more effective.

VII. Conclusion

Our study of a year in the life of a parallel file system provided a variety of insights into performance variation on production systems, including both long-term trends and transient anomalies. These insights, and the methods that we used to derive them, have implications for instrumentation methods, administrative expectations, and analysis techniques. Some of the recommendations in these findings can be acted on now by broadening the scope of instrumentation or enhancing analysis tools based on observations in this paper. Others more fundamentally motivate the need for "live" analysis of production systems so that the lessons learned here (especially those related to dynamic, time-dependent behavior) can be more generally extracted at any time from a running system to produce actionable feedback.

Near-term future work calls for the development of additional TOKIO connectors to incorporate additional storage resources such as burst buffers. In the long term, we plan to develop methods to systematically classify the similarity of different regions from one another and enable the determination of broad classes of performance regions. We will also use the measured data as input for simulation frameworks to enable the design of potential new file system features or policies that may reduce the amount of I/O performance variation seen in production.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contracts DE-AC02-05CH11231 and DE-AC02-06CH11357 (Project: A Framework for Holistic I/O Workload Characterization, Program manager: Dr. Lucy Nowell). 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 and the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. The authors also thank Kevin Harms and Zachary Nault for their insight on the Mira file systems and support during the experiment, and Kirill Lozinskiy for assisting with confirming root causes on the NERSC file systems.

REFERENCES

[1] 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 - PDSW-DISCS '17*. New York, New York, USA: ACM Press, 2017, pp. 55–60. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3149393.3149395

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

[3] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, nov 2014, pp. 154–165. [Online]. Available: http://ieeexplore.ieee.org/document/7013000/

[4] J. M. Kunkel, M. Zimmer, N. Hübbe, A. Aguilera, H. Mickler, X. Wang, A. Chut, T. Bönisch, J. Lüttgau, R. Michel, and J. Weging, "The SIOX architecture — coupling automatic monitoring and optimization of parallel I/O," in *Proceedings of the 29th International Conference on Supercomputing - Volume 8488*, ser. ISC 2014. New York, NY, USA: Springer-Verlag New York, Inc., 2014, pp. 245–260.

[5] S. A. Wright, S. D. Hammond, S. J. Pennycook, R. F. Bird, J. A. Herdman, I. Miller, A. Vadgama, A. Bhalerao, and S. A. Jarvis, "Parallel file system analysis through application i/o tracing," *The Computer Journal*, vol. 56, no. 2, pp. 141–155, Feb 2013.

[6] 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," Google, Inc., Tech. Rep., 2010. [Online]. Available: https://research.google.com/archive/papers/dapper-2010-1.pdf

[7] 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*, ser. HPDC '15, 2015, pp. 33–44. [Online]. Available: http://doi.acm.org/10.1145/2749246.2749269

[8] 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)*, May 2017, pp. 442–451.

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

[10] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf, "Managing Variability in the IO Performance of Petascale Storage Systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, no. November. IEEE, nov 2010, pp. 1–12. [Online]. Available: http://ieeexplore.ieee.org/document/5644883/

[11] 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 International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, may 2016, pp. 750–759. [Online]. Available: http://ieeexplore.ieee.org/document/7516071/

[12] 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)*, vol. 7, no. 3, p. 8, 2011.

[13] H. S. Gunawi, R. O. Suminto, R. Sears, C. Golliher, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey, G. Grider, P. M. Fields, K. Harms, R. B. Ross, A. Jacobson, R. Ricci, K. Webb, P. Alvaro, H. B. Runesha, M. Hao, and H. Li, "Fail-slow at scale: Evidence of hardware performance faults in large production systems," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*. Oakland, CA: USENIX Association, 2018, pp. 1–14. [Online]. Available: https://www.usenix.org/conference/fast18/presentation/gunawi

[14] G. K. Lockwood, N. J. Wright, S. Snyder, and P. Carns, "TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis," in *Proceedings of the 2018 Cray User Group (CUG'18)*, 2018.

[15] 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*. IEEE, 2009, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5289150

[16] S. Snyder, P. Carns, K. Harms, R. Latham, and R. Ross, "Performance evaluation of Darshan 3.0.0 on the Cray XC30," Argonne National Laboratory (ANL), Argonne, IL (United States), Tech. Rep., 2016.

