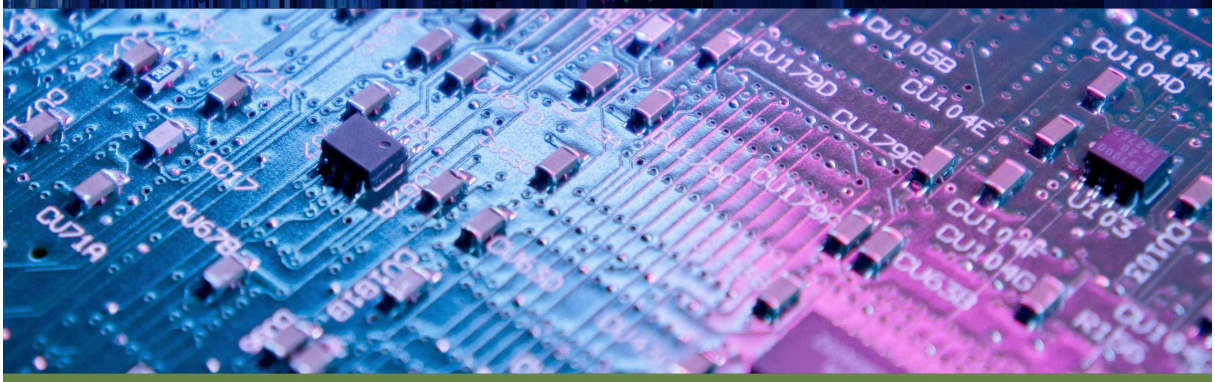
U.S. DEPARTMENT OF ENERGY | Office of Science

Report on the ASCR Workshop on Modeling and Simulation of Exascale Systems and Applications



Workshop Committee
Adolfy Hoisie (PNNL, Chair), Darren Kerbyson (PNNL), Robert Lucas (USC/ISI), Arun Rodrigues (Sandia), John Shalf (LBNL), Jeffrey Vetter (ORNL)

ASCR Program Managers
William Harrod (DOE/ASCR Research), Sonia R. Sachs (DOE/ASCR Computer Science)

Further Contributors to the Report
Kevin Barker (PNNL), Jim Belak (LLNL), Greg Bronevetsky (LLNL), Chris Carothers (RPI), Boyana Norris (ANL), Sudhakar Yalamanchili (GT)

August 9 – 10, 2012 **University of Washington Seattle, WA**

1. EXECUTIVE SUMMARY 3

2. WORKSHOP OVERVIEW 4

3. INTRODUCTION 5

4. STATE OF THE ART 7

4.1 MODELING STATE OF THE ART 7
4.1.1 CURRENT MODELING LANDSCAPE 7
4.1.2 IMPACT OF MODELING 11
4.1.3 GAPS IN MODELING STATE OF THE ART 12
4.2 EMULATION AND SIMULATION 13
4.2.1 CURRENT SIMULATION LANDSCAPE 13
4.2.2 FRAMEWORKS 14
4.2.3 COMPONENT SIMULATORS 15
4.2.4 PHYSICAL/TECHNOLOGY 16
4.2.5 IMPACT OF SIMULATION 16
4.2.6 GAPS IN THE SIMULATION STATE OF THE ART 17
4.3 OVERARCHING ISSUES 18
4.3.1 VALIDATION AND VERIFICATION 18
4.3.2 DATA REPOSITORY 18
4.3.3 INTEROPERABILITY 19
4.3.4 MULTI-SCALE 19
4.3.5 SOFTWARE ENGINEERING 19
4.3.6 SCALABILITY 20
4.3.7 MACHINE BENCHMARKING AND DATA CAPTURE 21
4.3.8 BEST PRACTICE VALIDATION PROCESSES 21
4.4 ASSOCIATED TECHNOLOGIES 22
4.4.1 PROXY APPLICATIONS 22
4.4.2 MEASUREMENT TOOLS 22
4.4.3 MODEL LANGUAGE TOOLS 22
4.4.4 TESTBEDS 23

5 CRITICAL TECHNOLOGIES OF MODELING AND SIMULATION 23

5.1 INTEGRATED MODSIM FOR ENERGY, PERFORMANCE, AND RELIABILITY 24
5.2 NEW INTEROPERABLE MACHINE AND APPLICATION ABSTRACTIONS FOR 24
5.3 MODEL GENERATION 24
5.4 DYNAMIC MODELING 25
5.5 MODEL AS AN ACTIONABLE TOOL 25
5.6 INTEGRATION OF METHODOLOGIES 26
5.7 LIFE CYCLE COVERAGE 27
5.8 STANDARDS, INTEGRATION, AND INTEROPERABILITY OF MODSIM METHODOLOGIES 27

6 SUMMARY OF CRITICAL TECHNOLOGIES 28

7 RECOMMENDATIONS AND PATH FORWARD 29

8 BIBLIOGRAPHY 31

APPENDIX. WORKSHOP AGENDA AND PARTICIPANT LIST 34

1. Executive Summary

In the last few years, numerous studies commissioned by the Department of Energy (DOE), the Defense Advanced Research Projects Agency (DARPA), and others have examined the challenges that will have to be overcome for computational science to achieve exascale, a two order-of-magnitude increase in capability beyond what is possible today. Exponential growth in the number of devices (Moore's Law) is expected to continue unabated, but the end of Dennard scaling, and hence the growth in performance of individual processors, has led to exponential growth in concurrency. Power consumption has also grown to the point where the cost of the electrical power for systems threatens to exceed the cost of acquiring them. Ever-shrinking devices, and the growth in their number, are expected to increase the rate of soft, or transient, errors such that applications will no longer be able to assume correct behavior of the underlying machine. Adapting to such systems will require new mathematical algorithms that minimize synchronization and data movement, and a new generation of scientific software.

Rather than addressing these issues in isolation, a new process of "Co-Design" is being pursued in which application and computer scientists work toward the common goal of an exascale ecosystem of systems and applications. Modeling and simulation (ModSim) is a critical part of this process. It enables scientists and engineers to analyze future algorithms, applications, and computing systems, long before they are realized, and make the necessary design decisions such that exascale science is broadly achieved. Later, when exascale applications and systems are realized, ModSim technology will enable us to understand their behavior, debugging and optimizing them.

Given the increasing importance of modeling and simulation, a workshop was organized by DOE/ASCR to assess the current state of the art and to identify research challenges that must be overcome in this field. Fifty experts, representing academia, industry, and government, participated in the workshop, and others contributed both beforehand and afterwards. This report summarizes their findings.

Today's ModSim research is widely dispersed in academia, national laboratories, and industry. It builds on a foundation of tools and techniques from an equally diverse research community. It is challenged by the emergence of new computing devices and heterogeneous systems as well as increasingly sophisticated, multi-physics applications. To enable the scientific community to succeed in the development and use of exascale systems, the following research challenges must be addressed:

- Create new modeling and simulation technology to cover the entire spectrum of scale from cycle accurate models of devices to full exascale systems and applications.
- Develop standards and interfaces allowing integration of independently developed models and tools.
- Enable new modeling and simulation technology for the emerging challenges of energy and resilience, as well as their impact on performance.
- Examine the new concept of ubiquitous, dynamic models that enable real-time understanding of running exascale systems, and optimization of their throughput.

The ModSim workshop participants believe that it is possible to achieve these goals. They recommend that DOE Advanced Scientific Computing Research program (ASCR) initiate a research program to do so.

2. Workshop Overview

The path to exascale is punctuated by multiple significant technical challenges. Creating billions of threads of parallelism needs to be tractable and manageable. New algorithms and programming models may be required that can use these extreme levels of parallelism and can be efficiently mapped to system resources.

Power efficiency is emerging as the overarching design constraint. The cost of data movement in both power and performance is larger than the cost of the arithmetic, and may soon be higher by orders of magnitude. High-speed interconnection networks and communication mechanisms must cope with this size and complexity within a limited power envelope, which may require streamlined, lower-dimensionality topologies. The increase in the depth of memory hierarchies, including additional layers in the input/output (I/O) and storage subsystem design via the likely addition of power-efficient, nonvolatile memories, leads to greater distances for data to travel, as well as increased programming complexity. A further issue to overcome at exascale will be the intrinsic limitation in reliability given the size and complexity of the systems, which will require increased resiliency. There is no one-size-fits-all solution, and the challenge ahead involves dealing with the combined reliability, power, and performance requirements—and their tradeoffs—at all levels of the stack between the hardware and the application. These tectonic shifts in computer architecture are anticipated to radically change the programming model and software environment at all scales in future computing systems. The designers of High-Performance Computing (HPC) hardware and software components have an urgent need for a systematic methodology that reflects future design concerns and constraints.

At this juncture, when the HPC community, led by the U.S. Department of Energy (DOE), is embarking on the path to exascale, hardware-software co-design was identified as a central strategy to help address these challenges. Co-design is an iterative process, in which the goal is to optimize both applications (algorithms and algorithmic implementations) and hardware in the space defined by performance, power, reliability, and cost. An effective co-design process requires the ability to accurately predict the performance and power consequence of any design tradeoff in algorithms or hardware configuration. ModSim of the entire hardware-software (HW-SW) stack plays an essential role in providing this predictive capability because it enables the quantitative exploration of HPC system design tradeoffs.

The ModSim workshop was called to address the role of modeling and simulation as a set of methodologies that can guide the optimal co-design of exascale systems and applications. The ModSim workshop included several areas of emphasis for attendees, including:

- surveying the state of the art in ModSim technology
- identifying key areas of co-design, where ModSim can have a significant impact
- discussing the emerging architectural trends likely to materialize at exascale and ways to capture those in models
- identifying research and development areas that require pointed and/or joint efforts
- engaging in in-depth technical discussions regarding increased accuracy, coverage, and ModSim impacts
- exploring the technical community and its interactions with other topical or broader exascale endeavors

- discussing the best ways of disseminating methodologies, tools, and software.

The workshop agenda also included a presentation on the exascale challenges [23], presentations by leading experts in ModSim from DOE laboratories, academia, and industry, as well as a focused panel discussion with industry, and several breakout discussions. Going forward, a plan has been adopted to host annual ModSim workshops focused on identified critical technologies.

