Confusion by All Means

Muhammad Faisal Iqbal and Lizy K. John
University of Texas at Austin
{iqbal,ljohn}@ece.utexas.edu

***Abstract*—Performance of computers is usually measured by using benchmark suites. There has been a long debate among computer architects on how to aggregate the individual program results to present a summary of performance over the entire suite. Many researchers have criticised the use of Geometric Mean (GM) but SPEC continues to use it to report performance. Mashey [7] has strongly supported the use of GM. According to Mashey, the programs in a benchmark suite like SPEC are samples of some population of programs. It is important that we model the distribution of population correctly before calculating any statisitics and making conclusions based on those statisitcs. Mashey also conjectures that Lognormal distribution is a better model than the Normal ditribution for such benchmark suites. Since GM is the back-transformed average of a Lognormal distribution, its use as a measure of central tendency is statistically correct. In this study, we evaluate the correctness of this Lognormal assumption using the large repository of performance results for SPEC CPU2006 published on SPEC's website. Utilizing different tests for normality, we find out that although Lognormal distribution models the performance results better than the Normal distribution, there is a very large percentage of machines which are neither Normal nor Lognormal. Our study indicates that most of the non-normality is caused by small number of outliers. We study the causes of these outliers and evaluate the use of *Coefficeint-of-Variance* to identify outliers. We also present some suggestions on how to deal with these outliers.**

***Index Terms*—Benchmark Means, Geometric Mean, Normality Test, Lognormal Distribution**

I. Introduction

There is a long history of debate on how to summarize the performance of a benchmark suite and which mean is an appropriate measure of the central tendency [6], [5], [8], [10], [3], [7]. There have been strong arguments both in favour of and against the use of each type of mean. Citron et al. [1] present a detailed history of this discussion. In this section, we discus some of the arguments made in this regard.

According to Lilja [6] arithmetic mean is proportional to execution time and hence is the right measure for time based metrics. He also argues that harmonic mean should be used for rate based metrics and weighted arithmetic or harmonic mean should be used for time and rate based metrics respectively if weights of individual programs are different. He strongly opposes the use of geometric mean as a measure of central tendency. He shows that although geometric mean produces consistent ordering of machines when Normalized times are compared, this ordering is consistently wrong with reference to the total execution times of the benchmarks. Hence he concludes that 'geometric mean is not the appropriate mean for summarizing times or rates, irrespective of whether they are Normalized'.

Similarly John [5] argues that weighted arithmetic or harmonic mean can be used to correctly represent performance. She shows with numerical examples that both arithmetic and harmonic means yield correct orderings with respect to execution times if these means are appropriately weighted. She also maintains that geometric mean is not an appropriate measure since it is not proportional to the execution times of the benchmarks.

On the other hand Fleming et al. [3] studies all three types of mean and vote in favour of geometric mean since it always produces consistent rank order among the machines. The most convincing arguments in favour of geometric mean have come from Mashey [7]. He has performed detailed characterization of workload analysis and argues that benchmarks like SPEC's CPU benchmarks are examples of *Sample Estimation of Relative Estimation of Programs (SERPOP)* i.e these benchmark are a sample which is used to represent a population of programs which might run on a particular machine. He argues that performance of machines on benchmarks like SPEC can be better modelled using Lognormal distribution than the Normal distribution. Geometric Mean which is the *back transformed average of Lognormal distribution* is the statistically appropriate measure to be used. Also, the analysis done by Lilja and John [6], [5] can be considered as *Workload Analysis With Weights (WAW)* where the user knows exactly which programs will run on the machine and the relative frequency/importance of the programs. In case of WAW analysis, weighted AM or HM are indeed the correct measures for algebraic calculations as pointed out by these researchers. Most of the benchmarking efforts, however, try to do the SERPOP analysis and hence we'll deal with this kind of anlaysis in the remainder of the paper. We evaluate the correctness of Lognormal assumption using SPEC CPU2006 data with three different tests for normality: Lillie Test, Shapiro-Wilks Test and D'Agostino-Pearson Test. We also evaluate the effectiveness of COV in identifying the outliers and present some suggestions on how to deal with these outliers.

