Nonparametric Distribution Models for Predicting and Managing Computational Performance Variability*

Thomas C. H. Lux[1], Layne T. Watson[2], Jon Bernard[1], Tyler H. Chang[1], Bo Li[1], Xiaodong Yu[1], Li Xu[3], Godmar Back[2], Ali R. Butt[2], Kirk W. Cameron[2], Yili Hong[4], Danfeng Yao[2]

***Abstract*— Performance variability can have a significant impact on many applications of computing. Cloud computing, high performance computing, and computer security communities each exert considerable effort managing and analyzing variability throughout the system stack. This work presents and evaluates a methodology for predicting precise characteristics of the computational performance variability of an input/output (I/O) application over varying system configurations. Results demonstrate that the presented methodology is capable of precisely modeling performance variability, which could allow applications that tighten service level agreements, maximize computational throughput, and obfuscate system configurations against malicious users.**

I. INTRODUCTION

Computational variability presents itself in a variety of forms. Processes that are apparently identical in a cloud computing or high performance computing (HPC) environment may take different amounts of time to complete the same job. This variability can cause unintentional violations of service level agreements in cloud computing applications or indicate suboptimal performance in HPC applications. The sources of variability, however, are distributed throughout the system stack and often difficult to identify. The methodology presented in this work is applicable to modeling the expected variability of useful computer system performance metrics without any prior knowledge of system architecture. Some examples of interesting performance metrics that could be modeled with the techniques in this work include computational throughput, power consumption, processor idle time, number of context switches, and RAM usage, as well as any other ordinal performance metric.

Predicting performance variability in a computer system is a challenging problem that has primarily been attempted in one of two ways: (1) build a statistical model of the performance data collected by running experiments on the system at select settings, or (2) run artificial experiments using a simplified simulation of the target system to estimate architecture and application bottlenecks. In this paper, the proposed modeling techniques rest in the first category and represent a notable increase in the ability to model precise characteristics of variability.

Many previous works attempting to model system performance have used simulated environments to estimate the performance of a system [1], [2], [3]. Some refer to statistical models as being oversimplified and not capable of capturing the true complexity of the underlying system. This claim is partially correct, noting that a large portion of predictive statistical models rely on simplifying the machine to one or two parameters [4], [5], [6], [7]. These limited statistical models have provided satisfactory performance in very narrow application settings. Many of the aforementioned statistical modeling techniques claim to generalize, while simultaneously requiring additional code annotations, hardware abstractions, or additional application level understandings in order to generate models. The approach presented here requires no modifications of the application, no architectural abstractions, nor any structural descriptions of the input data being modeled. The techniques used are purely mathematical and only need performance data as input.

Among the statistical models presented in prior works, [5] specifically mentions that it is difficult for the simplified models to capture variability introduced by I/O. System variability in general has become a problem of increasing interest to the cloud and HPC systems communities, however most existing work has focused on operating system (OS) induced variability [8], [9]. The work that has focused on managing I/O variability does not use any sophisticated modeling techniques [10].

This paper presents an application of advanced statistical modeling to the domain of computer system I/O throughput. The models are used to predict the cumulative distribution function (CDF) of the expected I/O throughput for a system at previously unseen configurations. The techniques in this paper can tractably model tens of interacting system parameters with tens of thousands of unique configurations.

Section II details the mathematical techniques used to approximate variability, the chosen measurement of error, and an optimization strategy for improving model performance. Section III summarizes the data collection process as well as provides summary statistics of the data used for the case study. Section IV presents the results of the I/O case study including analyses of approximation accuracy, model convergence, and system parameter importance. Section V discusses possible explanations for witnessed results, potential applications of the predictive methodology, and future improvements of the models.

*This work was supported by the National Science Foundation Grant CNS-1565314.

