NSF Computer Performance Evaluation Workshop: Summary and Action Items

December, 2001

On December 1-5, 2001, eighteen researchers from the computer architecture field convened in Austin, TX to discuss the experimental and evaluation problems that research in processor architecture faces today. Overall, the panel concluded that the development of new benchmarks, modular simulation frameworks, and new analytic models are essential to maintain the rate of innovation that the computer architecture community requires. NSF funding for these research efforts is vital.

Margaret Martonosi, Princeton University (co-organizer)
Kevin Skadron, University of Virginia (co-organizer)

http://www.ee.princeton.edu/~mrm/CPUPerf.html

Attendees

Sarita Adve, University of Illinois
David August, Princeton University
Todd Austin, University of Michigan
Doug Burger, University of Texas
Tom Conte, North Carolina State University
Joel Emer, Intel Corporation
Antonio Gonzalez, UPC Barcelona
Mark Heinrich, Cornell University
Mark Hill, University of Wisconsin
Lizy John, University of Texas
David Lilja, University of Minnesota
Mark Oskin, University of Washington
Vijay Pai, Rice University
Sanjay Patel, University of Illinois
Paul Reynolds, University of Virginia
Guri Sohi, University of Wisconsin

Karthik Sankaranarayanan, University of Virginia (assistant)
Sivakumar Velusamy, University of Virginia (assistant)

1 Introduction and Overview

Computing research based on quantitative evaluation has been the mainstay of our field in recent years. Because the systems we build are so complex, they are difficult to reason about, so detailed simulations have become essential for evaluating and publishing ideas in our field. For example, simulation-based research now comprises 80-90% of papers appearing in the International Symposium on Computer Architecture (ISCA), while papers based on direct measurements of real systems or on mathematical modeling have fallen to less than 10%.

Unfortunately, quantitative evaluation in our field is facing difficult times in the years ahead. As systems grow more complex, interacting assumptions make validation more important. Otherwise, unexpected and undetected interactions may lead the field astray by improperly gauging the impact of different design choices. And as systems grow more complex, developing adequate simulation infrastructure becomes more difficult and time-consuming, with little academic reward in many cases. This is likely to increase the already evident trend that many researchers focus on problems that can be easily evaluated using current infrastructure—"looking where the light is good." Simulation and benchmarking technology will require a leap in capability within the next few years in order to remain useful.

Research on multiprocessor (MP) systems, in particular, has already hit a performance-evaluation cliff. The recent downturn in MP papers in premiere computer-architecture forums—from a peak of 50% of the ISCA program in 1985 to less than 10% in 2001, for example[1]—seems clearly to be due in part to the difficulty of evaluating MP systems. Unfortunately, this research downturn is happening at a crucial time in MP research: over the next ten years, it is likely that even single-chip computers will be multiprocessor systems, yet insufficient knowledge is available to shape this evolution. Even in uniprocessor systems, trends towards increasing on-chip heterogeneity and specialization and the importance of new metrics like power and temperature will lead us to soon be building systems that are quite difficult to simulate or model.

Research is needed to understand the abstractions and evaluation methodologies that can make quantitative evaluations of these systems manageable again. On the other hand, if we continue on the current course, without more attention and funding for evaluation methodologies, computer systems research is likely to become increasingly incremental or irrelevant due to the limitations of the evaluation infrastructure. The goals of our discussions were to articulate this problem in detail and to brainstorm on possible solutions.

This report summarizes the findings of the workshop, with a focus on funding recommendations that we encourage NSF to support. The report focuses on four major areas: benchmarking, simulation infrastructure, abstractions and metrics, and accuracy and validation.

2 Benchmarking: Problems and Planned Solutions

There was almost unanimous agreement that large, real-world applications are necessary for meaningful evaluation, and hence the advent of the SPEC and SPLASH benchmarks has been beneficial. There was also almost unanimous agreement that the community must now move beyond SPEC and SPLASH in several ways.