II. Is Geometric Mean an Appropriate Metric?

When comparing performance of machines based on the benchmark results, we are actually comparing distribution of performances. It is important that we understand the nature of these distrubtions before calculating any statisitics and making conclusions based on those statistics. In case of Normal distributions, "mean" can be a good measure of central tendency. But if the distribution is not Normal then mean does not give us any useful information about the central tendency and we should be careful while interpreting the mean. Sometimes a

transformation of data can yield a Normal distribution and calculation of statistics in the transformed domain can be very useful. In the case of a benchmark suite like SPEC, Lognormal distribution is of particular interest. *Lognormal distribution* is the ditribution of samples whose Logarithm is normally distributed. As Mashey [7] has pointed out, GM can be thought of as the the back-transformed mean of a Lognormal distribution

$$GM = \overline{x_g} = (\prod_{i=1}^{n} x_i)^{\frac{1}{n}} = exp(\frac{1}{n}\sum_{i=1}^{n} ln(x_i) \quad (1)$$

i.e if we take the mean of logarithm of all samples and the back trasform from logarithm, we get the geometric mean. In other words if

$$\overline{x_g} = \frac{1}{n}\sum_{i=1}^{n} log_{10} x_i \quad (2)$$

then

$$Mean = exp(x_g) = GM \quad (3)$$

Thus it is statistically correct to use GM if the data can be modeled using the Lognormal distribution. Furthermore, the speedup numbers are calculated as the ratio of execution time of a program on a given machine to execution time on the base machine. There is nothing in the real world that distinguishes base machine A from other machines B. Ratios of A/B are just as valid as B/A. This is the real fundamental reason why one has to use some metric that works that way, so that if A is 2X faster than B, B better be .5X as fast as A, which only works if we take the logarithms. Arithmetic means of ratios dont have that property, although with small dispersions, normal may be a good quick approximation, and the AM and GM are close anyway. In the case of becnhmark suites like SPEC 2006, Lognormal distribution can cater for small outliers better than the Normal distribution and thus should be a better model for the results. Mashey has shown with one example from SPEC CPU2000 results that Lognormal distribution can better model the data. In this paper, we utilize the base results available for SPEC CPU2006 from SPEC's website for about 2000 machines and test how well the Normal or Lognormal distribution models the data.

A. Tests for Normality

The easiest and most obvious way of testing for normality is to draw the histogram and visually see how well this histogram resembles the bell-shaped curve. But this is not the most accurate way of testing for normality, especially when the sample sizes are very small as in our case (12 data samples for SPECint and 17 for SPECfp). With small sample sizes, discerning the shape of the histogram is a difficult task and the histogram shape can change significantly just by changing the interval width of the histogram. A better way of testing for normality is to use the normality tests. We perform three different normality tests to verify the assumption of normality for SPEC CPU2006 results. All three tests are *frequentist tests*. Frequentist tests use hypothesis testing and the decision is made using a *null hypothesis*. Null hypothesis is the basic assumption that is put forward when making a statistical inquiry and is usually denoted by H_0. The validity of the null hypothesis is tested using the statistical test which calculates a *test-statistic*. In hypothesis testing, the *significance level* (α) is the criterion used for rejecting the null hypothesis. First, the difference between the results of the experiment and the null hypothesis is determined. Then, assuming the null hypothesis is true, the probability $(p\text{-}value)$ is computed that the difference can be at least as large as observed. If the *p-value* is less than or equal to the significance level(α), then the null hypothesis is rejected. If the test shows that we should reject the null hypothesis, it is done in favour of an *alternative hypothesis*, represented as H_1. In our study the hypthesis testing can be formalized as:

```
H0: Samples are from a Normal Distribution
H1: Samples are not from a Normal Distribution
```

