Computer Performance Evaluation Workshop
Position Statement

Mark Oskin
Dept. of Computer Science and Engineering
University of Washington

Detailed cycle-by-cycle simulation has enabled a heightened level of empiricalness to the science of computer architecture. Recent work has been disturbing, however, in examining the faithfulness of the simulation approach. In [1] it was shown that the widely used "constant delay" memory interface was a gross simplification of the memory hierarchy. This was followed by [2, 3] where it was shown that a compounding collection of modeling errors lead to inaccuracies. In [4] it was demonstrated that the entire SPECint95 benchmark suite could be eerily summarized into a handful of statistical control parameters. Finally, in this position paper I present some new evidence that the metrics commonly used (cycles, IPC) deserve a closer inspection. After this, I briefly discuss a range of simulation topics.

The IPC / cycles / execution-time juggernaut

Figures 1,2 and 3, depict percent of total execution time as a function of instructions per cycle (IPC). This data was collected by modifying the SimpleScalar toolkit to track instruction completions every ten cycles. The result is a graph of the amount of time the superscalar processor spends performing well (completing many instructions per cycle) and performing poorly. The often-used "IPC" benchmark is really an average of microscopic execution performance. The analogy one can draw is that, reporting IPC for an application is like saying when you roll a ball down a U-shaped incline it will come to a complete stop at the bottom of the U instantaneously. We all recognize that the ball will roll down one side and up the other and eventually through friction come to rest at the bottom. A microprocessor, when viewed on a microscopic timescale, really fluctuates like this and reporting IPC as a single number masks this behavior. Unfortunately, reporting execution time also hides these microscopic fluctuations.

The focus on average IPC is problematic. On the one hand, we strive to design architectures that make applications execute faster. So it seems logical to study application execution time, or some corollary such as average IPC. On the other hand, the application-architecture interaction is extremely complex and simply looking at these gross metrics of performance can make proper interpretation difficult.

Currently, sensitivity analysis is key

So if average IPC/execution-time is problematic, what kind of metrics should we use? Until some useful way of transforming simulation results on microscopic timescales into meaningful understanding is developed, sensitivity analysis of IPC and its corollaries (domain-specific performance metrics) are useful. Sensitivity analysis is indirectly suggestive of microscopic behavior. It is difficult to generalize across domains, but to highlight a few (of many available) examples: Sazeides and Smith [5] explored data-predictability broken down by the sources and terminations of such predictability. Evers et al [6] explored the sources of branch predictability with prediction accuracy relative to history length. Lam et al [7] explored the locality of algorithms using cache miss rate against blocking factors (among other features). Finally, Brooks and Martonosi [8] explored the correlation between average power and other architectural metrics such as fetch and cache hit rates.

The decaying relevance of baseline comparisons (Value=e^{-kT})

A significant challenge with architecture research that looks forward ten to fifteen years is devising a scientifically sound exploration methodology. With some, arguable, degree of error we are able to predict the future (silicon) technological landscape. However, with far less accuracy we know what the architecture and application comparison points should be. Perhaps the best we can do is to use SPEC2000 and compare against a supped up super-scalar or VLIW. But, what is the quantitative value in this approach?

An interesting historical study may be to characterize the change in architectures, technology, and applications over the past 15 years and devise a function for the cumulative rate of change. This (exponentially compounding) rate can be inverted to place value on baseline comparisons. Returning to metrics, studies that examine sensitivity to parameters seem more appropriate for understanding. However, a token baseline comparison (a "straw man") may be required for acceptance of radically new architectures.

The need for accuracy is relative

Not all architecture research needs absolute accuracy. Research that seeks to advance the current state of the art should rely on a high degree of absolute accuracy that can be verified. On the other hand, due to the inherent inaccuracy in the technology and application projections, research that looks forward ten to fifteen years should strive for relative accuracy. In both cases, however, the current focus in constructing simulation infrastructure is on absolute accuracy. How might striving for relative accuracy change things? Here is a list of ideas that might be appropriate in certain circumstances: use random cache replacement policies instead of tracking precise LRU, replace 64 bit counters with 32 bit counters that normalize against each-other every 100000 cycles, cut all the delay constants by a factor of 2 and kludge around those with a delay of 1, for parallel systems relax the synchronization precision. Depending upon the architectural system being explored, even more obscene hybrid cycle-by-cycle/model techniques can be employed: replace the memory hierarchy with statistical access-delay models, replace the execution core with a pipeline model, or replace the application itself with a statistical code model.

It's the ideas that matter

Simulation creates a wealth of empirical results. Translating these results into useful understanding requires that the metrics are meaningful, the applications are representative, the assumptions are valid, and the simulation bug-free. These are real challenges with simulation, but certainly there are other mechanisms to explore architectural ideas. Are we seduced by the perceived precision of simulation into not exploring alternative methods?

[1] V. Cuppu, B. Jacob, B. Davis, and T. Mudge. *A Performance Comparison of Contemporary DRAM Architectures.* International Symposium on Computer Architecture (ISCA), 1999.

