Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A Brief History of HPC Simulation and Future Challenges

Kishwar Ahmed, Jason Liu
School of Computing and Information Sciences
Florida International University
Miami, FL 33199, USA

Abdel-Hameed Badawy
Klipsch School of Electrical & Computer Engineering
New Mexico State University
Las Cruces, NM, USA

Stephan Eidenbenz
Information Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, NM, USA

ABSTRACT

High-performance Computing (HPC) systems have gone through many changes during the past two decades in their architectural design to satisfy the increasingly large-scale scientific computing demand. Accurate, fast, and scalable performance models and simulation tools are essential for evaluating alternative architecture design decisions for the massive-scale computing systems. This paper recounts some of the influential work in modeling and simulation for HPC systems and applications, identifies some of the major challenges, and outlines future research directions which we believe are critical to the HPC modeling and simulation community.

1 Introduction

High-performance Computing (HPC) systems today consist of hundreds of thousands of compute nodes, and can perform tens or hundreds of quadrillion floating point operations per second (Tsay 2013, Courtland 2016)). HPC architectures have gone through rapid changes to facilitate the increasing computational demand of scientific applications in many areas, such as astrophysics, particle physics, earth and climate science, computational chemistry, computational biology, and so on. Novel technologies, for example, many-core processors, GPUs, persistent memory, and complex interconnection networks, have been introduced constantly to fulfill the increasing scale and performance of such systems. With the changing hardware architectures also comes the changing software design and implementation of scientific applications in order to take best advantage of the new computing resources.

Modeling and simulation plays an important role for performance prediction and analysis of current and future HPC systems. It can be particularly useful for evaluating the whole-system impact when new components are introduced, for comparing the performance of different system design alternatives, and for locating performance issues of computational code on novel HPC platforms even before their realization. It is thus not surprising to see many HPC models and simulation tools created in the past for exactly the same purposes. They model the HPC systems at different granularity: some are created to study specific components of the HPC systems (processors, memory, interconnect, storage, and so on), and others are meant for studying the overall system performance in aggregate. The important difference lies in the accuracy-performance trade-off that has been applied to effectively capture the salient features of the target system.

A microscopic view allows one to examine detailed operations of a system component, such as the cache performance for branch prediction or out-of-order instructions, and the effect of routing and multiplexing virtual channels on fast interconnection networks. The microscopic model of a component, however, cannot be linearly expanded to the entire HPC system. The cost for simulating at such level of details can be exorbitant, and consequently we can afford only studies limited to a small part of the system and for a short time duration, such as a few microseconds for simulating the execution of the instructions of a program running on a processor. In contrast, a macroscopic view considers the overall design of the HPC architecture and the large effect on the multitude of applications running concurrently. Such models can allow us to perform more longitudinal studies (say, at the job level) of an entire HPC system. However, we have to understand that there is significant risk that such models, due to the high level of modeling abstraction at various parts of the system, can ignore important aspects of the system, which are usually unforeseeable to a large extent before a significant insight of the system is obtained, and as such can render such studies unrealistic and producing simply wrong predictions.

Is it possible to combine the microscopic and macroscopic views of the system so that we can achieve an accurate model of the large-scale HPC systems? What would be a proper trade-off if we can select models with different modeling details? How to select and adjust models so that important aspects of the system (which by itself might be a moving target) can be correctly represented? We do not have an answer to these questions. Even attempting to partially address these questions would require a good understanding of the existing models, together with their advantages and limitations, and also a good understanding of the effect of these models at capturing the prominent behaviors of the target system for any specific study at hand.

In this paper, we intend to provide a historical view on the modeling and simulation efforts of the HPC systems, hoping that once we learn from the prominent existing approaches during the past few decades, we can have a good understanding of the problems and challenges we need to address as a community in the next decade to come. Here, we present a brief survey of the past and present state of modeling and simulation for performance prediction and analysis of HPC systems. More specifically, we review some of the existing models and simulators.

We consider models for different sub-systems, including processors, memory, interconnection networks in Sections 2, 3, and 4, respectively. We also discuss application models that can capture the runtime behavior of large-scale scientific applications in Section 5. Finally and more importantly, we outline research directions in response to the problems and challenges for creating proper HPC models and simulators as we learn from the past efforts. We present our vision for future modeling and simulation of HPC systems as a coordinated community-wide effort in Section 6.

