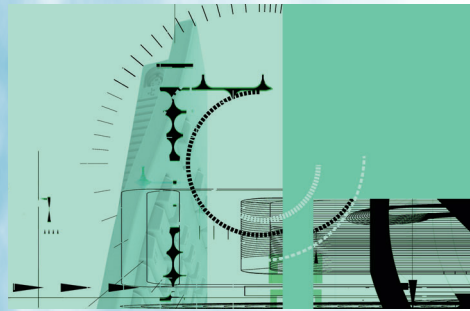


# A New Era of Performance Evaluation



*Sean M. Pieper*  
University of Wisconsin-Madison

*JoAnn M. Paul*  
Virginia Tech

*Michael J. Schulte*  
University of Wisconsin-Madison

**Long-standing techniques for performance evaluation of computer designs are beginning to fail. Computers increasingly interact with other computers, humans, and the outside world, leading to scenario-oriented computing, an emerging category of design that will enable future consumer devices and usher in a new era of performance evaluation.**

Since 1943, researchers have used latency and throughput as the primary metrics to describe computer performance. These metrics served us well because we used computers in fairly simple ways.

The unspoken assumption is that data is available on demand and only its quantity and content can affect execution time. This implies batch-style execution, in which measuring each program's speed, including the operating system, in isolation can determine overall performance. For performance-evaluation purposes, programs are merely an extension of instructions—reduced latency and higher throughput are always better. This perspective informs the design of benchmark suites such as those from the Standard Performance Evaluation Corporation (SPEC)<sup>1</sup> and the Embedded Microprocessor Benchmark Consortium (EEMBC),<sup>2</sup> which are composed of batch-style jobs executed in isolation.

For many new computer systems, such evaluation is misleading. Computers increasingly interact with humans, the physical world, and each other—often simultaneously. Overall performance in this context is a function not only of individual applications, but also of their interactions as they contend for resources both internal and external to the device. Cell phones, for example, often perform some baseband processing in software. Wireless communications arrive over time rather than on demand, and strict requirements dictate when outputs must occur. Other tasks such as video-conferencing depend on this software, but they also can compete with it for memory and processing resources. I/O subsystems over which the processor has little or no control and interdependencies between unrelated programs break the batch processing model, but they are essential aspects of this new computing style.

Researchers must describe modern computer usage in terms of scenarios consisting of numerous I/O streams, timing information, and parallel tasks that enter and leave the system, rather than in terms of programs executing in isolation from the physical world and each other.<sup>3</sup> Such use represents a new style of computing, which we call *scenario-oriented* to contrast it with other well-established computing styles such as general-purpose and application-specific. Table 1 compares these three styles.

Evaluation methods designed for general-purpose and application-specific computing are insufficient for scenario-oriented computing. Existing benchmarks do not reflect modern usage, and their metrics fail to describe performance as perceived by end users.

## COMPUTER USAGE EVOLUTION

As Figure 1a illustrates, with traditional computer usage, a single task, T1, enters the system, executes for some period of time,



**Table 1. Comparison of general-purpose, application-specific, and scenario-oriented computing.**

Computing style	User programmability	Design	Performance evaluation	Inputs
General-purpose	Complete programmability	Balanced performance	Each application evaluated individually	Sequenced by application
Application-specific	Limited or no programmability	Excellent performance for a single application	Compared against known requirements	Timed to external reference
Scenario-oriented	Can install software for new functionality	Variety of uses, but emphasizes performance of some	Holistic evaluation of scenario components and their interactions	Both sequenced by applications and timed to external reference

and then completes. Some time later, another task, T2, enters the system and takes its turn. This model abstracts schedulers and other operating system features to simplify performance evaluation. There is no contention between T1 and T2. Only program control flow sequences and interleaves data access; there is a single input stream and a single output stream.

In contrast, as Figure 1b depicts, modern usage is more complicated. Many tasks operate simultaneously, contend for resources, and communicate with each other. Unlike traditional usage, both asynchronous and streaming I/Os such as alerts and user inputs, webcams, and music are important to the functionality, and there is not necessarily a one-to-one mapping from inputs to outputs.

