Precise Computer Performance Comparisons Via Statistical Resampling Methods

Bin Li, Shaoming Chen, and Lu Peng
Louisiana State University
{bli, schen26, lpeng}@lsu.edu

***Abstract*—Performance variability, stemming from non-deterministic hardware and software behaviors or deterministic behaviors such as measurement bias, is a well-known phenomenon of computer systems which increases the difficulty of comparing computer performance metrics. Conventional methods use various measures (such as geometric mean) to quantify the performance of different benchmarks to compare computers without considering variability. This may lead to wrong conclusions. In this paper, we propose three resampling methods for performance evaluation and comparison: a randomization test for a general performance comparison between two computers, bootstrapping confidence estimation, and an empirical distribution and five-number-summary for performance evaluation. The results show that 1) the randomization test substantially improves our chance to identify the difference between performance comparisons when the difference is not large; 2) bootstrapping confidence estimation provides an accurate confidence interval for the performance comparison measure (e.g. ratio of geometric means); and 3) when the difference is very small, a single test is often not enough to reveal the nature of the computer performance due to the variability of computer systems. We further propose using empirical distribution to evaluate computer performance and a five-number-summary to summarize computer performance. We illustrate the results and conclusion through detailed Monte Carlo simulation studies and real examples. Results show that our methods are precise and robust even when two computers have very similar performance metrics.**

Keywords— Performance of Systems; Performance attributes; Measurement, evaluation, modeling, simulation of multiple-processor systems; Experimental design

I. INTRODUCTION

Traditionally, computer researchers have used the geometric mean (GM) of performance ratios of two computers running a set of selected benchmarks to compare their relative performances. This approach, however, is limited by the variability of computer systems which stems from non-deterministic hardware and software behaviors [1][12], or deterministic behaviors such as measurement bias [20]. The situation is exacerbated by increasingly complicated architectures and programs. Wrong conclusions could be drawn if variability is not handled correctly. Using a simple geometric mean cannot describe the performance variability of computers.

Recently, computer architects have been seeking advanced statistical inferential tools to address the problem of performance comparisons of computers. The two common statistical approaches of comparing two populations (e.g., two computers) are the hypothesis test and confidence interval estimation. As we know, most of the parametric tests such as t-tests require population distribution normality [11]. Unfortunately, computer performance measurements are often not normally distributed but either skewed or multimodal. Figure 1 shows 400 measurements of execution time from SPEC2006 benchmarks running on a commodity computer (Intel Core i7 CPU 960@3.20GHz, 1 processor with 4 cores, 10GB DDR3 RAM(1333 MHz)). We can see that the distributions of performance measures for the benchmarks are non-normal; benchmarks "gcc" and "mcf" are skewed to the right, while "bzip2" is multimodal.

In this paper, we propose three statistical resampling methods [14] to evaluate and compare computer performance. The first is a randomization test used to compare the performance between two computers; the second is a bootstrapping confidence interval method for estimating the comparative performance measurement, i.e. speedup, through a range; and the third is an empirical distribution method to evaluate the distributional properties of computer performance. The basic idea of resampling methods, as the name implies, is to resample the data iteratively, in a manner that is consistent with certain conditions (e.g. the general performance of two computers is equal.). Specifically, we first resample the data according to the purpose of each method. Second, for each iteration, we calculate the statistic of interest, such as the ratio of geometric means between two computers. Third, we repeat the previous two steps a number of times. Then the distribution of the calculated statistic is used as an approximation of the underlying distribution of the statistic under the assumed condition. Hence, the resampling methods set us free from the need for normal data or large samples so that Central Limit Theorem can be applied [19]. Note that the proposed three methods all follow the three steps described above. However, the resampling and calculating steps within each iteration are different according to the individual purpose for each method.

In summary, the main contributions of this paper can be listed as follows:

First, we propose and implement a randomization test [8] for testing the performances of two computers, which provides an accurate estimate of the confidence of a comparison when the performances of two computers are close to each other.

Second, we propose and implement a bootstrapping-based confidence interval estimation method [6] to estimate the confidence interval of the ratio of geometric means between two computers. As a result, we show that the confidence interval of the ratio of the geometric means between two computers can reliably summarize the comparative performance between them in the context of performance variation.

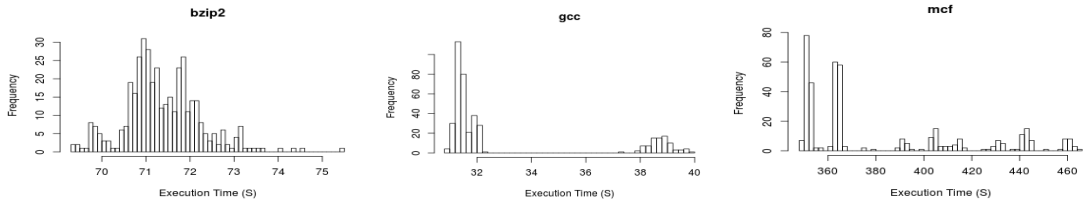

Figure 1. Histograms of execution times for three SPEC benchmarks from 400 repeated runs of each benchmark on the commodity computer.

Table 1. Configurations of the two computers in Figure 2.

	Configurations
Middle (blue dashed line)	NovaScale T860 F2 (Intel Xeon E5645, 2.40 GHz)
Middle (red solid line)	IBM System x3400 M3 (Intel Xeon E5649)

Table 2. Test results for the example in Figure 2.

T test p-value	Randomization test p-value	95% Bootstrapping
0.117	0.016	[0.974, 0.997]

Third, as a generic framework, the proposed method can directly be applied to arithmetic and harmonic means. We demonstrate that the arithmetic mean is very sensitive to outliers while geometric and harmonic means are much more stable.

Fourth, we point out that a single test is not enough to reveal the nature of the computer performance in some cases due to the variability of computer systems. Hence, we suggest using empirical distribution to evaluate computer performance and use five-number-summary to summarize the computer performance.

The remainder of this paper is organized as follows. We provide a motivating example to demonstrate the different results that can be generated from a t-test and the proposed resampling methods in Section 2. Then we describe the detailed algorithms of the proposed randomization test and confidence interval estimation in Sections 3 and 4 respectively. We suggest using an empirical distribution and five-number-summary to compare computer performances in Section 5. Section 6 presents the experimental results on data measured from our lab computers and collected from SPEC.org. Section 7 explains the sample size selection. We demonstrate the applicability of the proposed resampling methods on Arithmetic and Harmonic Means in Section 8. Related work is described in Section 9. Finally, we conclude the paper in Section 10.