[1]Doctoral student, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060, USA (tchlux at vt.edu)

[2]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060, USA

[3]Doctoral student, Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060, USA

[4]Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24060, USA

II. VARIABILITY MODELING

Performance and its variability can be summarized by a variety of statistics. Mean, range, standard deviation, variance, and interquartile range are a few summary statistics that describe performance and variability. However, the most precise characterization of any ordinal performance metric is the cumulative distribution function (CDF), or its derivative the probability density function (PDF). Previous techniques for predicting system performance have strictly modeled real-valued summary statistics because there exists a large base of mathematical techniques capable of approximating functions of the form $f : \mathbb{R}^d \to \mathbb{R}$. There is little systems work in approximating functions of the form $f : \mathbb{R}^d \to \{g \mid g : \mathbb{R} \to \mathbb{R}\}$.

A major hurdle when modeling functions such as the CDF or PDF is that certain properties must be maintained. It is necessary that a PDF $f : \mathbb{R} \to \mathbb{R}$ have the properties $f(x) \geq 0$ and $\int_{-\infty}^{\infty} f(x)dx = 1$. However, for a CDF $F : \mathbb{R} \to \mathbb{R}$ the properties are $F(x) \in [0, 1]$ and $F(x)$ is monotone increasing. For the purpose of function approximation, the properties of the CDF are more straightforward to maintain. This work utilizes the fact that a convex combination of CDFs results in a valid CDF. Given $G(x) = \sum_i w_i F_i(x)$, $\sum_i w_i = 1$, $w_i \geq 0$, and each F_i is a valid CDF, G must also be a valid CDF. A demonstration of how this is applied can be seen in Figure 1. Next, the three techniques used to model CDFs are summarized, a mechanism for identifying important features is proposed, and the error measure used throughout this work is presented. For each of the following, $n \in \mathcal{N}$ refers to the number of training samples (known system configurations), $d \in \mathcal{N}$ refers to the dimension of the data (ordinal system parameters), $x^{(i)} \in \mathbb{R}^d$ is a known system configuration with known performance CDF $F_{x^{(i)}}$, and $y \in \mathbb{R}^d$ refers to a previously unseen system configuration with performance CDF F_y.

A. Delaunay

The Delaunay triangulation is a well-studied geometric technique for producing an interpolant [11]. The Delaunay triangulation of a set of data points into simplices is such that there are no data points inside the sphere defined by the vertices of each simplex. For a d-simplex S with vertices $x^{(0)}$, $x^{(1)}$, ..., $x^{(d)}$, and functions $F_{x^{(i)}}$, $i = 0, \ldots, d$, $y \in S$ is a unique convex combination of the vertices,

$$y = \sum_{i=0}^{d} w_i x^{(i)}, \quad \sum_{i=0}^{d} w_i = 1, \quad w_i \geq 0, \quad i = 0, \ldots, d,$$

and the Delaunay interpolant F_y at y is given by

$$F_y = \sum_{i=0}^{d} w_i F_{x^{(i)}}.$$

The computational complexity of Delaunay interpolation (for the implementation used) is $\mathcal{O}(nd^4 \log d)$, which is capable of scaling reasonably to $d \leq 50$. In the present application, a Delaunay simplex S containing y is found, then the $d+1$ vertices (system configurations) of S are used to assign weights to each vertex and produce the predicted CDF for system configuration y.

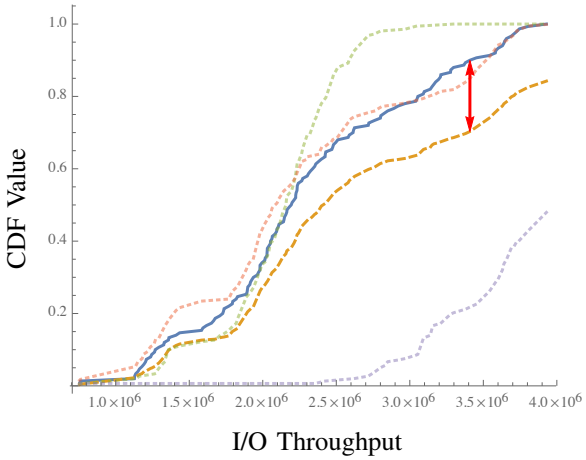

Fig. 1. In this example, the general methodology for predicting a CDF and evaluating error can be seen. The Delaunay method chose three source system configurations (dotted lines) and assigned weights {.3, .4, .3} (top to bottom). The weighted sum of the three known CDFs produces the predicted CDF (dashed line). The KS Statistic (red arrow) computed between the true CDF (solid line) and predicted CDF (dashed line) is 0.2 for this example. The KS test null hypothesis is rejected by p-value 0.01, however it is not rejected by p-value 0.001.