3. Introduction

The goal of computational science research is to find ways for present or future computing systems to address key national scientific missions in the most productive way. Productivity is a combination of metrics that includes time-to-solution, cost (to install and operate computing systems), accuracy of the results produced (both in terms of numerical precision and applicability of the answer to the target scientific domain), and overall usability of the computing system.

Figure 1 shows a continuum of activities undertaken by the scientific community to connect the nation's overall needs to the computations that support them. Each box in the figure notionally represents a different emphasis area and different HPC workflow scale. At the level of scientific policy, it first is necessary to articulate key national missions, such as energy independence or high economic productivity. These mission goals can be translated into concrete scientific questions that need to be answered, such as understanding the details of combustion or efficient information analysis and management for national security. The scientific questions become computational problems that must be solved to provide an answer or enable these activities. In turn, the computational problems need to be tackled via appropriate application, machine, and programming model designs. Finally, the executing systems and applications must be continuously monitored and optimized to deliver continued high performance in the face of changing application needs.

ModSim research is a vital component of every aspect of computational science planning and execution because it enables community leaders, individual system and application developers, and users to predict the outcomes of complex decisions undertaken during the design, implementation, and maintenance phases of developing an effective HPC ecosystem. In Figure 1, ModSim's widespread applicability is shown via red asterisks that indicate specific areas where ModSim is applied. This capability is instrumental for enabling computational science and for enabling HPC communities to design computing strategies, concrete systems, and applications that provide a balance of productivity metrics and enable new breakthrough science that will provide high value in a cost-effective manner.

Over time, the ModSim field has evolved advanced techniques and methodologies that have greatly improved accuracy, applicability, and targeted uses for this technology. This community is now well positioned to employ ModSim for a myriad of tasks related to system architecture, system software, and applications. The growth in this capability is not only visible on the technical front, but also through the existence of a significant talent pool, which is able and ready to tackle challenging new problems. The available expertise is diverse and exists in "centers of emphasis" at national laboratories, academia, and vendor research and development centers.

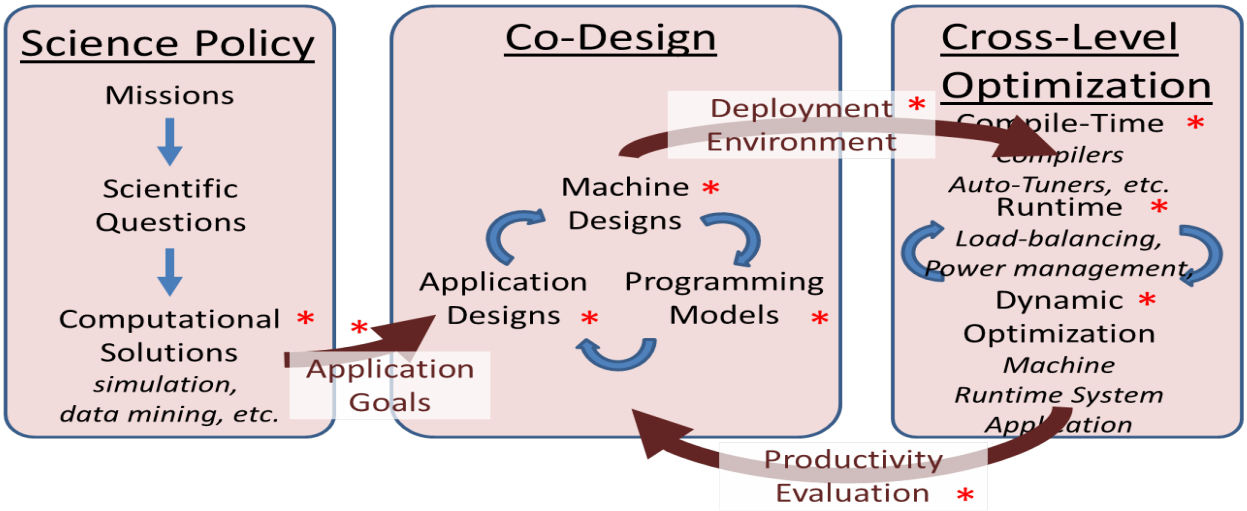


Figure 1. Notional HPC workflow. The * represent domains of use for modeling and simulation

While ModSim has evolved this advanced capability, it will require significant investments to focus the full force of the community on addressing the technology challenges of co-design for exascale systems. Existing techniques are primarily applied to performance prediction, but other crucial areas such as power and energy modeling are now emerging due to the anticipated power constraints expected at exascale. Reliability modeling, confined mostly to statistical analysis of faults, has evolved on a dimension orthogonal to performance and power. Much of the performance and power analysis in HPC still is dominated by the use of simple heuristic metrics, including efficiency, scalability, or "bytes/flop" that offer high-level information that does not offer any actionable advice. This exemplifies why new investments are needed to create more quantitative and diagnostic metrics: appropriate metrics and tools simply are not available to segments of the community. In our empirical observation, we recognize that ModSim is used "after the fact," which bypasses an opportunity to impact design stages in a predictive fashion. The limited role that ModSim currently plays relative to its potential impact on the co-design process impresses the need for increased investment in both the methodology and modeling tools for predictive modeling in earliest stages of HPC system and application design. With a few notable exceptions, the small projects and groups that develop ModSim methods and tools lack common interfaces and technical lingo due to uncoordinated and diffuse funding. There is a huge opportunity for the ModSim Nexus to organize this community around exascale to bring the full force of ModSim technology targeted at a common set of challenges. Section 4 of this report presents a detailed description of the research field in ModSim.

The evolution of previous generations of HPC machines was fueled by device-scaling behaviors that realized Moore's Law. However, the end of Dennard's scaling at the device level also changed performance-scaling laws at the system level. In particular, the physics of processing has fundamentally redefined the manner in which computation can be implemented toward scale performance to exascale. Moore's Law now leads to increasing power densities, and performance scaling will be realized via scaling of energy efficiencies. The result is major disruptions in hardware and software architectures and, consequently, a significant gap in our current models of feasible exascale system solutions, making ModSim a critical component and central for illuminating the path forward to exascale.

To achieve exascale, the community needs "ubiquitous modeling," i.e., modeling at all stages and levels. The anticipated adaptivity of exascale systems and applications will uncover new

boundaries that require tradeoffs to be negotiated, necessitating predictive ModSim methods. New execution models may be in order, and modeling will be expected to anticipate the advantages and disadvantages of new execution models before actual implementations are completed. Tackling performance, power, and reliability as disjoint targets will be replaced by simultaneous modeling and simulating. Adaptivity will require fundamental, new approaches to modeling, or what we call "dynamic modeling." New tools of the trade will need to be developed, as well as ways to make these dynamic models "actionable" and embed them into introspective runtimes or programming models. Significant new and focused investments are necessary to accomplish these tasks and integrate these methodologies into ModSim tools with broad use implications in all stages of the application and system life cycle, including research, design, implementation, optimization, and maintenance. The ModSim research directions necessary to respond to the exascale challenges are noted in Section 5.

ModSim has a multitude of potential applications for DOE, particularly within ASCR for architecture, programming models, runtimes, application development support (Scientific Discovery through Advanced Computing, or SciDAC), exascale application development (Co-Design Centers), systems optimization, and applications, to name only a few. Based on our assessment, exascale computing challenges require a focused investment strategy in ModSim, which will provide coverage of the entire spectrum, lead to trusted methodologies and tools that interoperate, and offer analysis and insight at various levels of accuracy and detail (as required by its diverse user community). The outcome is a strengthened and diverse ModSim community (government, academia, and vendors) that delivers on challenges posed to it in a cost-efficient manner.

4. State of the Art

4.1 Modeling State of the Art

Given the complexity of supercomputing architectures, it is difficult to predict how fast an application running on today's petascale supercomputers will run on an exascale supercomputer in the future. A thousand-fold increase in peak performance rarely translates to an identical increase in application performance. Accurately extrapolating both performance and power from one extreme-scale system to another is virtually impossible. However, the ability to predict performance and power is important because it helps system architects determine how to address various design tradeoffs, while computational scientists decide which supercomputer to port their applications to. The goal of performance/power modeling is to gain understanding of an extreme-scale system's performance/power on various applications by measurement and analysis, then to encapsulate these characteristics in a compact and possibly executable formulation. The resulting model can be used to gain greater understanding of the performance/power phenomena involved and to predict performance/power to other extreme-scale system/application combinations. This section addresses modeling methodology, impact, and gaps, primarily focusing on performance and power. We believe that performance/power models support not only prediction but also insight and interpretation, and they are invaluable tools for performance and power analysis and tradeoffs between the two.

4.1.1 Current Modeling Landscape

Most modeling methodologies focus on performance modeling, while fewer methodologies focus on power and energy consumption modeling. This subsection describes current performance/power modeling methodology.

Performance Modeling

Performance modeling is a methodology for accurately modeling large applications of ultra-scale systems at different stages in their life cycle, from early design through production use [1,2]. Models can be defined analytically, through code execution measurements (actual or simulated), source analysis, or a combination of these techniques. Modeling ease and accuracy is dependent on the architecture characteristics (e.g., heterogeneity, multilevel parallelism, shared resources, or speculative execution) and the type and granularity of measurements provided by the architecture and software stack. Table 1 depicts some of the currently available modeling approaches and a simplified view of the tradeoffs in terms of portability, maintainability, and other attributes.

Table 1. Modeling Approaches Summary

Metric/Model Type	Analytical/ Expert	Source Analysis	Regression, Machine Learning	SW Simulation (DES, MP, PN)	HW Simulation
Portability (architecture)	high	medium	low	low	low
Scalability (size, complexity of codes)	low	medium	low	low	low
Scalability (model eval. time)	high	medium	high	low	low
Ease of Model Creation	low	medium	low	low	medium
Maintainability	low	high	low	low	medium
Transferability	low	high	high	high	high

The following are some of the existing modeling approaches used in high-performance scientific computing:

Analytical or Quasi-Analytical Methods

With this method, models are constructed from capturing analytically (i.e., in a computer program, a set of mathematical formulas, or statistically) the salient performance feature of an application code running on a system. Examples of analytical methods and tools include the suite developed by Performance and Architecture Lab (PAL, first at Los Alamos, currently at PNNL) [3,4,5,6], ASPEN (Abstract Scalable Performance Engineering Notation, developed at Oak Ridge National Laboratory) [7], PMaC (University of California San Diego/San Diego Supercomputer Center), and APEX-MAP. As illustrated in Table 1, this variety of methodologies allows a different degree of insight depending on the level of information captured by the model. For example, a "model" based on curve fitting could lead to understanding various performance trends for a system, but it offers limited insight as no

fundamental performance issue is captured with this method. In this category, model generation is done using a variety of means, although a main thrust of current research involves semiautomatic model generation. Analytical models are parameterized in terms of app and architecture "knobs." Here, the spectrum of possible model accuracy depends on the model's specific use, ranging from highly accurate (5% average for the PAL models) to coarse level—basically, evaluation of global metrics, such as locality or concurrency.

