Modeling I/O Performance Variability in High-Performance Computing Systems Using Mixture Distributions

Li Xu[a], Yueyao Wang[a], Thomas Lux[b], Tyler Chang[b], Jon Bernard[b], Bo Li[b], Yili Hong[a,*], Kirk Cameron[b], Layne Watson[b,c]

[a]*Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, VA*
[b]*Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA*
[c]*Departments of Mathematics and Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA*

Abstract

Performance variability is an important factor of high-performance computing (HPC) systems. HPC performance variability is often complex because its sources interact and are distributed throughout the system stack. For example, the performance variability of I/O throughput can be affected by factors such as CPU frequency, the number of I/O threads, file size, and record size. In this paper, we focus on the I/O throughput variability across multiple executions of a benchmark program. For a given system configuration, the distribution of throughputs from run to run is of interest. We conduct large-scale experiments and collect a massive amount of data to study the distribution of I/O throughput under tens of thousands of system configurations. Despite normality often being assumed in the literature, our statistical analysis reveals that the performance variability is not normally distributed under most system configurations. Instead, multimodal distributions are common for many system configurations. We propose the use of mixture distributions to describe the multimodal behavior. Various underlying parametric distributions such as normal, gamma, and the Weibull are considered. We apply an expectation maximization (EM) algorithm for parameter estimation and use the Bayesian information criterion (BIC) for parametric model selections. We also illustrate how to use the estimated mixture distribution to calculate the number of runs needed for future experiments on variability analysis. The paper provides a useful tool set in studying the behavior of performance variability.

Keywords: I/O variability, Performance analysis, mixture model, uncertainty quantification, EM algorithm, model selection.

*Corresponding author

Email addresses: lix1992@vt.edu (Li Xu), yueyao94@vt.edu (Yueyao Wang), tchlux@vt.edu (Thomas Lux), thchang@vt.edu (Tyler Chang), jbernard@jbernard.io (Jon Bernard), libo.1024.com@gmail.com (Bo Li), yilihong@vt.edu (Yili Hong), cameron@cs.vt.edu (Kirk Cameron), ltw@cs.vt.edu (Layne Watson)

1. Introduction

1.1. Background

High performance computing (HPC) systems must continue to improve in order to address increases in the complexity and scale of computing demands. However, variability is a huge impediment to the effective scaling of HPC systems. HPC system variability has several forms. One common example is that given a computing task, repeated executions of that task will require different amounts of running time. In some scenarios, the difference in running time is nonnegligible and hard to predict. Observations and experiments have shown that performance variability in HPC systems exists and is common [1]. Variability makes system optimization challenging. Thus, variability control is an important problem in HPC research.

One important step in variability control is to describe the system variability. The description of system variability involves two steps, namely, running proper experiments on the system to collect enough data, and applying appropriate statistical tools to analyze the experimental data. It is important that the statistical tool chosen can capture the complex distribution patterns under different system configurations. Little comprehensive work has been done describing HPC system variability. One possible reason could be the lack of resources to run enough experiments to collect sufficient experimental data for statistical analysis. In addition, due to the complexity of computing systems, both in terms of hardware and applications, the source of performance variability is not easy to identify. In most existing work, the distribution of the performance of an HPC system is assumed to follow a normal distribution.

Our research finds that the distribution of the performance is much more complicated than a normal distribution. In the throughput data collected from the IOzone benchmark [2], multimodal distributions for performance are common across many system configurations. Although the details will be discussed in Section 2, Figure 1 shows typical examples of I/O throughput distributions under four different system configurations collected from hundreds of runs of IOzone for those configurations. The histograms of throughputs show little to no evidence of being normal distributions. Thus, the distribution of I/O throughputs is complicated and needs further investigation.

In this study, we apply advanced statistical tools to describe the characteristics of the distribution of system performance variability. We collect large scale data by running the IOzone benchmark on an HPC system. Mixture distributions are used to model the throughput data collected from the IOzone benchmark. We develop an expectation-maximization (EM) algorithm to automatically identify the distribution parameters, and use the Bayesian information criterion (BIC) to select the underlying distribution and the number of compo-

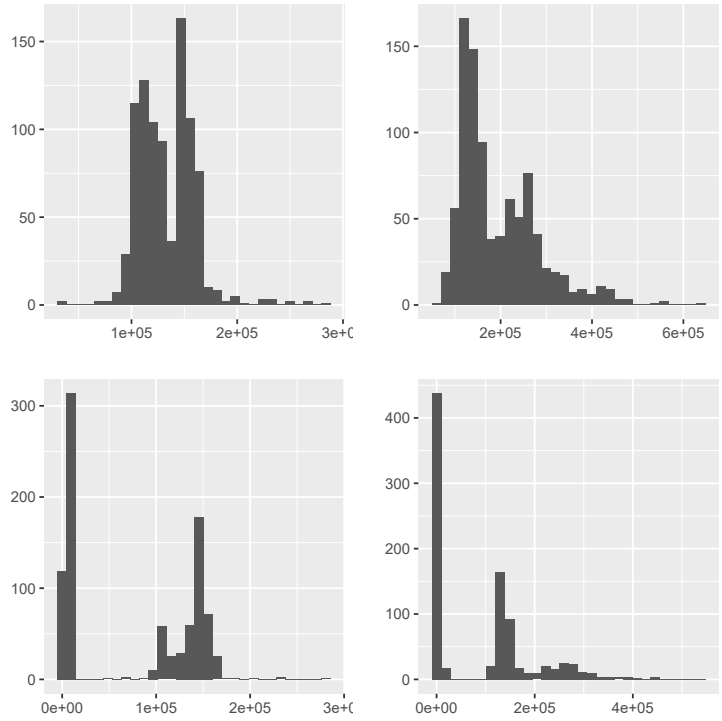

Figure 1: Histograms for typical throughput distributions from four example configurations with different number of modes. The x-axis is the throughput (KB/s), and the y-axis is the counts.

nents in the mixture distribution. The result of the data analysis implies that the standard normality assumption for system performance is not accurate. Multimodal phenomena show up in many system configurations. That is, the distribution of the throughput follows a multimodal mixture distribution for many of the cases. The parametric mixture distribution not only provides an automatic tool to discover the distributional behaviors of the performance, but also provides a basis for computing an appropriate sample size (i.e., the number of runs) for future experiments. We develop a procedure for estimating sample size using the parametric mixture distribution, which may be beneficial for future research in controlling variability.

1.2. Related Literature

Researchers have noticed the existence of variability in HPC systems for more than a decade [3, 4]. Performance variability was analyzed in some existing work [1, 5, 6, 7]. Prior to this paper, statistical modeling of variability was restricted to one or two system parameters [8, 9, 10, 11]. These previous statistical models provide useful results for specific applications and work with simplified systems, heavy data augmentation, or strong assumptions about performance variability.

Most existing work on performance variability has focused on operating system (OS) induced variability [12, 13]. Yet, system I/O variability has been particularly difficult for statistical models to capture [6, 9]. I/O variability under two system parameters was modeled in [14]. Recent generic I/O modeling has been nonparametric, and hence limits the number of conclusions that can be drawn from the model structure [15]. Other recent I/O modeling work uses black-box machine learning methods to predict variance and mean, but lacks the ability to identify underlying distributions [16, 17, 18]. Maricq et al. [19] propose an open source tool for analyzing the variability under different server configurations. They also use nonparametric statistical methods to see how representative an individual server is of the general population and determine how many experimental trials one needs to capture the system variability behavior.

Variability is becoming an important factor in building supercomputers [20, 21], cloud computing systems [22, 23, 24] and robust scalable network security [15]. For this reason it is critical that we improve our ability to understand, predict, and manage variability with sophisticated statistical models.