II. MOTIVATION EXAMAPLE

In this section, we show an example of comparing two computers based on t-test and the proposed resampling methods. Table 1 lists the configurations of the computers. The data is available on [26]. Figure 2 shows the empirical distributions of geometric mean for two computers. The horizontal axis shows the SPEC ratio. The blue dash line is the empirical distribution of geometric means for the NovaScale computer, while the red solid line is the one from IBM. The vertical dash line shows the geometric mean from the raw data. The basic idea of using an empirical distribution is to see the distribution of a statistic (e.g. geometric mean of computer performance). We can see many useful distributional properties from the empirical distribution, such as the center, mode, variation, and range of the statistic. The details of empirical distribution are described in Section 5. From Figure 2, although the two distributions overlap, the geometric mean of computer A (red solid curve) is well above that of computer B (blue dash curve). As shown in Table 2, the t-test does not detect the difference between two computers while the randomization test does. This implies that the randomization test is more powerful at detecting the difference even when there is an overlap between two distributions. The bootstrap interval also shows the ratio of geometric means is significantly below one (blue dashed curve against red solid curve) which implies that computer A runs faster than computer B.

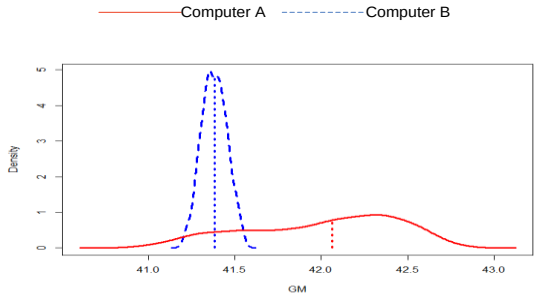

Figure 2. Density plots of the empirical distributions for the two computers. The Dotted lines are the geometric means.

III. STATISTICAL PERFORMANCE COMPARISON VIA RANDOMIZATION TEST

Statistical inference is based on the sampling distributions of sample statistics which answers the question: "if we recollect the data, what will the statistic be?" A sampling distribution of a statistic (e.g. geometric mean) can be well approximated by taking random samples from the population. Traditional parametric tests assume the sampling distribution has a particular form such as a normal distribution. If the distributional assumption is not satisfied, commonly there are no theoretical justifications or results available. On the other hand, the great advantage of resampling is that it often works even when there is no theoretical adjustment available. The basic idea of the randomization test [8] is as follows: in order to estimate the *p-value* (i.e. 1- confidence) for a test, we first estimate the sampling distribution of the test statistic given the null hypothesis is true. This is accomplished by resampling the data in a manner that is consistent with the null hypothesis. Therefore, after resampling many times, we can build up a distribution (called an empirical distribution) which approximates the sampling distribution of the statistic that we are interested in. Thus, we can estimate the *p-value* based on the empirical distribution.

Suppose we have two computers A and B to compare over a benchmark suite consisting of n benchmarks. For each computer, we ran the benchmarks m times and denote the performance scores of A and B at their jth runs of the ith benchmark as $a_{i,j}$ and $b_{i,j}$ respectively. The hypotheses are specified below.

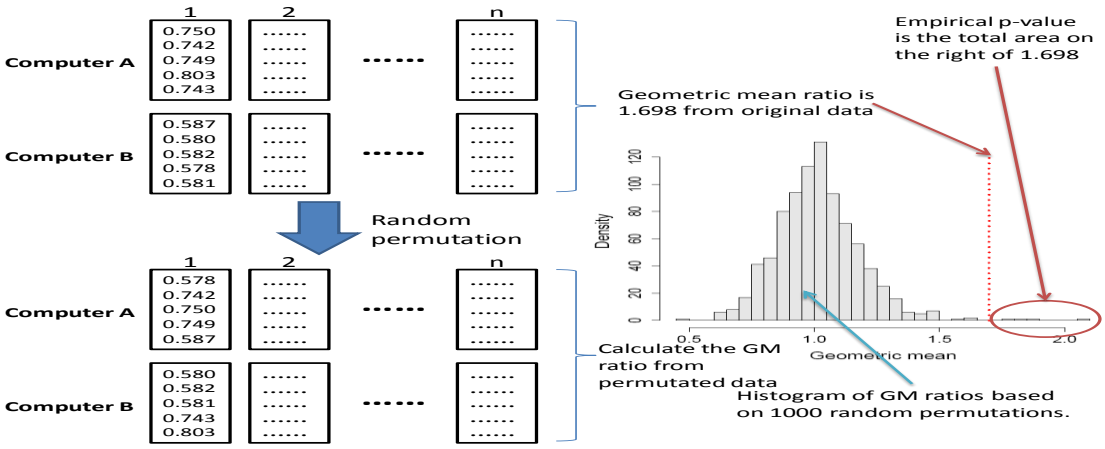

Figure 3. Illustration of the proposed randomization test.

Null hypothesis: the general performance of A and B over n benchmarks are equivalent.

Alternative hypothesis: we will use only one of the following three as our alternative hypothesis.

H_{1a}: the general performance of A is better than that of B.

H_{1b}: the general performance of B is better than that of A.

H_{1c}: the general performance of A is not the same as that of B.

We proposed the randomization test as follows:

1) For each benchmark i (i=1,…,n), we combine all the m performance scores from A and B into one list respectively.

2) We randomly permute the list, for each benchmark, and assign the first m scores to computer A and the other m to B for the i^{th} benchmark.

3) Calculate the ratio of the geometric mean of the performance scores for computer A and B over n benchmarks.

4) Repeat step 1-3 M times (M is usually a large number, e.g. 500), so we have M geometric mean ratios, denote as FM (i.e. the empirical distribution of geometric mean ratios under the null hypothesis) from M repetitions.

5) Calculate $g_{A|B}$, the ratio of the geometric mean of the performance scores for computer A and B over n benchmarks on the original data. Then we calculate an empirical *p-value* based on F_M and the alternative hypothesis as follows. If we use H_{1a}, then the empirical *p-value* is the proportion of F_M that is greater than or equal to $g_{A|B}$. If H_{1b} is selected, then the empirical *p-value* is the proportion of F_M that is less than or equal to $g_{A|B}$. If we use H_{1c}, then the empirical *p-value* is the twice of the smaller empirical *p-value* from H_{1a} and H_{1b}.

Figure 3 illustrates the proposed randomization test under the alternative H_{1a}. Note that the randomization test described above uses the geometric mean to evaluate the computer performance. However, the proposed method can be easily modified to adopt other measures such as harmonic and arithmetic mean.

IV. CONFIDENCE INTERVAL ESTIMATION BY BOOTSTRAPPING

Due to the performance variability, the comparative performance measure, such as the ratio of geometric means and speedups, between two computers varies on different measurements. Hence, presenting a single numeric estimate cannot describe the amount of uncertainty due to the performance variability. The basic idea of a confidence interval (CI) is to provide an interval estimate (which consists of a lower limit and an upper limit) on the statistic with some predetermined confidence level, instead of giving a single estimate. The interpretation of a confidence interval is based on recollecting the data or repeating the experiment.