B. Max Box Mesh

The Max Box mesh is an interpolation technique that utilizes overlapping box splines [12] as basis functions that are shifted and scaled to have support over box shaped regions. The boxes are constructed such that each box has exactly one point in its interior while maintaining the largest minimum distance between the interior point and the closest face of the box (a measure of centrality). Given a set of box splines $\{b^{x^{(i)}}\}$ with the max box properties anchored at interior points $\{x^{(i)}\}$,

$$F_y = \frac{\sum_i b^{x^{(i)}}(y) F_{x^{(i)}}}{\sum_i b^{x^{(i)}}(y)}.$$

Note that the box splines always satisfy $b^{x^{(i)}}(y) \geq 0$. The computational complexity of interpolation via the Max Box mesh is $\mathcal{O}(n^2 d)$.

C. Voronoi Mesh

A well-studied technique for classification and approximation is the nearest neighbor algorithm [13]. Nearest neighbor inherently utilizes the convex region $C^{x^{(i)}}$ (Voronoi cell [14]) consisting of all points closer to $x^{(i)}$ than any other point $x^{(j)}$. The Voronoi mesh smooths the nearest neighbor approximation by utilizing the Voronoi cells to define support via a generic basis function $v : \mathbb{R}^d \to \mathbb{R}_+$ given by

$$v^{x^{(i)}}(y) = \left(1 - \frac{||y - x^{(i)}||_2}{2\ d(y - x^{(i)} \mid x^{(i)})}\right)_+,$$

where $d(w \mid x^{(i)})$ is the distance between $x^{(i)}$ and the boundary of the Voronoi cell $C^{x^{(i)}}$ in the direction w.

$v^{x^{(i)}}\left(x^{(j)}\right) = 0$ and $v^{x^{(i)}}$ has local support, giving the interpolated value

$$F_y = \frac{\sum_i v^{x^{(i)}}(y) F_{x^{(i)}}}{\sum_i v^{x^{(i)}}(y)},$$

where $0 \leq v^{x^{(i)}}(y) \leq 1$. The computational complexity of interpolation via the Voronoi mesh is $\mathcal{O}(n^2 d)$. All of the approximations are an interpolant involving a convex combination of known functions $F_{x^{(i)}}$.

D. Feature Weighting

It is well-known that an important procedure in any application of predictive methodologies is identifying those features of the data that are most relevant to making accurate predictions [15]. Selection strategies such as the floating searches studied in [16] or others compared in [17] can be too expensive for large approximation problems. Rather, this work poses feature selection as a continuous optimization problem. Let X be an $n \times d$ matrix of n known system configurations with d parameters each normalized to be in $[0, 1]$. Define an error function that computes the error of a predictive model trained on X diag w, $w \in \mathbb{R}^d$, by performing ten random splits with 80% of the rows of X diag w for training and 20% for testing. A minimum of this error function could be considered an optimal weighting of the features of X. Minimization is performed using a zero order method in the absence of a readily computable gradient.

E. Measuring Error

The performance of approximation techniques that predict functions can be analyzed through a variety of summary statistics. Mean error, mean absolute error, mean squared error, and the max absolute error are all popular measures. This work uses the max absolute error, also known as the Kolmogorov-Smirnov (KS) statistic [18] for its compatibility with the KS test.