An arbitration layer uses preemptive scheduling to allow for interleaved execution and to enable multiple logical I/O streams to share a single physical link. This layer supports real-time requirements and user demand for concurrency. Advanced hardware also enables simultaneous execution. Complex interactions between tasks and hardware resources prevent describing an entire system's performance in terms of a single task evaluated in isolation or even of multiple tasks run one after the other. This break from traditional assumptions on computing causes Amdahl's law to fail for bounded systems with heterogeneous resources in that slowing down some tasks can actually improve overall performance.<sup>4</sup>

While Figure 1a accurately describes scientific and engineering usage, the computing industry has expanded beyond the small market of engineers and scientists who use computers to develop and run batch-style programs to an ever-growing group of nontechnical users. Software that uses processing power to deliver both new functionality and increasing ease of use has made this growth possible. These nontechnical users currently buy billions of processors every year in the form of cell phones, set-top boxes, and music players; and they expect these devices to make their lives easier and more enjoyable.

Examples such as e-mail, Web browsing, and gaming illustrate how researchers have historically harnessed increased processing power to create a virtual infrastructure unique to computing. These applications do more than simply facilitate problem solving; they actu-

ally create entirely new technological foundations that increase demand for computing. In turn, the applications themselves become ever more sophisticated. E-mail, for example, dates back to at least 1972, but increased memory and processing capability, as well as multithreaded operating systems, have expanded its capability far beyond the transmission of text messages.

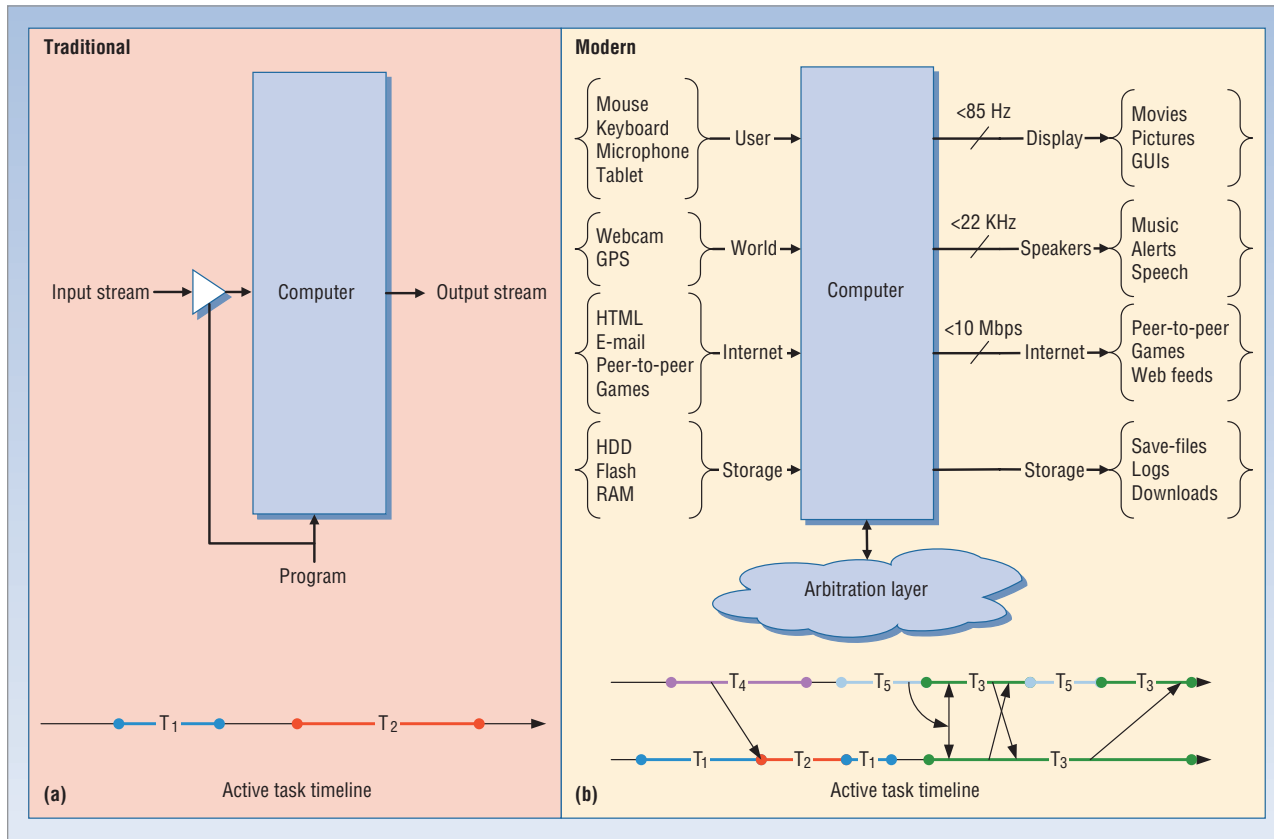
Modern e-mail programs have integrated this base functionality with features such as spell checking, junk-mail filtering, scheduling, sort and search functions, HTML support, image decompression, encryption, and virus checking. As a result, e-mail is significantly more useful, but it is also more complicated and computationally intensive.

