

# Efficiently Evaluating Speedup Using Sampled Processor Simulation

Yue Luo and Lizy K. John

Department of Electrical and Computer Engineering, University of Texas at Austin

Email: {luo, ljohn}@ece.utexas.edu

**Abstract**—Cycle accurate simulation of processors is extremely time consuming. Sampling can greatly reduce simulation time while retaining good accuracy. Previous research on sampled simulation has been focusing on the accuracy of CPI. However, most simulations are used to evaluate the benefit of some microarchitectural enhancement, in which the speedup is a more important metric than CPI.

We employ the ratio estimator from statistical sampling theory to design efficient sampling to measure speedup and to quantify its error. We show that to achieve a given relative error limit for speedup, it is not necessary to estimate CPI to the same accuracy. In our experiment, estimating speedup requires about 9X fewer instructions to be simulated in detail in comparison to estimating CPI for the same relative error limit. Therefore using the ratio estimator to evaluate speedup is much more cost-effective and offers great potential for reducing simulation time. We also discuss the reason for this interesting and important result.

## I. INTRODUCTION

Simulation of processors is the most important tool for computer architects to study design tradeoffs. Modern benchmarks, such as SPECcpu2000, are close to real world applications and execute hundreds of billions instructions. As a result, simulating a complete run of a benchmark in a cycle accurate simulator is extremely time consuming and requires days to weeks of simulation time. Sampling is a technique that can greatly reduce the simulation time while retaining good accuracy. Although sampling can be applied to a cache simulator to study the cache miss rate, in this paper we are interested in the sampling of cycle accurate simulation.

Previous research [3, 5-14] focuses on the accuracy of CPI or IPC. However, the goal of a simulation is usually to evaluate the benefit of some microarchitectural enhancement, in which case, an accurate estimate of the relative performance improvement is more desirable than the absolute value of CPI. We use *speedup* to evaluate the benefit of microarchitectural enhancement. We define the speedup, denoted by  $R$ , as the ratio of the execution times before and after the microarchitectural enhancement when the same benchmark is run. Assuming that the clock frequency remains the same, the speedup is also the ratio of the CPIs before and after the enhancement since the instruction count

remains the same.

The terminology used in this paper follows that from traditional statistical sampling theory. The original full instruction stream<sup>1</sup> is divided into  $N$  non-overlapping chunks of  $m$  continuous instructions. Each chunk is a basic simulation unit or a *sampling unit*. The *sampling unit size* is the number of instructions in each chunk ( $m$ ). The *population* refers to all the chunks that constitute the full instruction stream. A *sample* consists of selected chunks that are actually simulated and measured. (In practice, more instructions are simulated for warming up the microarchitectural structures.) The number of sampling units in a sample is the *sample size* expressed as  $n$ .

We not only want to estimate the value of the speedup but also want to determine the error in our estimation. The error can be quantified with a limit on relative error<sup>2</sup>, which can be obtained from the confidence interval. Prior research provides confidence interval for CPI [3, 13] but none has given a method to calculate the confidence interval for speedup. A straightforward method is to run sampled simulations for two processor configurations to get two confidence intervals for CPI and then calculate the confidence interval for speedup using *interval arithmetic*. Suppose that the 95% confidence intervals before and after the architectural enhancement are  $0.820 \pm 0.020$  and  $0.617 \pm 0.015$ , which correspond to a  $\pm 2.5\%$  error. One may compute the confidence interval for speedup to be  $(0.800/0.632, 0.840/0.602)$ , a relative error of  $\pm 5\%$ . From this example, we can see that constraining the relative error in speedup to be within  $e$  would require the error in CPI to be within  $e/2$ . Because at a given confidence level the confidence interval is inversely proportional to the square root of the sample size, reducing the relative error limit in half would require simulating 4 times more instructions in detail. However, as we will see later this estimation of error limit is too pessimistic, and there is a better way to quantify the error in speedup.

## II. EVALUATING PERFORMANCE IMPROVEMENT WITH THE RATIO ESTIMATOR

The ratio estimator in the sampling theory calculates the ratio of two population means from a sample [2]. For each sampling unit, there are two characteristics,  $y_i$  and  $x_i$  ( $i=1, 2$ ,

Manuscript submitted: 23 June 2004. Manuscript accepted: 18 Aug. 2004. Final manuscript received: 26 Aug. 2004. This research is partially supported by the National Science Foundation under grant number 0113105, by IBM through a CAS award and a SUR grant, and by AMD, Intel and Microsoft Corporations.

<sup>1</sup> "Instruction stream" refers to the stream of committed instructions in this paper.

