Memory System Simulation and the Right Energy Metric

Mark Heinrich
Computer Systems Laboratory, Cornell University, Ithaca, NY 14853

1. Introduction

While much of the focus of the workshop will be on microarchitecture simulation, I have decided to focus instead on issues regarding the simulation and evaluation of memory systems. Here by memory systems I mean anything "outside" the processor. This would include any or all of the following: lower-level processor caches, processor buses, memory controllers, physical SDRAM, multiprocessor network switches, communication or coherence protocols, I/O bridges, I/O buses, and I/O devices such as disks and ethernet controllers. Given the existence of the oft-cited memory wall [13] and situations where processor utilizations are less than 50% [4] it is clear that capturing the correct memory system behavior is critical in processor performance evaluations and may even be the most important factor determining overall execution time. As memory latencies increase relative to the processor cycle time the importance of the memory system only increases, and novel memory system techniques have become a recent focus of architects. The open question is what are the important features of simulators when evaluating tradeoffs in memory system design, and are they the same features that microarchitectural simulators need. For example, what are the right applications and benchmarks, and how important is execution-driven simulation or the complexity of the processor model.

In addition to this focus on memory systems, I would also like to espouse the use of a "new" metric when trying to simultaneously optimize both energy and performance. This metric is Et^2, where E is the energy per computation and t is the delay or time complexity per operation. Because it is independent of voltage to first-order, it is superior to other metrics like power-delay product, or energy-delay product (Et) for VLSI implementations.

2. Memory System Simulation

Microarchitectural simulations need to accurately model the memory system in terms of the number of outstanding references, the consistency model, the latency incurred at each level of the cache hierarchy, the contention on the processor bus and for the memory controller, the coherence protocol, and the proper delay at the physical memory parts. For accurate performance estimations, I believe that all of the above are necessary even for uniprocessor simulations. Multiprocessor simulations further add the need for modeling the network, which typically includes both topology concerns and contention in the network switches. Unfortunately this memory system detail only makes the job of microarchitectural simulation more difficult. However, that is not my focus here.

One could also look at architectural simulations from the perspective of a memory system designer–someone designing a memory controller, or coherence protocols, or network switches—and ask what kind of simulation technology is needed, what applications should be used, and how important is the processor model. In this case the innovations are happening in the memory system, and the memory system is expected to be simulated in detail. The question is how can these new features and organizations be evaluated quickly and fairly and therefore in how much

detail does one need to model everything else?

Trace-Driven or Execution-Driven? In processor modeling, while trace-driven simulation is useful for simple hit rate calculations, it lacks the feedback information necessary to accurately predict execution time. I also believe that execution-driven simulations are needed to properly evaluate memory system performance. Traces may be useful for evaluating sizes for node-level or directory caches, hit rates of various structures like memory-side predictors or prefetchers, or back-of-the-envelope calculations involving the number of cache misses and average bus or controller utilizations. However, hit rate is not execution time, and averages can be misleading in computer architecture, especially in memory system design where hot-spots and occupancy-induced contention can severely degrade performance [3,7]. Therefore we need to run realistic applications and benchmarks and feed back the performance of the memory system.

What are the Right Applications? For the evaluation of memory systems, applications that stress the memory system are far more interesting than compute-intensive benchmarks. The Olden benchmarks [1] are a set of linked-list intensive applications that exhibit poor cache behavior and therefore can stress the memory system. `lmbench` [11] tests a variety of memory system characteristics. Because of recent interest in active memory systems there has been a renewed interest in collecting worthwhile benchmarks that test memory system behavior, most notably the DIS stressmarks [5]. For testing multiprocessor performance there are applications from SPLASH-2 [14] and OpenMP. Despite the presence of the above, there is still a lack of memory-intensive or data-intensive benchmarks from "real" systems. Architects need to make a new and concerted effort to develop or gather more benchmarks for memory system evaluation.

