ADDRESSING WORKLOAD VARIABILITY IN ARCHITECTURAL SIMULATIONS

THE INHERENT VARIABILITY OF MULTITHREADED COMMERCIAL WORKLOADS CAN LEAD TO INCORRECT RESULTS IN ARCHITECTURAL SIMULATION STUDIES. RANDOM PERTURBATIONS AND STATISTICAL METHODS HELP COMPUTER ARCHITECTS DRAW VALID CONCLUSIONS.

Alaa R. Alameldeen
David A. Wood
University of Wisconsin-Madison

Multithreaded, throughput-oriented commercial applications, such as databases and Web servers, represent a dominant class of Internet service workloads. Computer architects increasingly optimize current and future server architectures (such as multithreaded processors and chip multiprocessors) for these important workloads. Architects use multithreaded benchmarks to evaluate alternative designs, by measuring current systems and simulating future ones. Execution-driven evaluation of such benchmarks requires full-system simulation, because these benchmarks spend a significant portion of their time in the operating system.[1]

Performance variability presents a major challenge for architectural simulation studies using multithreaded workloads.[2] *Variability* refers to the differences between multiple estimates of a workload's performance on a given system configuration. If not addressed, variability can cause computer architects to draw incorrect conclusions from their simulation experiments. Figure 1 shows that simulation variability causes a system with an 84-nanosecond (ns) dynamic RAM (DRAM) access latency to outperform a system with a faster, 81-ns DRAM access latency. Taken at face value, ignoring variability in this case leads to the incorrect conclusion that a slower memory performs better!

Architectural simulations incur two types of variability. *Time variability* occurs when a workload exhibits different characteristics during different phases of a single simulation run. This leads to errors when the simulated program phase does not represent the workload's average behavior. *Space variability* occurs when small timing variations cause runs starting from the same initial state to follow different execution paths from the space of all possible paths. This leads to errors when timing differences in two configurations result in the simulated workload taking widely divergent paths with different performance characteristics.

Although most architectural simulation studies ignore space variability's effects, our results demonstrate that space variability has serious implications for architectural simulation studies using multithreaded workloads.[2] The standard solution—running long enough—does not easily apply to simulation because of its enormous slowdown. To address this problem, we propose a simulation methodology combining multiple simulations with standard statistical techniques, such as

Published by the IEEE Computer Society

confidence intervals and hypothesis testing. This methodology greatly decreases the probability of drawing incorrect conclusions, and permits reasonable simulation times given sufficient simulation hosts.

What causes variability?

Time variability is a well-known phenomenon that earlier work frequently describes as phase behavior.[3] We prefer the more general term—time variability—because the throughput-oriented commercial workloads we study have a more homogeneous long-term behavior. Space variability is also well-known in the measurement community. Space variability can arise in any parallel or multithreaded workload where small timing variations can result in different execution paths, perhaps yielding different performance characteristics.

Small-scale variations arise in real systems due to factors such as interrupt timing and bus contention with direct memory access (DMA). Variations arise in simulation due to changes in system parameters (such as cache size, associativity, or miss latency). Small-scale variability can have a much larger effect on simulation results for several reasons, including:

- The operating system might make different scheduling decisions (for example, a scheduling quantum might end before an I/O event in one run, but not in another).[2]
- Threads might acquire locks in a different order, resulting in a different execution order.
- A transaction might complete during the measurement interval in one run, but not another.

Measurement experiments generally address time and space variability by averaging multiple observations or running long enough to minimize variability's effect. In previous experiments on a real system, our online transaction processing (OLTP) benchmark's performance varied by up to a factor of three, for measurement intervals of one second.[2, 4] Variability's effect on results largely vanishes for longer measurement intervals (60 seconds, for example). Unfortunately, running this long is impractical for detailed architectural simulations. Modeling a 16-processor system with out-of-order processors on a uniprocessor host, our simulation system slows down by approximately 24,000 times. This means that simulating 60 seconds of target execution time requires more than 16 days!