Bootstrapping [6] is a commonly used statistical technique which quantifies the variability of a statistic, e.g. estimate a 95% confidence interval of a statistic or its standard deviation, which are not yet available in theory [9]. The basic idea of bootstrapping is to use the sample as an approximation of the underlying population distribution, which is unknown, and resample the data with replacement (note that each observation can be sampled more than once). We proposed the following bootstrapping method to estimate the ratio of the geometric mean of the performance scores from two computers.

1) For each benchmark *i (i=1,…,n),* we combine all the m execution times from computer A and B into one list respectively.

2) We randomly sample the list with replacement for each benchmark, and assign the first m scores to computer A and the other m to B for the ith benchmark.

3) Calculate the ratio of the geometric mean of the execution times for computer A and B over n benchmarks.

4) Repeat step 1-3 T times (T is usually a large number, e.g. 500), so we have T geometric mean ratios, denote as H_T from T repetitions. Let $H_T^{\alpha/2}$ and $H_T^{1-\alpha/2}$ be the α/2 and 1-α/2 percentiles of H_T respectively. Then, a two-sided (1-α)×100% bootstrap confidence interval is $\left[H_T^{\alpha/2}, H_T^{1-\alpha/2}\right]$. A one-sided (1-α)×100% bootstrap confidence interval can be either

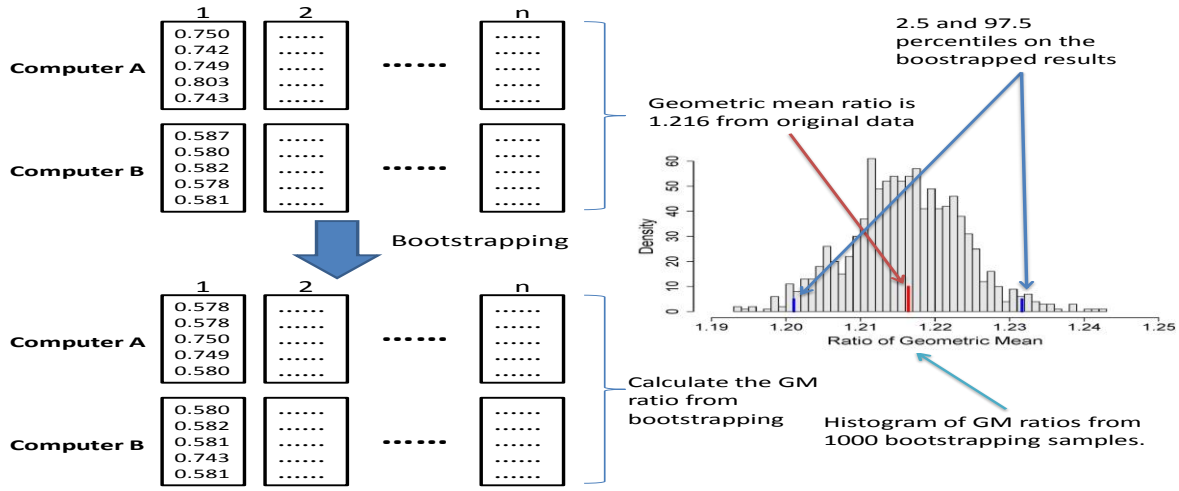

Figure 4. Illustration of proposed bootstrapping confidence interval estimation.

$[H_T^{\alpha}, +\infty]$ or $[-\infty, H_T^{1-\alpha}]$. The former one-sided confidence interval is explained as the ratio of GMs between computer A and B is at least H_T^{α} with confidence (1-α)×100%, while the latter as the ratio of GMs between computer A and B is at most $H_T^{1-\alpha}$ with confidence (1-α)×100%. Figure 4 illustrates the proposed bootstrapping method using an example.

V. Empirical Distribution and Five-Number-Summary

Although the proposed randomization test demonstrates more precise than conventional t-test, when two computers show overlapped distributions and close geometric mean, a single test such as t-test and randomization test can't identify their differences. Figure 5 shows three pairs of computers listed in Table 3. The *p-values* of both t-test and randomization test for all the three pairs are close to 1.0. For example, the p-values are 0.941 and 0.856 for t-test and randomization test respectively for the two computers shown in Figure 5(a). Similar situations also apply to the pairs in Figure 5(b) and 5(c). This indicates no performance differences could be identified by a single test. On the other hand, an insignificant test result does not necessarily mean the two computers have the same performance. For example, in Figure 5 we see that all three computers depicted by red solid lines have slightly higher geometric means than their competitors, but their performances are less consistent than the ones shown by blue dashed lines. Therefore in comparing performance, we need to consider the system variation effect especially when the means are close.

Hence, we suggest using the empirical distribution of the geometric mean and its five-number-summary to describe of performance for a computer as follows:

1) For each benchmark i (i=1,…,n), we randomly select one performance score.

2) Calculate the geometric mean of the performance score for this computer.

3) Repeat step 1-2 M times (M is usually a large number, e.g. 500), so that we have M geometric means, denoted as FG R(i.e. the empirical distribution of geometric mean) from M repetitions.

Table 3. Configurations of three pairs of computers in Figure 5.

	Configurations
Figure 5(a) (blue dashed line)	PowerEdge R510 (Intel Xeon E5620, 2.40 GHz)
Figure 5(a) (red solid line)	IBM BladeCenter HS22 (Intel Xeon X5550)
Figure 5(b) (blue dashed line)	SuperServer 5017C-MF (X9SCL-F, Intel G850)
Figure 5(b) (red solid line)	Acer AW2000h-AW170h F1(Intel Xeon X5670)
Figure 5(c) (blue dashed line)	IBM System x3850 X5 (Intel Xeon E7-4820)
Figure 5(c) (red solid line)	IBM System x3690 X5 (Intel Xeon E7-2830)

4) Then calculate the five elements of the five-number-summary of FG: minimum, first quartile (25th percentile, denoted as Q1), median, third quartile (75th percentile, denoted as Q3), and maximum.

Detailed results will be shown in section VI.E.

VI. Experimental Results

A. Monte Carlo Simulation Study on Statistical Power and False Discovery Rates (FDRs)

In order to show the effectiveness of a testing method, we examine the statistical power (the ability to detect an effect, i.e. deviation from the null hypothesis) and the false discovery rate which is the probability of having type I error (i.e. rejecting the null hypothesis while the null hypothesis is true) of our proposed method, t-test, and a recent proposed HPT approach [3]. A common way to evaluate and compare the statistical powers and false discovery rates (FDRs), which are defined below, of the tests is through Monte Carlo simulation study.

Statistical power: the probability of rejecting the null hypothesis while the null hypothesis is, in fact, not true. Note that we denote power as statistical power in this paper.

False discovery rates: the probability of rejecting the null hypothesis while the null hypothesis is, in fact, true.