2 Simulation of Processors

The processor architecture in HPC system has gone through perhaps the most rapid changes in recent years. Introduction of multicore and manycore architectures, support for various instruction sets, and the arrival of accelerator technologies (such as GPUs) are some examples of recent changes. Many simulators exist; a few have been proposed recently to incorporate new processor architectures with modified capabilities. They differ mostly in terms of how many instructions can be executed per second, how many cores they can support, and how accurately they can replicate the instruction execution behavior. In this section, we present some of the well-known processor simulators.

Processor simulators have gone from a single processor (core) trace-based, functional, atomic, in-order, and in-order with Cycles Per Instruction (CPI) of one, to out-of-order cycle-accurate multicore simulators. The most famous and most widely used single processor/core simulator is SimpleScalar (Austin et al. 2002). It simulated almost all of the complex interactions a "modern" superscalar processor (at the time) would have from reorder buffer, multiple issue, register renaming, complex branch prediction schemes, caches, and Simultaneous multithreading (Tullsen et al. 1995) (SMT, now known as Hyperthreading in Intel Processors). SimpleScalar implemented Alpha and PISA Instruction Set Architectures (ISAs). After Alpha

died as a commercial processor, Simplescalar suffered the stigma of an outdated ISA even though RISC ISAs were still similar to what Alpha implemented. Depending on which version and which research group used it, the simulator lacked a realistic memory system that simulated contentions on buses connecting the caches to the memory system and the contention in the DRAMs themselves and by the time that was widely available, Chip Multiprocessors (CMPs) now known as multicores had taken the processor world by storm. The advent of such complex processors pushed researchers to implement multicore simulators that spanned a wider range of capabilities. RSIM (Pai et al. 1997) was the only multiprocessor simulator that was publicly available but it was not maintained and thus was outdated by the time multicores were fashionable.

Some of the most known multicore processor simulators that came out publicly were: SIMICS (Magnusson et al. 2002), GEMS (Martin et al. 2005), M5 (Binkert et al. 2006) and lately GEMS merged with M5 generating the gem5 (Binkert et al. 2011). gem5 is a simulator that tries to simulate with varying degrees of accuracy and speed for different components of a multicore system. It has the entire spectrum from atomic cores to cycle-accurate out-of-order cores. It does have a memory subsystem including caches and coherence protocols. It also can accommodate an on-chip interconnection network (i.e., Network on Chip "NoC") model/simulator (e.g., Topaz (Abad et al. 2012) or Garnet (Agarwal et al. 2009)) both independently implemented. One can implement his own coherence protocol and plug it in seamlessly. As we will point out in Section 3, DRAMSim and DRAMSim 2.0 are two examples of how flexible gem5 is. It allows hooking up a detailed cycle-accurate memory system simulator seamlessly. GPGPUs as accelerators that have taken their fair share of publicity also has an existence in gem5 where there is an implementation of a GPGPU integrated into gem5 (called gem5GPGPU (Wang et al. 2012)).

Another aspect of gem5 is that it can perform either in System Call Emulation (SE) mode or in Full System (FS) mode. This last point pertains to the interaction of the programs running on the processor and the operating system (OS). In system call emulation mode, the simulator fakes the existence of an OS by only implementing the minimal set of services that a program needs to run (e.g., reading and writing files). Whereas, in Full System mode, the simulator really boots up an OS (e.g., Linux or Android) and the user gets a command prompt to run his or her program(s) on the simulator. There is a trade-off between these two modes, one gives a more realistic view of the interaction in a real system where the OS interacts and influences the benchmarks/applications (e.g., database applications, OLTP, Data Mining etc.), as opposed to the other applications where such interactions are not present and only the application/benchmark performance is observed (e.g., most scientific applications). Full system mode is far slower than system emulation mode and likewise an out-of-order cycle-accurate core will be far slower than an in-order CPI of one core. At the end of the day, it is a judgment call and an experimental design parameter that has to be made consciously by the research team.

The main advantages of gem5 are the facts that it is a community research project and that it is highly extensible. It does support a variety of instruction sets spanning commercially available CPUs such as x86 and ARM as well as famous outdated essentially extinct ISAs such as SPARC and Alpha. A new branch is implementing the RISCV (Waterman et al. 2011) open-source ISA. gem5 has taken the place of Simplescalar in the processor simulation world where it is the defacto processor simulator.