The two-sample KS test is a useful nonparametric test for comparing two CDFs while only assuming stationarity, finite mean, and finite variance. The null hypothesis (that two CDFs come from the same underlying distribution) is rejected at level $p \in [0, 1]$ when

$$KS > \sqrt{-\frac{1}{2}\ln\left(\frac{p}{2}\right)}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}},$$

with distribution sample sizes $n_1, n_2 \in \mathcal{N}$. For all applications of the KS test presented in this work $n_1 = n_2 = 150$. An example of the round-trip prediction methodology from known and predicted distributions to the calculation of error can be seen in Figure 1.

III. DATA

This paper presents a variability modeling case study with a five-dimensional dataset produced by executing the IOzone benchmark [19] on a homogeneous cluster of computers. Each node contains two Intel Xeon E5-2637 CPUs offering a total of 16 CPU cores with 16GB of DRAM. While the CPU frequency varies depending on the test configuration, the I/O from IOzone is performed by an ext4 filesystem sitting above an Intel SSDSC2BA20 SSD drive. At the time of data collection, Linux kernel Version 4.13.0 was used. The system performance data was collected over two weeks by executing IOzone 150 times for each of a select set of approximately 18K system configurations, for a total of approximately 2.7M executions of IOzone. A single IOzone execution reports the max I/O throughput in kilobytes per second seen for the selected test type. The summary of the data components in $x^{(i)}$ for the experiments for this paper can be seen in Table I. Distributions of raw throughput values being modeled can be seen in Figure 2.

System Parameter	Values
File Size (KB)	4, 16, 64, 256, 1024, 4096, 8192, 16384
Record Size (KB)	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384
Thread Count	1, 8, 16, 24, 32, 40, 48, 56, 64
Frequency (GHz)	1.2, 1.6, 2, 2.3, 2.8, 3.2, 3.5
Test Type	Readers, Rereaders, Random Readers, Initial Writers, Rewriters, Random Writers

TABLE I

A DESCRIPTION OF SYSTEM PARAMETERS CONSIDERED FOR IOZONE. RECORD SIZE MUST BE $\leq$ FILE SIZE DURING EXECUTION.

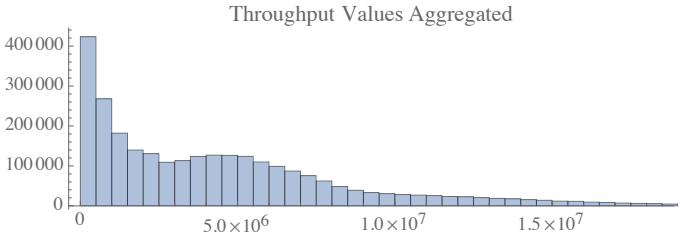

Fig. 2. Histogram of the raw throughput values recorded during all IOzone tests across all system configurations. The distribution is skewed right, with few tests having significantly higher throughput than most others.

Some mild preprocessing was necessary to prepare the data for modeling and analysis. All features were shifted by their minimum value and scaled by their range, mapping each feature independently into $[0, 1]$. This normalization ensures each feature is treated equally by the interpolation techniques and should be performed on all data before building models and making predictions regardless of application. All 150 repeated trials for a system configuration were grouped with that configuration. The only nonordinal feature in this data is the test type. All test types were treated as different applications and were separated for modeling and analysis, i.e., predictions for the "readers" test type were made using only known configurations for the "readers" test type.

IV. RESULTS

All three interpolation techniques are used to predict the distribution of I/O throughput values at previously unseen system configurations. In order to improve robustness of the error analysis, ten random selections of 80% of the IOzone data are used to train each model and the remaining 20% provide approximation error for each model. The recorded errors are grouped by unique system configuration and then

