Challenges in Computer Architecture Evaluation

Kevin Skadron
University of Virginia
Margaret Martonosi
David I. August
Princeton University
Mark D. Hill
University of Wisconsin–Madison
David J. Lilja
University of Minnesota
Vijay S. Pai
Rice University

A report to the US National Science Foundation argues that simulation and benchmarking technology will require a leap in capability within the next few years to maintain ongoing innovation in computer systems.

Quantitative evaluation has become the mainstay of computer architecture research. However, the tremendous complexity of computer systems is making them both difficult to reason about and expensive to develop. Detailed software simulations have therefore become essential for evaluating ideas in the computer architecture field. Industry uses simulation extensively during processor and system design because it is the easiest and least expensive way to explore design options. Simulation is even more important in research to evaluate radical new ideas and characterize the nature of the design space.

Table 1 shows a dramatic shift toward simulation-based research reflected in papers presented at the International Symposium on Computer Architecture. Papers on simulations now constitute 80 to 90 percent of the total at this premier conference—up from only 28 percent in 1985. By comparison, papers based on direct measurements of real systems or on mathematical models have fallen to less than 10 percent from almost 35 percent in 1985.

Unfortunately, it is becoming harder and more time-consuming to construct accurate simulation models of modern computer systems. Further, the substantial effort required to develop high-fidelity simulation tools typically yields few academic rewards. Finally, as the range of important applications becomes more diverse, creating a suitable set of publicly available benchmarks and metrics becomes more challenging. Together, these difficulties are likely to encourage researchers to focus on problems suited to the current evaluation infrastructure—a type of research that only "looks where the light is good."

Research on multiprocessor systems, in particular, has already hit a wall in performance evaluation. As Figure 1 shows, the proportion of multiprocessor papers submitted to ISCA dropped from a peak of 50 percent in 1985 to less than 10 percent in 2001. This trend is echoed in other forums and seems due in part to the difficulty of performing in-depth research evaluations related to these systems.

This research downturn comes at a crucial time. Over the next 10 years, even single-chip computers will likely be multiprocessor systems. Even in uniprocessor systems, increased on-chip heterogeneity and specialization as well as new metrics such as power and temperature will make simulation and modeling difficult, if not impossible, using current tools and methodologies.

Published by the IEEE Computer Society
0018-9162/03/$17.00 © 2003 IEEE

In December 2001, a number of researchers met under the aegis of the US National Science Foundation's Computer Systems Architecture program to discuss the experimental and evaluation problems that processor architecture research faces.

The 18 workshop panelists agreed that current limitations and trends in evaluation techniques are troublesome and could noticeably slow the rate of computer systems innovation. They recommended new research to help make quantitative evaluations of computer systems manageable again (www.ee.princeton.edu/~mrm/CPUperf.html).

Specifically, the panel's report advocated support for research in the areas of simulation frameworks, benchmarking methodologies, analytic methods, and validation techniques.

SIMULATION FRAMEWORKS

Quantitative evaluation of computer architectures relies heavily on simulators and simulator infrastructure, yet today's robust and publicly available simulation tools are by no means capable of supporting the full range of studies that the architecture community must pursue.

Problems

Current simulation infrastructures are written in ways that pose several problems. First, sequential C or C++ simulator code does not resemble the systems under study. This discrepancy forces simulator writers to perform a complex mapping from the concurrent, structural nature of the systems to the sequential, procedural nature of the simulator's underlying programming language. The mapping process is typically ad hoc and error-prone. As a result, simulator code obscures the machine actually being modeled, which can compromise the validity of conclusions drawn from the simulation results.

Further, the mapping schemes used between simulators—and even within a single simulator—are typically distinct. These differences limit the interoperability of simulator code, which in turn limits code reuse and the collaboration potential of current simulation methodologies.

Given the difficulty of reusing the code in current tools, computer architecture research tends to focus on questions relating to machines for which simulation models exist. This restricted focus means that parts of the design space as well as potentially interesting new ideas are inadequately explored because the available tools do not support them.

The challenge is even greater for multiprocessor simulators than for uniprocessor simulators.

Table 1. Performance evaluation methodologies used in a sampling of papers from the *Proceedings of the International Symposium on Computer Architecture*.

