Report of:

WORKSHOP ON MODELING & SIMULATION OF SYSTEMS AND APPLICATIONS

August 13 – 14, 2014 – University of Washington, Seattle

Adolfy Hoisie, *Pacific Northwest National Laboratory*
Laura Carrington, *San Diego Supercomputer Center*
Jon Hiller, *Science and Technology Associates*
Darren Kerbyson, *Pacific Northwest National Laboratory*
Martin Schulz, *Lawrence Livermore National Laboratory*
Dolores Shaffer, *Science and Technology Associates*
Ankur Srivastava, *University of Maryland*
Jeffrey Vetter, *Oak Ridge National Laborator/Georgia Institute of Technology*
Bill Ward, *U.S. Department of Defense*
Noel Wheeler, *U.S. Department of Defense*
Sudhakar Yalamanchili, *Georgia Institute of Technology*

PNNL-23916

Workshop on Modeling & Simulation of Systems and Applications

Organizing Committee

Adolfy Hoisie (Chair)
Pacific Northwest National Laboratory
Laura Carrington
San Diego Supercomputer Center
Jon Hiller
Science and Technology Associates
Darren Kerbyson
Pacific Northwest National Laboratory
Martin Schulz
Lawrence Livermore National Laboratory
Dolores Shaffer
Science and Technology Associates
Ankur Srivastava
University of Maryland
Jeffrey Vetter
Oak Ridge National Laboratory/ Georgia Institute of Technology
Bill Ward
U.S. Department of Defense
Noel Wheeler
U.S. Department of Defense
Sudhakar Yalamanchili
Georgia Institute of Technology

Steering Committee

Roy Campbell
U.S. Department of Defense
Almadena Chtchelkanova
National Science Foundation
William Harrod
U.S. Department of Energy, Office of Advanced Scientific Computing Research
Thuc Hoang
U.S. Department of Energy/National Nuclear Security Administration
Adolfy Hoisie
Pacific Northwest National Laboratory
Sue Kelly
U.S. Department of Energy/National Nuclear Security Administration
David Mountain
U.S. Department of Defense
Karen Pao
U.S. Department of Energy, Office of Advanced Scientific Computing Research
John West
U.S. Department of Defense
Noel Wheeler
U.S. Department of Defense

Workshop Date: August 13-14, 2014
University of Washington, Seattle

December 2014

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Executive Summary

In our efforts to continue the momentum toward quantitative understanding of the performance, power, and reliability of systems and applications, the modeling and simulation (hereafter ModSim) community gathered for its annual Workshop on Modeling & Simulation of Systems and Applications, held August 13-14, 2014 in Seattle.

Originating in 2012, the first ModSim workshop culminated with the *Report on the ASCR Workshop on Modeling and Simulation of Exascale Systems and Applications* (available at: http://hpc.pnl.gov/modsim/2013/). This report identified and defined the “critical technology” areas of emphasis for ModSim research activities. Among them, it emphasized the need for *Co-Design*, where application developers, computer scientists, hardware architects, and computer vendors work together, analyzing future algorithms, applications, and computing systems, to generate an exascale ecosystem of systems and applications. Moreover, ModSim was identified as an essential part of that process. Since then, each workshop has emphasized a subset of the critical technologies to review the state of the art in those areas, get the pulse of the research, and identify new promising venues of investigation.

In 2014, under the guidance of Dr. William Harrod, of the U.S. Department of Energy’s Office of Advanced Scientific Computing Research (ASCR), we expanded the workshop’s participation to include committee and technical participation from personnel representing academia, national laboratories, the U.S. Department of Defense, and National Nuclear Security Administration, among others. This expansion was a nod to the still-evolving need to meet performance, energy-efficiency, and resilience requirements of systems and applications at all scales—from embedded to exascale—recognizing their broad impacts to the larger computational science community in a range of research areas, including those affecting national security and domain sciences.

The 2014 Workshop on Modeling & Simulation of Systems and Applications, or ModSim 2014 (http://hpc.pnl.gov/modsim/2014/index.shtml), focused on three critical technological areas:

1. Integrated Modeling and Simulation of Performance, Power, and Reliability
2. Standards, Integration, and Interoperability of ModSim Methodologies and Tools
3. Advances in Core Modeling and Simulation Methods and Tools.

The two-day meeting featured 21 presentations (based on peer-reviewed submissions) delivered over six parallel sessions, and there were six invited talks. Discussions aimed toward consolidating and directing this diverse technical community were especially beneficial as we continue to map out the future and form of ModSim research and development.

This report presents a summary of the presentations, discussions, and findings culled from ModSim 2014. Each of the three critical technology areas are presented, structured as major research directions, gaps, cross-pollination, and path forward.

Acronyms and Abbreviations

ASCR	Advanced Scientific Computing Research
API	application program interface(s)
CPU	central processing unit(s)
DOE	U.S. Department of Energy
DRAM	dynamic random-access memory
GPU	graphic processing unit(s)
HPC	high-performance computing
I/O	input/output
MIC	many integrated core(s)
ModSim 2014	2014 Workshop on Modeling & Simulation of Systems and Applications
NUMA	non-uniform memory access
NVRAM	non-volatile random access memory
PAPI	Performance Application Program Interface
PDF	probability distribution function(s)
PPR	power, performance, and reliability
RAPL	running average power limit
SST	Structural Simulation Toolkit
SWM	scalable workload modeling

Acknowledgment

The ModSim 2014 Organizing Committee respectfully acknowledges and thanks Karen Pao and the Steering Committee for their many contributions throughout the process of organizing and facilitating the workshop. We also would like to thank our invited speakers and the many researchers who submitted position papers, describing their novel research approaches for performance modeling and simulation at extreme scales.

The report authors thank Charity Plata, of Pacific Northwest National Laboratory, for the thorough and significant multiple passes that improved the document considerably.

In addition, the Organizing Committee would like to thank Stefané Johnston, of Pacific Northwest National Laboratory, for her logistical support and overall assistance in organizing this workshop.