[17] J. Garlick and C. Morrone, "Lustre monitoring tools," 2010. [Online]. Available: https://github.com/LLNL/lmt

[18] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 44–60.

[19] S. Byna, M. Howison, and A. Sim, "Parallel I/O Kernel (PIOK) Suite," 2015. [Online]. Available: https://sdm.lbl.gov/exahdf5/software.html

[20] M. M. A. Patwary, S. Byna, N. R. Satish, N. Sundaram, Z. Lukić, V. Roytershteyn, M. J. Anderson, Y. Yao, Prabhat, and P. Dubey, "BD-CATS: Big Data Clustering at Trillion Particle Scale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015, pp. 6:1–6:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807616

[21] S. Habib, V. A. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. A. Insley, D. Daniel, P. K. Fasel, N. Frontiere, and Z. Lukic, "The universe at extreme scale: Multi-petaflop sky simulation on the BG/Q," *CoRR*, vol. abs/1211.4864, 2012. [Online]. Available: http://arxiv.org/abs/1211.4864

[22] "IOR - Parallel file system I/O benchmark." [Online]. Available: https://github.com/hpc/ior

[23] H. Bui, H. Finkel, V. Vishwanath, S. Habib, K. Heitmann, J. Leigh, M. Papka, and K. Harms, "Scalable Parallel I/O on a Blue Gene/Q Supercomputer Using Compression, Topology-Aware Data Aggregation, and Subfiling," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, feb 2014, pp. 107–111. [Online]. Available: http://ieeexplore.ieee.org/document/6787260/

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

[25] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. M. Wild, "Machine learning based parallel I/O predictive mod-

eling: A case study on Lustre file systems," in *International Conference on High Performance Computing*. Springer, 2018, pp. 184–204.

[26] F. James, "Monthly moving averages–an effective investment tool?" *Journal of financial and quantitative analysis*, vol. 3, no. 3, pp. 315–326, 1968.

[27] A. Gunasekarage and D. M. Power, "The profitability of moving average trading rules in south asian stock markets," *Emerging Markets Review*, vol. 2, no. 1, pp. 17–33, 2001.

[28] J. Fullop and J. Green, "Nuclear Meltdown? Assessing the impact of the Meltdown / Spectre bug at Los Alamos National Laboratory," in *Proceedings of the 2018 Cray User Group*, Stockholm, 2018.

[29] R. Falk and A. D. Well, "Many faces of the correlation coefficient," *Journal of Statistics Education*, vol. 5, 1997.

[30] S. Oral, J. Simmons, J. Hill, D. Leverman, F. Wang, M. Ezell, R. Miller, D. Fuller, R. Gunasekaran, Y. Kim *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*. IEEE Press, 2014, pp. 217–228.

APPENDIX

A. Abstract

This work demonstrates the benefits of applying active I/O performance probes with holistic I/O monitoring to identify the causes of short- and long-term performance variation. This work is based on four components that can be applied to either replicate the exact results presented in this study or reproduce this experiment on a completely new system. To replicate the exact results, the *year-long dataset* is required. To conduct this experiment on an entirely new system, however, the *TOKIO Automated Benchmark Collection* is used instead to create a new input dataset.

1) Year-long dataset: The year-long dataset contains all feature vectors that were generated during this study and used to create all figures and findings presented in this work. The dataset is a CSV file containing 11,986 feature vectors, each corresponding to a single job and containing up to 220 attributes. Of these attributes, 96 come from application-level monitoring, 101 come from file system workload monitoring, 21 come from file system health monitoring, 6 come from resource manager monitoring, and the remainder are job-wide metadata. The exact number of attributes defined in each vector depends on what connectors were available on the target platform at the time the job ran.

2) TOKIO Automated Benchmark Collection (TOKIO-ABC): TOKIO-ABC is a software repository containing specific versions of the benchmarks used, scripts that encode the specific build processes used at NERSC and ALCF, and the exact input parameters used in all tests. This repository contains all the necessary code and inputs to generate a new dataset for a new HPC system. For brevity, the input parameters are not reproduced here but are provided in the TOKIO-ABC repository.

3) TOKIO-ABC utilities library (tokio-abcutils): tokio-abcutils is a Python library upon which the analyses performed in this study were built. It contains routines for calculating SMAs from feature vectors and performing each correlation analysis presented. tokio-abcutils also includes the Jupyter notebooks that were used to generate all figures in this work.

