Going beyond mean and median programs performances

Julien Worms, Sid Touati

Going beyond mean and median programs performances

Julien WORMS
Université Versailles Saint-Quentin en Yvelines
Laboratoire de mathématiques de Versailles, France

Sid TOUATI
Université Côte d'Azur
I3S, INRIA-Sophia, France

***Abstract*—In the area of code performance optimisation and tuning, we are faced on the difficult problem of selecting the "best" code version based on empirical experiments and statistical analysis. With the massive introduction of general purpose multicore processors, programs performances become more and more instable, especially parallel programs. Usual statistical methods for computing performance speedups and comparing between programs are based on testing mean or median values. In this article, we explain why these metrics may be inadequate for making relevent decisions, and we propose new performance metrics based on parametric statistics using gaussian mixture models. Our new statistical methods are more accurate for decision making, they are formally defined, computed, implemented and distributed as free software in [1].**

***Keywords*-programs performances modelling, gaussians mixture, programs performances variability, parametric statistics.**

I. INTRODUCTION AND MOTIVATION

When you read books or articles on computer architectures, performance analysis, operating systems (OS) or compilation, some people still think that the execution time of a program P on a fixed machine M with fixed data input I is stable around a value, that can be noted as a single number `ExecutionTime(P,I,M)`. This situation was true in old computers, but nowadays nobody really observes constant execution times, except in rare situations: ideal execution environment, special processor architectures designed for performance stability, bare mode execution, etc. Even SPEC organisation resumes the execution times of a program as a single number (computed as the sample median of a set of observed values).

The consequence of performance variability is that the reported constant numbers of program performances in the literature are not reproducible, and it becomes more and more difficult to decide about the most effective program version or OS configuration. As in car industry, a random car driver can hardly ever reproduce the gas consumption reported by the vendor.

There are many factors that explain why the observed programs performances are instable: technological factors (electronics, variable clock frequency, input/output peripherals, etc), micro-architectures (hardware prefetching, out of order execution, speculation, memory hierarchy), thread and process competition on shared ressources or data, operating systems services and actions, algorithmic factors (some application are designed to not behave exactly from one run to another), etc. If the user has complete control of its execution environment, he may be able to stabilise performances by configuring the BIOS (to deactivate some hardware features), by configuring the OS, by reducing the workload of the shared machine, etc. Unfortunately regular users cannot get such privileged rights, they are just allowed to use some computing power. Such users observe that their programs have unstable performances. These users may rely on rigorous statistical methods to decide which code version they should use for their production computation.

The instability of programs performances makes difficult to check whether a code version is better than another. So people usually make standard statistics based on mean or median values. That is, people resume all the observed performance numbers by a single value, which is not adequate in practice. The reason is that the programs performances are generally not of simple gaussian type in practice, as demonstrated in [1]. So what is the consequence if the performances of a program do not follow a gaussian distribution ? In theory, it means that some statistical tests designed for gaussians (such as the Student t-test) would compute a wrong risk level. Likewise, the well known formula that computes the confidence interval of an average would correspond to a wrong confidence level. Of course, if the performances sample is large enough, the statistical error must be asymptotically bounded (but nobody is able to define how large it should be).

In practice, the observed performances are often multimodal data distributions (see Sect. II). This suggests us to use gaussian mixtures to model the density distributions. Based on this modeling, we are able to do rigorous and flexible *parametric statistics*. We define and compute new performance metrics that go beyond mean and median values. These metrics will provide additional decision criteria to the user to analyse and to compare programs performances.

Now, what does the word parametric mean for a statistical model ? This is fairly simple: if $\mathcal{X} = (x_1, \ldots, x_n)$ denotes some execution times data (n repetitions of program executions), we suppose that they are independent realisations of some probability distribution having a probability density function (p.d.f.) f_X. The approach will be parametric if we suppose that this p.d.f. belongs to some family of distributions f_θ where θ is a vector parameter composed of d real-valued sub-parameters, which exact values are unknown: d is the dimension of the model. Since the set of *all* possible distributions is infinite dimensional, we thus reduced the problem of estimating f_X to the problem of estimating a *finite*-dimensional parameter θ: this is the main purpose of the parametric approach.

Let us also mention that, despite the appeal of the non-parametric approach, the parametric approach is still very heavily used in the scientific activities involving statistics thanks to numerous reasons:

1) The parameters of a parametric model can often be interpreted (which helps understanding and summarising the data);
2) Some computations cannot be performed, or some mathematical results cannot be proved, without parametrically specifying some parts of the model (for example, the perturbation part);
3) For really large datasets, it is often proved that the parametric techniques are indeed more accurate than their non-parametric alternatives;
4) For high dimensional data, non-parametric techniques may suffer from a lack of accuracy which is called the "curse of dimensionality", and which is out of the scope of our present research work.

It is important to notice that our statistical methods treat any kind of continuous performance data (execution times, energy consumption, network traffic, memory bandwidth, etc). We make experiments and demonstrations with programs execution times because this sort of performance is the most common and easiest to collect. Any other kind of continuous performance can be analysed using the parametric statistics we present in this article.