Contents

Executive Summary i
Acronyms and Abbreviations ii
Acknowledgment iii
1.0 Power, Performance, and Reliability Co-Modeling 1
 1.1 Major Research Contributions 1
 1.2 Gaps 3
 1.3 Cross-pollination 5
 1.4 Path Forward 6
2.0 Interoperability 8
 2.1 Major Research Contributions 8
 2.2 Gaps 10
 2.3 Cross-pollination 11
 2.4 Path Forward 11
3.0 Modeling and Simulation Core Technologies 13
 3.1 Major Research Contributions 13
 3.2 Gaps 16
 3.3 Cross-pollination 18
 3.4 Path Forward 18
4.0 References 19

1.0 Power, Performance, and Reliability Co-Modeling

1.1 Major Research Contributions

The objective of power, performance, and reliability (PPR) co-modeling is to develop and leverage co-simulation tools that capture the interdependence between these physical phenomena for the purposes of understanding and balancing/optimizing system performance. The specific optimization goal may be any combination of execution time, energy and power efficiency, cost, lifetime reliability, etc. The 2014 Workshop on Modeling & Simulation of Systems and Applications (hereafter, ModSim 2014) featured two sessions devoted to PPR co-design, each including group discussions. Presentations during these sessions spanned four areas of interest: 1) system-level management, 2) architecture-application interface, 3) interprocessor communications, and 4) hardware-centric modeling. The underlying theme in each presentation included the need to develop application-architectural co-design schemes where the application—and the supporting software—are made more aware of the detailed hardware models and data from on-board (or on-chip) sensors while simultaneously driving application-aware hardware synthesis and integration. The overall objective for this integration is to develop techniques for improving the resilience, reliability, execution time, and energy efficiency. The major research challenges identified within each of the four areas of PPR include:

System-level Management. Utility costs, as well as limitations with power delivery and related reliability issues, limit the size and future expansion of high-performance computing (HPC) systems. For example, the cost of cooling facilities is fast approaching the cost of powering the computer systems (Moore et al. 2005). Consequently, facility temperature settings have become as important as setting the power state for the computer systems (achieved through dynamic voltage frequency scaling). Increasing the temperature set point will increase performance and maximize use of delivered power. However, the coupling between temperature and static power will further increase temperature and must be managed to avoid thermal violations. Furthermore, higher temperature leads to increased transient error rates and lower system lifetime reliability. Thus, navigating the PPR envelope at the facility level is a complex optimization problem modulated by specific application workloads. Another related concern stems from the fact that different devices have dissimilar vulnerabilities to temperature. Disks and dynamic random-access memory (DRAM) are more susceptible to soft and hard failures due to temperature than central processing units (CPUs). Moreover, conventional approaches to "setting" the HPC temperature focus on controlling the air temperature surrounding the HPC servers. This "remote" cooling approach exacerbates the mismatch between temperatures at the device level versus the surrounding environment. At a computing facility level, the challenge is to develop integrated PPR models that can capture these multi-level and multi-physics dependencies and be used to: 1) reduce thermal hot spots via load balancing or server power state management; 2) trade-off power and lifetime reliability with execution time performance; and 3) manage the facility (e.g., determine temperature set points). Still, a related question remains: *where within the system are these trade-offs managed?* Some options include: applications, libraries, runtimes, system software, hardware, or a combination of these options. In addition to determining the integrated PPR model construction, the specific management policies and algorithms also remain a challenge.

Architecture-Application Interface. Characterization of the energy behaviors of algorithms and applications is a nascent area. Application developers are disadvantaged by the lack of tools or models to help them understand the energy dissipation profiles of the applications. For example, there are currently no tools to illuminate the energy consequences of design decisions, e.g., data structure and algorithm choices. While time and space parameters are well-developed concepts, power and energy complexity are not. For example, different implementations of the same algorithm (e.g., breadth-first search of a graph) can have especially varied energy consequences over different input data sets. Therefore, a remaining key challenge is the ability to develop application-level models and associated tools for identifying “energy hotspots” in an application for the user. A second challenging problem involves assessing power and energy efficiency optimization techniques in a manner that can lead to concrete expectations of performance on alternative platforms and determining how can such assessment strategies can be constructed. By themselves, these types of test and validation approaches pose significant modeling and simulation challenges. To address this challenge, one discussion at ModSim 2014 introduced the concept of proxy architectures to be used in conjunction with benchmarking, along with using common baseline architectures to enable platform-independent analysis.

Interprocessor Communications. Data movement likely will dominate energy and performance costs in future HPC systems. Thus, the development of models for assessing communication performance and communication energy at scale will be increasingly important. These models must be able to capture aggregate behaviors at higher levels of abstraction because models at the level of packets, “flits” (flow control digits), or clock cycles are computationally infeasible for any reasonable-sized system. Further, systems are moving toward using heterogeneous processors (e.g., accelerators), diverse communication technologies (e.g., optics), and supporting new system abstractions (e.g., global address spaces). These novel system designs will require new unifying abstractions to model their communication interactions. Finally, the energy analysis of existing abstractions was addressed in several talks. For example, as current understanding is imprecise, the researchers identified a need for modeling and comprehending the energy behavior of collective communication and global addresses spaces at scale. Overall, ModSim 2014 introduced new abstractions for modeling and managing communications from the perspective of capturing the spatial and temporal locality of interprocessor communications.

Hardware-centric Modeling. This theme observed the need for integrated PPR models that must be abstracted from physical implementations. The importance of understanding the impact of physical phenomena, such as thermal fields and voltage noise, stems from the coupled multi-physics behaviors of these digital systems. For instance, temperature and static power have an exponential relationship that can quickly nullify the management schemes and optimizations that are not cognizant of this relationship (Butts and Sohi 2000). Moreover, temperature affects device degradation and lifetime reliability. Temperature also affects the transient error rate. When these phenomena are controlled, they have an impact on execution time performance in a manner that is determined by the processor and memory system microarchitecture, as well as the application behaviors. Thus, a method must be devised to capture thermal and power effects and couple them to key application behaviors to enable models at scale that allow reliable understanding of system behaviors and help users make informed trade-offs. For example, one discussion observed that the most effective degree of parallelism in the application is a function of the PPR behavior that, in turn, is platform-specific.

Herein, specific research agendas and major challenges are identified in Section 1.2. Addressing these gaps is most effectively done via opportunities for cross-pollination among researchers, who employ different aspects of modeling and simulation. A few examples for such cross-pollination are identified in Section 1.3.

1.2 Gaps

In the four major PPR research areas, several gaps were identified across the eight presentations featured at the ModSim 2014 workshop.

Metrics. Understanding the interplay of the PPR of a system requires capturing interactions between multiple physical phenomena across diverse architectures and applications. Capturing these interactions and dependencies in informative metrics that can be used to understand and optimize system behaviors remains a challenge. Some of these challenges identified were as follows:

- How does we capture the trade-off between energy efficiency and lifetime reliability (e.g., expressed as mean time between failures) when graceful degradation is permitted?
- How do we capture the effect of thermal fields on device degradation?
- What is the relationship between the degree of parallelism and energy efficiency, and how do we measure it?