Year	Total papers*	Simulation	Measurement	Mathematical modeling	Other
2001	25	22	2	0	2
1997	30	24	6	0	0
1993	32	23	9	6	1
1985	43	12	1	14	16
1973	28	2	0	5	21

*Note: Total papers are not necessarily the sum of papers across the columns because some papers used more than one evaluation technique.

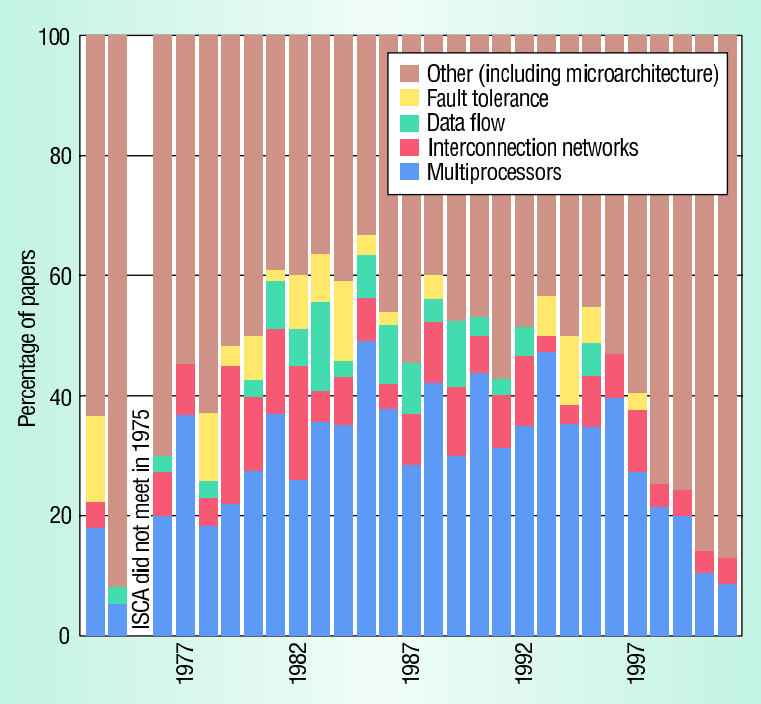

Figure 1. Relative percentage of papers by topic category submitted to the International Symposium on Computer Architecture from 1973 to 2001. Multiprocessor papers have declined especially rapidly since 1993. Source: M.D. Hill and R. Rajwar, "The Rise and Fall of Multiprocessor Papers in the International Symposium on Computer Architecture (ISCA)," 2001; www.cs.wisc.edu/~markhill/mp2001.html. Reprinted with permission.

Multiple processors and the system structures that connect them already stress the capabilities of current modeling tools. As system complexity continues to skyrocket, these tool-related problems will only worsen.

Simulation recommendations

All workshop participants agreed that simulation frameworks and simulator construction frameworks were superior to monolithic simulators or simulator code libraries written in sequential languages.

Simulation frameworks provide an infrastructure for pluggable components and hierarchical

Significant research is needed to develop reliable abstractions for multiprocessor simulators and to improve their speed, accuracy, and modularity.

abstraction by defining interfaces that resemble concurrent communication in real systems—not the function-call interface found in sequential programming languages such as C. Simulation frameworks encourage sharing of component models across research groups by nullifying conventions specific to any given monolithic simulator or code library.

Simulation construction frameworks enable rapid exploration of design alternatives by automatically weaving architectural component models together as implied by a machine description. Abstracting the machine description from the simulator code gives researchers a clear picture of the machine actually being modeled.

Modularity and portability. Framework modularity lets researchers consider different evaluation levels. For instance, they can use abstract evaluations with analytic models to study ideas in the earliest stages of development and increase the detail in evaluations as the work progresses. Frameworks also encourage code reuse and expose a machine's structure through its description, thereby speeding validation of machine models. Examples of recently developed frameworks include Asim[1] and the Liberty Simulation Environment.[2]

Support for constructing simulator frameworks will have important long-term benefits by making it easier to extend, reuse, and validate simulator components. Relatively unrestricted machine description languages encourage consideration of unconventional machines, and multilevel simulations let researchers substitute analytic or real-time-logic models for specific components.

