Assessing Computer Performance with SToCS

[Work in Progress Paper]

Leonardo Piga, Gabriel F. T. Gomes, Rafael Auler, Bruno Rosa, Sandro Rigo, Edson Borin
Institute of Computing, University of Campinas, Brazil
{leonardo.piga,gabriel.ferreira,rafael.auler,bruno.rosa}@lsc.ic.unicamp.br, {sandro,edson}@ic.unicamp.br

ABSTRACT

Several aspects of a computer system cause performance measurements to include random errors. Moreover, these systems are typically composed of a non-trivial combination of individual components that may cause one system to perform better or worse than another depending on the workload. Hence, properly measuring and comparing computer systems performance are non-trivial tasks.

The majority of work published on recent major computer architecture conferences do not report the random errors measured on their experiments. The few remaining authors have been using only confidence intervals or standard deviations to quantify and factor out random errors. Recent publications claim that this approach could still lead to misleading conclusions.

In this work, we reproduce and discuss the results obtained in a previous study. Finally, we propose SToCS, a tool that integrates several statistical frameworks and facilitates the analysis of computer science experiments.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques; G.3 [**Probability and Statistics**]: Experimental design, Nonparametric Statistics

Keywords

Performance Analysis, Statistics, Hypothesis Tests

1. INTRODUCTION

Computer performance measurement is subject to random and systematic errors. Mytkowicz *et al.* [15] show that the code layout produced by assemblers and linkers may impact hardware mechanisms, such as cache memory and branch predictors, in non-deterministic ways. Lilja [11] states that several other factors influence the measurement of computer performance, for example the precision of the measuring tools. In order to assist experimenters in the analysis of random errors in the measured data, we propose SToCS (Statistical Tool for Computer Science) [13], a tool that integrates several statistical frameworks and facilitates the analysis of computer science experiments.

Another problem faced when assessing computer performance is the summarization of performance metrics collected for individual benchmarks in a suite (e.g. SPEC CPU 2006). Several metrics have been proposed, leading to decades of debates. This became known as the “War of the benchmark means” [14], where several authors discussed about which summarization technique better describes the overall performance of a computer across benchmarks [8, 9]. We evaluate these metrics with data collected from the execution of benchmarks on real machines and with synthetic data and observe that these metrics provide similar results in our experiments.

More recently, Chen *et al.* [7] argued that parametric statistical techniques should not be used when assessing computer systems performance data and introduced a hierarchical performance test framework (HPT) that applies consecutive statistical hypothesis tests on the performance scores of two computing systems. The framework reports how much a system is faster than the other and with which confidence level. Our results indicate that HPT is conservative when compared with other summarization techniques.

Last, Lilja states that the analysis of computer systems performance should be thought of as a combination of measurement, interpretation, and communication [11]. According to the American Physics Society, the success and credibility of science rely on the exposition of ideas and results to independent testing [1]. We suggest that providing the maximum amount of information in computer science research contributes to the advance of the area.

The main motivation for the development of SToCS are the intriguing questions raised by the discrepancies between HPT and other, more conventional, statistical frameworks, such as the geometric mean of ratios used by SPEC.

The contributions of this work can be summarized as follows: 1) We conduct experiments similar to those performed by the authors of the HPT test and verify that the distribution of the samples is indeed not normal. 2) We also experiment with the test proposed by the authors and notice that the conclusions that can be inferred from our experimental results are conservative, when compared with the traditional means. 3) We propose a tool that implements all these data analysis and helps experimenters to report their results.

The remaining of the text is organized as follows: Sec-

.

tion 2 motivates the creation of SToCS by analyzing the overhead of a binary translator. Section 3 presents issues that are raised when summarizing performance scores. Section 4 evaluates the HPT framework on a synthetic experiment where the true means are known. Finally Section 5 presents our conclusions.