The definitions of suitable metrics will support the companion definition of benchmarking and comparative evaluation strategies.

Model-specific. PPR models are founded on physical behaviors that must be abstracted to higher levels to enable analysis of application and architecture combinations. Detailed physical models are computationally infeasible in such design-space exploration environments. Currently, there is a lack of compact models for computing two-, two-and-a-half-, and three-dimensional (2D/2.5D/3D) thermal fields and power delivery, which also have low asymptotic computational complexity and acceptable accuracy. In particular, models for system-level reliability in this domain and more recent notions of resilience are lagging behind efforts focused on power and energy modeling. Acquiring better physical models that lead to improved prediction of device failure probabilities and enhanced throttling for power and performance also was an identified need. Another deficiency involves integration of data-driven models with multi-physics-based models, which present physics modeling and data assimilation challenges. In addition, it is not clear if the benefit of using these models online outweighs the cost of validating and integrating them into the computing system. For example, the operation of power delivery networks can be coordinated with power management mechanisms of the cores and memory system to minimize energy losses. Modern hardware is capable of providing a host of dynamic measurements, such as power consumption, energy consumption, device aging, etc., via novel sensor designs. These measurements can be delivered to runtime software modules for analysis, which may then drive power management decisions. For example, aging sensors can be used to migrate tasks to other cores to extend the lifetime of a degraded core. A crucial challenge identified in this regard is in understanding the data required of models and using these models for selecting the right information at the proper time to make the most effective PPR trade-offs.

These described gaps are one reason we are also faced with a lack of fast design space exploration techniques. In part, these techniques depend on the availability of compact, interrelated models of physical phenomena (i.e., thermal fields). A related issue is the complexity of the models themselves, such as the time-consuming execution of integrated PPR models. When used with high-fidelity simulators, the execution of the integrated PPR models can take longer than the timing simulation of the multicore systems being modeled. Consequently, it important to find PPR modeling algorithms that are optimized and parallelized to reduce model execution times to practical levels. For example, the computation of thermal fields can be parallelized. The integration of such parallelized PPR models with system models of application-architecture combinations is challenging. There are no standardized interfaces or even commonly accepted ways of integrating physical and simulation models. Similarly, unlike the computer-aided design (CAD) community, there is no consensus on how technology dependencies should be handled. Ideally, we would like validated parametric models of power, energy, etc., across technology generations. While some public domain models exist (e.g., McPAT and DSENT), the community would benefit from adopting or modifying popular open-source tools or evolve a strategy for creating one. Finally, it was observed that PPR modeling requires that multi-physics interactions must be implemented, e.g., temperature-power and temperature-reliability. These fundamental interactions are independent of architectures and applications issues and are better served by being captured within the models, such as the enterprise infrastructure (EI) infrastructure (Song et al. 2014). Generally, this is not the case today, which leads to the need for developing a large-scale modeling toolkit that combines models developed from first principles with real dynamic measurements. Such a toolkit must have well-defined interfaces between modeling layers to enable tool selection based on the level of abstraction desired.

Runtime- and Application-level Models. To date, the bulk of PPR modeling has been at the hardware level, and this must evolve. We must be able to model PPR behaviors hierarchically at all levels of the software stack, all the way up to the application level. Models at each level would be based upon measurement data and models from lower levels of abstraction. The availability of performance measurement interfaces, such as PAPI and Perf, must become widespread in HPC machines. Energy measurement interfaces, such as Intel's RAPL (running average power limit) interface, are a good start. These are necessary to generate models for energy debugging of applications much in the same way performance debugging tools exist today. In particular, we lack energy models of communication behaviors at scale, such as collective communication, communication congestion, and behaviors in global address space models. Given the energy dominance of data movement in future HPC machines, large-scale communication models are particularly important. Some of these opportunities include investigating specific modeling attributes and modeling the effects of resource utilization and contention. Resource utilization is directly linked to power, efficiency, and reliability.

Software Infrastructure. One major practical concern is the lack of standardized interfaces and associated application program interfaces (APIs) for integrating PPR models into 1) large-scale simulation and modeling environments and 2) production applications and runtimes. There is a lack of standards or even a state of practice in using and integrating physical models. For instance, structure (perhaps, standardization) is lacking APIs for accessing/exercising physical models at multiple levels of abstraction. While standardization may not be necessary, basic library structures are lacking for integrated PPR modeling. Notably, public domain tools (e.g., McPAT and DSENT) and frameworks, such as Energy Introspector (which also wraps the other two tools), are emerging. However, we are far from any grassroots momentum toward using integrated PPR tools.

A related concern involves integration into runtimes. Any toolkit must have two distinct capabilities: 1) physical modeling of PPR using a combination of data-driven and multi-physics-based models and 2) optimization/runtime control approaches at software and hardware levels. At the lower level, software must have access to hardware counters for energy and/or power, as well as sensors for temperature, aging, and other possible physical behaviors. These measurements will feed robust PPR models for runtimes to make management decisions. At the high end, we need abstractions (or "knobs") that the runtimes can exercise to control PPR trade-offs. At the moment, the only hardware knobs are voltage-frequency scaling via power states or (in some select systems) power and temperature capping. Software knobs include task and thread placement. Still, questions remain, including those surrounding what other control variables are needed, e.g., network bandwidth-energy trade-off or input/output (I/O) energy. There also is an ongoing discussion as to how much of this runtime control should be made available to application developers, if at all.

PPR for Data Movement. Data movement energy expenditure will dominate total energy costs in future machines. However, there is a dearth of models and management techniques for understanding the PPR trade-offs of data movement within and between nodes. In addition to better models, we need enhanced tools to identify energy and power bottlenecks of data movement and opportunities for optimization. For instance, during certain phases of computation, link or memory channel speeds can be reduced significantly with minimal or no compromises in overall system performance. Identifying these phases will be especially useful in tuning the energy efficiency of applications. These memory and communication overhead models must be combined with CPU behavior models for predicting the overall system PPR. As mentioned earlier, such models would rely on a combination of multi-physics and dynamic measurements. Similarly, we can envision tools for identifying vulnerabilities in communication, energy hot spots, balance of compute versus communication energy, etc.

Finally, we recognize that the programming model, as well as the data/memory organization, affects the communication overhead at the system level. Based on this understanding, accurate tools are necessary for predicting network congestion and latencies.

Resilience Characterization and Management. In the future, a crucial challenge will be the ability to correlate hardware transient errors and low-level component failure probability with system-level events. This requires analysis techniques and models that can be encapsulated into tools and techniques that analyze error propagation through complex software stacks. Low-level component failure is poorly understood and critically important. Minimally, there must be complete logs and state descriptors noting the age of devices, data from aging sensors, alarm history, etc., for on- and off-line analysis. The PPR models drive our understanding of the relationships between application behaviors, performance, energy, and resilience and seed ideas for managing their relationships.