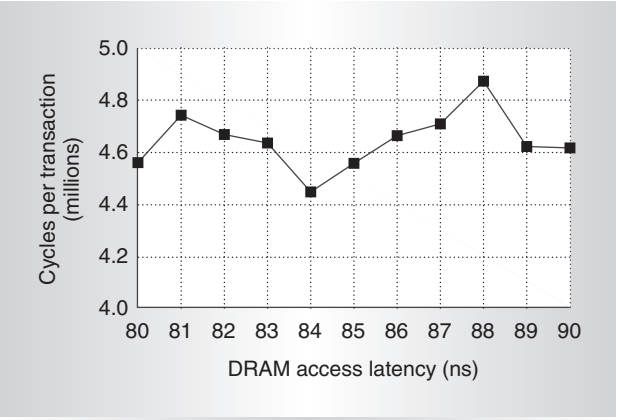

Figure 1. Performance of 500-transaction online transaction processing (OLTP) runs with different DRAM latencies.

Does variability matter for simulation?

Computer architects often use simulation to evaluate a design enhancement's performance relative to a base design. In this case, they care less about absolute performance than about a workload's relative performance on two (or more) different system configurations.

We conducted a simple experiment to illustrate that space variability can have enough effect on simulation results to cause incorrect conclusions. Figure 1 plots cycles per transaction as the DRAM access latency of a simulated multiprocessor increases from 80 to 90 ns, with all other system parameters fixed. All runs start from the same simulation checkpoint and complete 500 OLTP transactions. Intuitively, we expect that cycles per transaction should increase slightly with the increase in DRAM latency. However, small differences in memory system timing result in different execution paths, with very different execution times. For example, the system with 84-ns DRAM ran 7 percent faster than the one with 81-ns DRAM. Digging deeper, we found that after approximately 560,000 cycles, the operating system made different scheduling decisions in the two runs, resulting in very different execution paths. Although no computer architect is likely to conclude that slower memory leads to faster systems, clearly

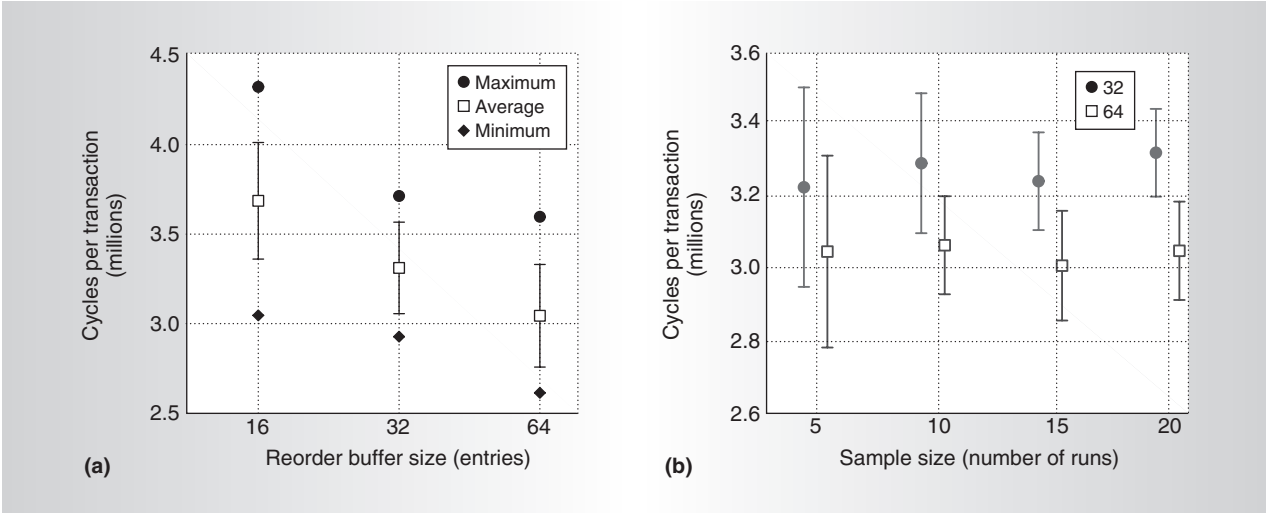

Figure 2. Performance variability of OLTP (cycles per transaction): Variability for different reorder buffer sizes and 20 simulation runs (a); 95 percent confidence intervals for different sample sizes and two reorder buffer sizes (b).

space variability could lead to incorrect conclusions in other, less intuitive studies.