4) pytokio framework: pytokio [14] is an implementation of the TOKIO framework that is required by tokio-abcutils. It is freely available on GitHub[4], and the instructions included in its README are sufficient to meet the requirements of tokio-abcutils.

B. Description

1) Checklist (artifact metainformation):

- **Program:** pytokio, TOKIO-ABC, tokio-abcutils
- **Compilation:** pytokio and tokio-abcutils require no compilation; TOKIO-ABC includes autoconf-based build scripts to aid in compilation and deployment
- **Data set:** Features vectors from all 11,986 benchmark runs
- **Run-time environment:** pytokio and tokio-abcutils require Python 2.7, pandas 0.23, matplotlib 2.2, numpy 1.13, and scipy 0.19; TOKIO-ABC requires MPI-3.0 and HDF5 1.8.
- **Run-time state:** No specific state is required a priori.
- **Execution:** pytokio and tokio-abcutils: command-line tools and example Jupyter notebooks; TOKIO-ABC: either a continuous integration system (e.g., Jenkins) or from cron jobs.
- **Output:** tokio-abcutils: graphical and numerical results of intermediate analyses; TOKIO-ABC: Darshan logs and job logs
- **Experiment workflow:** TOKIO-ABC is set up to run automatically via Jenkins, cron, or a similar scheduling tool; pytokio's `summarize_job` [14] tool is used to generate feature vectors; Jupyter notebooks in tokio-abcutils performs statistical analyses using these feature vectors.
- **Experiment customization:** pytokio: via additional TOKIO connector packages or new analysis tools; tokio-abcutils: by extending analysis routines and plots in Jupyter notebooks; TOKIO-ABC: via input decks
- **Publicly available?:** Yes

2) How software can be obtained (if available): All components of TOKIO used in this study are BSD-licensed. pytokio[5], TOKIO-ABC[6], and tokio-abcutils[7] are available online at the specific software versions used in this work.

3) Hardware dependencies: There are no specific hardware dependencies for TOKIO or TOKIO-ABC. There may be hardware dependencies of the component-level monitoring tools with which TOKIO integrates, but such tools are not considered artifacts of this work.

4) Software dependencies:

a) pytokio v0.10.0 and tokio-abcutils v1.0.0: pytokio and tokio-abcutils require Python 2.7, and the work here additionally used pandas 0.23.0, matplotlib 2.2.2, numpy 1.13.1, and scipy 0.19.1. For simplicity, we opted to use the software environment provided by Continuum IO's Anaconda version 4.4.7 with the aforementioned four libraries explicitly upgraded to the stated versions.

pytokio also relies on a number of component-level monitoring tools. As described in this paper, TOKIO can integrate with any tool that provides scalar or time-resolved data types, and pytokio includes interfaces for the following data sources:

- Darshan 3.1.3
- ggiostat
- LMT (as provided by Neo 2.x on Cray ClusterStor)
- Lustre 2.5.1 (health monitoring via lfs and lctl)
- Slurm 17.02.1-2 and CLE 6.0 (job topology at NERSC)

b) TOKIO-ABC v1.0.0: The Automated Benchmark Collection is a metapackage that contains the specific versions of each benchmark used, specific patches applied to those upstream versions, and scripts that configure and build the collection. Its external dependencies are those of the benchmark applications, which include the following:

- autoconf 2.69 or later
- automake 1.13 or newer
- an MPI 3.0-compliant implementation of MPI
- HDF5 1.8.14

Further details on known issues and specific version incompatibilities are documented in the TOKIO-ABC package.

[4]https://www.github.com/nersc/pytokio

[5]https://doi.org/10.5281/zenodo.1345790

[6]https://doi.org/10.5281/zenodo.1345784

[7]https://doi.org/10.5281/zenodo.1345786

5) Datasets: To validate this work and allow the community to build upon our experimental data, the entire year-long dataset is available online[8] and includes the following:

- All feature vectors corresponding to all 11,986 jobs run over the experimental period
- Unmodified Darshan logs for each TOKIO-ABC job

This CSV dataset represents TOKIO-ABC results from the five test environments presented: Edison scratch1, scratch2, scratch3; Cori scratch, and Mira mifa-fs1. The dataset is labeled with feature names encoded as the first CSV line. The mapping between feature names and their descriptions is also documented.