Performance analysis also appears in many areas. For example, Umar et al. [25] propose two energy models and estimation frameworks for Aspen DSL. Using experimental data, their model can reduce energy consumption by 45%. Rolinger et al. [26] present a performance analysis framework for tensor decomposition. Their study concentrates on measured memory usage, processor stall cycles, execution time, and scalability. Tong et al. [27] present a trace-based analysis tool that can classify MPI applications as bottlenecks based on their performance. The classification model can help better understand application performance limiting factors. In addition to performance control, managing energy consumption is also popular [28, 29]. Yu et al. [28] focus on the trade off between power consumption and performance using a trace based validation, and show that the model is accurate and scalable. Lemeire et al. [29] give a methodology to study the computing efficiency and the overhead impacts on runtime performance. They define three types of efficiency and propose analytical formulas to measure and predict the respective efficiencies.

For the related statistical work, mixture distributions are widely used to describe distributions with multimodal behaviors. The EM algorithm is usually used to find the maximum likelihood estimates of parameters in mixture distributions [30, 31]. For parametric model selection, many statistical criteria are based on the maximum likelihood criterion. An obvious drawback of the maximum likelihood criterion is that it has no penalty on the number of parameters in the model. As a result, the best models are always the ones that have the most parameters, which has the risk of over-fitting. Many extensions of maximum likelihood are based on the idea of adding a penalty on the number of parameters. Two examples are the Akaike information criterion (AIC) [32] and the Bayesian information criterion (BIC)

[33]. To estimate the sample size and quantify the uncertainty in estimation, confidence intervals can be used and early work can be traced back to [34]. In this paper, we use an approach that is similar to [35] to measure the uncertainty in estimating the quantiles of the distribution, and we develop a procedure for the sample size calculation. Since our mixture model is different from the lifetime model used in [35], we derive the theoretical formula for our specific model to construct confidence intervals.

In summary, our proposed parametric analysis uses a mixture distribution for I/O variability modeling and analysis that is new to the literature. The mixture distribution provides an effective tool for describing the multimodal behavior of the I/O performance distribution. The model is also useful for designing future experiments for I/O variability studies.

1.3. Overview

The rest of the paper is organized as follows. Section 2 gives a description to the IOzone experimental setup and the collected dataset. Section 3 describes the statistical model using mixture distributions, an EM algorithm for parameter estimation, and parametric model selections. Section 4 develops a method to use the mixture model to determine sample size for future experiments. Section 5 presents the modeling and data analysis results and discusses the findings. Section 6 gives a numerical example for sample size determination. Section 7 concludes and discusses areas for future research. Some technical details are available in the appendix.

2. Experiment Setup and Data Collection

2.1. Experiment Environment

We decided to focus our analysis on storage using the rationale that persistent storage would exhibit the highest variance among available local resources while offering the most defined and interesting performance characteristics. That is, we identified I/O as a likely high variance operation, and the IOzone benchmark is used to generate I/O workloads with various configurations. While IOzone is not identical to any particular workload, it does cover a board range of possible I/O patterns including operations such as random access that are likely to exhibit higher variance. The total execution time and throughput reports are used to analyze and compare different configurations to determine which selections are subject to the highest levels of performance variance.

We ran the experiments on a 12-node server. The nodes are all identical Dell PowerEdge R630s. Each node is configured as Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50 GHz, 16GB DRAM (2 DIMMs), and a new 200GB SSD with Intel model SSDSC2BA200G4R. There are 2 sockets with 4 cores per socket. So in total, there are 8 physical cores. With hyper-threading enabled, there are 16 CPUs seen by the OS and user space. The operating system

is Debian GNU/Linux with a 4.14 kernel and IOzone version 3.465. The installation is the most minimal possible, that is only systemd and sshd are running in user space. Turbo Boost is disabled and we use cpufreq-utils to manually set the CPU0 frequency prior to running the benchmark. The task scheduling we used is the default completely fair scheduling (CFS). The SSD block device is issued a TRIM command prior to benchmark execution. This allows us to begin each run with the most identical device state possible. Benchmarks are run in a background process and no remote access occurs during the execution of a workload.

2.2. Dataset collection

The dataset we used in this study has variables including HPC system hardware and application parameters. The attributes and levels are shown in Table 1. The hardware configuration includes CPU clock frequency. The application configurations include number of threads, file size, record size, and I/O operation mode. The file size (fs) and record size (rs) have a constraint that their ratio fs/rs must be an integer. In total we have 22734 different settings.

Regarding I/O access patterns, we studied six I/O operation modes in this paper as shown in the last row of Table 1. Specifically, the six I/O operation modes under consideration are random readers, rewriters, initial writers, readers, random writers, and re-readers. The detailed descriptions for those six modes are available in [2]. Similar behaviors have been observed in other I/O operation modes. Cameron et al. [1] studied 13 I/O operation modes in a small scale for the prediction of throughput variance, and saw similar multimodal behaviors.

The matrix plot in Figure 2 shows the combinations of all four numerical variables in the experiments. It describes how frequency, threads, file size, and record size are configured for each of the I/O operation modes. The diagonal plots show the name of the axes. The off-diagonal plots show the corresponding scatter plots with respect to the pair of variables. For example, the subplot in the first row, second column shows the scatter plot for all settings for frequency and file size.

While clock frequency and I/O mode have relatively few possible values on the chosen systems, other configurable system parameters have prohibitively many options for brute-force testing. Given that much of the system I/O software and underlying I/O hardware operate of powers of two, we use an exponentially spaced selection values for file size (powers of four) and record size (powers of two). The number of threads was chosen to be a multiple of eight because there are eight physical CPU cores available on the systems.

The output of an experiment is the throughput of IOzone given a certain system configuration (in a scale of 10^6 KB/s). For a given system setting shown in Table 1, we run the IOzone filesystem benchmark [2] at least 150 times to capture the characteristics of the throughput distribution. To illustrate some general patterns in the throughput distribution,

Table 1: System parameters and their levels used in the study of I/O variability.

	System Parameters	No. of Levels	Levels
Hardware	CPU Clock Frequency (GHz)	7	1.2, 1.6, 2.0, 2.3, 2.8, 3.2, 3.5
Application	Number of Threads	9	1, 8, 16, 24, 32, 40, 48, 56, 64
	File Size (KB)	10	4, 16, 64, 256, 1024, 4096, 8192, 32768, 65536
	Record Size (KB)	13	4, 8, 16, 32, 64, 128, 256, 512 1024, 2048, 4096, 8192, 16384
	I/O Operation Mode	6	random_readers, rewriters, initial_writers, readers, random_writers, re-readers

we select four typical settings and plot the histograms of the throughputs in Figure 1. From those histograms, we can see that the distributions of throughputs have a clear pattern of multiple modes. The top two histograms show two major modes together with some possible small modes in the tails. The bottom two histograms show three major modes with some possible small modes in the tails. All of them can not be described using a single normal distribution, indicating more sophisticated analytical methods are needed.

3. Mixture Model and Parameter Estimation

3.1. The Mixture Model

In this section, we introduce mixture models for the throughput data and develop a tailored EM algorithm for the estimation of the parameters in the mixture models. We represent the throughput under a given configuration by a random variable. In addition, the throughput of each run under the same configuration can be treated as independent and identically distributed (iid). Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ where the X_i are iid random variables each representing the throughput of one run under the same configuration, and n is the number of runs. We model X_i by a k-component mixture distribution with probability density function (PDF),

$$f(x, \boldsymbol{\theta}) = \sum_{j=1}^{k} \pi_j f_j(x, \boldsymbol{\theta}_j), \tag{1}$$

where $\pi_j \geq 0$ is the proportion of the jth component, $f_j(x, \boldsymbol{\theta}_j)$ is the PDF of the j-th component for a type of distribution (e.g., normal distribution) with parameter vector $\boldsymbol{\theta}_j$. Let $\boldsymbol{\theta}$ be the vector for all parameters and note that $\sum_{j=1}^{k} \pi_j = 1$.