There are many other recent multicore and manycore simulators. For example, McSimA+ (Ahn et al. 2013) is a lightweight, flexible, open-source simulator with detailed models for the micro-architecture of uni-core, multicore and manycore processors. The overall simulator design is divided into two parts: the (front-end) functional simulation and the (back-end) timing simulation. The functional simulation is based on Pin, a dynamic binary instrumentation tool (Luk et al. 2005) for generating the instruction stream. The timing simulation is based on an event-driven engine, responsible for calculating the correct timing for different operations (such as cache access, and packet traversal). McSimA+ supports simulation of various asymmetric core structures, hierarchical cache architectures (e.g., private, shared, and non-blocking), NoCs (e.g., buses, crossbars, ring, and 2D mesh), memory controller, and main memory. McSimA+ has been shown to achieve good accuracy by comparing with previously published results and with real machine runs.

McSimA+ has demonstrated to be able to scale to a processor with thousands of cores. Like McSimA+, ZSim (Sanchez and Kozyrakis 2013) is another multicore simulator that has been shown to be lightweight, accurate, fast, and scalable. The simulator is lightweight through the use of a user-level virtualization technique. It is accurate for its instruction-driven timing models and leveraging dynamic binary translation. The simulator is fast and scalable, and runs in parallel.

Manifold (Wang et al. 2014) is another multicore simulator. It can support full system simulation, including, for example, operating system and system binaries, and can incorporate various models for transient and steady-state simulation of power, thermal, energy and reliability. It has an open architecture with a component-based design, so that new components can be easily incorporated through community efforts. Manifold uses QSim (Kersey et al. 2012), which is a multicore emulator based on dynamic binary translation. The emulator communicates with the back-end timing model for transparent parallel discrete-event simulation. Experiment has not shown, however, that Manifold can scale to larger systems with more than 64 cores. Also, its performance can only achieve a few thousand instructions per second. This probably can be attributed partially to the fact that the simulator considers full system multicore architecture simulation (including processors, cache, memory). Graphite (Miller et al. 2010) is yet another multicore processor simulator that aims to scale to thousands of cores. One of its main strengths is that it is designed to run in parallel.

Other processor simulators exist such as ESESC (Ardestani and Renau 2013), SimFlex (Wenisch et al. 2006), and MARSS (Patel et al. 2011). Each like the others we introduced thus far has its advantages, disadvantages, limitations and shortcomings. It is the researcher's duty to think carefully about which simulator is the appropriate fit for his/her research project.

Several hardware-based processor simulators have also been proposed in the literature, including those based on FPGAs, such as Ramp Gold (Tan et al. 2010), HAsim (Pellauer et al. 2011), Arete (Khan et al. 2012), and PROTOFLEX (Chung et al. 2007). Hardware-based simulators are much faster than their software-based counterparts. A major challenge, however, is the complexities involved with the simulator development as well as the cost it takes to have a farm of FPGAs to run such simulators on. It would be challenging to incorporate these simulators as a part of an overall system model of an HPC system.

We should make no mistake about the trade-off between accuracy of modeling and the time it takes to get results back. Therefore, each researcher has to make an informed decision about what is critical in his/her particular study and which simulation is appropriate and meaningful.

3 Simulation of Memory

The processor memory has also gone through rapid changes in architectural design for increased capacity and performance. Dongarra et al. (2011) projected that the memory capacity would reach as much as 128 peta bytes, mixing different technologies from DRAM to non-volatile memory with varying performance aspects (in terms of throughput, access latency, etc.) There exist a number of simulators in the literature for modeling the memory system.

Among the early efforts for simulation of memory system, CACTI (Wilton and Jouppi 1996) is perhaps the most versatile tool with capability to model memory hierarchy at various levels: registers, buffers, caches, main memory. Another well-established memory simulator is DRAMsim (Wang et al. 2005). It is an open-source cycle-accurate DRAM simulator, which supports various DRAM types, including SDRAM, DDR, DDR II Memory. The simulator considers various components of DRAM (e.g., DRAM memory controllers, DRAM modules for bits/data storage, and buses through which the DRAM modules communicates), and model them at great detail. It can also model the power consumption of DRAMs. DRAMsim2 (Rosenfeld et al. 2011) is an extension of DRAMsim where it can simulate DDR II and III memory systems. It is important to note that both DRAMsim and DRAMsim2 are publicly available, and they can be incorporated with other simulators (e.g., gem5 as noted in Section 2) as part of a large system simulation framework.

