# 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, Thomas Lux, Yili Hong, Layne T. Watson, Ali R. Butt**

*Virginia Polytechnic Institute and State University*, †*George Mason University*

April 19, 2018

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060

**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. We create 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.

*This work has been submitted to IEEE Transactions on Parallel and Distributed Systems (TPDS).

**Index Terms:** Computer Systems, High Performance Computing, Parallel and Distributed Computing
**Keywords:** System variability, high performance computing, I/O systems.

| | 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. 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.

# 1 Introduction

mergent 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 [29, 18]. Unfortunately, variability (which includes OS jitter [21]) is both ubiquitous and elusive as its causes pervade and obscure performance across the systems stack from hardware [14, 23] to middleware [20, 1] to applications [12] to extreme-scale systems [29, 35]. Additionally, as a number of recent reports attest [29, 18], performance variability at scale can significantly reduce performance and energy efficiency [9, 4].

Variability is a persistent challenge in experimental systems research. Table 1 lists the 7 parameters in our HPC I/O variability study. 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 6 months in total with nearly uninterrupted runs and 6 dedicated systems.

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 [19, 15], variability is regularly discounted or ignored by the community. While there are notable exceptions (e.g., OS jitter [21], application interference [37, 17], reproducibility [32], scheduling [13]), 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

[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.

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;
- 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 properties 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 and 8 respectively discuss our work in the context of existing literature and limitations and future work.

[2] URL redacted pending paper review.

# 2 Experimental Variability Analysis

## 2.1 Experimental Setup

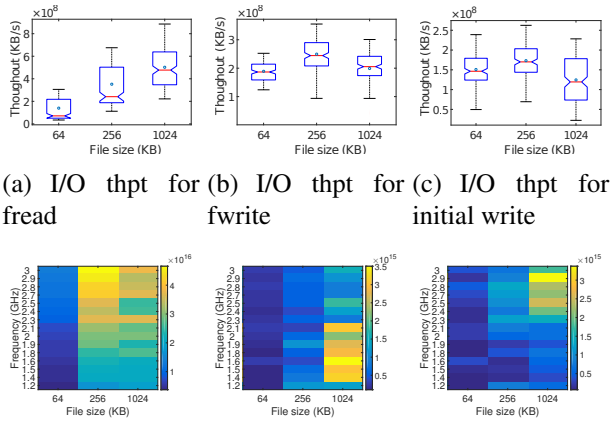
We focus on intra-node variability effects, identified as a key barrier to exascale [21, 29, 18], and perform our experiments on parallel shared-memory nodes common to HPC systems. Table 1 summarizes the parameter space for all experiments performed on a lone guest Linux operating system (Ubuntu 14.04 LTS//XEN 4.0) 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. 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 but we are conducting presently. These findings would not impact the MOANA techniques but could help to improve their accuracy.

## 2.2 Empirical Analysis

**Effect of file size** Figure 1 shows experiments designed to examine the effects of file size on I/O variability. Figure 1(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 1(b)) and file initial write operations (Figure 1(c)), the I/O throughput initially increases but eventually decreases. Here, first most writes are absorbed in the cache, but as the cache becomes full, the disk flush operations come into play and reduce overall throughput. Figure 1(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. As these interactions are the effect of CPU frequency and scheduling decisions, the observed variance increases. For file write operations, variance is higher for larger sized files at lower CPU frequency as shown in Figure 1(e). Figure 1(f) shows that for initial write operations, larger file sizes with higher frequency exhibit the most variance. This is potentially due to the need for more disk flushes, and the higher CPU frequencies triggering faster scheduling decisions, resulting in increased non-deterministic interactions between the I/O sub-components, which in turn can increase measure variance.

**Effect of record size** Figure 2 shows experiments designed to examine the effects of record size on I/O variability. Figure 2(a) shows raw I/O throughput decreases with increases in record size for file




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



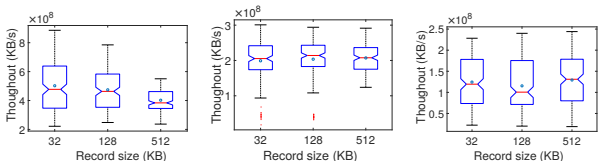

(d) I/O var for fread (e) I/O var for fwrite (f) I/O var for initial write

Figure 1: 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.