This trend is not confined to general-purpose devices and their applications. Cell phones have reached the point where they no longer can be considered traditional application-specific devices. They now use multithreaded operating systems, such as Symbian and Windows CE, and they can run applications such as video-editing software and 3D games that would traditionally run on a PC. Users are thinking less in terms of general-purpose computers or single-purpose systems, such as phones, and more in terms of programmable convergence devices that integrate into various aspects of their lives. They expect such devices to facilitate common tasks and to enable novel ways of interacting with the world.

## TURNING POINT IN HARDWARE DESIGN

Modern users' demands are rapidly outpacing the capabilities of existing design methodologies. The computer community has been in this position before. Vacuum tubes gave way to discrete transistors, then came simple integrated circuits, followed by very large-scale integration systems. In each case, the response was to create entirely new foundational principles. As uniprocessor design hits its limits, researchers must find new design methodologies to deliver next-generation functionality.

Sematech's most recent *International Technology Roadmap for Semiconductors* suggests that single-core designs cannot scale beyond 20 to 25 million transistors. Multiprocessor designs with "SOC-like integration of less efficient, general-purpose processor cores with more



**Figure 1. Traditional and modern computer usage. (a) Common traditional tasks include compilation, data compression, physics simulation, placement and routing, and discrete event simulation. (b) Common modern tasks include image manipulation, video and audio playback, e-mail, virus scanning, Web browsing with dynamic content, voice over IP, and 3D gaming.**

efficient special-purpose ‘helper engines’” are projected to be the next step in computer evolution.<sup>5</sup> Developers expect as many as 63 processing engines—cores and custom logic blocks—on a single chip by 2009. The migration to single-chip heterogeneous multiprocessors (SCHMs) will pick up over the next few years and ultimately allow exponential increases in performance to continue while reducing reliance on clock scaling.<sup>5</sup>

Two early SCHM architectures for commercial devices are the Sony, Toshiba, and IBM (STI) Cell and the Sandbridge Sandblaster.<sup>6,7</sup> Cell is geared toward set-top boxes and game consoles, while Sandblaster targets wireless handsets. In both of these areas, the most difficult problems, such as physics processing for games and baseband processing in cell phones, contain significant data- and thread-level parallelism. To exploit this parallelism, both Cell and Sandblaster combine a cluster of single-instruction, multiple-data processors with a single scalar processor. Cell uses its scalar processor to coordinate the SIMD units, and Sandblaster uses its to handle user-interface tasks. Neither architecture reserves processors for specific functions. As a result of their nontraditional design and programming model, processors such as Cell and Sandblaster have been described as system-on-chip designs, but this is not strictly accurate. Both


Cell and Sandblaster are more accurately described as “processors of processors.”

SoC descends from application-specific integrated circuit design and provides a methodology to rapidly develop integrated circuits for complex, but well-defined, task sets that are fixed at design time. The SoC design style divides the chip into several units; some of these units can be programmable, but each has a fixed purpose. Cell and Sandblaster diverge from this model by considering the entire chip as a programmable device that must be able to dynamically reallocate resources. They also are intended for devices that are marketable based on their compelling features, rather than sheer processing power.

### SCENARIO-ORIENTED COMPUTING

The changes in usage combined with developments in technology point to a new organizing principle for design—rather than being general-purpose or application-specific, computing is becoming scenario-oriented. Consider an onboard navigation system that determines its current location using GPS, and receives verbal instructions, such as “Go to 1600 Pennsylvania Avenue.”

In response to the user’s command, the system connects to a map server and checks for traffic advisories,



calculates and displays an optimized route, and transmits the directions through speech synthesis software as the user nears the destination. If a traffic advisory arrives, the computer drops the speech synthesis and seeks an alternate route. Unlike application-specific computing, the processor performs different tasks over time, but unlike general-purpose computing, these tasks share a common goal.