There are also many recently-proposed memory simulators to support simulation of DRAM memory system. For example, USIMM (Chatterjee et al. 2012) is a DRAM main memory system simulator with support for power model based on the Micron power calculator (Janzen, Jeff 2010). The USIMM simulator is capable of working with multiple workload traces, where each trace represent a different program being executed on a different processor. USIMM provides interfaces to trace-based processor model. The simulator supports a number of memory scheduling algorithms proposed in the literature, including FCFS, credit-fair, power-down, close-page, first-ready-round-robin, and MLP-aware. The trace based nature of the simulator is both an advantage and disadvantage at the same time. DrSim (Min Kyu Jeong and Doe Hyun Yoon and Mattan Erez 2012) is another open-source cycle-based DRAM memory system simulator. A main feature of this simulator is its focus on achieving flexibility, which makes it easy to incorporate a variety of DRAM system topologies. Ramulator (Kim et al. 2015) is a recent open-source simulator for current and future DRAM systems. Ramulator aims to be fast, efficient and easily extensible. Currently, it provides cycle-accurate performance models for a variety of DRAM standards (such as DDR III/IV, LPDDR3/4, GDDR5, SALP, AL-DRAM, etc.) Ramulator's memory model has been validated against an actual implementation of DDR3 memory (with Micron's DDR3 Verilog model). Compared to the other simulators thus far, Ramulator can be shown to have achieved the best simulation performance.

4 Simulation of Interconnection Networks

HPC interconnection network offers a systematic way to connect compute nodes, processors, memory, and storage units. Important aspects of an interconnection network model include accurate representation of the network topology, routing, resource scheduling (such as flow control), and network queuing. Different interconnection network topologies exist in current HPC systems, including fat-tree (e.g., IBM's Infiniband), torus (e.g., IBM's Blue Gene/Q, Cray's Gemini), and dragonfly (e.g., Cray's Aries). An accurate model of these interconnection networks is important for us to understand the communication cost as one of the most important constraints on the performance of scientific applications running on HPC systems.

BigSim (Zheng et al. 2004) is an early effort for performance prediction of large-scale parallel machines (e.g., Blue Gene/L machines), based on the model of parallel applications on the target architecture (such as using MPI). The interconnection network models developed in BigSim are relatively simple as they do not consider network congestion (Casanova et al. 2014). BigSim is implemented using Charm++, an object-based and message-driven parallel programming system (Kale and Krishnan 1993). The simulator adopts optimistic parallel simulation for scalability, using inherent determinacy of the target parallel applications to reduce the synchronization overhead. Experiments show that BigSim is capable of scaling up to 64K simulated processors.

Structural Simulation Toolkit (SST) is an all-inclusive simulation framework for modeling large-scale HPC systems, including processors, memory, interconnection networks, and I/O systems (Rodrigues et al. 2011). SST consists of models of various hardware components with different levels of accuracy and granularity, and attempts to achieve scalability by using conservative parallel simulation based on the "distance" between the system components. SST supports generic router models which can be used to build different network topologies, such as binary tree, fat-tree, hypercube, flattened 2D butterfly, 2D and 3D mesh, and fully-connected graph. The interconnection network models in SST, however, do not support flow control between routers and the links between routers are assumed to have infinite capacity. SST is an ongoing project and is able to include active contributions of many advanced component models as part of a scalable and open-source simulation framework.

Extreme-scale Simulator (xSim) is a performance-prediction toolkit for future HPC architectures (Engelmann and Lauer 2010, Böhm and Engelmann 2011). xSim applies parallel discrete-event simulation using lightweight threads and has achieved good scalability with millions of MPI ranks running a simple MPI program. Jones and Engelmann (2011) extended xSim to incorporate a network model with different topologies (star, ring, mesh, torus, and tree), and different hierarchical network combinations (such as

network-on-chip and network-on-node). However, the xSim models do not consider traffic congestion or any detailed blocking behavior which can be important to applications on real interconnection networks.