The three tests that we use have different ways of calculating the test statistic and differ in how they quantify the deviation of the actual distribution from a Gaussian distribution. A good discussion on normality tests can be found in [4]. We present a summary of the three tests that we are using:

1) Lillie Test: This test is an adaptation of Kolmogrov-Smirnov test with mean and variance of the Normal distribtion not specified in the null hypothesis. This test first estimates the population mean and variance from the sample data. It then compares the cumulative distribution of samples with the expected cumulative Normal distribution. The test statistics is based on the largest discrepancy similar to KS-test i.e. for a vector x of samples the test statistic is given as

$$KS = max|SCDF(x) - CDF(x)| \quad (4)$$

where SCDF is the empirical cdf estimated from the sample and CDF is the Normal cdf with mean and standard deviation equal to sample mean and standard deviation. We performed this test using the `lillietest()` function available in Matlab. We performed a two sided `lillietest()` with an α of 0.05. The result `h` returned by this test is 1 when we can safely reject the null hypothesis i.e, When the `p-value` calculated by the test is smaller than the signigicance level α.

2) Shapiro-Wilk Test: This test is (semi/non) parametric analysis of variance and is useful in detecting broad range of departures from the normality of sampled data. This test is considered to be more powerful in detecting the non-normality than the "distance" tests like the Lillie Test. This test is shown to work for number of samples between 3 and 5000. Most authors agree that this is the most reliable test for normality for small to medium size samples. The test statisitic for this test is given as

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \quad (5)$$

where x_i are the ordered sample values(x_1 being the smallest) and a_i are the constants generated from the means, variances and covariances of the ordered statistics of a sample of size n from a Normal distribution. The small values of W are

an evidence of departure from normality. This test was also performed in matlab with α of 0.05.

3) D'Agostino-Pearson test: This test asseses the normality using *skewness* (to quantify the asymmetry of the distribution) and *kurtosis* (to quantify the shape i.e, peakedness of the distribution). A Normal distribution is assumed to have a kurtosis value equal to 0. A higher kurtosis means that the distribtion is peakier and a negative kurtosis means that the distribtion is flatter than the Normal distribution. Also, a Normal distribution has a skew of zero. A positive skew means that there is a long tail to the right of mean and a negative skew means a tail to the left. D'Agostino-Pearson test first calculates skew and kurtosis of the sample data and then calculates how far each of these values differs from the value expected with a Normal distribution. Finally it calculates a single p-value based on these discrepencies. A smaller p-value means departure from the normality. Again, we performed this test using an α of 0.05.

B. Do SPEC CPU2006 results follow a Lognormal Distribution?

We performed all three normality tests for SPEC CPU2006 (both SPECint and SPECfp) results obtained from SPEC's website. The data used in this paper includes all the results which were published on or before September 9, 2010. For normality-testing, we apply the tests on speedup data i.e. runtime on machine under test/run time on the base machine and for Lognormality testing, we use logarithm(speedup) data.

Table I lists the results of normality tests. Columns labeled 'Normal' and 'Lognormal' represent the number of machines which passed the normality test for speedup and logarithm(speedup) numbers respectively. The numbers given in the two columns are not exclusive i.e. a machine can be considered both Normal and Lognormal. The columns labeled "None" show number of machines which were identified as neither Normal nor Lognormal. We can see that, although Lognormal models data better than the Normal distribution, the proportion of machines showing Lognormal (or Normal) behaviour is very small. If the sample values are close to each other, both Normal and Lognormal assumptions are equally correct to model the data. When the standard deviation increases i.e. the distrubutions start having a long tail or a skew, the fit for Normal distribution worsens but the Lognormal distribution still fits in case of small outliers. Figure 1(a) shows a typical example of a machine whose results (SPECInt in this particular example) show a non-Normal behaviour. But the logarithm of speedup numbers can be considered Normal as identified by shapiro-wilk test. Taking logarithm of speedup numbers decreases both skew and kurtosis and brings the distribution closer to an ideal Normal distribution. This is typically the case with this category of machines where skew is caused by presence of a small-medium outlier. Figure 1(b) shows an example of the second category of machines. Here the outlier is far away from other programs and even taking logarithm cannot reduce the skew to the desired values.