Trace-based Modeling

Trace-based simulation techniques have found widespread use as a means of avoiding the computational burden of cycle-level multicore architecture simulation. Traces of application behaviors are generated and used to drive simulation models at various levels of fidelity. Events that are traced may range from low-level, such as instruction execution, memory accesses, and packet injections, to higher-level events, such as runtime activities, e.g., message send/receive, interactions with accelerators, and disk accesses. Typically, a trace is a partial order of temporal events that reflect one feasible sequence of events in an applications execution. However, rigorous statistical formulations of trace-gathering methodologies can support a robust analysis of system behaviors of interest through appropriate sampling of application behaviors. Once time is invested in acquiring traces, the simulations themselves can be orders of magnitude faster. The hierarchical organization of traces can further improve the speed of simulation, as well as extend the scope of experiments that are feasible with a given set of traces. Acquiring traces also can be a challenge, necessitating access to hardware instrumentation or detailed (and time-consuming) validated cycle-level models.

Queuing Models

For decades, Markov chains, queuing networks, and Petri nets have been used for computer systems performance and reliability and are representatives of state-space modeling approaches. A Markov chain consists of both a set of states and a set of labeled transitions between states representing the system being studied. While these approaches are popular in academia and industry (e.g., in network modeling), these approaches have not been applied extensively to high-performance parallel scientific codes. Because evaluation of large state-space models is expensive, these approaches are not practical in the context of DOE exascale systems and applications.

Roofline Models

The roofline model can be used to bound performance or time-to-solution through examination of potential data movement bottlenecks. These bottlenecks can include dynamic random-access memory (DRAM) and cache bandwidths, as well as floating point or integer throughputs. Typically, empirical benchmarks are used to parameterize a processor's microarchitectural characteristics. Conversely, manual analysis typically is used to characterize a kernel's locality and parallelism. Although current roofline formalism models performance, it potentially could be used to model energy as well. Roofline-based performance modeling is not designed to accurately predict the performance of a particular software implementation running on a specific architecture. Rather, it is used to provide a performance bound based on idealized implementations of architecture and software. Thus, it may be used as a guide for assessing the quality of either and the need for further optimization.

Power Modeling

Estimating power and energy consumption is critical for software/hardware co-design of future systems; however, obtaining the entire system's power consumption, or that specific to various subsystems such as the central processing unit (CPU) or memory is nontrivial. Currently, power modeling is approached in one of two ways: 1) circuit designers use detailed models to predict component design power under well-defined workloads, or 2) facilities managers use "worst-case" steady-state models to adequately scale up power delivery resources for large-scale systems. Both approaches are inadequate for understanding and predicting workload-dependent power consumption characteristics of dynamic scientific workloads. Some work has been conducted using models to optimize the power consumption of applications [8] and communication libraries [9]. However, this isolated work is not well integrated with other performance modeling efforts.

The lack of fine-grained and high-fidelity power measurement tools also hinders development of application-specific power models that parallel the capabilities of well-established performance modeling capabilities. The current state of the art in power measurement capability is a high-frequency measurement capability at small system scales and/or a coarse-grained measurement at large scales. At the small scale, power measurements are obtained by intrusively inserting power measurement probes at key points on the power delivery network specific to the main board and reading the outputs of these sensors on high-fidelity data acquisition systems. Not only does this methodology not scale to large systems, but it requires expertise and physical access to the system under consideration. The advantage of such an approach is that with sufficient measurement frequency, it is possible to develop an understanding of the correlation between software features and power consumption characteristics. With newer processor architectures, such as the Intel® Sandy Bridge processor, power counters are beginning to be included on the processor, analogous to hardware performance counters that have been included in the past. While these are proving useful, it should be noted that no hardware counters are currently available for direct measurement of power consumption for components within the node that are not also part of the processor cores themselves.

For large system scales, different approaches are required. The current state of the art is to obtain outlet-level power measurements using external sensors that are placed between the power receptacles and the nodes that compose the systems. While this scenario affords large-scale power readings across a cluster, two drawbacks limit its usefulness. First, power readings are at a node (or multi-node) level, so gaining an understanding of how components within the nodes use power is not directly possible. Second, such external sensors have a low sampling frequency, limiting the ability to correlate power measurements with software characteristics.

Resilience Modeling

In designing HPC systems, there are several reasons why failures are becoming an increasing concern. Notably, the feature sizes of electronics are growing so small that wires are being separated by a mere few atoms. Second, the increasing popularity of power-saving techniques based on lower device operating voltages means that the signals within processors can become easily corrupted by a range of ambient physical phenomena. Third, as the total number of transistors in an HPC system's processors, memories, and networks increases exponentially, the probability that any one of them will fail rises. This makes it imperative to understand how various types of failures affect both hardware and software.

Systems may be affected by different types of failures. Fail-stop faults remove some component(s) from operation without warning and do not corrupt computational results. In contrast, silent data corruptions cause invalid results to be produced but otherwise allow execution to proceed unimpeded. This can cause the application to produce invalid output to computational scientists. In addition, failures may occur uniformly throughout time and space or may be correlated; for example, a hot processor has a higher probability of corrupting data, and the failure of a power supply causes the fail-stop failures of multiple nodes.

Multiple approaches are available to quantify the vulnerability of computing systems to failures. Fail-stop faults are analyzed by modeling them as being completely independent and measuring the mean time between faults (MTBF) of any given component. The MTBFs of individual components can be combined to compute the entire system's MTBF, which is used to configure various resilience techniques, such as setting the time between application checkpoints to minimize total execution time. Unfortunately, it has been shown experimentally that failures are not independent and, in fact, follow a Weibull distribution. This causes configurations based on the naïve independent failure assumption to be suboptimal, a fact that becomes increasingly critical as more components fail and the costs of checkpointing and global synchronization rise. Naïve MTBF-based models also are not useful for analyzing the behavior of various resilience algorithms, such as checkpoint coordination and resilience network protocols, which require much more expensive simulations that account for the system details and protocol.

Silent data corruptions present a more formidable challenge because it is difficult to even determine whether the application state is valid or corrupt. The traditional approach to analyzing application vulnerability to such faults is fault injection, where a large number of representative corruptions are injected into the application, and their effects on the outputs of individual modules or the application as a whole are observed. The primary difficulty for fault injection methodology is that while data corruptions are caused by low-level physical phenomena (e.g., neutron strikes or current instability), the prohibitive expense of modeling these phenomena directly means these events are simulated at a high level as bit-flips in gate-level (e.g., latches), microarchitectural (e.g., reorder buffer), or architectural (e.g., registers) states. The gain in performance comes at a high cost in accuracy. Although various individual errors introduced by this high-level abstraction have been documented, to date there are no studies that quantify the overall error introduced. Furthermore, the many different skill sets required to perform such an analysis means that it will not happen without dedicated investments in this area. Finally, despite the increase in efficiency due to high-level fault injection, a large number of faults (thousands or more) need to be injected to get sufficient accuracy. This limits the applicability of fault injection to a few small-scale applications on a few restricted inputs. Thus, techniques must efficiently apply fault injection to full-scale applications, such as developing modular injection methodologies where fault injection studies on many individual components can be combined to predict the vulnerability of the applications that include them.

4.1.2 Impact of Modeling

Performance modeling has an impact in all phases of application and system life cycle, from enabling a rapid a quantitative exploration of a large design space, to system procurement, to the maintenance and upgrade of both systems and large-scale applications. Performance models allow researchers to quickly answer "what if" types of questions, test hypotheses that would require significant investments to prototype, and quickly quantify the impact of new technologies

and architectures at the system level. Given a validated performance model, the impact in terms of new system design performance can be tested in advance of implementation. This is a critical capability, especially in addressing large-scale system design in which full-scale system implementations are not possible prior to deployment. As we move toward exascale systems, the cost in terms of hardware and facilities will prohibit full-scale prototype implementations. Models provide the tools to explore the performance of expensive systems while affording a detailed quantitative analysis.

Models are equally useful in exploring the application design space. Large-scale codes are themselves complex examples of software engineering, and implementing substantive changes often requires significant effort and expense. Models are able to provide crucial information regarding the value of these changes in advance of prototypes and implementation, potentially yielding a large benefit.

4.1.3 Gaps in Modeling State of the Art

The current state of the art in modeling methodologies was developed in response to the architectures and application structures typically found in today's HPC landscape. For some time now, the complexities inherent in supercomputer architectures have meant that increases in peak performance do not translate directly into improvements in application performance. Modeling has served as an ideal tool for quantifying the performance impact expected from changes in application and system architectures. However, as we move toward exascale, many of the assumptions built into these methods will no longer hold true. Models and modeling methodologies will need to adapt to an increasingly complex landscape. Furthermore, many of today's modeling methods were developed to focus on traditional processor performance, not to support new technologies like entirely novel memory systems, high-throughput processors (such as graphics processing units or GPUs), or advanced network interfaces with support for global address space languages.

Models are critical tools in application/system co-design. As applications and systems evolve simultaneously, models must be able to track ongoing complex changes and predict the impact of developments in both software and hardware design. While today's methods tend to focus on application performance as the metric of concern, modeling methods must evolve to consider performance, power consumption, and reliability in concert. While this represents a significant step forward over today's techniques, it is a requirement for future large-scale design as power becomes a first-class concern, and system complexity continues to increase.

As mentioned, models are ideal tools for navigating a complex design space because they allow for rapid evaluation and exploration of detailed "what if" questions. Nonetheless, the effort required to create a model often is substantial and requires close collaboration of both modeling and domain experts. As the complexity of application and target systems grows, this modeling effort may become prohibitive. A vital component of model research involves lowering the barrier to entry and easing the burden associated with model development. It is critical to develop tools and techniques that will allow modeling capability to spread into the larger computational science community, where it will have the greatest possible impact.