$\mu\pi$ (Perumalla 2010) is an MPI simulator built upon an efficient conservatively-synchronized parallel simulator that presents a feature-oriented world-view. $\mu\pi$ supports simulation of large-scale MPI applications. Experiments show that it can run up to 27 million virtual MPI ranks on as many as 216,000 cores of a Cray XT5. More recently, Perumalla et. al. Perumalla and Park (2014) proposed to extend $\mu\pi$ and include direct execution to run even larger number of tasks. The simulator focus on MPI communication of applications, but does not contain any detailed interconnection network models.

Co-Design of Exascale Storage System (CODES) simulator is another comprehensive simulation platform to model various large-scale systems, including storage systems, interconnection networks, HPC and data center applications (Cope et al. 2011). CODES provides detailed interconnect models for various interconnect topologies, including torus (Liu and Carothers 2011), dragonfly (Mubarak et al. 2014), and fat-tree (Liu et al. 2015). The simulator is built on Rensselaer Optimistic Simulation System (ROSS), a parallel discrete-event simulation engine using reverse computation (Carothers et al. 2002), and is capable of simulating very large interconnect configurations (with millions of nodes). Trace-driven capabilities have also been added to CODES to replay large execution traces for studying network performance (Acun et al. 2015).

Performance Prediction Toolkit (PPT) is a simulator developed as part of the DOE co-design project that aims at the comprehensive prediction capability for large-scale scientific applications on existing and future HPC architectures. PPT includes a library of hardware models, middleware models, and application models. The interconnection network models in PPT include most common production interconnection networks, such as Cray's Gemini and Aries, IBM's Infiniband and Blue Gene/Q networks (Ahmed et al. 2016, Ahmed et al. 2016). These models have been carefully validated against empirical studies. The interconnection model is fully integrated with an implementation of Message Passing Interface (MPI), which is capable of capturing the most common interactions between parallel applications. PPT is built on top of Simian, which is a fast parallel discrete-event simulation engine developed with interpreted languages, including Python, LUA, and Javascript (Santhi et al. 2015). As such, one can find it relatively easy to integrate other models for a large-scale HPC simulation without any significant performance loss.

Garnet (Agarwal et al. 2009) is an on-chip interconnection network simulator, which builds upon the lacking of GEMS (Martin et al. 2005) and performs "detailed" interconnect communication between on-chip routers. Through modeling detailed router microarchitecture, Garnet is able to capture various details such as virtual channel arbitration and realistic link contention. Garnet has provision for easy-configuration of various parameters (e.g., different network topologies, interconnect bandwidth configuration through flit size, router parameters such as arbitrary number of input and output ports, various routing algorithms). TOPAZ (Abad et al. 2012) is yet another open-source NoC simulator with broader analysis spectrum (e.g., tradeoffs between accuracy, simulation speed for various interconnection network). It has an easy incorporation capability with other simulation tools (e.g., GEMS and gem5) in runtime. TOPAZ is multithreaded, therefore providing more accurate prediction capability without compromising simulation execution time. In addition to being capable of simulating NoCs in multicore processors, TOPAZ also supports simulation of large-scale interconnection networks and is able to simulate networks consisting of millions of routers.

5 Modeling HPC Applications

HPC applications exhibit distinct behaviors. An accurate model for these applications is crucial for determining the impact of technological advances in novel HPC architectures. Some of the HPC applications are data-intensive (e.g., molecular dynamics simulation and computational fluid dynamics applications). Some are communication-intensive (e.g., NERSC MIMD Lattice Computation application and NAS parallel benchmark applications). And yet others are I/O-intensive. There exist many application models and analysis tools in the literature to describe the behavior of the HPC applications with various details.

Vampir (Knüpfer et al. 2008) is a performance analysis tool for parallel MPI/OpenMPI applications. Vampir consists of two major components: a runtime instrumentation and measurement system and a visual analysis tool. The former supports program instrumentation in different programming languages. It also supports different types of programs: sequential programs, MPI programs, OpenMP programs, and hybrid MPI and OpenMP programs. Vampir also supports various types of instrumentation, including compiler instrumentation, library instrumentation, and manual instrumentation. It provides runtime measurement capability to capture dynamic application behaviors (such as application's memory usage, I/O performance, user-defined performance counters, etc.)

