MOANA: Modeling and Analyzing I/O Variability in Parallel System Experimental Design

Kirk W. Cameron, Ali Anwar, Yue Cheng*, Li Xu, Bo Li, Uday Ananth, Jon Bernard, Chandler Jearls, Thomas Lux, Yili Hong, Layne T. Watson, Ali R. Butt
Virginia Tech, **George Mason University*

Abstract—Exponential increases in complexity and scale make variability a growing threat to sustaining HPC performance at exascale. Performance variability in HPC I/O is common, acute, and formidable. We take the first step towards comprehensively studying linear and nonlinear approaches to modeling HPC I/O system variability in an effort to demonstrate that variability is often a predictable artifact of system design. Using over 8 months of data collection on 6 identical systems, we propose and validate a modeling and analysis approach (MOANA) that predicts HPC I/O variability for thousands of software and hardware configurations on highly parallel shared-memory systems. Our findings indicate nonlinear approaches to I/O variability prediction are an order of magnitude more accurate than linear regression techniques. We demonstrate the use of MOANA to accurately predict the confidence intervals of unmeasured I/O system configurations for a given number of repeat runs – enabling users to quantitatively balance experiment duration with statistical confidence.

Index Terms—Variability, performance modeling, machine learning.

1 Introduction

Emergent high performance computing (HPC) systems must scale to meet the ever-growing demands of many grand challenges for scientific computing. Performance variability[1] increases with system scale and complexity.

Highly variable observations in large complex systems can make it difficult, and sometimes even impossible, to fully optimize for performance. Such variability is cited as a significant barrier to exascale computing [21, 35]. Unfortunately, variability (which includes OS jitter [27]) is both ubiquitous and elusive as its causes pervade and obscure performance across the systems stack from hardware [17, 29] to middleware [1, 26] to applications [14] to extreme-scale systems [35, 42]. Additionally, as a number of recent reports attest [21, 35], performance variability at scale can significantly reduce performance and energy efficiency [4, 11].

Variability is a persistent challenge in experimental systems research. Table 1 lists the 7 parameters in our HPC I/O variability study (more details about the study are available in Section 3). In this example, one would need $\binom{7}{2} = 21$ pairwise plots to display the two-way relationship of parameter effects on variability. Assuming it takes 30 seconds for a benchmark run, it would take over 8 hours to test just 1,000 configurations. For 95% statistical significance, we would run this configuration tens of times or more which would take weeks. Brute force experiments were used in this work to provide ground truth values for comparison to our variability prediction methods. For the 7 variables studied with runtimes averaging more than 30 seconds per run, the experiments in this paper took nearly 8 months in total with nearly uninterrupted runs and 6 dedicated systems.

1. We use the term *performance variability* to describe the general spread or clustering of a measured performance metric such as execution time or throughput. Variability can be expressed in standard statistical terms such as standard deviation.

	Parameters	Number of levels	Levels
Hardware	CPU Clock Frequency (GHz)	15	1.2, 1.4, $\cdots$, 2.9, 3.0
OS	I/O Scheduling Policy	3	CFQ, DEAD, NOOP
	VM I/O Scheduling Policy	3	CFQ, DEAD, NOOP
Application	Number of Threads	9	1, 2, 4, 8, $\cdots$, 256
	File Size (KB)	3	64, 256, 1024
	Record Size (KB)	3	32, 128, 512
	I/O op mode	13	fread, fwrite, $\cdots$

TABLE 1: Parameters used in our study of I/O variability. More details about the study is available in Section 3.

For higher order (e.g., three way or more) relationships and a larger number of parameters (e.g., 100 to 200) that are common in parallel and distributed systems, the parameter space explodes quickly to tens of thousands or millions of possible configurations. Experimental system work at exascale quickly becomes impracticable requiring years of experiments or thousands of nodes to obtain significant results.

While HPC experimental systems researchers have long acknowledged its existence [18, 25], variability is regularly discounted or ignored by the community. While there are notable exceptions (e.g., OS jitter [27], application interference [20, 45], reproducibility [39], scheduling [15]), many researchers in our community have published quality contributions that fail to report box plots or p-values or assumed single-mode population distributions without certainty. This does not typically discredit the contributions and is likely not shoddy science but a consequence of a focus on system design hypothesis testing under practical time constraints. One consequence of accepting the status quo however, is year after year the prevailing literature builds on previous work and the community becomes implicitly complicit in perpetuating these techniques. Furthermore, as noted above, the significance and impact of variability at exascale is potentially impracticable. In our work, we seek to identify scalable variability prediction techniques beginning with statistical first principals.

We believe studying variability is essential to the long term viability of parallel and distributed experimental systems research. In this work, we have conducted experiments on highly variable I/O codes over more than a year to gain the statistical confidence necessary *to determine if variability is a predictable artifact*. We've solicited the collaborative expertise of statisticians to predict the combined multi-variant effects of variability on HPC I/O. Rather than isolating the effects of jitter and resource contention, we propose MOdeling and ANAlysis techniques or MOANA to model the simultaneous, combined effects of a number of variables on I/O application kernels performance variability. MOANA enables researchers to determine the number of repeat experiments required to guarantee a given level of statistical confidence and convergence – without resorting to brute force approaches. While we demonstrate our approach on HPC I/O performance variability, the techniques are widely applicable to experimental design in parallel and distributed systems.

Specifically, we make the following contributions in this paper:

- a detailed empirical study of HPC I/O variability in shared-memory systems for 95K permutations of benchmark and system variables;
- design, implementation, and analysis of MOANA, a nonlinear variability prediction methodology;
- direct validation of the accuracy of MOANA variability predictions versus exhaustive brute-force data (ground truth) collected from 6 highly parallel, shared-memory servers for 48 server-months (6 servers x 8 months each) of runtime;
- a demonstration of the use of MOANA to determine the runs required for a given statistical confidence and convergence for measured and predicted HPC I/O system configurations; and
- a commitment to make the MOANA techniques and our datasets open source and available to the community. [2]

What follows is a detailed study of variability analysis and prediction. In Section 2, we apply the linear analysis approach to the study of variability to highlight the characteristics of variability and the limitations of the linear approach. Section 3 provides an overview of the MOANA approach and the abstract concept of a *variability map*: a model combining application and system characteristics that mathematically captures the differences among experimental configurations. Section 4 demonstrates the use of non-linear training and the application of the variability map concept to variability prediction of unseen system and application configurations. Next, Section 5 describes the three prediction techniques we applied to MOANA: LSP (modified linear Shepard algorithm), MARS (multivariate adaptive regression splines), and Delaunay triangulation approximation. Section 6 demonstrates the application of MOANA to optimize experimental design by quantifying the tradeoffs between repeating experiments and statistical confidence. Sections 7, 8 and 9 respectively discuss our work in the context of existing literature and limitations and future work.

2. URL redacted pending paper review.

2 Limits of Linear Empirical Analyses

In this section, after detailing our experimental setup, we describe a select set of linear, empirical analyses for our dataset. Our goal is to demonstrate that while linear empirical analyses have served the community well for many years, the non-linear characteristics of variability on emergent systems limit the use of linear approaches for prediction. We show that at times linear correlations can provide insights to variability, but as the non-linear effects grow with the introduction of simultaneous changes to more variables, correlations between design and performance variability artifacts become more difficult to predict. We focus on three representative isolated experiments (i.e., varying thread count or file size or record size across frequencies and modes[3] and another experiment relying on the more holistic and widely used "analysis of variance" (ANOVA) technique [33].

2.1 Experimental Setup

We focus on intra-node variability effects, identified as a key barrier to exascale [21, 27, 35], and perform our experiments on parallel shared-memory nodes common to HPC systems. Table 1 summarizes the parameter space for all experiments performed on a lone guest Linux operating system (Ubuntu 14.04 LTS//XEN 4.0)[4] on a dedicated 2TB HDD on a 2 socket, 4 core (2 hyperthreads/core) Intel Xeon E5-2623 v3 (Haswell) platform with 32 GB DDR4. System parameters include: CPU frequency, host I/O scheduling policy (CFQ, DEAD, NOOP), and VM I/O scheduling policy (CFQ, DEAD, NOOP). I/O application parameters include: I/O operation modes (file write, file read, file initial write, etc.), file size, record size, and up to 256 threads. The IOZone benchmark enables control of these settings and up to 6 identical systems were used to speed up the experiments and tasks were distributed to account for any minor manufacturing differences across these machines.

Brute force experiments using all valid permutations of the parameters from Table 1 result in a total of over 95K unique configurations[5]. For each configuration, we conduct 40 runs. Assuming data normality this results in a 95% statistical confidence in the resulting data set. The standard deviation of these 40 runs is used as a proxy for variability without loss of generality. By assuming normality for now, we mirror prevailing approaches in the extant literature and avoid more time-consuming population studies that might take years. Furthermore, we tested and found the accuracy

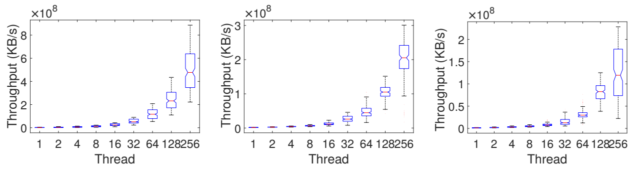
3. Since we observed few differences across the various host and VM scheduling policy combinations, we fix this parameter in our select examples.

4. We selected a widely used, stable version of Ubuntu as representative of current datacenter deployments. The proposed measurement, analysis, and prediction techniques are independent of the system being measured.

