Statistical Performance Comparisons of Computers*

Tianshi Chen[†§], Yunji Chen[†§], Qi Guo[†], Olivier Temam[‡], Yue Wu[†], Weiwu Hu[†§]

† State Key Laboratory of Computer Architecture,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
‡ INRIA, Saclay, France
§ Loongson Technologies Corporation Limited, Beijing, China

Abstract

As a fundamental task in computer architecture research, performance comparison has been continuously hampered by the variability of computer performance. In traditional performance comparisons, the impact of performance variability is usually ignored (i.e., the means of performance measurements are compared regardless of the variability), or in the few cases where it is factored in using parametric confidence techniques, the confidence is either erroneously computed based on the distribution of performance measurements (with the implicit assumption that it obeys the normal law), instead of the distribution of sample mean of performance measurements, or too few measurements are considered for the distribution of sample mean to be normal. We first illustrate how such erroneous practices can lead to incorrect comparisons.

Then, we propose a non-parametric Hierarchical Performance Testing (HPT) framework for performance comparison, which is significantly more practical than standard parametric techniques because it does not require to collect a large number of measurements in order to achieve a normal distribution of the sample mean. This HPT framework has been implemented as an open-source software.

1 Introduction

A fundamental practice for researchers, engineers and information services is to compare the performance of two architectures/computers using a set of benchmarks. As trivial as this task may seem, it is well known to be fraught with obstacles, especially the selection of benchmarks [31, 33, 3] and performance variability [1]. In this paper, we focus on the issue of performance variability. The variability can have several origins, such as non-deterministic architecture+software behavior [6, 26], performance measurement bias [27], or even applications themselves. Whatever the origin, the performance variability is well known to severely hamper the comparison of computers. For instance, the geometric mean performance speedups, over an initial baseline run, of 10 subsequent runs of SPLASH-2 on a commodity computer (Linux OS, 4-core 8-thread Intel i7 920 with 6 GB DDR2 RAM) are 0.94, 0.98, 1.03, 0.99, 1.02, 1.03, 0.99, 1.10, 0.98, 1.01. Even when two computers/architectures may have fairly different performance, such variability can still make the quantitative comparison (e.g., estimating the performance speedup of one computer over another) confusing or plain impossible. Incorrect comparisons can, in turn, affect research or acquisition decisions, so the consequences are significant.

The most common and intuitive way of addressing the impact of variability is to compute the confidence of performance comparisons. The most broadly used techniques for estimating the confidence are *parametric* confidence estimate techniques [1, 27]. However, at least two kinds of wrongful practices commonly plague the usage of such techniques, potentially biasing performance comparisons, and consequently, sometimes leading to incorrect decisions.

The first issue is that computing a confidence based on a statistical distribution implicitly requires that distribution to be normal. However, the Central Limit Theorem (CLT) [4] says that, even if a distribution of performance measurements is not normal, the distribution of the *sample mean* (a sample is simply a set of performance measurements) tends to a normal distribution as the sample size (the number of measurements in each sample) increases. The problem, though, is that the confidence estimate is often erro-

*T. Chen, Y. Chen, Q. Guo, and W. Hu are partially supported by the National Natural Science Foundation of China (under Grants 61100163, 61003064, 61173006, 61133004, 60736012, and 60921002), the National S&T Major Project of China (under Grants 2009ZX01028-002-003, 2009ZX01029-001-003, and 2010ZX01036-001-002), and the Strategic Priority Research Program of the Chinese Academy of Sciences (under Grant XDA06010400). O. Temam is supported by HiPEAC-2 NoE under Grant European FP7/ICT 217068 and INRIA YOUHUA.

978-1-4673-0825-0/12/$31.00 ©2012 IEEE

neously based on the *distribution of performance measurements* (with the implicit assumption that it obeys the normal law), instead of the distribution of the sample mean. That wrongful practice could still lead to a correct estimate of the confidence if the distribution of performance measurements is, by chance, normal, but we will empirically show that the distribution of performance measurements can easily happen to be non-normal, so that such erroneous usage of the confidence estimate is harmful.

The second issue is that, even if parametric confidence estimate techniques are correctly based on the distribution of the sample mean, the number of measurements collected is usually insufficient to achieve a normally distributed sample mean. In day-to-day practices in computer architecture research, such techniques are commonly applied when about 30 performance measurements have been collected. However, we will empirically show that a very large number of performance measurements, on the order of 160 to 240, are required for the CLT to be applicable, so that current practices can again lead to harmful usage of the parametric techniques.

In this paper, we introduce the Hierarchical Performance Test (HPT) which can correctly quantify the confidence of a performance comparison even if only a few performance measurements are available. The key insight is to use *non-parametric Statistic Hypothesis Tests (SHTs)*. Parametric statistical methods (such as the paired t-test or confidence interval) rely on some distribution assumptions, while the non-parametric SHTs quantify the confidence using data rankings. Non-parametric SHTs can work when there are only a few performance measurements, because they do not need to characterize any specific distribution. To help disseminate the use of non-parametric SHTs, we design a self-contained Hierarchical Performance Test (HPT) and the associated software implementation, which will be openly distributed.

In summary, the contributions of this paper are the following. First, we empirically highlight that traditional performance comparisons simply based on means of performance measurements can be unreliable because of the variability of performance results. As a result, we stress that every performance comparison should come with a confidence estimate in order to judge whether a comparison result corresponds to a stochastic effect or whether it is significant enough. Second, we provide quantitative evidence that two common wrongful practices, i.e., using the distribution of performance measurements instead of the distribution of sample mean, and using too few performance measurements, are either harmful or render parametric confidence techniques impractical for our domain. Third, we propose and implement the HPT framework based on non-parametric SHTs, which can provide a sound quantitative estimate of the confidence of a comparison, independently of the distribution and the number of performance measurements.

The rest of the paper is organized as follows. Section 2 motivates the need for systematically assessing the confidence of performance measurements, and for using the appropriate statistical tools to do so. Section 3 investigates the impact of using the wrong distribution or of collecting an insufficient number of performance measurements for the normality assumption of parametric confidence techniques. Section 4 introduces the non-parametric hierarchical performance testing framework. Section 5 empirically compares the HPT with traditional performance comparison techniques. Section 6 reviews the related work.

2 Motivation

In this section, we introduce two examples to motivate this study. The first example shows the significance of using statistical techniques in performance comparisons, and the second example shows the importance of selecting appropriate statistical techniques.

In a quantitative performance comparison, the performance speedup of one computer over another is traditionally obtained by comparing the geometric mean performance of one computer (over different benchmarks) with that of another, a practice adopted by SPEC.org [31]. Following this methodology, the performance speedup of PowerEdge T710 over Xserve on SPEC CPU2006, estimated by the data collected from SPEC.org [31], is 3.50. However, after checking this performance speedup with the HPT proposed in this paper, we found that the confidence of such a performance speedup is only 0.31, which is rather unreliable (≥ 0.95 is the statistically acceptable level of confidence). In fact, our HPT reports that the reliable performance speedup is 2.24 (with a confidence of 0.95), implying that the conclusion made by comparing geometric mean performance of two computers brings an error of 56.2% for the quantitative comparison.