2. MOTIVATION

To motivate the discussion, we present an experiment where we analyze the overhead caused by emulation in a dynamic binary translator (DynamoRIO [4]) when running SPECint 2006 using the Hierarchical Performance Test framework (HPT) [7].

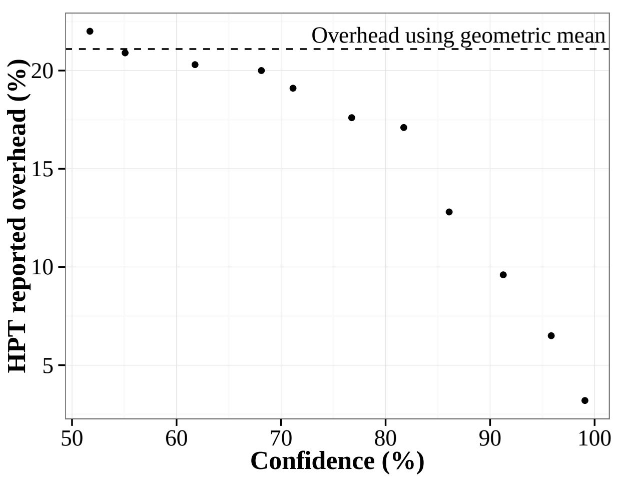

Figure 1: HPT reported overhead for a varying confidence interval caused by the DynamoRIO binary translator. Note that HPT reports a lower bound for the overhead.

The quality of a DBT (Dynamic Bynary Translator) is measured by its fitness to translate computer programs efficiently and transparently [3, 10]. Figure 1 shows the emulation overhead (slowdown) reported by HPT given different confidence levels. The graph shows that the higher the confidence level, the lower is what can be asserted about the overhead. HPT uses Wilcoxon tests to determine lower bounds on the differences between two systems. For example, with a confidence level of 61%, the overhead is *no less* than 20.3%. Moreover, with a confidence level of 99%, we can only assert that the delay added by emulation is *no less* than 3.2%.

The outputs of the HPT framework are always accompanied by confidence levels, on the other hand they report more modest speedups in comparison with the geometric mean of ratios (dashed line in Figure 1). We refer to this behavior as "conservative". The difference between the methods raised the question about which of them we should use and motivated the creation of a tool that aggregates several statistical frameworks.

To help experimenters to report their performance number we developed SToCS, a command line tool implemented in Python that calculates several statistics of an experiment and compares measurements between two computers using different statistical frameworks. It processes CSV files to calculate statistics such as arithmetic, geometric, harmonic means, standard deviation, minimum and maximum values of a given experiment. Furthermore, when two data files are provided, the tool is able to compare the measurements of the given files using different metrics such as the HPT-speedup and the geometric mean of ratios (e.g. SPECrate) [7, 11]. The tool implements the HPT framework, which is not available in conventional tools such as R [16]. R does have the Wilcoxon tests that are used by HPT, but it does not include the HPT framework itself.

3. SUMMARIZING COMPUTER PERFORMANCE SCORES

This section describes issues that are raised when summarizing performance scores, the consequences of making wrong assumptions about data distribution, and the applicability of the Central Limit Theorem (CLT) in computer science experiments.

Confidence intervals can be used to represent the index of dispersion of individual benchmarks provided that the experimenter guarantees that the actual data distribution is normal and techniques such as resampling are not used. But even when this is the case, summarizing the overall performance score of a computer across several benchmark applications with a single number can be misleading.

Figure 2 shows the individual execution time ratios in Actina Solar 220 X3 system compared with the base system defined by SPEC – data collected from SPEC.org. The discrepancy observed in *libquantum* influences indexes of central tendency in different ways. For example, in the ratio of the arithmetic means, a single value has more influence than in the geometric mean of ratios, which amortizes the discrepancies in the summarized data.