Our article is organised as follows. Sect. II presents our method of modelling the performance data using gaussian mixtures. Sect. III formally defines and computes four new performance metrics, and explain why they are useful in practice. Sect. IV resumes our experimental study. Related work and discussion are given in Sect. V, then we conclude.

II. Modeling programs performances with gaussian mixtures

The central idea of our work in this direction stems from the following remark: in our own experience in parallel and sequential application performance analysis, the variability of such data is not necessarily of the "deviation around a single mean value" kind, it often exhibits a clear clustering pattern: in other words, the observed performances often vary from each other by clustering around two or more central values. Examples of such observed performances are given in pages 16 and 17 in [1]. Therefore, we cannot choose for our modelling classical families of distributions such as the Gaussian, Exponential, Gamma or Weibull family, which are by nature monotonic or unimodal.

We propose to model the performances by *mixtures of gaussian distributions*: this family of distributions has demonstrated to be an essential tool in many areas of scientific activities for many years now (biology, engineering, astronomy, among many others), particularly in the image analysis and pattern recognition fields. Mixtures of gaussian distributions is a highly flexible family of distributions which can fit a great variety of data: it is not only naturally adequate for modelling data which exhibits clusters, but it can also handle the problem of possible skewness in the data, despite the symmetry of the gaussian components of the mixtures.

A. Definition of the gaussian mixtures family

Remind that we consider data sample $\mathcal{X} = (x_1, \ldots, x_n)$ which are independent realisations of a probability density function (p.d.f.) f_X. We say that the data are issued from a finite mixture of gaussian distributions (or simply a gaussian mixture, which we will abbreviate by GM from now on) if f_X is equal to some p.d.f. $g_{\theta,K}$ (parametrized by θ and K described below) of the form:

$$\begin{aligned} g_{\theta,K}(x) &= \pi_1\varphi(x;\mu_1;\sigma_1) + \ldots + \pi_K\varphi(x;\mu_K;\sigma_K) \\ &= \textstyle\sum_{k=1}^{K} \pi_k\varphi(x;\mu_k;\sigma_k) \end{aligned} \tag{1}$$

where:

- $\pi_1, \ldots, \pi_K \in \mathbb{R}^+$ are the mixture weights, which are positive and sum to 1;
- $\mu_1, \ldots, \mu_K$ and $\sigma_1, \ldots, \sigma_K$ are the mean values and standard deviations of the mixture individual components;
- $\varphi(\,\cdot\,;\mu_k;\sigma_k)$ denotes the p.d.f. of the gaussian/normal distribution $\mathcal{N}(\mu_k, \sigma_k)$. Also, for the sequel of this article, we note Φ the cumulative distribution function (c.d.f.) of $\varphi(\,\cdot\,;\mu_k;\sigma_k)$;
- $K \in \mathbb{N}$ is the number of components in this mixture;
- θ is a vector gathering all the parameters (except K) in a single notation,

$$\theta = (\pi_1, \ldots, \pi_K\,;\,\mu_1, \ldots, \mu_K\,;\,\sigma_1, \ldots, \sigma_K)$$

We will denote by $\mathcal{F}^{\text{GM}}$ the set of all mixtures of gaussian distributions, and say that X is GM-distributed if its cumulative distribution function (c.d.f.) F_X belongs to $\mathcal{F}^{\text{GM}}$, which means there exists some parameters K and $\theta = (\pi_1, \ldots, \pi_K\,;\,\mu_1, \ldots, \mu_K\,;\,\sigma_1, \ldots, \sigma_K)$ such that

$$\forall x \in \mathbb{R},\ F_X(x) \ \text{ equals } \ F_{\theta,K}(x) = \sum_{k=1}^{K} \pi_k \Phi(x;\mu_k;\sigma_k) \tag{2}$$

which is (of course) itself equivalent to f_X being equal to the density $g_{\theta,K}$ defined in Equ. 1. Naturally, the more components the mixture has, the more flexible the shape of the distribution can be (but this has a cost: the model has more parameters to be estimated).

The next section presents the method for building a GM model based on a data sample. This is called clustering in the literature.

B. Clustering method

Our aim is then, for the moment, to estimate the parameters of the distribution described in Equ. 1, from which we assume our data are issued: the parameters are the clusters weights $(\pi_k)_{k=1..K}$, the clusters means $(\mu_k)_{k=1..K}$ and the clusters standard deviations $(\sigma_k)_{k=1..K}$, and they are gathered in one single notation, θ (which is $3K$-dimensional). The formal details of our clustering method are given in [1], this section is a synthesis.

1) Estimation of the parameters of the mixture for fixed K: How then is θ estimated? Remind that the dimension of θ depends on the value of K, which is fixed for the moment. We naturally adopt a parametric approach, and intend to compute the maximum likelihood estimator $\hat{\theta}$ of θ, which is defined as the (global) maximiser of the log-likelihood function. This function, given the observations $x_1, \ldots, x_n$ and Equ. 1, is defined by:

$$L(\theta) = \sum_{i=1}^{n} \log g_{\theta,K}(x_i) = \sum_{k=1}^{K} \pi_k \varphi(x_i; \mu_k; \sigma_k)$$

Maximising it consists in calculating the various partial derivatives of L with respect to the different parameters π_k, μ_k, σ_k ($k = 1, \ldots, K$), and equalising them to 0 to obtain the so-called score equations. It is rather clear that any attempt to directly solve these score equations will turn out to be an unsolvable problem, due to the presence of a log of a sum.