Finally, models today are static in their representation and often are used in a predictive manner to quantify the impact of design decisions prior to implementation. A key characteristic of future large-scale systems is adaptivity, or dynamic execution, and in this area, today's models are of little benefit. Models must evaluate application behavior within a dynamic execution

environment, ideally guiding and optimizing ongoing execution. This is likely to require the model forms to change. Models will take direct measurements of ongoing execution as inputs and provide output not to human researchers but to underlying runtime software able to act on a model's recommendations. A detailed discussion of these and other gaps and challenges are discussed as critical technologies in Section 5.

4.2 Emulation and Simulation

4.2.1 Current Landscape

Current simulation tools operate over a diverse range of levels, methods, and scales. Most architectural simulators address a single component of a system (e.g., DRAMSim models only implement double data-rate [DDR] DRAM; SimpleScalar models only implement a single processor core), although some simulators will model an entire system (e.g., BigSim; Structural Simulation Toolkit (SST)/Macro). In addition, an architectural simulator also may provide some modeling of a system's physical characteristics, such as temperature (Hotspot), reliability (RAMP), or power (McPAT) [24-26]

Simulation tools also use a range of methodologies. Device- and circuit-level simulations are the most detailed, but they are slow and seldom used for architectural exploration. It is more common to abstract away the low-level device characteristics and focus on the higher-level subcomponents or behaviors of a system component. These more abstract structural or behavioral models can blur the lines between simulation and modeling by using statistical behavior or queuing theory models. Depending on the simulation method, simulators can explore a single transistor to a multi-node system. Simulators themselves may be realized as software artifacts (the most common) or as hardware implementations, most commonly using field-programmable gate array (FPGA) technology [27-29].

Often, simulators will use functional emulators as front ends to provide an execute-at-fetch simulation model. This provides for full-system (including the operating system) simulation capabilities, as well as execution of stock binaries. However, they are quite slow and do not scale well. Moving forward, a major challenge is development of emulation concepts that will permit software analysis of exascale systems in parallel with hardware development. A second challenge is being able to use such a capability as part of a comprehensive system-level co-design methodology.

Field-programmable Gate Array/Hardware Accelerated

FPGAs are programmable hardware devices used to implement very fast emulators and simulators. FPGA-based emulators or prototypes are implementations of the target system model on FPGAs. This requires a complete hardware design of the target model that is typically realized via the implementation of a register-transfer level (RTL) design. Porting RTL intended for custom hardware to FPGAs can be challenging. For example, implementing a heavily multiported register file in an FPGA that consists of lookup tables, registers, and dual-ported static random-access memory (SRAM) can result in either an inefficient implementation that is slow and takes up a large amount of FPGA resources or a redesign of the target. In addition, emulators require full-target RTL. However, a working emulator is an especially fast and accurate way to estimate target performance and develop and run software.

An alternative approach to FPGA-based emulation is FPGA-accelerated simulation. Rather than a one-to-one mapping of the target model components onto an FPGA, an FPGA simulator

uses abstractions to accurately model the target—without all the costs. For example, if the floating point operation can be done more simply, a fully pipelined floating point unit might be modeled as a delay rather than requiring the floating point to be faithfully implemented on the FPGA.

Many FPGA simulators are partitioned into a functional model that implements target functionality, such as the ISA, and a timing model that models performance, power, etc. Some FPGA-based simulators are timing directed, where the timing model directs components of the functional model to execute at specific times. These simulators are very accurate, but they incur significant communication between the functional model and the timing model, which requires that they be implemented on the same FPGA. Other simulators are functional first, meaning the functional model executes first without timing information. However, such simulators are inherently inaccurate. Other FPGA simulators are speculative functional first, enabling significant decoupling between the functional and timing models but providing the capability for the timing model to correct the functional model, resulting in a highly accurate, easier-to-implement simulator.

4.2.2 Frameworks

Frameworks for simulation and emulation are prevalent in the broader community (wireless networks, material science, etc.). However, this is a relatively recent trend in the HPC simulation community. ModSim has been dominated by mature but isolated point tools specific to subsystems, such as cores, caches, interconnection networks, and I/O. Integration of these point tools for system-level modeling has been difficult and time consuming and has inspired the need for frameworks that can scale to large system sizes while integrating models of all subsystem components.

In general, frameworks seek to model a complete system with well-defined infrastructure components and interfaces. The desire to accommodate third-party tools and new functionalities and to couple applications and architectures are common themes in their development. However, pragmatic considerations require the ability to make specific tradeoffs between fidelity and scale.

BigSim is a simulation and emulation framework that emphasizes fidelity and scale at the application end of the spectrum. The execution of large application codes on hundreds of thousands of cores can be emulated on systems with tens of thousands of cores. The software emulation can trace specific system behaviors, such as network packet communication, that drive the BigSim simulator. The simulator is supported by post-mortem analysis and predictive models that enable exploration of a wider architectural design space. System behaviors, e.g., at the packet level, are modeled rather than cycle-level behaviors. A similar approach is taken by the SST/Macro simulation framework. High-level application descriptions are used to drive large-scale network models. A node of a large-scale parallel application is modeled with a skeletal representation of the application—execution delays interspersed with communication events. The network uses flow-based models to capture the cumulative behavior of these communication events and generate a model of internode communication traffic and interactions. Such models have been shown to scale to simulations of up to a million nodes.

The SST and Manifold frameworks represent approaches that support application-architecture full-system simulation at high fidelity. First, common simulation services, such as timing, event management, and synchronization, are extracted and implemented in a parallel simulation kernel. The target system is modeled as a set of components that

communicate across well-defined event interfaces. Components can be invoked by events from other components or may be advanced in a time-stepped fashion by a simulation kernel. In all other respects, the components are independent. Simulation frameworks encourage sharing and reuse of existing components. The system model can be partitioned across multiple logical processes, each containing a simulation kernel that controls the local components. All simulation kernels coordinate the global advancement of simulated time. This design decouples the component implementation from the knowledge of parallel simulation, enhancing the model's scalability and portability. Individual components may be simple timing models, cycle-level timing models, or emulated full-system models coupled to cycle-level timing models of cores, networks, memory hierarchy, or I/O components. The challenge with these frameworks is the ability to scale the models to a large number of nodes/cores when the associated timing models are operating at the cycle level. This leads to novel virtual time synchronization algorithms that attempt to use model-specific information to trade simulation accuracy for simulation speed in a controllable manner.

4.2.3 Low-Level Component Simulators

Low-level simulation is required for detailed implementation, testing, and debugging of an architecture. Unlike high-level architectural simulation, low-level simulation is focused on producing a specific design in great detail, allowing implementation as an application-specific integrated circuit (ASIC) or FPGA. The ModelSim tool, which is commonly used for FPGA or ASIC design and verification, is one example of a low-level simulator. ModelSim can perform gate- or register-transfer-level simulation of components written in a hardware description language, such as Verilog, VHDL, SystemC, or SystemVerilog. The simulation output usually is a waveform, mimicking an oscilloscope output, which allows the user to isolate individual signals and memory values. Tools such as ModelSim are critical for the integrated circuit design cycle, but are often complicated and difficult to use for wide-ranging design space exploration.

Other examples of component simulators operating at the microarchitecture level include those for memory (e.g. DRAMSIM2), interconnection network (Garnet, IRIS), processor cores and cache hierarchy (GEM5), and disks (disksim). Typically, instances of these components are available at multiple levels of fidelity, but they are not easily integrated because they employ individual mechanisms for time management and event handling. They are not necessarily designed for ease of integration into system simulators or frameworks. In the area of end-to-end storage systems, the CodeProject (http://www.codeproject.com/) has developed a collection of models focused on the exploration and co-design of exascale storage systems by providing a detailed, accurate, and highly parallel simulation toolkit for exascale storage that leverages the ROSS massively parallel simulation engine. With this new toolkit, design options and tradeoffs are being investigated with the goal of improving the reliability and performance at scale of potential exascale storage architectures. To date, an end-to-end, application-level storage simulation of a massively parallel storage system augmented with burst buffers in the form of solid-state storage has demonstrated the performance and cost savings potential of deploying burst-buffer storage in massively parallel storage systems. In addition, packet-level, multimillion-node network topologies (torus and dragonfly) have been simulated using Blue Gene/L, /P, and /Q supercomputer systems with up to 65,536 cores/message-passing interface ranks.

4.2.4 Physical/Technology

System designers want to explore a wide design space with various system components, rapidly evaluate the feasibility of emerging technologies, and achieve cross-layer optimization. Effective exploration requires understanding not only the system's performance (time-to-solution), but also the physical parameters of power, energy, reliability, and area/cost. Detailed circuit-level and physics simulations (e.g., SPICE and other electronic design automation, or EDA, tools) can be leveraged to extract the necessary parameters for system-level simulations. These tools are accurate, but they incur laborious design efforts and long simulation times. Estimating the impact of future technologies is another challenge. Existing tools do not support new technologies, but system designers still need early evaluation of emerging technologies.

Analytical modeling tools play an important role in both research and the early design phase. These tools are reasonably accurate (recent tools have 10–15% error compared to EDA tools), make it easy to compare different design parameters and styles, and fast. Hence, a wide design space can be explored within a practical time, even with not-yet-available technologies. For example, Cacti [10] estimates power, area, and timing of memory structures for computer architecture research. It includes transistor, various cache, and wire models, as well as optimization features. Initially, Cacti supported SRAM and DRAM caches. Recent releases include advances in microarchitectural techniques, as well as emerging technologies: power gating in large caches [11], three-dimensional-stacked memory with TSV (through silicon via) [12], off-chip I/O [13], and emerging nonvolatile memory [14,15].

McPAT [16] is an integrated power, area, and timing modeling framework for chip multiprocessors. McPAT models a multicore processor: components in cores (e.g., an instruction fetch unit, a load-store unit, an execution unit, an out-of-order issue/dispatch unit, etc.), on-chip routers and wires, on-chip caches, memory controllers, clocking, etc. With McPAT, designers can easily explore a wide design space of many-core architectures, finding an optimum configuration for a given objective (e.g., minimum energy-delay product or energy-delay-area product).