Problems

First, the community needs to stop its almost exclusive focus on the high-performance/desktop workloads that SPECcpu benchmarks represent and the scientific workloads that SPLASH benchmarks represent, and develop additional benchmark suites that represent other areas of computing. No benchmark suite can be a one-size-fits-all solution, and while the SPEC and SPLASH classes of applications remain important, other classes are rapidly becoming too important to ignore. Examples include embedded systems (for which

[1] M.D. Hill and R. Rajwar, "The Rise and Fall of Multiprocessor Papers in the International Symposium on Computer Architecture (ISCA)." http://www.cs.wisc.edu/~markhill/mp2001.html.

Univ. of Michigan recently released a benchmark suite, MiBench[2]), mobile computing, real-time computing, server workloads, networking workloads, and so forth. In addition, benchmark suites for various types of *behavior* might also be useful. Examples include dependability, power, and pointer usage.

Second, all benchmark suites, including SPEC and SPLASH, need to be better characterized to understand what portion of the total "behavior space" these suites really represent. We cannot currently be sure whether SPEC is adequately representative of the range of high-performance/desktop applications it supposedly represents, or whether SPEC might actually be representative of wider classes of applications, like mobile computing. Similarly, many in the MP-research community believe that non-scientific workloads have become at least as important as scientific workloads; this is especially true of small-scale multiprocessor systems that might be found in servers. The community needs a way to identify important characteristics of any benchmark and how these characteristics are embodied in each benchmark application. A characterization system like this also helps identify what types of characteristics are not well represented in a suite and hence guides the development of new benchmarks. A suite of parameterized benchmarks might even be possible, in which a single benchmark can be used to produce a variety of behaviors and cover a broader portion of the behavior space. This concept can also be extended to create parameterized *workloads*, in which the mix of program behaviors and rates can be varied.

Third, the appropriate use of "micro-benchmarks" was also discussed. Full-size ("macro") benchmarks usually confound many behaviors in ways that can be hard to reason about. There was widespread agreement that micro-benchmarks, which isolate individual program behaviors or the performance of individual aspects of a processor design, are important tools. Micro-benchmarks help researchers and designers in identifying problems, applying stress tests, and understanding the interaction of multiple behaviors and design parameters. Micro-benchmarks are not themselves sufficient to evaluate an idea, but their usefulness as diagnostic, investigatory, and explanatory tools makes them useful, and development of micro-benchmark suites should be supported.

Fourth, the batch-processing mode of running benchmarks (in which benchmarks are run independently as if they had exclusive access to the CPU for the duration) is not realistic. In many systems, multiple processes are running simultaneously, often sharing the hardware in a fine-grained way, and the community needs a sound methodology for modeling *workloads* in addition to individual benchmarks. An important component of this is the inclusion of operating-system (OS) effects, which is especially important in multiprocessor applications.

Fifth, many benchmarks are so large that they simply take too long to simulate. Simulation time remains a major bottleneck for architecture research and performance analysis. Much work has been done on reducing input sets and sampling, but more work is needed to reach convergence on the best and most rigorous techniques. And an important new direction is to find ways to shorten and/or abstract benchmarks without sacrificing fidelity.

Recommendations

Developing benchmarks and benchmarking methodologies and tools is hard work. While developers of widely-accepted benchmark suites and methods see significant impact and payoff for their work, the rewards have a "winner takes all" flavor to them. NSF can play an important role by aggressively supporting research programs to develop and/or characterize robust and portable benchmarks for application domains outside the traditional high-performance/desktop domain, as well as techniques for characterizing and abstracting benchmarks, reducing simulation times, and providing parameterized, synthetic workloads.