In contrast with the assumptions of both general-purpose and application-specific design, actual usage of devices such as cell phones, PDAs, and set-top boxes is modal. Users view these devices differently according to their immediate purpose—they might use a smart phone as a scheduler, music player, game-playing device, digital camera, or simply as a phone. These devices also can implement a single mode in several ways—for example, a user might play songs in several different formats while using the device as a music player. The user's expectations distinguish the modes, rather than the actual hardware or software that enables them.

Because customers expect a variety of uses for a finite amount of silicon, heterogeneous programmable cores become the central elements in scenario-oriented hardware. In contrast to the fixed-purpose resources in SoC and other application-specific design styles, the processing power of these cores is intended for a wide range of tasks. Unlike general-purpose computers, scenario-oriented devices must accommodate varying demands for different types of processing within a finite amount of silicon and certain time constraints. Heterogeneity is a response to this challenge. Software designers can leverage modality to inform scheduling decisions and use heterogeneous cores more effectively.

## FAILURE OF EXISTING METRICS AND BENCHMARKS

A benchmark suite is a set of applications that provide a representative sample of usage. SPEC CPU,<sup>1</sup> the primary benchmark suite computer architects use, contains a variety of real engineering and scientific applications that are selected and modified to run with minimal operating system support and interaction. The EEMBC benchmarks, which contain representative kernels and applications from the embedded domain, support embedded systems design.<sup>2</sup> The Stanford SPLASH benchmark suite, which measures the runtime of parallelizable algorithms, provides similar services for traditional multiprocessor architectures.<sup>8</sup> All applications in these benchmark suites are batch jobs and are executed in isolation. Figure 1a illustrates this type of usage.

Performance in SPEC is measured as speedup,  $s = \tau_{\text{reference}} / \tau_{\text{measured}}$ , over a reference system. Because the

design goal is to provide excellent performance under arbitrary usage, each application's speedup is treated equally, using the geometric mean to generate a composite score. The geometric mean places greater weight on entries with low performance than those with high performance—if a single result is 0, the entire output is 0, giving this entry infinite weight. This rewards balanced performance, which is appropriate for general-purpose usage, but does not accurately describe scenario-oriented performance.

SPEC also includes SPEC\_rate, a throughput measurement intended for multiprocessor systems. To generate SPEC\_rate scores, the computer executes  $n$  copies of each task simultaneously and then measures the time to complete all  $n$  copies. This measurement anticipates homogeneous usage appropriate to industrial applications such as simulation, Web hosting, database processing,

and supercomputing. Scenario-oriented design, in contrast, anticipates diverse usage, as is common in most consumer applications.

Some newer benchmarks such as Business Applications Performance's SYSmark and Futuremark's 3Dmark<sup>9</sup> are more representative of commercial use. SYSmark evaluates computer performance in a business setting. It uses common commercial applications such as Adobe Acrobat Reader, Macromedia Dreamweaver, McAfee VirusScan, and Microsoft Office in combination with input events and data generated by observing real users. Multiple applications execute together under different scenarios, such as communication (e-mail and Web browsing) and data analysis (database queries and spreadsheet operations). The benchmark reports separate scores for each scenario. SYSmark focuses on response times rather than runtimes, reflecting the fact that many applications are event-driven and can go idle while the user is not interacting with them.<sup>10</sup>

SYSmark comes closer than SPEC to describing modern usage, but it still does not include any real-time applications. Real-time tasks such as streaming media, baseband processing, and voice recognition are essential to multimedia, mobile usage, and human-computer interaction. Their absence limits SYSmark's ability to describe scenario-oriented usage. SYSmark's focus on current usage also limits its applicability to scenario-oriented hardware design. Several years can pass between the time developers make fundamental design decisions and when a new device hits the market. Designers need the ability to evaluate performance under anticipated future workloads.

3Dmark evaluates gaming performance under next-generation loads. It measures the real-time frame rate of a set of games with extremely demanding graphics.

**The processing power of heterogeneous programmable cores is intended for a wide range of tasks.**

3Dmark originally focused on graphics processing units, but recently it has added a CPU portion to model the impact of AI and physics calculations on frame rate. While 3Dmark can describe real-time performance of future gaming workloads, its dependence on frame rate as a figure of merit limits its applicability to other areas.