Let us now use another example to briefly illustrate why using parametric confidence techniques incorrectly can be harmful. In this section we emulate the wrongful practice of computing the confidence based on the distribution of performance measurements, instead of the distribution of sample mean. We consider the quantitative performance comparison of another pair of commodity computers, Asus P5E3 Premium and CELSIUS R550. In Figure 1, we show the histograms of the SPEC ratios of the two computers (collected from SPEC.org). Clearly, neither of the histograms appears to correspond to a normal distribution. We further evaluated the normality of the distributions using the Lilliefors test (Kolmogorov-Smirnov test) [24], and we found that, for both computers, the normality assumption is indeed incorrect for confidences larger than 0.95.

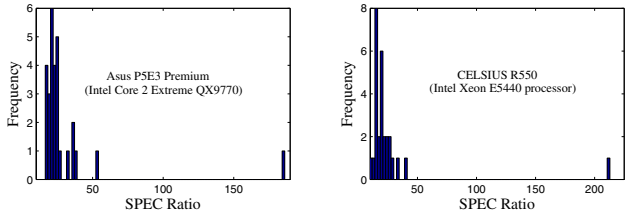

Figure 1. Histograms of SPEC ratios of two commodity computers.

Voluntarily ignoring that observation, we incorrectly use the confidence interval based on the non-normal distribution of performance measurements for the quantitative comparison. The confidence interval technique says that Asus P5E3 Premium is more than 1.02 times faster than CELSIUS R550 with a confidence of 0.95, and 1.15 times faster with a confidence of 0.12. According to the HPT, the Asus P5E3 Premium is actually more than 1.15 times faster than the CELSIUS R550 with a confidence of 0.95. So the confidence of the claim "Asus P5E3 Premium is 1.15 times faster than CELSIUS R550" is drastically under-evaluated (0.12 instead of 0.95), or conversely, the assessment of the speedup for a fixed confidence of 0.95 is again largely under-evaluated (1.02 instead of 1.15).

In summary, it is important not only to assess the confidence of a performance comparison, but also to correctly evaluate this confidence.

3 Issues with Current Performance Comparison Practices

In this section, we empirically highlight that the distribution of performance measurements may not be normal, and thus it cannot be used to directly compute the confidence. Then, we empirically again show that a large number of measurements are required in order to use the Central Limit Theorem (CLT).

3.1 Checking the Normality of Performance Measurements

To compare the performance of different computers, we expect that the performance score on each benchmark can stably reflect the exact performance of each computer. However, the performance of a computer on every benchmark is influenced by not only architecture factors (e.g., out-of-order execution, branch prediction, and chip multiprocessor [34]) but also program factors (e.g., data race, synchronization, and contention of shared resources [2]). In the presence of these factors, the performance score of a computer on a benchmark is usually a random variable [1]. For example, according to our experiments using SPLASH-2 [33], the execution time of one run of a benchmark can be up to 1.27 times that of another run of the same benchmark on the same computer. Therefore, it is necessary to provide some estimate of the confidence of a set of computer performance measurements.

As mentioned before, the confidence is sometimes incorrectly assessed based on the distribution of performance measurements, which can only lead to a correct result if that distribution is normal. We empirically show that this property is not valid for the following set of rather typical performance measurements. In our experiments, we run both single-threaded (*Equake*, SPEC CPU2000 [31]) and multi-threaded benchmarks (*Raytrace*, SPLASH-2 [33] and *Swaptions*, PARSEC [3]) on a commodity Linux workstation with a 4-core 8-thread CPU (Intel i7 920) and 6 GB DDR2 RAM. Each benchmark is repeatedly run for 10000 times, respectively. At each run, *Equake* uses the "test" input defined by SPEC CPU2000, *Raytrace* uses the largest input given by SPLASH-2 (car.env), and *Swaptions* uses the second largest input of PARSEC (simlarge). Without losing any generality, we define the performance score to be the execution time.

To check the normality of the performance, we empirically study whether the normal Probability Density Function (PDF) obtained by assuming the normality of the execution time complies with the real PDF of the execution time. In our experiments, we utilize two statistical techniques, Naive Normality Fitting (NNF) and Kernel Parzen Window (KPW) [28]. The NNF technique assumes that the execution time obeys a normal law and estimates the corresponding normal distribution, while the KPW technique provides the real distribution of the execution time without assuming a normal distribution. If the normal distribution obtained by the NNF complies with the real distribution estimated by the KPW, then the performance score obeys a normal law. Otherwise, it does not. Statistically, the NNF technique directly employs the mean and deviation of the sample (with 10000 measurements of the execution time) as the mean and standard deviation of the normal distribution. In contrast, without assuming the normality, the KPW technique estimates the real distribution of the execution time in a Monte-Carlo style. It estimates directly the probability density at each point via histogram construction and Gaussian kernel smoothing. By comparing the PDFs obtained by the two techniques, we can easily identify whether or not the distribution of performance measurements obeys a normal law.

According to the experimental results illustrated in Figure 2, the normality does not hold for the performance score of the computer on all three benchmarks, as evidenced by the remarkably long right tails and short left tails of the

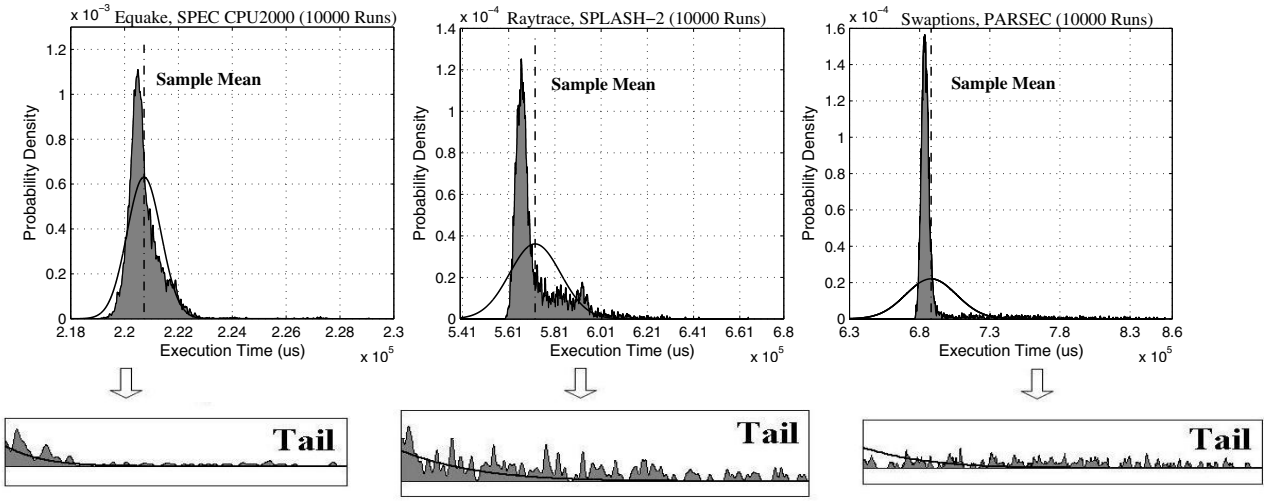

Figure 2. Estimating Probability Density Functions (PDFs) on *Equake* (SPEC CPU2000), *Raytrace* (SPLASH-2) and *Swaptions* (PARSEC) by KPW (black curves above the grey areas) and NNF (black curves above the white areas), from 10000 repeated runs of each benchmark on the same computer.

estimated performance distributions for *Equake*, *Raytrace* and *Swaptions*. Such observations are surprising but not counter-intuitive due to the intrinsic non-determinism of computers and applications. Briefly, it is hard for a program to execute faster than a threshold, but easy to be slowed down by various events, especially for multi-threaded programs which are affected by data races, thread scheduling, synchronization order, and contentions of shared resources.