Injecting variability into simulations

Workloads exhibit space variability on real systems due to small timing variations. Our simulator, however, is deterministic: It produces the same execution path every time for each combination of workload and system configuration. To evaluate space variability, we must inject small timing variations to create a space of possible executions starting from the same initial conditions.

To do this, we artificially introduce small perturbations (disturbances) in memory system timing. On each Level 2 cache miss, we add a uniformly distributed pseudo-random integer between 0 and 4 ns. For multiple simulations, each run uses a unique random seed, leading to a different sequence of miss latencies, but the same average miss latency. These small perturbations lead to different execution paths and different runtime results. This models, though at a smaller scale, the variations in memory access time due to DRAM refreshes or I/O.

This artificial method of introducing timing perturbations produces a space of runs for our simulation experiments. We use the mean of these runs as our performance metric.

The wrong conclusion ratio

Most architectural simulation studies evaluate workloads' performance on two (or more) system configurations using a single simulation run for each combination of workload and system configuration. For workloads with significant space variability, using only a single run incurs nontrivial risk of drawing a wrong conclusion (as in Figure 1).

To illustrate this risk, we present experimental results for multiple simulation runs and compute the wrong conclusion ratio (WCR), which is the percentage of comparison experiment pairs that reach an incorrect conclusion. For example, when conducting an experiment to compare the performance of systems A and B, we use N runs of workload W on each system. The correct conclusion is the relationship between the averages of the N runs on A and the N runs on B (for example, A outperforms B). The WCR is the percentage of the N^2 pairs of runs that lead to the opposite conclusion (in this case, that B outperforms A).

The experiment compares the performance of three systems whose processors have three different reorder buffer (ROB) sizes. We simulated twenty, 50-transaction runs for our OLTP workload using timing-first simulation for systems that have reorder buffers of 16, 32, and 64 entries, respectively.[5] Figure 2a shows the average (with error bars representing +/- one standard deviation), maximum, and minimum number of cycles per transaction for the three systems. The averages confirm the expected conclusion that runtime decreases

when the ROB size increases. However, the result ranges overlap (for example, the minimum of the 16-entry system is smaller than the maximum of the 64-entry system), leaving the possibility of an incorrect conclusion.

To estimate the risk of a wrong conclusion, we computed the WCR values for each pair of experiments: WCR (16, 32) = 18 percent, WCR (16, 64) = 7.5 percent, and WCR (32, 64) = 26 percent. This means, for example, that 26 percent of the possible experiment pairs lead to the wrong conclusion that a 32-entry ROB system outperforms a 64-entry system. This demonstrates that ignoring variability can lead to incorrect conclusions, even for simple microarchitectural simulation experiments.

Accounting for space variability

Classical statistics provides many techniques for coping with variability. We can use several of these methods to account for simulation variability while limiting simulation time.

Most statistical methods call for running multiple trials and estimating the true performance using the arithmetic mean of the sample. The larger the sample size, the more accurate the performance estimate. Thus, to account for space variability, we need to simulate enough runs to reduce the probability of drawing wrong conclusions to an acceptable level. Fortunately, because we can run independent simulations in parallel on different machines, the elapsed wall-clock time remains essentially constant, given sufficient host machines.

We apply two standard statistical techniques—confidence intervals and hypothesis testing—to estimate the probability of drawing wrong conclusions. The *wrong conclusion probability* estimates the probability of drawing wrong conclusions about the relationship between two configurations. Confidence intervals place a conservative upper limit on the wrong conclusion probability. Hypothesis testing, which we discussed in our first paper on this topic, provides a tighter, more accurate estimate of the wrong conclusion probability, allowing us to achieve the same confidence level using a smaller sample.[2]