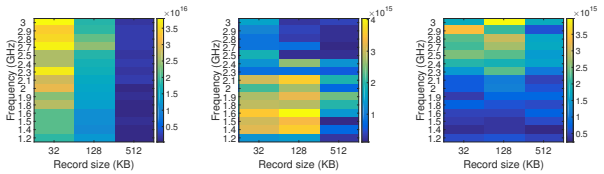
read operations. For file write operations (Figure 2(b)) and file initial write operations (Figure 2(c)), the I/O throughput remains unchanged. I/O variance results vary extensively. Smaller record size and higher CPU frequency give the highest variation for file read operations. Medium record size and 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 2(d), 2(e), and 2(f), respectively. Once again, the interaction of different record sizes and the I/O sub-system yield complex non-deterministic interactions that are not straight-forward to explain via understanding of system-level implementation details only. Consequently, there is a need for designing better approaches to capturing, modeling, and mitigating variance in such systems.

**Effect of number of threads** Figure 3 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 3(a), 3(b), and 3(c). I/O throughput variance also increases with the number of threads: highest for file read operations (Figure 3(d)) and file write operations (Figure 3(e)) at the highest frequency. File read operations have high variance in the lower frequency range as well (Figure 3(d)).

**Variability Trends** 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 CPU frequency variations. Figure 4(c) shows that variability when changing file sizes is not particularly sensitive to CPU frequency variations (i.e., only the highest frequency seems to matter). For a system where file size and CPU frequency variations are common (e.g., most shared-memory HPC systems), there




(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

Figure 2: 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.

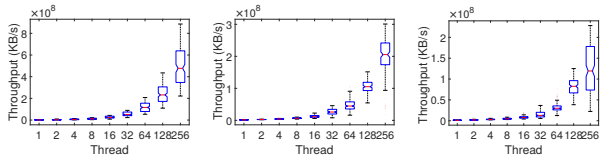
is no clear optimal operating configurations. This points to the challenge identified above that understanding the runtime interactions requires more than just the system-level implementation details, mainly due to the many dynamic interactions between various sub-systems.

To illustrate further, 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 Kbyte file size. All of the subfigures on the right of Figure 5 (b, d, and f) use 3.0 GHz and 1024 Kbyte file size. There is a stark contrast between these two columns of subfigures. There seems no discernible pattern to the resulting change in the variance and no clear optimal operating configuration.

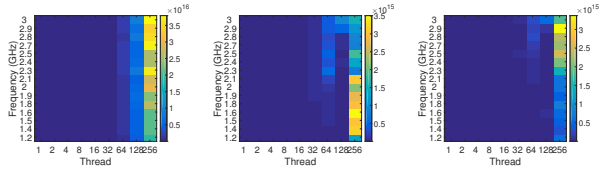
**Limitations** In these examples, we are only considering a few hundred permutations of I/O modes, CPU frequency, thread count, file size, and record size from among over 95K. This limits this type of analysis to a very small subset of the experimental data set. Analyses of the combined effects of multiple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

# 3 MOANA Methodology

We used statistical analysis of variance (ANOVA) [27] experiments to identify first-order (one-parameter changes) and second-order (two-parameter changes) effects of performance variability. We omit the full results due to space limitations, but in summary, this ANOVA experiment showed




(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

Figure 3: 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.

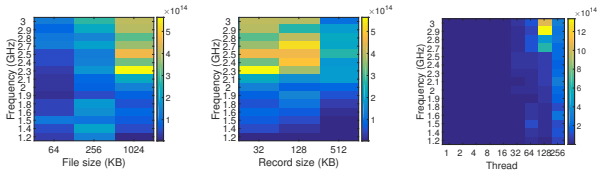
that nearly all of the parameters studied (and their second order configurations, e.g., Filesize x Thread) affect I/O variability in a statistically significant way. Unfortunately, this linear method does not expose the relative magnitude of the variability contribution for a given variable. Table 2 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.

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. We provide a detailed mathematical description of the LSP, MARS, and delaunay triangulation methods in Section 5. For now, we provide an overview of MOANA for use in our variability analyses discussed in Section 4.

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

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

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




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

Figure 4: Heat map of change ((a) and (b)) in I/O throughput variance ($y$-axis-right) from 256 threads (Figure 1(e) and Figure 2(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 3(d)) down to 64 KBytes. Results for fwrite are shown.