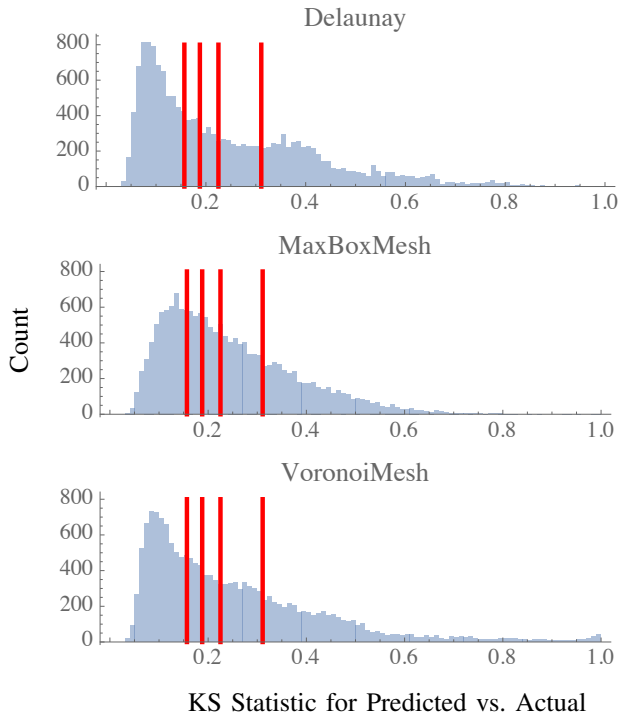

Fig. 3. Histograms of the prediction error for each modeling algorithm from ten random splits when trained with 80% of the data. The distributions show the KS statistics for the predicted throughput distribution versus the actual throughput distribution. The four vertical red lines represent commonly used p-values {0.05, 0.01, 0.001, 1.0e-6} respectively. All predictions to the right of a red line represent CDF predictions that are significantly different (by respective p-value) from the actual distribution according to the KS test.

averaged within each group. The samples are identical for each interpolation technique, ensuring consistency in the training and testing sets.

The aggregation of errors across all IOzone tests given 80% of the data as training can be seen in Figure 3. Agglomerate errors for each technique resemble a Gamma distribution. The percentages of significant prediction errors with varying p-values are on display in Table II. The primary p-value used for analyses in this work is 0.001, chosen because close to 2K predictions are made for each test type. Also, applications executed in cloud and HPC systems that could benefit from statistical modeling will be executed at least thousands of times. In line with this knowledge, it is important to ensure that only a small fraction of interpretable results could occur solely under the influence of random chance. When considering the $p = 0.001$ results for each technique, a little under half of the predicted CDFs are significantly different from the measured (and presumed) correct CDFs. A rejection rate of 45% would seem a poor result, however in this situation the complexity of the problem warrants a slightly different interpretation. These predictions are a very *precise* characterization of performance variability, in fact the strongest possible characterization of variability. Globally, only a little under half of the predictions fail to capture *all* of the characteristics of performance variability at new system configurations. It is also demonstrated later in this Section that this result can likely be improved.

Algorithm	P-Value	% N.H. Rejections
Delaunay	.05	58.4
Max Box Mesh		69.3
Voronoi Mesh		61.9
Delaunay	.01	51.1
Max Box Mesh		58.4
Voronoi Mesh		53.4
Delaunay	.001	44.1
Max Box Mesh		46.9
Voronoi Mesh		45.1
Delaunay	1.0e-6	31.4
Max Box Mesh		26.6
Voronoi Mesh		28.7

TABLE II

PERCENT OF NULL HYPOTHESIS REJECTIONS RATE BY THE KS-TEST WHEN PROVIDED DIFFERENT SELECTIONS OF p-VALUES. THESE ACCOMPANY THE PERCENT OF NULL HYPOTHESIS REJECTION RESULTS FROM FIGURE 3.

While interpreting null hypothesis rejection rates for these interpolation techniques, it is important to consider how the rejection rate reduces with increasing amounts of training data. Figure 4 displays the change in $p = 0.001$ null hypothesis rejection rate with increasing density of training data up to the maximum density allowed by this set. Delaunay interpolation provides the best results with the least training data by about 5%, but these low density rejection rates are unacceptably high (90%). Figure 4 clearly shows that this data set and/or the system variables used in the models of performance variability is inadequate to capture the full variability map from system parameters to performance CDF. Which or both obtains is not clear. A few well chosen data points can significantly improve the interpolants, and thus a careful study of the rejection instances is warranted, besides enlarging the set of system variables being modeled.