Other analytical tools are available for different purposes: Orion [17] for modeling power in on-chip networks, HotSpot [18] for thermal modeling, VARIUS [19] for process variation, and VARIUS-NTV [20] for process variation in near-threshold voltage computing. Recently, tools have been developed for integrated modeling of multiple physical phenomena [21].

4.2.5 Impact of Simulation

Simulation and emulation tools serve several constituencies. The first is the pool of application and algorithm developers who will rely on these tools for exploring and developing new algorithmic solutions and subsequently tuning these solutions for specific target architecture configurations and characteristics. This community emphasizes the ability to accurately sweep large machine parameter spaces and explore subtle—but performance and efficiency critical—interactions between cores, the interconnection network, memory systems, and I/O devices. The second constituency includes system researchers from industry, academia, and government. This community emphasizes high fidelity in key subsystems and modeling the interactions between subsystems, and pushes the envelope in terms of scale and detail. Their primary goal is knowledge generation to drive future development and influence research and development

investments. The third constituency is the educators who will use this infrastructure in the classroom in both undergraduate and graduate education. This community emphasizes visualization, clarity, ease of use, and access and display of pedagogically significant behaviors, such as synchronization behaviors and performance consequences of architectural decisions. Rather than pushing the technological boundaries, they are interested in illuminating the state of the practice. Finally, a fourth constituency is the procurement professionals whose acquisition and procurement strategies are guided by the insights provided by the simulation and emulation tools. The primary goal is to guide appropriate investments and specification of system characteristics by using accurate projection of technology trends. Across these constituencies, there may be many specific-use cases, including:

- architectural design studies involving broad system parameters, such as network topology and computational density versus power, performance, and cost
- component implementation studies, such as network routing, cache implementation, processor-in-memory, and processor-in-network
- assessing the impact of novel technologies, such as optics, stacked memory, and nonvolatile random-access memory, or NVRAM
- exploration of optimal scheduling, mapping, and load balancing of large parallel applications on varying architectures
- refactoring applications to use different programming models
- exploiting malleable component models to explore software/hardware co-design alternatives
- using instrumented models that provide richer observability than hardware or software profiling for application characterization
- estimating the impact of components used in different market segments, e.g., using mobile processors and memory in HPC and server domains
- hardware/software co-design
- development of physical environment models and their use in the exploration of tradeoffs between power, energy, performance, and resilience
- exploration of power management techniques, especially comanagement of power between hardware and software (application and/or runtime).

4.2.6 Gaps in the Simulation State of the Art

Many of the anticipated gaps in emulation and simulation are byproducts of the tradeoffs typically made between fidelity and scale, and ease of implementation. Gaps include:

- Interoperability: There are few standardized techniques for integrating point tools and models into cohesive system models. Consequently, there is little leveraging of significant prior investments.

- Physical Environment: Coupled models of time-thermal-energy-reliability are critical as the interactions between various physical phenomena and microarchitecture become visible and can no longer be masked without major sacrifices in performance.
- Scale: The execution performance of the ModSim infrastructures themselves will have to exhibit, at a minimum, weak scaling with model complexity to model systems of exascale complexity.
- Co-design: Modeling and simulation tools and methodologies generally do not support system-level co-design, yet they must to support exascale systems design. Much can be learned from the embedded systems community in this regard.
- Interface-to-Design Flows: Investments in models and simulators currently stand alone with little support for interacting with commercial modeling and design flows. Better interfaces would enable valuable feedback and refinement, as well as provide an opportunity for investments in tools to flow to hardware and software designs.

4.3 Overarching Issues

4.3.1 Validation and Verification

Uncertainty in modeling, simulation, and emulation for HPC can arise from a variety of sources. Often, to protect industry advantage, implementation of a component is completely unknown to the research community, and model construction either abstracts its characteristics or approximates the behavior. Also, experimental data used for validation may contain some amount of "noise" from the operating system, environment, or other jobs running on a shared machine. In addition, the conglomeration of component models into larger simulations introduces uncertainty through their interfaces and interaction.

Formal uncertainty quantification (UQ) methods provide a means for dealing with uncertain parameters and data that feed into the models used in simulation and emulation. Forward uncertainty propagation estimates the amount of error produced from a simulation given the uncertainty inherent in its parameters, which is useful for gauging the accuracy of a simulation's output, especially in predictive experiments. Bias correction and parameter calibration are validation time methods that attempt to correct imperfect models based on experimental data, improving the simulation's accuracy. However, these UQ techniques seldom are used in computer science studies.

4.3.2 Data Repository

To support modeling and simulation, a repository of measured performance data will be one key to success. This data could be used to assist in model building and validate/verify the model built.

The performance data will be obtained through various methods in the system, both from hardware and software. For example, hardware counters provide information about the events occurring in the hardware, such as the number of cache misses. Application instrumentation provides information about application behavior, such as the communication pattern.

Two major issues arise when working with performance data: 1) collecting the performance

data and 2) organizing/storing the performance data. Various tools and methods already exist to collect performance data from various sources. As the scale of the system increases, the size and variety of the performance data will be overwhelming. Minimizing overhead so that only "interesting parts" of the performance data are collected/filtered and transferred will be the challenge.

A standardized mechanism will be needed to efficiently organize, store, and retrieve the data.

4.3.3 Interoperability

For the most part, the wealth of simulators and simulation components exist in isolation from each other. Many simulators focus on a single component of the system (e.g., memory) and may not connect to simulators that model the rest of the computer. Although most performance simulators are based around a discrete-event model, there are no commonly accepted standards to provide interoperability between simulators.

While it is technically feasible to construct common interfaces for simulation components, simulator writers have not created such an interface due to nontechnical concerns, such as the lack of long-term support for this type of infrastructure, fear of added complexity, and even a "not-invented-here" syndrome. Without adequate input and a widespread base for a common interface, it is seen as an unwanted imposition, such as the way the U.S. Department of Defense mandate to use the Ada programming language was viewed by many programmers. However, if an appropriate interface was agreed upon, it could have a positive transformative effect on the simulation community, allowing greater reuse of components. There may be significant lessons to be learned from the embedded and EDA ecosystem, where standardized file formats greatly ease interoperability of many design tools.

4.3.4 Multi-scale

ModSim spans a range of time scales, from the sub-nanosecond for device simulation to days or months for workload scheduling simulation, and levels of detail, from the RTL level to highly abstract models. Simulation experiments must confront the tradeoff between detailed and precise models that are very slow and more abstract fast models that may have lower precision.

A key research area is the ability to create mixed-resolution or multi-scale models that combine detailed and abstracted models in the same simulation. Ideally, this would allow experiments that use detailed models for the most critical portions of a system but rely on more abstract models for peripheral portions.

Multi-scale modeling has some engineering challenges, but it also faces significant methodological challenges. Large-scale computer systems have complex feedback paths. It is unclear whether combining detailed and abstract models will ignore some of those feedback paths and miss critical effects. For example, a detailed network model connected to a simplified memory model may miss the contention between network traffic and processor memory requests in the memory system. Without care and research, a multi-scale model may end up providing the "worst of both worlds"—low precision and slow speed—rather than the "best" (high precision and speed).

4.3.5 Software Engineering

While innovative technical approaches and methodologies often are the focus of simulation and emulation efforts, simulation and emulation tools represent significant software engineering efforts in their own right. To make tools and technologies accessible to the user community, a substantial investment must be made in the development and maintenance of software that embodies state-of-the-art techniques. Software development tools and processes, such as version control systems, regression testing, code reviewing, and continuous integration, must be instantiated and promulgated by dedicated staff because many community members do not have a software development background. Furthermore, new methodologies developed as research projects often are prototyped in one-off tools that are quickly abandoned. Additional effort is required to incorporate new technologies into standard, stable, well-maintained community tools. Because architectural exploration must build from previous work, it is critical that this maintenance be a long-term, and not a one-off, integration effort.

Unlike other software tools, such as compilers or profilers, typical use cases often involve "opening up" the source code and adding instrumentation, prototyping features such as new CPU instructions, and extending or developing component models. Thus, users are exposed to both internal and external interfaces and behaviors. This situation highlights the need for extensive documentation—not only of each tool's superficial features and options, but also its internal structure and interfaces, organizing principles, and any assumptions underlying its models and methods.

Open-source tools are ideal for fostering adoption and cooperation and lowering barriers to entry. However, given the range of academic, government, and industry participants in this sphere, licensing and intellectual property protection options should impose the fewest mandatory restrictions while remaining flexible to encompass the widest range of usage models. For example, core software should be released under a non-viral (e.g., BSD-like) license to avoid constraints on industry usage, but frameworks should allow protection and controlled sharing of proprietary models either via legal means, such as nondisclosure agreements, and/or technical means, such as encryption.

4.3.6 Scalability

Ironically, architectural exploration for highly parallel, multicore machines usually is performed with serial simulation tools. There are some notable exceptions (e.g., BigSim or SST). However, even these simulators tend to parallelize at the simulated node level and leave the internals of a node as a serial simulation. Some of the key issues in parallel architectural simulation include:

- Component parallelization has been hampered by the tight coupling and frequent communication between components on a node (e.g., frequent low-latency cache coherency traffic between processor cores). Strategies to decompose the system more effectively are needed.
- Determining the mechanics of parallel simulation presents many options. Most architectural simulators use discrete-event simulation (DES), and there are many parallelization strategies for DES. Using rollback, different communication strategies, and different look-ahead strategies have been tried, but there is little consensus on which is best.

- Determining where time relaxation is possible is an open area. It is possible that some models (e.g., thermal modeling) may not lose much accuracy if they run slightly out of sync with the behavioral model.
- The effect of multi-resolution modeling on scaling and parallelization is unknown. However, this problem has been addressed for a variety of multi-scale physical simulations (e.g., materials codes), and the architectural simulation community should be able to learn from those experiences.
- Recasting parallel simulation as a data parallel execution model and using new accelerator technologies, such as GPUs or vector extensions is another option. The ability to harness these new devices will require innovative solutions to architecture modeling.