The NSF workshop panelists accordingly recommended that the computer architecture community encourage research to develop modular and portable frameworks. Government and industry sources should fund efforts in two phases:

- develop new frameworks to foster maximal innovation, and
- advance the most successful frameworks for wide distribution in the R&D community.

Model accuracy. The community must also arrive at some consensus on what constitutes appropriate validation for various types of research studies. Models of even an existing architecture are unlikely to capture every nuance of actual hardware behavior, and for many studies such detail simply slows research unnecessarily.

A key question is whether absolute or relative accuracy is required—that is, are exact performance projections necessary or is it sufficient to produce only a reliable projection of how different parts of the design space perform relative to each other? The answer may dictate different validation strategies.[3]

Multiprocessor infrastructure is a particular concern. Significant research is needed to develop reliable abstractions for multiprocessor simulators and to improve their speed, accuracy, and modularity. Both shared-memory and message-passing systems need models.

Other challenges lie in simulating the heterogeneity inherent in many upcoming embedded systems while preserving portability of common components between general-purpose and embedded-system processor simulations.

Funding and disseminating research. Funding agencies can play an important role in advancing simulation technology by aggressively supporting research programs to develop new frameworks and components. The academic community can in turn support this important research area by creating a forum for disseminating these contributions and recognizing them in peer-reviewed publications.

BENCHMARKING

While benchmarking issues are diverse, a few problems seem particularly acute for architecture research.

Problems

First, current benchmarks focus almost exclusively on high-performance/desktop workloads, represented by the SPECcpu benchmarks, and on scientific workloads, such as the SPLASH benchmarks. No benchmark suite can be a one-size-fits-all solution. While the SPEC and SPLASH application classes remain important, other classes are growing too rapidly to ignore. Examples include embedded systems (for which the University of Michigan recently released the MiBench suite[4]), mobile computing (for which the University of California, Los Angeles, has released the MediaBench suite[5]), real-time computing, server workloads, and networking workloads. Behavior types such as dependability, power, and pointer usage also need benchmarks.

Second, all benchmark suites, including SPEC and SPLASH, require better characterization to show what portion of the total "behavior space" they really represent. Unfortunately, it is unclear whether the current SPECcpu suite adequately represents today's high-performance/desktop applica-

tions, let alone other application classes such as mobile computing. Similarly, nonscientific workloads have become at least as important as scientific workloads in multiprocessor systems, especially in the small-scale systems used by servers. Yet the R&D community does not have a system for identifying important benchmark characteristics and how benchmark applications embody them. Such a system would also help in identifying characteristics that a suite does not represent well and thereby guide the development of new benchmarks.

Third, full-size ("macro") benchmarks can combine many behaviors in ways that can be hard to conceptualize. *Microbenchmarks* offer a way to isolate individual program behaviors or individual aspects of a processor's performance. In addition, researchers can combine them to construct more thorough application and system performance models and to determine the importance of various system behaviors.[6,7] Yet microbenchmarks are almost completely lacking at this time.

Fourth, the batch-processing mode of running benchmarks—as if each one had exclusive access to the CPU for the duration—is unrealistic. Many systems run multiple processes simultaneously, often sharing the hardware at a fine granularity. The community needs a sound methodology for modeling workloads in addition to individual benchmarks. The methodology must include operating system effects, especially in multiprocessor applications. Moreover, we need more work to understand complex workloads, such as database management systems, whose exact execution path is a function of many system parameters.[8]

Benchmarking recommendations

The NSF workshop panelists recommended aggressive support for programs to develop and characterize robust, portable benchmarks for application domains outside the traditional high-performance/desktop domain. In addition, research is needed to develop techniques for characterizing and abstracting benchmarks and for providing parameterized, synthetic workloads.

Reward structure. Developing benchmarks, benchmarking methodologies, and tools is hard work. While developers of widely accepted benchmark suites and methodologies see significant payoff for their work in citations and other recognition, the community's tendency to support only a small set of benchmark suites leads to a winner-takes-all situation. Consequently, the current reward structure makes research in this area too risky for most academics.