- The academic community should provide a forum for evaluating and disseminating benchmarks in the form of a benchmark committee. Important goals of this committee are to encourage benchmarks creation in areas of increasing importance, and to vet the quality of benchmarks and their

[2] M.R. Guthaus *et al*. "MiBench: A Free, Commercially Representative Embedded Benchmark Suite." In the Fourth IEEE Workshop on Workload Characterization, Dec. 2001.

characterization. Committee service and benchmark submission should both, in time, be seen as prestigious. The committee might take on some characteristics of a conference program committee, and approval of a benchmark might include an accompanying publication that can be regarded as a peer-reviewed publication. Initial funding support from NSF is likely to be necessary to help this forum become self-sustaining.

- All benchmarks should be made available with source code. Users should be permitted to improve the algorithms in these benchmarks with the proviso that improvements must be made publicly available. Some means for incorporating improvements back into the benchmarks is important.
- Synthetic benchmarks that are coded to run on a real computer, but parameterized to provide a range of different behaviors, would allow researchers to explore a wider range of the application behavior space even when publicly available benchmarks are not available. With a suitable choice of parameters, the synthetic benchmark should be able to demonstrate behavior similar to that of existing benchmarks.
- Programs are very repetitive. Some technique for extracting the main features, or generating a summary model or synthetic benchmark that is smaller and more efficient but provides representative behavior, would allow shorter simulation time while preserving representative behavior. It would be especially helpful if machine-learning techniques could do this even when we do not know the characteristics important to us. For example, genetic programming from a trace of a real benchmark to construct a benchmark with characteristics that match, but has a shorter runtime, might be one approach. We can also look to developing new techniques for detecting patterns, *e.g.*, based on Fourier analysis or factorial analysis.

3 Simulation Infrastructure: Problems and Planned Solutions

Quantitative evaluation of computer architectures relies heavily on simulators and simulator infrastructure, yet the simulation tools that are robust and publicly available today are by no means able to support the entire range of simulation studies that the architecture community needs to pursue.

Problems

Current simulation infrastructure is written in ways that limit code sharing and limit the ability of different research groups to validate each others' results. Current tools tend to dictate the majority of research that gets done, because it is easier to study questions that can be answered with existing tools. This means that there are parts of the design space that end up only lightly explored, because of the difficulty of doing so. As system complexity skyrockets, simulation will become even more difficult. Now is the time to invest in evaluation techniques so that we will be better prepared as problems deepen over the next few years.

Recommendations

One of the key consensus points of the workshop is that *simulation frameworks* and *simulator construction frameworks* (as opposed to monolithic simulators or simulator code libraries) would be extremely valuable to the community, and their future development should be encouraged. A *simulation framework* provides an infrastructure for pluggable components and hierarchical abstraction by defining interfaces resembling 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.

A *simulator construction framework* is a simulation framework that enables rapid exploration of design alternatives by automatically weaving architectural component models together as implied by a *machine description*. By abstracting the machine description from the simulator code, the simulator construction framework presents the researcher with a clear picture of the machine actually being modeled.

Support for simulation and simulator construction framework implementation will pay off long-term as simulator components will be more easily extended, reused, and validated. Relatively unrestricted machine description languages will encourage consideration of unconventional machines. Frameworks will also encourage multi-level simulations by allowing analytic or RTL model substitutions for specific

components. Recently developed frameworks discussed were Asim[3] and Liberty[4]. Both would require additional work before widespread release.

