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]

[a] *Department of Statistics, Virginia Tech, Virginia 24061, USA*
[b] *Department of Computer Science, Virginia Tech, Blacksburg, Virginia 24061, USA*

Abstract

Performance variability is an important dimension of high-performance computing (HPC) systems. HPC performance variability is often complicated because it is affected by complicated interactions of numerous factors. 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 variability, which is measured by the variability of I/O throughputs that vary from run to run. For a given system configuration, the distribution of throughputs from run to run is of interest. We conduct large-scale experiments and collect a mass amount of data to study the distribution of variability under tens of thousands of system configurations. Despite normality is often assumed in literature, our statistical analysis reveals that the performance variability is not normally distributed under most system configurations. Instead, multimodal distributions are more common for many system configurations~~, which is new to literature~~. We propose to use mixture distributions to describe the multimodal behavior. Various underlying parametric distributions such as normal, gamma, and the Weibull are considered. We apply an EM-based 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 needed for future experiments for variability analysis. The results shed light on ~~understanding~~ the behavior of performance variability under different system configurations.

Keywords: Performance analysis, mixture model, uncertainty qualification, 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) system needs to scale up so that the exponential increases in computing complexity and scale can be addressed. One hindering factor in HPC scale-up is the performance variability. HPC system variability has several forms. One common example is that given a computing task running in identical system for several times, the running time for different replicates could be different. In some scenarios, the difference in running time is non-ignorable and it is hard to predict due to randomness. Observations and experiments have shown that variability in HPC system exists and could significantly affect performance. Variability makes system optimization challenging. Thus, variability control is an important problem in HPC research.

One important step in variability control is to have a good description of the system variability. The description of system variability involves the following two steps, namely, running proper experiments in HPC system to collect enough data, and applying appropriate statistical tools to analyze the experimental data and capture the complex distribution patterns under different system configurations. Little comprehensive work has done ~~in~~ describing the HPC system variability. One possible reason could be due to the lack of resource in running enough experiments in HPC system to collect sufficient experiment data for statistical analysis. In addition, due to the complexity in computing system, both in terms of hardware and applications, the source of variability is not easy to identify. In most existing work, the distribution of the variability of an HPC system is assumed to follow a normal or a single mode distribution.

Our research finds that the distribution of the variability is much more complicated than a normal distribution. In the throughput data collected from the IOzone benchmark [1], multi-mode distributions for performance variability are common for many system configurations. Although the details will be discussed in Section 2, Figure 2 shows examples of IO throughput distributions under four different configurations. In particular, we collected data from hundreds of runs of IOzone ~~test~~ for the four configurations. We plot the histograms of throughputs for each configuration. All ~~the~~ four histograms do not show a clue for normal distribution. Thus, the distribution of IO 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 conduct analysis for large scale data collected from IOzone benchmark testing from an HPC system. Mixture distributions are used to model the throughput data collected from IOZone benchmark ~~testing~~. We develop an expectation-maximization (EM) algorithm to automatically identify the distribution param-

eters, and use the Bayesian information criterion (BIC) to select the underlying distribution and the number of components in the mixture distribution. The result of the data analysis implies that the usually-made normality assumption for the system performance variability is not accurate. Multi-mode phenomenon show up in the majority of the configurations. That is, the distribution of the throughput follows a multi-mode mixture distribution for most of the cases. The parametric mixture distribution not only provides automatic tool to discover the distributional behaviors of the performance variability, but also provides a basis for computing the sample size (i.e., the number of runs) for future experiments. We ~~also~~ develop a procedure for sample size using the estimated parametric mixture distribution, which may have benefit for future research in controlling variability.

1.2. Related Literature

Researchers have noticed the existence of variability in HPC performance for more than a decade [2, 3]. Performance variability was analyzed in some existing work [4, 5, 6]. Prior to this paper, statistical modeling of variability was restricted to one or two system parameters [7, 8, 9, 10]. 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 [11, 12]. Yet, system I/O variability has been particularly difficult for statistical models to capture [5, 8]. I/O variability under two system parameters was modeled in [13]. Recent generic I/O modeling has been nonparametric, and hence limits the number of conclusions that can be drawn from the model structure [14]. 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 [15]. Maricq et al [16] propose an open source data and tools 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 experiment trials you need to capture the system variability behavior.