1.3 Cross-pollination

The two PPR sessions led to the identification of several opportunities for cross-pollination:

Co-Design Activities. It is important to be able to iterate between the development of the applications and the design of processor and systems architectures with respect to the resulting PPR behaviors. For example, application implementation has energy and power consequences, as well as execution time and memory space consequences. While minimizing time and energy/power is an accepted and well-understood optimization goal, optimization of reliability in HPC applications is not. With proper tools, it is possible to iterate between application implementation and assessing the PPR consequences on a specific hardware target. Significant opportunities remain in such application-architecture co-design activities with respect to optimization of PPR behaviors. As of yet, we have not developed the ability to use integrated PPR models to inform the implementation of application codes and hardware architectures toward specific goals, such as maximizing mean time to failure (MTTF), and there are no modeling tools that can be used to help design reliable software with minimal impacts on performance. The ability to minimize performance impact will require exposing and eliminating redundant vulnerabilities in the system, e.g., reducing processor state and vulnerability to transient faults at the expense of occasional increases in memory access overheads.

High-level PPR Modeling. Currently, there is a lack of reliable, accurate models that depict the energy consumption of software. Consequently, energy and power debugging of applications is infeasible (e.g., *what are the energy bottlenecks?*). Reasoning regarding the energy and power behaviors of software is difficult. Interactions between the software (both applications and system) and PPR communities (currently tending to be [low-level] hardware-centric) will lead to major benefits in application optimizations for power efficiency and the ability to make trade-offs across execution time, energy, or power efficiency and reliable operation. Specifically, one area of concern is in understanding how faults (especially transient) propagate through modern software stacks. The investment in hardware resilience is of limited value if we cannot control their propagation—even if hardware occurrences are reduced. Developing such schemes requires collaboration across the system software, hardware, and resilience modeling communities.

System-level Interactions. There has been relatively little PPR modeling activity addressing system-wide inter-processor communication or I/O subsystems. The majority of the effort has focused on processors and, to a lesser extent, the memory systems. The networking architecture community—both hardware (network/switch hardware) and software (message layers)—should be engaged with the PPR modeling and simulation communities to develop modeling and simulation techniques dedicated to understanding energy-efficient design of interconnects, networks, and I/O at system scale.

Measurement and Modeling Communities. There is a significant opportunity for productive interactions between the community that constructs hardware and software instrumentation and the one that constructs energy and power models of platforms and software. The most obvious would benefit model validation. Model quality also could improve because measurements will enable delineation of platform-specific and platform-neutral (software-specific) energy and power consumption models. Furthermore, model requirements can guide the functionality of hardware and software measurement techniques. Interactions between these communities could evolve interfaces and associated APIs that would contribute to interoperability. Standardization of model interfaces to PPR models would extend naturally to their use in simulators and runtimes, expanding the benefits of increased interoperability between PPR models and various infrastructures.

1.4 Path Forward

In summary, participants noted that PPR effects permeate the complete system software and hardware stacks, identifying those PPR aspects across the entire system stack that need to be understood and modeled. For example, reliability loss occurs through a combination of device failure and soft errors, and these errors propagate through deep, complex software stacks. Power and performance dissipation are a function of device utilizations, which depend on system architecture and the software stack. A related challenge would be to identify acceptable levels of PPR, quantified in metrics that would be used to setup overall optimization constraints. Finally, it was noted that these types of modeling and optimization frameworks must use a combination of data-driven (i.e., measurements) and multi-physics-based modeling approaches. Such PPR frameworks can perform PPR trade-offs both in space and time.

2.0 Interoperability

The second major focus area addressed in ModSim 2014 was interoperability. Historically, models crafted to simulate computer system components or architectures were predominately one-off creations that developers crafted to solve their own specific problem set. After their intended use, the tools were, for the most part, abandoned quickly, rendering the models and simulations unusable and forgotten. Generally, computer technology evolves in steps, not leaps. At each step, only a few components are changed. For example, in the memory subsystem, a hybrid memory cube (HMC) may be substituted for traditional DRAM devices. At each evolutionary step, the need to model and simulate the system to understand the value of these changes is highly desirable. For both practicality and efficiency, the ability to leverage models from other research and a variety of developers, as well as reuse models, is highly desirable. To enable such activities, the ability of models to interoperate with each other and within supporting frameworks can be considered a crucial attribute to integrate into any tool design.

The interoperability area received few position paper submissions: only three position papers out of the 22 accepted were in this area. This reflects the scarcity of research currently in progress, but it does not diminish the importance of this area, echoed accordingly during the presentations and comments made as part of the workshop.

The three position papers represented work in different areas of modeling and simulation and provoked much discussion. The first discussed work in progress at Sandia National Laboratories on the Structural Simulation Toolkit (SST), a lightweight software framework that uses third-party simulation components. SST was viewed as needing standards to link core simulation components. It is motivated by reusing simulation components that span the spectrum of micro and macro levels. However, the lack of standards and code reuse is hindering its progress. The second paper described using field-programmable gate array (FPGA) emulation to achieve orders of magnitude higher execution speeds than simulation, with a memory analysis and data reorganization example. However, there are no standard hardware emulation platforms, resulting in a lack of adoption. In addition, there is a clear need to interoperate with other simulators for a full system view. The third paper described CoDEx, representing a collection of individual tools from hardware emulation to software simulation. Interoperability was the biggest challenge in this work as there was a need to interoperate within the CoDEx tools, as well as between CoDEx and other toolkits.

Herein, we detail the discussion points made both by presenters and group participants during the sessions: current major research contributions, existing gaps in the research, the need and potential of cross-pollination, and recommendations and path forward.

2.1 Major Research Contributions

This section describes current research contributions in interoperability but also points out shortcomings (further discussed in terms of gaps in Section 2.2). Several efforts exist to promote interoperability, including SST (http://sst.sandia.gov/), using the SystemC programming language (http://www.accellera.org/community/systemc/), and defining hardware interoperability. However, they have achieved limited success. Most models and simulation tools are developed within a stove-pipe environment. Many feel justified in doing this based on a lack of ongoing funding for tool development and deployment and an anticipated short life span expected by researchers who merely seek to solve their problems without consideration for a second life the tool might lead serving the modeling and simulation community. A tool's life span frequently is defined by its support, documentation, and interoperability with other tools, not its utility.