Tuning and Analysis Utilities (TAU) is a well-established, flexible, portable, robust performance instrumentation, measurement, analysis and visualization framework for HPC applications (Shende and Malony 2006). TAU provides flexible instrumentation capability, allowing the user to select performance instrumentation at different levels of application code. The instrumentation provides various performance information, including various system events and user-defined events, which can be later used for profiling and tracing. Similar to Vampir, TAU supports various types of instrumentation inside a program, such as source-based instrumentation, preprocessor-based, compiler-based, and interpreter-based instrumentation, etc. HPCTOOLKIT (Adhianto et al. 2010) is another application performance measurement, analysis, and presentation toolkit for both sequential and parallel applications. Instead of using source code instrumentation, HPCTOOLKIT works directly with application binaries. HPCTOOLKIT provides effective application analysis by providing measurement ability for a number of derived performance metrics (e.g., peak and actual performance difference rather than simple raw data such as operation counts). HPCTOOLKIT's graphical user interface carries out the computation of these derived metrics.

The above tools can be used for measuring and displaying the runtime performance of specific applications. There also exist several analytical models to capture high-level performance.

Performance and Architecture Lab Modeling Tool (Palm) is an analytical performance model for parallel applications (Tallent and Hoisie 2014). Palm performs static and dynamic analysis of the source code and generates a tree-like hierarchical data structure following some well-defined rules. Palm combines top-down semantic insight and bottom-up static and dynamic analysis capability for parallel application execution. Aspen (Spafford and Vetter 2012) is a domain-specific language for analytical performance modeling to enable exploration of novel algorithms and architectures. A formal definition in Aspen includes application behavior (e.g., parameters, kernels, control flow) and the abstract machine (e.g., node, interconnect, cache, memory, core).

6 Future Challenges of HPC Modeling and Simulation

Although there exist plenty of models and simulators that focus on different aspects of the high-performance computing systems with varying degree of granularity, accuracy, effectiveness, performance and scalability, plenty of challenges remain to prevent them from becoming more widely used and accepted. The field of HPC modeling and simulation for performance prediction and analysis will need to transform itself. As a vision to aspire to we conjure a comprehensive, open-source performance prediction capability with easy-to-use and mutually compatible models for (i) every past and present hardware technology as well as for newly emerging hardware concepts, (ii) every past and present system software components including middleware technologies, such as MPI, and emerging concepts, such as task-based runtimes, and (iii) every major category of application software from all application areas that rely on high-performance computing, including but not limited to the physics, chemistry, biology, operations research, optimization, and finance.

Our assessment of the present state of the field in the previous sections is that we are still in a "wild west" stage of development: a few individual building blocks for hardware, middleware, and software models exist. Mostly, they are not compatible with each other. Some are open-source, but most are closed-source. The nascent community attempts to keep up with the pace of development on the hardware and to a lesser extent the software side, but is mostly viewed as an after-thought. Modeling new concepts before implementation is the exception, and implementing models after technologies have been introduced is the

norm. The potential value of comprehensive models has not been realized, as the modeling community appears to play catch-up game with the actual hardware and software trends as opposed of driving the trends and allowing for early, cost-efficient assessment of novel ideas.

In order to move from our present "wild west" and "anything goes" stage of evolution, we propose the following five-step plan. Our recipe proposed here serves only to open the discussion. Success will come only with a growing community that agrees on a few base rules of engagement.

Establish clearly defined use cases. A simulation and modeling capability needs to have a set of widely applicable and accepted use cases. In our opinion, these use cases should include the following. (i) Early assessment of hardware technologies and concepts, such as new caching strategies or speculative execution methods. This practice is actually done widely already in the architecture community. (ii) Early assessment of algorithmic variations for both middleware software and application software. For instance, the basic functionality of task-based parallelism runtimes, such as HPX or Legion, should be tested before many years of implementation work go into developing an actual software capability. Similarly, algorithmic variations of large computational physics codes, such as accelerated molecular dynamics, should be implemented as application simulations and tested before full-fledged implementation, such as the work done for Speculative Temperature Adjusted Dynamics (Zamora et al. 2016). (iii) Assessment of procurement bids for new HPC systems begs for the use of simulation modeling. The state of the practice in large-scale system procurement consists of expert opinions from both buyers and sellers. While the process often works, there have been issues in the past. Reliance on models in addition to expert opinion has the potential to remove biases. (iv) Bottleneck resource identification explores a hardware and software design space by performing sensitivity analyses across parameters. On the hardware side, this could include increasing and decreasing cache sizes for instances or clock frequencies. The gradient of the search space will indicate which technical improvement direction in hardware would be most beneficial to pursue for the given set of software applications.