While researchers have invested much effort and creativity in the design of these benchmarks and their associated metrics, they are insufficient for guiding scenario-oriented design for the following reasons:

- *Their composite metrics weight all applications equally.* This is a relic of sharing general-purpose processors for batch jobs. With interactive usage, fast responses to some events are more important than the response time to others.
- *They judge hardware on its ability to accelerate, rather than enable.* Customers expect increasing integration of new features such as speech recognition, rather than faster execution of existing features such as spell checkers.
- *They can't describe cooperation across tasks.* If several tasks operate toward a common purpose, the acceleration of some tasks is not necessarily beneficial and can even degrade overall performance.

Fundamentally, none of these approaches identifies the modifications that would improve a scenario-oriented design. Computer architecture has historically been the art of identifying performance bottlenecks and then identifying performance facilitators such as caches and branch predictors to alleviate these bottlenecks. Different performance facilitators will exist for future architectures, but these will facilitate critical cases—the instances when application software overloads the system and performance rapidly deteriorates. Scenario-oriented benchmarks must enable designers to identify critical cases and, in doing so, aid the discovery of new performance facilitators.

## EVALUATING SCENARIO-ORIENTED PERFORMANCE

Figures 2 and 3 describe a hypothetical system's performance in an onboard navigation scenario in terms of usefulness and timeliness. Usefulness indicates the degree to which the device helps the user navigate, and timeliness indicates the device's ability to perform calculations

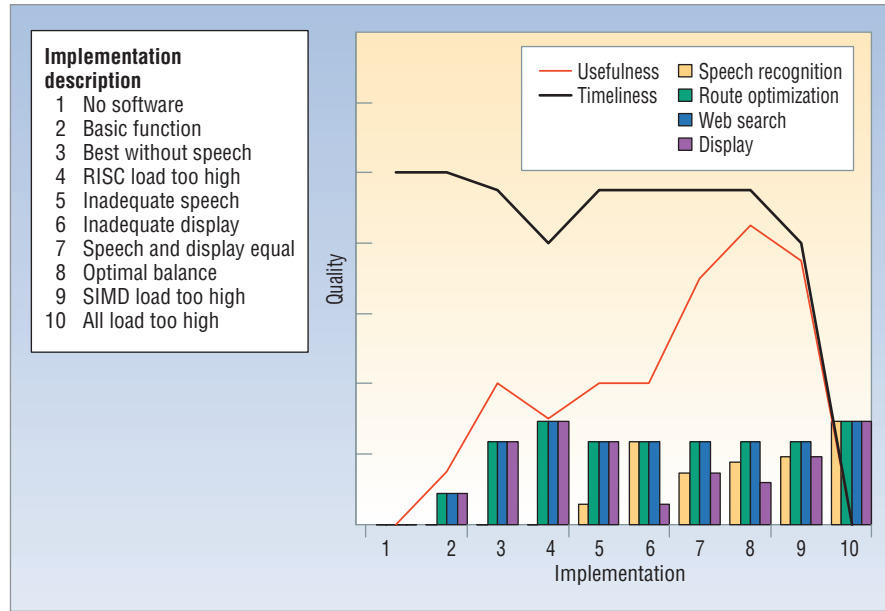


Figure 2. Timeliness and usefulness of various implementations of a navigation scenario.