As a follow-up experiment, we use a more rigorous statistical technique to study whether the execution times of the 27 benchmarks of SPLASH-2 and PARSEC (using "simlarge" inputs) distribute normally; each benchmark is repeatedly run on the commodity computer for 10000 times again. Based on these measurements, the Lilliefors test (Kolmogorov-Smirnov test) [24] is utilized to estimate the confidence that the execution time does not obey the normal law, i.e., the confidence that the normality assumption is incorrect. Interestingly, it is observed that for *every* benchmark of SPLASH-2 and PARSEC, the confidence that the normality assumption is incorrect is above 0.95. Our observation with SPLASH-2 and PARSEC is significantly different from the observation of Georges et al. [12] that single-benchmark performance on single cores (using SPECjvm98) distributes normally, suggesting that the performance variability of multi-threaded programs is fairly different from that of single-threaded programs.

In fact, the same can be observed for the performance variability of a computer over a set of different benchmarks. Considering the performance score of a computer which may vary from one benchmark to another, we empirically study whether the distribution of the performance score obeys the normality law. Figure 3 illustrates the normal probability plot [5] for the performance of a commodity computer (4-Core Intel Core i7-870, Intel DP55KG motherboard), where the data is collected from the SPEC online repository [31]. In each probability plot presented in Figure 3, if the curve matches well the straight line, then the performance distributes normally over the corresponding benchmark suite; if the curve departs from the straight line, then the performance does not distribute normally. Obviously, neither of the figures shows a good match between the curve and straight line, implying that the performance of the computer does not distribute normally over both SPECint2006 and SPECfp2006.

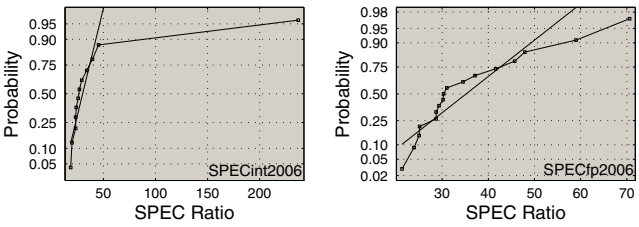

Figure 3. Graphically assessing whether the performance measurements of a commodity computer distribute normally (normal probability plots [5]) using performance scores (SPEC ratios) for SPECint2006 and SPECfp2006.

Finally, we use the Lilliefors test [24] to analyze the data of 20 other commodity computers randomly selected from the SPEC online repository [31]. Using the whole

SPEC CPU2006 suite, all 20 computers exhibit non-normal performance with a confidence larger than 0.95. For SPECint2006, 19 out of 20 computers exhibit non-normal performance with a confidence larger than 0.95, and for SPECfp2006, 18 out of 20 computers exhibit non-normal performance with a confidence larger than 0.95.

So we can consider that, in general, the distribution of performance measurements does not obey the normal law, and thus, it should never be used to directly estimate the confidence of these measurements with parametric techniques. Only the distribution of the sample mean should be used, as stated by the Central Limit Theorem.

3.2 Checking the Applicability of the Central Limit Theorem

So far we have empirically shown that the performance of computers cannot be universally characterized by normal distributions. However, it is still possible to obtain a normally distributed mean of performance measurements (in order to apply parametric techniques) given a sufficiently large number of measurements, as guaranteed by the Central Limit Theorem. More specifically, let $\{x_1, x_2, \ldots, x_n\}$ be a size-n sample consisting of n measurements of the same non-normal distribution with mean μ and finite variance σ^2, and $S_n = (\sum_{i=1}^{n} x_i)/n$ be the mean of the measurements (i.e., sample mean). According to the classical version of the CLT contributed by Lindeberg and Lévy [4], when the sample size n is sufficiently-large, the sample mean approximately obeys a normal distribution. Nevertheless, in practice, it is unclear how large the sample size should be to address the requirement of "a sufficiently large sample". In this section, we empirically show that the appropriate sample size for applying the CLT is usually too large to be compatible with current practices in computer performance measurements.

In order to obtain a large number performance measurements, we use the KDataSets [7]. A notable feature of KDataSets is that it provides 1000 distinct data sets for each of 32 different benchmarks (MiBench [14]), for a total of 32,000 distinct runs. We collect the detailed performance scores (performance ratios normalized to an ideal processor executing one instruction per cycle) of a Linux workstation (with 3GHz Intel Xeon dualcore processor, 2GB RAM) over the 32,000 different combinations of benchmarks and data sets of KDataSets [7]. In each trial, we fix the number of different samples to 150, leading to 150 observations of sample mean, which are enough to obtain a normal distribution as predicted by the CLT [15]. In order to construct each sample, we randomly select n performance scores out of the 32,000 performance scores, where the sample size n (i.e., number of measurements) is varied from 10 to 280 by increments of 20. For each trial with a fixed sample size, we estimate the probability distribution of the *sample mean* via the statistical technique called Kernel Parzen Window (KPW) [28]. Technically, the KPW has a parameter called "window size" or "smoothing bandwidth", which determines how KPW will smooth the distribution curve. In general, the larger the window size, the smoother the curve. But if we choose a too large window size, there is a risk that a non-normal curve is rendered as normal by KPW. In order to avoid that, in each trial we start from a small window size, and we increase it until the distribution curve becomes smooth enough. After that, we judge whether the distribution has approached a normal distribution by checking if it is symmetric with respect to its center. The different probability distributions over the 15 trials are illustrated in Figure 4.

Clearly, the sample mean does not distribute normally given a small (e.g., $n = 10 - 140$) sample size. When the sample size becomes larger than 240, the distribution of the mean performance seems to be a promising approximation of a normal distribution. In addition, we further carry out the Lilliefors test for each trial, and we find that the mean performance does not distribute normally when $n < 160$. The above observation implies that at least 160 to 240 performance measurements are necessary to make the CLT and parametric techniques applicable, at least for this benchmark suite. However, such a large number of performance measurements can rarely be collected in day-to-day practices; most computer architecture research studies rely on a few tens of benchmarks, with one to a few data sets each, i.e., far less than the aforementioned number of required performance measurements. In order to cope with a small number of performance measurements, we propose to use a non-parametric statistical framework.

4 Non-parametric Hierarchical Performance Testing Framework

Statistical inference techniques are popular for decision making processes based on experimental data. One crucial branch of statistical inference is called Statistical Hypothesis Test (i.e., SHT).