This major obstacle was overcome thanks to the approach synthesised in the celebrated paper [2]: the *EM algorithm*. The acronym EM means a succession of E-steps (E for Expectation) and M-steps (M for Maximisation), which eventually lead to obtaining (numerically) the value of the maximum likelihood estimator $\hat{\theta}$. We do not detail this algorithm here, we sketch/vulgarise/explain it in [1].

2) Determination of the number K of components of the mixture: Let us explain how the number K of components of the gaussian mixture model can be chosen. In practice, several candidate values are considered for K, and one of them, noted $\hat{K}$, is chosen so that the corresponding GM model best fits the data at hand. The determination of $\hat{K}$ is nearly the most important issue in clustering analysis, and in this work we adopt a simple and widespread strategy: using the BIC criterion (BIC stands for Bayesian Information Criterion).

The principle of the BIC criterion for determining $\hat{K}$ is the following. If, for a given $K \geq 1$, we note L_K the maximum value of the log likelihood for the model with K components, then it should be easy to conceive that the greater K is, the greater the value L_K will be: indeed, for instance, if you consider the model with $K + 1$ components which best fits your data, then it will certainly better fit your data than the best model having K components (maybe not far better, but better all the same). Therefore, choosing K which maximises L_K would not work. The likelihood value needs to be penalised by a value which grows with K, in order to counterbalance the fitting gain that more complex models yield. That is the idea of the so-called information criterions, for instance the BIC criterion: to choose the value of K that minimises the value $BIC(K) = -2L_K + K\log(n)$ (this value of the penalisation term $K\log(n)$ has theoretical justifications, which will not be detailed here). Therefore, if $BIC(\hat{K}) = \max_{K\geq 1} BIC(K)$, then the model with $\hat{K}$ components will be a tradeoff between good data fitting and reasonable complexity. Note that, in practice, the maximum is chosen among a finite number of candidate values, for instance $1 \leq K \leq K_{max}$ (where K_{max} does not exceed 10 in general).

This section presented the skeleton of our clustering method based on EM algorithm. Now, we are able to build a GM model for any sample $\mathcal{X}$. The next section provides two simple examples.

3) Simple examples: Let us consider two well known SPEC benckmarks which are named `galgel` and `apsi`. Running these benchmarks multiple times on the same low overhead machine results in variable execution times, as illustrated by histograms in Fig. 1. The histograms show the frequency of observed execution times (sample of 35 runs). After clustering, the approximate theoretical models based on GM are illustrated with the curves in Fig. 1. As can be seen, the execution times of these benchmarks exhibit multi-modal behaviour.

However, how can we check if this computed model fit well the data or not? This is the purpose of a goodness-of-fitting test. We developed such test in details in [1], and demonstrated its robustness. We will discuss it later in Sect. V. We draw in Fig .2 the empirical CDF (step function) versus theoretical CDF (cuve). For both examples, the fitting is very good.

The next section presents new performance metrics for analysing and comparing programs performances, based on parametric statistics.

III. New program performance metrics

In the literature, people are mainly focused on the average or median program performance. However in practice, the average or the median value may not be the most interesting summary measure that reflects the program performance, or may not be the best performance metric to make a decision about selecting the most suitable program version.

For instance, consider the situation of a very long running application that a user executes very few times. The user has the choice between many code versions, which one should