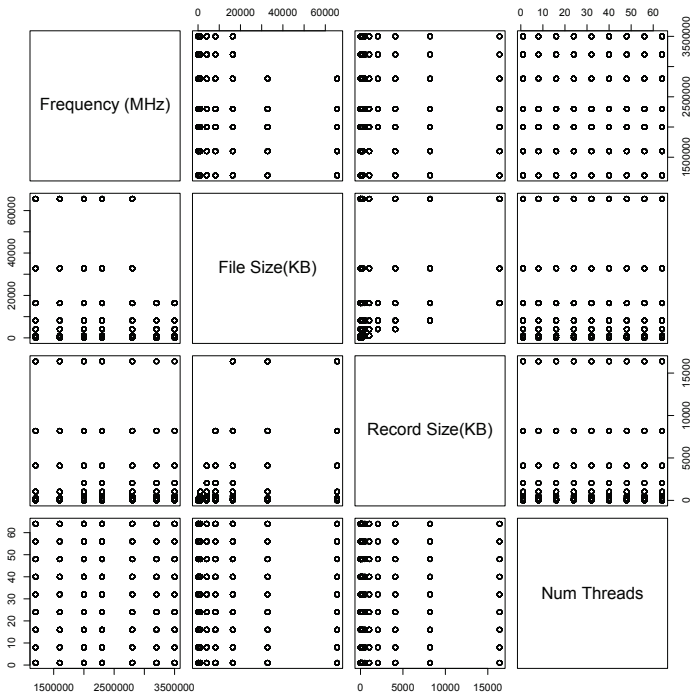

Figure 2: Grid of plots showing all pairwise combinations of the four numeric variables. Note that at each configuration (dot) represented in each plot, there are six separate experiments covering different I/O operation modes.

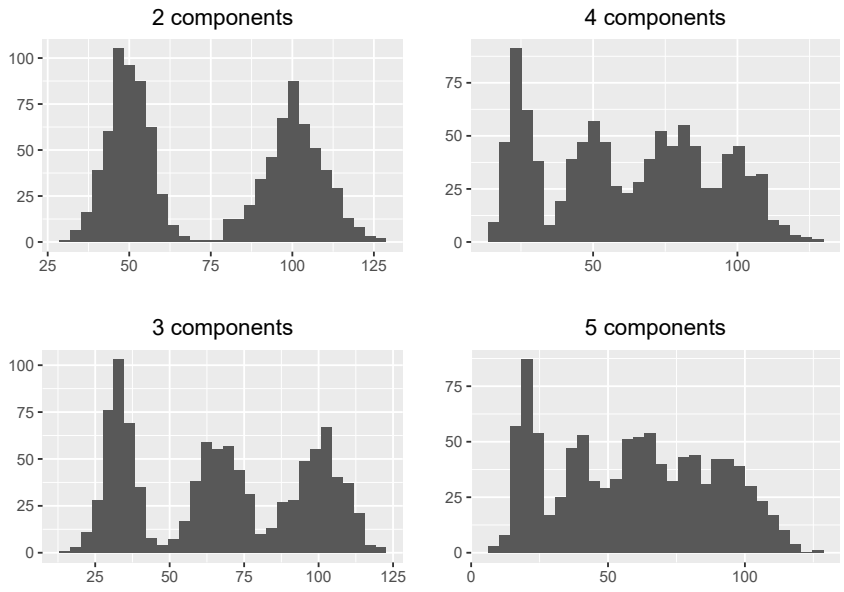

Figure 3: Histograms of random samples from mixture normal distributions with different number of components.

For the model in (1), each component represents a mode in the throughput distribution. To illustrate the flexibility of the mixture model in describing multimodal data, we simulate some data from a normal mixture model (i.e., using a normal distribution as the underlying component distribution in (1)) with the number of components ranging from two to five. The simulated data is shown in Figure 3, which reveals similar patterns to those in Figure 1 from the real data. From the figure, we can see that by choosing the number of components and the parameters $\boldsymbol{\theta}$ for the mixture distribution, one can obtain a much more accurate description of throughput data than when using a single normal distribution.

To use the mixture model in (1), one needs to specify the number of components k, and the underlying distributions $f_j(x, \boldsymbol{\theta}_j)$. The specification of k will be discussed in Section 3.3. For the underlying distribution type, we consider the normal, gamma, the Weibull, lognormal, loglogistic, and the Frechét distributions, which are widely-used in statistical modeling. The PDFs of those distributions are listed in Table 2.

Let $\boldsymbol{\theta}_j = (\mu_j, \sigma_j)^T$. For the normal, lognormal, the Weibull, loglogistic, and the Frechét distributions, μ_j is the location parameter and σ_j is the scale parameter. For the gamma distribution, μ_j is the shape parameter and σ_j is the scale parameter. Thus the parameter vector to be estimated for the mixture model is $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_{k-1}, \mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k)^T$. Note that the total number of parameters here is $p = 3k - 1$ for the distributions listed in Table 2. For example, if the number of components in a normal mixture model is k, there are k location parameters, k scale parameters, and $k - 1$ parameters for the proportion π_j

Table 2: Probability density functions (PDFs), $f(x, \mu_j, \sigma_j)$, for the different distributions under consideration. Here, $\phi_{\text{sev}}(z) = \exp[z - \exp(z)]$, $\phi_{\text{logis}}(z) = \exp(z)/[1 + \exp(z)]^2$, $\phi_{\text{lev}}(z) = \exp[-z - \exp(-z)]$, and $\Gamma(\cdot)$ is the gamma function.

Distribution	PDF
normal	$\frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left[-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right]$
gamma	$\frac{1}{\Gamma(\mu_j)\sigma_j^{\mu_j}} x^{\mu_j - 1} \exp\left(-\frac{x}{\sigma_j}\right)$
Weibull	$\frac{1}{\sigma_j x}\phi_{\text{sev}}\left[\frac{\log(x)-\mu_j}{\sigma_j}\right]$
lognormal	$\frac{1}{x\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{[\log(x)-\mu_j]^2}{2\sigma_j^2}\right\}$
loglogistic	$\frac{1}{\sigma_j x}\phi_{\text{logis}}\left[\frac{\log(x)-\mu_j}{\sigma_j}\right]$
Frechét	$\frac{1}{\sigma_j x}\phi_{\text{lev}}\left[\frac{\log(x)-\mu_j}{\sigma_j}\right]$

of the jth distribution component. $\sum_j \pi_j = 1$ determines π_k.

3.2. The EM Algorithm for Parameter Estimation

The EM algorithm uses an iterative approach to find the maximum likelihood estimates (MLE) of parameters, which is widely used to estimate statistical parameters for various models ([30]). The implementation of the EM algorithm under our mixture model with many component distributions needs tailored work. For the model in (1), we add a latent random variable $\boldsymbol{Z} = (Z_1, \ldots, Z_n)$ to indicate which component the X_i is sampled from. Given the component, X_i follows the distribution with PDF $f_i(x, \boldsymbol{\theta}_j)$. That is

$$X_i | Z_i = j \sim f_j(x, \boldsymbol{\theta}_j). \tag{2}$$

Here, "$\sim$" means "follows." Then the likelihood for $(\boldsymbol{X}, \boldsymbol{Z})$ is

$$L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{Z}) = \prod_{i=1}^{n} \prod_{j=1}^{k} \left[\pi_j f_j\left(X_i, \boldsymbol{\theta}_j\right)\right]^{\mathbb{I}(Z_i = j)}, \tag{3}$$

where

$$\mathbb{I}(Z_i = j) = \begin{cases} 1, & Z_i = j \\ 0, & Z_i \neq j \end{cases}. \tag{4}$$

The marginal likelihood for $\boldsymbol{X}$ is

$$L(\boldsymbol{\theta}, \boldsymbol{X}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \pi_j f_j(x_i, \boldsymbol{\theta}_j). \tag{5}$$

The EM algorithm can find the maximum likelihood estimates of $\boldsymbol{\theta}$ without knowing the latent variable $\boldsymbol{Z}$. With initial values for $\boldsymbol{\theta}$, the EM algorithm performs a fixed point iteration to determine a final value for $\boldsymbol{\theta}$, which is described in the following sections.