The academic architecture community must provide a forum for evaluating and disseminating benchmarks. Such a forum, perhaps organized as a standing committee, could encourage benchmark creation in areas of increasing importance as well as vet benchmark quality and characterization. Hopefully, both committee service and benchmark submission will, in time, be seen as prestigious. The committee might take on some characteristics of a conference program committee, and a benchmark's approval might include an accompanying publication that will have the stature of a more traditional peer-reviewed paper.

Nonscientific workloads have become at least as important as scientific workloads in multiprocessor systems.

Open source. The panelists envision a benchmark suite for *research* purposes and so recommend that all benchmarks be publicly available with source code. Users should be permitted to improve the algorithms in benchmarks with the proviso that they make such changes publicly available as well. Some means for incorporating improvements back into the standardized form of the benchmarks is also important.

Synthetic benchmarks. Finally, the panelists recommend synthetic benchmarks that are coded to run on a real computer but parameterized to provide a range of different behaviors. Synthetic benchmarks would let researchers explore a wider range of the application behavior space, even when no publicly available benchmark exists. Parameterization would allow a single benchmark to produce a variety of behaviors, covering a larger portion of the behavior space. With a suitable choice of parameters, the synthetic benchmarks should be able to demonstrate behaviors similar to those demonstrated by "real" benchmarks. This concept could be extended to create parameterized *workloads* with a variable mix of program behaviors and rates.

ABSTRACTIONS AND METHODOLOGY

Computer architecture's heavy emphasis on simulation effectively discourages the research community from exploring other useful and possibly more informative modeling techniques. The few published papers using and proposing analytic models have not stimulated significant follow-up efforts. Stories abound of such papers receiving knee-jerk negative rejections from program committees and other researchers. Among the NSF workshop participants, however, even skeptics admit that analytic models have a place and that some aspects of the research community's hostility are cause for concern.

Researchers cannot pursue futuristic investigations when they are limited to systems for which no benchmark programs are available.

Analytic models and simulation are not mutually exclusive. Analytic models can help to understand a system in ways that simulation does not. They can also be used to validate a simulation-based model. Some panelists described experiences in which such validation exercises proved extremely valuable.

Problems

The research community's preference for papers that emphasize measurements with detailed simulation has generated valuable results, but it has also undermined work on more far-reaching approaches that cannot yet be adequately simulated. In general, we should be careful to value quantitative results for the understanding they provide, applying Richard Hamming's dictate to computer systems evaluation: "The purpose of computing is insight, not numbers."

Further, some aspects of very detailed evaluation can be unrealistic and wasteful when exploring a technology that is sufficiently speculative to lack detailed behavior and timing data. More importantly, researchers cannot pursue futuristic investigations when they are limited only to systems that can already be simulated and for which no benchmark programs are available. For instance, analytical performance-evaluation tools can model the expected behavior of future systems; they can also model the expected impact of future compiler and hardware modifications, thereby avoiding unnecessary costs associated with more detailed simulation models or actual implementations.

In domains currently without good benchmarks, new abstract workloads could help if we can justify the abstractions and characterize their representativeness. Further, as system and workload complexity increases, detailed simulation studies will take far too much time. This constraint already limits multiprocessor simulations. Increasing heterogeneity in uniprocessor systems-on-a-chip foretells a similar problem. We need scientific methods for abstracting evaluations to explore the desired design space accurately but efficiently.

Recommendations for analytic models

Workshop panelists agreed that developing scientific methods for abstracting evaluations to explore large design spaces is imperative. Analytic models, both as modules within an overall evaluation framework and as a way of validating simulator behavior, are important tools that the research community is mostly lacking today.

Statistical or other techniques that clearly demonstrate how analytic models capture important behaviors would substantially improve the way most practitioners view such models, especially if the models are easy to use and understand. Some techniques, such as performance counters and related tools, are driven by direct measurements yet offer many of the same benefits as analytic models.

METRICS, EVALUATION ACCURACY, AND VALIDATION

Quantitative evaluations must give accurate insights about trends and behavior. Model inaccuracies can lead to incorrect predictions and even spurious research threads that take years to resolve.

Yet many studies need not predict exact values for performance, power, and other metrics. Rather, they need only provide a reliable projection of how different parts of the design space perform *relative* to one another. Such results are especially important for exploring hypothetical architectures and targeted future technologies in which the lack of detailed design information makes absolute accuracy impossible.