More specifically, an SHT is a procedure that makes choices between two opposite hypotheses (propositions), the NULL (default) hypothesis and the alternative hypothesis. The NULL hypothesis represents the default belief, i.e., our belief before observing any evidence, and the alternative hypothesis (often the claim we want to make) is a belief opposite to the NULL hypothesis. In performance comparison, a typical NULL hypothesis may be "computer A is as fast as computer B", and a typical alternative hypothesis may be "computer A is faster than computer B". At the beginning of an SHT, one assumes the NULL hypothesis to be correct, and constructs a statistic (say, Z) whose value

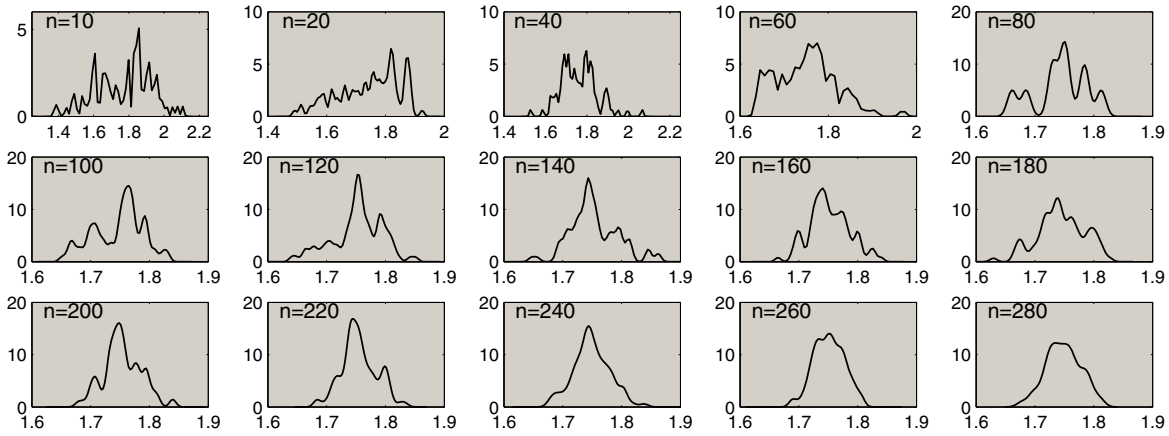

Figure 4. How many performance measurements are required to construct a sufficiently-large sample?

can be calculated from the observed data. The value of Z determines the possibility that the NULL hypothesis holds, which is critical for making a choice between the NULL hypothesis and the alternative hypothesis. The possibility that the NULL hypothesis does hold is quantified as the so-called $p\text{-}value$ (or significance probability) [21], which is a real value between 0 and 1 that can simply be considered as a measure of *risk* associated with the *alternative* hypothesis. The $p\text{-}value$ is an indicator for decision-making: when the $p\text{-}value$ is small enough, then the risk of incorrectly rejecting the NULL hypothesis is very small, and the confidence of the alternative hypothesis (i.e., $1 - p\text{-}value$) is large enough. For example, in an SHT, when the $p\text{-}value$ of the NULL hypothesis "computer A is as fast as computer B" is 0.048, we only have a 4.8% chance of rejecting the NULL hypothesis when it actually holds. In other words, the alternative hypothesis "computer A is faster than computer B" has confidence $1-0.048 = 0.952$. Closely related to the $p\text{-}value$, the significance level acts as a scale of the ruler for the $p\text{-}value$ (frequently-used scales include 0.001, 0.01, 0.05, and 0.1). A significance level $\alpha \in [0, 1]$, can simply be viewed as the confidence level $1 - \alpha$. As a statistical convention, a confidence no smaller than 0.95 is often necessary for reaching the final conclusion.

Among the most famous and broadly used SHTs, many are parametric ones which rely on normally distributed means of measurements. In the rest of this section, we introduce a Hierarchical Performance Testing (HPT) framework, which integrates non-parametric SHTs for performance comparisons.

4.1 General Flow

Concretely, the HPT employs Wilcoxon *Rank-Sum* Test [35] to check whether the difference between the performance scores of two computers on each benchmark is significant enough (i.e., the corresponding significance level is small enough), in other words, whether the observed superiority of one computer over another is reliable enough. Only significant (reliable) differences, identified by the SHTs in single-benchmark comparisons, can be taken into account by the comparison over different benchmarks, while those insignificant differences will be ignored (i.e., the insignificant differences are set to 0) in the comparison over different benchmarks. Based on single-benchmark performance measurements, the Wilcoxon *Signed-Rank* Test [9, 35] is employed to statistically compare the general performance of two computers. Through these non-parametric SHTs, the HPT can quantify the confidence for performance comparisons. In this section, the technical details of the HPT will be introduced. [1]

Let us assume that we are comparing two computers A and B over a benchmark suite consisting of n benchmarks. Each computer repeatedly runs each benchmark m times ($m \geq 3$). Let the performance scores of A and B at their j-th runs on the i-th benchmark be $a_{i,j}$ and $b_{i,j}$ respectively. Then the performance samples of the computers can be represented by *performance matrices* $S_A = [a_{i,j}]_{n\times m}$ and $S_B = [b_{i,j}]_{n\times m}$, respectively. For the corresponding rows of S_A and S_B (e.g., the τ-th rows of the matrices, $\tau = 1, \ldots, n$), we carry out the Wilcoxon Rank-Sum Test to investigate whether the difference between the performance scores of A and B is significant enough. The concrete steps of Wilcoxon Rank-Sum Test are the following:

• Let the NULL hypothesis of the SHT be "$H_{\tau,0}$*: the performance scores of A and B on the τ-th benchmark are equivalent to each other*"; let the alternative hypothesis of the SHT be "$H_{\tau,1}$*: the performance score of A is higher than that of B on the τ-th benchmark*" or "$H_{\tau,2}$*: the performance score of B is higher than that of A on the τ-th benchmark*", depending on the motivation of carrying out

[1] Users who are not interested in the mathematical details of the non-parametric SHTs can omit the rest of this subsection.

the SHT. Define the significance level be α_τ; we suggest setting $\alpha_\tau = 0.05$ for $m \geq 5$ and 0.10 for the rest cases.

• Sort $a_{\tau,1}, a_{\tau,2}, \ldots, a_{\tau,m}, b_{\tau,1}, b_{\tau,2}, \ldots, b_{\tau,m}$ in ascending order, and assign each of the scores the corresponding rank (from 1 to $2m$). In case two or more scores are the same, but their original ranks are different, then renew the ranks by assigning them the average of their original ranks[2]. Afterwards, for A and B, we can define their rank sums (on the τ-th benchmark) to be:

$$\mathrm{R}_{a,\tau} = \sum_{j=1}^{m} \mathrm{Rank}_\tau(a_{\tau,j}), \quad \mathrm{R}_{b,\tau} = \sum_{j=1}^{m} \mathrm{Rank}_\tau(b_{\tau,j}),$$

where $\mathrm{Rank}_\tau(\cdot)$ provides the rank of a performance score on the τ-th benchmark.

• **Case[**$m < 12$**]**[3]**:** When the alternative hypothesis of the SHT is $H_{\tau,1}$, we reject the NULL hypothesis and accept $H_{\tau,1}$ if $\mathrm{R}_{a,\tau}$ is *no smaller than* the critical value (right tail, Wilcoxon Rank-Sum Test) under the significance level α_τ. When the alternative hypothesis of the SHT is $H_{\tau,2}$, we reject the NULL hypothesis and accept $H_{\tau,2}$ if $\mathrm{R}_{b,\tau}$ is *no smaller than* the critical value under the significance level α [15].