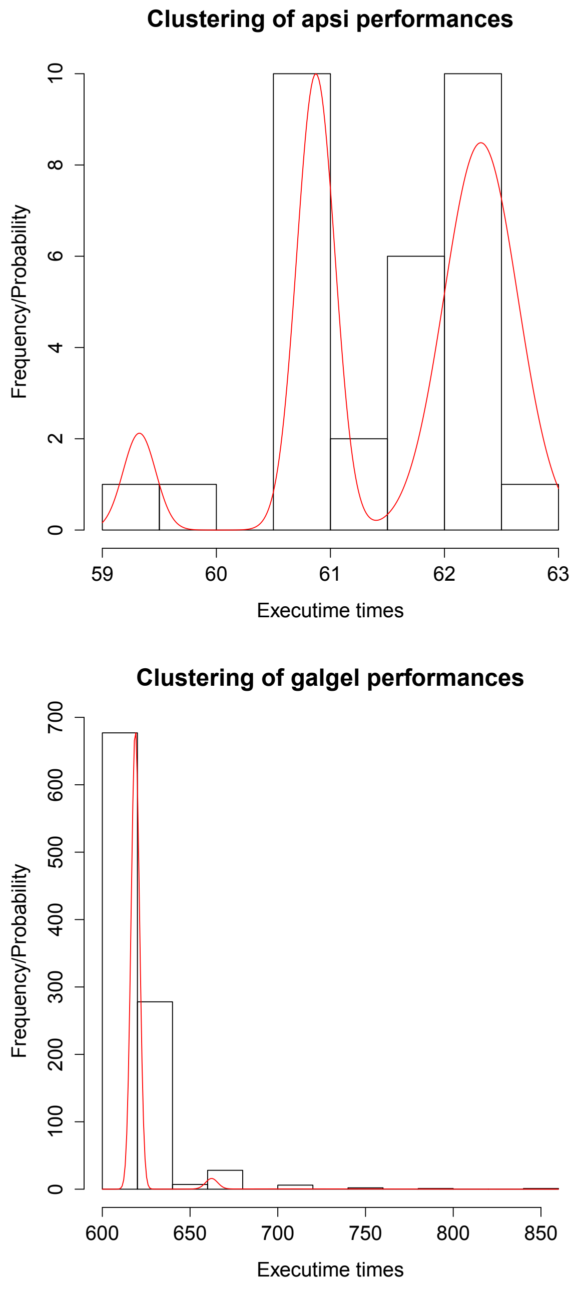

Figure 1. Gaussians mixture modelling

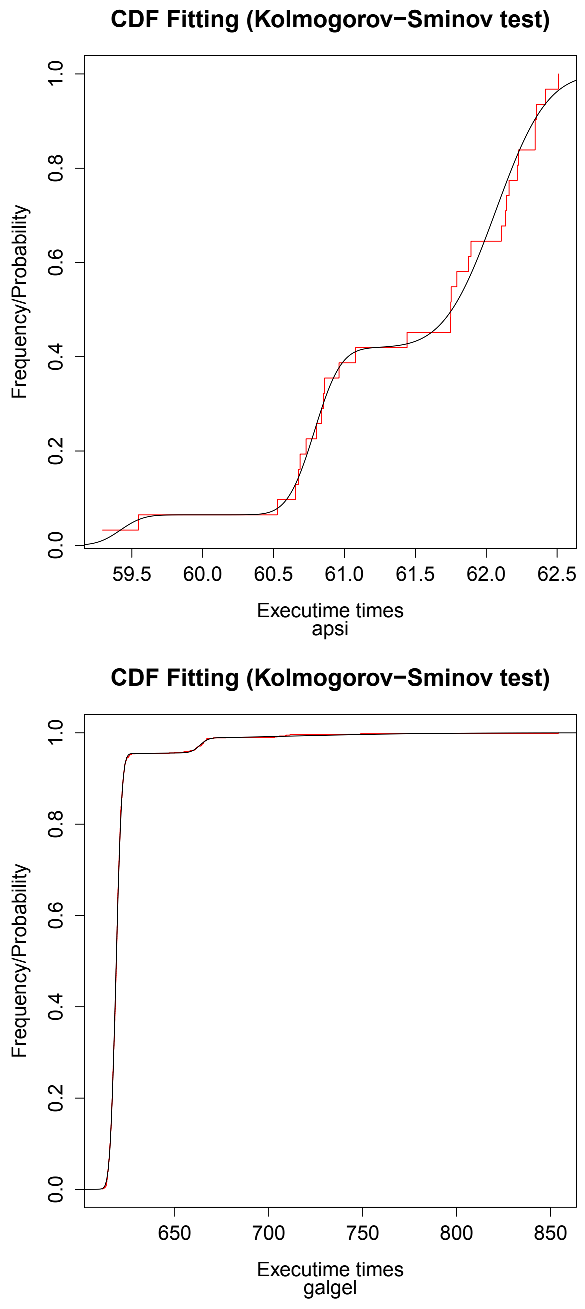

Figure 2. Quality of data fitting (empirical vs. theoretical CDF)

he select? If he bases his choice only on the expected average or median execution time, he may be disappointed if he executes his application very few times. If an application is rarely executed, the mean or median performances are not felt, especially if the performance distribution is multi-modal. So, additional performance metrics can help him making better selection.

Also, consider the situation where a user wants to know if the performances of his application are stable or not. Which metric can he use? The well known variance is a metric that measures how data is spread out around the average only: knowing how to interpret this metric is not so widespread among the practicians, and can be misleading when the data distribution is multi modal, because the variance is a measure of dispersion around a single value, the average. Therefore it cannot be the unique metric used for evaluating performance stability, additional metrics can be introduced and used, as we will see later.

This section provides new performance metrics that help the user to select a *good* program version based on performance analysis. Let X and Y two random variables, representing the performances of two code versions. Let $\mathcal{X}$ be a sample of X and $\mathcal{Y}$ be a sample of Y, meaning that $\mathcal{X} = (x_1, \dots, x_n)$ and $\mathcal{Y} = (y_1, \dots, y_m)$, with n and m denoting the respective sample sizes.