It may be misleading to consider the global performance of each prediction technique across all test types, as some test types are more difficult than others to predict and have more apparent latent variables. In Figure 5, the relative difficulty of each IOzone test type can be compared. The I/O test types analyzing reads are typically approximated with lower error than those test types analyzing writes. Regardless of test type, in the aggregate results the KS statistics hover consistently around 0.15, demonstrating an impressively low KS statistic for predictions. In order to address the opacity of aggregate analysis, another case study and an application of the methodology from Section II-D is presented in Table III.

The results presented in Table III are achieved by permitting each approximation technique 300 iterations of simulated annealing. In each iteration, the impact of potential weights on the average KS statistic were considered. All weights were kept in the range [0,2], and were applied to the normalized features for frequency, file size, record size, and number of threads. All three approximation techniques had similar optimal weights of approximately $(.001, 2, 1.7, 1.5)$ for frequency, file size, record size, and number of threads, respectively. Recall that each interpolation technique uses

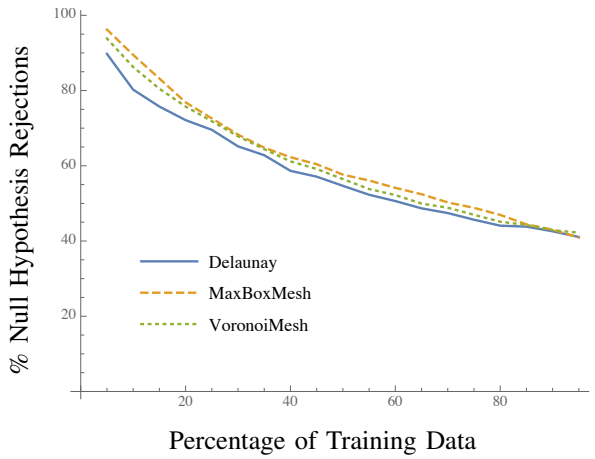

Fig. 4. The performance of each algorithm on the KS test ($p = 0.001$) with increasing amounts of training data averaged over all IOzone test types and ten random splits of the data. The training percentages range from 5% to 95% in increments of 5%. Delaunay is the best performer until 95% of data is used for training, at which Max Box mesh becomes the best performer by a fraction of a percent.

Algorithm	P-Value	Unweighted % N.H. Rejection	Weighted % N.H. Rejection
Delaunay	.05	24.9	30.2
Max Box Mesh		21.3	21.2
Voronoi Mesh		18.7	11.3
Delaunay	.01	21.6	27.4
Max Box Mesh		16.4	16.4
Voronoi Mesh		14.9	7.0
Delaunay	.001	19.7	25.4
Max Box Mesh		13.1	13.1
Voronoi Mesh		12.3	4.6
Delaunay	1.0e-6	17.9	23.4
Max Box Mesh		11.3	11.3
Voronoi Mesh		8.5	2.3

TABLE III

THE NULL HYPOTHESIS REJECTION RATES FOR VARIOUS p-VALUES WITH THE KS-TEST. THESE RESULTS ARE STRICTLY FOR THE "READERS" IOZONE TEST TYPE AND SHOW UNWEIGHTED RESULTS AS WELL AS THE RESULTS WITH WEIGHTS TUNED FOR MINIMUM ERROR (KS STATISTIC) BY 300 ITERATIONS OF MODIFIED SIMULATED ANNEALING. NOTICE THAT THE WEIGHTS IDENTIFIED FOR THE DELAUNAY MODEL CAUSE DATA DEPENDENT TUNING, REDUCING PERFORMANCE. MAXBOXMESH PERFORMANCE IS IMPROVED BY A NEGLIGIBLE AMOUNT. VORONOIMESH PERFORMANCE IS NOTABLY IMPROVED.

small distances to denote large influences on predictions, meaning that frequency was the most important feature when predicting variability for the "readers" test type, followed not-so-closely by number of threads, then record size.