<sup>2</sup> Relative error for speedup is  $|R_{sample} - R_{real}|/R_{real}$ . When speedup is large the user is usually willing to tolerate larger absolute error in speedup. Therefore, we use the relative error instead of the absolute error in this paper.

...,  $N$ ). We randomly take a sample of size  $n$  and measure  $y_i$  and  $x_i$  for each sampled unit ( $i=1, 2, \dots, n$ ). The goal is to estimate  $R$ , the ratio of the population mean of  $y$  to the population mean of  $x$  ( $R = \bar{Y} / \bar{X} = \sum_{i=1}^N y_i / \sum_{i=1}^N x_i$ ). Based on the sampling theory,  $R$  is estimated as<sup>3</sup>

$$\hat{R} = \frac{\bar{y}}{\bar{x}} = \frac{\sum_{i=1}^n y_i}{\sum_{i=1}^n x_i}. \quad (1)$$

Assuming  $N \gg n$ , the variance for  $R$  is estimated as

$$v(\hat{R}) = \frac{(1-n/N)}{n\bar{x}^2} (s_y^2 + \hat{R}^2 s_x^2 - 2\hat{R}s_{yx}) \approx \frac{s_y^2 + \hat{R}^2 s_x^2 - 2\hat{R}s_{yx}}{n\bar{x}^2}, \quad (2)$$

$$\text{where } s_{yx} = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{n-1} \quad (3)$$

and  $s_y$  and  $s_x$  are the sample standard deviation of  $y$  and  $x$ . If the sample is large enough so that the normal approximation applies, the confidence interval for  $R$  at the confidence level of  $1-\alpha$  can be obtained as

$$(\hat{R} - z_{1-\alpha/2} \sqrt{v(\hat{R})}, \hat{R} + z_{1-\alpha/2} \sqrt{v(\hat{R})}). \quad (4)$$

where  $z_{1-\alpha/2}$  is the  $(1-\alpha/2)$  quantile of a unit normal distribution. Similarly, given a relative error limit of  $e$  at a confidence level of  $1-\alpha$ , the required sample size is

$$n \approx \frac{z_{1-\alpha/2}}{e\bar{y}} \sqrt{s_y^2 + \hat{R}^2 s_x^2 - 2\hat{R}s_{yx}} \quad (5)$$

If for each sampling unit,  $x_i$  and  $y_i$  are the CPI of the unit before and after the microarchitectural enhancement, then  $R$  is the speedup. The estimation of speedup (Equation 1) is quite intuitive and is the same as in the interval arithmetic method, but the calculation of the confidence interval is completely different. Based on the above theory, we propose the following general procedure to calculate the speedup and quantify its error.

1. Before the measurement, the user sets a target accuracy expressed as a relative error limit  $e$  at a certain confidence level  $1-\alpha$ .

2. Divide the full instruction stream into  $N$  chunks of  $m$  continuous instructions. Take a random sample of size  $n$ .

3. Measure the CPI of each sampled unit before the microarchitectural enhancement. Record all CPIs ( $x_i$ ).

4. Measure the CPI of the same sampled units after the enhancement. Record all CPIs ( $y_i$ ).

5. Calculate the speedup and its relative error limit or confidence interval with Equations 1 through 4. If the error limit meets the target accuracy, then stop. Otherwise, calculate the new sample size from Equation 5 and repeat step 3 and 4.

The key point in the procedure is to make sure that the same sampled units are measured in the two simulation steps (steps 3 and 4 above). But the instruction stream may be different in each run of the same benchmark. For a user mode simulator like SimpleScalar [1], this is normally caused by operating system calls (e.g. *gettimeofday*) returning different result in each run. For example, in two runs of *gcc-166*, the difference in the number of dynamic instructions was

332,372. It would cause different units to be sampled in the two runs if a small sampling unit size is used (e.g. 100-10,000 instructions as in SMARTS [13]). To solve this problem, we capture the *io* trace with SimpleScalar *sim-io* utility in our experiments. Using the *io* trace guarantees the same instruction sequence each time the benchmark is simulated.

### III. EXPERIMENTS AND RESULTS

We conducted experiments to study the application of the ratio estimator in sampled processor simulation. The procedure in the previous section does not specify how to measure the CPI of each sampling unit. It can be done by simulating the complete benchmark and switching between cycle accurate mode and functional simulation modes [6, 13]. Or it can be done by checkpointing the state before each sampling unit and simulating these checkpoints directly. In our experiment, we use the former. Caches and branch predictors are continuously warmed up functionally to approximate perfect warm up [13]. To warm up other microarchitecture structures, 4,000 instructions before each sampling unit are simulated in the cycle accurate mode. An 8-way and a 16-way out-of-order superscalar processor are simulated to calculate the speedup. The microarchitecture configurations given in Table I are adapted from [13].