A. The metric $\mathcal{I}_1$: the mean difference

We may be interested in quantifying the average difference between the performances of two code versions. That is, we may be interested in computing the expected value $\mathbb{E}\left[|X - Y|\right]$. This defines our first performance metric as $\mathcal{I}_1 = \mathbb{E}\left[|X - Y|\right]$. Our parametric estimation, noted $\widehat{\mathcal{I}_1}$, assumes that both X and Y are modelled with gaussian mixture distributions. This means that the underlying p.d.f. f_X and f_Y of X and Y equal weighted combinations of gaussian p.d.f.

$$f_X(x) = \sum_{i=1}^{K} \pi_i \varphi(x; \mu_i; \sigma_i) \quad \text{and} \quad f_Y(x) = \sum_{j=1}^{K'} \pi'_j \varphi(x; \mu'_j; \sigma'_j)$$

where we recall here that $\varphi(x; \mu; \sigma)$ denotes the p.d.f. of the gaussian distribution $\mathcal{N}(\mu, \sigma)$, and K and K' are the respective number of clusters of these gaussian mixtures. Under this model, we readily have

$$\mathcal{I}_1 = \iint |x - y| \, f_X(x) f_Y(y) \, dx \, dy$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \iint |x - y| \varphi(x; \mu_i; \sigma_i) \, \varphi(y; \mu'_j; \sigma'_j) \, dx \, dy$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \mathbb{E}\left[|Z_i - Z'_j| \right]$$

where Z_i and Z'_j denote independent gaussian variables with distributions $\mathcal{N}(\mu_i, \sigma_i)$ and $\mathcal{N}(\mu'_j, \sigma'_j)$. By classical properties of the gaussian family, $Z_i - Z'_j$ has distribution $\mathcal{N}\left(\mu_i - \mu'_j, \sqrt{\sigma_i^2 + (\sigma'_j)^2}\right)$. Therefore, we use the following formula (proved in [1]): if Z has distribution $\mathcal{N}(\mu, \sigma)$ then $\mathbb{E}\left[|Z|\right] = (2\Phi(\mu/\sigma) - 1)\mu + 2\sigma\varphi(\mu/\sigma)$. This entails the following formula for our theoretical performance metric:

$$\mathcal{I}_1 = \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \left((\mu_i - \mu'_j) \left(2\Phi\left(\frac{\mu_i - \mu'_j}{\sqrt{\sigma_i^2 + (\sigma'_j)^2}} \right) - 1 \right) + \sqrt{\frac{2(\sigma_i^2 + (\sigma'_j)^2)}{\pi}} e^{-(\mu_i - \mu'_j)^2 / (2(\sigma_i^2 + (\sigma'_j)^2))} \right)$$

Consequently, using the estimations $\hat{\theta}, \hat{K}, \hat{\theta}', \hat{K}'$ of the parameters θ, K, θ', K' (*i.e.* the parameters of the estimated gaussian mixtures distributions $\widehat{F}^{\text{GM}}_{\mathcal{X}}$ and $\widehat{F}^{\text{GM}}_{\mathcal{Y}}$), the parametric estimation $\widehat{\mathcal{I}_1}$ of our first performance metric $\mathcal{I}_1$ comes:

$$\widehat{\mathcal{I}_1} = \sum_{i=1}^{\hat{K}} \sum_{j=1}^{\hat{K}'} \hat{\pi}_i \hat{\pi}'_j \left((\hat{\mu}_i - \hat{\mu}'_j) \left(2\Phi\left(\frac{\hat{\mu}_i - \hat{\mu}'_j}{\sqrt{\hat{\sigma}_i^2 + (\hat{\sigma}'_j)^2)}} \right) - 1 \right) + \sqrt{\frac{2(\hat{\sigma}_i^2 + (\hat{\sigma}'_j)^2)}{\pi}} e^{-(\hat{\mu}_i - \hat{\mu}'_j)^2 / (2(\hat{\sigma}_i^2 + (\hat{\sigma}'_j)^2))} \right)$$

B. The metric $\mathcal{I}_2$: the probability that a single program run is better than another

When the user needs to select which code version to execute, he may base his selection criteria on the expected average speedup for instance. But if his application is rarely executed, the average performance gain may not be interesting for him. He may be interested in executing a single time his application, and he wishes that his single run has the best chances of being the fastest between the two code versions. Formally, to help him to decide, we can compute $\mathbb{P}\left[X < Y\right]$, the probability that a single run of X would be better than a single run of Y. This defines our second metric of program performances $\mathcal{I}_2 = \mathbb{P}\left[X < Y\right]$.

Our parametric estimation assumes that both X and Y are modeled with gaussian mixture distributions.

$$\begin{aligned}
\mathcal{I}_2 &= \iint \mathbb{1}_{x<y} \, f_X(x) f_Y(y) \, dx \, dy \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \iint \mathbb{1}_{x<y} \, \varphi(x; \mu_i; \sigma_i) \, \varphi(y; \mu'_j; \sigma'_j) \, dx \, dy \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \mathbb{P}\left[Z_i - Z'_j < 0 \right]
\end{aligned}$$

where Z_i and Z'_j denote independent gaussian variables such that $Z_i - Z'_j$ is gaussian distributed with expectation $\mu_i - \mu'_j$ and variance $\sigma_i^2 + (\sigma'_j)^2$. Since $\mathbb{P}\left[Z < 0\right] = \Phi(-\mu/\sigma)$

whenever Z has distribution $\mathcal{N}(\mu, \sigma)$, we thus have $\mathcal{I}_2 = \sum_{i=1}^{K} \sum_{j=1}^{K'} \pi_i \pi'_j \; \Phi \left(\frac{\mu'_j - \mu_i}{\sqrt{\sigma_i^2 + (\sigma'_j)^2}} \right)$. Consequently, plugging in the estimators of the parameters of the gaussian mixture distributions leads to the following parametric estimator of $\mathcal{I}_2$:

$$\widehat{\mathcal{I}_2} = \sum_{i=1}^{\hat{K}} \sum_{j=1}^{\hat{K}'} \hat{\pi}_i \hat{\pi}'_j \; \Phi \left(\frac{\hat{\mu}'_j - \hat{\mu}_i}{\sqrt{\hat{\sigma}_i^2 + (\hat{\sigma}'_j)^2}} \right)$$