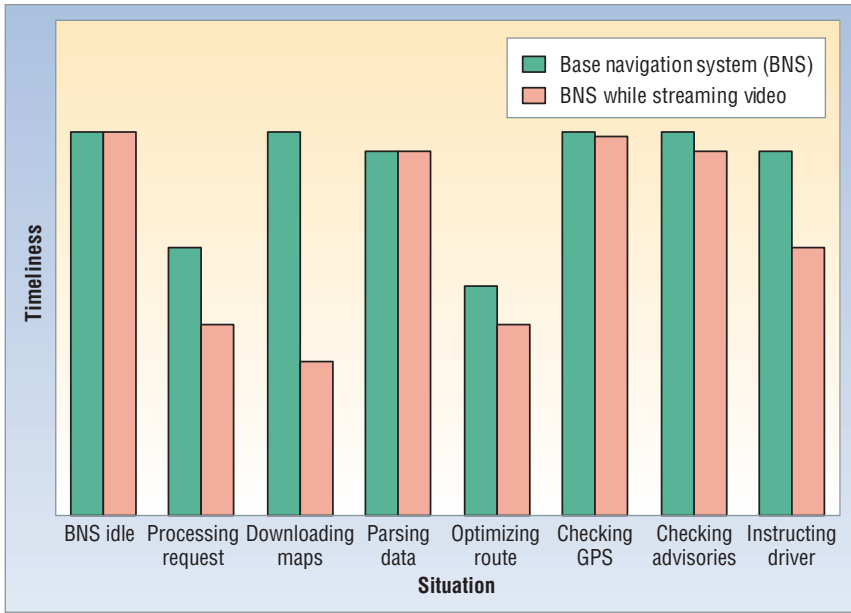
in a timely manner. Although subjective, human or even marketing studies can measure usefulness. Timeliness is a complex metric—some deadlines can be more important than others (this relative importance, of course, is also subjective) and creating a composite can be difficult. The importance of such metrics lies in bringing performance evaluation in line with user satisfaction.

The navigation system is assumed to involve speech recognition, route optimization, Web search, and graphical display, executing on a SCHM with both reduced-instruction-set computer and SIMD cores. The speech and display software run better on the SIMD cores, while the route optimization and Web search run better on the RISC cores.

Figure 2 compares the performance of several possible implementations of a navigation scenario as the component algorithms' complexities vary. Each group of bars represents an implementation, and each bar—whose height indicates relative computational complexity—represents a task. Reasons for complexity changes can include algorithm selection, the amount of data the system is processing, or control dependencies on input values. We assume that, given infinite processing power, higher complexity results in increased usefulness.

Figure 2 illustrates three important points:

- Software and hardware are not evaluated independently.
- Adding a new feature can significantly increase a device's usefulness even if the individual quality of other features is sacrificed.
- The relative amount of computing power dedicated to each feature has a significant effect on usefulness and timeliness.



**Figure 3. Navigation system timeliness.** The green bar shows the timeliness of the navigation system executing alone, while the yellow bar describes timeliness when the system is downloading and displaying a streaming video at the same time.

In implementation 1 in Figure 2, no software is running. As a result, no deadlines are missed, and the timeliness rating is perfect. However, the lack of functionality brings the usefulness score to zero.

Implementation 8 does not have perfect timeliness, but it incorporates enough functionality in a sufficiently timely manner that the device is very useful. In implementation 10, the load is too high, and all deadlines are missed. Timeliness bottoms out, and this degradation destroys usefulness. On a more powerful processor, however, timeliness would improve, and the usefulness of these more complex implementations would increase with it. Software must match hardware to optimize performance.

Comparing implementations 1-4, which do not include speech recognition, with implementations 5-8 shows the impact of adding an additional feature. Usefulness hits a local maximum in implementation 3, and then begins to decrease because the requirements for route optimization and Web search are too high, and they must be performed on SIMD processors where they execute less efficiently. Using these processors to implement speech recognition, rather than improve existing features, will make the device far more useful overall.

Implementations 5-8 also demonstrate the importance of striking the correct balance. When the speech recognition is prioritized too much in implementation 6, the display must be sacrificed to maintain timeliness. In implementations 7 and 8, the balance adjusts to improve usefulness without reducing timeliness. This leads to an unequal division of computing resources.

Figure 3 illustrates a navigation system's dynamic behavior and the impact of adding an extra feature to

this system. The figure shows different situations in roughly chronological order from left to right. The navigation system is idle until it receives a verbal request, which triggers a chain of computation that continues until the system finds a route. From then on, the navigation system periodically polls the GPS and traffic advisory Web sites and announces each turn. Two bars illustrate each situation. The green bar shows the timeliness of the navigation system executing alone, while the yellow bar shows the timeliness when the system is downloading and displaying a streaming video at the same time.

This type of graph can reveal unexpected consequences of adding new functionality that might not be apparent from the macroscopic view of Figure 2. For

example, downloading map data is not a problem for the base navigation system, so we might not think about how adding a new feature affects performance. Figure 3, however, reveals that this is the worst case when simultaneously streaming video. Further analysis might reveal that limited bandwidth is to blame, which could lead to solutions such as compressing map data or designing around a more sophisticated wireless protocol.