The "readers" test type results demonstrate that the underlying prediction techniques work and are capable of seeing rejection rates below 5% when tuned for a given application. It is important to emphasize that the roughly 95% of predictions for which the null hypothesis was not rejected are predicting the *precise* distribution of I/O throughput that will be witnessed at a previously unseen system configuration. To the authors' knowledge, there is no existing methodology that is generically applicable to any system performance metric, agnostic of system architecture, and capable of making such powerful predictions.

V. DISCUSSION

The results of the IOzone case study indicate that predicting the CDF of I/O throughput at previously unseen system configurations is a challenging problem. The KS statistic captures the worst part of any prediction and hence provides a conservatively large estimate of approximation error. The average absolute errors in the predicted CDFs are always lower than the KS statistics. However, the KS statistic was chosen because of the important statistical theory surrounding it as an error measure. Considering this circumstance, a nonnegligible volume of predictions provide impressively low levels of error. Powerful predictive tools such as those presented in this work allow for more in-depth analysis of system performance variability. For example, system configurations that are most difficult to predict in these tests are likely "outlier" configurations that do not resemble those configurations that share many similar parameters. Analysis of these configurations may provide valuable insight into effective application specific operation of computer systems.

As mentioned in Section I, no prior work has attempted to model an arbitrary performance metric for a system to such a high degree of precision. All previous statistical modeling attempts capture a few (< 3) ordinal performance metrics. Generating models that have such high degrees of accuracy allows system engineers to identify previously unused configurations that present desired characteristics. Service level agreements (SLAs) in cloud computing environments are cause for capital competition that is affected heavily by system performance [20]. Users prefer SLAs that allow the most computing power per monetary unit, incentivizing service providers to guarantee the greatest possible performance. Overscheduling and irregular usage patterns force cloud service providers to occasionally overload machines, in which case precise models of system performance can be used to statistically minimize the probability of SLA violation. Similar targeted performance tuning techniques can be applied to HPC system configuration to maximize application throughput or minimize system power consumption.

A final application domain affected by this methodology is computer security. Collocated users on cloud systems have received attention recently [21]. If a malicious collocated user is capable of achieving specific insight into the configuration of the system, or the activity of other collocated users by executing performance evaluation programs (i.e., IOzone), a new attack vector may present itself. Malicious users could be capable of identifying common performance distributions of vulnerable system configurations and vulnerable active user jobs. This knowledge may allow targeted exploits to be executed. Light inspection of raw IOzone I/O throughputs provides substantial evidence that distinct performance distributions coincide closely with specific system configuration parameters. Conversely, a service provider may

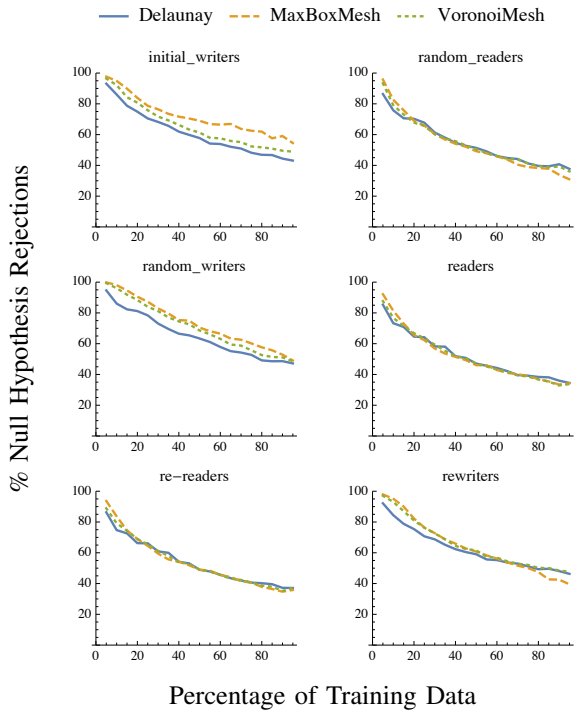

Fig. 5. The percentage of null hypothesis rejections for predictions made by each algorithm on the KS test ($p = 0.001$) over different IOzone test types with increasing amounts of training data. Each percentage of null hypothesis rejections is an average over ten random splits of the data. The training percentages range from 5% to 95% in increments of 5%. The read test types tend to allow lower rejection rates than the write test types.

defend against such attacks by deliberately obfuscating the performance of the machine. Models such as those presented in this paper could identify optimal staggering and time-delay whose introduction into the system would prevent malicious users from identifying system configurations and active jobs.