Confidence intervals

A *confidence interval* (CI) is the range of values that we expect to include a population parameter (such as the mean).[6] The *confidence probability* is the probability that the true population parameter will fall inside the confidence interval. You can use confidence intervals to estimate an upper limit on the wrong conclusion probability when comparing two alternatives. If the two alternatives' confidence intervals do not overlap, the probability of reaching a wrong conclusion will be at most $(1-p)$, where p is the confidence probability.[2]

Figure 2b shows the 95-percent confidence intervals for the ROB design experiment with different sample sizes. As expected, the confidence intervals get tighter as the sample size increases. Confidence intervals for samples of 20 runs do not overlap, which implies that the probability of reaching a wrong conclusion is less than 5 percent (compared to the 26 percent WCR for single experiments). For the smaller sample sizes, the results are not statistically significant (at the 95-percent confidence level), because the confidence intervals overlap. If we reduce the confidence probability to 90 percent, a sample size of 15 runs becomes statistically significant, but at most a 10-percent chance remains of reaching a wrong conclusion.

Although more conservative than hypothesis testing, confidence intervals are more convenient to use in architectural simulation studies. As Figure 2b illustrates, adding error bars to ubiquitous line and bar graphs clearly indicates the results' significance, while requiring little additional effort.

Time and space variability are important phenomena that architectural simulation studies using multithreaded workloads must address. The standard practice of ignoring variability can lead to incorrect conclusions in a significant percentage of microarchitectural and system design simulation experiments. Our simple methodology compensates for variability by combining pseudo-random perturbations, multiple simulations, and standard statistical techniques. This methodology helps computer architects determine when they can safely draw conclusions from simulation experiments. MICRO

Acknowledgments

We thank the Wisconsin Multifacet project, Mark Hill, Virtutech AB, the Wisconsin Condor group, and reviewers of our first paper on

this topic for their contributions and support. This work is supported in part by the National Science Foundation with grants CCR-0324878, EIA-0205286, and EIA-9971256, a Wisconsin Romnes Fellowship (Wood) and donations from IBM, Intel, and Sun Microsystems. Prof. Wood has a significant financial interest in Sun Microsystems, Inc.

References

1. L.A. Barroso, K. Gharachorloo, and E. Bugnion. "Memory System Characterization of Commercial Workloads," *Proc. 25th Int'l Symp. on Computer Architecture* (ISCA 98), pp. 3–14.
2. A.R. Alameldeen and D.A. Wood, "Variability in Architectural Simulations of Multi-threaded Workloads," *Proc. 9th Int'l Symp. on High-Performance Computer Architecture* (HPCA 03), pp. 7–18.
3. T. Sherwood, et al., "Automatically Characterizing Large Scale Program Behavior," *Proc. 10th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (ASPLOS-X), 2002, pp. 45–57.
4. A.R. Alameldeen et al., "Simulating a $2M Commercial Server on a $2K PC," *Computer*, vol. 36, no. 2, Feb. 2003, pp. 50-57.
5. C.J. Mauer, M.D. Hill, and D.A. Wood, "Full System Timing-First Simulation," *Proc. 2002 ACM Sigmetrics Conf. on Measurement and Modeling of Computer Systems*, ACM Press, 2002, pp. 108-116.
6. H. Frank and S. G. Althoen, *Statistics: Concepts and Applications*, Cambridge Univ. Press, first edition, 1994, pp. 274-452.

Alaa R. Alameldeen is a PhD candidate in the computer sciences department at the University of Wisconsin-Madison. His research interests include chip multiprocessor memory system design and performance evaluation of multithreaded workloads. Alameldeen has master's degrees in computer science from Alexandria University, Egypt and from the University of Wisconsin-Madison. He is a student member of IEEE and ACM.

David A. Wood is a professor and Romnes Fellow in the computer sciences department and the electrical and computer engineering department at the University of Wisconsin-Madison. His research interests include techniques for improving the availability, designability, programmability, and performance of commercial multiprocessor servers. Wood has a PhD in computer sciences from the University of California, Berkeley. He is a Fellow of the IEEE, and a member of the IEEE Computer Society and the ACM.

Direct questions and comments about this article to Alaa R. Alameldeen, 1210 W. Dayton St., Madison, WI 53706; alaa@cs.wisc.edu.