In application, variability is becoming an important factor for building super computers [22, 23], cloud computing systems [24, 25, 26] and robust scalable network security [14]. 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. [17], they propose two energy models and estimation frameworks for Aspen DSL. Using experimental data, their model can reduce at least 45% energy consumption. Rolinger et al. [18] presents a performance analysis framework for tensor decomposition. Their study concentrates on

measure memory usage, processor stall cycles, execution time and scalability. Tong et al. [19] presents a trace-based analysis tool that can classify MPI applications as different source of bottom-neck based on their performance. The classification model can help people better understand application performance limiting factors. In addition to control performance, managing energy consumption is also a hot area [20, 21]. Yu et al. [20] focus on the trade off between power consumption and performance. They use a trace based validation and show that the model is accurate and scalable. [21] gives a methodology to study the computing efficiency and impact of overheads on runtime performance. They define three types of efficiency and they propose analytical formulas to measure and predict the respective efficiencies.

For the related statistical work, mixture distributions are widely used to describe distributions with multi-modal behaviors, and the EM algorithm is usually used to find the maximum likelihood estimates of parameters in mixture distribution [27, 28]. For parametric model selection, many statistical criteria are based on maximum likelihood criterion [29]. An obvious drawback of maximum likelihood criterion is that it has no penalty on the number of parameters in the model. As a result, the best model are always the one which has the most parameters which has the risk of over fitting. Many extensions of maximum likelihood are based on the idea that add one penalty to the number of parameters. Two examples are Akaike information criterion (AIC) which is firstly discussed in [30] and Bayesian information criterion (BIC) which is discussed in [31]. To estimate the sample size and quantify the uncertainty in estimation, the confidence interval can be used and early work can be traced back to [32]. In this paper, we use an approach that is similar to [33] to measure the uncertainty in estimating the quantile of the distribution, and we show to sample size calculation.

In summary, our proposed parametric analysis using mixture distribution for I/O variability modeling and analysis is new to literature. The mixture distribution provide an effective tool to describe the multi-modal behavior in the distribution of I/O performance. The model also be useful for designing future experiments for I/O variability study.

1.3. Overview

The rest of the paper is organized as follows. Section 2 gives a description to the IOzone experiment setup and the collected dataset. Section 3 describes the statistical model using mixture distributions and an EM algorithm for parameter estimation and parametric model selection. 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 the discussion of the findings. Section 6 gives a numerical example for sample size determination. Section 7 discusses the conclusion and areas for future research. Some technical details are

available in the appendix.

2. Experiment Setup and Data Collection

The dataset we used in 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) has a constrain that their ratio fs/rs must be an integer.

Totally, we have 22,734 different settings. The matrix plot in Figure 1 shows the combinations of all four numerical variables in the experiments. It describes how frequency, threads, file size and record size are combined. The diagonal plots show the name of the axis. 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. We see that frequency and threads spread evenly in Euclidean space while file size and record size spread in the power of two.

The output 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 [1] for more than 150 times to capture the characteristic of the throughput distribution.

To illustrate some general patterns in the throughput distribution. We select four settings and plot the histograms of the throughputs in Figure 2. From those histograms, we can see that the distribution of throughputs have a clear pattern of multiple modes. The top two histograms show the distribution comes from mixture models with two components. While the bottom two are hard to identify how many components they have by eyeballing. Most of their throughputs stand at low level but there are still non-ignorable high throughput parts. All of them cannot be described using a single normal distribution, indicating 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 assume the throughput under a given configuration can be represented by a random variable. In addition, the throughput of each run under same configuration can be treated as independent and identically distributed (iid). Let $\boldsymbol{X} = (X_1, \cdots, X_n)$ be an n-element iid random variables and each element represents the throughput of one run under same configuration,

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

	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

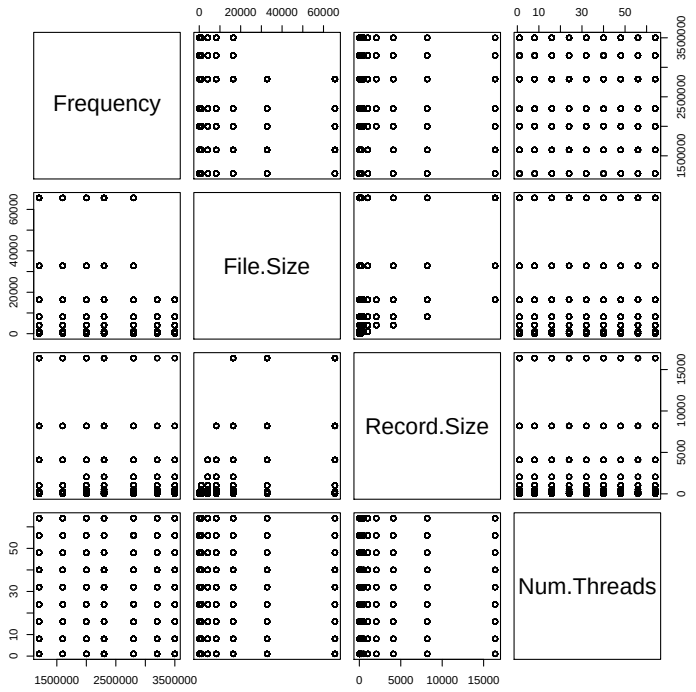

Figure 1: Scatter plot for for all four numeric settings.