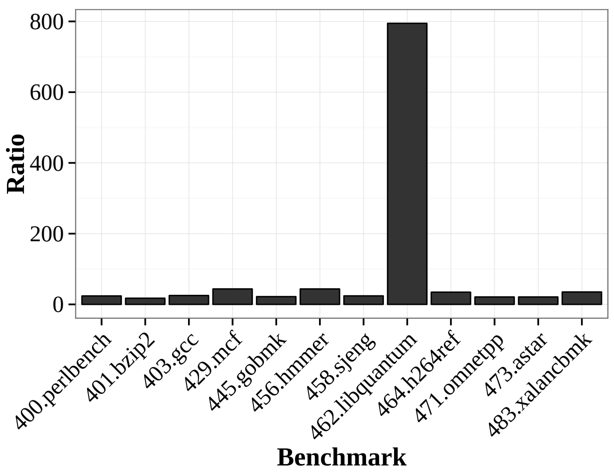

Figure 2: Speedups for all SPECint 2006 applications for the Actina Solar 220 X3 system

On the other hand, Lilja [11] advocates that the geometric mean of ratios does not represent the performance metric that we are ultimately interested in (i.e. shortest execution time). He suggests using ratio of arithmetic means. Calculating the speedup of the Actina system relative to the base system defined by SPEC (i.e. geometric mean of ratios) results in a value of 35.87 while using the ratio of arithmetic means results in a value of 31.50.

3.1 Checking variability

We verify the hazard of assuming that computer data follow a normal distribution in an experiment of 10000 executions of the benchmark libquantum from the SPEC CPU 2006 suite and empirically show that the data distribution may not be normal. Due to time constraints, we select the *test* input subset of the libquantum and run the application on an Intel Core 2 Quad at 2.4GHz with 4GB RAM.

Figure 3 shows the probability density function generated by Naive Normality Fitting (NNF) – bell curve line – and Kernel Parzen Window (KPW) – line above the gray area. The NNF technique assumes a normal distribution and plots the conventional bell curve, while KPW displays a smoothed picture of the actual distribution. Following the same methodology presented by Chen *et al.* [7], we visually compare the plots against each other and determine that the data do not follow a normal distribution. We also apply Lilliefors test (Kolmogorov-Smirnov test) [12] on the measurements and confirm that they were not taken from a population that follows a normal distribution.

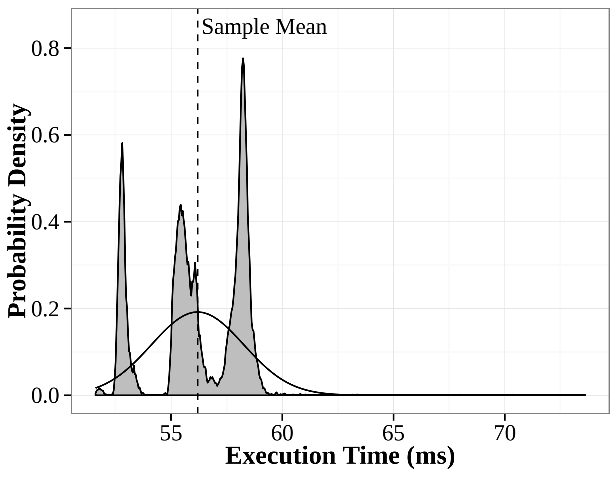

Figure 3: Probability density function for the test input of libquantum running on an Intel Core 2 Quad at 2.4GHz with 4GB RAM

Our result complies with those obtained by Chen *et al.* [7], which has shown that computer science experiments can easily happen to be non-normal. In these cases, parametric techniques that assume a normal distribution of the actual data (e.g. confidence intervals) should not be used. We advise that experimenters check if the sample distribution conforms with the normal before using any statistical framework. In this sense, SToCS implements a statistical framework to check the normality of the distribution by applying Lilliefors (Kolmogorov-Smirnov) test on a sample.