Alternatively, we can also generalise this metric to consider a constant real shift $\Delta \in \mathbb{R}$ to check between X and Y, and thus consider $\mathcal{I}_2 = \mathbb{P}\left[X < Y + \Delta\right]$ with its parametric estimation:

$$\widehat{\mathcal{I}_2} = \sum_{i=1}^{\hat{K}} \sum_{j=1}^{\hat{K}'} \hat{\pi}_i \hat{\pi}'_j \; \Phi \left(\frac{\Delta + \hat{\mu}'_j - \hat{\mu}_i)}{\sqrt{\hat{\sigma}_i^2 + (\hat{\sigma}'_j)^2}} \right)$$

C. The metric $\mathcal{I}_3$: the probability that a single run is better than all the others

In practice, a user may have more than only two code versions. How can he decide about the best code version among many others, for a single run only ? Comparing code versions two by two is misleading. All code versions must be compared together. Let $X_1, X_2, \cdots, X_r$ denote r random variables corresponding to r distinct code versions. We propose to compute the probability that one code version, say the first one, executes faster than all the others for a single run only (not in average or in median), *i.e.* that $X_1 < \min(X_2, \ldots, X_r)$. This defines the following program performance metric:

$$\mathcal{I}_3 = \mathbb{P}\left[X_1 < \min(X_2, \cdots, X_r)\right] = \mathbb{E}\left[\mathbb{1}_{X_1 < \min(X_2, \cdots, X_r)}\right]$$

The parametric estimation of $\mathcal{I}_3$ is noted $\widehat{\mathcal{I}_3}$. Here we assume that, for every given $j \in \{1, \ldots, r\}$, the random variable X_j is distributed as a gaussian mixture with parameters $K = K_j$ and $\theta = \theta_j = (\pi_{1,j}, \ldots, \pi_{K_j,j}; \mu_{1,j}, \ldots, \mu_{K_j,j}; \sigma_{1,j}, \ldots, \sigma_{K_j,j})$.

As we did for $\mathcal{I}_1$ and $\mathcal{I}_2$, we need to obtain a formula for the metric $\mathcal{I}_3$ in terms of the parameters. Let us note $Y = \min(X_2, \cdots, X_r)$, and let G be the c.d.f. of Y. By the mutual independence of $X_1, X_2, \ldots, X_r$, the variables X_1 and Y are independent and therefore a classical probability property yields, since $\mathbb{P}\left[x < Y\right] = (1 - G)(x)$,

$$\mathcal{I}_3 = \mathbb{P}\left[\, X_1 < Y \,\right] = \mathbb{E}\left[\mathbb{1}_{X_1 < Y}\right]$$

$$= \mathbb{E}\left[(1 - G)(X_1)\right] = \int_{-\infty}^{\infty} (1 - G(x)) f_1(x) dx.$$

By independence of the variables $X_2, \ldots, X_r$, and Equ. (2), we have:

$$(1 - G)(x) = \mathbb{P}\left[\, X_2 > x, \ldots, X_r > x \,\right] = \prod_{j=2}^{r} \mathbb{P}\left[X_j > x\right]$$

$$= \prod_{j=2}^{r} \sum_{i=1}^{K_j} \pi_j (1 - \Phi(x; \mu_{i,j}; \sigma_{i,j})).$$

Our parametric estimator $\widehat{\mathcal{I}_3}$ of the metric $\mathcal{I}_3$ is then equal to the following integral (which we compute numerically, by using the R software for instance)

$$\widehat{\mathcal{I}_3} = \int_{-\infty}^{\infty} (1 - \hat{G}(x)) \hat{f}_1(x) dx$$

where $\hat{G}(x) = \prod_{j=2}^{r} \sum_{i=1}^{\hat{K}_j} \hat{\pi}_j (1 - \Phi(x; \hat{\mu}_{i,j}; \hat{\sigma}_{i,j}))$ and $\hat{f}_1(x) = \sum_{i=1}^{\hat{K}_1} \hat{\pi}_{i,1} \varphi(x; \hat{\mu}_{i,1}; \hat{\sigma}_{i,1})$

D. The metric $\mathcal{I}_4$: the variability level

People do not always know how to quantify the variability of programs performances. By default, they use the variance, but they may not know how to interpret it. The variance measures how the data spread out around the average: but if the data are multi-modal or present clusters, then the average is not necessarily a good measure of the variability, especially when the modes or clusters are particularly distant from each other, and therefore the variance loses its attractiveness.

We propose to consider an alternative or complementary measure of the variability of programs performances: the number of modes of the underlying p.d.f. of the data, which we will note $\mathcal{I}_4$. It is simply equal to the number of *local maxima* of the p.d.f., which is supposed to represent the different values around which the data are spreading or clustering. A local maxima is called a *mode* in statistics.