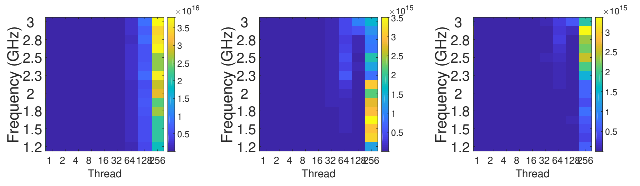
5. We used every unique combination of thread count, frequency, scheduling policy, and op mode mentioned in Table 1. The unique combinations for (file size, record size) are limited to file size > record size resulting in the following combinations: (64, 32), (256, 32), (1024, 32), (256, 128), (1024, 128), and (1024, 512).

of our techniques were only slightly improved without the normality assumption while the experimental work took significantly longer over a span of months.

2.2 Isolated Linear Empirical Analysis

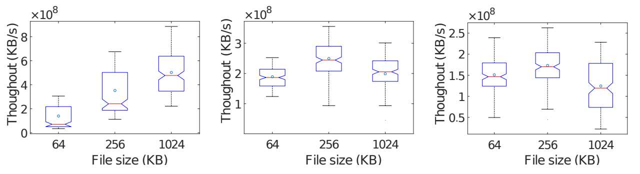


(a) I/O thpt for fread (b) I/O thpt for fwrite (c) I/O thpt for initial write

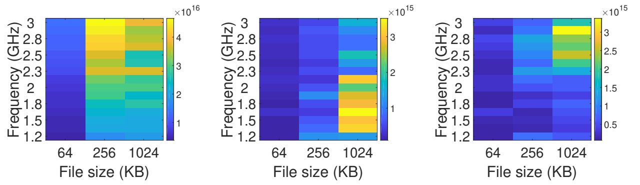


(d) I/O var for fread (e) I/O var for fwrite (f) I/O var for initial write

Fig. 1: I/O throughput (@freq: 1.5 GHz, 2.0 GHz, 2.5 GHZ, and 3.0 GHZ) as a function of number of threads for three different I/O op modes (a, b, and c). Heat map of I/O throughput variance (y-axis-right) as a function of CPU frequency (y-axis-left) and number of threads (x-axis) for three different I/O op modes (d, e, and f). File size = 1024 KBytes, Record size = 32 KBytes.



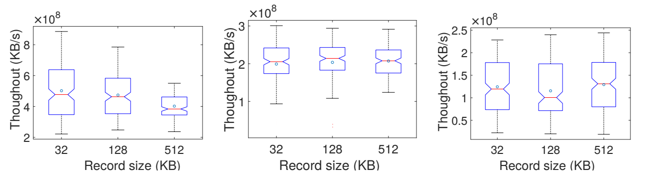
(a) I/O thpt for fread (b) I/O thpt for fwrite (c) I/O thpt for initial write



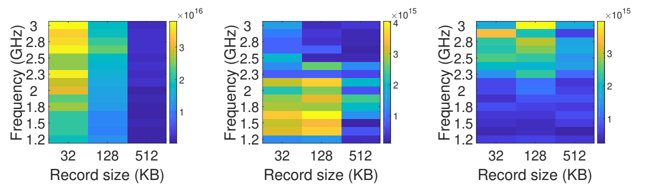
(d) I/O var for fread (e) I/O var for fwrite (f) I/O var for initial write

Fig. 2: I/O throughput (@freq: 1.5 GHz, 2.0 GHz, 2.5 GHZ, and 3.0 GHZ) as a function of file size for three different I/O op modes (a, b, and c). Heat map of I/O throughput variance (y-axis-right) as a function of CPU frequency (y-axis-left) and file size (x-axis) for three different I/O op modes (d, e, and f). Record size = 32 KBytes, Threads = 256.

Effect of number of threads Figure 1 shows experiments designed to examine the effects of number of threads on I/O variability. In this case, raw I/O throughput increases with the number of threads for all three modes (file read, file write, and file initial write)—see Figures 1(a), 1(b), and 1(c). I/O throughput variance also increases with the number of

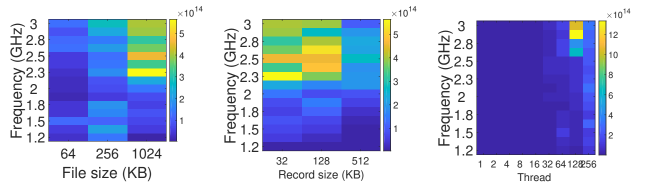


(a) I/O thpt for fread (b) I/O thpt for fwrite (c) I/O thpt for initial write



(d) I/O var for fread (e) I/O var for fwrite (f) I/O var for initial write

Fig. 3: I/O throughput (@freq: 1.5 GHz, 2.0 GHz, 2.5 GHZ, and 3.0 GHZ) as a function of record size for three different I/O op modes (a, b, and c). Heat map of I/O throughput variance (y-axis-right) as a function of CPU frequency (y-axis-left) and record size (x-axis) for three different I/O op modes (d, e, and f). File size = 1024 KBytes, Threads = 256.



(a) file size (b) record size (c) # of threads

Fig. 4: Heat map of change ((a) and (b)) in I/O throughput variance (y-axis-right) from 256 threads (Figure 2(e) and Figure 3(e)) down to 64 threads as a function of CPU frequency (y-axis-left) and file size (x-axis) and record size (x-axis). Heat map of change (c) in I/O throughput variance (y-axis-right) from 1024 KBytes file size (Figure 1(d)) down to 64 KBytes. Results for fwrite are shown.

threads: highest for file read operations (Figure 1(d)) and file write operations (Figure 1(e)) at the highest frequency. File read operations have high variance in the lower frequency range as well (Figure 1(d)). Among our experiments, increases in thread count (independent of scheduler as noted above) had the strongest correlation to increases in variability. Additionally, the variability correlated with thread count mostly dampens the effects of frequency changes on variability. This can be explained by the increase in contention for fixed, shared resources introduced as the number of threads increases. This is a prime example of the value of linear correlation techniques for providing insight to the root causes of variability.

Effect of file size Figure 2 shows experiments designed to examine the effects of file size on I/O variability. Figure 2(a) shows raw I/O throughput increases with file size for file read operations. This is expected as the major I/O time is for seek operations, and increasing file sizes imply each seek is followed by a sequential read of larger data. For file write operations (Figure 2(b)) and file initial write operations

(Figure 2(c)), the I/O throughput initially increases but eventually decreases. Most writes are absorbed in the cache, but as the cache becomes full, the disk flush operations dominate and reduce overall throughput. Figure 2(d) shows that I/O variance for file read operations is highest for medium file sizes and higher CPU frequency. A reason for this is that initially, the throughput is driven by disk seeks, but as the role of seeks decrease, the role of I/O-system interactions become more pronounced. These effects are seemingly amplified by changes in frequency. For file write operations, variance is higher for larger sized files at lower frequency as shown in Figure 2(e). Figure 2(f) shows that for initial write operations, larger file sizes with higher frequency exhibit the most variance. This is harder to explain but potentially due to the need for more disk flushes amplified at the higher CPU frequencie resulting in increases in variability.

We note two observations overall for the file size experiments: 1) file size generally impacts variability though the trends are not consistent across modes; 2) frequency also impacts the variability in a way that is not consistent with the impact of file size. These variables show some correlation to variability, but the correlations are murky, making causality often difficult to determine. Fortunately, several of the authors spent 2 years studying the effects of processor frequency on I/O performance [9]. This work showed that processor frequency alters the arrival rates of I/O requests and ultimately affects the number of epochs required to service requests—and by proxy performance varies with the number of epochs. This phenomenon likely explains more precisely some of the aforementioned observations. For example, observations where smaller requests suffer smaller variability could be caused by these arrival rate and epoch effects. In such cases, the variability is pronounced. However, for other cases such as *fwrite*, these effects are not as pronounced and likely amortized over the duration of the runtime. These experiments exemplify the challenges for linear correlation techniques since they can help identify potential culprits of variability, but finding the root cause can be time consuming or potentially impossible. For example, if the root cause of performance variability was hidden within the hardware design, it could potentially be impossible to isolate without a cycle accurate simulator.

Effect of record size Figure 3 shows experiments designed to examine the effects of record size on I/O variability. Figure 3(a) shows raw I/O throughput decreases with increases in record size for file read operations. For file write operations (Figure 3(b)) and file initial write operations (Figure 3(c)), the I/O throughput remains unchanged. I/O variance results vary extensively. Smaller record size and higher CPU frequency give the highest variation for file read operations. Medium record size and lowest CPU frequency give the highest variation for write operations. Medium record size and higher frequency give the highest variation for initial write operations. See Figures 3(d), 3(e), and 3(f), respectively.

These experiments exemplify the growing challenge for experimental systems. Varying just a few parameters results in two observations: 1) there is no discernible pattern for increases in record size across modes; and 2) frequency impacts variability but in ways significantly different than record size. The combined effects of changes in record size and frequency are non-linear and it is difficult to draw meaningful conclusions from these experiments. This motivates the need to take a more holistic view of variability under such conditions.

2.3 Holistic Linear Empirical Analysis

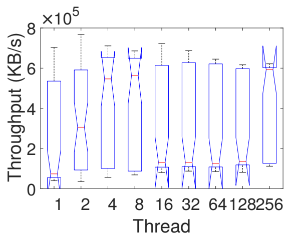
Meta-analysis A meta-analysis of Figures 1- 3 is appealing in search of trends in the data. Consider the following experiment shown in detail for fwrite in Figure 4. We repeat the file size (Figure 4(a)) and record size (Figure 4(b)) experiments calculating the change in variability when the number of threads decreases from 256 to 64. The resulting heat maps show that the variability when changing thread counts (at large and small file sizes) is sensitive to frequency variations. Figure 4(c) shows that variability when changing file sizes is not particularly sensitive to frequency variations (i.e., only the highest frequency seems to matter).

Furthermore, consider Figure 5 showing the per thread I/O throughput as the number of threads increases (x-axis). All of the subfigures on the left of Figure 5 (a, c, and e) use 1.2 GHz for CPU frequency and 64 KBytes file size. All of the subfigures on the right of Figure 5 (b, d, and f) use 3.0 GHz and 1024 KBytes file size.