3.2.1. Choosing Initial Values

We need initial parameter values, which are denoted by,

$$\boldsymbol{\theta}^{(0)} = (\pi_1^{(0)}, \ldots, \pi_{k-1}^{(0)}, \mu_1^{(0)}, \ldots, \mu_k^{(0)}, \sigma_1^{(0)}, \ldots, \sigma_k^{(0)}).$$

We use the k-means clustering technique to divide our sample x_i, $i = 1, \ldots, n$ into k groups. Let n_j, $j = 1, \ldots, k$ be the number of points in each group, and we have $\sum_{j=1}^{k} n_j = n$. Then we set $\pi_j^{(0)} = n_j/n$. For the initial values $\mu_j^{(0)}$ and $\sigma_j^{(0)}$, suppose the j-th group has data points $\{x_{j_1}, \ldots, x_{jn_j}\}$. Then $\mu_j^{(0)}$ and $\sigma_j^{(0)}$ are chosen to be the MLE using these data points under the corresponding underlying component distribution. For example, if we use a normal mixture model, then $\mu_j^{(0)}$ and $\sigma_j^{(0)}$ are the MLE of the normal distribution using data $\{x_{j_1}, \ldots, x_{jn_j}\}$.

3.2.2. Fixed Point Iteration

The expected loglikelihood for $\boldsymbol{\theta}$ given the current estimate $\boldsymbol{\theta}^{(t)}$ is defined by

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) =& \mathbf{E}_{\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{(t)}} \log\left[L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{Z})\right] \\ =& \sum_{j=1}^{k} \sum_{i=1}^{n} P(Z_i = j|X_i = x_i, \boldsymbol{\theta}^{(t)}) \cdot \log\left[\pi_j f_j(x_i, \mu_j, \sigma_j)\right]. \end{aligned} \tag{6}$$

The updated estimate for $\boldsymbol{\theta}$, denoted by $\boldsymbol{\theta}^{(t+1)}$, is the value of $\boldsymbol{\theta}$ that maximizes $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$. That is

$$\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}). \tag{7}$$

The update for π_j has a closed-form expression. Let T_{ij} be the conditional probability of being in component j given the data. In particular,

$$T_{ij}^{(t)} = P(Z_i = j|X_i = x_i, \boldsymbol{\theta}^{(t)}) = \frac{\pi_j^{(t)} f_j(x_i, \boldsymbol{\theta}_j^{(t)})}{\sum_{l=1}^{k} \pi_l^{(t)} f_l(x_i, \boldsymbol{\theta}_l^{(t)})}. \tag{8}$$

Then π_j is updated as

$$\pi_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} T_{ij}^{(t)}. \tag{9}$$

For the normal and lognormal mixture models, the update of the location and scale parameters (i.e., μ and σ in $\boldsymbol{\theta}$) has closed forms. Specifically, for a normal mixture model, the updating formula is

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{ij}^{(t)} x_i}{\sum_{i=1}^{n} T_{ij}^{(t)}} \quad \text{and} \quad \sigma_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{ij}^{(t)} \left(x_i - \mu_j^{(t+1)}\right)^2}{\sum_{i=1}^{n} T_{ij}^{(t)}}. \tag{10}$$

For a lognormal mixture model, the updating formula is

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{ij}^{(t)} \log(x_i)}{\sum_{i=1}^{n} T_{ij}^{(t)}} \quad \text{and} \quad \sigma_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{ij}^{(t)} \left(\log(x_i) - \mu_j^{(t+1)}\right)^2}{\sum_{i=1}^{n} T_{ij}^{(t)}}. \tag{11}$$

The Weibull, gamma, loglogistic and Frechét mixture models do not have closed-form updates. So we use the gradient descent method to update $\boldsymbol{\theta}^{(t)}$. We repeat the fixed point iteration until $\boldsymbol{\theta}^{(t)}$ converges.

3.3. Model Selection

To specify the mixture model in (1), we need to determine the number of components k and the underlying distribution type $f_j(x, \boldsymbol{\theta}_j)$. We use the Bayesian information criterion (BIC) for model comparisons, which is defined by

$$\text{BIC} = -2\log(\widehat{L}) + \log(n)p, \tag{12}$$

where n is the number of iid samples, and $p = 3k - 1$ is the number of parameters in the model. Here,

$$\widehat{L} = \prod_{i=1}^{n}\sum_{j=1}^{k} \widehat{\pi}_j f_j(x_i, \widehat{\boldsymbol{\theta}}_j) \tag{13}$$

is the maximum likelihood value evaluated at $\widehat{\pi}_j$ and $\widehat{\boldsymbol{\theta}}_j$being the parameters estimated by the EM algorithm. The second item in equation (12) is the penalty for the number of parameters, which enables us to determine the number of components in the mixture model. The best model should have the smallest BIC value since it has a larger likelihood value and fewer number of parameters.

4. Sample Size Determination

One application of the mixture model is to determine the number of throughput runs (i.e., sample size n) for future experiments, which is a sample size determination problem. For a given configuration, researchers are typically interested in how many runs are needed so that the distribution of the throughput can be estimated with enough precision. Among statistical tools, confidence intervals provide an easy way to display the scale of uncertainty. A narrow confidence interval often means a smaller uncertainty in parameter estimation. The width of a confidence interval is related to the standard error, which typically reduces as the number of runs (i.e., the sample size n) increases.

Note that we use X to denote the random variable for the throughput under a given system configuration. The cumulative distribution function (CDF) of X is defined as $F(x) = \Pr(X \leq x)$. The quantile function x_q is defined as the smallest x such that $F(x) = q$ for continuous distributions, where $0 < q < 1$. Because we are interested in estimating the distribution function $F(x)$, it is not straightforward to measure the uncertainty in the estimation of a function. In the literature, people use the uncertainties in lower and upper quantiles (e.g., $x_{0.1}$ and $x_{0.9}$) to represent the uncertainties in the estimation of a distribution function (e.g., [35]). This is because the lower and upper tails of a distribution are typically difficult to estimate. A good estimation (i.e., with low uncertainty) for the lower and upper quantiles typically implies a good estimation for the distribution. Further, x_q is centered so that $x_{0.1}$ and $x_{0.9}$ are far from zero.

To develop this idea, let $\widehat{x}_q$ be the estimator of the q quantile, x_q, of the throughput distribution. Let $\sqrt{\text{var}(\widehat{x}_q)}$ be the standard error for the estimator. To account for the scale of X, we consider the following ratio,

$$\Gamma_q = \frac{\sqrt{\text{var}(\widehat{x}_q)}}{\widehat{x}_q}, \tag{14}$$

as a metric for measurement of uncertainty, which is similar to those used in some statistical literature (e.g., [35]). We refer to Γ_q as the scaled standard error (SSE) for the q quantile estimator.

To estimate $\text{var}(\widehat{x}_q)$ under different sample sizes n, we choose a mixture model with k components as the underlying distribution for a system configuration. Then the PDF for the throughput distribution is

$$f(x, \widehat{\boldsymbol{\theta}}) = \sum_{j=1}^{k} \widehat{\pi}_j f_j(x, \widehat{\mu}_j, \widehat{\sigma}_j), \tag{15}$$

where $f_j(x, \mu_j, \sigma_j)$ is the PDF of a component distribution, with the same notation as in Section 3. The parameter vector $\widehat{\boldsymbol{\theta}} = (\pi_1, \ldots, \pi_{k-1}, \mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k)^T$ is estimated by the EM algorithm. The detailed formula for the calculation of $\text{var}(\widehat{x}_q)$ is given in Appendix A.1.