Thanks to the gaussian mixture modelling, which yields an explicit formula for the estimated p.d.f., we can compute a parametric estimation $\widehat{\mathcal{I}_4}$ of $\mathcal{I}_4$, which is simply equal to the number of local maxima of the gaussian mixture p.d.f. $f_{\hat{\theta}, \hat{K}}$ estimated from the data. Often, this estimation $\widehat{\mathcal{I}_4}$ turns out to be equal to $\hat{K}$, the estimated number of clusters of the fitted gaussian mixture distribution. But it is not necessarily always the case, since sometimes the gaussian mixture fitting algorithm proposes a higher value of $\hat{K}$ than the actual number of groups in the data, in order to flexibly account for assymetry. Formally, the variability level is computed as:

$\widehat{\mathcal{I}_4}$ =number of local maxima of the estimated gaussian mixture p.d.f., which is often equal to $\hat{K}$, the number of components in this model.

IV. Implementation and experiments

We implemented all our statistical methods using the R software, and we delivered our software with a free academic licence. We did a huge and extensive set of experiments and simulations, all are detailed in [1]. We collected during more than 10 years a great amount of performance data, resulted from many empirical studies: SPEC CPU applications (2001, 2006), all SPEC OMP applications, NAS Parallel Benchmark, own micro-benchmarks, other parallel applications, various compilers versions and options, Linux versions, different HPC machines architectures and generations, etc. The number of samples is 2438, each one contains between 30 and 1000 execution times. In this section we present the list of the most important conclusions.

The gaussian mixture model fits well the data:: The first empirical study was devoted to validating the robustness of our goodness-of-it test, based on Kolmogorov-Smirnov distances. We used extensive statistical simulations to check the robustness and the error ratio of the test and bootstrap calibration, the results are very satisfying compared to what is used in practice for other tests [1]. Based on this test, we are able to ensure that 83% of the samples are very well fitted with gaussian mixtures. The remaining 17% samples were rejected from gaussian mixture modelling. After investigation, we found that the reason is mostly because they contain *ties* (identical values), that result from rounding errors when collecting performance data. It remains however a marginal set of samples that cannot be modelled by gaussian mixtures, other distributions should be considered.

The performance metrics are accurate enough:: we also did an extensive set of experiments to check the accuracy of our new performance metrics $\widehat{\mathcal{I}_1}$, $\widehat{\mathcal{I}_2}$, $\widehat{\mathcal{I}_3}$ and $\widehat{\mathcal{I}_4}$. We computed by simulations their mean square errors and mean absolute percentage errors, compared to the theoretical exact values $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$ and $\mathcal{I}_4$. The accuracy of our metrics are empirically satisfactory [1].

Empirical study of variability levels of programs execution times:: Based on our metric for the estimation of variability level ($\widehat{\mathcal{I}_4}$), we computed the value of this metric for all the samples. We found:

- $\approx$ 37% of the samples have a variability level equal to one, which means that the execution times are spread around a single value.
- $\approx$ 32% of the samples have a variability level equal to 2, which means that the execution times are spread around two values.
- $\approx$ 12% of the samples have a variability level equal to 3, which means that the execution times are spread around three values.
- $\approx$ 19% of the samples have a variability level $\geq$ 4, which means that the execution times are spread around more than three values.

The above observations reinforce our opinion that program performances follow multimodal distributions, thus mean and median values are not always adequate to resume performance data.

V. Related work and discussion

Program performance evaluation:: In the field of code optimisation and high performance computing, most of the published articles declare observed speedups or other performance metrics. Unfortunately, few studies based on rigorous statistics are really conducted to check whether the observations of the code performance improvements are statistically significant or not.

Program performance analysis and optimisation may rely on two well known books that explain digest statistics to our community [3], [4] in an accessible way. These two books are good introductions for doing fair statistics for analysing performance data. Based on these two books, previous work on statistical program performance evaluation have been published [5]. In the later article, the authors rely on the Student's t-test to compare between two average execution times (the two sided version of the student t-test) in order to test if two theoretical means are unequal. We improved the previous work in [6]: first, we showed how to conduct a *one-sided* Student's t-test to validate that $\mu_X > \mu_Y$. Second, we show how to check the normality if small samples and the equivalence of their variances (using the Fisher's F-test) in order to use the classical Student's t-test instead of the Welch's variant.

In [7], we presented a rigorous statistical methodology regarding program performance analysis. We rely on well known statistical tests (Shapiro-wilk's test, Fisher's F-test, Student's t-test, Kolmogorov-Smirnov's test, Wilcoxon-Mann-Whitney's test) to study if the observed speedups are statistically significant or not. By fixing $0 < \alpha < 1$ a desired risk level, we are able to analyse the statistical significance of the average execution time as well as the median.

In the current article, we go beyond the classical mean and median performance factors. We provide additional interesting performance metrics, based on parametric statistics (gaussian mixture modelling).

References on gaussian mixtures:: References about mixture models and particularly gaussian mixture models are numerous, in the statistics literature, machine-learning literature, as well as in many statistics-using fields (in particular image analysis, bioinformatics, biology, medicine, etc); we therefore only cite the reference book on mixture models [8], [9], [10].