4.3.7 Machine Benchmarking and Data Capture

Because each ModSim component requires a distinct, but potentially similar, set of input parameters to drive the modeling process, it is difficult (if not impossible) for hardware architectures and system vendors to adequately provide all the required information. In many cases, the data needed for these models has a degree of similarity, although not identicalness, to other models or can be sufficiently synthesized from base metrics that can reduce the burden on vendors or system benchmarks. Discussions with vendors and computer architecture researchers indicate that much of this data is routinely stored as part of the ongoing assessment of machines or hardware prototypes. By forming a more standard list of metrics and a standard approach for their benchmarking, ModSim components can enjoy a greater degree of portability. Although not all data can be captured to sufficiently drive every model, capturing at least a sufficient subset of data could alleviate some of the pressure associated with benchmarking activities and make performance data more broadly available to the research community.

Another potential research gap is the lack of a common set of micro-benchmarks that are accepted by the research community as providing accurate information. If, as suggested herein, an interoperable, common view of metrics is taken, an important research question arises as to how these metrics will be taken from machines. These metrics will need to be available on a variety of architectures and not be biased toward a single solution. Suggestions to address this concern involve using existing established benchmark suites developed in previous programs as a basis for further development with the possibility of adding further micro-benchmarks to obtain metrics relevant specifically to simulators and analytic models.

4.3.8 Best Practice Validation Processes

Currently, validation processes performed during the development of performance models are highly varied, leading to a lack of trust in reported results. In part, this is driven by a lack of clarity in understanding how a successful validation activity may be conducted. Simply put, even if simulators and performance models are highly successful in mapping application performance trends toward exascale, it is vital that the community not only believes the reported results, but also trusts the models' abilities to track future trends. Therefore, a gap exists regarding how performance model validation should be conducted and how it may be performed for single-component studies and those involving multiple components, such as integrated

simulators, where each component may have a non-consistent error.

4.4 Associated Technologies

4.4.1 Proxy Applications

At the core of the Co-Design Center's activities is a loop in which application codes, represented by proxy applications ("apps") that explore the breadth of algorithm space (including programming models and other implementation choices) are co-optimized with the available architecture designs concerning price, power, performance, and resilience within externally imposed constraints. The proxy apps are a condensation of the "real" apps that capture the broader workflow, but are built so that strategies such as data layout and solution algorithms can be explored, and overlay aspects such as power management strategies.

That said, the generation of proxy apps, their maintenance, and their validation is nontrivial, yet vital to the success of the co-design process. Specific challenges exist. First, how easily can proxy apps be created from larger applications that represent the specific metrics of interest with high fidelity? Second, how can proxy apps for future algorithms and apps that may not yet have a large-scale application (from which to be derived) be generated? Ideally, a user could employ the model to drive the design of the eventual application. Third, how can abstractions in the proxy apps capture the language and runtime complexity of the real applications? Finally, how can these proxy apps be created in a way that exploits specific architectural features across a diverse set of possible architectures yet remain portable, valid, and useful?

4.4.2 Measurement Tools

Many versatile performance measurement tools currently are available, including both open-source (PAPI, TAU, HPCToolkit, Scalasca, Pin, OpenSpeedshop) and proprietary (Cray's CrayPat, IBM's HPM, Intel's VTune) software. While this variety is beneficial for addressing different measurement needs, such as sampling- and instrumentation-based approaches, profile and trace generation, the differences among tool interfaces and the data formats they produce make their use challenging. Some standards for both profile and trace data are emerging, but they are not uniformly supported by tools. To enable the scalable and consistent collection of data required for performance analysis, modeling, and improvement of large-scale parallel codes, the following gaps must be addressed: 1) portable automation of the measurement process across architectures; 2) the ability to analyze and create models in a tool-independent manner (e.g., either more widespread use of profile and trace data standards by tools or tools for converting among representations); and 3) scalability of the data collection and analysis, especially for tracing.

4.4.3 Model Language Tools

Aside from simulation, scientists often rely on analytical modeling because it provides a flexible, fast estimate without the need for exact, low-level details of either the application source code or architectural details. Examples of these analytical models include LogP, BSP, and Roofline. Often, scientists are willing to accept the advantages of these models at the cost of accuracy (compared to cycle-accurate simulation). However, the community lacks a common, structured analytical modeling methodology that allows composable models to be created,

refined, exchanged, reused, and maintained for complex applications and architectures being designed by large teams. In addition, the ModSim community currently does not have a methodology that crystallizes these modeling approaches in a common, shareable toolkit for developing and using these analytical models.

A structured, artifact-driven approach to analytical modeling would have several advantages over traditional approaches. First, a common toolkit approach constrains models to fit a specification, which enforces similar concepts across models and allows for automatic correctness checks. Second, because models are written in a structure, the toolkit can provide formal rules for modularity, so models are easily composed, reused, and extended. Third, as with normal application development, this structured approach would promote collaboration between application and computer scientists by allowing them to share and refine models using a common set of concepts and tools. Some preliminary work in this area includes Python-based approaches and the Aspen domain-specific language (DSL).

4.4.4 Testbeds

Testbeds and development access to target platforms are critically important to simulation and modeling for many reasons, but ultimately they provide for verification and validation of ModSim results. However, access to architectural testbeds for development and testing continues to be a challenge. Most platforms are scattered throughout laboratory, academic, and vendor locations. Typically, they have higher costs in terms of initial procurement and ongoing maintenance. On another dimension, scalable testbeds are vitally important to verify and validate ModSim results. Moreover, given the diversity of possible architectures at these early stages, the community will need consistent and ongoing support for accessing, using, maintaining, and exploring these devices.

On another front, certain platform characteristics may be useful in accelerating simulators or other modeling tools. Clear examples include emulation via FPGAs. However, these specialized platforms are not widely available, and they can require extensive access to the hardware and system software, which is a user scenario that contrasts with traditional production computing within DOE.

5 Critical Technologies of Modeling and Simulation

The development of many critical ModSim technologies is required to assure success of future systems and applications that will function at exascale and beyond. As recognized by previous activities and workshops, there will be a step change in future systems that will put increased emphasis on software throughout the software stack that is needed to tackle issues of data locality, concurrency increases, power constraints, overcoming faults, and systems and applications adaptivity. In moving to exascale, this emphasis must address the change in the principal determinants of performance scaling. If successful investments are made in ModSim technologies now, they can be applied to explore systems and applications designs in advance of implementation and follow through to assisting in dynamic optimizations during system and application execution, providing a powerful environment that is designed and performance-engineered to achieve the highest level of science from the expected hardware and software available within the next decade.

The ModSim workshop identified several critical technologies to be addressed, including

methodology development, use, and value added to potential system and application outcomes.

5.1 Integrated ModSim for Energy, Performance, and Reliability

Both integrated modeling of multiple physical phenomena and the ability to capture the impact on the architecture and applications are significant challenges. There is a critical need to be able to: 1) model and understand the relationships between physical phenomena, such as temperature, energy, power, and reliability, and their impact on devices; 2) derive the consequent impact at the microarchitecture and system levels; and 3) determine how these can be controlled to maximize system-level performance and power attributes. The ability to model performance, power/energy, and reliability simultaneously is an important prerequisite to design and optimize adaptive systems and applications for optimal performance under power and reliability constraints. As the main target of the multi-objective optimization problem changes at exascale, an integrated approach is needed to answer that challenge.

5.2 Develop New Interoperable Machine and Application Abstractions for Co-Design

There is general consensus that the methodology and tools for capturing, exchanging, refining, and evaluating models of applications and architectures are insufficient for the co-design process. Application developers need new abstract machine models that expose expected hardware features (e.g., SIMD, lightweight cores, or specialized functional units) and performance targets in terms of hierarchical parallelism, scale, data movement, computational intensity, bandwidths, latencies, and storage capacities of possible exascale configurations. Meanwhile, system architects require architecture-independent models of important applications to identify, understand, and evaluate the resource requirements and sensitivities against their proposed architectures. These abstractions can exist across the modeling, simulation, and measurement infrastructures and serve as the common parlance for driving the co-design process forward and facilitating collaboration, sharing, and refinement. In applications, for example, these models may describe loop- and data-level parallelism for computation on subdomains, including relatively simple approaches for specifying fine-grained parallelism that can be translated into lightweight parallel implementations. These abstractions may have multiple resolutions, depending on their target use, and range from specific implementations to general descriptions (independent of programming models, languages, etc.).

5.3 Model Generation

The construction of performance models (e.g., manually, through machine learning approaches) is a labor-intensive process, typically requiring “expert” involvement. This situation precludes the widespread use of models, which must be easy to create and maintain without incurring significant development overhead or requiring scarce expertise. Once created, models are not exploited sufficiently because they tend to not evolve with changing implementations. The performance research community has decades of experience and approaches for manual and automated model creation, but, so far, no usable and scalable solution sufficiently addresses the community’s needs. While much expertise exists in specific application cases, models still typically are one-off exercises with limited long-term impact on the performance of the code being modeled.

5.4 Dynamic Modeling

ModSim techniques need to become dynamic in that their predictive and optimization capabilities must be available at runtime. This will require development of rapid evaluation techniques as developing interfaces for incorporating models into the software stack. Thus far, today's methods have been limited to exemplars and proof of concept. For example, there are many uses for dynamic modeling [22] facilitated by both the high accuracy and rapid execution potential of models. Another example is the use of the predicted time to decide when to checkpoint. That is, if the next cycle will exceed a predefined limit of accumulated time plus the optimal checkpoint interval, then a checkpoint should be done before the next cycle. Alternatively, the predicted time could be used to dynamically determine the number of cycles for a particular execution. Different methods for the generation of dynamic models need to be considered: for example, either being generated offline and incorporated into software that can be rapidly executed or being generated online using observations from prior executions. Dynamic models also can be used to verify the system's health (i.e., are we getting the best, expected performance at the highest energy efficiency?).

5.5 Model as an Actionable Tool

Intelligent decision making regarding the use of available resources within a large-scale system can be facilitated by using dynamic models. There is a need for models to be actionable—that is, to have the capability of reacting to the instantaneous state of the system, (both hardware and software), and provide the necessary information to guide and optimize system operation. Mechanisms for translating the knowledge and insights contained within a dynamic model are needed that will achieve high performance and energy efficiency, as well as the mitigation of faults during system execution via tools, policies, and application software optimization.

Multi-objective

Actionable models will need to encompass performance, power, and reliability model concerns and be able to advise runtime environments to optimize application operation. These activities must be both scalable and low in overhead. An accurate picture of system state must be created with minimal perturbation to both application performance and system power consumption. The fundamental tradeoff is accuracy against overhead. Such a tradeoff must be continually reevaluated during the execution of application tasks by comparing measured reality against modeled predictions.