• **Case[**$m \geq 12$**]:** Define two new statistics $z_{a,\tau}$ and $z_{b,\tau}$ as follows:

$$z_{a,\tau} = \frac{\mathrm{R}_{a,\tau} - \frac{1}{2}m(2m+1)}{\sqrt{\frac{1}{12}m^2(2m+1)}}, \quad z_{b,\tau} = \frac{\mathrm{R}_{b,\tau} - \frac{1}{2}m(2m+1)}{\sqrt{\frac{1}{12}m^2(2m+1)}}.$$

Under the NULL hypothesis, $z_{a,\tau}$ and $z_{b,\tau}$ approximately obey the standard normal distribution $\mathcal{N}(0,1)$. When the alternative hypothesis of the SHT is $H_{\tau,1}$, we reject the NULL hypothesis and accept $H_{\tau,1}$ if $z_{a,\tau}$ is *no smaller than* the critical value (right tail, standard normal distribution) under the significance level α; when the alternative hypothesis of the SHT is $H_{\tau,2}$, we reject the NULL hypothesis and accept $H_{\tau,2}$ if $z_{b,\tau}$ is *no smaller than* the critical value under the significance level α [15].

After carrying out the above SHT with respect to the τ-th benchmark ($\tau = 1, \ldots, n$), we are able to assign the difference (denoted by d_τ) between the performance of A and B. Concretely, if the SHT accepts $H_{\tau,1}$ or $H_{\tau,2}$ with a promising significance level (e.g., 0.01 or 0.05), then we let

$$\begin{aligned} d_\tau = &\ \mathrm{median}\{a_{\tau,1}, a_{\tau,2}, \ldots, a_{\tau,m}\} \\ &-\mathrm{median}\{b_{\tau,1}, b_{\tau,2}, \ldots, b_{\tau,m}\}, \quad \tau = 1, \ldots, n. \end{aligned}$$

Otherwise (if the NULL hypothesis $H_{\tau,0}$ has not been rejected at a promising significant level), we let $d_\tau = 0$, i.e., we ignore the insignificant difference between the performance scores of A and B. $d_1, d_2, \ldots, d_n$ will then be utilized in the following Wilcoxon Signed-Rank Test for the performance comparison over different benchmarks:

• Let the NULL hypothesis of the SHT be "H_0*: the general performance of A is equivalent to that of B*"; let the alternative hypothesis of the SHT be "H_1*: the general performance of A is better than that of B*" or "H_2*: the general performance of B is better than that of A*", depending on the motivation of carrying out the SHT.

• Rank $d_1, d_2, \ldots, d_n$ according to an ascending order of their absolute values. In case two or more absolute values are the same, then renew the ranks by assigning them the average of their original ranks. Afterwards, for A and B, we can define their signed-rank sums be:

$$\begin{aligned} \mathrm{R}_A &= \sum_{i:d_i>0} \mathrm{Rank}(d_i) + \frac{1}{2}\sum_{i:d_i=0} \mathrm{Rank}(d_i), \\ \mathrm{R}_B &= \sum_{i:d_i<0} \mathrm{Rank}(d_i) + \frac{1}{2}\sum_{i:d_i=0} \mathrm{Rank}(d_i), \end{aligned}$$

where $\mathrm{Rank}(d_i)$ provides the rank of the absolute value of d_i, which was described above.

•**Case[**$n < 25$**]**[4]**:** When the alternative hypothesis of the SHT is H_1, we reject the NULL hypothesis and accept H_1 if R_B is *no larger than* the critical value (one-side Wilcoxon Signed-Rank Test) under the significance level α; When the alternative hypothesis of the SHT is H_2, we reject the NULL hypothesis and accept H_2 if R_A is *no larger than* the critical value under the significance level α. The critical values of Wilcoxon Signed-Rank Test are available in statistics books [15].

•**Case[**$n \geq 25$**]:** Define two new statistics z_A and z_B as follows:

$$z_A = \frac{\mathrm{R}_A - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}}, \ z_B = \frac{\mathrm{R}_B - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}}.$$

Under the NULL hypothesis, z_A and z_B approximately obey the standard normal distribution $\mathcal{N}(0,1)$. Hence, when the alternative hypothesis of the SHT is H_1, we reject the NULL hypothesis and accept H_1 if z_B is *no larger than* the critical value (lower tail, standard normal distribution) under the significance level α; when the alternative hypothesis of the SHT is H_2, we reject the NULL hypothesis and accept H_2 if z_A is *no larger than* the critical value under the significance level α. For the comparison over different benchmarks, the outputs of the HPT, including the comparison result and its confidence, are finally presented by the above Wilcoxon Signed-Rank Test. Formally, given a fixed significance level α for the HPT, we

[2]For example, if two scores are both 50, and their original ranks are 5 and 6 respectively, then both of them obtain a rank of 5.5.

[3]In statistics, when $m < 12$, the critical values for Wilcoxon rank sum test are calculated directly. When $m \geq 12$, the corresponding critical values are often estimated by studying the approximate distribution of the rank sum.

[4]In statistics, when $n < 25$, the critical values for Wilcoxon signed rank test are calculated directly. When $n \geq 25$, the corresponding critical values are often estimated by studying the approximate distribution of the signed rank sum.

utilize $Confidence(\text{HPT} : S_A \succ S_B) \geq r$ to represent the following conclusion made by the HPT: "A outperforms B with the confidence r", where $r = 1 - \alpha$.

4.2 Quantitative Comparison: Statistical Speedup Testing

So far we have shown how to carry out qualitative performance comparison with the HPT. In addition to the qualitative comparison, in most cases we are more interested in quantitative comparison results such as "Computer A is more than γ times faster than Computer B", where $\gamma \geq 1$ is defined as the *speedup-under-test*. Traditionally, such kind of arguments are often obtained directly by comparing the means of performance scores with respect to computers A and B. Taking the SPEC convention as an example, if the mean (geometric) SPEC ratios of A is ten times that of B, then one would probably conclude that "A is ten times faster than B". Such a quantitative comparison is dangerous since we do not know how much we can trust the result. Fortunately, the HPT framework offers two solutions for tackling speedup arguments. The first solution requires us to specify the concrete value of γ before the test. Afterwards, we shrink the performance scores of computer A by transforming the corresponding performance matrix S_A to S_A/γ (without losing generality, we employ the normalized performance ratio as the performance score with respect to each benchmark, where a larger performance score means better performance). Considering a virtual computer with performance matrix S_A/γ, if the HPT framework states that the virtual computer outperforms computer B with confidence r, then we can claim "A is more than γ times faster than B with confidence r". *In general, if we specify a more (less) conservative speedup γ before speedup testing, the corresponding speedup argument will have a larger (smaller) confidence r.* Users should keep a balance between the speedup and the corresponding confidence, so as to make a convincing yet not-too-conservative conclusion.