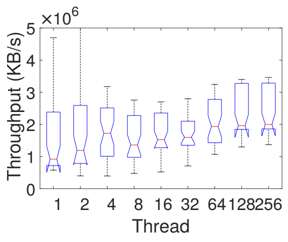
These experiments highlight another challenge for variability-aware systems designs. The combined effects of changes in multiple variables can result in a multivariate optimization problem with multiple, competing variability minimization configurations. This further emphasizes the need for accurate prediction of the non-linear effects of system and application configurations on variability.

ANOVA We used statistical analysis of variance (ANOVA) [33] experiments to identify first-order effects (one-parameter changes) and second-order effects (two-parameter changes) of performance variability. We omit the full results due to space limitations, but in summary, this ANOVA experiment showed that nearly all of the parameters studied (and their second order configurations, e.g., Filesize x Thread) affect I/O variability in a statistically significant way. Unfortunately, this linear method does not expose the relative magnitude of the variability contribution for a given variable. Table 2 shows the use of a related technique, linear regression (LR), results in large inaccuracies ($> 300\%$ average relative error or ARE) when used to predict the non-linear effects of variability.

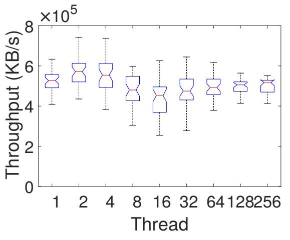
Limitations In these examples, we are only considering a few hundred permutations of I/O modes, CPU frequency, thread count, file size, and record size from among over 95K. This limits these types of analyses to a very small part of the experimental data set. Analyses of the combined effects of more than two variables and their nonlinear interactions is severely limited. While some causality can be inferred from the analyzed data as discussed, any conclusions lack the full context of the data set and cannot be easily generalized. For these reasons, and the manual nature of these approaches, we next consider methods for automating analysis of variability.



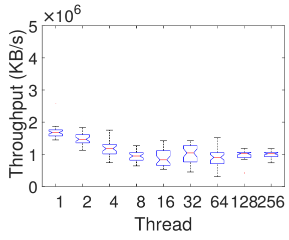
(a) I/O thpt for fread



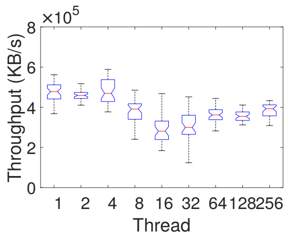
(b) I/O thpt for fread



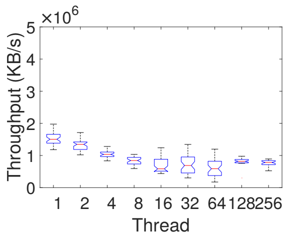
(c) I/O thpt for fwrite



(d) I/O thpt for fwrite



(e) I/O thpt for init write



(f) I/O thpt for init write

Fig. 5: Each subfigure shows the per thread I/O throughput as the number of threads increases. All of the subfigures on the left ((a), (c), and (e)) use 1.2 GHz for CPU frequency and 64 KBytes file size. All of the subfigures on the right ((b), (d), and (f)) use 3.0 GHz and 1024 KBytes file size. Other fixed parameters: host scheduler = CFQ, VM I/O scheduler = NOOP, record size = 32 KBytes.

3 MOANA Methodology

We propose a non-linear modeling and analysis approach (MOANA) that leverages advanced approximation methods to predict I/O performance variability. The methods we have selected—modified linear Shepard (*LSP*) algorithm, multivariate adaptive regression splines (*MARS*), and delaunay triangulation—are capable of approximating nonlinear relationships in high dimensions. This increases the likelihood that given a system and application configuration, the resulting models will accurately predict the variability. In this section, we assume a general familiarity with non-linear, training-based machine learning techniques. For those that would prefer to understand the mathematics and statistics of these approaches before continuing, we suggest reading Section 5 before proceeding with this section. What immediately follows is an overview of the MOANA methodology for accurate variability prediction.

We propose the concept of a *variability map* to describe and predict variability. Let the configuration x be an m-dimensional vector of parameters. The variability map is a function $f(x)$ that gives the variability measure (i.e., standard deviation in our context) at x. The variability map approximation $\tilde{f}(x)$ is constructed from experimental data

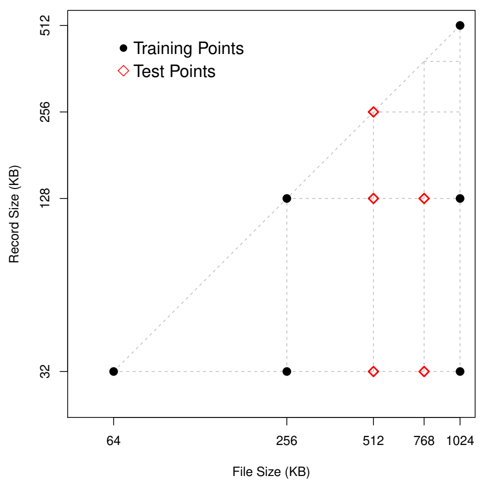


Fig. 6: Plot shows the file size and record size combinations used for training and prediction test sets.

using the LSP, MARS, and delauney methods and can be used to predict variability for any given configuration x.

We use data collected from our brute force approach (Section 2.1) to train our LSP, MARS, and delauney models. Recall from Table 1 the total number of measured configurations is:

$$15(\text{Freq}) \times 9(\text{Thread}) \times 6(\text{Filesize x Recsize}) \times 3(\text{I/O Sche}) \times 3(\text{VM I/O Sche}) \times 13(\text{I/O Op Mode}) = 94770.$$

Note that both File Size and Record Size have three levels; but there are only 6 distinct, valid combinations of File Size and Record Size in the training set. Figure 6 shows the six combinations of the file size and record size that we used for the training set (marked by solid dots).

The methods require a numeric value, x, defined in our experiments as:

$$x = (\text{Frequency, Threads, File Size, Record Size}),$$

which has $15 \times 9 \times 6 = 810$ distinct configurations assuming fixed values for I/O Scheduler, VM I/O Scheduler, and I/O Operation Mode. For each distinct I/O Scheduler, VM I/O Scheduler, and I/O Operation Mode combination, we will build a variability map approximation $\tilde{f}(x)$. In total, we will construct $3 \times 3 \times 13 = 117$ variability maps.

Thus, we can denote the configuration as $x^{(k,\,l)}$, and denote the corresponding variability value as $f_k^{(l)}$, where $k = 1, \cdots, 810$ and $l = 1, \cdots, 117$. For a given l, the dataset is $\{x^{(k,\,l)}, f_k^{(l)}\}, k = 1, \cdots, 810$, and the corresponding variability map approximation $\tilde{f}^{(l)}(x)$ can be obtained by using the LSP, MARS, or delauney algorithms described in Section 5.

Predictor evaluation We define the following relative error as an evaluation criterion. That is

$$r = \frac{|\tilde{f} - f|}{f} \tag{1}$$

where $\tilde{f}$ is the predicted variability at a x, and f is the true variability (obtained from direct measurements).

We use statistical cross validation to compute the average relative error (*ARE*). For a given dataset $\{x^{(k,\,l)}, f_k^{(l)}\}, k = 1, \cdots, 810$, we randomly divide the dataset into two parts where the proportion of one part is p and the remaining proportion is $(1-p)$. We use all configurations from the p portion of the data set as samples to train our predictive models. We use our trained predictors to predict all configurations from the $(1-p)$ portion of the data set. We compute the ARE value for each data point in the test using the average of the relative error, r, for each data point. We repeat this random data set division procedure to construct 117 variability maps. The ARE is averaged again over the 117 trained models, and it will be assumed to be the true variability for the method.

Predictor comparisons By varying p we can observe the tradeoffs between predictor accuracy and the ratio of the training set to the predicted data points. Table 2 shows the ARE for linear regression (LR), LSP, MARS, and delaunay triangulation as functions of the training sample proportion p, averaged over 117 variability maps.

We are unaware of any existing methods to predict performance variability. Hence, we compare the proposed techniques to the general linear regression model. From the results in the table, it is clear that the proposed nonlinear methods out predict the linear regression by an order of magnitude. In this random division testing, the LSP method consistently outperforms MARS. For example, the ARE is around 15% under LSP for a setting that uses 30% of the data for training and predicts 70% of the data. The ARE is around 30% if one uses 10% data for training and 90% data for LSP. If we use half of the data set to predict the other half of the data set ($p = .5$) the ARE of the LSP method is about 12%. Table 2 shows that supervised learning is possible in MOANA runtime systems as good accuracies are possible for small training sets for algorithms such as LSP (20% ARE when only 20% of the data set is used for training).

Training Prop. p	Testing Prop.	LR (%)	LSP (%)	MARS (%)	Delaunay (%)
0.9	0.1	323.09	11.13	56.61	9.97
0.8	0.2	323.47	11.27	57.59	10.19
0.7	0.3	324.00	11.47	58.35	10.48
0.6	0.4	324.58	11.72	59.94	10.88
0.5	0.5	324.59	12.22	62.35	11.42
0.4	0.6	325.82	13.25	66.52	12.22
0.3	0.7	327.02	15.44	71.49	13.56
0.2	0.8	329.56	19.33	79.27	16.13
0.1	0.9	339.32	30.44	100.14	23.39

TABLE 2: ARE for linear regression (LR), LSP, and MARS as functions of p, averaged over 117 variability maps and based on $B = 200$ repeats for random division.