Hence, ideally we would like the statistical power to be as large as possible and the FDR as small as possible. In real examples, we usually do not know the underlying truth. In order

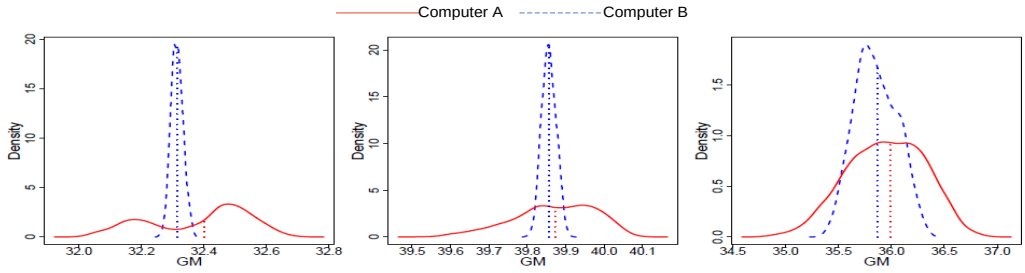

Figure 5. Density plots of the empirical distributions for three pairs of computers. The dot lines are the geometric means.

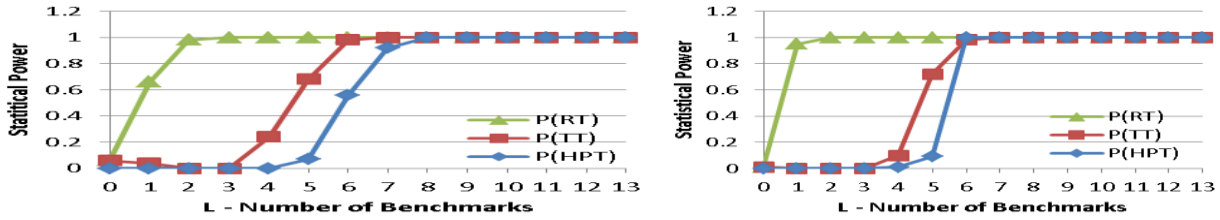

Figure 6. Results of Monte Carlo simulation study 1 (part (a)) and study 2 (part (b)) on statistical power and FDR.

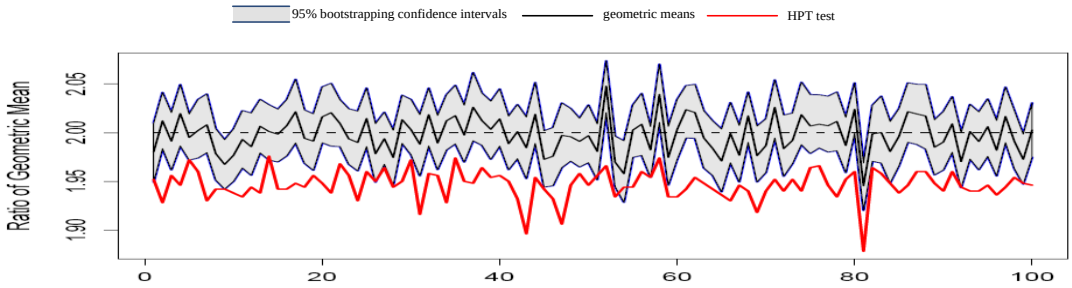

Figure 7. The 95% bootstrapping confidence intervals (boundaries of shaded region), measured ratios of geometric means performance speedups (solid line within the confidence interval) and 0.95-speedups from HPT test (red lines) based on 100 random replications.

to investigate the properties of HPT, t-test, and randomization test we applied a Monte Carlo simulation study where the truth is known. Below are the settings for the Monte Carlo simulation study on power and FDR for two imaginary computers X and Y that uses the following steps.

a. For each benchmark running on computer X, we randomly select m (m=5 in this study) execution times without replacement (i.e. each execution time can be selected at most once) from the 1000 execution times measured from that benchmark running on computer A shown in Table 4.

b. Then we randomly pick L (L is between 0 and 13) benchmarks and add a constant 1.0 to all the execution times for the selected L benchmarks running on the real computer, and assign the sum to be the execution time of the benchmarks running on Computer Y. The reason that we use constant 1.0 in step b to make a difference between two computers is that the standard deviations of the performance from all 13 benchmarks range from 0.012 to 0.91. Hence, adding 1.0 to any benchmark can guarantee that there is at least one standard deviation difference between computer X and Y.

c. The HPT test, t-test, and our proposed randomization test are carried out on the data generated through steps a & b.

d. Repeat steps a-c 100 times.

Remarks: In step a, notice that the execution times for computer X and Y are selected from the same population (from the selected commodity computer). In step b, if L is greater than zero, then the truth is computer X has better performance than computer Y which has longer execution times for the L benchmarks. It is ideal if the test can detect the difference by rejecting the null hypothesis (i.e. the general performance of X is better than that of Y). Hence, P, the proportion of times (among 100 repetitions) a test rejects the null hypothesis, can be viewed as an approximate estimate of its power for nonzero L. On the other hand, when L is zero, that proportion, P, becomes an estimate of its FDR.

In this study, we set the significance level at 0.05 and use the two-sided alternative hypothesis (H_{1c}). Figure 6(a) shows the Monte Carlo simulation results (i.e. P, the proportion of times the null hypothesis is rejected) on HPT, t-test (TT) and the proposed randomization test (RT) using the execution time measurements from the selected computer as the underlying population. Notice that the first point (L=0), the value of P is an estimate of the FDR, which should be close to the specified significance level (here it is 0.05) for a good test. For other points (L=1,…,13), the value of P is an estimate of the power, which is supposed to be large for a good test. So we can see that our proposed randomization test has much higher power than the other two tests when L is between one and seven.

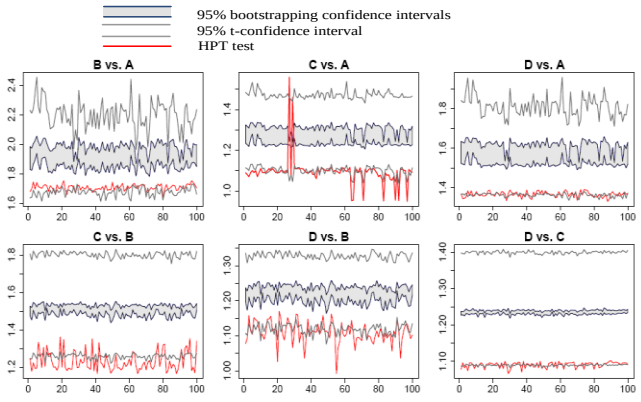

Figure 8. The 95% bootstrapping confidence intervals (boundaries of shaded region), 0.95-speedups from HPT test (red lines) and 95% t-confidence interval (grey lines) on six pairwise comparisons among Computer A, B, C and D from 100 replications.

When L is greater than seven, all tests achieve perfect power. When L is zero, the FDRs for all tests are small and close to the specified significance level (here it is 0.05).