In many cases, instead of deciding a speedup γ before the statistical test, one would like to know *the largest speedup that results in a reliable comparison result* (for a given confidence r). To address this need, our HPT framework also offers an alternative way of estimating the speedup and corresponding confidence. Guided by the above notion, we formally define the r-Speedup (computer A over computer B, r-Speedup(A, B)) to be

$$\sup\left\{\gamma \geq 1; confidence\left(\text{HPT} : \frac{1}{\gamma}S_A \succ S_B\right) \geq r\right\}.$$

To be specific, the r-Speedup of computer A over computer B is the largest speedup of A over B for confidence r. In practice, we can restrict the precision (e.g., 2 decimals) when estimating the r-Speedup via heuristic optimization techniques. After predefining the confidence level r (e.g., $r = 0.95$), the r-Speedup can be viewed as a quantitative indicator of performance speedup with the guarantee of confidence r. The whole HPT framework, including the r-Speedup, has been implemented, and it will be disseminated as an open-source software [16].

4.3 An Example of Quantitative Performance Comparison

In this subsection, the quantitative performance comparison of two commodity computers, X (Linux OS, 4-core 8-thread Intel i7 920 with 6 GB DDR2 RAM) and Y (Linux OS, 8-core AMD Opteron 8220 with 64 GB DDR2 RAM) is presented as an example of applying the HPT and speedup test. In our experiments, each SPLASH-2 benchmark (8 threads) is repeatedly run 5 times on each computer, using the default workloads of SPLASH-2. By specifying the speedup-under-test γ to be 1.76, we use the HPT to test how reliable is the proposition "Computer X is more than 1.76 times faster than Computer Y" over SPLASH-2. Testing such a proposition is equivalent to testing "Computer $\tilde{X}$ is faster than Computer Y" over SPLASH-2, where $\tilde{X}$ is a virtual computer whose performance scores are always 1/1.76 of the corresponding scores of the real computer X. Table 1 presents the details of the comparison. To be specific, all performance scores are normalized to the first run of computer Y on each benchmark. In order to conduct a quantitative comparison, we divide all performance scores of X by 1.76 times (we store these reduced scores in $S_{\tilde{X}}$), and utilize the HPT to compare the reduced scores against those of computer Y (stored in S_Y). For the τ^{th} benchmark ($\tau = 1, \ldots, n$), "Stat. Win." indicates the winner whose performance on the τ^{th} benchmark is significantly (with confidence 0.95) better. We indicate "$\tilde{X}$" if the reduced performance of X still wins, and we indicate "Y" if the performance of Y wins over the reduced performance of X. In case there is no definite winner, we indicate "Tie". "Med." indicates the median of the five performance scores (of A and B), "Diff." shows the (significant) difference between the median performance scores of A and B, "Rank" shows the rank of the absolute value of d_τ. According to the HPT, the virtual computer $\tilde{X}$ beats computer Y significantly on 8 benchmarks, ties on 2 benchmarks, loses on 4 benchmarks. Following the flow introduced in Section 4.1, the proposed HPT concludes that "Computer $\tilde{X}$ is faster than Computer Y with confidence 0.95", suggesting that "Computer X is more than 1.76 times faster than Computer Y with confidence 0.95" (i.e., the 0.95-Speedup of Computer X over Computer Y is 1.76 over all SPLASH-2 benchmarks), where 0.95 is the statistically acceptable level of confidence.

Table 1. Statistical quantitative comparison of computers X and Y over SPLASH-2 (***Speedup: 1.76***).

	$S_{\tilde{X}}$					Med.	Stat. Win.	Diff. (d_τ)	Rank	Med.	S_Y					
1.barnes	0.53	0.54	0.54	0.53	0.54	0.54	Y	-0.50	10	1.04	1.00	1.05	1.04	1.03	1.04
2.cholesky	0.97	0.95	0.93	0.96	0.96	0.96	Y	-0.03	3	0.99	1.00	0.98	1.01	0.99	0.98
3.fft	0.74	0.76	0.74	0.78	0.76	0.76	Y	-0.27	6.5	1.03	1.00	1.03	1.02	1.05	1.03
4.fmm	1.07	1.03	1.05	1.02	1.05	1.05	Tie	0(0.01)	1.5	1.04	1.00	1.05	1.04	1.04	1.05
5.lu-con	1.29	1.26	1.27	1.27	1.25	1.27	$\tilde{X}$	0.27	6.5	1.00	1.00	1.01	1.02	0.98	1.00
6.lu-ucon	1.46	1.48	1.38	1.53	1.55	1.48	$\tilde{X}$	0.49	9	0.99	1.00	0.96	1.04	0.87	0.99
7.ocean-con	1.17	1.15	0.94	1.16	1.13	1.15	$\tilde{X}$	0.17	5	0.98	1.00	0.91	1.00	0.98	0.86
8.ocean-ucon	1.95	1.98	1.92	1.93	1.93	1.93	$\tilde{X}$	0.95	13	0.98	1.00	0.98	0.97	0.90	0.98
9.radiosity	1.01	1.01	1.01	0.99	1.01	1.01	Tie	0(0.01)	1.5	1.00	1.00	1.00	1.00	1.00	1.00
10.radix	2.47	2.51	2.53	2.44	2.11	2.47	$\tilde{X}$	1.50	14	0.97	1.00	0.86	0.95	1.03	0.97
11.raytrace	1.41	1.39	1.43	1.21	1.37	1.39	$\tilde{X}$	0.32	8	1.07	1.00	1.09	1.07	1.14	1.07
12.volrend	0.92	0.94	0.92	0.92	0.93	0.92	Y	-0.08	4	1.00	1.00	1.00	1.00	1.00	1.00
13.water-ns	1.64	1.66	1.59	1.64	1.63	1.64	$\tilde{X}$	0.69	11	0.95	1.00	0.95	0.84	0.93	0.96
14.water-sp	1.84	1.88	1.78	1.80	1.77	1.80	$\tilde{X}$	0.80	12	1.00	1.00	1.02	0.98	0.87	1.04

5 Experimental Comparisons

5.1 Comparisons Using the Geometric Mean Performance

In traditional quantitative comparisons, the Geometric Mean (GM) of the performance scores of a computer over different benchmarks is often utilized to estimate the performance speedup of one computer over another. In most cases, such comparison results, presented without confidence estimates, are often unreliable. Taking the performance comparison between computers X and Y presented in Section 4.3 as an example, the GM-speedup (the performance speedup obtained by comparing the geometric mean performance score) of computer X over computer Y is 2.14. The corresponding confidence of this performance speedup (estimated by the proposed HPT) is 0.64, which is far less than the acceptable level 0.95. We then perform the following more extensive experiments: we collect the (SPEC CPU2006) performance reports of 14 different computers from SPEC.org [31], and analyze both the 0.95-Speedup (performance speedup estimated by the HPT, with the guaranteed confidence 0.95) and GM-Speedup of one computer over another with the proposed HPT. Table 2 presents the performance speedups and the corresponding confidences over 7 pairs of computers[5]. It can be observed from Table 2 that the GM-Speedup is higher than the 0.95-Speedup on all 7 pairs of computers, and the largest error between the GM-Speedup and 0.95-Speedup can be 56.3%. Meanwhile, compared with the acceptable confidence 0.95, the loss of confidence brought by the unreliable GM-Speedup ranges from 28.4% to 87.4%, showing that all 7 GM-Speedups are rather unreliable.