From (A.5), we can see that $\text{var}(\widehat{x}_q)$ is on the order of $1/n$ and $\Gamma_q = \Gamma_q(n)$ is a function of n. Thus, a larger n will lead to a smaller $\Gamma_q(n)$, which means higher confidence for the quantile x_q. The choice of $\Gamma_q(n)$'s threshold depends on the computing resources and accuracy requirement. A smaller threshold will lead to better accuracy while requiring more runs and vice versa.

The procedure we used to determine the optimal sample size for a given system configuration is illustrated in Figure 4 and described as follows.

1. For a given system configuration, run a pilot experiment with n runs to collect some initial data. A sample size of 30 to 40 can typically provide good estimates for the parameters.

2. Use the EM algorithm to fit the proposed 30 mixture models over the pilot data, then select the best fitted model among the 30 using BIC.

3. Use the best estimated mixture model, one can calculate $\Gamma_{0.1}(n)$ and $\Gamma_{0.9}(n)$ as a function of n by (A.5). Here we use 0.1 quantile (lower quantile of the distribution) and 0.9 quantile (upper quantile of the distribution) to control the accuracy in the estimation of the distribution. In a conservative situation, one can use the 0.05 quantile and 0.95 quantile to control the accuracy of the estimation.

4. Select a suitable sample size n such that $\Gamma_{0.1}(n)$ and $\Gamma_{0.9}(n)$ are both small enough. This choice depends on the application and the available resources. One can choose the threshold to be 0.5 (for fewer runs) or 0.1 (for better accuracy). Here a threshold of 0.5 means the standard error is of the 50% of the true value, while a threshold of 0.1 means the standard error is of the 10% of the true value.

Section 6 provides an illustration of this algorithm for sample size determination.

5. Modeling and Data Analysis Results

In this section, we provide the modeling and analysis results for the data presented in Section 2. We vary the possible number of components from 1 to 5. Because we use 6 candidate distributions for a single component, there are 5 (number of components) $\times$ 6 (kinds of distributions) = 30 mixture models in total. We fit all the 22734 HPC settings with these 30 mixture models and record their corresponding BIC values. All throughputs under one configuration are standardized before running the EM algorithm.

Regarding the computing time of the EM algorithms, Table 3 shows the mean computing time (in seconds) and its corresponding standard deviation (in parentheses) over 1000

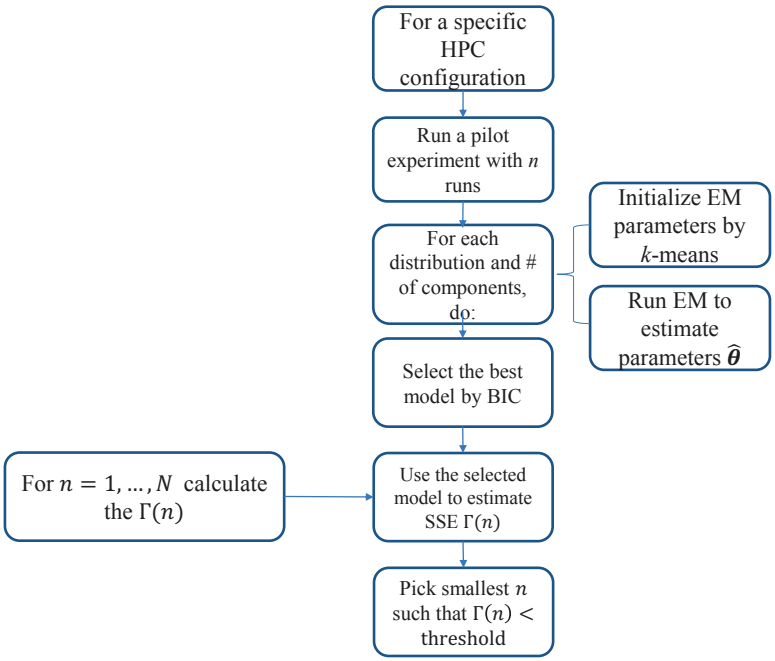

Figure 4: Flow chart of the statistical analysis framework.

randomly selected configurations, for different combinations of distribution components and the number of components. One can see that the computing time increases as the number of components increases. Note that for the normal and lognormal distributions, the computing is fast due to the availability of the closed-form expressions in updating. For other distributions, it takes longer due to the need for numerical optimization.

We first focus on which distributions have the best overall fits. The results are shown in Table 4. The second column of Table 4 (i.e., Sum of Best BIC), using a normal distribution as an example, is calculated as follows.

1. For the m-th setting (m is from 1 to M =22734), select the minimum BIC among all the normal mixture models with 1 to 5 components, denote as $\text{BIC}_m^{\text{norm}}$.

2. The sum of BIC for normal distributions is calculated as $\sum_{m=1}^{M} \text{BIC}_m^{\text{norm}}$.

The third column of Table 4 shows the number of times that each kind of component distribution is best through the whole dataset, while the last column shows the corresponding proportion. We see that the normal, lognormal, and loglogistic distributions have the lowest sum of BIC. The normal and lognormal distributions appear to be the most frequently selected component distributions. In particular, 50.00% of the 22734 configurations are selected to be either normal or lognormal mixture models. This reveals that the throughput of I/O systems, which in the past was thought to be normally distributed, follows not only

Table 3: The mean computing time (in seconds) and its standard deviation (in parentheses) for 1000 randomly selected configurations, for different distribution components and the number of components. "0.0" means the computing time is less than 0.01.

Distribution	Number of Components				
	1	2	3	4	5
normal	0.00 (0.00)	0.03 (0.05)	0.07 (0.08)	0.12 (0.17)	0.18 (0.21)
gamma	0.01 (0.01)	2.85 (2.59)	11.63 (12.41)	28.97 (26.92)	56.27 (43.79)
Weibull	0.00 (0.00)	4.27 (3.69)	13.92 (15.16)	33.43 (38.76)	61.84 (68.44)
lognormal	0.00 (0.00)	0.03 (0.11)	0.08 (0.12)	0.12 (0.13)	0.18 (0.18)
loglogistic	0.01 (0.00)	7.21 (6.46)	24.63 (24.89)	59.51 (61.81)	113.33 (114.27)
Frechét	0.00 (0.00)	4.52 (3.29)	13.70 (12.52)	32.55 (31.63)	67.36 (67.15)

normal but also lognormal distribution. In addition, compared to a normal distribution, a lognormal distribution is more right skewed. As a result, the previous assumptions about I/O variability ignore the right long tail, which is also seen in Figure 1.

When it comes to the number of components, the results are shown in Table 5. In particular, 70.2% of the 22734 configurations are detected to have a better fit for mixture models. Our result shows that multimodal behavior is common in I/O variability. Using a single component distribution does not accurately describe I/O variability. Also, the proportion of configurations that have more than three components is nonnegligible. In other words, it is not rare that I/O throughput distributions are complex, which makes the prediction and control of system variability challenging.

To visualize the fitting results, Figure 5 shows the mixture model fitting results for the same four configurations as seen in Figure 1. We plot the empirical CDF and estimated mixture distributions for the corresponding data. From the plot we see that mixture models can fit the data very well, under different settings. For those four configurations, we can see that four to five components are needed to describe the multimodal behavior.