The LSP and MARS 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 or MARS algorithms described in Section 5.

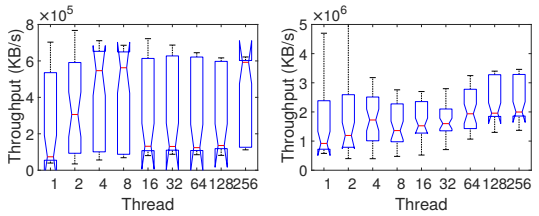
**Predictor evaluation** We define the following relative error as an evaluation criterion. That is

$$r = \frac{|\tilde{f} - f|}{f},$$

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.

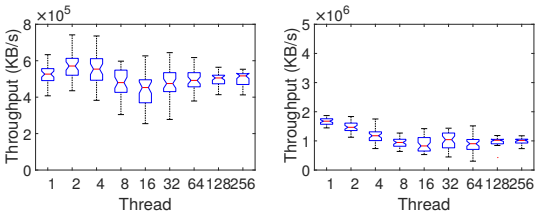
**Predictor comparisons** By varying $p$ we can observe the tradeoffs between predictor accuracy and the ratio of the training set to the predicted data points. Table 2 shows the ARE for linear regression (LR), LSP, MARS, and delaunay triangulation as functions of the training sample proportion $p$,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




(a) I/O thpt for fread



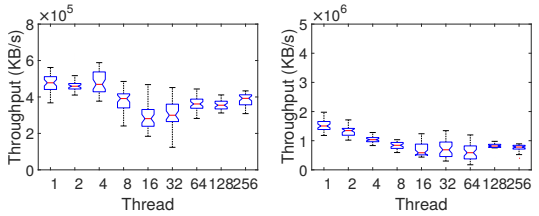

(b) I/O thpt for fread




(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

Figure 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 Kbyte file size. All of the subfigures on the right ((b), (d), and (f)) use 3.0 GHz and 1024 Kbyte file size. Other fixed parameters: host scheduler = CFQ, VM I/O scheduler = NOOP, record size = 32 KBytes.

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).

# 4 MOANA Variability Analysis

In this section we attempt to predict 585 configurations not considered in the full, 95K-configuration training set described in Section 2.1. We calculate the average relative error (ARE) as discussed in

| 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.

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 we evaluate both LSP and MARs – though in a later section we evaluate a 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 6) 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 6 – 8 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: New configurations used for model validation.

**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 well generally. Figure 6 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 6) are generally clustered closer to the diagonal compared to the LSP data points (circle point type in Figure 6). This is confirmed with average relative error (ARE) calculations: MARS has a 29.75% ARE while LSP has 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

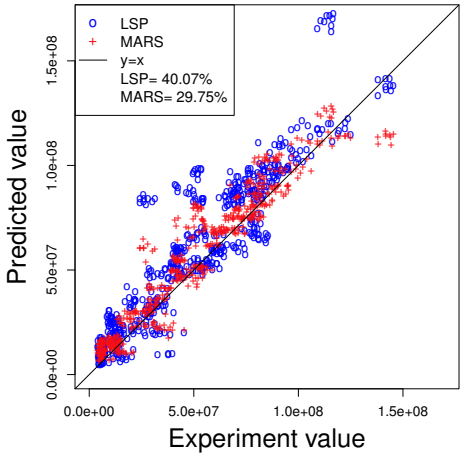



Figure 6: Comparison of the two proposed prediction models on new configurations. The $y$-axis shows the predicted standard deviation from our two models, while the $x$-axis shows the empirical standard deviation from 40 runs. The ARE for each method is also shown on the legend.

Table 2) but consistently much 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 7 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 6). 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 opera-