- We should encourage research to develop modular and portable frameworks. Not only do they further research, they also eliminate duplication of efforts by fostering module sharing between many researchers. NSF should fund efforts in two stages: First, the development of various new simulator frameworks to foster maximal innovation in the community. Subsequently, the most successful frameworks should receive further funding to make them portable and bulletproof and widely distributed.
- The modularity offered by framework approaches allows one to consider different levels of evaluations: more abstract evaluations like analytic models for immature ideas, followed by increasingly detailed evaluations as work progresses.
- Ultimately, the purpose of our research is to generate results that are sufficiently well-validated to be compelling to the research community as well as those in industry considering further implementations of the idea. Careful validation speeds research progress of the community as a whole. Frameworks encourage code reuse and expose the machine description, thereby speeding validation of machine models.
- In some domains, industry tools may also be of use. While in other research domains commercial tools are widely used, this has not been the norm in the architecture community. NSF should consider more liberal funding for academic licensing of these tools when it eliminates duplicated effort. On the other hand, there are legitimate outstanding research issues that academics, not industry, need to address and industry tools should not be considered a substitute for further research on building good simulation infrastructure.
- Multiprocessor infrastructure is of particular concern. Cache coherence protocols, synchronization mechanisms, and data sharing in general are all inherently complex and thus add to simulator complexity. We need significant research on good abstractions for MP simulators, to improve their speed, accuracy, and modularity. Models are required for both shared-memory and message-passing systems.
- Other challenges arise in simulating the heterogeneity inherent in many upcoming embedded systems.

4 Abstractions, Methodology, and Metrics

The emphasis on simulation may be causing practitioners to overlook other useful and possibly more informative modeling techniques. The emphasis on simulation also creates a disincentive to research alternative modeling techniques, possibly starving the community of important new ideas and tools. Although a few papers using/proposing such models have been successfully published, they are not always approachable enough to stimulate significant follow-on research. Stories abound of papers that use/propose analytic models which receive knee-jerk negative rejections. Even the panelists who are highly skeptical of analytic models tended to feel that they have their place, and expressed concern about some aspects of the research community's reaction to them. Analytic models and simulation are also not mutually exclusive. Analytic models can be used to understand a system in ways that simulation cannot, and they can be used to validate a simulation-based model. Some participants described their own experiences in which they found such validation exercises to be extremely valuable.

Problems

Over the past decades, the community has put increasing emphasis on papers that show measurements with detailed simulation. Such results are certainly valuable, but this emphasis also makes it difficult to do research on more far-reaching ideas that cannot yet be easily simulated. In general, the community should be careful to value numbers for the sake of the understanding they provide, rather than valuing "numbers for numbers' sake". Some aspects of evaluation precision can be unrealistic and wasteful when exploring a technology that is sufficiently speculative that detailed behavior and timing are not yet available. More

[3] J. Emer *et al.* "Asim: A Performance Model Framework." *IEEE Computer*, 35(2):68-76, Feb. 2002.
[4] `http://liberty.princeton.edu/`

importantly, there are a variety of futuristic investigations that cannot be pursued when research is limited to only systems that can be simulated and for which we have benchmark programs. Analytical performance-evaluation tools can model the epxected behavior of future systems before they exist. Such tools can also model the expected impact of future compiler and hardware modifications to see if those ideas have any merit, avoiding unnecessary costs associated with more detailed simulation models or actual implementations.

The development of new analytic models that can be *integrated* with simulations for modules that do not require detailed modeling (*e.g.*, via pluggable modules in a modular framework, see above) and/or used to *validate* simulators is especially valuable and should be supported.

New abstract workloads could be incredibly useful in domains where we do not yet have good benchmarks, if the abstractions can somehow be justified and the representativeness of these workloads can somehow be characterized. Furthermore, as system complexity increases, detailed simulation studies will take far too long. It is imperative to develop scientific methods for abstracting evaluations to explore the desired design space accurately but efficiently. Multiprocessor simulations have already hit this wall, and increasing heterogeneity in uniprocessor systems-on-chip will cause them to encounter this soon also.

Recommendations

A variety of attractive research directions were discussed:

- It is imperative to develop scientific methods for abstracting evaluations to explore large design spaces accurately but efficiently.
- Development of analytic models, both as modules within an overall simulation framework and as a way of validating simulator behavior, are important tools that the community is mostly lacking today.
- Statistical or other techniques showing that analytic models do in fact capture the important behavior that researchers desire would probably substantially influence the way such models are viewed by most practitioners, especially for models that are easy to use and understand.
- Techniques that are driven by direct measurements, performance counters, or the like offer many of the same benefits as analytic models, in terms of either providing a fast and simple way to model a particular component within an overall simulation framework, or an independent model that can be used for validation.