TABLE I. PROCESSOR CONFIGURATIONS

Parameter	8-way (baseline)	16-way
Machine Width	8	16
RUU/LSQ size	128/64	256/128
Memory System	32KB 2-way L1 I & D, 2 ports, Unified 1M 4-way L2	64KB 2-way L1 I & D, 4 ports, Unified 2M 8-way L2
ITLB / DTLB	4-way 128 entries 4-way 256 entries 200 cycle miss penalty	4-way 128 entries 4-way 256 entries 200 cycle miss penalty
L1/L2/Memory Latency	1/12/100 cycles	1/16/100 cycles
Functional Units	4 I-ALU 2 I-MUL/DIV 2 FP-ALU 1 FP-MUL/DIV	16 I-ALU 8 I-MUL/DIV 8 FP-ALU 4 FP-MUL/DIV
Branch Predictor	Combined 2K tables 7 cycle misprediction penalty 1 prediction/cycle	Combined 8K tables 10 cycle misprediction penalty 2 predictions/cycle

Eight benchmarks from SPECcpu 2000 are simulated in a modified SimpleScalar 3.0 *sim-outorder* simulator that does the sampling. We use three sampling unit sizes: 10,000 instructions, 1 million instructions, and 10 million instructions. Wunderlich et al [13] report that the optimal sampling unit size is in the range of 100 to 10,000 instructions in their experiment setup. We choose the size of 10,000 instructions from their study. In simulation of superscalar processors, instructions at the beginning and the end of each sampling unit may account for only a partial cycle. These partial cycles must be counted accurately when the sampling unit size is small. We have also noticed that the warm up error becomes pronounced with small sampling units for some benchmarks. To avoid the problem of partial cycles and warm up error, we did not use sampling unit size below 10,000 instructions. Sampling unit sizes of 1 million and 10 million instructions are used in the latest Variance and Early SimPoint method [8]. We choose an initial sample size

<sup>3</sup>Capital letters refer to characteristics of the population and lowercase letters to those of the sample. The symbol  $\hat{\cdot}$  denotes an estimate of a population characteristic made from a sample.

of 3,000 for the sampling unit size of 10,000 instructions, and an initial sample size of 1,000 for sampling unit sizes of 1 million and 10 million instructions.

We set a relative error limit of 2% at 95% confidence level as our example target accuracy. After the initial sampling, we calculate from Equation 5 the sample size required to achieve the target accuracy for speedup. For comparison, we also calculate the sample size required to achieve the same target accuracy for CPI using standard equations [3, 13]. The result is drawn in Figure 1. It shows that for most benchmarks the sample size for measuring speedup is only a small fraction of that for measuring CPI. We calculate the ratio of sample size for speedup to the sample size for CPI on the 16-way issue processor. The geometric mean of the ratio for the benchmarks is 0.127 (for Figure 1a), 0.107 (for Figure 1b) and 0.115 (for Figure 1c). Therefore, it is more cost-effective to directly measure the speedup than to measure the CPI. In other words, to achieve a relative error limit of 2% on the speedup, we do not need to estimate CPI to the same relative error limit. Instead, we need to measure only 1/9 of the instructions required for estimating CPI to the same relative error limit. Compared with the interval arithmetic method, the savings are even more. The interval arithmetic method yields the same value of speedup but is far too pessimistic in quantifying its error. As discussed in Section I, the interval arithmetic method would require that the relative error limit for the CPI be reduced to half of 2%, which would in turn quadruple the sample size. In our experiment configuration, using the ratio estimator technique the geometric mean of the reduction in sample size will be about 36X compared to the interval arithmetic method. Therefore, using the ratio estimator equations can significantly reduce the sample size. It is also worth noting that using different target accuracy will not change the savings because varying the target accuracy equally affects the ratio estimator method and the arithmetic interval method.

There is no reason to doubt the ratio estimator theory, but we still want to verify that the computed sample size does meet the target accuracy. We modify the cycle accurate simulator *sim-outorder* to dump the CPI for every sampling unit in the benchmark execution so that we can get the population and the real speedup value  $R_{real}$ . We use the Monte Carlo method to validate the target accuracy. Suppose the computed sample size is  $n$ . We take a random sample of size of  $n$  from the population and compute the speedup from this sample using Equation 1. Then we take another random sample of size  $n$  from the population and compute the speedup again. We repeat the process many times (10,000 times in our experiment). If 95% of these speedup values lie within  $\pm 2\%$  of the real value  $R_{real}$ , then the computed sample size  $n$  meets our target accuracy. If a much lower percentage of speedup values are within the relative error limit, then our computed sample size is too small and a larger sample size is required.