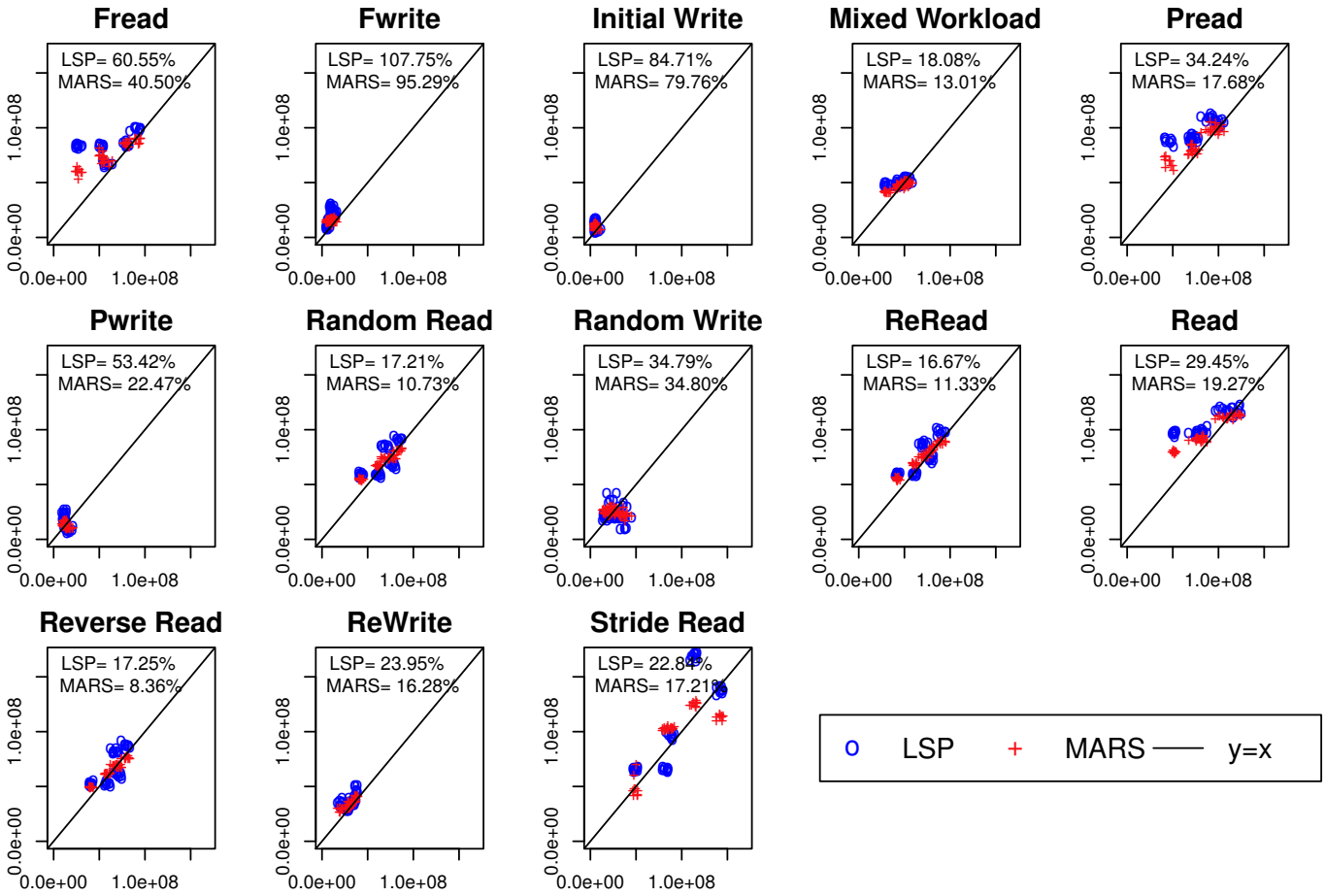



Figure 7: Prediction scatter plots of different I/O operation modes. Points in the same plot have the same mode. The $x$-axis is the empirical standard deviation observed from 40 runs. The $y$-axis is the predicted standard deviation obtained from our two models. The ARE for each method is also shown on the legend.

tions 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 [16]. 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 8 plots the prediction results for valid file and record size combinations from Table 3. In all but 1 case, the LSP or MARS