Resiliency

Models that can assist in mitigating faults during system execution also must be developed. Such resiliency will be incorporated into the system hardware and execution environment and even affect the way applications are expressed. However, this will come at a cost, impacting both performance and power usage. Models can be used to explore the impact of resiliency techniques in an application-specific way and assist dynamically in mitigating the effects of faults when they occur.

Introspective Runtime Systems

Actionable models need to be developed that will assist in intelligent and informed decision making within the runtime system, enabling dynamic optimization of power consumption, performance, and reliability concerns regarding the available resources. Runtime software employed to manage execution at exascale can benefit from using quantitative and predictive models that capture behavioral knowledge to make intelligent decisions, e.g., when considering data movement, data replication, and computation migration, and will employ tools provided by the hardware to route power to where it can be most effectively used.

5.6 Integration of Methodologies

There is no one single ModSim approach capable of undertaking all in-demand predictive analyses. Rather, a set of tools is required where each component has specific and defined roles and usages. At the coarsest level, techniques can be separated into modeling, simulation, and emulation categories, and synergies between the combinations of approaches can be achieved. There is a critical need for development of an interface sufficient to enable simulation components or analytic models to be easily reused and integrated. In an ideal scenario, models of machine components or applications would be effortlessly reusable through a specified interface, allowing input parameters and output predictions to be shared. The benefit of such an approach is that components can be developed in isolation, aggressively tested, and validated, then reused by the community, reducing model development time while providing increased accuracy through method integration.

Multiple Methodologies

Multiple levels of abstractions can be applied within ModSim. These result in different granularities, representations, and evaluation methods and requirements (time-to-evaluation). For example, in dynamic modeling, rapid evaluation is required to inform and guide runtime optimization, requiring almost instantaneous evaluation, whereas DES may require large computational resources to simulate a limited time window of the execution of an exascale application on an exascale system. There is a need to establish suitable abstractions that can be used by and optimized with ModSim.

Common Interfaces

A set of interfaces must be developed that allows ModSim components to share a common infrastructure, including aspects such as time progression mechanisms (e.g., in discrete-event generation), parameter sharing, component querying for internal data, and interfaces to dynamic models. Through standardization and gradual refinements, a robust set of interfaces have allowed parallel applications considerable portability across successive generations of architectures. A similar approach to interfaces between simulation components has the potential to enable genuinely portable components between models, as well as component longevity, because the components can be updated and refined independently.

Component Integration

Development of a set of interfaces allowing for multiple methods to work in unison, share input parameters, and perform cross-model analysis and comparison is another necessity. Potentially, hybrid models will permit higher-fidelity models to be evaluated in shorter time

frames, where accurate analytic models can be queried in place of expensive simulation. Similarly, analytic models will benefit from the ability to query simulation components for predictions of complex behaviors, such as network congestion.

5.7 Life Cycle Coverage

There is a critical need for ModSim technologies to cover and be coherent across all life cycle phases, from design space exploration and analysis of early implementation to deployment and run-time optimizations, as well as to cover both hardware and software.

Design Space

Potentially, there is a large payoff if systems and applications are co-designed. Multiple boundaries exist with associated tradeoffs that can be captured via ModSim. The co-design space for exascale applications and systems includes boundaries such as system-to-application execution models, programming models for applications, architecture and runtime system, etc. They also need to consider performance, power, and reliability in unison. In addition, the tools must capture increased architectural complexity, which may include configurable caches; reduced cache coherence and relaxed memory consistency; a higher number of non-uniform memory architecture, or NUMA, domains; potentially novel threading models; and, above all, drastically increased heterogeneity through accelerators use. In addition, new interconnect technologies are expected. On the application side, ModSim can explore algorithmic designs and optimize them in the architectural space context. This process is most beneficial if undertaken in advance of implementations and used to guide future implementations.

Procurement

Modeling can be actively employed throughout the procurement process for exascale systems. This activity includes quantifying the impact of various technologies on application performance and culminates in ranking various potential designs and bids. From a performance standpoint, this type of process has been successfully followed in the procurement of many current petascale systems.

System Deployment

Modeling can be used as a diagnostic tool to examine the performance delivered by exascale systems, as well as for DOE applications of interest to see whether they are within acceptable bounds of the expected performance and, if not, to explain the reasons why.

Runtime Optimization

Dynamic models that are actionable at runtime can form the core of future intelligent introspective runtime systems that will monitor, mitigate and manage the system behavior for performance optimization within power and reliability envelopes. Modeling is an ideal vehicle for these activities given their rapid execution and accurate predictive qualities.

5.8 Standards, Integration, and Interoperability of ModSim Methodologies and Tools

One of the most pressing problems with state-of-the-art application performance ModSim is the interoperability of components and performance models. Along with component or model

interoperability, both forms of performance prediction are unable to integrate. In addition, there is a lack of an adequate, commonly accepted process for validating the predictive nature of models that port naturally to multiple simulation techniques or modeling approaches. This situation is driven, in part, by the specialist nature of specific simulators or approaches to model development that demand highly tailored processes. However, the effects of limited interoperability and opportunities on integration are nontrivial and include: significant increases in simulator development times as each component must have infrastructure developed independently; longer model development times because benchmarking and performance data cannot easily be reused between multiple model activities; and skepticism or lack of trust in many modeling activities because they cannot be easily compared, contrasted, or reproduced by other members of the computing community.

The issues outlined herein must be addressed if ModSim activities are to beneficially impact exascale computing. Early adoption of portable simulation infrastructure and analytic models will provide the greatest return value as many projects related to modeling for exascale machines are in the earliest stages of development and are ideal to engage with this research effort and benefit from its output. Interactions with vendors also indicate these issues are a concern within the industry, particularly in the context of co-design activities where data sharing between domain experts and computer architectures is vital. Without interoperable, best-practice-based, validated models, ModSim risks becoming an expensive process whose actual development may exceed a useful timeframe to beneficially impact exascale computing.

6 Summary of Critical Technologies

The workshop committee noted significant global progress in ModSim development, both in appropriate methodologies and in its use for a variety of design, analysis, optimization, and other activities. While this progress is real and significant, many important and substantial gaps were identified in the current state of the art, as well as in the areas where ModSim will be central toward blazing the path to exascale.

In summary, these observations focused on the following:

- development of ModSim capabilities that cover the entire spectrum of scales, from cycle accurate to global metrics
    - There is no one single modeling and simulation approach capable of undertaking all predictive analyses demanded of it. Rather, a set of tools is required where each component has specific and defined roles and usages. The benefit of this approach is that components can be developed in isolation, aggressively tested, validated, and then reused by the community, reducing model development time and affording increased accuracy through integration.
- need for standards interoperability of ModSim methodologies and tools
    - One of the most pressing problems with state-of-the-art application performance ModSim is the interoperability of components and performance models. Without interoperable, best-practice-based, validated models, ModSim risks becoming an expensive, time-prohibitive process with little or no impact on exascale computing.

- extension of ModSim to integrate power, reliability, and performance
    - The challenge involves both integrated modeling of multiple physical phenomena and the models' abilities to capture the impact on architecture and applications. As the primary target of the multi-objective optimization problem changes at exascale, an integrated approach is needed to answer that challenge.
- methodology development for the emerging dynamic modeling area
    - ModSim techniques need to become dynamic, i.e., their predictive and optimization capabilities must be available at runtime. This requires development of rapid evaluation techniques as developing interfaces for incorporating models into the layers within the software stack.
- development of modeling methods that allow the model to be actionable, for example, in guiding introspective runtimes
    - Intelligent decision making regarding the use of available resources within a large-scale system can be facilitated by dynamic models. There is a need for models to be actionable—to have the capability of reacting to the instantaneous state of the system and provide the necessary information to guide and optimize system operation.
- ModSim tools that can be used throughout the life cycle of systems and applications
    - There is a critical need for ModSim technologies to cover and be coherent across all phases of the life cycle and to address hardware and software.
- improved methods for model generation to accelerate use and dissemination of ModSim
    - No matter what methodology is employed, performance model construction is an effort-intensive process, typically requiring expert involvement. This situation precludes the widespread use of models, which must be easy to create and maintain without incurring significant development overhead or requiring scarce expertise.
- closer interaction with the exascale architecture, programming models, Co-Design Centers, and runtime communities
    - Closer interaction with these varied communities could strengthen the central role ModSim plays.

7 Recommendations and Path Forward

At the community level we decided to make every effort to unite as a like-minded group, and ensure steady, high-bandwidth, open dialogue on a continuous basis. The workshop's success set the course toward that goal. As part of our path forward, we recommend holding annual ModSim events. At Bill Harrod's (ASCR) suggestion, the intent is to focus each future workshop along the lines of the critical technologies for ModSim identified and described herein. The goals of

each event would be to foster in-depth discussions of the various critical technologies that could accelerate ModSim's progress; chart the path forward for the technical community for the upcoming year; synchronize activities across a myriad of projects; make sure that progress is on track; enable ASCR, as well as other funding agencies cosponsoring the event, an opportunity to assess progress and identify synergies within their own portfolios; and showcase important breakthroughs (as appropriate).

Given ModSim's criticality at this time, its increased role in exascale activities, and the interest from the multiple communities it serves, focused, new funding for the critical technologies identified is a significant recommendation resulting from this workshop. This can be best achieved through a specialized Request for Proposals, allowing ASCR to coordinate and motivate these future activities. The concern is that the status quo in the current funding model would lead to further fragmentation, inhibit efforts to solve the conspicuous gaps, and otherwise inadequately address the community's future needs in the ModSim arena.

8 Bibliography

[1] D. J. Kerbyson, A. Vishnu, K. J. Barker, A. Hoisie, "Co-design Challenges for Exascale Systems: Performance, Power, and Reliability", IEEE Computer 44(11):37-43, Nov. 2011.

[2] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, J. C. Sancho, "Using Performance Modeling to Design Large-Scale Systems", IEEE Computer 42(11): 42-49, Nov. 2009.

[3] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, J. C. Sancho, "Entering the Petaflop Era: The Architecture and Performance of Roadrunner", In Proc. IEEE/ACM SC'08, Austin, TX, Nov. 2008.