The verification results for sampling unit sizes of 1 million and 10 million instructions are shown in Table II. Columns 2 and 4 show the sample size calculated from Equation 5, which is the same as in Figure 1b and 1c. Columns 3 and 5 give the percentage of the speedup values that lie within the

relative error limit of  $\pm 2\%$ . Ideally, this percentage should exactly be 95% but there is inevitably some error in Monte Carlo experiment, and the ratio estimator itself has a slight bias (see [2] for detail), so the percentage numbers in the table are not exactly 95%. However, none of the numbers are much lower than 95%. Therefore, we can conclude that the sample size computed from the ratio estimator theory does satisfy the target accuracy requirement. We did not verify for the sampling unit size of 10,000 instructions. Firstly, the data set of population is large, thus difficult to process. Secondly, with sampling unit size of 10,000 instructions the warm up error for some benchmarks is not negligible with respect to the  $\pm 2\%$  error limit. Therefore, we chose not to verify for this sampling unit size.

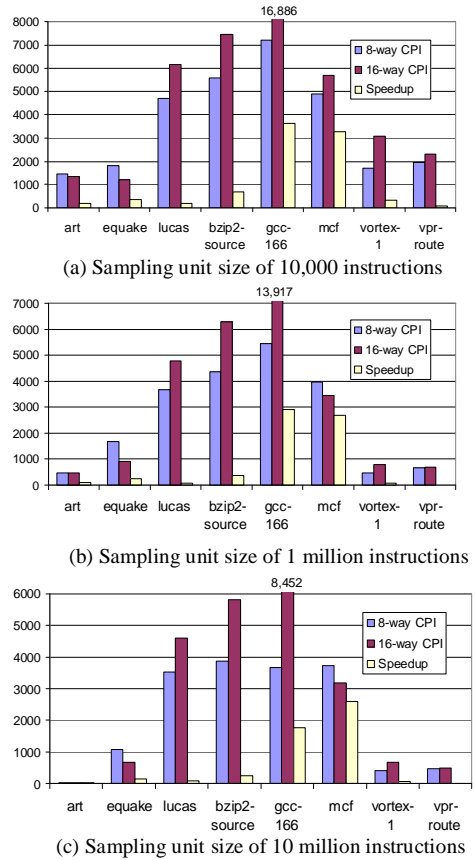


Fig. 1. Sample sizes required to achieve relative error limit of 2% at the confidence level of 95% for selected SPECcpu2000 benchmarks.

The above result may be surprising at first glance. How could the speedup be more accurate than the CPIs from which it is computed? This is because the two configurations being evaluated are usually not fundamentally different. The CPI values may vary widely during the benchmark execution, but if the CPI is high (or low) for one part of the code on the first configuration, then the CPI for this part of code is probably also high (or low) on the second configuration. Thus the CPIs are usually correlated and the ratio of the two CPIs (i.e. the speedup) does not change as much as the CPIs themselves. At a given accuracy, the sample size is largely determined by the variation normalized to the mean (the coefficient of variation). The small variation in the speedup leads to the

small sample size. Figure 2 shows a graph of CPIs and the speedup for a small portion in the execution of *vortex-1*. All the metrics (CPIs and speedup) are normalized to their respective mean so that we can compare the normalized variation. It is obvious that the CPIs on the two configurations follow each other and the variation in the speedup is much smaller than in the CPIs<sup>4</sup>. As a result, estimating the speedup requires a smaller sample size.

TABLE II. VERIFICATION OF THE SAMPLE SIZE COMPUTED FROM RATIO ESTIMATOR THEORY

Benchmark	Sampling unit size of 1 million instructions		Sampling unit size of 10 million instructions	
	Sample size	Percentage within error limit	Sample size	Percentage within error limit
art	101	95.0%	24	95.4%
equake	254	94.7%	147	95.3%
lucas	87	94.7%	84	94.8%
bzip2-source	359	95.0%	254	95.2%
gcc-166	2902	96.0%	1769	98.8%
mcf	2694	95.7%	2587	99.0%
vortex-1	76	95.2%	60	95.2%
vpr-route	12	94.8%	8	95.3%

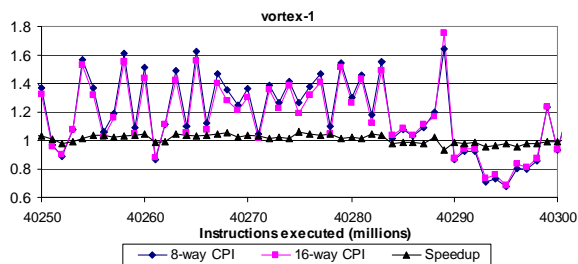


Fig. 2. Normalized CPIs and Speedup. Each data point is for one million instructions.