[5]The 14 computers are: **A1**: Dell Precision T7500 (Intel Xeon); **A2**: ProLiant DL380 G4 (Intel Xeon); **B1**: PowerEdge T710 (Intel Xeon); **B2**: Xserve (Intel Dual-Core Xeon); **C1**: PRIMERGY TX100 S2 (Intel Core i3-550); **C2**: Intel DG965WH motherboard (Intel Core 2); **D1**: Acer AB460 F1 (Intel Xeon); **D2**: Dell Precision 380 (Intel Pentium Exteme Edition); **E1**: Asus P5E3 Premium (Intel Core 2); **E2**: Intel D975XBX motherboard (Intel Pentium Extreme Edition); **F1**: Asus P6T Deluxe (Intel Core i7-920); **F2**: HP Integrity rx6600 (Dual-Core Intel Itanium 2); **G1**: Asus P5E3 Premium (Intel Core 2); **G2**: CELSIUS R550 (Intel Xeon). Their SPEC ratios are available at SPEC.org [31].

5.2 Comparisons Using Parametric Techniques

In this subsection, the proposed non-parametric HPT is compared against two parametric statistical techniques (confidence interval and paired t-test) requiring normally distributed sample mean. Ideally, a fair comparison between a non-parametric technique and a parametric technique requires that both the non-parametric and parametric techniques are built upon the same statistic, the mean of the measurements. However, using this statistic in a non-parametric manner requires more performance measurements, e.g., several hundred different machines, and more sophisticated statistical techniques such as the permutation test or bootstrap. Due to the lack of available performance data on SPEC.org, we leave this improved comparison for future work.

Let us first recall the example presented in Section 4.3. Using the assertion "Computer X is more than 1.76 times faster than Computer Y", we compare the effectiveness of HPT against that of the parametric techniques. Technically, the effectiveness of each technique (HPT, paired t-test and confidence interval) can be statistically measured by the "statistical power" of the technique [9, 11], which is a broadly adopted criterion for comparing the effectiveness of SHTs. The power of a statistical technique for testing the confidence of a hypothesis can be approximately estimated by the probability of accepting the alternative hypothesis [9]. For the comparison result "Computer X is more than 1.76 times faster than Computer Y", the power of each statistical technique can be estimated by carrying out 1000 repeated runs of SPLASH-2 on X and Y, and studying the average confidence of the comparison result over 1000 re-

Table 2. Quantitative Performance Comparisons based on SPEC CPU2006, where the 0.95-Speedups are obtained by the proposed HPT, each GM-Speedup is obtained by comparing the geometric mean SPEC ratios of the corresponding pair of computers, and all HPT-confidences are estimated by the proposed HPT.

	A1-A2	B1-B2	C1-C2	D1-D2	E1-E2	F1-F2	G1-G2
0.95-Speedup	2.64	2.24	1.39	2.45	1.76	1.54	1.15
HPT-Confidence	0.95	0.95	0.95	0.95	0.95	0.95	0.95
GM-Speedup	3.35	3.50	1.70	3.26	1.98	1.67	1.27
Speedup Error	*+26.9%*	*+56.3%*	*+22.3%*	*+33.1%*	*+12.5%*	*+8.4%*	*+10.4%*
HPT-Confidence	0.18	0.31	0.33	0.17	0.12	0.68	0.15
Confidence Loss	*-81.1%*	*-67.4%*	*-65.3%*	*-82.1%*	*-87.4%*	*-28.4%*	*-84.2%*

Table 3. Comparisons of confidences obtained by the HPT and parametric techniques, where the HPT-Confidences are obtained by the proposed HPT, CI-Confidences are obtained by the confidence interval, and t-Confidences are obtained by the paired t-test. According to the statistical convention, the confidences in bold are the acceptable ones (≥ 0.95), and the rest are unacceptable ones (< 0.95).

	A1-A2	B1-B2	C1-C2	D1-D2	E1-E2	F1-F2	G1-G2
Speedup	2.64	2.24	1.39	2.45	1.76	1.54	1.15
HPT-Confidence	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**	**0.95**
CI-Confidence	0.82	0.90	0.91	0.89	0.79	0.73	0.12
Confidence Loss	*-13.7%*	*-5.3%*	*-4.2%*	*-6.3%*	*-16.8%*	*-23.2%*	*-87.4%*
t-Confidence	0.91	**0.95**	**0.96**	0.94	0.89	0.87	0.52
Confidence Loss	*-4.2%*	*0%*	*0%*	*-1.1%*	*-6.3%*	*-8.4%*	*-45.3%*

peated statistical tests using the same technique. According to our experiments, the power of HPT, estimated over 1000 repeated runs of SPLASH-2 on computers X and Y, is 0.89, which is significantly larger than that of the confidence interval (0.67), and that of the paired t-test (0.76). Statistically, this shows that HPT significantly outperforms the paired t-test for the performance comparison of computers.

Now, we carry out another experiment using the data collected from SPEC.org [31], with the goal of showing that the parametric techniques can be rather inaccurate compared to HPT. Our empirical study still involves the 7 aforementioned pairs of computers. By fixing the performance speedup for each pair of computers, Table 3 compares the confidence obtained by the HPT with those obtained by parametric techniques. We can observe that the confidence provided by the confidence interval technique is often rather inaccurate, and the largest confidence estimate error is 87.4%. At the acceptable confidence level of 0.95, the confidence interval technique is so conservative that all 7 comparison results are incorrectly considered to be unreliable (while the HPT accepts all of them). The paired t-test performs slightly better, though it still rejects 5 out of the 7 comparison results due to the inaccurate t-Confidence. In summary, using inappropriate statistical techniques can simply result in incorrect conclusions on the validity of a performance comparison.

Finally, we also compare the performance speedups obtained by parametric techniques against those obtained by HPT when the level of confidence is set to 0.95. As presented in Table 4, the speedup error with respect to the confidence interval ranges from 6.5% to 44.7%, while the error with respect to the t-test ranges from 1.4% to 21.6%. Again, these results highlight the impact of statistical techniques on the outcome of performance comparisons.

6 Related Work

Traditionally, performance comparisons of computers mainly rely upon one metric (e.g., geometric mean and harmonic mean) [8, 17, 20, 25, 30], though this approach can be rather unreliable. Having realized the importance of statistical inference, Lilja suggested to introduce several parametric statistical methods (e.g, confidence interval) to evaluate computer performance [23]. Alameldeen and Wood carried out in-depth investigations on the performance variability of multi-threaded programs, and they suggested to use the confidence interval and t-test, two parametric techniques, in order to address the issue of variability [1]. Later, in the context of Java performance evaluation, Georges et al. [12] found that single-benchmark per-

Table 4. Comparisons of 0.95-performance speedups obtained by the HPT and parametric techniques, where each speedup has a confidence of 0.95 for each technique.