3.2 Applicability of the Central Limit Theorem

The Central-Limit Theorem (CLT) states that, even if the data are not normally distributed, the distribution of the sample means with a sufficiently large number of elements can be safely approximated by a Gaussian [6]. However, checking this approximation is usually neglected in computer science experiments [7]. This section checks the applicability of the CLT using the execution times of 1000 runs of the TeraGen Hadoop workload [5] configured to create a 8 GB HDFS file. This benchmark is one of the key workloads that are used in cloud deployments characterization [2, 17]. Due to its distributed nature, this environment presents large variability.

We characterize the I/O of TeraGen on a cluster composed of 10 AMD A8-3850 APUs which contains 4 CPU cores at 2.9 GHz, each with a 1MB L2 cache. Each node is configured with 16GB of DDR3-1333 memory, 2 SATA HD (1 TB each @ 7200 rpm) configured in RAID 0 mode, where each cluster node is connected using a 1Gbit Ethernet switch running Red Hat Enterprise Linux 6.2 and Hadoop 0.20.2.

SToCS splits the data set of 1000 points into m samples of size n so that $m \times n = 1000$. Then, it applies Lilliefors test on the m sample means to check whether they were taken from a normal distribution. The test shows that the distribution of sample means follows a normal distribution when n is greater than 60. Therefore, to provide a confidence interval for the mean, the experimenter must have at least 180 measurements to have 3 samples of 60 measurements. Such large amount of measurements may require a large amount of time to perform the experiments. This result corroborates the finds in Chen *et al.* [7], which reports that computer science experiments may require large number of measurements to use CLT.

We emphasize that this result is specific for this benchmark. This large amount of measurements may not be required for other applications in order to use CLT.

4. SYNTHETIC DATA COMPARISON

So far, we have discussed different metrics to compare computer performance. We have analyzed real computer performance using ratios of arithmetic means, geometric means of ratios, and the HPT framework. In this section, we evaluate the behavior of these metrics using synthetic data that comes from populations which have normal distribution with known mean and standard deviation.

To create the synthetic data, we create two hypothetical computers A and B, running at 2.4GHz and 2.9GHz, respectively. We generate 10 benchmarks with a varying number of instructions and define an IPC (instructions per cycle) for each benchmark, b, on each computer, c. Using the IPCs and the number of instructions of the benchmarks, we calculated the execution time, $t_{b,c}$ of each application on each machine and generate 1000 samples with a normal probability density function with mean $t_{b,c}$ and standard deviation $t_{b,c}/100$. Table 1 presents these values.

We define, for this experiment, that the total execution time of all applications is the metric that interest us, as suggested by Lilja [11]. Since we generate the samples of execution time with a normal probability density function, we know the population mean, which is the true execution time of each benchmark; hence, we also know the true ratios between the total execution time on each computer. By using SToCS to calculate the geometric mean of ratios, the ratios of arithmetic means, and the HPT-speedup, we evaluate which computer is faster when running the whole benchmark suite. Then, we compare these metrics, shown in Figure 4, with the expected values.

Table 1: Benchmark characteristics and IPCs for hypothetical computers A (2.4GHz) and B (2.9GHz)

Bmk.	Number of Instructions	IPC		Time (s)	
		A	B	A	B
a	$9.770 \cdot 10^{12}$	2.86	2.64	1423	1276
b	$9.650 \cdot 10^{12}$	3.04	0.69	1322	4822
c	$8.050 \cdot 10^{12}$	0.82	2.66	4090	1043
d	$9.120 \cdot 10^{12}$	1.46	2.44	2602	1288
e	$10.490 \cdot 10^{12}$	0.48	1.28	9105	2825
f	$9.330 \cdot 10^{12}$	1.74	1.60	2234	2010
g	$12.100 \cdot 10^{12}$	1.23	1.00	4098	4172
h	$20.720 \cdot 10^{12}$	3.42	2.37	2524	3014
i	$22.130 \cdot 10^{12}$	2.32	1.12	3974	6813
j	$6.250 \cdot 10^{12}$	1.58	1.15	1648	1874
Total	$117.61 \cdot 10^{12}$			33025	29142