Standards. Dr. Andrew S. Tanenbaum (1988) said, "The nice thing about standards is that you have so many to choose from." Within this forum, and much of the software community, this sadly is a relevant truth. Standards are a critical aspect for interoperability, but there is little current work being pursued or funding available to do so. Persuading modeling and simulation researchers to consider creating standards for interoperability between tools and components is relatively easy. However, everybody has a tool or technique that he or she wants promoted and recognized as the "standard."

- *Is it possible to achieve the goal of creating interoperability between modeling and simulation tools?* In theory, this is possible and has been achieved in other software design spaces. However, to move toward this goal, it will be necessary to start with extensibility in mind.
- *Do we value interoperability?* In short, yes. The ability to leverage tools and techniques developed and validated by other researchers is essential to modeling complex architectures, particularly when considering emerging technologies.

Data Formats. Researchers and developers should not fixate on formats for the data into and out of modeling and simulation tools. They should focus on the contents of the data, then on tools that translate the data formats (commas, colons, and so forth). The I/O from simulation tools should consider both meta and event data. However, it is recognized that data interchanged between modeling and simulation tools will likely generate event information that is vague in meaning and utility to the components. A mechanism to process this, or disregard it when appropriate, must be considered.

Component Models. There are many examples of component models developed by individual research teams for specific purposes. These range from hardware emulation to software modeling and simulation tools. Their scope ranges from hardware technology analysis to application software. However, the ability of individual models to leverage other component models and support frameworks and analysis tools depends on establishing standards for interoperability. A language capable of defining models and how they operate in simulations is highly desirable. One option to consider is the SystemC programming language.

It may be necessary to consider tight coupling between component models in a simulation. It is recognized that a "one-size-fits-all" solution may, in some cases, be an impediment to accurately modeling a system. However, tighter coupling would need to emphasize thoughtful implementation to promote the highest reasonable degree of interoperability.

Reproducibility. In any scientific research pursuit, the ability to reproduce experimental results is critical. The current best practices are for individual components by different teams to be validated against their counterpart's physical implementations in appropriate scenarios, but it is rare for tools to undergo outside use or validation. Additional elements of reproducibility could include validation of modeling and simulation tools against existing and known computer systems and their supporting applications. Reproducibility of experiments will greatly aid in developing a model/simulation tool's reputation and trustworthiness.

To encourage openness in data exchange, a repository should exist to contain, share, and reproduce experimental results. Submissions to a repository would include data and message traces, documents defining the format and meaning of the data, and (if applicable) conversion tools to migrate traces to other formats or distill metrics. Data standards for interoperability and data/message interchange should be extensible. For example, additional fields should be available for optional data.

Interfaces (APIs). The development of a useful and long-lasting API is a challenging task—one that has yet to be tackled for interoperability. Questions arise regarding how to describe the interfaces, what metadata should be included, how extensibility can be ensured, and how to overcome proprietary issues. When possible and appropriate, APIs should strive to contain high-level data (meta) and avoid non-essential low-level details. The development of new model/simulation components should begin with inclusion of the API as part of the basic design. An API specification should begin with well-documented interfaces and highlight the best practices ("dos" versus "don'ts"), such as those seen during development of the message passing interface (MPI). It was suggested that API development should begin with the creation of a common core with basic functionality and a high degree of extensibility. This would allow time and practical application of the API to further define/refine the specification.

2.2 Gaps

The discussion regarding current gaps in interoperability focused on the following aspects:

Standards. At this time, interoperability standards between modeling and simulation components simply do not exist. This hinders development of tools, contributes to their lack of reuse, and stifles the potential use of research that modeling and simulation tools are being developed to analyze. Fostering interoperability between components is a difficult process. Deferring or ignoring it is simple and, in the short term, inexpensive. However, this mentality inhibits modeling complex architectures and leveraging the work of others.

Research or Development. There is a difference between research and development (R&D), and enabling interoperability is deemed as "D" and not glamorous. Enabling interoperability will require substantial initial engineering, as well as revisiting existing models/components. Challenges exist when describing composition. Pros, cons, and uncertainties must be considered when defining model compositions. This will be essential for the models to function in a reasonable capacity and generate trustworthy results. The impact of featuring interoperability will need to be considered. Side effects of interoperability may include: impeded capabilities; reductions in flexibility; and, of course, performance/execution speed penalties.

Incentives. There appears to be more incentive *not* to enable interoperability. To some degree, there seems to be a willful lack of interoperability between components. Incentives should be considered to coax researchers and model and simulation tool developers to integrate a standard API instead of developing yet another boutique tool for the task at hand. Several natural questions emerge when considering this, including: *what is gained by doing this*, or *what would the maintenance model look like?*

Some incentives could include:

- The concept of "Likes" and "Not Likes" (from Facebook)
- An avenue to publish that offers credibility to the researcher or developer

  - For example, distinguished credibility (and resume builder) as developers/maintainers of a set of modeling and simulation tools that could be implemented using a certification model, such as those employed by Microsoft, Cisco, and many others.
- A sense of ownership in a tool that is widely accepted and used by the community.

Longevity. Modeling and simulation tools appear to lack longevity. In general, the computer industry is fast-paced. When considering the evolution of components, it is easily noted how models can quickly become dated, if not obsolete. Frequently tools are developed for use within individual projects and remain unadvertised or published.

Documentation. Numerous excellent modeling components are available for use. However, many are minimally documented and fairly incompatible with each other. Details concerning models and simulation tools frequently are minimal. Routinely, they are nonexistent. Although not core research, such activities are critical, for example, in any software engineering project.

Trustworthiness. The development of a model's credibility can be accomplished by demonstrating correctness in practice. Simply stated, there is no other way to achieve this, and it is a "must-have" requirement for any modeling and simulation tool. It was suggested that modeling and simulation components, frameworks, and tools could seek "certification" from a community testing body. One example of this could be the Programming Language Design and Implementation (PLDI) certification. A mechanism would be necessary to evaluate the correctness of the models and simulation tools, compared to physical hardware. In addition, knowing and documenting the limitations and scope of the tools' capabilities will aid in mitigating erroneous experimental results. Accountability for the capabilities and results from modeling and simulation tools also would be available. This could include test cases that ensure the tool's integrity and sample metrics that document accuracy.

It also was acknowledged that proprietary issues likely will exist. To protect intellectual property and observe legal ownership rights, national laboratory and academic researchers, as well as industry partners, may be reluctant to share information describing experiments.

2.3 Cross-pollination

Interoperability is a core aspect to enhance the development of tools within a modeling and simulation activity, as well as enabling the interaction and integration across research teams and activities. As such, it is central to the cross-pollination of other modeling and simulation activities already discussed elsewhere in this section.