How Important is the Processor Model? From a memory system evaluation point of view, the processor model only needs to produce the correct stream of references at roughly the same rate and interval as the processor in the system. Detailed modeling of the processor is not of concern to the memory system designer as long as it does not appreciably affect the contents or timing of the memory stream. Of course, that is not as simple as it sounds. Studies have shown that it is possible to achieve both accurate absolute and relative simulation accuracy using detailed memory system simulators with simple processor models [6]. Such techniques allow faster simulation times and therefore the ability to run larger data sets, often important in evaluating memory systems. The drawback of this approach is determining how fast to run the simpler processor model to properly mimic the stream of references from a more complex processor. If a simple mechanism could be found that allowed unilateral use of this technique or other ways to simplify the modeling of all other parts of the system it would be an important advance in memory system evaluation.

3. Voltage-Independent Energy/Delay Metric

In modern VLSI systems, voltage can be used as a "knob" (different design styles and manufacturing techniques result in different operating ranges for the supply voltage, some can be quite large [10]). Increasing the voltage increases the energy but decreases the time per computation. Decreasing the voltage results in lower energy implementations at the cost of performance. Attempts to simultaneously optimize both energy and performance began with the introduction of the *power-delay product* [2] as the metric to minimize. However, since dissipated power is energy per unit time, and delay is simply t, power-delay is nothing more than a measure of energy. Clearly

minimizing energy can be done by making performance go to zero, and hence power-delay product is a poor metric. *Energy-delay product* (Et) is a better metric [8], but since $E \propto V^2$ and $t \propto 1/V$, Et can be minimized by simply turning down the voltage at the expense of performance.

A better metric is Et^2, which to first order is independent of voltage. A chip that "beats" another chip on Et^2 will be higher performance than the other chip at the same energy, or use less energy at the same performance level [9]. Et^2 can be used as the metric for the overall system or for individual components of the system—for example, what pipeline depth is optimal using Et^2 versus t [12], how big should the instruction window be?, etc. The Et^2 metric allows the comparison of designs at different voltages and performance levels where each claims to be more "power-efficient". Including Et^2 in architectural simulators may give new insights into optimal low-energy designs.

References

[1] M. C. Carlisle and A. Rogers. Software Caching and Computation Migration in Olden. In *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, July 1995, pages 29–38.

[2] A. P. Chadrakasan, S. Sheng, and R. W. Broderson. Low Power CMOS Digital Design. *Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473–484, April 1992.

[3] M. Chaudhuri, M. Heinrich, C. Holt et al. Latency, Occupancy, and Bandwidth in DSM Multiprocessors: A Performance Evaluation. Technical Report CSL-TR-2001-1018, Computer Systems Laboratory, Cornell University, November 2001.

[4] Z. Cvetanovi and D. D. Donaldson. AlphaServer 4100 Performance Characterization. *Digital Technical Journal*, vol. 8, no. 4, 1996.

[5] DIS Benchmark Suite. http://www.aaec.com/projectweb/dis/

[6] J. Gibson et al. FLASH vs. (Simulated) FLASH: Closing the Simulation Loop. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 49–58, November 2000.

[7] M. Heinrich, R. Soundararajan, J. Hennessy et al. A Quantitative Analysis of Distributed Shared Memory Cache Coherence Protocols. *IEEE Transactions on Computeres*, **48**(2):205–217, February 1999.

[8] M. Horowitz, T. Indermaur, and R. Gonzalez. Low-Power Digital Design. In *Proceedings of the International Symposium on Low Power Electronics*, 1994.

[9] R. Manohar and M. Nystrom. Implications of Voltage Scaling in Asynchronous Architectures. Cornell Computer Systems Lab Technical Report CSL-TR-2001-1013, April 2001.

[10] A. J. Martin, A. Lines, R. Manohar et al. The Design of an Asynchronous MIPS R3000. In *Proceedings of the 17th Conference on Advanced Research in VLSI*, pages 164–181, September 1997.

[11] L. McVoy and C. Staelin. lmbench: Portable tools for performance analysis. *USENIX technical conference*, pages 279–284, January 1996.

[12] J. Teifel, D. Fang, D. Biermann et al. Energy Efficient Pipelines. To appear in the *Proceedings of the 8th International Symposium on Asynchronous Circuits and Systems*, March 2002.

[13] A. Saulsbury, F. Pong, and A. Nowatzyk. Missing the Memory Wall: The Case for Processor/Memory Integration. In *Proceedings of the 23rd International Symposium on Computer Architecture*, pages 90–101, May 1996.

[14] S. C. Woo et al. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 24–36, Santa Margherita Liguere, Italy, June 1995.