4 Accurate Variability Prediction Using MOANA

In this section we attempt to predict 585 configurations not considered in the full, 95K-configuration training set described in Section 2.1. Figure 6 shows the five combinations of the file size and record size that we used for prediction (marked by diamonds). We calculate the average relative error (ARE) as discussed in the previous section for these new sets of experiments for comparison. Without loss of generality, we limit the experiments to a fixed CPU frequency (2.5 GHz) and a fixed number of threads (128) and

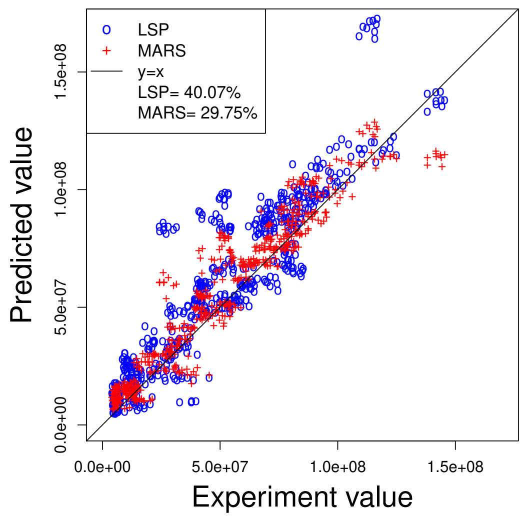


Fig. 7: Comparison of the two proposed prediction models on new configurations. The y-axis shows the predicted standard deviation from our two models, while the x-axis shows the empirical standard deviation from 40 runs. The ARE for each method is also shown on the legend.

we apply both LSP and MARs – though in a later section we apply the third predictor (i.e., Delaunay). We select 5 valid combinations of file size and record size with all possible permutations of I/O scheduler, VM I/O scheduler, and I/O operation modes. Table 3 lists all the parameters in this study.

Throughout this section, we use scatter plots (e.g., Figure 7) to discuss the accuracy of our MOANA methodology for the $5 \times 3 \times 3 \times 13 = 585$ configurations described in Table 3. In Figures 7 – 9 and all subfigures, the y-axis shows the predicted standard deviation for both LSP and MARS models. The x-axis shows the empirical (measured) standard deviation with the unseen configuration. The $y = x$ diagonal line is a reference line that represents predicted values equal to the empirical (measured) values.

File size	Record size	I/O scheduler	VM I/O scheduler	I/O Mode
512	32, 128, 256	CFQ, DEAD, NOOP	CFQ, DEAD, NOOP	All 13 levels
768	32, 128	CFQ, DEAD, NOOP	CFQ, DEAD, NOOP	All 13 levels

TABLE 3: These configurations are not included in the training set. We use MOANA to predict these ”unseen” configurations and compare to our ground truth brute force data to ascertain accuracy.

Use Case I: A general model In this use case, we attempt to determine whether the MOANA approach results in a single model that predicts variability well generally. Figure 7 shows a scatter plot of the general prediction accuracy of our derived models for the $5 \times 3 \times 3 \times 13 = 585$ configurations described in Table 3. The MARS data points (cross point type in Figure 7) are generally clustered closer to the diagonal compared to the LSP data points (circle point type in Figure 7). This is confirmed with average relative error (ARE) calculations: MARS has a 29.75% ARE while LSP has

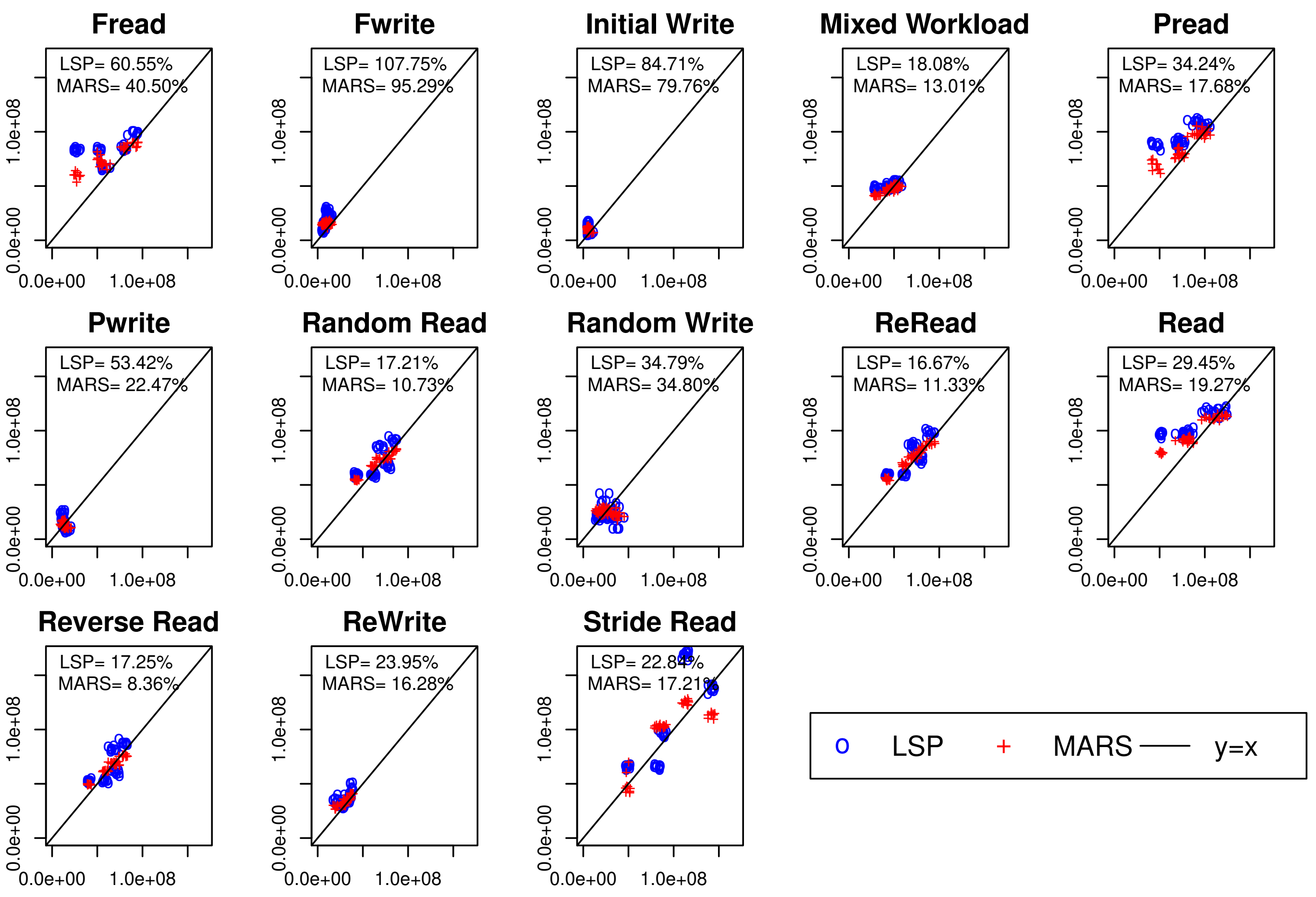


Fig. 8: Prediction scatter plots of different I/O operation modes. Points in the same plot have the same mode. The x-axis is the empirical standard deviation observed from 40 runs. The y-axis is the predicted standard deviation obtained from our two models. The ARE for each method is also shown on the legend.

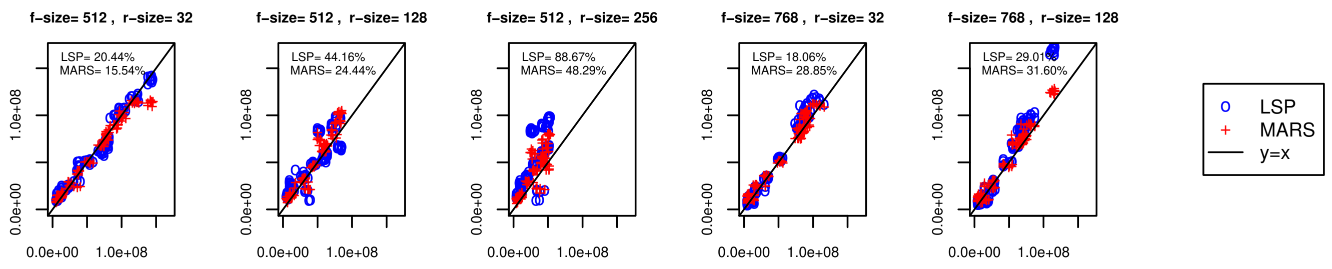


Fig. 9: Prediction scatter plots of different file and record size combinations. Points in the same plot have other parameters fixed. The x-axis is the empirical standard deviation observed from 40 runs. The y-axis is the predicted standard deviation obtained from our two models. The ARE for each method is also shown on the legend.

a 40.07% ARE. This is the opposite finding from the previous section where LSP consistently outperforms MARS.

Upon deeper inspection, we were able to determine that MARS is more sensitive to file size and record size parameters in the training set. Since much of the accuracy gains come from tight data points for continuous variables (due to the logarithmic distance between record sizes), MARS likely gains accuracy due to its sensitivity to these continuous variables. LSP likely performs better when the variables predicted are randomly selected from within the population as done in the training set experiments in the previous section. As expected, the ARE for both methods is generally larger than observed values for LSP and MARS from the previous section (see Table 2) but consistently an order of magnitude better than linear regression.

We use an example to illustrate the detailed analysis for two data points. Consider the cases of file size=512 and file size=768. Depending on the record size variable settings, the ARE values using MARS for file size=512 vary from 15.54% to 48.29% with an average ARE of 29.4% and for file size=768 from 28.85% to 31.60% with an average ARE of 30.23%. These errors are close to the average for the general model, but they would likely be improved with more data points. Upon deeper inspection, it is the record size=256 configuration for the file size=512 that causes the error to be higher in that case. The inaccuracy comes from the sparsity of points at the higher record sizes as mentioned previously.