Problems

Some modeling assumptions are essential for achieving relative accuracy, while others add needless complexity. The current understanding of correct abstraction levels and other important aspects of accurate models is poor. This leads to wasted effort on models and simulations that contain unnecessary detail while simultaneously lacking certain essential information. For hypothetical systems, high precision—no matter how detailed the model—can be wasted if the assumptions that underlie the detail are inappropriate or change over time. Early-stage studies should focus on characterizations of broad parameter spaces.

Another problem with simulation evaluation accuracy and validation is that most architecture research uses the same basic, aggregated statistics: average instructions per second (IPC), cache miss rate, branch misprediction rate, and so on. However, average values conceal bursty behavior and can therefore be misleading by aggregating underestimates and overestimates over time. Unfortunately, simple standard deviations are not helpful because the events being measured seldom fit a Gaussian distribution. We need a wider range of methods and metrics for analyzing processor performance as well as a better understanding of how to use them appropriately.

In fact, the computer architecture community continually searches for new metrics to provide better insight and simplify evaluation. Thermal packaging is a recent case in point: How do we abstract away specific thermal packaging details to obtain a generalized metric for heat regulation, similar to the way IPC abstracts away circuit-design and clock-cycle-time details? The difficulty of evaluating and generalizing new ideas in temperature-aware computing is a serious obstacle to effective research in this area, and similar difficulties appear in many areas of architecture research.

More generally, how do we effectively capture tradeoffs between the continuing need for performance and other design needs? How do we reflect real-time and reliability requirements? How do we reflect soft real-time application requirements, with a range of acceptable quality-of-service levels, so that architects can understand how to trade quality of service against other design goals?

Recommendations for validation techniques

The community must improve its understanding of an accurate model's essential components. This understanding underlies the development of techniques for defining less-detailed simulations that still provide relative accuracy. It also supports the development of methods to verify that accuracy.

Computer architects need better metrics as well as statistical techniques and tools that are accessible and easy to use.[9] They also need metrics for new areas, including power, temperature, reliability, and quality of service. Even existing metrics, such as the energy-delay product now widely used for power-aware computing, need expansion to encompass real-time computing and other design goals.

The field is rife with different simulation techniques. There is little agreement on when to use certain benchmarks or inputs or, despite recent work,[10-12] on what configurations to model for various types of experiments and what areas require the greatest investment in modeling detail. Sound and verifiable modeling methodologies require further research.

Unlike most other scientific research disciplines, published computer architecture results are rarely independently verified. This lack of independent experiment replication appears to result primarily from the lack of an appropriate reward structure. Workshops such as the successful 2002 and 2003 Workshops on Duplicating, Deconstructing, and Debunking (www.ece.wisc.edu/~wddd/) are a step in the right direction, but a publication with the imprimatur of a refereed journal is needed to completely legitimize the replication and verification of results. Existing journals might offer a distinct track for publishing results of this nature.

Without major advances in the areas discussed here, limitations in the current evaluation infrastructure will likely restrict computer architecture research to a narrow, incremental, and ultimately irrelevant enterprise. The research community must play its part by embracing high-quality work in these areas. However, because the work involves a large investment of time and effort, the NSF workshop panel argued that a substantial investment of government and industry research funds is required to jump-start it.

New venues for publishing and disseminating work are also necessary. Without funding and promising prospects for academic recognition, research and development in these areas is likely to languish at its current slow pace. The current shortcomings in computer systems evaluation could ultimately even obstruct the innovation that is driving the information-technology revolution. ■

References

1. J. Emer et al., "Asim: A Performance Model Framework," *Computer*, Feb. 2002, pp. 68-76.
2. M. Vachharajani et al., "Microarchitectural Exploration with Liberty," *Proc. 35th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, IEEE CS Press, 2002, pp. 271-282.
3. R. Desikan, D.C. Burger, and S.W. Keckler, "Measuring Experimental Error in Microprocessor Simulation," *Proc. 28th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, 2000, pp. 266-277.
4. M.R. Guthaus et al., "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," *Proc. 4th Ann. IEEE Int'l Workshop Workload Characterization*, IEEE CS Press, 2001, pp. 3-14.
5. C. Lee et al., "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," *Proc. 30th Ann. Int'l Symp. Microarchitecture*, IEEE CS Press, 1997, pp. 330-335.
6. R.H. Saavedra and A.J. Smith, "Analysis of Benchmark Characteristics and Benchmark Performance Prediction," *Trans. Computer Systems*, Nov. 1996, pp. 344-384.
7. M. Seltzer et al., "The Case for Application-Specific Benchmarking," *Proc. 7th Workshop Hot Topics in Operating Systems*, IEEE CS Press, 1999, pp. 102-107.