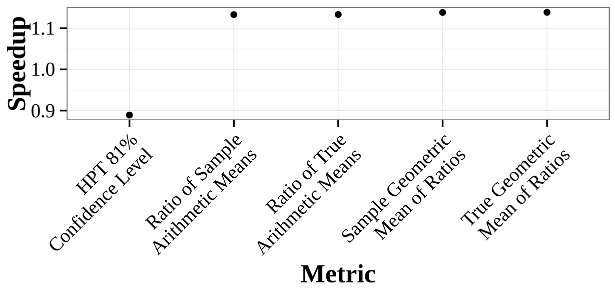

Figure 4: Performance comparison using different metrics

As expected the sample geometric mean of the ratios and the sample ratio of arithmetic means for the total execution time are very close to the expected result (i.e. Computer B is 1.13 times faster then Computer A). However, we point that HPT might not be comparable to geometric mean of ratios or to ratio of arithmetic means. HPT uses hypothesis tests and, in this particular scenario, it is not able to infer which computer is faster rendering the test inconclusive.

5. CONCLUSION

In this paper we propose SToCS, a tool that assists in the analysis of experimental data by centralizing several statistical frameworks. We used our tool to evaluate these techniques and concluded that they present similar results.

Confidence intervals give an estimate of the precision of measurements, but they require knowledge about the actual distribution of these errors. This requirement does not apply to HPT. On the other hand, the results obtained by this tool indicate that it is conservative when compared to other summarization techniques.

We conclude that publications in computer science should include the maximum amount of information available and that statements about the comparison of computers should be as clear as possible, making it easier for readers to understand the choices and decisions of the authors.

6. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions to improve this paper. They are also grateful to FAPESP, CNPq and CAPES for their financial support.

7. REFERENCES

[1] American Physics Society. What is science? http://www.aps.org/policy/statements/99_6.cfm.

[2] C. Bennett, R. L. Grossman, D. Locke, J. Seidman, and S. Vejcik. Malstone: towards a benchmark for analytics on large data clouds. In *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[3] E. Borin and Y. Wu. Characterization of DBT overhead. In *Proceedings of the IEEE International Symposium on Workload Characterization*, 2009.

[4] D. Bruening, T. Garnett, and S. Amarasinghe. An infrastructure for adaptive dynamic optimization. In *Proceedings of the International Symposium on Code Generation and Optimization*, 2003.

[5] J. Bughin, M. Chui, and J. Manyika. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch, 2010.

[6] L. L. Cam. The central limit theorem around 1935. *Statistical Science*, 1986.

[7] T. Chen, Y. Chen, Q. Guo, O. Temam, Y. Wu, and W. Hu. Statistical performance comparisons of computers. In *Proceedings of HPCA-18*, 2012.

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

[9] P. J. Fleming and J. J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Commun. ACM*, 1986.

[10] J. D. Hiser, D. W. Williams, W. Hu, J. W. Davidson, J. Mars, and B. R. Childers. Evaluating indirect branch handling mechanisms in software dynamic translation systems. *ACM Transactions on Architecture and Code Optimization*, 2011.

[11] D. J. Lilja. *Measuring computer performance: a practiotioner's guide.* Press Syndicate of the University of Cambridge, 2000.

[12] H. W. Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 1967.

[13] Statistical Tool for Computer Science (SToCS). http://lampiao.lsc.ic.unicamp.br/stocs/.

[14] J. R. Mashey. War of the benchmark means: time for a truce. *SIGARCH Comput. Archit. News*, 2004.

[15] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. Producing Wrong Data Without Doing Anything Obviously Wrong! In *Proceedings of the International Conference on architectural support for programming languages and operating systems*, 2009.

[16] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[17] N. B. Rizvandi, J. Taheri, and A. Zomaya. On using pattern matching algorithms in mapreduce applications. In *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops*, 2011.