Use Case II: Variability by application (I/O Mode) In this use case, we attempt to determine whether the MOANA approach can be used to classify the predicted applications for the previously unseen configurations (i.e., I/O Mode in Table 3) by the magnitude of their variability. Figure 8 plots the prediction results for varying I/O operation modes. In all but 3 cases (Fread, Fwrite, and Initial Write), the LSP or MARS predictions beat the average ARE values for the general model from Use Case I (MARS=29.75%, LSP=40.07% in Figure 7). For the relatively accurate cases, the magnitude of variance is lowest and tightly clustered for Pwrite, Rewrite, Random Write. As the variance grows it generally becomes less tightly clustered for Reverse Read and Random Read. Pread, Read and Stride Read show the highest variances and the least clustered results. For 10 of the 13 applications examined, this constitutes a reasonably accurate rank ordering of variability by magnitude and the isolation by application provides some notion of causality by variable.

MOANA's strength, illustrated by this example, is classifying variables by magnitude of variability across a large set of experiments accounting for the nonlinear effects of high-order variables. We leave it to our future work to determine exactly why inaccuracies occur in the Fwrite, and Initial Write (i.e., file write) predictions, but we speculate there are two reasons: (1) page cache operations result in flushes of commit operations to update the disk store and the variability introduced is unaccounted for in our model; and (2) though we did not observe differences in the bare-metal versus VM variability studies, it is possible that under certain extreme write conditions (e.g., the buffer cache fills) the interactions of the nested (host and VM) file systems result in unanticipated performance changes [19]. For Fread, the inaccuracies are not as drastic as Fwrite and Initial Write but still exceed the average ARE for the general model. We speculate these inaccuracies are due to a combination of caching and prefetching effects that can be influenced significantly by the experimental setup (e.g., multi-level cache buffers).

Use Case III: Variability by file and record size In this use case, we attempt to determine whether the MOANA approach can be used to classify the predicted variability for previously unseen configurations of file and record size by the magnitude of their variability. Figure 9 plots the prediction results for valid file and record size combinations from Table 3. In all but 1 case, the LSP or MARS predictions beat the average ARE values for the general model from Use Case I (MARS=29.75%, LSP=40.07% in Figure 7). For the inaccurate case (file size=512, record size=256), we know from Use Case I that the inaccuracy comes from the sparsity of points at the higher record sizes; in other words, the best accuracies occur at the smallest record sizes for both file sizes.

Recall that for this experiment, for each file and record size pair, we vary I/O Scheduler, VM I/O Scheduler, and I/O Mode (or application). The results in Figure 9 indicate that second order effects are influential in the magnitude of the predicted variability. For example, for file size=768 and record size=32, LSP ARE is under 20% and 2-3 distinct clusters are observable in the data. These clusters are affected by the application class (i.e., they are clustered by I/O modes that share characteristics such as writes). Identification of these higher order effects are another valuable contribution of the MOANA approach to analyzing variability. For 4 of the 5 scenarios studied, with consideration of the higher order effects, we can (with reasonable accuracy) rank order variability by magnitude and the isolation by file size, record size, and mode classification provides some notion of causality for a set of variables.

5 MOANA Prediction Models

TABLE 4: Acronyms

GCV	Generalized cross validation error
MARS	Multivariate adaptive regression splines
LSP	Linear Shepard
LR	Linear regression
ARE	Average relative error

TABLE 5: Notation

$x, x^{(k)}$	Vector for system configuration
y	A general notation for variability
f_k	Measured variability
$f(x), \tilde{f}(x)$	True/estimated variability function, mapping x to system variability
r	Relative error of the predicted variability
$W_k(x)$	Weight function for the local approximation $P_k(x)$ at data point $x^{(k)}$
$P_k(x)$	Local linear approximation function in Modified Linear Shepard Algorithm
$d_k(x)$	Euclid distance from k-th sample point to x
$R_\omega^{(k)}$	Radius of influence about $x^{(k)}$ used to define $W_k(x)$
$R_p^{(k)}$	Radius about $x^{(k)}$ within which points are used to compute the least squares fit $P_k(x)$
N_w, N_p	Thresholds used in LSP to define $R_w^(k)$ and $R_p^(k)$
$\omega_{i_j k}$	Coefficient in $P_k(x)$
$\mathcal{B}$	Basis function space in MARS
$\mathcal{M}$	Set for selected function basis in MARS
$h(x)$	Selected basic function in MARS
β	Coefficient for selected basic function

For completeness, we provide details for LSP (modified linear Shepard algorithm), MARS (multivariate adaptive regression splines), and Delaunay triangulation approximation methods. Section 3 described a variability map as a function $f(x)$ that gives the variability measure (i.e., standard deviation in our context) at x. The variability map approximation $\tilde{f}(x)$ is constructed from experimental data and can be used to predict variability for any given configuration x. The modeling framework is currently implemented in R. MARS is available in the R package 'earth'. For the LSP and delauney algorithms, we tailored the existing FORTRAN code and linked it to R. Table 4 and Table 5 provide summaries of acronyms and terms used throughpout this section.

5.1 Modified Linear Shepard Algorithm

The linear Shepard algorithm is derived from the modified Shepard algorithm [40]. Given n distinct data points $x^{(1)}$, …, $x^{(n)}$ and values $f_k = f\big(x^{(k)}\big)$, the linear Shepard algorithm constructs an interpolant to f of the form

$$\tilde{f}(x) = \frac{\sum_{k=1}^{n} W_k(x) P_k(x)}{\sum_{k=1}^{n} W_k(x)} \tag{2}$$

where the locally supported weights are

$$W_k(x) = \left[\frac{\Big(R_w^{(k)} - d_k(x)\Big)_+}{R_w^{(k)} d_k(x)} \right]^2, \quad d_k(x) = \big\|x - x^{(k)}\big\|_2,$$

and the local (linear, here) approximations $P_k(x)$ have the form

$$P_k(x) = f_k + \sum_{j=1}^{m} a_j^{(k)} \big(x_j - x_j^{(k)}\big) \tag{3}$$

The function $P_k(x)$ is the local linear weighted least squares fit around $x^{(k)}$ with the ith data point having weight

$$\omega_{ik} = \left[\frac{\Big(R_p^{(k)} - d_i\big(x^{(k)}\big)\Big)_+}{R_p^{(k)} d_i\big(x^{(k)}\big)} \right]^2, \qquad i \neq k.$$

Let $N_p = \min\{n, \lceil 3m/2 \rceil\}$ and $D = \max_{i,j} \big\|x^{(i)} - x^{(j)}\big\|_2$, and define

$$R^{(k)} = \min\{r \mid \overline{B\big(x^{(k)}, r\big)} \text{ contains at least } N_p \text{ points}\} \tag{4}$$

where $\bar{B}$ is the closure of the open ball $B\big(x^{(k)}, r\big) = \{x \mid \big\|x - x^{(k)}\big\| < r\}$. The values of $R_p^{(k)}$ and $R_w^{(k)}$ are then specified as

$$R_w^{(k)} = \min\left\{\frac{D}{2}, R^{(k)}\right\}, \qquad R_p^{(k)} = 1.1 R^{(k)} \tag{5}$$

Let $S = \{i_1, i_2, i_3, \ldots, i_{N_p - 1}\}$ be the set of indices corresponding to the $N_p - 1$ points that are closest to $x^{(k)}$, which determine the local least squares approximation $P_k(x)$. The weights satisfy $\omega_{i_j k} > 0$ because $R_p^{(k)}$ is slightly larger than $R^{(k)}$. Define an $(N_p - 1) \times m$ matrix A by

$$A_{j\cdot} = \sqrt{\omega_{i_j k}} \big(x^{(i_j)} - x^{(k)}\big)^t \tag{6}$$

and $(N_p - 1)$-vector b by

$$b_j = \sqrt{\omega_{i_j k}} (f_{i_j} - f_k) \tag{7}$$

The coefficients $a^{(k)} \in E^m$ of $P_k(x)$, where E^m is m-dimensional real Euclidean space, are the minimum norm solution of the least squares problem

$$\min_{a \in E^m} \|Aa - b\|_2.$$

5.2 Multivariate Adaptive Regression Splines (MARS)

Let

$$\mathcal{C} = \big\{1, (x_j - t)_+, (t - x_j)_+ \bigm| t = x_{kj}, \\ 1 \le k \le n,\ 1 \le j \le m\big\}$$

The basis functions $\mathcal{B}$ for MARS are chosen from products of functions in the set $\mathcal{C}$:

$$\mathcal{B} = \mathcal{C} \cup \mathcal{C} \otimes \mathcal{C} \cup \mathcal{C} \otimes \mathcal{C} \otimes \mathcal{C} \cup \cdots.$$

At each iteration the model of the data is a C^0 m-dimensional spline of the form

$$\sum_{h_\alpha \in \mathcal{M}} \beta_\alpha h_\alpha(x) \tag{8}$$

where $\mathcal{M} \subset \mathcal{B}$, $|\mathcal{M}| \le n$, and the coefficients β_α are determined by a least squares fit to the data. $h_\alpha \in \mathcal{M}$ is constrained to always be a spline of order ≤ 2 (piecewise linear) in each variable x_j. The initial model is $\tilde{f}(x) \equiv 1$. Let $\mathcal{M} \subset \mathcal{B}$ be the set of basis functions in the model at iteration q. The basis at iteration $q + 1$ is that basis

$$\mathcal{M} \cup \big\{h_\ell(x)(x_j - t)_+,\ h_\ell(x)(t - x_j)_+\big\} \tag{9}$$

which minimizes the least squares error of the model using that basis over all $h_\ell(x) \in \mathcal{M}$ and $t = x_j^{(k)}$, $1 \le j \le m$, $1 \le k \le n$, subject to the constraint that $h_\ell(x)(x_j - t)_+$ is a spline of order 2 in x_j, and a spline of degree at most n_I in x (where n_I is the most variable interactions permitted). The iteration continues for some given number n_B of iterations or until the data are overfit, at which point the generalized cross-validation criterion

$$\text{GCV}(\lambda) = \frac{\sum_{k=1}^{n} \Big(\tilde{f}_\lambda\big(x^{(k)}\big) - f_k\Big)^2}{(1 - M(\lambda)/n)^2} \tag{10}$$

(where λ is the number of basis functions in the model $\tilde{f}_\lambda$, $M(\lambda)$ is the effective number of parameters in $\tilde{f}_\lambda$), having been computed for each λ, is used to choose the final approximation $\tilde{f}_\lambda(x)$ that minimizes GCV(λ) with $\lambda \le n_B$. This C^0 m-dimensional spline $\tilde{f}_\lambda(x)$ is the multivariate adaptive regression spline (MARS) approximation to the data. The constraints and greedy way $\mathcal{M}$ is constructed mean that $\tilde{f}_\lambda(x)$ is not necessarily the best approximation to the data by a spline of degree n_I generated from $\mathcal{B}$, or by a spline with n_B basis functions from $\mathcal{B}$.