2.4 Path Forward

From the interoperability position papers presented and conclusions drawn from the open discussions held as part of ModSim 2014, the following measures were suggested as a means to evolve this community:

- The community recognizes the need to foster interoperability between modeling and simulation components. Sources for models and frameworks should be able to be derived from various sources, including federal agencies, national laboratories, industry partners, and academic institutions.

- The community must identify mechanisms that can be used to incentivize development of interoperable models and frameworks. These may be in the form of awards or other informal or formal recognition. Additional ideas to enhance such promotion certainly are welcome.
- Ultimately, the community will need to identify resources and funding models for maintaining and evolving modeling and simulation tools. The Linux community refers to this function as a "Kernel Janitor," a person who has undertaken the less-than-glamorous role of maintaining and documenting key software. It was suggested that a similar model for modeling and simulation tool maintenance be employed.
- To raise awareness, it was suggested that "birds of a feather" (BoF) sessions at major computing conferences could draw attention to the need for interoperability between modeling and simulation tools.
- Developing standards for interoperability between modeling and simulation tools was recognized as a complex topic with many facets. It was suggested that there may be value in creating working groups within the community focused on creating and documenting APIs, new component model standards, and other software to foster reproducibility of experiments. These groups also could create incentive mechanisms to promote using development standards and adhering to documented APIs, ensuring the tools are usable and produce validated results.

All of these activities require the recognition that interoperability is, indeed, important. Seed funding could spark several of these activities and may provide motivation over time to enable interoperability to become a first-class concern. An investigation and adoption of best practices for the modeling and simulation community would aid in developing the integrity and credibility of tools and capabilities. A cultural mindset change will be necessary to enable interoperability, such as promoting the truth that interoperability will enable greater research capabilities and opportunities (one-dimensional versus multiple dimensions).

3.0 Modeling and Simulation Core Technologies

3.1 Major Research Contributions

Four major core technology themes emerged in ModSim 2014. First, large-scale, high-level approaches were significant. These included modeling and simulation technologies using multiple frameworks, modeling workflows in addition to systems, and scalability modeling. Next, workload generation and data formats for specifying workloads were identified as important. Third, modeling memory/data movement and its optimization (e.g., for non-uniform memory access, or NUMA) was viewed as critical. Finally, new simulation approaches were introduced, including those for simulating non-volatile random access memory (NVRAM), identifying bottlenecks for critical instructions, and power measurement/modeling in complex memory systems.

Large-scale, High-level Approaches. A significant modeling and simulation problem is that if a simulation model incorporates too much detail with respect to system features or time scale, it will run too long to be useful. This is particularly true of cycle-accurate simulators. Either the models themselves must be implemented as highly parallel programs, or some of the system details must be abstracted away.

Modeling at the workflow level (as opposed to the instruction, message, or routine level) has emerged as one such approach. Many HPC jobs are just one of several subtasks in a more complex workflow. Simulation models that take advantage of this employ a more coarse-grained approach that is faster but still provides reasonable approximations to system resource usage. To implement these sorts of models, workflows are represented as directed acyclic graphs (DAGs). Profile data for critical application parameters (e.g., cores required, compute intensity, I/O rates, and power consumption) are used to build probability distribution functions (PDFs) of workflow application features. Because of the granularity of this approach, the data need only be collected at the overall application/job level. Simulated applications then are created using the PDFs to select appropriate values for tunable parameters in an application template. Simulations proceed as these synthetic applications queue up at systems, providing the necessary compute resources. By modifying scheduling policies and varying resources available in the systems, these types of models allow optimization of performance and resource use across entire workflows instead of only individual jobs. In addition, by their very nature, many workflows use a distributed computing infrastructure, and workflow models have the capability to predict future resource availability and capacity at distributed system sites.

However, several serious problems have been identified that hinder progress in workflow-level modeling. Although obtaining performance data and queue wait time is relatively easy, performing energy-aware profiling of workflow applications remains quite difficult. This is an important issue because predictions based on a single variable (i.e., time in queue) are not consistently reliable. Moreover, workflow performance on actual systems often is chaotically sensitive to system load, and the workflow models themselves are sensitive to the time scale of the predictions attempted, resulting in usable predictions only out to short time spans. To mitigate these problems and increase the timespan for reliable predictions, strong—and often non-realistic—assumptions (e.g., homogeneity of applications, systems, and environments) are made. Based on these difficulties and model complexities, multi-objective optimization has not been frequently attempted.

Workload Generation. Workload modeling and generation have been extensively studied in the context of job scheduling strategies for parallel processing.[1] In this context, logs (lists) of jobs executed on an HPC system are used as input to simulators that allow modification and optimization of job scheduling policies to minimize wait time, better serve some job class, or improve system responsiveness in another way. However, in the context of modeling and simulation for exascale system design, scheduling models have serious shortcomings. Because job logs typically provide only job submit time, start time, and end time, system cores are recorded as busy or not, but little or no useful results are obtained about the detailed use of system compute, interconnect, I/O, and other facilities. Previous attempts to resolve this using process communication traces and/or statistical traffic pattern generators have proven viable, but either will require vast trace files or approximate communication patterns (e.g., characterizing the blocking behavior), or both.

Recent modeling and simulation work in the area of scalable workload modeling (SWM) remedies these shortcomings and is similar (in spirit) to using "skeleton apps" in SST/macro (http://sst.sandia.gov/doxygen/sstmacro/pages.html), or, in a separate representation, such as the Aspen (Abstract Scalable Performance Engineering Notation) performance modeling language. In these approaches, the representations are created via discussions with application developers and studies of the application codes. They represent the inter-processor communications patterns from an application abstractly and implement them in an application-independent manner. The resulting scripts are written to allow creation of pseudo applications with arbitrary numbers of processes and application parameters. Then, collections of representations of these pseudo applications may be used to drive simulators of various types. However, the manual labor required to create the scripts is one shortcoming of this method, and progress in this area has slowed because of an absence of a standard workload description format. Still, to date, SWMs have been created for a dozens of benchmarks, algorithms, and workloads, and pseudo workloads have been used to drive simulations up to 64,000 nodes.

Modeling of Memory/Data Movement. Since 1968 when cache was introduced in the IBM 360 Model 85 (Liptay 1968), one trend has remained constant: the increase in compute speed has outpaced the increase in memory speed year after year. In the 46 years since, computer architects have introduced additional levels of cache, translation lookaside buffers (TLBs), and other similar features to compensate for this problem, commonly known as the "memory wall." Introduction of HPC clusters requiring message passing added to this problem, and the push to exascale—requiring vast amounts of memory and data movement—will further emphasize the issue. The research community has recognized this as a key problem and is beginning to use modeling and simulation to explore ways to increase memory speed, reduce data movement, or (at least) reduce its cost in power.