C. Installation

After unpacking the python package source distribution, installing these libraries is a matter of performing "`pip install -r .`". Build scripts for Mira, Cori, and Edison are included in the TOKIO-ABC source distribution, and the specific configure options used for each benchmark are defined near the top of each script. These scripts should be portable to any Blue Gene/Q and Cray Linux 6 environment, and only minor modification should be required to build TOKIO-ABC on commodity platforms. These build scripts configure, build, and install the benchmarks within the unpacked source repository itself, and all of the utility scripts provided assume this self-contained installation directory structure.

D. Experiment workflow

The tokio-abcutils source repository includes a `README` file that describes the experimental workflow in detail. The following summarizes the workflow.

1) TOKIO-ABC is set up to run the benchmark applications (HACC, VPIC, BD-CATS, and IOR) daily. This can be done via any automated scheduling mechanism; NERSC uses cron, and ALCF uses Jenkins.
2) The `summarize_job` [14] utility included with pytokio is used to generate feature vectors for each benchmark result and save them as CSV-formatted values that can be read by tokio-abcutils.
3) The tokio-abcutils repository contains the plotting routines, statistical analysis functions, and Jupyter notebooks to explore the resulting feature vectors.

Each figure presented in this manuscript is the result of an analysis notebook that is provided in tokio-abcutils repository. Furthermore, tokio-abcutils includes a special subpackage specific to this manuscript, `sc18paper.py`, that implements and describes the specific data filtering used here. For example, it demonstrates that this paper discounted all jobs whose Darshan log reflected less than 1 GiB of I/O being performed; this criterion was required in order to invalidate results from several days where the BD-CATS binary on Edison was conflicting with an update to the default HDF5 library on that system that resulted in BD-CATS performing no actual I/O.

[8]https://dx.doi.org/10.5281/zenodo.1345780

All components of the analyses presented here (including the feature vector dataset, pytokio, and tokio-abcutils) have been tested and confirmed to run on generic UNIX-like environments outside of NERSC and ALCF (e.g., on standard MacBooks with macOS 10.13).

E. Evaluation and expected result

The figures presented in this paper can all be regenerated or manipulated by using the dataset, pytokio, and tokio-abcutils. The Jupyter notebooks included in tokio-abcutils generate PDFs or PNG images, and most notebooks also generate numerical results that can be exported as CSVs.

Detailed documentation for each analysis, its expected inputs and outputs, and the analytical methods they apply are provided as Markdown cells in each notebook. Special care has been taken to explain each step in each notebook's analysis process for the benefit of the authors' collaboration, but these notes will also benefit anyone interested in reproducing the results presented in this work.

F. Experiment customization

Customization can be done by either adding new TOKIO connectors to interface with different component-level monitoring tools or creating new analysis tools which operate on the year-long dataset. As an example of the former, contributing a connector that can access traffic counters to burst buffer file systems (such as collectd for DataWarp [14]) would allow pytokio, and by extension this work, to provide more insight into how various factors may contribute to burst buffer performance in contrast to disk-based parallel file systems. An example of the latter would be modifying the analysis notebooks to generate plots for the combinations of systems and analysis that were *not* presented in this manuscript because of space constraints.

The input parameters for each performance probe benchmark are defined in `.params` files in the `inputs` subdirectory of the TOKIO-ABC repository. These files are simple text files that specify the desired parallelism, working set size, and read/write behavior for an individual job on each line. Application-specific documentation is provided in the repository; the most common customization is to alter the data volume and parallelism to suitably stress the underlying file system without using an excessive amount of core hours.

G. Notes

Significantly more documentation on pytokio, tokio-abcutils, TOKIO-ABC, and the year-long dataset used in this study are contained in each repository. For additional clarity, many of the tools used to generate the figures for this manuscript are also included as self-documented Jupyter notebooks. Both pytokio and tokio-abcutils follow the Google Python Style Guide[9]. In combination with Sphinx Autodoc,[10] pytokio and tokio-abcutils include extensive self-documentation of their APIs that can be rendered as PDF- or HTML-formatted manuals.

[9]https://google.github.io/styleguide/pyguide.html

[10]http://www.sphinx-doc.org/en/stable/ext/autodoc.html