5.3 Piecewise Linear Interpolation via Delaunay Triangulation

A d-dimensional triangulation $T(P)$ of a finite set of points P in $\mathbb{R}^d$ is any set of d-simplices with vertices in P that are disjoint except along their boundaries and whose union is the convex hull of P. Given n distinct data points $P = \{x^{(1)}$, …, $x^{(n)}\}$ and values $f_k = f\big(x^{(k)}\big)$, a piecewise linear interpolant to f, denoted $\tilde{f}_T$, can be defined for any triangulation $T(P)$ as follows.

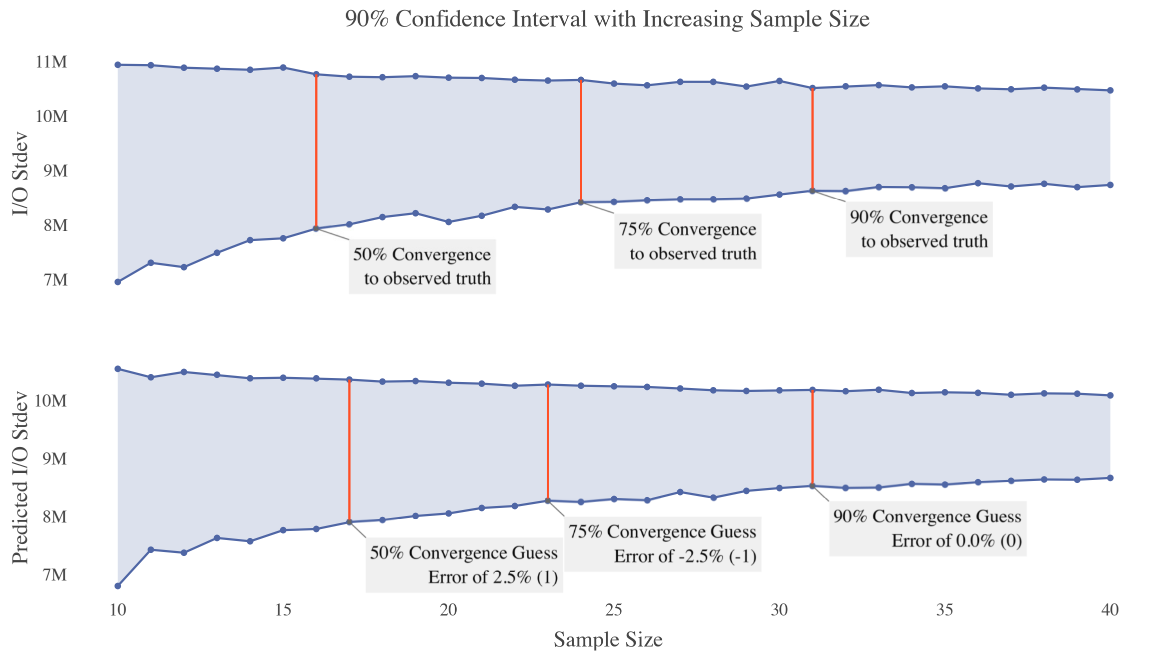


Fig. 10: Measured and predicted 90% confidence intervals for I/O performance variability. Convergence values indicate the proximity of the indicated number of samples to full convergence (at 40 samples). Predicted values are for an unobserved configuration and based upon use of a Delaunay predictor with a $p = 90\%$ training set from the measured results. Errors are calculated relative to the measured confidence interval at each sample size.

At any point x in the convex hull of P, x must be contained in *some* simplex in $T(P)$. Let S be a d-simplex in $T(P)$ with vertices $\{s^{(1)}, \ldots, s^{(d+1)}\}$ such that $x \in S$. Then for $k = 1, \ldots, d+1$, there exist weights $W_k \geq 0$ such that $x = \sum_{k=1}^{d+1} s^{(k)} W_k$ and $\sum_{k=1}^{d+1} W_k = 1$, and

$$\tilde{f}_T(x) = f(s^{(1)})W_1 + \ \ldots \ + f(s^{(d+1)})W_{d+1} \qquad (11)$$

Note that in most cases, any triangulation of P is not unique. The Delaunay triangulation, denoted $DT(P)$, is a (generally unique) triangulation, that has many properties considered optimal for the purpose of interpolation [30]. Therefore, the Delaunay interpolant $\tilde{f}_{DT}$ is often used as an approximation to a multivariate function f.

Discussion. The unknown parameters are derived from the measured data. For example, we can list the coefficients of the MARS bases. However, this closed-form expression is typically too long to practically show in a paper and varies with a set of measurements making it less intuitive. For LSP, the parameters constitute a relatively large matrix (810 * 4 size) that also depends on the measured data and does not present well in a manuscript. Hence, our focus is on presenting the general formulae for these predictors along with the average relative error.

6 Using MOANA for Design of Experiments

Predicting confidence. In Section 4, we used MOANA and two non-linear predictors (LSP and MARS) with promising results for predicting variability accurately. We used these predictions to analyze the variability of configurations not included in the training data set with some success. In this section, we demonstrate another use of variability prediction – to predict the convergence of statistical confidence intervals for unseen system and application configurations.

To further demonstrate the flexibility of MOANA, we use a Delaunay predictor [12]. We evaluated the Delaunay technique for predicting 90% confidence intervals as we did the LSP and MARS techniques in Section 3 and achieved an average relative error of 4% using a $p = 90\%$ training set without loss of generality.

Figure 10 demonstrates how the proposed MOANA convergence estimation works for a sample configuration. The topmost graph shows the I/O standard deviation in measured throughput for a 90% confidence interval. From left to right we observe the change in standard deviation as we increase the number of randomly selected, measured samples from 10 to 40. The vertical lines provide markers to indicate how close the marked sample size is to the "best" (or least) standard deviation at 40 samples. 75% convergence at a sample size of 24 means that with 24 samples we get 75% of the way towards full convergence to the maximum 40 samples. Taking just 7 more samples (for a total of 31) brings us 90% of the way towards full convergence to the maximum 40 samples. Each sample at this configuration has a cost in time (e.g., 10 seconds). Identifying the correlation between samples and convergence enables tradeoff analyses such as: 7 more samples costs 70 seconds but brings us 15% closer to convergence – this in turn informs experimental design space tradeoff decisions of coverage versus time.

The MOANA approach enables projection of design space costs for predicted values as well. Figure 10 demonstrates how the proposed MOANA convergence estimation works for a predicted configuration. The bottommost graph shows the I/O standard deviation in predicted throughput. From left to right we observe the change in standard deviation as we increase the number of randomly selected, measured samples used in the prediction from 10 to 40.

The vertical lines provide markers to indicate how close the marked sample size is to the "best" (or least) predicted standard deviation at 40 samples. 75% convergence at a sample size of 24 means that with 24 samples we get almost 75% of the way towards full convergence to the maximum 40 samples – in this case with an error of less than 3% compared to the measured value obtained during the brute force experiments. The prediction of these values has further implications for design space decisions well beyond the measured data set – projections of the time costs of additional experimental configurations for design space exploration.

Large file sizes. Since file size had the greatest impact on runtime, we limited experiments in the first 6 months of data collection to file sizes smaller than 1MB. However, this potentially ignores behaviors that might be common in enterprise datacenter environments with intense I/O workloads. To address this limitation, we spent almost two months collecting 80 samples for each unique configuration of thread count (1; 2; 3; 4; 5), file size (200MB; 500MB), record size (4KB; 256KB; 16384KB), and frequency (1.2 GHz; 2.1 GHz; 3.0 GHz)[6]. We repeated the aforementioned collection methods on the same physical hardware but due to some internal repurposing and others using the systems in between our experiments, we used Ubuntu 14.04 LTS/KVM Qemu 2.12 instead of Ubuntu 14.04 LTS/XEN 4.0. As a sanity check, we repeated and compared our results to thousands of small file size experiments from the first six months of data. Even at the higher sample rate of the large file experiments, we observed populations that were statistically analogous across the respective old and new data sets despite the change in hypervisors. This confirmed our hypothesis that the new and old data sets exhibited similar behavior and are worth inclusion in this manuscript. However, due to the differences in system software across the datasets, we purposely did not combine the results directly to analyze trends from small to large file sizes.

We limit the analyses of the large file size data to prediction of confidence intervals – the results are very similar to those in Figure 10 and are not included because there are few new conclusions and due to space considerations. Since we started with a larger number of samples, our raw data shows 75% convergence happens at a sample size of 36; this means that with 36 samples we get 75% of the way towards full convergence to the maximum 80 samples. Taking 21 more samples (for a total of 57) brings us 90% of the way towards full convergence to the maximum 80 samples. The I/O standard deviation in predicted throughput shows 75% convergence at a sample size of 35; this means that with 35 samples we get almost 75% of the way towards full convergence to the maximum 80 samples – in this case with an error of less than 2.5% compared to the measured value obtained during the brute force experiments.