The current bulk synchronous parallel mode of computation requires sufficient load balancing throughout a job run to achieve good performance. Modern codes are attacking problems using algorithms with irregular memory access patterns, and the path through the computation varies depending on the input data. In addition, massive on-chip parallelism, the accompanying multiple levels in an on-chip memory hierarchy, and introduction of heterogeneous architectures (graphic processing units [GPUs], many integrated cores [MICs], and the like) all contribute to the difficulty of moving data efficiently. Automated dynamic load balancing appears to be a necessity—not just for computation, but for memory accesses as well. One approach is the increased use of on-chip tiling to partition work

[1] Refer to "Job Scheduling Strategies for Parallel Processing" International Workshop Series (ongoing) in *Lecture Notes in Computer Science*. Springer, Heidelberg, Germany.

effectively across the hundreds or thousands of cores expected to be on-chip. Current cluster nodes already use NUMA, but its (required) use on 1,000-plus core chips will pose serious performance problems. Success here relies on a topologically aware runtime system that can select the best tile size and map threads and memory to the tiles to optimize using NUMA memory. These decisions may be made using input from parameterized analytical models with parameter values based on results from simulation models or on simulation results themselves. This technique already has been prototyped, and reductions in runtime of up to 50 percent versus default data mappings have been observed.

The sheer quantity of memory required for an exascale system is another problem that must be addressed. Due to cost and reliability, the 1 GB, 2 GB, or more per-core requirement (depending on your favorite application) simply cannot be maintained using DRAM for systems with millions of cores. Using disk-based virtual memory is not an option because of performance, and a disk on every node for local virtual memory would consume significant space and power. NVRAM, whether or not it uses current flash technology, or some future replacement (e.g., phase change random access memory [PCRAM] or resistive random access memory [ReRAM]) is a probable solution because it fits neatly between DRAM and disk in terms of performance and cost. However, understanding how to effectively use NVRAM at exascale levels will require modeling and simulation. Fortunately, those same current DRAM-heavy nodes can be used to simulate the performance of future NVRAM-equipped nodes with small DRAM footprints. Current simulation technology allows simulation of large workloads running on NVRAM simulated in DRAM with the capability of setting latencies and bandwidths to conduct experiments to determine optimal memory configurations for a given workload.

A fundamental requirement in the study of memory and data movement operations is the ability to measure the energy required for those activities. This is particularly important for exascale systems with large amounts of distributed memory. Unfortunately, nearly all studies of this subject have been qualitative. However, recent work has developed a method to measure energy costs for moving data between each level of the memory hierarchy (between registers and L1 cache, between L1 and L2 cache, and so forth). The methodology is not as straightforward as might be expected. Both external and internal (to the processor) power meters are used, and baseline energy consumption (e.g., from fans and volatile memory) must be subtracted away. Energy costs incurred from out-of-order and speculative instruction execution also must be removed. At this point, judicious design of parameterized microbenchmarks that allow appropriate selection of loop stride permits isolation of the energy costs between the various memory levels. Using this approach, it will be possible to conduct experiments to better design energy-efficient memory subsystems.

New Simulation Approaches. A number of recent modeling and simulation activities have begun to bear real fruit in the form of useful tools. One such effort reasonably assumes that application runtime may be modeled as a linear combination of terms of the form p^i * (log2 p)^j for a range of different parameter values for i and j. Performance measurements are taken at different core counts, and functions are ranked by resource use. Linear regression then is used to determine which components in the linear combination are significant, i.e., for which parameter values of i and j the model fits best to observed data. The tool already has been used to find scalability problems in real codes.

Another successful approach uses rewriting of an application's binary executable file to quantify the effects of various performance problems, (e.g., floating-point result and memory dependencies and memory bank conflicts). For example, this is accomplished by substituting no-op instructions for memory operations, so the computation is all that remains. This provides a ceiling on performance in that computations proceed without any memory stalls and quantifies the performance due to memory operations. Conversely, it also is possible to replace all of the floating-point instructions with no-ops to study how much performance is due to those operations.

3.2 Gaps

High Level. A significant number of gaps were identified in the modeling and simulation process. At the highest level, there is a need to emulate and simulate systems at scale. To impact exascale system designs, we require an improved understanding of what applications will run on such systems and at what scale. More specifically, *will exascale systems be used primarily as capacity systems running jobs using some fraction of the machine*, or *will they be capability systems executing full machine "hero" runs*? Without such understanding, it will be impossible to create model workloads for input to simulations. Just as the tools and techniques used to build three-story townhouses fail if they are used to build skyscrapers, an additional significant barrier to "at-scale" simulation is that tools effective for modeling current-generation systems almost certainly will fail in unexpected ways when the models are scaled up in size by an order of magnitude.

Validation and Uncertainty Quantification. Validation and uncertainty quantification go hand in hand. Without some set of standard modeling problems for which answers are known, validation will be much more difficult, even for current-scale systems. Moreover, input perturbation approaches for uncertainty quantification will not be possible. Sources of uncertainty include:

- Inexact values of model parameter (Kennedy and O'Hagan 2001)
- Variability (from one system to another) of model parameters
- Failure of the model to include all system aspects
- Use of inexact algorithms
- Observational variations/errors in input data.

Many modeling approaches have been proposed in the current (and prior) modeling and simulation workshops, yet few have devoted much effort to addressing any of these sources of error. Only one position paper in ModSim 2014 offered serious attention to this topic. Given the large body of mathematical and statistical theory addressing this subject, it is disappointing that even analytical modeling methods have not given this subject much attention.

As a related problem, modeling and simulation approaches often rely on simplified inputs, such as proxy applications or proxy architectures. This simplification is necessary and accepted to handle complex systems with current simulation technology, but it raises the question of how well these proxies represent the initial target application and system and which aspects are or are not covered by the proxy. This is a particular problem for applications with data-dependent behavior. We need tools and techniques to answer these questions, both to ensure that simulation and modeling results are meaningful and to guide future proxy developments.