The use of the EM algorithm as a solution to finite mixtures fitting is a classical subject in the statistics and pattern recognition literature, since the release of the breakthrough paper [2] (note that the strength of the EM algorithm is that it is not restricted to the estimation of gaussian mixtures, but extend to other mixtures, often with a computational cost though). In the present work, we decided to address the important issue of choosing the appropriate number K of components by relying on the BIC criterion: there are however several popular alternatives, for which some references are [11], [12].

Discussion:: Concerning the performance of the goodness-of-fit test we introduced in [1], we found out that it was very satisfying. We nonetheless observed that, in the presence of too much a proportion of equal data in the dataset, our fitting test tends to artificially reject the gaussian mixture model too often: a simple solution to this problem would be to increase the precision of the measurement method so that the risk of observing exæquo is reduced.

Gaussian mixtures seem to be good model for most of the performances data that we observed in practice. Also, such data distributions provide interesting perspective regarding multi-dimensional performance data. Indeed, an interesting performance analysis must consider the performance as multi-dimensional data, each dimension corresponds to a specific performance nature. For instance we can consider the triplet $\{$Execution time, energy consumption, network traffic$\}$. Fortunately gaussian mixtures are a good solution for modelling multi-dimensional data in order to analyse the relationship between multiple dimensions.

Some people are interested in extreme values statistics (Worst Case Execution Times, Best Case Execution Times). In that case, gaussian mixture modelling is not necessarily the adequate way of addressing this issue: either mixtures of other data distributions must be considered, or alternative methods should be considered (extreme value analysis techniques in particular, although they require quite a large number of data values to be truly reliable).

VI. Conclusion

In the presence of program performance variability, we must rely on formal statistics to decide about the best code version. Currently, people analyse mainly mean and median performances only. In the current research work, we extend these metrics as follows:

1) We build a statistical modelling based on gaussian mixtures to fit multi-modal data distributions;
2) We build a statistical test to quantify the quality of data-model fitting, based on Kolmogorov-Smirnov distances;
3) We define new code performance metrics that go beyond mean and median performances.;
4) We implemented all our statistical methods using R software, and we demonstrate its practical efficiency.
5) Our software, called `VARCORE`, is publicly distributed as a free open source code.

Our gaussian mixture modelling provides a new and interesting metric to evaluate the variability of performances. Indeed, instead of only considering means and variances (the variances being, by the way, difficult to interpret in practice), we propose to consider as well the *modes* of the data distributions. Thus, the variability level of performances can be measured by the number of these modes, which we can compute with a parametric method based on gaussian mixtures. The number of modes is a natural way of giving an idea of the performance variability: a data distribution with a single mode means that the performances are quite stable around a single value; with two modes, it means that the performances are varying around two values, etc. In addition, if the number of modes is greater than one, this can be a good indication of being careful with the interpretation (and comparison) of the variance (since the usual interpretation generally assumes that the data are issued from a unimodal distribution). Moreover, the existence of these modes can be further investigated by trying to explain them with auxiliary measurements made during the execution of the program, which is certainly the most fruitful perspective of this research work.

References

[1] J. Worms and S. Touati, "Parametric and Non-Parametric Statistics for Program Performance Analysis and Comparison," Université Nice Sophia Antipolis ; Université Versailles Saint Quentin en Yvelines, Research Report RR-8875, Mar. 2016, https://hal.inria.fr/hal-01286112.

[2] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[3] R. Jain, *The Art of Computer Systems Performance Analysis : Techniques for Experimental Design, Measurement, Simulation, and Modelling*. New York: John Wiley and Sons, inc., 1991, ISBN-13: 978-0-471-503361.

[4] D. J. Lilja, *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2000, ISBN-13: 978-0521641050.

[5] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous Java performance evaluation," in *Proceedings of the Twenty-Second ACM SIGPLAN Conference on OOPSLA*, ser. ACM SIGPLAN Notices 42(10), Montréal, Canada, Oct. 2007, pp. 57–76.

[6] A. Mazouz, S. Touati, and D. Barthou, "Study of Variations of Native Program Execution Times on Multi-Core Architectures," in *International Workshop on Multi-Core Computing Systems (MuCoCoS)*. Krakow, Poland: IEEE, Feb. 2010.

[7] S. Touati, J. Worms, and S. Briais, "The Speedup-Test: A Statistical Methodology for Program Speedup Analysis and Computation," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 10, pp. 1410–1426, 2013. [Online]. Available: https://hal.inria.fr/hal-00764454

[8] G. MacLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley series in Probability and Statistics, 2000.

[9] K. L. Mengersen, C. P. Robert, and D. M. Titterington, *Mixture Estimation and Applications*. Wiley, 2011.

[10] D. M. Titterington, A. F. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

[11] A. Fujita, D. Y. Takahashi, and A. G. Patriota, "A nonparametric method to estimate the number of clusters," *Computational Statistics and Data Analysis*, vol. 73, pp. 27–39, 2014.

[12] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society, Series B*, vol. 63, no. 2, pp. 411–423, 2001.