Without losing generality, we also repeat the above described Monte Carlo study by using the measurements from computer C shown in Table 4 running with PARSEC in step a. Figure 6(b) shows the Monte Carlo simulation results (i.e. the proportion of times the null hypothesis is rejected) on HPT, TT, and the proposed RT using execution time measured from another computer as the underlying population. From this figure, similar observations can be made. When L is between 1 and 5, RT demonstrates stronger statistical power than HPT does. This is because, unlike our proposed RT, HPT is calculated using rank-based nonparametric tests (i.e. using Wilcoxon rank-sum test in Step 1 and Wilcoxon signed-rank test in Step 2). In statistics it is well known that the statistical power for the nonparametric tests based on ranks are usually less likely to detect the effects due to the loss of some information on magnitude by ranking [10]. Regarding the t-test, we see it starts to have positive power when L is four and reaches the perfect power when L becomes seven. In fact, t-test shows higher power than the HPT when L is between four and seven. The reason is that the parametric tests are usually more efficient (i.e. higher power) than their nonparametric rank-based counterparts which was used in the HPT method [21].

Thanks to high performance computers, the proposed randomization test (with M=500) takes an average CPU timing of 0.41 seconds running on a regular Dell workstation with an Intel Xeon 2.66GHz processor for the above experiment. The algor ithm is implemented as R language functions.

B. Monte Carlo Simulation Study on Confidence Interval

Like the Monte Carlo simulation in Section VI.A, we also investigate the property of the proposed bootstrapping confidence interval and HPT speedup-under-test estimate from a simulation with known data generation mechanism. Below are the settings for the Monte Carlo simulation study on two imaginary computers X and Y.

a. For each benchmark running on computer X, we randomly select m (m=5 in this study) execution times without replacement from the 1000 execution times measured from that benchmark running on computer A shown in Table 4.

b. Then we multiply all the execution times (all n benchmarks) of computer X by a constant 2.0. We assign the new values as execution times for computer Y.

c. The 95% speedups from HPT test and the proposed 95% bootstrapping confidence intervals are carried out on the data generated through step a & b.

d. Repeat step a-c 100 times.

Table 4. Configurations of the four commodity computers.

Computer	Configurations
A	AMD Opteron CPU 6172 @ 2.10GHz, 2 processors, each with 12 cores, with 12GB DDR3 RAM(1333 MHz)
B	Intel Core i7 CPU 960 @ 3.20GHz, 1 processor with 4 cores (Hyperthreading enabled), 10GB DDR3 RAM(1333 MHz)
C	Intel Xeon CPU X5355 @ 2.66GHz, 2 processors, each with 4 cores, 16GB DDR2 RAM (533MHz)
D	Intel Xeon CPU E5530 @ 2.40GHz, 2 processor, each with 4 cores, 12GB DDR3 RAM (1333MHz)

Table 5. Results of pairwise comparison among four computers based on 100 random replications. The numbers shown in the table are the number of times the null hypothesis is rejected at the significance level 0.01 (the numbers in the parenthesis are for the significance level at 0.05).

Comparison	B vs. A	D vs. A	C vs. A	D vs. B	C vs. B	D vs. C
HPT	100 (100)	100 (100)	5 (91)	90 (99)	100 (100)	99 (100)
T-test	100 (100)	100 (100)	91 (100)	100 (100)	100 (100)	100 (100)
RT	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)

Figure 7 shows the one hundred 0.95-Speedups from HPT test (red curves) and the proposed 95% bootstrapping confidence intervals (blue curves on the boundaries with the grey region in the middle). The black dashed line is the true ratio, 2, and the solid black line is the measured ratio of geometric mean. Note that the t-test confidence interval (t-interval), which is not shown in Figure 7, is much wider than the bootstrapping confidence interval and outside the range of the plot. This implies our bootstrapping confidence interval is more accurate than t-interval. Based on Figure 7, we have the following remarks.

1) Among all 100 bootstrapping confidence intervals, there are ninety-five intervals holding the true value, 2, which follows the pre-specified confidence level, 95%.

2) We see that the 0.95-Speedups from HPT test are consistently below the true value and the bootstrapping confidence intervals (lower than most of the lower limits of the bootstrapping CIs). This is because of the low power for the rank-based nonparametric tests.

3) The measured ratio of geometric mean varies around the true value 2 and falls within the bootstrapping CIs. This implies the ratio of geometric means is still a good estimate of comparative performance between two computers.

We also performed the above experiment on other commodity computers (listed in Table 4). The results are similar to Figure 7. The Bootstrapping method also runs fast in R. It takes an average time of 0.51 seconds running on a Dell workstation

Table 6. Quantitative comparisons of 0.95-performance speedups obtained by HPT, the 95% confidence intervals obtained from t-test, and bootstrapping method.

	A1-A2	B1-B2	C1-C2	D1-D2	E1-E2	F1-F2	G1-G2
GM Speedup	3.339	3.495	1.698	3.259	1.984	1.675	1.27
HPT Speedup	2.64	2.24	1.39	2.45	1.76	1.546	1.15
T-interval	[2.626,4.245]	[2.364,5.167]	[1.417,2.035]	[2.540,4.182]	[1.733,2.272]	[1.429,1.964]	[1.139,1.417]
Bootstrap CI	[3.326,3.352]	[3.476,3.513]	[1.696,1.700]	[3.257,3.262]	[1.983,1.986]	[1.674,1.676]	[1.268,1.273]

Table 8. Comparative summary results on comparing another seven pairs of computers.

	H1-H2	I1-I2	J1-J2	K1-K2	L1-L2	M1-M2	N1-N2
GM Speedup	1.122	1.135	1.127	1.318	1.11	1.13	1.167
HPT confidence HPT Speedup	0.732 0.950	0.868 0.928	0.576 0.944	0.885 0.962	0.753 0.94	0.804 0.908	0.825 0.932
T confidence T-test CI	0.849 [0.956,1.316]	0.896 [0.973,1.325]	0.878 [0.967,1.314]	0.975 [1.037,1.675]	0.814 [0.948,1.298]	0.872 [0.963,1.325]	0.891 [0.964,1.413]
RT confidence Bootstrap CI	>0.999 [1.117,1.126]	>0.999 [1.13, 1.14]	>0.999 [1.117,1.138]	>0.999 [1.31,1.325]	>0.999 [1.109, 1.11]	>0.999 [1.127,1.132]	>0.999 [1.166,1.168]

Table 7. Configurations of another seven pairs of computers.

Computer 1	Computer 2
H1: Fujitsu, CELSIUS R570, Intel Xeon E5506	H2: Fujitsu Siemens Computers, CELSIUS M460, Intel Core 2 Quad Q9550
I1: Fujitsu, CELSIUS R570, Intel Xeon E5506	I2: Sun Microsystems, Sun Fire X4450
J1: Supermicro A+ Server 2042G-6RF, AMD Opteron 6136	J2: Supermicro, Motherboard H8QI6-F, AMD Opteron 8435
K1: Huawei RH2285,Intel Xeon E5645	K2: Fujitsu CELSIUS W380, Intel Core i5-660
L1: Tyan YR190-B8228, AMD Opteron 4238	L2: Fujitsu CELSIUS W380, Intel Core i5-660
M1: Tyan YR190-B8228, AMD Opteron 4180	M2: Fujitsu Siemens Computers, CELSIUS M460, Intel Core 2 Quad Q9550
N1: Fujitsu, CELSIUS M470, Intel Xeon W3503	N2: Sun Microsystems, Sun Fire X4150

equipped with an Intel Xeon 2.66GHz processor for the above experiment.