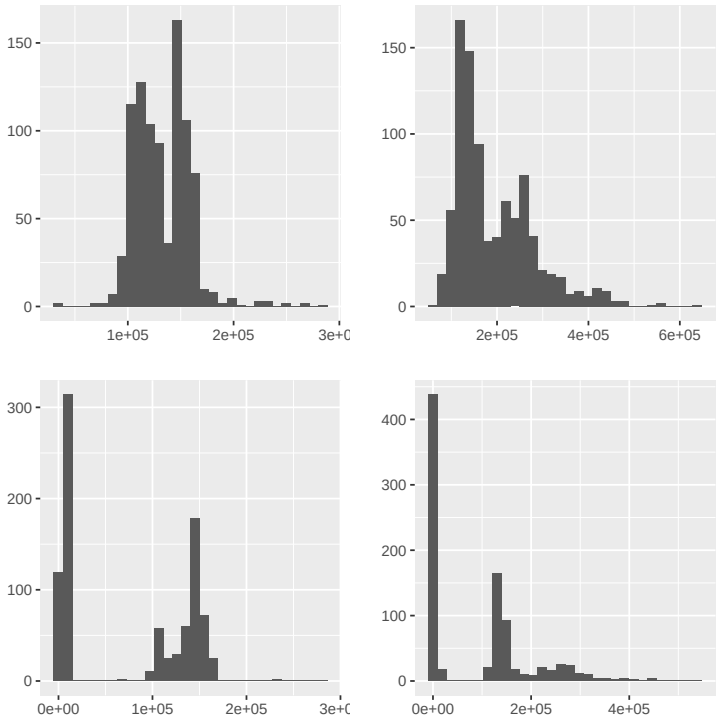

Figure 2: Histograms for 4 example settings. X-axis is the throughput (KB/s), Y-axis is the counts.

where n is the number of runs under a given configuration. We assume X_i has a k-component mixture distribution with probably density function (pdf),

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

where π_j is the proportion of 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). Let $\boldsymbol{\theta} = (\pi_1, \boldsymbol{\theta}_1, \cdots, \pi_k, \boldsymbol{\theta}_k)$ be the vector of parameters, where $\boldsymbol{\theta}_j$ is the parameter for the jth component. Note that $\sum_{j=1}^{k} \pi_j = 1$.

For the model in (1), each component represents a mode in the throughput distribution. To illustrate the flexibility of the mixture model in describing multi-modal data, we simulate some data a normal mixture model (i.e., using normal distribution as the underlying component distribution in (1)) with number of components ranging from 2 to 5. The simulated data are shown in Figure 3, which shows similar patterns as those in Figure 2 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 much accurate description of throughput data than only using a common normal distribution.

To use the mixture model in (1), one needs to specify the number of components k, and the underlying distribution $f_j(x, \boldsymbol{\theta}_j)$. The specification of k will be discussed in Section 3.3.