Use cases need to come with an appreciation that choosing the appropriate level of detail is essential: pick a model resolution that explicitly models the functionality of the suspected bottleneck resource. In reality, there will be a trade-off between model scalability and accuracy. Use cases need to find a well-established balance in this trade-off space.

Agree on a single tool or ensure composability. The most successful modeling communities agree on a dominant tool and start building up its capabilities as a community effort. The communication network simulation community is a perhaps the prime example with ns-2 (and now increasingly ns-3) as the dominant tool of choice for most users. An alternative model is to agree on a communication API for different simulation components, resulting in federated models. The venerable HLA—tolerated by many, beloved by few—still sets the standard for this approach. As a pragmatic approach, we could see the three main communities of application and middleware software models, interconnect models, and compute node models establish interfaces between each other that are commonly accepted in an HLA-like fashion. Within each of the three communities, however, a single dominant tool should emerge. Unfortunately, federated systems rarely scale well, thus leaving room for well-integrated comprehensive tools.

Build and maintain a comprehensive model library of all hardware and software components. The HPC performance prediction community needs to focus its resources on building comprehensive easy-to-use libraries that allow a non-expert user to quickly build a composed model of hardware and software components to test. Once established with a reasonably large user base, the community can then shift into a much more manageable maintenance mode, where emerging technologies can be quickly modeled and assessed. The architecture community partially operates in this fashion already, the wireless communication network community used this model very successfully to test protocol ideas before testbeds were widely available. Credibility of models needs to be established through validation runs whenever possible.

Ensure reproducibility. The simulation community in general needs to ensure that results are reproducible. The existence of standard input formats and a single or few simulation tools should make such reproducibility technically simple. The community culture should demand reproducible descriptions. In the

end, any tool is a model of reality, usually an approximation. A certain diversity of tools can be desirable to reproduce results across different tools. If results should hold across multiple tools, their credibility increases. Learning again from the network simulation community, an over-reliance on a single simulator at times could lead to reproducible results of protocol performance that all used the same bug, which was discovered years later.

Going beyond our current vision, once our performance prediction vision has been realized with a strong, thriving community, it should aim to expand its reaches beyond the traditional HPC architectures into related fields. In particular, novel computing techniques, such as quantum computing, neuromorphic computing, inexact computing, or even application-specific integrated circuits (ASIC) or co-processors should be modeled for potential performance gains before they become widely available.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their feedback that improved the paper significantly. This research is supported in part by the NSF grant CNS-1563883, the DOD grant W911NF-13-1-0157, the DOE LANL LDRD Program, and a USF/FC2 SEED grant. This work was also performed under the auspices of the US Government contract DE-AC52-06NA25396 for Los Alamos National Laboratory, operated by Los Alamos National Security, LLC, for the US Department of Energy (LA-UR-17-24198).

REFERENCES

Abad, P., P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J.-A. Gregorio. 2012. "Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers". In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, 99–106. IEEE.

Acun, B., N. Jain, A. Bhatele, M. Mubarak, C. D. Carothers, and L. V. Kale. 2015. "Preliminary evaluation of a parallel trace replay tool for HPC network simulations". In *Euro-Par 2015: Parallel Processing Workshops*, 417–429.

Adhianto, L., S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. 2010. "HPCToolkit: Tools for performance analysis of optimized parallel programs". *Concurrency and Computation: Practice and Experience* 22 (6): 685–701.

Agarwal, N., T. Krishna, L.-S. Peh, and N. K. Jha. 2009. "GARNET: A detailed on-chip network model inside a full-system simulator". In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 33–42. IEEE.

Ahmed, K., J. Liu, S. Eidenbenz, and J. Zerr. 2016, Dec. "Scalable Interconnection Network Models for Rapid Performance Prediction of HPC Applications". In *2016 IEEE 18th International Conference on High Performance Computing and Communications (HPCC)*, 1069–1078.

Ahmed, K., M. Obaida, J. Liu, S. Eidenbenz, N. Santhi, and G. Chapuis. 2016. "An Integrated Interconnection Network Model for Large-Scale Performance Prediction". In *Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*, 177–187.