We also found very small fraction of machines (12/2125) which could be modeled by Normal distribution but not by Lognormal. In all these cases, taking logarithm made kurtosis negative, resulting in a a distribution which is flatter than a Normal distribution. Figure 1(c) shows example of such a machine. Results from D'agistino-pearson Test in Table I for SPECint show that percentage of machines exhibiting Normal or Lognormal behaviour is only 13% and 19% respectively. Even with Lillie Test, which is considered the weakest, percentage of Normal and Lognormal machines is only 16% and 34% respectively. The percentage of Lognormal machines is a little higher in case of SPECfp i.e, 50%, 30% and 25% using Lillie, Shapiro-Wilk and D'Agostino-Pearson Tests repectively. From these results Lognormal seems to be a better representation of distribution than Normal. In these situations GM is statistically the correct measure of cetral tendency. But, Lognormality cannot be assumed in general as suggested by high percentage of non-Lognormal machines in the results. In such situations, the results and statistics should be interpreted very carefully.

C. What are the causes of Non-normality?

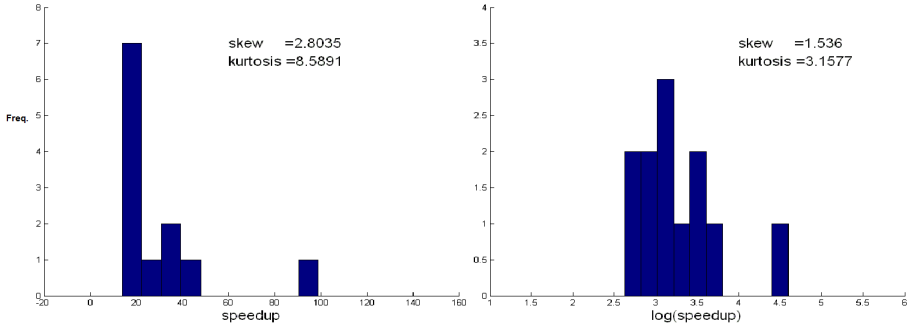
In order to analyze our data set, we calculated the first four moments; Arithmetic Mean, Standard Deviation, Skew and Kurtosis. Then we performed exploratory data analysis to hunt for the odd cases.

1) SPECint: Almost all of the machines which show the non-Normal behavior have high standard deviation. This high standard deviation is usually caused by presence of an outlier. On a detailed inspection we found that this non-normality is caused by a single outlier i.e. `462.libquantum`. These machines compile `462.libquantum` with `-parallel` flag enabled. These machines are multi-core machines and support multiple threads, so the performance of libquantum on these machines shoots up. `462.libquantum` is a C library for simulation of quantum mechanics and is easy to parallelize. Part of the speedup also comes from the availability of 64 bit hardware since the benchmark uses 64-bit arithmetic very extensively in its algorithm [2]. Infact, all of the top 10 machines for SPECint have the speedup number for `462.libquantum` greater than 600. Compiler teams of most of the vendors seem to have cracked this benchmark with compiler flags and cache management instructions. They can focus on just this particular program and get very high values of GM. Thus optimizing for `462.libquantum` is just blowing the numbers away. Such high numbers for one or two outliers badly wreck any statistics approach. Similar things have happened in the past. For example, in the original SPEC89 benchmarks, cache-blocking compilers achieved similar performance gains for `matrix300`. It is important that we identify and isolate the outliers otherwise any statistics calculated with such a data set will not be reliable.