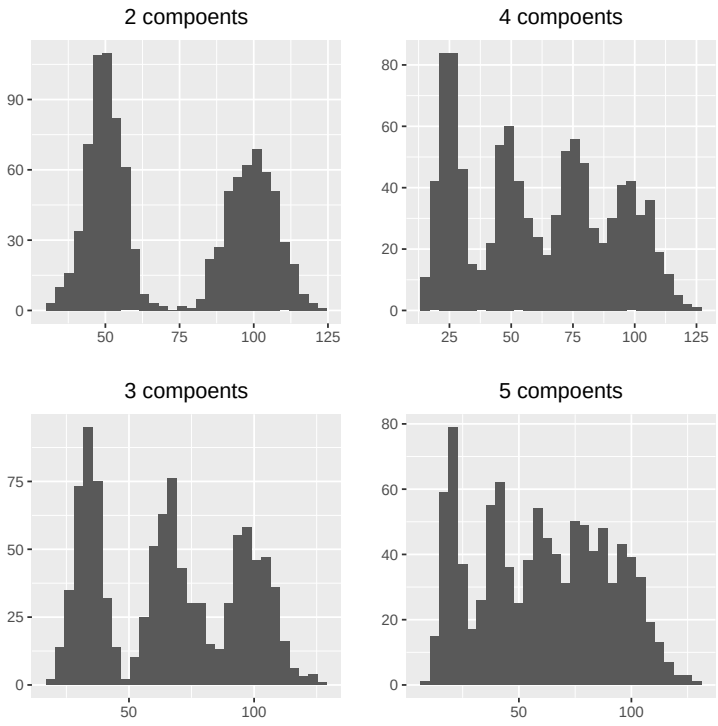

Figure 3: Histograms of random samples from from mixture normal distribution with components form 2 to 5.

For the underlying distribution, the commonly-used distributions such as normal, gamma, the Weibull and lognormal distribution can be used. In particular, the pdfs of the normal, gamma, the Weibull, and lognormal distributions are,

$$f(x,\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}}\exp\left[-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right], \quad f(x,\mu_j,\sigma_j) = \frac{1}{\Gamma(\mu_j)\sigma_j^{\mu_j}}x^{\mu_j-1}\exp(-\frac{x}{\sigma_j}),$$

$$f(x,\mu_j,\sigma_j) = \frac{1}{\sigma_j x}\phi\left[\frac{\log(x)-\mu_j}{\sigma_j}\right], \text{ and } \quad f(x,\mu_j,\sigma_j) = \frac{1}{x\sqrt{2\pi\sigma_j^2}}\exp\left[-\frac{(\log x-\mu_j)^2}{2\sigma_j^2}\right],$$

respectively. Here, $\phi(z) = \exp[z-\exp(z)]$. Let $\boldsymbol{\theta}_j = (\mu,\sigma)$. For normal, lognormal and the Weibull distribution, μ_j is location parameter and σ_j is scale parameter. For gamma distribution, μ_j refers to the shape parameter and σ_j refers to the scale parameter. Thus the parameter to estimated for the mixture model is $\boldsymbol{\theta} = (\pi_1,\cdots,\pi_{k-1},\mu_1,\cdots,\mu_k,\sigma_1,\cdots,\sigma_k)^T$.

3.2. The EM Algorithm for Parameter Estimation

With respect to the estimation of the parameters of mixture model, we use the EM algorithm for estimation. The ~~the~~ EM algorithm uses an iterative approach to find the maximum likelihood estimates (MLE) of parameters. For the model in (1), we add a latent random variable $\boldsymbol{Z} = (Z_1,\cdots,Z_n)$ to indicate which component the X_i is sampled from,

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

Then the likelihood for $\boldsymbol{X},\boldsymbol{Z}$ is,

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

where

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

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

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

The EM algorithm can estimate the maximum likelihood estimates of $\boldsymbol{\theta}$ without knowing the latent variable $\boldsymbol{Z}$. With an initial values for $\boldsymbol{\theta}$, the EM algorithm alternatives between the E-step and the M-step, which will be introduced in following sections.

3.2.1. Choosing Initial Values

First, we need to determine a starting value of the parameter,

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

for the EM algorithm. We use the k-means clustering technique to divide our sample $x_j, j = 1, \cdots, n$ into k groups. Let $n_j, j = 1, \cdots, n$ be the number of points in each group, and we have $\sum_{j=1}^{k} n_j = n$. Then we obtain $\pi_j^{(0)} = n_j/n$. For the initial values $\mu_j^{(0)}$ and $\sigma_j^{(0)}$, suppose the j-th group has data points $\{x_{j1}, \cdots, x_{jn_j}\}$. Then $\mu_j^{(0)}$ and $\sigma_j^{(0)}$ are chosen to be the maximum likelihood estimators (MLE) using these 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 $\{x_{j1}, \cdots, x_{jn_j}\}$.

3.2.2. E-step

For the E-step, we calculate the expected loglikelihood given the current estimate $\boldsymbol{\theta}^{(t)}$. That is to calculate the following quanity,

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

3.2.3. M-step

For the M-step, we obtain an update of $\boldsymbol{\theta}$, denoted by $\boldsymbol{\theta}^{(t+1)}$, by obtain the value of $\boldsymbol{\theta}$ that maximizes $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$. That is,

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

The update for π_j in $\boldsymbol{\theta}$ has a closed-form expression. Let

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

Then, π_j is updated as,

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

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

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

For lognormal mixture model, the updating formula is,

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

In the Weibull, and gamma mixture model, we do not have close forms for updating location and scale parameters. We use gradient descent method to update them. We repeat the E-step and M-step until the convergence of $\boldsymbol{\theta}^{(t)}$ reaches.