There are several takeaways here. First, the MOANA approach works effectively for large file sizes up to 500 MB in our experiments. This is not particularly surprising as the nonlinear analysis and prediction techniques work well for smaller file sizes and are independent of the measured data. Second, the MOANA approach showed that the overall behaviors for large file sizes, despite a change to the hypervisor, were very similar to the overall behaviors for small file sizes. This supports our hypothesis that the black-box, end-to-end analyses using MOANA are useful for a broad array of experiments and the variance is not simply noise but includes predictable behavior to some extent. Third, the behaviors we are observing appear to persist across hypervisor implementations which provides indications that hypervisors are not the root cause of the observed variabilities. This is a somewhat contrary finding as conventional wisdom purports that hypervisors introduce variability. But admittedly, this warrants further investigation in future work.

7 Related Work

The most closely related work to ours is reproducibility in benchmarking since variability plays a role. Hoefler et al. [15] recently summarized the state of the practice for benchmarking in HPC and suggested ways to ensure repeatable results. The main contribution of their work is a series of best practice rules based in existing statistical and mathematical first principles. Ricci et al. [24] also describe best practices for OS systems researchers when encountering variability across heterogeneous nodes in a cluster. Recently, studies of the combined effects of hardware, applications, and operating systems on I/O variability [7] have seen interest due to implications for performance analysis, QoS gurantees, etc. Introducing determinism to achieve reproducibility has also been explored using environment categorization [32], statistical modeling [36], or variance-aware algorithm design [2]. Environment categorization considers the interactions and composition of hidden factors in the system (e.g., DataMill [13]). Madireddy et. al [22] investigated the challenge of I/O variability in HPC environments caused by concurrent activities across the system. They analyzed the correlation of I/O performance of applications and I/O contention [23] and they developed a performance variability model using a machine learning approach on Lustre file systems considering application and system characteristics. Some other research projects [16, 37, 44] explored the issue of I/O variability and proposed models and optimizations for parallel file systems in HPC. Our focus is on predicting variability and the application of MOANA to experimental analysis and design.

OS jitter studies [27] are also related to variability. Jitter in HPC is typically described as performance loss caused by the competition for resources between background processes and applications, or application interference [20, 45]. Some OS jitter researchers have simulated these effects at scale [11], while others have proposed applications [14] or systems [4] that can account for these types of variability. Additionally, schedulers are often identified as causes of significant variability in HPC systems [39] and might be classified as a form of jitter. We specifically studied the effects of schedulers in this work and found the effects of thread count and other variables (e.g., processor speed) to contribute more substantially to variability than scheduler design. As mentioned, we leave isolation of the effects of additional variables, background jitter, and cross-application interference to future work.

6. Due to increases in the per-experiment time, we did not test as many combinations as we did in the first set of experiments.

There have also been a number of high-profile projects in computer architecture that explore variability. These projects are mainly focused on the consequences of on-chip power budgets [3, 6, 17, 31, 43]. These techniques are orthogonally related to MOANA. Since we view variability as an artifact of system design, we capture variations due to all aspects of design including the underlying architecture. In the current study, software artifacts tend to dominate variability though we do see some architectural effects (e.g., voltage and frequency scaling). So, for observable architectural parameterizations, we can capture the effects and in future work potentially expose causality.

Our use of predictors of variability is related to performance prediction. Performance prediction of HPC and distributed applications is a well-studied field and recent works have used analytical [10, 38, 41], profile based [5, 34], and simulation based [8, 28] approaches, or a combination of these [46], to accurately predict overall performance. In contrast, detailed studies like ours that result in models or predictors that consider variability are almost nonexistent. In our work, we explored the use of multiple non-linear predictors for analysis and prediction.

8 Discussion

MOANA is the first comprehensive end-to-end system methodology of its kind and one step towards improving our understanding of I/O system performance variability. The growing impact and prevalence of variability coupled with a deficit in our understanding of the implications on future design make this work particularly timely. We consider variability unavoidable, an artifact of software and hardware designs. MOANA demonstrates that these variability artifacts can be predicted; and that these predictions can be used to revisit the design of our experiments. The use of MOANA to predict variability for classes of applications has the potential for broad impact on experimental system design. However, the current work is not without limitations.

Causality. MOANA is a black-box approach to predicting variability. We measure the execution performance for an application in its entirety and characterize and predict the performance variability. We then use these predictions to determine which configurations see the most variability and quantify their affect on experimental design. Our focus, for now, is not on identifying causality in these experiments. Our focus is on demonstrating that variability is a predictable artifact of system design.

Experimental Breadth. Previously observed performance variability influenced our selection of I/O benchmarks and variables for this study. The time investment in this work is substantial and tradeoffs abound for fidelity (and therefore trustworthiness and repeatability in the results) versus the number and scope of applications and variables studied. Since the research space abounds in studies of the parameterization of systems and applications and is lacking in studies of variability, we opted for gathering results that were limited in application number and scope but robust in their statistical significance. This led to selecting benchmarks that are well-understood and deterministic in nature.

Simplifying Assumptions. We purposely made simplifying assumptions about population distributions to align our techniques with those in the prevailing literature. While our direct measurements and analyses of the populations confirm that distributions at times exhibit normality and at other times do not, our prediction accuracy demonstrates that the effects of this assumption on prediction accuracy are minimal for the regular I/O codes studied. We believe this helps to explain why many past studies have provided significant insights despite assumptions of normality. However, as we demonstrated in our comparisons with linear approaches, it is increasingly likely that such assumptions will lead to inaccuracies that will be unacceptable at exascale and beyond. The MOANA black-box approach, despite the aforementioned assumptions and limitations, is scalable and can be used for accurate interpolation and extrapolation.

9 Conclusions and Future Work

MOANA uses nonlinear statistical techniques to predict the high order effects of high-performance systems and application configurations on I/O variability. We demonstrated that by building variability maps, or correlation vectors between configurations and variance, we can accurately predict variability for hundreds of unseen configurations of systems and applications. The MOANA approach results in a single model for all parameter settings ($>$70% accuracy for MARS) that is an order of magnitude more accurate than best available linear regression techniques. Our analyses showed MOANA can readily identify both first order effects (rank ordering modes by the variability magnitude) and higher order effects (clustering variability by mode for file size and record size studies) through comparison of variability maps. We also demonstrated the use of MOANA for predicting convergence of unmeasured configurations for use in experimental design. Effectively, MOANA enables users to optimize the tradeoffs between system space coverage and time/confidence.

During our work, we have identified a number of avenues for future research. First and foremost, we would like to deepen our understanding of performance variability as an artifact of our software and hardware designs. To this end, we plan to increase the breadth of our studies to include a broader number and type of applications and systems. Additionally, we would like to explore other hardware configurations (accelerators, emergent memory and I/O systems) to consider performance variability with regard to hardware classes of systems and heterogeneity. In another direction, we want to study more deeply the effects of population distribution assumptions (common in the literature) against experimental system research accuracy and the accuracy of our prediction techniques at scale.

References

[1] H. Akkan, M. Lang, and L. M. Liebrock. Stepping towards noiseless linux environment. In *Proceedings of the 2Nd ACM International Workshop on Runtime and Operating Systems for Supercomputers*, 2012.

[2] J.-Y. Audibert, R. Munos, and C. Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[3] A. Bacha and R. Teodorescu. Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors. *ACM SIGARCH Computer Architecture News*, 41(3):297–307, 2013.

[4] P. Beckman, K. Iskra, K. Yoshii, S. Coghlan, and A. Nataraj. Benchmarking the effects of operating system interference on extreme-scale parallel machines. *Cluster Computing*, 11(1):3–16, Mar. 2008.

[5] J. Bourgeois and F. Spies. Performance prediction of an nas benchmark program with chronosmix environment. In *Euro-Par 2000 Parallel Processing*, pages 208–216. Springer, 2000.

[6] K. Bowman, J. W. Tschanz, S.-L. L. Lu, P. Aseron, M. M. Khellah, A. Raychowdhury, B. M. Geuskens, C. Tokunaga, C. B. Wilkerson, T. Karnik, et al. A 45 nm resilient microprocessor core for dynamic variation tolerance. *Solid-State Circuits, IEEE Journal of*, 46(1):194–208, 2011.

[7] Z. Cao, V. Tarasov, H. P. Raman, D. Hildebrand, and E. Zadok. On the performance variation in modern storage stacks. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pages 329–344, Santa Clara, CA, 2017. USENIX Association.

[8] H. Casanova, A. Legrand, and M. Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Computer Modeling and Simulation. IEEE UKSIM 2008. Tenth International Conference on*.

[9] H.-C. Chang, B. Li, G. Back, A. R. Butt, and K. W. Cameron. Luc: Limiting the unintended consequences of power scaling on parallel transaction-oriented workloads. In *IEEE IPDPS*, 2015.

[10] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. Von Eicken. *LogP: Towards a realistic model of parallel computation*, volume 28. ACM Sigplan Notices, 1993.

[11] P. De and V. Mann. jitsim: A simulator for predicting scalability of parallel applications in presence of os jitter. In *Euro-Par 2010 - Parallel Processing*, volume 6271 of *Lecture Notes in Computer Science*, pages 117–130. Springer Berlin Heidelberg, 2010.

[12] J. A. De Loera, J. Rambau, and F. Santos. *Triangulations Structures for algorithms and applications*. Springer, 2010.

[13] A. B. de Oliveira, J.-C. Petkovich, T. Reidemeister, and S. Fischmeister. Datamill: Rigorous performance evaluation made easy. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, 2013.

[14] A. Hammouda, A. R. Siegel, and S. F. Siegel. Noise-tolerant explicit stencil computations for nonuniform process execution rates. *ACM Trans. Parallel Comput.*, 2(1):7:1–7:33, Apr. 2015.

[15] T. Hoefler and R. Belli. Scientific benchmarking of parallel computing systems. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015.