	A1-A2	B1-B2	C1-C2	D1-D2	E1-E2	F1-F2	G1-G2
0.95-Speedup (HPT)	2.64	2.24	1.39	2.45	1.76	1.54	1.15
0.95-Speedup (Confidence Interval)	1.46	1.41	1.30	1.79	1.45	1.29	1.02
Speedup Error	*-44.7%*	*-37.1%*	*-6.5%*	*-26.9%*	*-17.6%*	*-16.2%*	*-11.3%*
0.95-Speedup (Paired t-test)	2.07	2.28	1.41	2.39	1.59	1.38	1.04
Speedup Error	*-21.6%*	*+1.8%*	*+1.4%*	*-2.45%*	*-9.7%*	*-10.4%*	*-9.6%*

formance (SPECjvm98) on several single-core computers can, in general, be characterized using normal distributions, and thus, they can recommend using the confidence interval technique in this case. While valid, their observation does not seem to generalize to the broader case of multi-core systems and multi-threaded applications, based on our own experiments. Iqbal and John's empirical study [19] generally supported the log-normality for characterizing the SPEC performance of computers. However, their experiments were conducted after removing all "outlier benchmarks" in SPEC CPU2006. They also proposed a performance ranking system [19]. But unlike the Wilcoxon test which uses rank information to construct statistics for computing confidence, the system directly offers a performance ranking without presenting the corresponding confidence.

However, we observed that few computer architecture studies yet acknowledge the importance of proper confidence estimates for performance comparisons and measurements: among 521 papers surveyed at ISCA (194 papers, 2006–2010), HPCA (158 papers, 2006–2010) and MICRO (169 papers, 2006–2009), only 28 papers (5.4%) resort to confidence estimates in order to assess the variability of performance measurements, among which 26 (5%) rely upon the confidence interval technique, and only 3 (0.57%) use the more sophisticated but still parametric t-test.

At the same time, many other statistical techniques have already been used to cope with various issues in computer architecture research. For instance, statistical techniques were used for sampling simulation [36], principal components analysis was used to evaluate the representativeness of benchmarks [3, 29], and regression techniques were used to model the design space of processors [10, 13, 18, 22]. Therefore, the computer architecture community is already largely familiar with complex statistical tools, so that embracing a more rigorous performance measurement and comparison process is only a logical extension of the current trend.

7 Conclusion

We first highlight the importance and impact of variability in performance measurements and comparisons, as well as the risk of inappropriately using current parametric confidence techniques, and the fact they require a large number of performance measurements when applied to multi-cores and multi-threaded applications, making them largely impractical.

We propose a framework for achieving both a rigorous and practical comparison of computer architecture performance. In the proposed HPT, we adopt non-parametric SHTs which do not require a normal distribution of the sample mean of performance measurements, and thus, which can accommodate few such measurements. Besides the benefits for performance comparisons, we have implemented the HPT as an easy-to-use open-source software, requiring no mathematical background.

Acknowledgements

We are grateful to Babak Falsafi and to the anonymous reviewers for their helpful comments and insightful suggestions. We also thank Yang Chen and Chengyong Wu for sharing KDataSets with us.

References

[1] A. R. Alameldeen and D. A. Wood, "Variability in Architectural Simulations of Multi-Threaded Workloads", in *HPCA-9*, 2003.

[2] T. Bergan, O. Anderson, J. Devietti, L. Ceze, and D. Grossman, "CoreDet: A Compiler and Runtime System for Deterministic Multithreaded Execution", in *ASPLOS-15*, 2010.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", in *PACT'08*, 2008.

[4] L. Le Cam, "The Central Limit Theorem Around 1935", *Statistical Science* 1(1), 1986.

[5] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey, *Graphical Methods for Data Analysis*, Wadsworth, 1983.

[6] Y. Chen, W. Hu, T. Chen, and R. Wu, "LReplay: A Pending Period Based Deterministic Replay Scheme", In *ISCA-37*, 2010.

[7] Y. Chen, Y. Huang, L. Eeckhout, G. Fursin, L. Peng, O. Temam, and C. Wu, "Evaluating iterative optimization across 1000 datasets", in *PLDI'10*, 2010.

[8] D. Citron, A. Hurani, and A. Gnadrey, "The harmonic or geometric mean: does it really matter?", *ACM SIGARCH Computer Architecture News* 34(4), 2006.

[9] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research* 7, 2006.

[10] V. Desmet, S. Girbal, and O. Temam, "ArchExplorer.org: A methodology for facilitating a fair Comparison of research ideas", in *ISPASS'10*, 2010.

[11] P. D. Ellis, *The Essential Guide to Effect Sizes: An Introduction to Statistical Power, Meta-Analysis and the Interpretation of Research Results*, Cambridge University Press, 2010.

[12] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation", in *OOPSLA'07*, 2007.

[13] Q. Guo, T. Chen, Y. Chen, Z.-H. Zhou, W. Hu, and Z. Xu, "Effective and Efficient Microprocessor Design Space Exploration Using Unlabeled Design Configurations", In *IJCAI'11*, 2011.

[14] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite", In *Proceedings of the IEEE 4th Annual International Workshop on Workload Characterization (WWC)*, 2001.

[15] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*, Prentice Hall, 2005.

[16] HPT Software: http://novel.ict.ac.cn/tchen/HPT

[17] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufman, 4th Edition, 2007.

[18] E. Ïpek, S. A. McKee, R. Caruana, B. R. Supinski, and M. Schulz, "Efficiently Exploring architectural design spaces via predictive modeling", in *ASPLOS-12*, 2006.

[19] M. F. Iqbal and L. K. John, "Confusion by All Means", in *Proceedings of the 6th International Workshop on Unique Chips and Systems (UCAS-6)*, 2010.

[20] L. K. John, "More on Finding a Single Number to Indicate Overall Performance of a Benchmark Suite", *ACM SIGARCH Computer Architecture News* 32(1), 2004.

[21] R. A. Johnson and G. K. Bhattacharyya, *Statistics: Principles and Methods*, John Wiley & Sons, 2009.

[22] B. C. Lee, J. Collins, H. Wang, and D. Brooks, "Cpr: Composable performance regression for scalable multiprocessor models", in *MICRO-41*, 2008.

[23] D. J. Lilja, *Measuring computer performance: a practitioner's guide*, Cambridge University Press, 2000.

[24] H. W. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown", *Journal of the American Statistical Association* 62, 1967.

[25] J. R. Mashey, "War of the benchmark means: time for a truce", *ACM SIGARCH Computer Architecture News* 32(4), 2004.

[26] P. Montesinos, M. Hicks, S. T. King, and J. Torrellas, "Capo: a software-hardware interface for practical deterministic multiprocessor replay", in *ASPLOS-14*, 2009.

[27] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong", in *ASPLOS-14*, 2009.

[28] E. Parzen, "On estimation of a probability density function and mode", *Annals of Mathematical Statistics* 33, 1962.

[29] A. Phansalkar, A. M. Joshi, L. Eeckhout, and L. K. John, "Measuring Program Similarity: Experiments with SPEC CPU Benchmark Suites", in *ISPASS'05*, 2005.

[30] J. E. Smith, "Characterizing computer performance with a single number", *Communications of the ACM* 31(10), 1988.

[31] SPEC, http://www.spec.org/.

[32] Student (W. S. Gosset), "The probable error of a mean", *Biometrika* 6(1), 1908.

[33] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations", in *ISCA-22*, 1995.

[34] J. Suh and M. Dubois, " Dynamic MIPS rate stabilization in out-of-order processors ", in *ISCA-36*, 2009.

[35] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics* 1(6), 1945.

[36] R. E. Wunderlich, T. F. Wenisch, B. Falsafi and J. C. Hoe, "SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling", in *ISCA-30*, 2003.