3.3. Model Selection

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

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

where n is the number of iid samples and p is the number of parameters in the model. Also,

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

is maximized likelihood value with $\widehat{\pi}_j$, and $\widehat{\boldsymbol{\theta}}_j$ to be the parameter estimated by the EM algorithm.

For example, if the number of components in a normal mixture model is k, then $p = 3k-1$. That is, there are k location parameters, k scale parameters, and $k-1$ parameters for the π_j because π_j has the sum-to-one constraint.

4. Sample Size Determination

One application of the mixture model is to use it to determine the number of runs 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 a easy way to display the scale of uncertainty. A narrow confidence interval often means a smaller uncertainty of the parameter. The width of confidence interval is related to the standard error, which typically reduces as the number of runs (i.e., the sample size n) increases.

Let X be the random variable for the throughput under a given system configuration. The cumulative distribution function of X is defined as $F(x) = \Pr(X \leq x)$. The q quantile of x_q is defined as x such that $F(x) = q$ for continuous distribution. 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 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 the distribution function (e.g., [33]). This is because the lower and upper tails of a distribution are typically difficult to estimate. If one has a good estimation (i.e., with less uncertainty) for the lower and upper quantiles, it typically provides good estimation for the distribution.

To develop this idea, let $\widehat{x}_q$ be the estimated q quantile 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,

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

as a metric for measurement of uncertainty, which is similar to those used in some statistical literature (e.g., [33]). We refer Γ_q as the scaled standard error (SSE) for the q quantile estimator.

To estimate $\text{var}(\widehat{x}_q)$ in different sample size n, we choose a mixture model with k components as the underlying distribution of an HPC setting. Then the pdf for the throughput is,

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

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

$$\widehat{\boldsymbol{\theta}} = (\pi_1, \cdots, \pi_{k-1}, \mu_1, \cdots, \mu_k, \sigma_1, \cdots, \sigma_k)^T$$

is estimated by EM algorithm. By the delta method (e.g., [33]), we have,

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

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

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

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 the appendix. To obtain $\text{Cov}(\widehat{\boldsymbol{\theta}})$, we use the result of the Fisher information matrix,

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

Here,

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

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

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

The procedure we used to determine sample size can be described as follows for a give system configuration,

1. For a given system configuration, run a pilot experiment to collect some intimal data. One can run a relative small n such as $n = 40$.
2. Use the EM algorithm to fit the mixture model and estimate $\widehat{\boldsymbol{\theta}}$ using the pilot data.
3. Use the estimated mixture model, one can calculate $\Gamma_{0.1}(n)$ and $\Gamma_{0.9}(n)$ as a function n.
4. Select a suitable sample size n such that $\Gamma_{0.1}(n)$ and $\Gamma_{0.9}(n)$ are both small enough. Depends on the application and the available resource, one can choose the threshold to be .5 or 0.1.

Section 6 provides an illustration for this algorithm for sample size determination. We summarize our analysis framework in a flow chart in Figure 4.

5. Modeling and Data Analysis Results

In this section, we provide the modeling and analysis results for the data presented in Section 2. We set the possible number of components from 1 to 5. Since we have 4 kinds of distributions for a single component, there are total 5 (number of components) $\times$ 4 (kind of distribution) mixture models. We fit all the 22734 HPC settings with these 20 mixture models and record their BIC. All throughputs under one configuration are standardized before fitting the EM algorithm.

Firstly, we focus on which kind of distribution has the best overall fittings. The results are shown in Table 2. For the second row of Table 2, we use normal distribution as example to describe how the sum of BIC is calculated,