Figures 2 and 3 demonstrate the potential for new performance representations to isolate critical cases and describe tradeoffs. When performance evaluation breaks free of traditional assumptions, many representations become possible.


## TAKING PERFORMANCE EVALUATION INTO A NEW ERA

Because it differs so vastly from both general-purpose and application-specific computing, scenario-oriented computing requires an overhaul of performance evaluation. We can divide this challenge into benchmark selection and metric design.

### Scenario-oriented benchmark criteria

Benchmark selection is the problem of defining a computational load in terms of inputs (programs and data) and timing information. New scenario-oriented benchmarks should satisfy the following criteria:

- *Include software and hardware interactions.* Tasks can wait for each other, communicate, spawn children, and leave the system. As a result, the processor's



load can change dramatically during usage. Figure 3 illustrates this type of dynamic behavior as well as an example in which multiple tasks compete for a shared hardware resource.

- *Provide timing information for inputs and outputs.* Because a mouse click can spark a chain of calculations, its occurrence relative to other events is important. For example, Figure 3 shows how processing directions while streaming a video can hurt performance, but running the streaming video is fine when the base navigation system is idle. Outputs such as video and music have associated timing requirements that play an important role in determining perceived quality.
- *Exercise critical cases.* Scenarios should isolate the knees of performance curves. Because this will vary according to the underlying hardware, mechanisms must exist to adjust requirements in a variety of ways. Implementation 9 in Figure 2 illustrates this type of operating point.
- *Describe sets of usage modes.* Although they are geared toward certain uses, scenario-oriented processors can provide additional value in other modes. Quantitative evaluation of tradeoffs between primary and secondary modes is necessary. For example, is it better to have an excellent handheld TV or to sacrifice some TV functionality to allow using the same device as a phone?

We propose a fundamental change in the structure of benchmarks for scenario-oriented computing. Accordingly, metrics for evaluation must also change.

### Scenario-oriented metric criteria

Metric design is the problem of quantitatively assessing the execution of a benchmark. Scenario-oriented performance metrics should satisfy the following criteria:

- *Differentiate application elements by relative impact on usefulness.* Users do not have equal performance requirements for all tasks or even all portions of a single task. In Figure 2, for example, speech recognition is slightly more important than display because drivers try to avoid looking away from the road, and speech recognition assists with this requirement.
- *Account for nonlinearity.* A hard real-time task will not perform correctly if it can't meet its deadlines. Once it meets all deadlines, however, there might be no benefit to further accelerating that task. Human interaction is similar—beneath some threshold, humans can't perceive faster response times. For example, a graphics task doesn't benefit from frame rates higher than the monitor can support.
- *Describe critical cases.* Understanding the tradeoffs in a scenario-oriented computer requires knowing when the interaction of time, data, functionality, and

hardware causes overall performance to degrade. For example, there should be a way to describe exactly what happens in implementation 9 in Figure 2 that causes performance to rapidly drop or to explain the interactions that occur in Figure 3 when downloading a map while displaying a streaming video.

- *Have a visual representation.* Pictures and graphs are powerful tools for rapidly conveying complex information and tradeoffs. Developers often use bar charts in conjunction with SPEC to demonstrate improvements in throughput and latency. Scenario-oriented designers will frequently need to select one approach from numerous alternatives when a single “best” choice is not apparent. Intuitive ways of expressing the results of future benchmarks are necessary to guide such decisions.

These properties for future benchmarks and metrics depart sharply in structure and focus from those that historically have guided computer design. This overhaul is necessary for computer designers to develop and evaluate the new principles required to deliver compelling devices to end users.

**D**evelopers must reconsider performance evaluation in light of emerging hardware and software trends. A new era of scenario-oriented computing is dawning. The means to evaluate new performance facilitators and decisions shape, both directly and indirectly, approaches to design and their ultimate success—or failure. The advancement of scenario-oriented design, therefore, hinges on the development of appropriate evaluation methods. We invite the community to join us in considering this challenge. Contact us at [soar@ece.wisc.edu](mailto:soar@ece.wisc.edu) or visit [www.ece.wisc.edu/~soar](http://www.ece.wisc.edu/~soar) for more information. ■

---

### Acknowledgments

This work was supported in part by the National Science Foundation under grants 0607934 and 0606675. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

---

### References

1. J.L. Henning, “SPEC CPU2000: Measuring CPU Performance in the New Millennium,” *Computer*, July 2000, pp. 28-35; [www.spec.org](http://www.spec.org).
2. M. Levy, “Evaluating Digital Entertainment System Performance,” *Computer*, July 2005, pp. 68-72; [www.eembc.org](http://www.eembc.org).