C. Pairwise Comparison of Four Commodity Computers

Here, we applied our methods, t-test and HPT on pairwise comparison of four computers denoted as A, B, C and D which are specified in Table 4. For each computer, we run 1000 times for each benchmark in PARSEC [2] and SPLASH-2 [25] and then measure the execution time. All benchmarks are using their 8-thread version. In order to mimic the reality and have a full evaluation, we randomly select 5 out of 1000 execution times (without replacement) for each benchmark and computer. Then we applied HPT, t-test, and our methods (RT) on the selected sample which is a subset of the whole dataset. To avoid sampling bias, we repeat the experiment 100 times.

Table 5 shows the Monte Carlo results (i.e. the number of times the null hypothesis is rejected based on 100 random repetitions) on t-test, HPT and proposed randomization test on all six pairwise comparisons among four computers. Based on Table 5, we have the following observations:

1) In four pairwise comparisons (i.e. B vs. A, D vs. A, C vs. B and D vs. C), all methods have the same conclusions (i.e. reject the null hypothesis and conclude two computers have significantly different performance.)

2) For comparing computer A and C, we see that HPT rejects the null hypothesis only 5 out of 100 times while our methods rejects the null in all 100 trials at significance level 0.01. When we change the significance level to 0.05, the number of times the null hypothesis is rejected for HPT increases to 91. T-test performs similar to randomization test, except it fails to reject the null hypothesis 9 times at significance level 0.01.

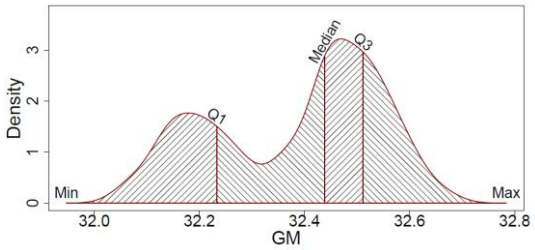

Figure 9. Illustration of five-number-summary on IBM BladeCenter HS22.

3) For comparing computer B and D, we see that HPT rejects the null hypothesis 90 out of 100 times while both randomization test and t-test reject the null in all 100 trials at significance level 0.01. When we change the significance level to 0.05, the number of times the null hypothesis is rejected for HPT increases to 99.

For this experiment, we conclude that when the performance difference between two computers is large[1], all three tests will have the same significant conclusion. However, when performance gap between two computers is small, then the randomization test has the highest chance to detect the difference.

Figure 8 shows the one hundred 0.95-Speedups from HPT test (red curves), the proposed 95% bootstrapping confidence intervals (blue curves on the boundaries with the grey region in the middle), and 95% t-confidence interval (gray lines). We see that the speed-up estimates from HPT approach are smaller than the bootstrapping estimates most of the time, which concurs with the Monte Carlo simulation results in Figure 7. This confirms that the speed-up estimates of HPT are relatively conservative than the bootstrapping estimates. Regarding the t-confidence interval, it is much wider than its bootstrapping counterpart, indicating that the bootstrapping method estimate is more precise than t-test. One interesting thing we found is

[1] In practice, we can use the critical value (e.g. for 95% confidence level the critical value is about 1.96 for a large sample size) multiplied by the standard error of the statistic (e.g. the geometric mean ratio) as the threshold. If the performance difference (e.g. the geometric mean ratio) is greater than the threshold, then it is considered "large".

Table 9. An illustration of choosing the sample size (*m*) based on the width of confidence interval.

m	3	5	7	10	13	15	16
Bootstrap CI	[1.203, 1.228]	[1.204, 1.223]	[1.207, 1.227]	[1.212, 1.228]	[1.216, 1.231]	[1.216 1.232]	[1.217, 1.232]
CI Width	0.0256	0.0198	0.0194	0.0166	0.0153	0.0155	0.0149

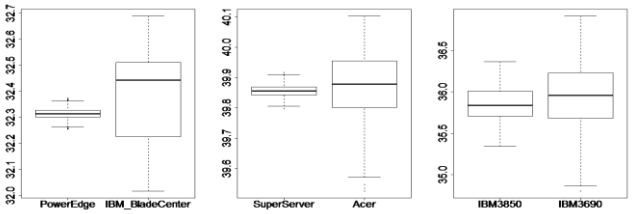

Figure 10. Graphic representation of five-number-summaries corresponding to the computers in Figure 5.

that the HPT 0.95 speedup is very close to the lower bound of the 95% t-confidence interval. This implies that the HPT speedup estimate is conservative and tends to underestimate the true speedup value.

D. SPEC CPU2006 Results

Now we carry out another experiment using the data collected from SPEC.org and have been used in Chen et al. [3]. Table 6 shows the comparative results of the 0.95-performance speedups obtained by HPT, 95% t-intervals, and the 95% bootstrapping confidence intervals of the ratio of geometric means performance speedups. The first row shows the ratio of geometric means performance speedups from the data. Interestingly, we see that the bootstrapping CI holds the ratio of geometric means performance speedups from the data. The 0.95-performance speedups obtained by HPT are all below the bootstrapping CIs. The 95% t-intervals are much wider than the ones from bootstrapping method, indicating its relatively low precision for estimation compared with bootstrapping method. In addition, the HPT 0.95 speedups are close to the lower limits of the t-intervals.

The above experiment shows that the HPT and our methods can identify the difference between each pair of computers, although the absolute Speedup numbers are different. Now we select another seven pairs of computers from SPEC.org [26] listed in Table 7 and perform the same experiment.

The results are listed in Table 8. We see that HPT shows low confidence and conservative estimate of Speedups in all cases while our proposed RT method demonstrates high confidence (>0.999). Similar as above results in Table 6, the 95% t-intervals are wider than the ones from bootstrapping method. Again, the GM Speedup is in the range of bootstrapping confidence intervals.