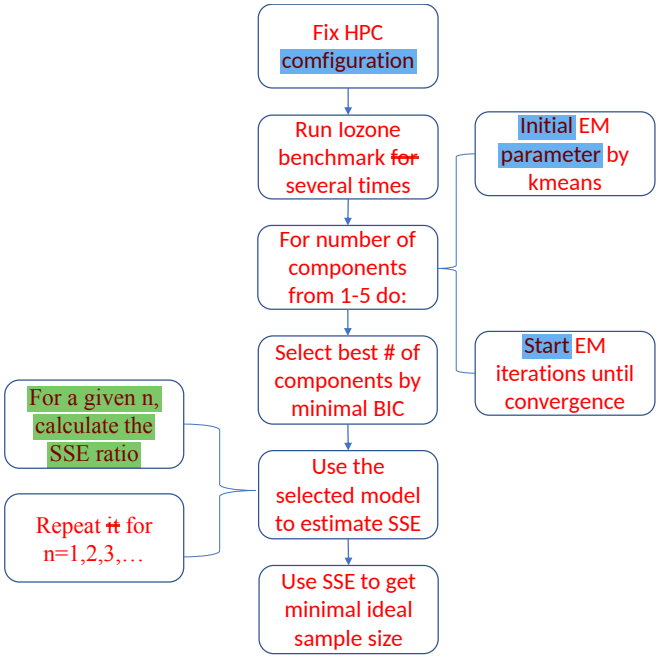

Figure 4: Flow chart of the statistical analysis framework.

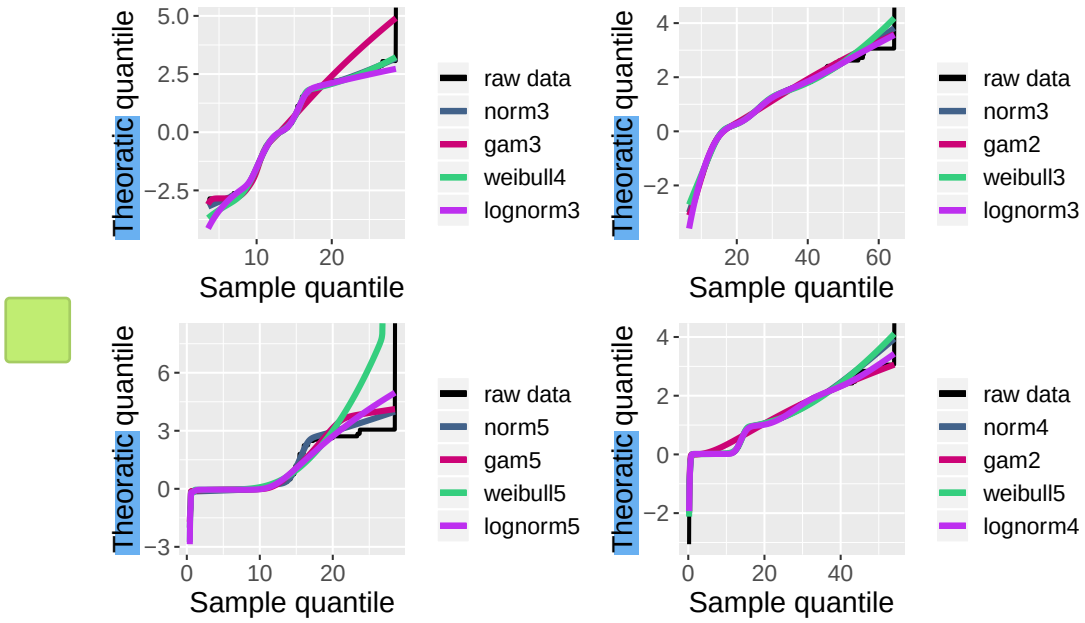

Figure 5: Four examples for probability plot corresponding to the data used in Figure 2. The best number of components of 4 distribution is plotted. The normal distribution has the best fitting. Black line are the raw throughputs and "norm3" means normal mixture model with 3 components.

Table 2: Table of sum of BIC values for each setting of 4 distributions.

Distribution	Sum of best BIC	Count of best distributions
Normal	7104395	7027
Gamma	7777854	3680
Weibull	8132240	3674
Lognormal	7103900	8353

1. For the m-th setting (m is from 1 to M =22734), select the minimal BIC among all the normal mixture model with 1 to 5 components, noted the selected minimal BIC for m-th setting as $\text{BIC}_m^{\text{norm}}$.
2. The sum of BIC for normal distribution is calculated as $\sum_{m=1}^{M} \text{BIC}_m^{\text{norm}}$

The third row of Table 2 shows the counts that each kind of component distribution are selected through the whole dataset. We see that normal distribution and log-normal distribution have the least sum of BIC and appear frequentest. 67.65% of the 22734 configurations are selected to be normal or lognormal mixture model. This reveals that the throughput of ~~an~~ HPC system which in the past are thought to be normally distributed, follows not only normal but also log-normal distribution. In addition, compared to normal distribution, lognormal distribution is more left skewed. As result, the previous assumption of the HPC variability ignores the right long tail. This verifies the bottom histograms in Figure 2.