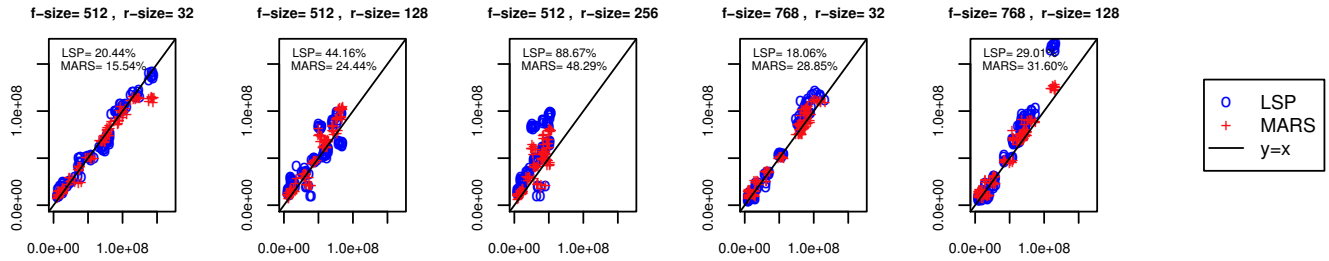



Figure 8: Prediction scatter plots of different file and record size combinations. Points in the same plot have other parameters fixed. The $x$-axis is the empirical standard deviation observed from 40 runs. The $y$-axis is the predicted standard deviation obtained from our two models. The ARE for each method is also shown on the legend.

predictions beat the average ARE values for the general model from Use Case I (MARS=29.75%, LSP=40.07% in Figure 6). 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 8 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

For completeness, we provide details for LSP (modified linear Shepard algorithm), MARS (multivariate adaptive regression splines), and Delaunay triangulation approximation methods. Recall in Section 3 we 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 algorithm, we tailored the existing FORTRAN code and linked it to R.

## 5.1 Modified Linear Shepard Algorithm

The linear Shepard algorithm is derived from the modified Shepard algorithm [33]. Given $n$ distinct data points $x^{(1)}, \ldots, x^{(n)}$ and values $f_k = f\left(x^{(k)}\right)$, the linear Shepard algorithm constructs an

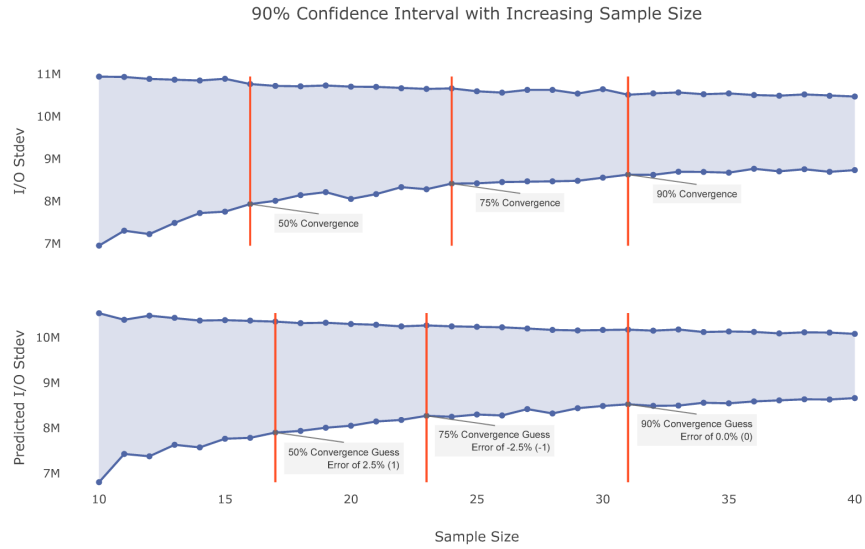



Figure 9: Measured and predicted 90% confidence intervals for I/O performance variability. Convergence values indicate the proximity of the indicated number of samples to full convergence (at 40 samples). Predicted values are for an unobserved configuration and based upon use of a Delaunay predictor with a $p = 90\%$ training set from the measured results. Errors are calculated relative to the measured confidence interval at each sample size.

interpolant to $f$ of the form

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

where the locally supported weights are

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

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

$$P_k(x) = f_k + \sum_{j=1}^{m} a_j^{(k)} \left( x_j - x_j^{(k)} \right).$$

The function $P_k(x)$ is the local linear weighted least squares fit around $x^{(k)}$ with the $i$th data

point having weight

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

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

$$R^{(k)} = \min\Big\{ r \;\Big|\; \overline{B\big(x^{(k)}, r\big)} \text{ contains at least } N_p \text{ points} \Big\},$$

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

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

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

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

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

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

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

$$\mathcal{C} = \Big\{ 1, (x_j - t)_+, (t - x_j)_+ \;\Big|\; t = x_{kj}, \\ 1 \leq k \leq n,\ 1 \leq j \leq m \Big\}$$