Panelists feel that ideas like these are important to pursue in order to widen the range of techniques, tools, and methods available to the architecture community.

5 Accuracy and Validation

The main goal of quantitative evaluations in research papers is to give insights about trends and behavior, not to give numbers just for numbers' sake. Model accuracy is nonetheless important, because inaccuracies can lead to incorrect conclusions about trends, and may even lead to spurious threads of research that can take years to clear up.

Problems

One problem is that some modeling assumptions are *essential* for achieving relative accuracy, while others add needless complexity. The correct level of abstraction and an understanding of the most important aspects of accurate models is poorly understood today. This leads to a great deal of wasted effort in developing models and running simulations that contain unnecessary detail—even though they may simultaneously lack certain essential details. And for hypothetical systems, high precision, no matter how detailed the model, can be wasted effort if the assumptions that underlie that detail change over time. Early-stage studies should focus on characterizations of broad parameter spaces.

Another problem is that the same basic, aggregated statistics (average IPC, cache miss rate, branch misprediction rate, etc.) are used in most architecture research. Yet averages conceal burstiness and can

therefore be misleading—they may get "the right answer for the wrong reason" by aggregating underestimates and overestimates over time. Unfortunately, simple standard deviations are not terribly helpful, because the events being measured often do not fit a normal distribution. The community needs to develop a wider range of methods and metrics for analyzing processor performance and the communal knowledge of how to appropriately use these metrics.

Recommendations

The community needs to develop better methodologies in a variety of areas:

- Techniques for less-detailed simulations that can still provide relative accuracy and ways to verify that accuracy.
- Better metrics and statistical techniques and tools that are accessible and easy to use for architects.
- The community is rife with different ways to perform simulation, with no agreement on when to use certain benchmarks or inputs, how to accurately sample their behavior, what configurations to model for various types of experiments, and what areas of a model require the greatest investment in modeling detail Research is needed to develop a sound and verifiable methodology.
- Published results are rarely independently verified in our community, because there is no reward structure. More independent verification is needed. Workshops such as the newly-announced "Workshop on Duplicating, Deconstructing, and Debunking" may be a step in the right direction, but a publication with the imprimatur of a refereed journal is also required.
- Occasional validations via Verilog/VHDL designs and their associated, lower-level circuit simulations can be extremely useful.

6 Conclusions

The panelists unanimously agreed that current trends in evaluation techniques for computer architecture are troublesome. The range of computer systems and interesting workloads is growing, as is their sophistication. New research is needed to understand the abstractions, modeling, and evaluation methodologies that can make quantitative evaluations of these systems manageable again. Without major advances in these areas, the limitations in current evaluation infrastructure will likely make computer systems research become increasingly incremental and irrelevant. NSF funding for these efforts—both research into new techniques as well as the development of new benchmarking and evaluation infrastructure—is essential. Research in these areas is sufficiently high risk that most academics are unlikely to devote sufficient effort to these problems without support in the form of research grants.

The panel's recommendations fall into four categories. The primary recommendations are listed below.

- **Benchmarks:** NSF support for research to characterize the nature of different applications and develop synthetic, parameterized benchmarks that "span" the application space; development of new benchmarks that represent interesting workloads for mobile, real-time, server, and multi-processor domains; research into techniques for abstracting benchmark behavior and reducing simulation times.

- **Simulation Frameworks:** NSF support for development of new, modular and portable simulation frameworks that can model the variety of computer systems (mobile, high-performance, etc.) and design targets (performance, power, reliability) of interest in coming years. Modular frameworks are especially important because they are well-suited to the construction and use of multiple modules from multiple research groups, and they permit simulation and analytic models to be integrated into a single simulation. Simulator frameworks need to be developed for both uni- and multi-processor workloads. NSF should also support academic purchase/licensing of industry modeling/analysis tools where appropriate.