When it comes to the number of components, the results are shown in Table 3. 67.88% of the 22734 configurations are detected to have better fit for mixture models. Our result shows that multi-mode behavior are common in HPC variability. Using single component distribution is not accurate to describe HPC variability. Also, the proportion of configuration which have more than 3 components is non-ignorable. In other word, it is not rare that HPC throughput~~'s~~ distribution will be complex, which makes the prediction and control of system variability very challenging.

To visualize some specific fitting results, we show the mixture model fitting results for four configurations as shown in Figure 2. We plot the probability plots of empirical probability density function and estimated mixture distributions for corresponding data in Figure 2. From the plot we see that normal, gamma and lognormal mixture model can fit the data very well. The Weibull model fails at the bottom left plot and succeed in other three. One reason for this can be that the throughputs of the bottom left plot has the maximum value quite greater than rest of the data and the Weibull model cannot capture this pattern well.

Table 3: Table of best number of components selected.

Num of Components	1	2	3	4	5
Count	7302	9811	2559	1500	1562

Table 4: Table for the setting of the data used in CI construction.

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

6. Applications on Sample Size Determination

We apply the construction of SSE Γ_q with $q = 0.1, 0.9$ using the historical data from the system configuration as shown in Table 4. Similar to the data analysis in Section 5, we result in a normal mixture model with 2 components. The parameter estimates for this model are listed in Table 5.

Using the result in Table 5, we calculate the $\Gamma_{0.1}(n), \Gamma_{0.9}(n)$ for n from 1 to 900 and plot Γ_q vs n in Figure 6. The SSE convergence rate is in the order of $\sqrt{n}$. We see that although we got 900 replicates, less than 250 points is need to capture the characteristic of the mixture normal distribution in Table 5. At sample size $n = 250$, the total deviation of the two SSE 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. Note that we standardize the throughput data in the estimation so it is possible that Γ_q could be negative.

So we can say that at $n = 250$ we can reduce the uncertainty by 93.33%. In addition, when n is larger, the decrease ~~of reduction~~ of uncertainty is slower and slower. So the marginal benefit of the increase of sample size vanishes. Overall, this gives us a useful to balance the reduction of uncertainty and number of runs needed. In this example, although we have done 900 runs, from Figure 6, we see that 250 runs should provide comparable precision in estimation.

Table 5: Table for parameters estimated by EM algorithm.

π	μ_1	σ_1	μ_2	σ_2
0.03977	1.6023	2.3462	−0.06634	0.8376

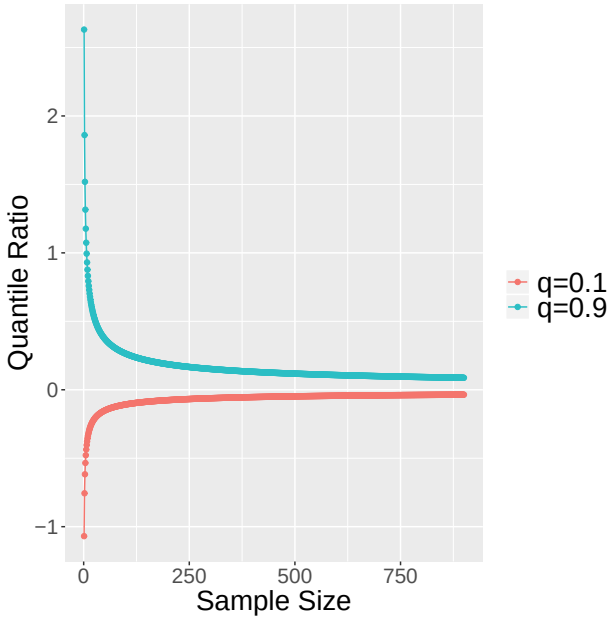

Figure 6: Plot for the quantile ratio $\Gamma_{0.1}$ and $\Gamma_{0.9}$ vs n.

7. Conclusions and Areas for Future Research

In this paper, we propose to use parametric mixture model to model the IO throughput data and perform EM algorithm analyze the data from 22734 system configurations. We use the BIC as a model criteria in model selection. The results show that the throughput distribution can have very complex behavior and using single normal distribution to describe its regular pattern could be misleading. The mixture model gives us a promising method to model the distribution of ~~the~~ HPC variability because compared to the single parametric component model, mixture model is more flexible and better at describing sophisticated patterns.