6. Applications of Sample Size Determination

We construct Γ_q with $q = 0.1, 0.9$ using the historical data from one system configuration as an illustration. The system configuration information is given in Table 6. Using the data

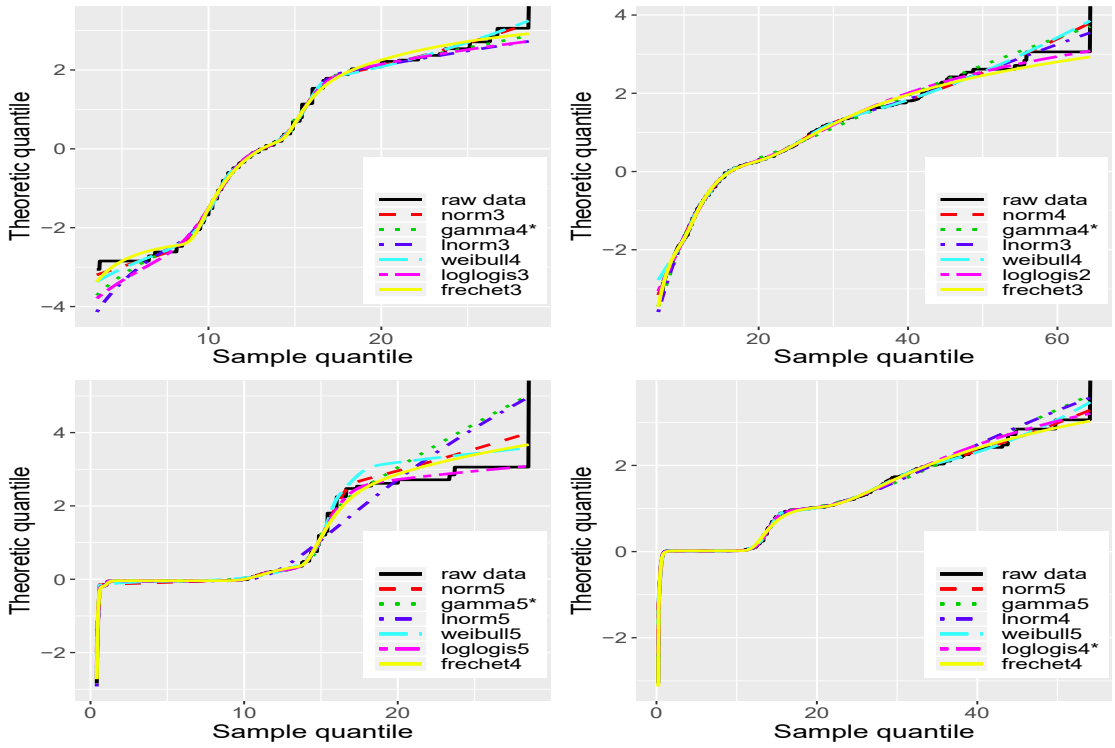

Figure 5: Four examples for probability plots corresponding to the data used in Figure 1. The best number of components for the six distribution types are plotted, and the black lines are the empirical CDF for the throughputs. The label "gamma4*" means a gamma mixture model with four components, while the "*" means the lowest BIC among all the mixture distributions.

Table 4: Sum of best BIC values for each setting of different distribution types and summary counts for best distributions for all 22734 configurations.

Distribution	Sum of Best BIC	Best Distributions	
		Counts	Proportion
normal	7066154	5378	0.237
gamma	7746079	1784	0.078
Weibull	8096009	2981	0.131
lognormal	7014309	5983	0.263
loglogistic	6857704	3339	0.147
Frechét	7160068	3269	0.144

Table 5: Summary for the best number of components selected for all 22734 configurations.

Summary	Number of Components				
	1	2	3	4	5
Count	6774	9000	2957	1945	2058
Proportion	0.298	0.396	0.130	0.086	0.091

analysis procedure in Section 5, we get a normal mixture model with two components for this configuration. The parameter estimates for this model are listed in Table 7.

Using the result in Table 7, we calculate $\Gamma_{0.1}(n), \Gamma_{0.9}(n)$ for n from 1 to 900 and plot Γ_q vs. n in Figure 6. The SSE convergence rate is in the order of $1/\sqrt{n}$. We see that although we collected data on 900 runs, less than 250 data points are needed to capture the characteristics of the mixture normal distribution in Table 7. At sample size $n = 250$, the total deviation of the two SSEs from zero (i.e., $|\Gamma_{0.1}(n)| + |\Gamma_{0.9}(n)|$), is about 6.32% of the starting deviation $|\Gamma_{0.1}(1)| + |\Gamma_{0.9}(1)|$ when the sample size is one. Recall that x_q was centered so that $\Gamma_q(n)$ can be negative.

So we can say that at $n = 250$ we can reduce the uncertainty by 93.38%. In addition, the rate of decrease in uncertainty slows with increasing n. So the marginal benefit of the increase in sample size vanishes. Overall, this gives us a useful way to balance the reduction of uncertainty with the number of runs needed. In this example, although we have done 900 runs, Figure 6 shows that 250 runs should provide comparable precision in estimation. Our

Table 6: System configuration used for the illustration of confidence interval construction.

Freq	fs	rs	Threads	Test	Replicates
3.5	4	4	1	initial_writers	900

Table 7: Parameter estimates under a normal mixture model with two components for the configuration in Table 6.

Proportion	Component 1		Component 2	
π_1	μ_1	σ_1	μ_2	σ_2
0.03977	1.6023	2.3462	−0.06634	0.8376

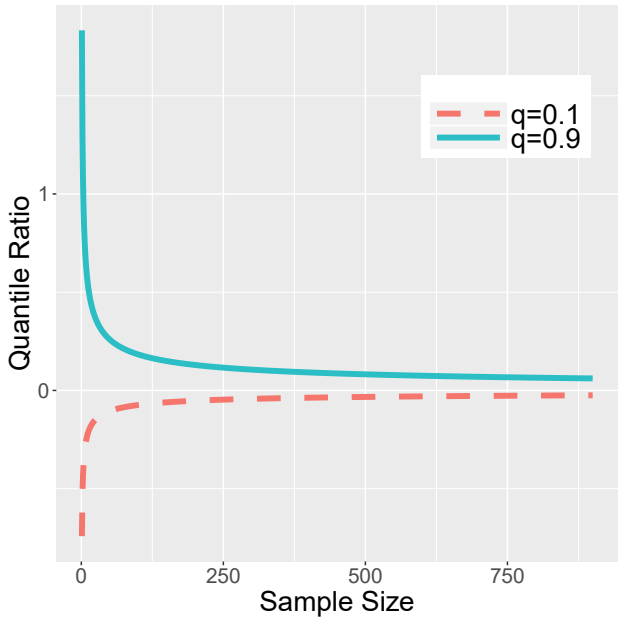

Figure 6: Plot for the quantile ratios $\Gamma_{0.1}$ and $\Gamma_{0.9}$ vs n.

tool can also be used to quantify uncertainty if one is limited to a relatively small number of runs. If one only has a budget of 40 runs for the configuration in Table 7, we can estimate the amount of uncertainty when $n = 40$ (i.e., as measured by $|\Gamma_{0.1}(40)| + |\Gamma_{0.9}(40)|$), which is about 15.8% of the starting deviation (i.e., $|\Gamma_{0.1}(1)| + |\Gamma_{0.9}(1)|$) according to Figure 6.

7. Conclusions and Areas for Future Research

In this paper, we focus on using parametric mixture models to describe the common multimodal behavior in the distribution of the I/O throughput from 22734 HPC system configurations. The data collected and our statistical analyses are generalizable to performance variability regardless of the specific deployment configuration. An EM algorithm is developed for parameter estimation using the Bayesian information criterion for model selection. The developed algorithm can automatically determine the number of components (up to five) and the underlying distribution (from a list of six) for the mixture model. The modeling and analysis results show that a throughput distribution can have very complex multimodal

behavior, which can not be well-modeled by a normal distribution. The proposed mixture models are flexible enough to accurately represent the complex throughput distributions and provide a handy tool to automatically identify those configurations with multimodal behavior. An important discovery is that right skewing is common in HPC variability, evidenced by the large proportion of lognormal distributions being selected as the best component distribution in Table 4. In other words, for many configurations, most throughput will be at a relatively low level but there will also be a substantial number of high throughputs. This provides insight into the variability control of HPC systems.