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

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

where $\mathcal{M} \subset \mathcal{B}$, $|\mathcal{M}| \leq n$, and the coefficients $\beta_\alpha$ are determined by a least squares fit to the data. $h_\alpha \in \mathcal{M}$ is constrained to always be a spline of order $\leq 2$ (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

$$\mathcal{M} \cup \left\{ h_\ell(x)(x_j - t)_+,\ h_\ell(x)(t - x_j)_+ \right\}$$

which minimizes the least squares error of the model using that basis over all $h_\ell(x) \in \mathcal{M}$ and $t = x_j^{(k)}$, $1 \leq j \leq m$, $1 \leq k \leq n$, 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

$$\text{GCV}(\lambda) = \frac{\sum_{k=1}^{n} \left( \tilde{f}_\lambda\left(x^{(k)}\right) - f_k \right)^2}{(1 - M(\lambda)/n)^2},$$

(where $\lambda$ 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 $\lambda$, is used to choose the final approximation $\tilde{f}_\lambda(x)$ that minimizes GCV($\lambda$) with $\lambda \leq n_B$. 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 $P = \{x^{(1)}, \ldots, x^{(n)}\}$ and values $f_k = f\left(x^{(k)}\right)$, a piecewise linear interpolant to $f$, denoted $\tilde{f}_T$, can be defined for any triangulation $T(P)$ as follows.

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

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

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 interpolation [24]. 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 MOANA Experimental Design

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 [10]. 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 9 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 9 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.

# 7 Related Work

The most closely related work to ours is reproducibility in benchmarking since variability plays a role. Hoefler et al. [13] 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. Introducing determinism to achieve reproducibility has also been explored using environment categorization [26], statistical modeling [30], or variance-aware algorithm design [2]. Environment categorization considers the interactions and composition of hidden factors in the system (e.g., DataMill [11]). Our focus is on predicting variability and the application of MOANA to experimental analysis and design.

OS jitter studies [21] 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 [37, 17]. Some OS jitter researchers have simulated these effects at scale [9], while others have proposed applications [12] or systems [4] that can account for these types of variability. Additionally, schedulers are often identified as causes of significant variability in HPC systems [32] 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 scheduler design. As mentioned, we leave isolation of the effects of additional variables, background jitter, and cross-application interference to future work.

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 [6, 14, 3, 36, 25]. 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 [8, 31, 34], profile based [5, 28], and simulation based [7, 22] approaches, or a combination of these [38], 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 Conclusions and Future Work

MOANA uses nonlinear statistical techniques to capture the high order effects of systems and application configurations on HPC I/O variability. We showed that by building variability maps, or correlation vectors between configurations and variance, we can accurately predict variability for hundreds of unseen configurations. We demonstrated that we can use MOANA to create 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.

MOANA is one step towards improving our understanding of HPC system variability. Our current study is limited to 95K configurations and highly-parallel shared-memory systems, which form the building blocks of high-performance systems and supercomputers. To align with current state of the art practices, we made simplifying assumptions about population distributions that we will reconsider in future work. In this instance, for smaller parameterizations, we find the accuracy of our scalable black-box MOANA approach reasonable. This could also explain why false conclusions relying on such assumptions in the prevailing literature are incorrupt. However, as we demonstrated in our linear variability studies, it is increasingly likely that such assumptions will lead to inaccuracies that will be unacceptable at exascale. The MOANA black-box approach works effectively for the scale and systems herein and is inherently scalable since upon training it can be used for accurate interpolation and extrapolation. We are also considering alternatives to our exponential training strategy potentially in favor of a continuous variable training strategy that is more linear. This involves direct tradeoffs between parameterization for systems boundaries versus the expected inputs to most statistical models. We are also investigating the use of clustering analysis of variability to improve identification of causality and higher order effects. In this work, we purposely ignored interconnect effects as they have been considered substantially in isolation by the networking community. However, the MOANA approach could be extended readily to encompass networking effects and is another subject of future work.

# References


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

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

[3] Anys Bacha and Radu 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] Pete Beckman, Kamil Iskra, Kazutomo Yoshii, Susan Coghlan, and Aroon Nataraj. Benchmarking the effects of operating system interference on extreme-scale parallel machines. *Cluster Computing*, 11(1):3–16, March 2008.

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

[6] Keith Bowman, James W Tschanz, Shih-Lien L Lu, Paolo Aseron, Muhammad M Khellah, Arijit Raychowdhury, Bibiche M Geuskens, Carlos Tokunaga, Chris B Wilkerson, Tanay 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] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Computer Modeling and Simulation. IEEE UKSIM 2008. Tenth International Conference on*.