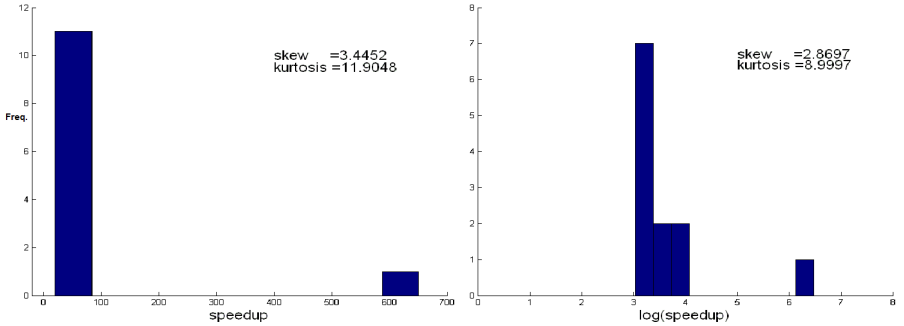
2) SPECfp: The situation in SPECfp is not very different. There is a high percentage of machines which show the non-(log)-Normal behaviour. If we sort the machines with respect to standard deviation, we can easily find out the two outliers in non-Normal machines namely `410.bwaves` and `436.CactusADM`. Both these programs are compiled using autoparallelization. The vendors are able to get very high

		Lillie Test			Shapiro-Wilk Test			Dagos-Pearson Test		
benchmark	Total Machines	Normal	Lognormal	None	Normal	Lognormal	None	Normal	Lognormal	None
SPECint	2125	341	709	1415	362	526	1587	266	398	1723
SPECfp	2066	690	1469	597	696	1169	882	565	987	1057

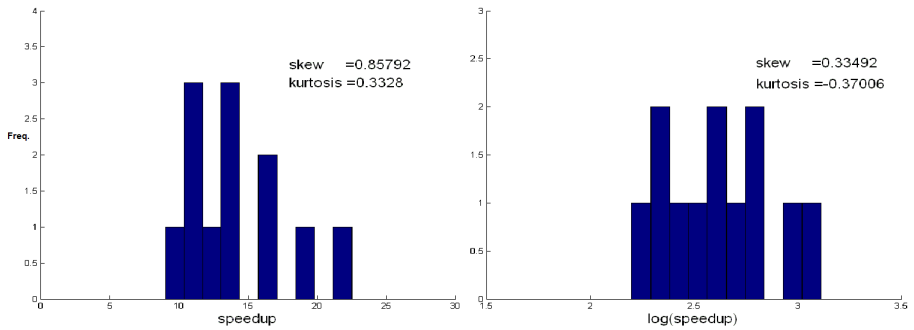
TABLE I
RESULTS OF THE NORMALITY TEST FOR SPEC CPU2006



(a) Typical Machine which is non-Normal but Lognormal (NovaScaleR410 F2 Intel Core i3-540, 3.06 GHz)



(b) Typical Machine which is neither Normal nor Lognormal (ASUS RS300-E6 (P7F-E) Intel Xeon X3470)



(c) Typical machine which is Normal but not Lognormal (IBM System X 3250 Intel Xeon X3220)

Fig. 1. Histograms of Typical Cases from the SPEC CPU2006 results

performance numbers for these programs as compared to the other programs. In contrast to SPECint programs which are relatively easy to group, there is a possibility that SPECfp programs need to be categorized into scalar, vectorizable, and parallelizable etc programs. Indeed, programs like `410.bwaves` and `436.cactusADM` do begin to form a second distribution and should be treated separately from other programs. SPEC has encountered similar situations in the past. Initially they begun with one single benchmark suite containing both intger and floating point benchmarks. But as soon as they realized the existence of bi-modal distribution in case of integer and floating point programs, they separated the benchmark suite into separate integer (SPECint) and floating point (SPECfp) benchmarks. Similarly SPECfp programs may need to further get split into scalar, vectorizable, parallelizable, and not mixed together. All it takes is one like `410.bwaves` to skew results and badly damage the predictability.

III. How to deal with Non-normality?

Although Lognormal distribution is able to model SPEC2006 data better than the Normal distribution, it can do so only in case of small outliers. If the outliers are very far away or there are multi-modal distributions, data cannot be modeled correctly even by Lognormal distribution. It is important that we identify these outliers and deal with them accordingly. In this section we present our recommendations to deal with such situations.