[2] R. Desikan, D.C. Burger, and S.W. Keckler. *Measuring Experimental Error in Microprocessor Simulation.* International Symposium on Computer Architecture (ISCA), 2001.

[3] R. Desikan, D.C. Burger, S.W. Keckler, L. Cruz, F. Latorre, A. Gonzalez, and M. Valero. *Errata on "Measuring Experimetnal Error in Microprocessor Simulation".* 2001.

[4] M. Oskin, F. Chong, and M. Farrens, *HLS: Combining Statistical and Symbolic Simulation to Guide Microprocessor Designs.* International Symposium on Computer Architecture (ISCA), 2000.

[5] Y. Sazeides and J. Smith, *Modeling Program Predictability*. International Symposium on Computer Architecture (ISCA), 1998.

[6] M. Evers and T. Yeh, *Understanding Branches and Designing Branch Predictors for High Performance Microprocessors.* Submitted to the Proceedings of IEEE.

[7] M. Lam, E. Rothberg and M. Wolf, *The Cache Performance and Optimizations of Blocked Algorithms*. Architectural Support for Programming Languages and Operating Systems (ASPLOS), 1991.

[8] D. Brooks and M. Martonosi, *Dynamic Thermal Management for High-Performance Microprocessors*. International Symposium on High-Performance Computer Architecture, 2001.

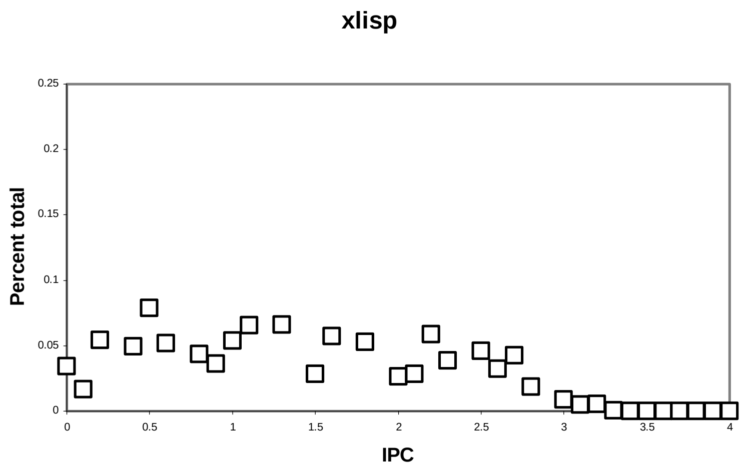

Figure 1

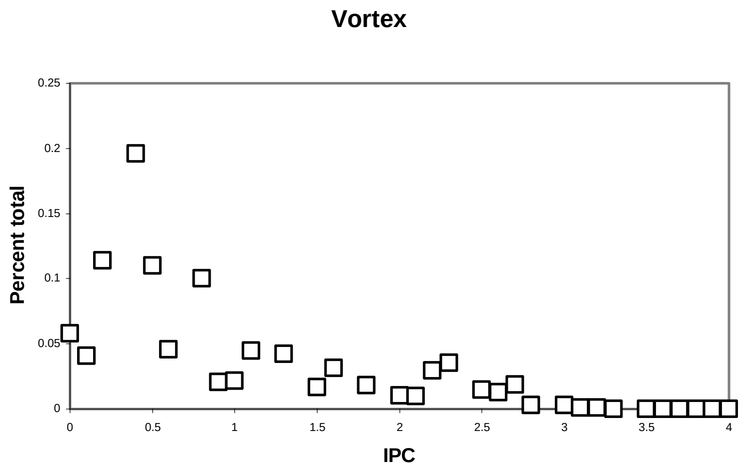

Figure 2

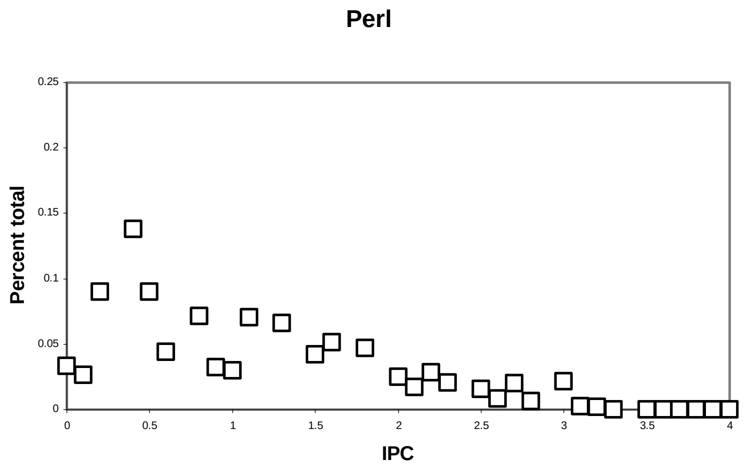

Figure 3