[8] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. *LogP: Towards a realistic model of parallel computation*, volume 28. ACM Sigplan Notices, 1993.

[9] Pradipta De and Vijay 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.

[10] Jesús A De Loera, Jörg Rambau, and Francisco Santos. *Triangulations Structures for algorithms and applications*. Springer, 2010.

[11] Augusto Born de Oliveira, Jean-Christophe Petkovich, Thomas Reidemeister, and Sebastian Fischmeister. Datamill: Rigorous performance evaluation made easy. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, 2013.

[12] Adam Hammouda, Andrew R. Siegel, and Stephen F. Siegel. Noise-tolerant explicit stencil computations for nonuniform process execution rates. *ACM Trans. Parallel Comput.*, 2(1):7:1–7:33, April 2015.

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

[14] Youngtaek Kim, Lizy Kurian John, Sanjay Pant, Srilatha Manne, Michael Schulte, W Lloyd Bircher, and Madhu Saravana Sibi Govindan. Audit: Stress testing the automatic way. In *Microarchitecture (MICRO), 45th Annual IEEE/ACM International Symposium on*, 2012.

[15] William TC Kramer and Clint Ryan. *Performance variability of highly parallel architectures*. Springer, 2003.

[16] Duy Le, Hai Huang, and Haining Wang. Understanding performance implications of nested file systems in a virtualized environment. In *USENIX FAST*, 2012.

[17] Jay Lofstead, Fang Zheng, Qing Liu, Scott Klasky, Ron Oldfield, Todd Kordenbrock, Karsten Schwan, and Matthew 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.

[18] Robert Lucas, James Ang, Shekhar Borkar, William Carlson, Laura Carrington, George Chiu, Robert Colwell, William Dally, Jack Dongarra, Al Geist, Gary Grider, Rud Haring, Jeffrey Hittinger, Adolfy Hoisie, Dean Klein, Peter Kogge, Richard Lethin, Vivek Sarkar, Robert Schreiber, John Shalf, Thomas Sterling, and Rick Stevens. Ascac subcommittee for the top ten exascale research challenges. 2014.

[19] Ronald 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.

[20] Jiannan Ouyang, Brian Kocoloski, John R. Lange, and Kevin Pedretti. Achieving performance isolation with lightweight co-kernels. In *ACM HPDC*, 2015.

[21] Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q. In *ACM ICS*, 2003.

[22] Sundeep Prakash and Rajive 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.

[23] A. Rahimi, D. Cesarini, A. Marongiu, R.K. 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.

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

[25] Vijay Janapa Reddi, Svilen Kanev, Wonyoung Kim, Simone Campanoni, Michael D Smith, Gu-Yeon Wei, and David Brooks. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *IEEE/ACM MICRO*, 2010.

[26] Robert Ricci, Gary Wong, Leigh Stoller, Kirk Webb, Jonathon Duerig, Keith Downie, and Mike Hibler. Apt: A platform for repeatable research in computer science. *ACM SIGOPS Operating Systems Review*, 49(1):100–107, 2015.

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

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

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

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

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

[32] Vahid Tabatabaee, Ananta Tiwari, and Jeffrey K Hollingsworth. Parallel parameter tuning for applications with performance variability. In *ACM/IEEE SC*, 2005.

[33] William I. Thacker, Jingwei Zhang, Layne T. Watson, Jeffrey B. Birch, Manjula A. Iyer, and Michael W. Berry. Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Trans. Math. Softw.*, 37(3):34:1–34:20, September 2010.

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

[35] Sarp Oral Feiyi Wang, David A Dillow, Ross Miller, Galen M Shipman, Don Maxwell, and Dave Henseler Jeff Becklehimer Jeff Larkin. Reducing application runtime variability on jaguar xt5. 2010.

[36] Paul N Whatmough, Shidhartha Das, Zacharias Hadjilambrou, and David 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*.

[37] Orcun Yildiz, Matthieu Dorier, Shadi Ibrahim, Rob Ross, and Gabriel Antoniu. On the root causes of cross-application i/o interference in hpc storage systems. In *IEEE International Parallel and Distributed Processing Symposium*, 2016.

[38] Jidong Zhai, Wenguang Chen, and Weimin 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.