A. Report a Measure of Dispersion

A measure of dispersion should be very useful in identifying the weird cases. It helps in quantifying the ranges and confidence within which to expect most of the benchmarks. Digital Review magazine in 1980s used to report confidence interval, standard deviation and variances for this purpose. In our opinion, Coefficient of Variation (COV) should even be a better measure than standard-deviation and variances. COV is defined as

$$COV = standard\_deviation/mean \tag{6}$$

COV is a better measure because standard deviation must be understood relative to the mean and if one is interested in comparing distributions with different means, co-efficient of variation should be used.
At the moment SPEC gives just one number (GM) and it does not provide any measure of dispersion. Although measure of dispersion can be calculated directly from SPEC's data, a single number like COV can really alert the user about weirdness of results i.e. outliers or multi-modal distributions. In fact in our results, all the machines which have co-efficient of variance greater than 1 are identified as non-Lognormal by all three normality tests. Table II shows the COV of both Lognormal and non-Lognormal machines in detail. We have used the results of Shapiro-Wilk test for table II.

	SPECint		SPECfp	
	Lognormal	non-Lognormal	Lognormal	non-Lognormal
COV(Avg.)	0.41	1.57	0.42	0.95

TABLE II
AVERAGE VALUE OF COV FOR NORMAL AND NON-NORMAL MACHINES

Fig. 2 plots the COV for SPECint for both Lognormal and non-LogNormal machines. We can see that COV of all the Lognormal machines is less than 1. We found some cases where non-Lognormal machines showed small COV. The non-Lognormality in these cases is due to high kurtosis (more peakier of distribution) value. This means that more benchmarks are closer to each other. Obviously, if more benchmarks are close to each other, then GM (or any other mean) is a correct measure of central tendency and can be used safely. High COV value always correctly identifies the weird cases of outliers.

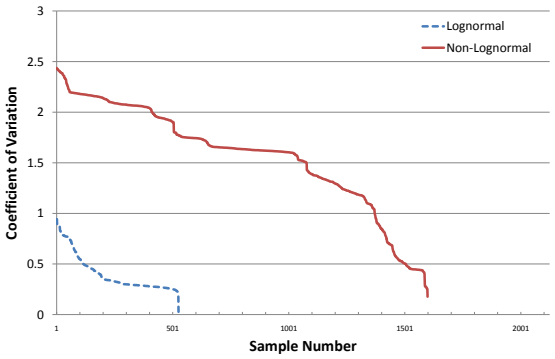


Fig. 2. COV values for Lognormal and non-Lognormal Machines (samples are in decreasing order of COV)

B. Isolate and treat the outliers separately

Once we have identified the outliers, we need to treat them separately from other programs. In case of SPECint, since we find only one outlier, it is easy to just reomve it from the stats and use the mean of remaining benchmarks. We removed the outliers, `462.libquantum` from SPECint, `410.bwaves` and `436.cactusADM` from SPECfp and ran the normality tests again. Results are listed in Table III.

From the table we can see that more than 97%, 90% of the machines in both shapiro and Dagos test are Lognormal for integer and floating point benchmarks respectively. Thus we can see that after removing the outliers, the distribution can be considered as Lognormal and GM can give a good measure of central tendency. The non-normality in the remaining cases is generally due to high kurtosis value. This means that distributions are peakier than the Normal distribution and now GM (or any mean) is a good measure of central tendency since we don't have any outliers.

