Position Paper for
NSF Workshop on Computer Performance Evaluation

Sarita Adve
Department of Computer Science
University of Illinois at Urbana-Champaign
sadve@cs.uiuc.edu

I discuss two independent concerns related to performance evaluation that I believe require concerted action by the architecture research community.

What to evaluate – new workloads and metrics

To discuss performance evaluation, we need to discuss what is important to evaluate. Architecture research today is mostly driven by SPEC CPU benchmarks. As the number of companies and fraction of computers that care about SPEC benchmarks dwindles, can we continue to justify such a large effort on "computers that run SPEC?" In contrast, there is a growing diversity and market for systems running a variety of other workloads. I will specifically focus on multimedia, communications, and network applications, running on diverse systems such as handheld, laptop, desktop, and server systems. These workloads are no longer the domain of ASICs or special-purpose systems. General-purpose processors and/or techniques from such processors are beginning to be increasingly employed for such workloads. And yet, the quantitative approach for which our community is well known has barely been applied to this class of systems and applications.

An argument that SPEC benchmark driven results are applicable to the above workloads is not convincing. These workloads are significantly different, not only in their behavior, but in the ***metrics*** that measure good behavior. A few examples follow:

- The applications discussed are typically *periodic* and *real-time*; i.e., they periodically present a unit of work (e.g., a frame) that must be completed within a deadline. On the one hand, this imposes a challenging requirement – the computation must be finished within the deadline; otherwise, the consequences may be catastrophic. On the other hand, finishing the computation much before the deadline is of no value to the end user, and may be wasteful of other precious resources such as energy. Thus, the execution time metric (which is the primary measure of performance in conventional architecture) is not meaningful by itself.

- Typically, systems running these applications have *dynamic and multidimensional resource constraints*. For example, in wireless devices, time, energy, and bandwidth are all first-class resources and change dynamically. Again, previous metrics such as energy-delay product do not suffice since any reduction in energy may be acceptable with even a significantly higher delay (as long as it is within the deadline).

- There is often *no one correct output*. Instead, these applications allow a tradeoff between output quality and resource usage. A low quality video frame computation may use less processing time and energy and may be more acceptable to the user than a high quality video computation that quickly exhausts the battery.

- The real-time nature also means that *predictability* of the architecture is as important as "high performance." There is also a noteworthy difference between hard and soft real-time applications; the former does not tolerate any unpredictability, but the latter can tolerate some.

Thus, conventional metrics for performance evaluation fail with these applications and conventional workloads cannot be used to measure the appropriate metrics. Further, the relationship of the architecture with other system layers is also not well understood. For example, hardware must interact with the O.S. to ensure that slowing down an application to save energy will not interfere with the timely execution of another real-time application admitted into the system. Similarly, it is unclear how much of the network protocol stack must be modeled.

In summary, there are at least three major pieces of work related to performance evaluation required in this area:

- *Development of appropriate metrics for evaluation*, as motivated above.

- *Development of appropriate workloads*. Benchmarks for many basic tasks are lacking; e.g., network control plane processing tasks. We also need to consider workloads with multiple threads for a high-level application running on a real system (e.g., video encoder, multiple video decoders, speech codecs, and channel codecs for wireless video conferencing), with a realistic schedule that mimics real-time scheduling policies. We also need to consider adaptive applications that can change their output quality.

- *Understanding the interaction with other system layers* (operating system, network protocols, adaptive applications) *and abstractions for modeling them.*

Overall, I believe that emerging systems running real-time applications for multimedia, communications, and networking workloads are ripe for the quantitative analysis expertise that we have developed in our community, providing both intellectually challenging problems and a potential for high impact.

How to evaluate – getting valid results with unvalidated models

This part is relevant to both conventional and new workloads. The emphasis on quantitative evaluation is a key advance of the last decade. Unfortunately, we often seem to lose sight of the reason for this emphasis; i.e., to make effective architectural choices. Instead, often numbers produced from detailed simulations become an end in themselves. Some recent work has further heightened skepticism of research using simulators that are not carefully validated against existing hardware. Some skepticism is healthy, but we must be careful to avoid a blind promotion of such skepticism. A culture that requires all ideas to be evaluated with excruciatingly detailed simulators that are validated against some hardware can at best lead to incremental work. More dangerous, such a culture can lead to a false sense of confidence in the universal applicability of results that can only be really valid for the hardware and the workloads against which the simulator was validated. Further, given the slow speed of simulators today, it is likely that "validations" will be done only with some approximations of real workloads, making them less valuable.

As researchers, we can view our role as developing new design principles for practitioners' use or as developing isolated design points. In the former case (which forms the bulk of our research), we must be careful to not over-emphasize numbers that are output by a simulator reflecting one design point, no matter how detailed or validated the simulator. The more crucial aspect of a quantitative study instead should be the analysis that determines where exactly that X% benefit came from, and how X would change if the base hardware or software were different. It then follows that performance models need only be as detailed as necessary for this analysis.

Abstracting a model from a real system at the appropriate level of detail is a difficult exercise (as is writing a detailed simulator which itself is a low-level abstraction). The process of developing a relatively high-level abstraction in itself, however, can be valuable, potentially leading to insights that are not as easily evident with a brute-force detailed simulation approach. Regardless, a conscious awareness that any performance model is an abstraction, highlighting the abstractions made in the model when explaining the results, and understanding the sensitivity of the results to the different design choices is likely to lead to better science than statements such as "technique A gives a benefit of 4.2%."

To perform such science effectively, we need to revert back to some practices that have recently been less evident:

- Recognize that performance models do not have to be simulations. Analytic models are powerful tools for high-level understanding of many issues. Use of both analytic models and simulation can be an ideal combination – each can be used to validate the other, the simulation can give detailed information on a small part of the design space, and the analytic model can be used to explore the larger design space.

- Value the importance of (i.e., encourage publication of) new evaluation techniques that enable better (faster to run, easier to code, etc.) abstractions of real systems.

- Maintain a healthy skepticism for results that come from *any* single modeling technique (including detailed simulation or real hardware), and respect work that duplicates results with multiple modeling techniques.

- Emphasize the reason behind the numbers more than the numbers themselves.