[4] D. J. Kerbyson, A. Hoisie, "Analysis of Wavefront Algorithms on Large-Scale Two-Level Heterogeneous Processing Systms", In Proc. Workshop on Unique Chips and Systems (UCAS2), IEEE Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, March 2006.

[5] D. J. Kerbyson, K. J. Barker, "Modeling the Performance of Direct Numerical Simulation on Parallel Systems", Parallel Processing Letters 21(3): 275-277, 2011.

[6] D. J. Kerbyson, P. W. Jones, "A Performance Model of the Parallel Ocean Program", International Journal of High Performance Parallel Computing Applications (IJHPCA) 19(3): 261-276, 2005.

[7] K. Spafford and J.S. Vetter, "Aspen: A Domain Specific Language for Performance Modeling," in SC12: ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis, 2012

[8] D. J. Kerbyson, A. Vishnu, K. J. Barker, "Energy Templates: Exploiting Application Information to Save Energy", IEEE Cluster, Sept. 2011.

[9] A. Vishnu, S. Song, A. Marquez, K. J. Barker, D. J. Kerbyson, P. Balaji, "Designing Energy Efficient Communication Runtime Systems: A View from PGAS Models", Special Issue on Green Computing and Communications, Journal of Supercomputing, 2011.

[10] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi. CACTI 6.0. Technical report, HP Labs., Apr. 2009.

[11] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. In Proc. the IEEE/ACM Int'l Conf. Computer-Aided Design (ICCAD), Nov. 2011.

[12] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory. In Proc. the Design

Automation and Test in Europe (DATE), Mar. 2012.

[13] N. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas. CACTI-IO: CACTI with off-chip power-area-timing models. In T. B. D.

[14] X. Dong, N. Jouppi, and Y. Xie. PCRAMSim: System-level performance, energy, and area modeling for phase-change RAM. In Proc. the Intl Conf. Computer-Aided Design (ICCAD), Nov. 2009.

[15] C. Xu, X. Dong, N. Jouppi, and Y. Xie. Design implications of memristor-based RRAM cross-point structures. In Proc. the Design, Automation, and Test in Europe (DATE), Mar. 2011.

[16] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In Proc. the 42nd Ann. IEEE/ACM Int'l Symp Microarchitecture (MICRO), Dec. 2009.

[17] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In Proc. the Design, Automation, and Test in Europe (DATE), Apr. 2009.

[18] W. Huang, M. Stan, K. Skadron, K. Sankaranarayana, and S. Ghosh. HotSpot: A compact thermal modeling method for CMOS VLSI systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 14(5):501–513, May 2006.

[19] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. IEEE Transactions on Semiconductor Manufacturing, 21(1):3–13, Feb. 2008.

[20] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torellas. VARIUS-NTV: A microarchitectural model to capture the increased sensitivity of many cores to process variation at near-threshold voltages. In Proc. the Intl Conf. Dependable Systems and Networks (DSN), Jun. 2012.

[21] W. J. Song, M. Cho, S. Yalamanchili, S. Mukhopadhyay, and A. F. Rodrigues "Energy Introspector: Simulation Infrastructure for Power, Temperature, and Reliability Modeling in Manycore Processors," SRC TECHCON, Sept. 2011.

[22] M. M. Mathis, D. J. Kerbyson, "Dynamic Performance Modeling of an Adaptive Mesh Application", In Proc. Workshop on System Management Tools for Large-Scale Parallel Systems (SMTPS) International Parallel and Distributed Processing Symposium (IPDPS), 2006.

[23] W. Harrod and S. R. Sachs. Introduction to the ASCR Exascale programming challenges work- shop. Presentation given at DOE 2011 Workshop on Exascale Programming Challenges, July 2011.

[24] W. J. Song, S. Mukhopadhyay, A. Rodrigues and S. Yalamanchili, “Instruction-Based Energy Estimation Methodology for Asymmetric Manycore Processor Simulations,” IEEE/ICST International Conference on Simulation Tools and Techniques, March 2012.

[25] M. Cho, W. Song, S. Yalamanchili, and S. Mukhopadhyay, “Modeling of the Thermal Field of Many-Core System using Frequency Domain System Identification,” SRC TECHCON, Sept. 2011.

[26] W. Song, S. Mukhopadhyay, and S. Yalamanchili, “Reliability Implications of Power and Thermal Constrained Operation of Asymmetric Multicore Processors,” Dark Silicon Workshop, June 2012.

[27] M. Cho, W. Song, S. Yalamanchili, and S. Mukhopadhyay, “Thermal System Identification (TSI): A Methodology for Post-silicon Characterization and Prediction of the Transient Thermal Field in Multicore Chips,” IEEE Symposium on Thermal Measurement, Modeling, and Management, March 2012.

[28] M. Cho, N. Sathe, M. Gupta, S. Kumar, S. Yalamanchili, and S. Mukhopadhyay, “Proactive Power Migration to Reduce Maximum Value and Spatiotemporal Non-uniformity of On-chip Temperature Distribution in Homogeneous Many-Core Processors,” Proceedings of the 26th IEEE Annual Thermal Measurement, Modeling and Management Symposium, 2010.

[29] C. Kersey, A. Rodrigues, and S. Yalamanchili, “A Universal Parallel Front-End for Execution-Driven Microarchitecture Simulation,” HIPEAC Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, January 2012.

Appendix. Workshop Agenda and Participant List

Workshop Agenda

Thursday, August 9th, 2012			
07:00-08:00 a.m.	Registration, Welcome	ASCR, Committee	Main Room
08:00-08:05 a.m.	Introduction to the Modeling and Simulation Workshop	Adolfy Hoisie	Main Room
08:05-08:20 a.m.	Extreme Scale Architectural Issues	William Harrod	Main Room
08:20-08:30 a.m.	Extreme Scale Software Issues	Sonia Sachs	Main Room
08:30-09:00 a.m.	Application Modeling and Simulation	Adolfy Hoisie	Main Room
Drivers for Modeling and Simulation Session			
09:00-09:30 a.m.	Architecture / Systems	Sudhakar Yalamanchili	Main Room
09:30-10:00 a.m.	Applications / Algorithms	Jim Belak	Main Room
10:00-10:30 a.m.	Breakout Session		
State-of-the-Art Session			
10:30-11:00 a.m.	Modeling	Adolfy Hoisie	Main Room
11:00-11:30 a.m.	Simulation	Arun Rodrigues	Main Room
11:30-12:00 p.m.	Emulation, Testbeds, V&V, Proxy, Apps, System Access	Jeff Vetter	Main Room
12:00-01:00 p.m.	Working Lunch: Industry Panel	All	Main Room
Gaps, New Directions, Priorities and Investment Discussions			
01:00-03:00 p.m.	Discussions of gaps in the ModSimS R&D Agenda	All	Main Room

Parallel Sessions, Preliminary Topics:			
• Modeling and Simulation for Exascale Application Development and Optimization. Moderator: Bob Lucas • Modeling and Simulation for Architecture Exploration. Moderator: Darren Kerbyson • Modeling for Intelligent Runtime Systems, System Software. Moderator: John Shalf			
03:00-03:30 p.m.	Breakout Session		
03:30-04:30 p.m.	Plenary: Discussions of Priorities, New Directions	All	Main Room
04:30-05:00 p.m.	Plenary: Recap / Goals / Homework for next day	All	Main Room

Friday, August 10th, 2012			
07:00-08:30 a.m.	Recap of goals, brief summary of first day and homework	All	Main Room
Gaps, New Directions, Priorities and Investment, Continued			
08:30-10:00 a.m.	Refinement of priorities, new directions, etc.	All	Main Room
10:00-10:30 a.m.	Breakout Session		
10:30-11:30 a.m.	Prepare research plan (scale, roadmap, timeline)	All	Main Room
11:30-12:00 p.m.	Action Items, Wrap up	All	Main Room

Participant List

NAME	Affiliation
Sanjay Kale	University of Illinois at Urbana Champaign
Keren Bergman	Columbia University
Andrew Lumsdaine	University of Indiana
Derek Chiou	University of Texas Austin
Chris Carothers	Rensselaer Polytechnic Institute
Sudhakar Yalamanchili	Georgia Technology Institute
Krste Asanovic	University of California at Berkeley
Luiz Ceze	University of Washington
David Mountain	Federal
Steve Reinhardt	AMD
Shekhar Borkar	Intel
Brad Beckmann	AMD
Jim Khale	IBM
Mike Parker	NVIDIA
Jackie Chen	Sandia National Laboratories
Kevin Barker	Pacific Northwest National Laboratory
Lenny Oliker	Lawrence Berkeley National Laboratory
Jim Belak	Lawrence Livermore National Laboratory
Greg Bronevetsky	Lawrence Livermore National Laboratory
Joseph Manzano	Pacific Northwest National Laboratory
Nathan Tallent	Pacific Northwest National Laboratory
Nick Wright	Lawrence Berkeley National Laboratory
Xingfu Wu	Texas A&M University
Lizy John	The University of Texas at Austin
Benjamin Lee	Duke University
Gilbert Hendry	Sandia National Laboratories
Doe-Hyun Yoon	HP Laboratories
Erich Strohmaier	Lawrence Berkeley National Laboratory
Simon Hammond	Sandia National Laboratories
Thomas Sterling	Indiana University
Boyana Norris	Argonne National Laboratory
Christian Engelman	Oak Ridge National Laboratory
Wilf Pinfold	Intel
Dean Klein	Micron
Kevin Lin	Micron
Jim Sexton	IBM
Jon Hiller	DARPA
Ananta Tiwari	San Diego Super Computer Center
Kalyan Kumaran	Argonne National Laboratory
Alfred Park	Oak Ridge National Laboratory

Abhinav Vishnu	Pacific Northwest National Laboratory
Daniel Chavarria	Pacific Northwest National Laboratory

Committee

Adolfy Hoisie	Pacific Northwest National Laboratory
Darren Kerbyson	Pacific Northwest National Laboratory
Robert Lucas	University of Southern California
Arun Rodriques	Sandia National Laboratories
John Shalf	Lawrence Berkeley National Laboratory
Jeff Vetter	Oak Ridge National Laboratory
William Harrod	U.S. Department of Energy
Sonia R. Sachs	U.S. Department of Energy