[16] E. C. Inacio, P. A. Barbetta, and M. A. Dantas. A statistical analysis of the performance variability of read/write operations on parallel file systems. *Procedia Computer Science*, 108:2393 – 2397, 2017. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[17] Y. Kim, L. K. John, S. Pant, S. Manne, M. Schulte, W. L. Bircher, and M. S. S. Govindan. Audit: Stress testing the automatic way. In *Microarchitecture (MICRO), 45th Annual IEEE/ACM International Symposium on*, 2012.

[18] W. T. Kramer and C. Ryan. *Performance variability of highly parallel architectures*. Springer, 2003.

[19] D. Le, H. Huang, and H. Wang. Understanding performance implications of nested file systems in a virtualized environment. In *USENIX FAST*, 2012.

[20] 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 *High Performance Computing, Networking, Storage and Analysis (SC), 2010 IEEE International Conference for*, 2010.

[21] R. Lucas, J. Ang, S. Borkar, W. Carlson, L. Carrington, G. Chiu, R. Colwell, W. Dally, J. Dongarra, A. Geist, G. Grider, R. Haring, J. Hittinger, A. Hoisie, D. Klein, P. Kogge, R. Lethin, V. Sarkar, R. Schreiber, J. Shalf, T. Sterling, and R. Stevens. Ascac subcommittee for the top ten exascale research challenges. 2014.

[22] 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 *2017 International Conference on Networking, Architecture, and Storage (NAS)*, pages 1–10, Aug 2017.

[23] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. M. Wild. Machine learning based parallel i/o predictive modeling: A case study on lustre file systems. In *International Conference on High Performance Computing*, pages 184–204. Springer, 2018.

[24] 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, Carlsbad, CA, 2018. USENIX Association.

[25] R. Mraz. Reducing the variance of point to point transfers in the ibm 9076 parallel computer. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, 1994.

[26] J. Ouyang, B. Kocoloski, J. R. Lange, and K. Pedretti. Achieving performance isolation with lightweight co-kernels. In *ACM HPDC*, 2015.

[27] F. Petrini, D. J. Kerbyson, and S. Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q. In *ACM ICS*, 2003.

[28] S. Prakash and R. L. Bagrodia. Mpi-sim: using parallel simulation to evaluate mpi programs. In *Proceedings of the 30th conference on Winter simulation*. IEEE Computer Society Press, 1998.

[29] A. Rahimi, D. Cesarini, A. Marongiu, R. Gupta, and L. Benini. Task scheduling strategies to mitigate hardware variability in embedded shared memory clusters. In *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015.

[30] V. Rajan. Optimality of the delaunay triangulation in $\mathbb{R}^d$. *Discrete & Computational Geometry*, 12(2):189–202, 1994.

[31] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G.-Y. Wei, and D. Brooks. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *IEEE/ACM MICRO*, 2010.

[32] R. Ricci, G. Wong, L. Stoller, K. Webb, J. Duerig, K. Downie, and M. Hibler. Apt: A platform for repeatable research in computer science. *ACM SIGOPS Operating Systems Review*, 49(1):100–107, 2015.

[33] A. Rutherford. Introducing anova and ancova: A glm approach.

[34] R. H. Saavedra and A. J. Smith. Analysis of benchmark characteristics and benchmark performance prediction. *ACM Transactions on Computer Systems (TOCS)*, 14(4):344–384, 1996.

[35] J. Shalf, S. Dosanjh, and J. Morrison. Exascale computing technology challenges. In *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, 2011.

[36] D. Skinner and W. Kramer. Understanding the causes of performance variability in hpc workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, 2005.

[37] S. W. Son, S. Sehrish, W.-K. Liao, R. Oldfield, and A. Choudhary. Reducing i/o variability using dynamic i/o

path characterization in petascale storage systems. *Journal of Supercomputing.*, 73(5):2069–2097, May 2017.

[38] D. Sundaram-Stukel and M. K. Vernon. Predictive analysis of a wavefront application using loggp. *ACM SIGPLAN Notices*, 34(8):141–150, 1999.

[39] V. Tabatabaee, A. Tiwari, and J. K. Hollingsworth. Parallel parameter tuning for applications with performance variability. In *ACM/IEEE SC*, 2005.

[40] W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry. Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Trans. Math. Softw.*, 37(3):34:1–34:20, Sept. 2010.

[41] A. J. Van Gemund. Symbolic performance modeling of parallel systems. *Parallel and Distributed Systems, IEEE Transactions on*, 14(2):154–165, 2003.

[42] S. O. F. Wang, D. A. Dillow, R. Miller, G. M. Shipman, D. Maxwell, and D. H. J. B. J. Larkin. Reducing application runtime variability on jaguar xt5. 2010.

[43] P. N. Whatmough, S. Das, Z. Hadjilambrou, and D. M. Bull. 14.6 an all-digital power-delivery monitor for analysis of a 28nm dual-core arm cortex-a57 cluster. In *IEEE International Solid-State Circuits Conference-(ISSCC), 2015*.

[44] 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*, HPDC '17, pages 181–192, New York, NY, USA, 2017. ACM.

[45] 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 *IEEE International Parallel and Distributed Processing Symposium*, 2016.

[46] J. Zhai, W. Chen, and W. Zheng. Phantom: predicting performance of parallel applications on large-scale parallel machines using a single node. In *ACM Sigplan Notices*, volume 45, pages 305–314, 2010.

Kirk W. Cameron is Professor of Computer Science at Virginia Tech and Director of the stack@cs Center for Computer Systems. The central theme of his research is to improve performance and power efficiency in computer systems. Accolades for his research include U.S. National Science Foundation and U.S. Department of Energy Career Awards, IBM and AMD Faculty Awards, and a Distinguished Visiting Fellowship from the U.K. Royal Academy of Engineering. He received the PhD degree in computer science from Louisiana State University and is a member of the IEEE and a Distinguished Member of the ACM.

Ali Anwar is a research staff member at IBM Almaden Research Center. He received his Ph.D. degree in Computer Science from Virginia Tech. In his earlier years he worked as a tools developer (GNU GDB) at Mentor Graphics. Ali's research interests are in distributed computing systems, cloud storage management, file and storage systems, Internet of Things, AI platforms, and intersection of systems and machine learning.

Yue Cheng received his Ph.D. degree in Computer Science (2017) from Virginia Tech. He is an assistant professor of Computer Science at George Mason University. His research interests include distributed systems, cloud and serverless computing, high performance computing, and the Internet of Things. At George Mason he leads the Experimental Scalable & Efficient Systems Laboratory (EXCEL).

Li Xu is a PhD student in statistics at Virginia Tech. His research interests include machine learning and engineering applications, reliability analysis, and spatial statistics.

Bo Li received his BS degree in automation from Dalian University of Technology in 2006, MS degree in control theory & control engineering from Dalian University of Technology in 2009, and MS degree in computer science from Rochester Institute of Technology in 2012. He is currently working toward the PhD degree at the Virginia Polytechnic Institute and State University. His research interests include power-aware computing in the high-performance computing domain and performance modeling of scientific parallel applications under DVFS, DCT, and DMT.

Uday Ananth received his M.S degree in Computer Science from Virginia Polytechnic Institute and State University in 2017. He is a currently a Member of Technical Staff at Viasat Inc and his interests include cloud computing / orchestration, firmware development and network layer optimizations.

Jon Bernard is a PhD student in computer science at Virginia Tech. His research interests include operating systems, performance variability, and security.

Chandler Jearls is an undergraduate in Computer Engineering at Virginia Polytechnic Institute and State University. His research interests include parallel computing, distributed computing systems and computer architecture.

Thomas Lux received a B.S. degree (cum laude) in Computer Science with minors in Mathematics and Physics from Roanoke College in 2016. He is a Ph.D. candidate in Computer Science at Virginia Polytechnic Institute and State University. His research interests include mathematical modeling, optimization, numerical analysis, and reinforcement learning for artificial intelligence.

Yili Hong received his PhD in statistics (2009) from Iowa State University. He is an associate professor of statistics at Virginia Tech. His research interests include machine learning and engineering applications, reliability analysis, and spatial statistics. He has over 60 publications in venues such as Journal of the American Statistical Association, Annals of Applied Statistics, Technometrics, and IEEE Transactions on Reliability. He is currently an associate editor for Technometrics and Journal of Quality Technology. He is an elected member of International Statistical Institute. He won the 2011 DuPont Young Professor Award, and the 2016 Frank Wilcoxon Prize in statistics.

Layne T. Watson (F '93) received the B.A. degree (magna cum laude) in psychology and mathematics from the University of Evansville, Indiana, in 1969, and the Ph.D. degree in mathematics from the University of Michigan, Ann Arbor, in 1974. He is currently a professor of computer science, mathematics, and aerospace and ocean engineering at Virginia Polytechnic Institute and State University. He serves as senior editor of Applied Mathematics and Computation, and associate editor of Computational Optimization and Applications, Evolutionary Optimization, Engineering Computations, and the International Journal of High Performance Computing Applications. He is a fellow of the National Institute of Aerospace and the International Society of Intelligent Biological Medicine. He has published well over 300 refereed journal articles and 200 refereed conference papers. His research interests include fluid dynamics, solid mechanics, numerical analysis, optimization, parallel computation, mathematical software, image processing, and bioinformatics.

Ali R. Butt received his Ph.D. degree in Electrical and Computer Engineering from Purdue University. He is a professor of Computer Science and ECE (by courtesy) at Virginia Tech. He is a recipient of several awards such as an NSF CAREER Award. He is an alumni of the National Academy of Engineering's US Frontiers of Engineering (FOE) Symposium, US-Japan FOE, and National Academy of Science's AA Symposium on Sensor Science. Ali's research interests are in distributed computing systems, cloud/edge computing, file and storage systems, Internet of Things, I/O systems, and operating systems. At Virginia Tech he leads the Distributed Systems & Storage Laboratory (DSSL).