8. A. Alameldeen et al., "Simulating a $2M Commercial Server on a $2K PC," *Computer*, Feb. 2003, pp. 50-57.
9. J.J. Yi, D.J. Lilja, and D.M. Hawkins, "A Statistically Rigorous Approach for Improving Simulation Methodology," *Proc. 9th Int'l Symp. High-Performance Computer Architecture*, IEEE CS Press, 2003, pp. 281-291.
10. T. Sherwood et al., "Automatically Characterizing Large Scale Program Behavior," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, ACM Press, 2002, pp. 45-57.
11. J.W. Haskins Jr. and K. Skadron, "Memory Reference Reuse Latency: Accelerating Sampled Microarchitecture Simulation," *Proc. 2003 IEEE Int'l Symp. Performance Analysis of Software and Systems*, IEEE Press, 2003, pp. 195-203.
12. T. Conte, M.A. Hirsch, and K.N. Menezes, "Reducing State Loss for Effective Trace Sampling of Superscalar Processors," *Proc. 1996 Int'l Conf. Computer Design*, IEEE CS Press, 1996, pp. 468-477.

Kevin Skadron *is an assistant professor in the Department of Computer Science at the University of Virginia. His research interests are in computer architecture and performance analysis, especially temperature- and power-aware computing, applications of feedback control, and branch prediction. Skadron received a PhD in computer science from Princeton University. He is a member of the IEEE and the ACM. Contact him at skadron@cs.virginia.edu.*

Margaret Martonosi *is an associate professor in the Department of Electrical Engineering at Princeton University. Her research interests are in computer architecture and the hardware/software interface, particularly power-efficient microarchitectures and power-adaptive mobile systems. Martonosi received a PhD in electrical engineering from Stanford University. She is a senior member of the IEEE and a member of the ACM. Contact her at martonosi@princeton.edu.*

David I. August *is an assistant professor in the Department of Computer Science at Princeton University, where he also directs the Liberty Research Group to develop open source tools for systematic processor design-space exploration (http://liberty.princeton.edu). His research interests are in computer architecture and back-end compilation. August received a PhD in electrical and computer engineering from the University of Illinois at Urbana-Champaign. He is a member of the IEEE and the ACM. Contact him at august@cs.princeton.edu.*

Mark D. Hill *is professor and Romnes Fellow in both the Computer Sciences Department and the Electrical and Computer Engineering Department at the University of Wisconsin–Madison. With David Wood, he also codirects the Wisconsin Multifacet project to improve commercial servers. His research interests include cache design, cache simulation, translation buffers, memory consistency models, parallel simulation, and parallel computer design. Hill received a PhD in computer science from the University of California, Berkeley. He is an IEEE Fellow. Contact him at markhill@cs.wisc.edu or www.cs.wisc.edu/~markhill.*

David J. Lilja *is a professor of electrical and computer engineering at the University of Minnesota in Minneapolis. His research interests are in computer architecture, parallel processing, nanocomputing, and computer systems performance analysis. Lilja received a PhD in electrical engineering from the University of Illinois at Urbana-Champaign. He is a senior member of the IEEE and a member of the ACM. Contact him at lilja@umn.edu.*

Vijay S. Pai *is an assistant professor in the Department of Electrical and Computer Engineering at Rice University. His research interests include computer architecture, high-performance networking, and performance evaluation. Pai received a PhD in electrical and computer engineering from Rice University. Contact him at vijaypai@rice.edu.*

Computer Wants You

Computer **is always looking for interesting editorial content. In addition to our theme articles, we have other feature sections such as Perspectives, Computing Practices, and Research Features as well as numerous columns to which you can contribute. Check out our author guidelines at**

http://computer.org/computer/author.htm

for more information about how to contribute to your magazine.

Innovative Technology for Computer Professionals
Computer