In this paper, we considered CPU clock speed, the number of threads, file size, record size, and six I/O operation modes as system factors. Due to the limit in time, we chose to focus on the impact of those five factors, and collected data from more than 22,000 configurations. Other hardware such as the memory and secondary storage, and I/O access patterns can also affect the I/O performance. However, the modeling framework introduced in this paper can be easily extended to more factors. For example, we observed in a small-scale experiment that different types of storage, such as SSD and harddisk drive (HDD), have similar multimodal behaviors. Branching out to other benchmarks and workloads is also something we are exploring in ongoing and future work.

One important future step in management of performance variability is to develop a general tool to predict the throughput distribution for a new system configuration. Being able to identify and describe performance variability is also an important step. Thus, in this paper, we focussed on modeling the common multimodal behavior of I/O throughputs. The mixture model can describe the distribution of throughputs well, although it is not designed for the prediction of the distribution of throughput for a new configuration. Some existing work can predict the variance in throughput based on machine learning and statistical methods [15, 16, 17, 18]. Using scalar measures like the variance to describe complex behavior, however, may be inadequate. For example, a throughput distribution with uni-modal may have similar variance to another distribution that is multimodal. However, predicting the distribution of throughput is a challenging task. Future effort will be devoted to develop a prediction tool that takes a system configuration as the input and provides a parametric distribution of throughput as output.

Another important future step is to understand the causes of variability. The current method can model variability well but one would not know what causes the variability. Being aware of this, we are now working on collecting data for "performance counter statistics" with the identification of those high variability configurations (i.e., those with multimodal behavior), which could provide finer details to find the cause of the variability. Our goal for this future work is to provide a systematic analysis of I/O performance variability to separate noise from variability that we can control. We intend to offer a crucial and initial

benchmark that underpins the current research. In the future, it is important to investigate other sources of variability such as garbage collection, congestion, or black-swan effects, and model those sources of variability.

Acknowledgments

The authors thank the editor, associate editor, and three referees, for their valuable comments that helped in improving the paper significantly. The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the statistical results reported within this paper. The work is supported by funds from NSF under Grant CNS-1565314 and CNS-1838271 to Virginia Tech.

Appendix A. Technical Details

Appendix A.1. Calculation of var($\widehat{x}_q$)

By the delta method (e.g., [35]), we have

$$\text{var}(\widehat{x}_q) = \left(\frac{\partial \widehat{x}_q}{\partial \widehat{\boldsymbol{\theta}}}\right)^T \text{Cov}(\widehat{\boldsymbol{\theta}}) \left(\frac{\partial \widehat{x}_q}{\partial \widehat{\boldsymbol{\theta}}}\right), \tag{A.1}$$

where $\text{Cov}(\widehat{\boldsymbol{\theta}})$ is the covariance matrix, and

$$\frac{\partial \widehat{x}_q}{\partial \widehat{\boldsymbol{\theta}}} = \left(\frac{\partial \widehat{x}_q}{\partial \pi_1}, \ldots, \frac{\partial \widehat{x}_q}{\partial \pi_{k-1}}, \frac{\partial \widehat{x}_q}{\partial \mu_1}, \ldots, \frac{\partial \widehat{x}_q}{\partial \mu_k}, \frac{\partial \widehat{x}_q}{\partial \sigma_1}, \ldots, \frac{\partial \widehat{x}_q}{\partial \sigma_k}\right)^T \Bigg|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}. \tag{A.2}$$

To obtain $\text{var}(\widehat{x}_q)$, we need to know $\text{Cov}(\widehat{\boldsymbol{\theta}})$ and $\partial \widehat{x}_q / \partial \widehat{\boldsymbol{\theta}}$. The calculation of $\partial \widehat{x}_q / \partial \widehat{\boldsymbol{\theta}}$ is given in Appendix A.2. To obtain $\text{Cov}(\widehat{\boldsymbol{\theta}})$, we use the inverse of the the Fisher information matrix. That is

$$\text{Cov}(\widehat{\boldsymbol{\theta}}) = \frac{1}{n}\left[I_1(\widehat{\boldsymbol{\theta}})\right]^{-1}. \tag{A.3}$$

Here,

$$I_1(\widehat{\boldsymbol{\theta}}) = -\mathbb{E}\left[\frac{\partial^2 f(X, \widehat{\boldsymbol{\theta}})}{\partial \widehat{\boldsymbol{\theta}} \partial \widehat{\boldsymbol{\theta}}^T}\right] \tag{A.4}$$

is a $(3k-1)\times(3k-1)$ matrix. With the above formulas, we can calculate $\Gamma_q(n)$ under each sample size n as

$$\Gamma_q(n) = \frac{\sqrt{\left(\frac{\partial \widehat{x}_q}{\partial \widehat{\boldsymbol{\theta}}}\right)^T \left[I_1(\widehat{\boldsymbol{\theta}})\right]^{-1} \left(\frac{\partial \widehat{x}_q}{\partial \widehat{\boldsymbol{\theta}}}\right)}}{\widehat{x}_q \sqrt{n}}. \tag{A.5}$$

Appendix A.2. Calculation of $\partial\widehat{x}_q/\partial\widehat{\boldsymbol{\theta}}$

To calculate $\partial\widehat{x}_q/\partial\widehat{\boldsymbol{\theta}}$, we use implicit function theory to derive the formula. The CDF for the mixture normal distribution with k components is

$$F(x,\widehat{\boldsymbol{\theta}}) = \sum_{j=1}^{k} \pi_j F_j(x,\mu_j,\sigma_j), \tag{A.6}$$

where $F_j(x,\mu_j,\sigma_j)$ is the CDF of one component with location parameter μ_j and scale parameter σ_j. Let $x = \widehat{x}_q$ be the q-th quantile of the mixture normal distribution, we have

$$F(\widehat{x}_q,\widehat{\boldsymbol{\theta}}) = q = \pi_j F_j(\widehat{x}_q,\mu_j,\sigma_j) + \sum_{l\neq j} \pi_l F_j(\widehat{x}_q,\mu_l,\sigma_l). \tag{A.7}$$

Take derivatives on π_j, we have

$$0 = F_j(\widehat{x}_q,\mu_j,\sigma_j) + \sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l)\frac{\partial\widehat{x}_q}{\partial\pi_j} \tag{A.8}$$

$$\frac{\partial\widehat{x}_q}{\partial\pi_j} = -\frac{F_j(\widehat{x}_q,\mu_j,\sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l)}. \tag{A.9}$$

For $\partial\widehat{x}_q/\partial\mu_j$, we have

$$0 = \pi_j\left(\frac{\partial\widehat{x}_q}{\partial\mu_j} - 1\right) f_j(\widehat{x}_q,\mu_j,\sigma_j) + \sum_{l\neq j} \pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l)\frac{\partial\widehat{x}_q}{\partial\mu_j} \tag{A.10}$$

$$\frac{\partial\widehat{x}_q}{\partial\mu_j} = \frac{\pi_j f_j(\widehat{x}_q,\mu_j,\sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l)}. \tag{A.11}$$

For $\partial\widehat{x}_q/\partial\sigma_j$, let $z_j = (\widehat{x}_q - \mu_j)/\sigma_j$. Then, we have

$$0 = \pi_j f_j(\widehat{x}_q,\mu_j,\sigma_j)\left(\frac{\partial\widehat{x}_q}{\partial\sigma_j} - z_j\right) + \sum_{l\neq j} \frac{\partial\widehat{x}_q}{\partial\sigma_j}\pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l) \tag{A.12}$$

$$\frac{\partial\widehat{x}_q}{\partial\sigma_j} = \frac{\pi_j z_j f_j(\widehat{x}_q,\mu_j,\sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q,\mu_l,\sigma_l)}. \tag{A.13}$$

Therefore, we obtain every element in $\partial\widehat{x}_q/\partial\widehat{\boldsymbol{\theta}}$.