Results presented in Table III are particularly interesting, demonstrating that Delaunay appears most vulnerable to data dependent tuning, Max Box mesh is largely insensitive to such tuning, and Voronoi mesh benefits (for this data set) from the tuning.

There are many avenues for extending this modeling methodology. One extension is to add categorical variables to the models. Presently the rejection rate of distribution predictions can only be reduced with large volumes of performance data, however the judicious choice (via experimental design, e.g.) of new data points may be able to effectively reduce the amount of training data required. Finally, more case studies need to be done to test the robustness of the present modeling techniques to changes in domain and performance metric.

VI. CONCLUSION

The methodology presented is capable of providing new insights, extending existing analyses, and improving the management of computational performance variability. Delaunay, Max Box mesh, and Voronoi mesh interpolation are viable techniques for constructing approximations of performance cumulative distribution functions. A case study on I/O throughput demonstrated that the models are capable of effectively predicting CDFs for most unseen system configurations for any of the available I/O test types. The present methodology represents a notable increase in the ability to statistically model arbitrary system performance metrics involving the interaction of many ordinal system parameters.

REFERENCES

[1] E. Grobelny, D. Bueno, I. Troxel, A. D. George, and J. S. Vetter, "Fase: A framework for scalable performance prediction of hpc systems and applications," *Simulation*, vol. 83, no. 10, pp. 721–745, 2007.

[2] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS'09)*. IEEE, 2009, pp. 1–11.

[3] G. Wang, A. Khasymski, K. R. Krish, and A. R. Butt, "Towards improving mapreduce task scheduling using online simulation based predictions," in *International Conference on Parallel and Distributed Systems (ICPADS), 2013*. IEEE, 2013, pp. 299–306.

[4] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha, "A framework for performance modeling and prediction," in *Supercomputing, ACM/IEEE Conference*. IEEE, 2002, pp. 21–21.

[5] D. H. Bailey and A. Snavely, "Performance modeling: Understanding the past and predicting the future," in *European Conference on Parallel Processing*. Springer, 2005, pp. 185–195.

[6] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, and J. C. Sancho, "Using performance modeling to design large-scale systems," *Computer*, vol. 42, no. 11, 2009.

[7] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment," in *12th IEEE International Conference on High Performance Computing and Communications (HPCC)*. IEEE, 2010, pp. 273–280.

[8] 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*, vol. 11, no. 1, pp. 3–16, 2008.

[9] P. De, R. Kothari, and V. Mann, "Identifying sources of operating system jitter through fine-grained kernel instrumentation," in *IEEE International Conference on Cluster Computing*. IEEE, 2007, pp. 331–340.

[10] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf, "Managing variability in the io performance of petascale storage systems," in *International Conference on High Performance Computing, Networking, Storage and Analysis (SC), 2010*. IEEE, 2010, pp. 1–12.

[11] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[12] C. De Boor, K. Höllig, and S. Riemenschneider, *Box Splines*. Springer Science & Business Media, 2013, vol. 98.

[13] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[14] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen." *Journal für die Reine und Angewandte Mathematik*, vol. 40, pp. 209–227, 1850.

[15] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[16] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[17] F. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection," *Pattern Recognition in Practice IV*, vol. 1994, pp. 403–413, 1994.

[18] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[19] W. D. Norcott. (2017) Iozone filesystem benchmark. [Online]. Available: http://www.iozone.org

[20] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service level agreement in cloud computing," *Wright State University CORE Scholar Repository*, 2009.

[21] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.