We also discover that left skewing commonly exists in HPC variability by the large proportion of lognormal distribution being selected as the best ~~mixture model~~ in Table 2. In other word, under a given configuration, most throughput will lay in a relative low level but we will also have non-ignorable part in very high level. This gives ~~us~~ insight in controlling and scaling ~~the~~ HPC systems.

Our paper mainly focus on summarizing the HPC configurations. Some possible future works could be transferring the summarizing the IOzone data to a prediction tools. Currently, our model have no prediction power given a fixed configuration. We will develop predicting method which uses the system configurations as input and the distribution of throughput as output. In addition, we will also build tools which can gives the optimal sample size given the configuration.

Acknowledgments

The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper.

Appendix A. Technical Details

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,

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

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,

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

Take derivatives on π_j, we have,

$$\begin{aligned} 0 =& F_j(\widehat{x}_q, \mu_j, \sigma_j) + \sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l) \frac{\partial \widehat{x}_q}{\partial \pi_j} \\ \frac{\partial \widehat{x}_q}{\partial \pi_j} =& - \frac{F_j(\widehat{x}_q, \mu_j, \sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l)} \end{aligned}$$

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

$$\begin{aligned} 0 =& \pi_j \left(\frac{\partial \widehat{x}_q}{\partial \mu_j} - 1 \right) f_j(\widehat{x}_q, \mu_j, \sigma_j) + \sum_{l \neq j} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l) \frac{\partial \widehat{x}_q}{\partial \mu_j} \\ \frac{\partial \widehat{x}_q}{\partial \mu_j} =& \frac{\pi_j f_j(\widehat{x}_q, \mu_j, \sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l)} \end{aligned}$$

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

$$\begin{aligned} 0 =& \pi_j f_j(\widehat{x}_q, \mu_j, \sigma_j) \left(\frac{\partial \widehat{x}_q}{\partial \sigma_j} - z_j \right) + \sum_{l \neq j} \frac{\partial \widehat{x}_q}{\partial \sigma_j} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l) \\ \frac{\partial \widehat{x}_q}{\partial \sigma_j} =& \frac{\pi_j z_j f_j(\widehat{x}_q, \mu_j, \sigma_j)}{\sum_{l=1}^{k} \pi_l f_j(\widehat{x}_q, \mu_l, \sigma_l)} \end{aligned}$$

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,

$$\begin{aligned}
\frac{\partial \widehat{x}_q}{\partial \mu_1} &= \frac{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1)}{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \\
\frac{\partial \widehat{x}_q}{\partial \sigma_1} &= \frac{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) \times \frac{\widehat{x}_q - \mu_1}{\sigma_1}}{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \\
\frac{\partial \widehat{x}_q}{\partial \mu_2} &= \frac{(1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)}{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)} \\
\frac{\partial \widehat{x}_q}{\partial \sigma_2} &= \frac{(1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2) \times \frac{\widehat{x}_q - \mu_2}{\sigma_2}}{\pi f_{\text{norm}}(\widehat{x}_q, \mu_1, \sigma_1) + (1-\pi) f_{\text{norm}}(\widehat{x}_q, \mu_2, \sigma_2)}
\end{aligned}$$

References

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

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

[3] 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.

[4] 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).

[5] 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.

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

[7] 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.

[8] 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.

[9] 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).

[10] 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.

[11] 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.

[12] 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.

[13] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, 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.

[14] 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, et al., Nonparametric distribution models for predicting and managing computational performance variability, in: SoutheastCon 2018, IEEE, 2018, pp. 1–7.

[15] T. C. Lux, L. T. Watson, T. H. Chang, 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.

[16] 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.

[17] 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.

[18] T. B. Rolinger, T. A. Simon, C. D. Krieger, Performance considerations for scalable parallel tensor decomposition, Journal of Parallel and Distributed Computing.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] S. Canon, N. J. Wright, K. Muriki, L. Ramakrishnan, J. Shalf, H. J. Wasserman, S. Cholia, K. R. Jackson, Performance analysis of high performance computing applications on the amazon web services cloud, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science(CLOUDCOM), Vol. 00, 2010, pp. 159–168.

[27] 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) (1977) 1–38.

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

[29] F. Y. Edgeworth, On the probable errors of frequency-constants (contd.), Journal of the Royal Statistical Society 71 (3) (1908) 499–512.

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

[31] G. Schwarz, et al., Estimating the dimension of a model, The annals of statistics 6 (2) (1978) 461–464.

[32] J. Neyman, Xoutline of a theory of statistical estimation based on the classical theory of probability, Phil. Trans. R. Soc. Lond. A 236 (767) (1937) 333–380.

[33] 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.