- **Abstractions, Methodology, and Metrics:** NSF support for research on how to balance abstraction and precision to explore large design spaces accurately but efficiently. This should include research into new analytic models; how to integrate analytic models with detailed simulation; statistical or other

techniques to validate analytic models against real and simulated systems; and use of alternative measurement techniques like the use of built-in performance counters.

- **Accuracy and Validation:** NSF support for research into a new metrics and statistical techniques that provide deeper insight into system behavior; development of sound and verifiable simulation methodology; and investigation of how to validate research models with no existing physical implementation that can serve as a reference.

Of course, the research community must play its part too, by embracing high-quality research in these areas. But the panel argues that because work in these areas involves a large investment of time and effort, a substantial NSF investment of research funds is required to jump-start work in these areas. Without NSF funding, research and development in these areas is likely to languish at its current slow pace, and computer systems evaluation is likely to become more and more of an obstacle to maintaining the current fast pace of computer-systems innovation that is driving the information-technology revolution.

7 Meeting Process and Attendees

The meeting was held for approximately 4 hours per day on three days (December 1, 2, and 5), interspersed with the 2002 Micro Symposium in Austin, TX. The dates were chosen for two reasons. First, co-locating with Micro simplified travel plans for several attendees. Second, the gap of a couple days between the initial brainstorming and the final wrap-up of the workshop was helpful in letting ideas gel, and in letting the events and papers of the Micro conference influence our thoughts.

Before the workshop, we asked each attendee to provide a brief position paper with ideas on key problems and issues in the area of computer performance evaluation. To foster discussion, we emailed participants these questions:

- Metrics: What are the proper metrics to use for various architectural studies?

- Benchmarks: What are the appropriate benchmarks to use for various architectural studies in the next 5 years?

- Abstractions: How can researchers determine when abstractions that accelerate simulation (but may reduce accuracy) are appropriate and when they are not? How can their accuracy and precision be characterized?

- * Sampling: How/when/whether can researchers use mixed-mode simulations that vary between high and low amounts of detail in a single simulation? How can the accuracy of such simulations be verified?

- Sensitivity: How can researchers structure sensitivity studies to determine when a low-level error will translate into large errors, as opposed to when the low-level error's effect will be masked when aggregated into the full simulation?

- Validation methods: How can a processor simulation be validated before a similar processor is built?

- Hardware assists: How can simulations make use of hardware performance counters and other hardware assists for evaluating next-generation processors (which may have different organizations than the systems from which we are obtaining hardware counts)?

- Power: What new simulation questions are raised by the growing interest among architects in modeling CPU power/temperature?

- The above questions address immediate questions and problems facing the architectural community. Over the longer term (8-10 years) additional problems will arise. It is difficult to predict exactly what

these problems will be, and one goal of the workshop will be to try to identify some of them. But already we can identify several important questions.

- Technology and fabrication limits will play a more important role in determining feasible architectures. Estimating floor plans and modeling wire delays will become even more important. How should these effects be modeled in typical architectural studies?

- In addition to improvements in clock speed, increasing performance must clearly continue to come from exploiting various types of parallelism, instruction-level parallelism, task-level parallelism, and coarse-grained parallelism. Mechanisms for exploiting all these types may appear on the same chip. How should this be modeled?

- Heterogeneous chips, in which different CPUs, various specialized computation units, and various I/O drivers appear on the same die, may become increasingly common. How should these systems be modeled?

Prior to the workshop, Skadron and Martonosi collected position paper responses and organized answers. The initial 2 hours of the workshop consisted of further discussion on these questions, and others. Based on the position papers plus additional discussion, we identified a set of driving problems, and we grouped these into categories. The remainder of Saturday and Sunday's discussions focused on small breakout groups (3-5 people) discussing these problems.

On Wednesday afternoon, the participants met to begin summarizing the discussion. We also discussed actions to take that would most likely lead to success in addressing the problems covered by the workshop. The results of these discussions are summarized in this document.