E. Five-number-summary Results

As we shown in Figure 5, the empirical distribution described above fully embraces the variability of computer systems which stems from non-deterministic hardware and software behaviors. However, sometimes it is desired to summarize the results through a few numbers instead of the empirical distribution, which usually contains hundreds of numbers. This can be achieved through the five-number-summary of the empirical distribution. Figure 9 illustrates the five-number-summary on the IBM BladeCenter HS22. We know that the

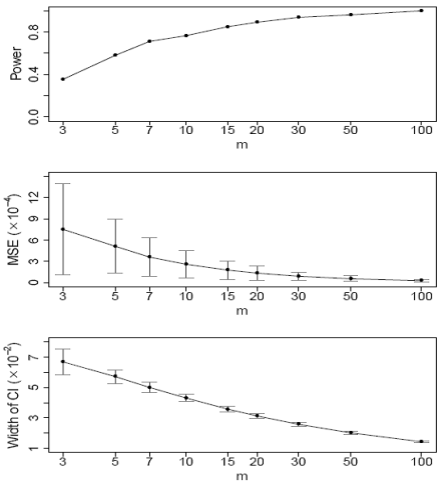

Figure 11. The sample size effect on the statistical power, MSE and the width of confidence interval under various sizes of m.

total area under the density curve is 100%. The first quartile (Q1), median, and the third quartile (Q3) cut the total area into four equal areas, which has 25% under curve area. Hence, five-number-summary is a compact way to summarize the distribution of a random variable and it shows the following characteristics of the distribution: 1) the range of data; 2) the range of the middle 50% of the data is Q3-Q1, which is called the Interquartile range (IQR) in the statistics community; 3) the center of the distribution. Both the range and IQR are often used as measuring the variation of a random variable. Figure 10 shows the boxplots, which are the graphic presentation of five-number-summary, of the computers listed in Table 3. Note that in boxplot, the bottom and the top of the boxplot are the minimum and maximum. The bottom and top of the box are the Q1 and Q3, respectively. The line inside the box is the median.

VII. THE SAMPLING SIZE

Due to the performance variability, we usually measure the performance score more than once for each benchmark. Hence, it remains a question that how many measurements (performance scores) for each benchmark, m, we should take. Generally, the size of m depends on two factors:

1) The size of the performance variability. If there is no performance variability, then measuring once, m=1, gives an accurate performance score. On the other hand, if the performance variability is large, then we need m be large to have a good estimation of performance.

2) The quality of the statistical inference. Hypothesis testing and estimation are the two major branches of statistical inference. A good test procedure should have a high probability to detect the deviation from the specified null hypothesis (i.e. high statistical power) when the null hypothesis is not true. On the other hand, the width of the confidence interval and the mean squared error (MSE) of an estimated parameter (e.g. speedup), gives us some idea about how uncertain we are about the unknown parameter. The smaller the width of a confidence

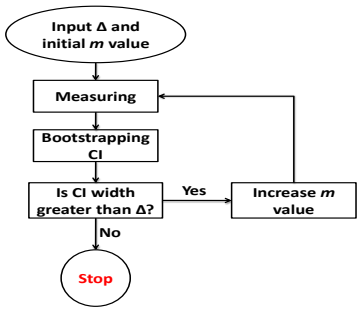

Figure 12. Flowchart of choosing the sample size based on the width of confidence interval.

interval (with fixed confidence level, e.g. 95%) and MSE, the more precise the estimate is. Hence, the statistical power, MSE and the width of confidence interval are widely used to examine the quality of statistical inference.

Here, we redo the Monte Carlo simulation study on power, described in Section VI.A, with L=1 on the commodity computer (AMD Opteron CPU 6172 @ 2.10GHz, 2 processors, each with 12 cores, with 12GB DDR3 RAM(1333 MHz)) using different sizes of m, m=3, 5, 7, 10, 15, 20, 30, 50, 100. The top panel of the proposed bootstrap estimate with different sizes of m. The vertical grey bar indicates the standard deviation of MSE. We see that the size of MSE (the smaller the MSE, the more accurate the estimate is) and its standard deviation decreases with the increase of m. Sometimes we may constrain the width of the confidence intervals. For example, we want to have a 95% confidence interval with width (i.e. upper limit – lower limit) no greater than 0.03. Notice that the smaller the width, the more consistency the estimate has. The bottom panel of Figure 11 shows the width of 95% confidence interval with different size of m. The vertical grey bar indicates the standard deviation of width. Similar to MSE, we see that the width of confidence interval decreases as m increases.

The above study shows the statistical properties of the proposed methods by increasing the size m. However, in practice we usually don't know the truth. Hence, the power of the test and MSE are unknown. A common way to determine the size of m is by setting the width of the confidence interval in advance. Figure 12 shows the flowchart of selecting the size of m in practice based on the predetermined width of confidence interval Δ. Basically, we need specify an initial value of m, usually a small value like 3, and a threshold for the width of confidence interval Δ. Then we sample m measurements for each benchmark and computer. We calculate a bootstrapping confidence interval based on the sample data. If the width of confidence interval is greater than the threshold Δ, then we increase the size of m and sample more measurements for each benchmark and computer. Then we recalculate the confidence interval. We stop sampling when the width of confidence interval is no greater than the predetermined threshold Δ.

For the example below, we use two computers: A and C described in section VI.C. We would like to find the size of m by restricting the width of the bootstrapping confidence interval of the ratio of geometric means performance speedups to be no greater than 0.015. Table 9 shows the bootstrapping confidence intervals and corresponding width with various sizes of m. We see that the sample size of m should be at least 16 under the restriction.

VIII. Applicability to Other Means

As a generic framework, our proposed methods can be directly applied to arithmetic and harmonic means while the HPT framework cannot apply since it uses rank instead of any performance metric. We applied the propose methods using these three means on an example in which we compare SPEC scores of two machines: IBM System x3500 M3 with Intel Xeon E5530, and CELSIUS R570 with Intel Xeon X5560, which are obtained from SPEC website [26]. Table 10 shows the confidences and confidence intervals using three metrics on the example. We see that both harmonic mean and geometric mean identify the difference between two computers while arithmetic mean cannot. This is because the arithmetic mean is subject to extreme values. For example, among 29 benchmarks, CELSIUS R570 has 25 benchmarks with a larger mean performance score than their counterparts for IBM System x3500 M3. However, IBM System x3500 M3 has much higher performance scores in the libquantum and bwaves benchmarks than their counterparts in CELSIUS R570. If the two benchmarks are eliminated from the data, then changes in the confidence and confidence interval using the arithmetic mean are much larger than the ones using the geometric and harmonic means.

IX. Related Work

Over decades, the debate over the method and metrics for computer performance evaluation has never ended [4][15][18]. Fleming and Wallace [10] argued that using geometric mean to summarize normalized benchmark measurements is a correct approach while arithmetic mean will lead to wrong conclusions in this situation. Smith [24], however, claimed that geometric mean cannot be used to describe computer performance as a rate (such as mflops) or a time by showing counter examples. Furthermore, John [16] advocated using weighted arithmetic mean or harmonic mean instead of geometric mean to summarize computer performance over a set of benchmarks. Hennessy and Patterson [13] described the pros and cons of geometrics mean, arithmetic mean, and harmonic mean. Eeckhout [7] summarized that arithmetic and harmonic means can clearly describe a set of benchmarks but cannot apply the performance number to a full workload space, while geometric mean might be extrapolated to a full benchmark space but the theoretic assumption cannot be proven.