C. Use a Ranking System not just the Mean

A ranking system can be very useful when a user is comparing multiple alternative machines. Instead of just using mean to compare the performance of machines, one can use a ranking system like "Borda Counts"[9]. This is a single winner election method and has roots in French Revolution. In this method the voters rank candidates in order of preference. The Borda count selects the winner by giving each candidate a certain number of points corresponding to the position in which he or she is ranked by each voter. The person with most points is declared the winner. In our context, if we are trying to compare `n` alternative machines, we can rank order the machines for each program based on the performance. Then based on these ranks, each machine will get points for each program. The sum of all the points will decide the final rank for that machine. This type of ranking system is good, since one outlier does not blow away all the statistics. A machine has to perform consistently well in order to be declared as

		Lillie Test		Shapiro-Wilk Test		Dagos-Pearson Test	
benchmark	Total Machines	Normal	Lognormal	Normal	Lognormal	Normal	Lognormal
SPECint	2125	1137	1684	2050	2112	1826	2079
SPECfp	2066	1886	2017	1782	1852	1683	1899

TABLE III
RESULTS OF THE NORMALITY TEST AFTER REMOVING THE OUTLIERS

winner. Thus a proper ranking system should be used when we are rank ordering the machines. Use of a single mean, as done by Citron et al in their study [1] to rank the machines, is not the correct approach.

D. Generating a Single Number

Computer architects agree that one number like GM or HM can not tell the whole story. But it is sometimes important to get only one number for the purpose of comparisons. We believe that, in these situations, the dispersion of data should be incorporated into this number. One way to do it is to make the final benchmark score inversely proportional to COV. Something like

$$BenchmarkScore \propto (\frac{1}{1+COV})(GM) \tag{7}$$

or

$$BenchmarkScore = (\frac{k}{1+COV})(GM) \tag{8}$$

In this way the machines with high Co-efficient of variance will be penalized more. And the machines in which all the programs perform almost equally will not be penalized. This score number will ensure that nobody will be able to rank better, just by doing program specific optimization on one or two programs of the benchmark suite. More research needs to be done in order to find appropriate values of `k` and to idetify more variables that can be incorporated in equation 8.

IV. CONCLUSIONS

Evaluating multiple machines based on performance on a becnhmark suite is generally a SERPOP analysis. In case of small outliers, Lognormal is a better model for distribution of performance than Normal distribution. The results of normality tests show that Lognormal distribution can not be assumed in general. The existence of outliers and multimodal distributions can badly wreck any statistics approach. With relatively small numbers of benchmarks, it is almost inevitable that there be outliers, and one of the questions raised for future research is: how many benchmarks do you need to improve confidence? A measure of dispersion such as COV can be very useful in identifying such situations. Once an outlier or a multi-modal distribution is identified, one should treat the weird cases very carefully. We also advocate the use of a proper ranking system instead of just the GM in order to rank order the machines. Also, even if a single number is ultra important, the score should take the measure of dispersion into account in addition to the mean as shown in equation 8. A lot of research needs to be done in order to find a proper ranking system and fine tuning the performance score numbers like the one in equation 8.

ACKNOWLEDGEMENTS

We would like to than John R. Mashey for his valuable feedback. We would also like to thank Vincent Davis and Youngtaek Kim for their help in improving this manuscript.

REFERENCES

[1] Daniel Citron, Adham Hurani, and Alaa Gnadrey. The harmonic or geometric mean: does it really matter? *SIGARCH Comput. Archit. News*, 34(4):18–25, 2006.

[2] Dong Ye et al. Performance characterization of spec cpu2006 integer benchmarks on x86-64 architecture. *IISWC*, 2006.

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

[4] Zar J. H. *Biostatistical Analysis (2nd edition)*. Prentice-Hall, 1999.

[5] Lizy Kurian John. More on finding a single number to indicate overall performance of a benchmark suite. *ACM Computer Architecture News*, 2004.

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

[7] John R. Mashey. War of the benchmark means: time for a truce. *SIGARCH Comput. Archit. News*, 32(4):1–14, 2004.

[8] Patterson and Hennessy. *Computer Architecture: The Hardware/Software approach*. Morgan Kaufman Publishers.

[9] Donald G. Sari. Mathematical structure of voting paradoxes, positional voting. *Economic Theory*, 15, 2000.

[10] J. E. Smith. Characterizing computer performance by a single number. *Communications of ACM*, october 1998.