Need for Better/More Raw Instrumented Data from Systems. At the hardware level, modelers are at the mercy of microprocessor manufacturers for available type and number of on-chip event counters (generally available via the PAPI layer). Typically, the meaning of counters often is poorly documented, and the number of counters are inadequate (i.e., significantly fewer counters than event types), requiring either sampling or multiple runs of the same experiment to get a full set of event data. Newer techniques, such as Intel's Precise Event-based Sampling (PEBS) and RAPL or Advanced Micro Devices' Instruction-Based Sampling (IBS), often are poorly supported in operating systems. In some cases, they also may require root access for use, which is prohibitive on production platforms. In many systems, some metrics (e.g., temperature and power) have no sensors at all, either on-chip or on-blade, requiring owner modification of the system to obtain data, or metrics are handled out of band and cannot be accessed by tools in a way that enables a clean correlation with application execution. Furthermore, there are a number of tools for obtaining these data (e.g., CrayPAT, Open|Speedshop, PMaC, TAU), yet no systematic study has been done to ensure that results are valid and consistent. Studies using counter data often neglect the impact of compiler optimization levels on the results. Software-related metrics, such as those pertaining to MPI message passing, suffer from similar issues. Finally, visualizing, or even summarizing, these data will be difficult because of the vast quantities of results that could be obtained on large core-count systems. Possible solutions include establishing an archive for data that have been validated or developing tools that can create synthetic data based on input parameters.

Integration/Interoperability. Though not the fault of any one tool/simulator, a major barrier to developing more comprehensive and capable models is the lack of integration and interoperability among tools. Various researchers have focused on particular areas, developing innovative modeling approaches and implementing them in special-purpose tools. However, the lack of APIs, or even standard file formats for input and output of results, limits progress toward modeling entire systems. Also, standardization of APIs and model I/O file formats would help with model validation and uncertainty quantification because different models could be run on the same inputs and differences in output could be measured. With respect to file formats for workload description, perhaps the "standard workload format" could serve as a motivational example or starting point (Chapin et al. 1999).

Advanced Analysis and Visualization. Modeling and simulation approaches are capable of providing a vast amount of data, but current techniques for using these data are lacking. To deal with the data explosion, results often are reported as aggregated values only, which can hide important parts of the data. We need new, generalizable approaches, which are not one-off, to capture and analyze modeling and simulation results that can work at scale, enable us to find areas of interest, and visualize the results in an intuitive manner.

Contention Modeling. Contention is a significant aspect in communication performance, both for on- and cross-node traffic. Particularly or future network architectures, we need to understand how node placement and routing algorithm choices affect contention and with performance. However, we currently do not have the means to extract the relevant data from systems or simulators and lack the ability to correlate contention with the causing locations in an application code or message pattern.

Modeling of Accelerators and Heterogeneous Systems. Future extreme-scale machines likely will use some kind of accelerator (GPU, MIC, etc.) or will otherwise be heterogeneous. Modeling and simulation techniques must follow this trend and support the design of such systems, as well as the development of applications for them. In particular, this includes modeling data transfers between accelerator devices and host nodes, optimal mapping of codes to different devices, and partitioning work between multiple devices and a host system.

3.3 Cross-pollination

Modeling and simulation requires input from many sources, particularly from both system (hardware and software) and application designers. Only close interaction within those communities will ensure that models and simulations are relevant to current and future systems and provide useful results. Furthermore, closer collaboration with the data analysis and visualization community would help in extracting insight from any modeling and simulation results. Finally, the modeling and simulation community should develop closer ties within itself. This would aid the development of integrated and/or interoperable tools, which was identified as a significant gap both in this and the previous section regarding interoperability. It also would support creation of standardized or (at least) community-accepted workload definitions and generation tools.

3.4 Path Forward

The discussions on core simulation technologies at ModSim 2014 led to the following recommendations:

- We need interoperable simulation frameworks that are lightweight (by themselves) and enable integration of tools and techniques and various levels of granularity, covering multiple system aspects. This will allow us to close the gap between both individual system simulations and simulations at multiple layers. These efforts should include both the development of standardized APIs and file formats (see Section 3.0), as well as the design and use of continuous integration and testing frameworks to ensure correct interoperability.
- We must develop a taxonomy of modeling and simulation systems that enables us to define and compare their scope. This is a necessary foundation to facilitate the interoperability envisioned in the previous point.
- We must cultivate better end-to-end validation approaches to ensure modeling and simulation results are useful and capture the right system aspects. This includes not only validation of models and simulators, but the validation of proxy applications and systems used as input. Validation efforts also should include the use of techniques for uncertainty quantification to allow for meaningful sensitivity analysis and enable a thorough understanding of the input parameter space and its influence on the results.
- We need to co-design new capabilities for contention measurement in networks to afford a better understanding of this critical performance issue. This is not limited to HPC networks or links within data centers: it is equally important for wide area networks (WANs).

4.0 References

Butts JA and GS Sohi. 2000. “A static power model for architects.” In *Proceedings of the 33rd annual ACM/IEEE International Symposium on Microarchitecture (MICRO 33)*, pp. 191-201. December 10-13, 2000, Monterey, California. Association for Computing Machinery, New York. DOI: 10.1145/360128.360148.

Chapin SJ, W Cirne, DG Feitelson, JP Jones, ST Leutenegger, U Schwiegelshohn, W Smith, and D Talby. 1999. “Benchmarks and Standards for the Evaluation of Parallel Job Schedulers.” In *Job Scheduling Strategies for Parallel Processing*, eds. DG Feitelson and L Rudolph, *Lecture Notes in Computer Science*, vol. 1659, pp. 66-89. Springer-Verlag, Heidelberg, Germany.

Kennedy MC and A O’Hagan. 2001. “Bayesian calibration of computer models.” *Journal of the Royal Statistical Society, Series B* 63(3):425-464. DOI: 10.1111/1467-9868.00294.

Liptay JS. 1968. “Structural Aspects of the System/360 Model 85 II: The Cache.” *IBM Systems Journal* 7(1):15-21. DOI: 10.1147/sj.71.0015.

Moore J, J Chase, P Ranganathan, and R Sharma. 2005. “Making Scheduling ‘Cool’: Temperature-Aware Workload Placement in Data Centers.” In *Proceedings of the 2005 USENIX Annual Technical Conference*, pp. 61-74. April 10-15, 2005, Anaheim, California. USENIX Association, Berkeley, California.

Song WJ, S Mukhopadhyay, and S Yalamanchili. 2014. “Energy Introspector: A Parallel Composable Framework for Integrated Power-Reliability-Thermal Modeling for Multicore Architectures.” In *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. March 23-25, 2014, Monterey, California. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey. Available at: http://casl.gatech.edu/wp-content/uploads/2014/09/song_ispass2014.pdf.

Tanenbaum AS. 1988. *Computer Networks: 2nd Edition*. Prentice-Hall Inc., Upper Saddle River, New Jersey.