Relying on only a single number is difficult to describe system variability stemming from complex hardware and software behaviors. Therefore, parametric statistic methods such as confidence interval and t-test have been introduced to evaluate performance [17][1]. Nevertheless, Chen et al. [3] demonstrated that these parametric methods in practice require a normal distribution of the measured population which is not the case for computer performance. In addition, the number of regular benchmark measurements from SPEC or PARSEC is usually not sufficient to maintain a normal distribution for the sample mean. Therefore, Chen et al. [3] proposed a non-parametric Statistic Hypothesis Tests to compare computer performance. As demonstrated in the paper, our proposed resampling methods can identify smaller differences between two computers even in a situation where a single test is not enough to reveal it.

Oliveira et al. [22] applied quantile regression to the non-normal data set and gained insights in computer performance evaluation that Analysis of variance (ANOVA) would have failed to provide. Curtsinger and Berger [5] proposed STABILIZER to control the layout effects by repeatedly randomizing the layouts of code, stack, and heap objects in the sample space of memory configurations at runtime. Our approach considers different variation sources (non-deterministic or deterministic behaviors) for the fixed computer configurations and handles the non-normality by using resampling technique such as boostrapping and permutation.

Patil and Lilja [23] demonstrated the usage of resampling and Jackknife in estimating the harmonic mean of an entire dataset. Unlike their approach, we applied resampling methods on a more complicated situation - comparing two computers on multiple benchmarks with multiple measurements. Hence, the bootstrapping method in our paper is different from the one in [23]. Namely, we bootstrap the samples within each benchmark instead of on the entire dataset.

X. CONCLUSION[i]

We propose a randomization test framework for achieving a both accurate and practical comparison of computer architectures performance. In the proposed test, we adopt within-benchmark-resampling which does not require a few distributional assumptions needed by parametric tests and HPT, such as the normality, independence and homogeneity assumptions between benchmarks.

We also propose a bootstrapping confidence interval estimation framework for estimating a confidence interval on a quantitative measurement of comparative performance between two computers. Like randomization test, the proposed bootstrapping method relaxes the distributional assumptions required by parametric tests and HPT through within-benchmark-resampling. We illustrate the proposed methods through both Monte Carlo simulations where the truth is known and real applications.

Interestingly, even though geometric mean as a single number cannot describe the performance variability, we find that the ratio of geometric means between two computers always falls into the range of Boosted Confidence Intervals in our experiments. This implies that geometric mean is still a good indicator to quantify the performance difference between two computers.

We also illustrate and compare the three metrics (geometric, arithmetic and harmonic means) on the motivating example and show that using arithmetic mean is sensitive to extreme values in the dataset.

In cases where two computers have very close performance metrics, we propose using empirical distribution to evaluate computer performance and using five-number-summary to summarize the computer performance.

Table 10. Summary of comparing geometric, harmonic and arithmetic means on confidence and confidence interval (CI).

	G-Mean	H-Mean	A-Mean
Confidence	>0.99	>0.99	0.492
CI	[0.913, 0.920]	[0.887, 0.892]	[1.019, 1.031]
Confidence*	>0.99	>0.99	>0.99
CI*	[0.882, 0.889]	[0.881, 0.886]	[0.880, 0.889]

* Confidence and confidence interval after eliminating the *libquantum* and *bewaves* benchmarks.

REFERENCES

[1] Alaa R. Alameldeen and David A. Wood, "Variability in Architectural Simulations of Multi-threaded Workloads," in HPCA-9, Feb. 2003.

[2] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," In Proceedings of the 17th PACT, October 2008.

[3] T. Chen, Y. Chen, Q. Guo, O. Temam, Y. Wu, and W. Hu, "Statistical performance comparisons of computers," in HPCA-18, 2012.

[4] Daniel Citron, Adham Hurani, and Alaa Gnadrey, "The harmonic or geometric mean: does it really matter?," ACM SIGARCH Computer Architecture, vol. 34, no. 4, pp. 18 - 25, September 2006.

[5] Charlie Curtsinger and Emery D. Berger, "STABILIZER: statistically sound performance evaluation," in ASPLOS-18, 2013

[6] A. C. Davison and D. V. Hinkley, Bootstrap Methods and their Application, Cambridge University Press, 1997.

[7] Lieven Eeckhout, "Computer Architecture Performance Evaluation Methods," Morgan & Claypool Press, 2010.

[8] E. S. Edgington, Randomization tests, 3rd ed. New York: Marcel-Dekker, 1995.

[9] Bradley Efron and Robert J. Tibshirani, "An Introduction to the Bootstrap," Chapman and Hall/CRC, 1994.

[10] Philip J. Fleming and John J. Wallace, "How not to lie with statistics: The correct way to summarize benchmark results," Communications of the ACM, 29(3):218–221, March 1986.

[11] Rudolf J. Freund, Donna Mohr, and William J. Wilson, "Statistical Methods," Academic Press, 3rd edition, 2010.

[12] Andy George, Dries Buytaer, and Lieven Eeckhout, "Statistically rigorous java performance evaluation", in OOPSLA'07, 2007.

[13] John L. Hennessy and David A. Patterson, "Computer Architecture: A Quantitative Approach," 4th ed., Morgan Kaufmann, 2007.

[14] Myles Hollander and Douglas A. Wolfe, "Nonparametric Statistical Methods," Wiley-Interscience, 2nd ed. 1999.

[15] Muhammad Faisal Iqbal and Lizy Kurian John, "Confusion by All Means," in Proceedings of the 6th International Workshop on Unique chips and Systems(UCAS-6), 2010.

[16] Lizy Kurian John, "More on finding a single number to indicate overall performance of a benchmark suite," ACM SIGARCH Computer Architecture, vol. 32, no. 1, pp. 3 - 8, March 2004.

[17] David J. Lilja, "Measuring Computer Performance: A Practitioner's Guide," Cambridge University Press, 2000.

[18] John R. Mashey, "War of the benchmark means: time for a truce," ACM SIGARCH Computer Architecture, vol. 32, no. 4, pp. 1 - 14, September 2004.

[19] David Moore, George P. McCabe, and Bruce Craig, "Introduction to the Practice of Statistics," W. H. Freeman Press; 7th Ed. 2010.

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

[21] Richard A. Johnson, Statistics: Principles and Methods, Wiley; 6th edition, 2009.

[22] Augusto Oliveira, Sebastian Fischmeister, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney, "Why you should care about quantile regression," in ASPLOS-18, 2013.

[23] Shruti Patil and David J. Lilja, "Using Resampling Techniques to Compute Confidence Intervals for the Harmonic Mean of Rate-Based Performance Metrics," IEEE Computer Architecture Letters, Jan.-June, 2010, pp. 1-4.

[24] James E. Smith, "Characterizing computer performance with a single number," Communications of the ACM, vol. 31, no. 10, pp. 1202 - 1206, Oct. 1988.

[25] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in ISCA-22, pages 24-36, Jun. 1995.

[26] http://www.spec.org/cpu2006/results/.

[i] This work is supported in part by an NSF Grant CCF-1017961. We appreciate the invaluable comments from anonymous reviewers and the shepherd professor Michael Huang which help us finalize the paper.