#### IV. CONCLUSION AND FUTURE RESEARCH

Simulation of processors is mostly used for evaluating the benefit of some microarchitectural enhancement, in which the speedup is a more important metric than the CPI. We apply the ratio estimator method from sampling theory to sampled processor simulation to quantify the error of speedup. We show that to achieve a given error limit for speedup it is not necessary to estimate CPI to the same accuracy. For the same relative error limit, measuring speedup requires fewer instructions to be simulated than measuring CPI. In our experiments using the ratio estimator to estimate speedup results in a sample size 9X smaller than estimating CPI, and 36X smaller than estimating speedup using interval arithmetic.

This technique has great potential to reduce simulation time for computer architects. Especially when checkpoint files or trace files are used and each sampling unit is simulated directly (with explicit warm up [4]), the reduction in the sample size will directly translate into savings in storage space and simulation time. A 9X smaller sample size will result in 9X shorter simulation.

The research in this paper only employs simple random

<sup>4</sup>Note that the absolute value of speedup is about 1.5. The impression that the CPIs for the two configurations are almost the same is just an artifact of normalization.

sampling. The required sample size is still large for some benchmarks. In future work we will combine more sophisticated sampling algorithms [8, 14] with the ratio estimator to further reduce simulation cost.

In real-world simulations, the microarchitectural changes are often smaller than those in our experiment, so the CPIs may be even more correlated and the ratio estimator technique is probably highly effective. However, it is important in future work to explore the limitation of the technique and to provide guidelines to assess its effectiveness if drastically different configurations are being compared.

#### REFERENCES

- [1] D. Burger, and T. M. Austin. "The SimpleScalar tool set, version 2.0," Technical Report 1342, Computer Sciences Department, University of Wisconsin-Madison, June 1997.
- [2] W. G. Cochran, *Sampling Techniques*, 3rd ed. John Wiley & Sons, 1977.
- [3] T. M. Conte, M. A. Hirsch, and K. N. Menezes, "Reducing state loss for effective trace sampling of superscalar processors," In *Proceedings of the 1996 International Conference on Computer Design (ICCD)*, Oct 1996, pp. 468-477.
- [4] J.W. Haskins, Jr. and K. Skadron, "Memory Reference Reuse Latency: Accelerated Sampled Microarchitecture Simulation," In *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, Mar. 2003, pp. 195-203.
- [5] V. S. Iyengar, L. H. Trevillyan, P. Bose, "Representative Traces for Processor Models with Infinite Cache", In *Proceedings of the 2nd IEEE Symposium on High-Performance Computer Architecture*, 1996.
- [6] L. M. Jimeno-Ochoa, P. Ibez, and V. Vials, "Warm time sampling: fast and accurate simulation of cache memory," In *Proceedings of the 22nd. Euromicro International Conference*, September 1996, pp. 39-44.
- [7] W. Liu and M. Huang. "EXPERT: Expedited Simulation Exploiting Program Behavior Repetition," In *Proceedings of the International Conference on Supercomputing (ICS'04)*, Jun. 2004.
- [8] E. Perelman, G. Hamerly, and B. Calder. "Picking statistically valid and early simulation points," In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, September 2003, pp. 244-255.
- [9] T. Sherwood, E. Perelman, and B. Calder. "Basic block distribution analysis to find periodic behavior and simulation points in applications," In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, September 2001, pp. 3-14.
- [10] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. "Automatically characterizing large scale program behavior," In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002, pp. 45-57.
- [11] K. Skadron, P. S. Ahuja, M. Martonosi, and D. W. Clark. "Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques," *IEEE Trans. Computers*, vol. 48, Nov. 1999, pp. 1260-1281.
- [12] R. Todi. "SPECLite: Using Representative Samples to Reduce SPEC CPU2000 Workload," IEEE 4th Annual Workshop on Workload Characterization. 2001.
- [13] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. "SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling," In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, June 2003, pp. 84-95.
- [14] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe "An Evaluation of Stratified Sampling of Microarchitecture Simulations," Third Annual Workshop on Duplicating, Deconstructing, and Debunking, June 20, 2004.