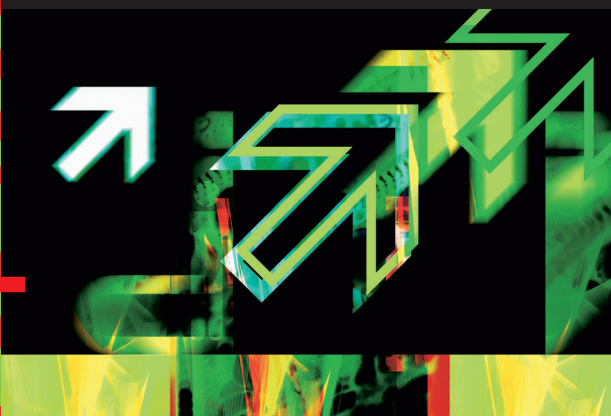
3. J.M. Paul, D.E. Thomas, and A. Bobrek, "Scenario-Oriented Design for Single-Chip Heterogeneous Multiprocessors," *IEEE Trans. VLSI*, Aug. 2006, pp. 868-880.
4. J.M. Paul and B.H. Meyer, "Amdahl's Law Revisited for Single Chip Systems," *Int'l J. Parallel Programming*, Apr. 2007, pp. 101-123.
5. Sematech, *International Technology Roadmap for Semiconductors (ITRS)*, 2005; [www.itrs.net/Links/2005ITRS/Home2005.htm](http://www.itrs.net/Links/2005ITRS/Home2005.htm).
6. H.P. Hofstee, "Power-Efficient Processor Architecture and the Cell Processor," *Proc. 11th Conf. High-Performance Computing Architectures*, IEEE CS Press, 2005, pp. 258-262.
7. M.J. Schulte et al., "A Low-Power Multithreaded Processor for Software Defined Radio," *J. VLSI Signal Processing Systems*, June 2006, pp. 143-159.
8. S.C. Woo et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Proc. 22nd Ann. Int'l Symp. Computer Architecture*, ACM Press, 1995, pp. 24-36.
9. N. Renqvist and M. Kallinen, "3DMark06," white paper v1.0.2, Jan. 2006; [www.futuremark.com/products/3dmark06](http://www.futuremark.com/products/3dmark06).
10. J.M. Sammons and C.H. Sauer, "Measuring the Performance of Personal Computers," *Proc. 37th IEEE Computer Society Int'l Computer Conf. (Comcon 92)*, IEEE CS Press, 1992, pp. 311-313.

*Sean M. Pieper is a PhD student in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. His research interests include scenario-oriented and power-efficient computer architectures. Pieper received an MS in electrical and computer engineering from Carnegie Mellon University. Contact him at [spieper@wisc.edu](mailto:spieper@wisc.edu).*

*JoAnn M. Paul is an associate professor in the Department of Electrical and Computer Engineering at Virginia Tech. Her research interests include the design, modeling, simulation, and evaluation of single-chip heterogeneous multiprocessors. Paul received a PhD in electrical engineering from the University of Pittsburgh. She is a member of the IEEE and the IEEE Computer Society. Contact her at [jmpaul@vt.edu](mailto:jmpaul@vt.edu).*

*Michael J. Schulte is an associate professor in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. His research interests include high-performance embedded processors, computer architecture, domain-specific systems, and computer arithmetic. Schulte received a PhD in electrical engineering from the University of Texas at Austin. He is a senior member of the IEEE and the IEEE Computer Society. Contact him at [schulte@ece.wisc.edu](mailto:schulte@ece.wisc.edu).*

Sign Up Today



For the  
IEEE  
Computer Society  
Digital Library  
E-Mail Newsletter

- Monthly updates highlight the latest additions to the digital library from all 23 peer-reviewed Computer Society periodicals.
- New links access recent Computer Society conference publications.
- Sponsors offer readers special deals on products and events.

Available for FREE to members, students, and computing professionals.

Visit [http://www.computer.org/services/csdl\\_subscribe](http://www.computer.org/services/csdl_subscribe)