For further illustrations, the following gives formulas for $\partial\widehat{x}_q/\partial\boldsymbol{\theta}$ on the normal mixture

model with two components,

$$\frac{\partial \widehat{x}_q}{\partial \mu_1} = \frac{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1)}{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \tag{A.14}$$

$$\frac{\partial \widehat{x}_q}{\partial \sigma_1} = \frac{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) \times \frac{\widehat{x}_q - \mu_1}{\sigma_1}}{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \tag{A.15}$$

$$\frac{\partial \widehat{x}_q}{\partial \mu_2} = \frac{(1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)}{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \tag{A.16}$$

$$\frac{\partial \widehat{x}_q}{\partial \sigma_2} = \frac{(1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2) \times \frac{\widehat{x}_q - \mu_2}{\sigma_2}}{\pi_1 f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi_1) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)}, \tag{A.17}$$

where $f_{\text{norm}}(x, \mu_j, \sigma_j)$, $j = 1, 2$ is the PDF of the normal distribution.

References

[1] K. W. Cameron, A. Anwar, Y. Cheng, L. Xu, B. Li, U. Ananth, J. Bernard, C. Jearls, T. Lux, Y. Hong, L. T. Watson, A. R. Butt, MOANA: Modeling and analyzing I/O variability in parallel system experimental design, IEEE Transactions on Parallel and Distributed Systems 30 (8) (2019) 1843–1856.

[2] Iozone filesystem benchmark.
URL `http://http://www.iozone.org/`

[3] W. T. Kramer, C. Ryan, Performance variability of highly parallel architectures, in: International Conference on Computational Science, Springer, 2003, pp. 560–569.

[4] R. Mraz, Reducing the variance of point to point transfers in the IBM 9076 parallel computer, in: Supercomputing '94: Proceedings of the 1994 ACM/IEEE Conference on Supercomputing, 1994, pp. 620–629.

[5] G. Shipman, P. McCormick, K. Pedretti, S. L. Olivier, K. B. Ferreira, R. Sankaran, S. Treichler, A. Aiken, M. Bauer, Analysis of application sensitivity to system performance variability in a dynamic task based runtime., Tech. rep., Sandia National Lab. (SNL-NM), Albuquerque, NM (United States) (2016).

[6] E. Grobelny, D. Bueno, I. Troxel, A. D. George, J. S. Vetter, Fase: A framework for scalable performance prediction of HPC systems and applications, Simulation 83 (10) (2007) 721–745.

[7] G. Wang, A. R. Butt, P. Pandey, K. Gupta, A simulation approach to evaluating design decisions in mapreduce setups, in: 2009 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009, pp. 1–11.

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

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

[10] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, J. C. Sancho, Using performance modeling to design large-scale systems, Computer 42 (11).

[11] K. Ye, X. Jiang, S. Chen, D. Huang, 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.

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

[13] P. De, R. Kothari, 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.

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

[15] T. C. Lux, L. T. Watson, T. H. Chang, J. Bernard, B. Li, X. Yu, L. Xu, G. Back, A. R. Butt, K. W. Cameron, Y. Hong, D. Yao, Nonparametric distribution models for predicting and managing computational performance variability, in: SoutheastCon 2018, IEEE, 2018, pp. 1–7.

[16] T. C. Lux, L. T. Watson, T. H. Chang, J. Bernard, B. Li, L. Xu, G. Back, A. R. Butt, K. W. Cameron, Y. Hong, Predictive modeling of I/O characteristics in high performance computing systems, in: Proceedings of the High Performance Computing Symposium, Society for Computer Simulation International, 2018, p. 8.

[17] T. C. Lux, L. T. Watson, T. H. Chang, J. Bernard, B. Li, X. Yu, L. Xu, G. Back, A. R. Butt, K. W. Cameron, D. Yao, Y. Hong, Novel meshes for multivariate interpolation and approximation, in: Proceedings of the ACMSE 2018 Conference, ACM, 2018, p. 13.

[18] T. H. Chang, L. T. Watson, T. C. Lux, J. Bernard, B. Li, L. Xu, G. Back, A. R. Butt, K. W. Cameron, Y. Hong, Predicting system performance by interpolation using a high-dimensional Delaunay triangulation, in: Proceedings of the High Performance Computing Symposium, Society for Computer Simulation International, 2018, p. 12.

[19] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, R. Ricci, Taming performance variability, in: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), USENIX Association, Carlsbad, CA, 2018, pp. 409–425.

[20] F. Fraternali, A. Bartolini, C. Cavazzoni, L. Benini, Quantifying the impact of variability and heterogeneity on the energy efficiency for a next-generation ultra-green supercomputer, IEEE Transactions on Parallel and Distributed Systems 29 (7) (2018) 1575–1588.

[21] M. Shafique, D. Gnad, S. Garg, J. Henkel, Variability-aware dark silicon management in on-chip many-core systems, in: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, EDA Consortium, 2015, pp. 387–392.

[22] V. Persico, P. Marchetta, A. Botta, A. Pescapé, On network throughput variability in Microsoft Azure cloud, in: Global Communications Conference (GLOBECOM), 2015 IEEE, IEEE, 2015, pp. 1–6.

[23] A. Anwar, Y. Cheng, A. R. Butt, Towards managing variability in the cloud, in: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, IEEE, 2016, pp. 1081–1084.

[24] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, N. J. Wright, Performance analysis of high performance computing applications on the Amazon web services cloud, in: Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, CLOUDCOM '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 159–168.

[25] M. Umar, S. V. Moore, J. S. Meredith, J. S. Vetter, K. W. Cameron, Aspen-based performance and energy modeling frameworks, Journal of Parallel and Distributed Computing 120 (2018) 222 – 236.

[26] T. B. Rolinger, T. A. Simon, C. D. Krieger, Performance considerations for scalable parallel tensor decomposition, Journal of Parallel and Distributed Computing 129 (2019) 83–98.

[27] Z. Tong, S. Pakin, M. Lang, X. Yuan, Fast classification of MPI applications using lamports logical clocks, Journal of Parallel and Distributed Computing 120 (2018) 77 – 88.

[28] L. Yu, Z. Zhou, S. Wallace, M. E. Papka, Z. Lan, Quantitative modeling of power performance tradeoffs on extreme scale systems, Journal of Parallel and Distributed Computing 84 (2015) 1 – 14.

[29] J. Lemeire, B. da Silva, A. Braeken, J. G. Cornelis, A. Touhafi, Efficiency analysis methodology of FPGAs based on lost frequencies, area and cycles, Journal of Parallel and Distributed Computing 113 (2018) 204 – 217.

[30] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society. Series B (methodological) 39 (1977) 1–38.

[31] R. Sundberg, Maximum likelihood theory for incomplete data from an exponential family, Scandinavian Journal of Statistics 1 (1974) 49–58.

[32] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Selected papers of Hirotugu Akaike, Springer, 1998, pp. 199–213.

[33] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (2) (1978) 461–464.

[34] J. Neyman, Outline of a theory of statistical estimation based on the classical theory of probability, Philosophical Transactions of the Royal Society of London. Series A 236 (767) (1937) 333–380.

[35] Y. Hong, W. Q. Meeker, Confidence interval procedures for system reliability and applications to competing risks models, Lifetime Data Analysis 20 (2) (2014) 161–184.