Ahn, J. H., S. Li, O. Seongil, and N. P. Jouppi. 2013. "McSimA+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling". In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, 74–85.

Ardestani, E. K., and J. Renau. 2013. "ESESC: A fast multicore simulator using time-based sampling". In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, 448–459. IEEE.

Austin, T., E. Larson, and D. Ernst. 2002. "SimpleScalar: An infrastructure for computer system modeling". *Computer* 35 (2): 59–67.

Binkert, N., B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti et al. 2011. "The gem5 simulator". *ACM SIGARCH Computer Architecture News* 39 (2): 1–7.

Binkert, N. L., R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. 2006. "The M5 simulator: Modeling networked systems". *IEEE Micro* (4): 52–60.

Böhm, S., and C. Engelmann. 2011. "xSim: The extreme-scale simulator". In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, 280–286.

Carothers, C. D., D. Bauer, and S. Pearce. 2002. "ROSS: A high-performance, low-memory, modular Time Warp system". *Journal of Parallel and Distributed Computing* 62 (11): 1648–1669.

Casanova, H., A. Giersch, A. Legrand, M. Quinson, and F. Suter. 2014. "Versatile, scalable, and accurate simulation of distributed applications and platforms". *Journal of Parallel and Distributed Computing* 74 (10): 2899–2917.

Chatterjee, N., R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi, and Z. Chishti. 2012. "Usimm: the utah simulated memory module". *University of Utah, Tech. Rep.*

Chung, E. S., E. Nurvitadhi, J. C. Hoe, B. Falsafi, and K. Mai. 2007. "PROToFLEX: FPGA-accelerated hybrid functional simulator". In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 1–6.

Cope, J., N. Liu, S. Lang, P. Carns, C. Carothers, and R. Ross. 2011. "Codes: Enabling co-design of multilayer exascale storage architectures". In *Proceedings of the Workshop on Emerging Supercomputing Technologies*, 303–312.

Courtland, R. 2016. "China inches toward the exascale[News]". *IEEE Spectrum* 53 (8): 14–15.

Dongarra, J., P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig et al. 2011. "The international exascale software project roadmap". *International Journal of High Performance Computing Applications* 25 (1): 3–60.

Engelmann, C., and F. Lauer. 2010. "Facilitating co-design for extreme-scale systems through lightweight simulation". In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, 1–8.

Janzen, Jeff 2010. "The Micron system-power calculator". https://www.micron.com/support/tools-and-utilities/power-calc.

Min Kyu Jeong and Doe Hyun Yoon and Mattan Erez 2012. "DrSim: A Platform for Flexible DRAM System Research". http://lph.ece.utexas.edu/public/DrSim.

Jones, I. S., and C. Engelmann. 2011. "Simulation of large-scale HPC architectures". In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, 447–456.

Kale, L. V., and S. Krishnan. 1993. "CHARM++: a portable concurrent object oriented system based on C++". In *ACM Sigplan Notices*, Volume 28, 91–108.

Kersey, C. D., A. Rodrigues, and S. Yalamanchili. 2012. "A universal parallel front-end for execution driven microarchitecture simulation". In *Proceedings of the 2012 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, 25–32.

Khan, A., M. Vijayaraghavan, S. Boyd-Wickizer et al. 2012. "Fast and cycle-accurate modeling of a multicore processor". In *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, 178–187.

Kim, Y., W. Yang, and O. Mutlu. 2015. "Ramulator: A fast and extensible DRAM simulator". *IEEE Computer Architecture Letters*.

Knüpfer, A., H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel. 2008. "The vampir performance analysis tool-set". In *Tools for High Performance Computing*, 139–155.

Liu, N., and C. D. Carothers. 2011. "Modeling billion-node torus networks using massively parallel discrete-event simulation". In *Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*, 1–8.

Liu, N., A. Haider, X.-H. Sun, and D. Jin. 2015. "FatTreeSim: Modeling Large-scale Fat-Tree Networks for HPC Systems and Data Centers Using Parallel and Discrete Event Simulation". In *Proceedings of the 3rd ACM Conference on SIGSIM-Principles of Advanced Discrete Simulation*, 199–210.

Luk, C.-K., R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. 2005. "Pin: building customized program analysis tools with dynamic instrumentation". In *Acm sigplan notices*, Volume 40, 190–200.

Magnusson, P. S., M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. 2002. "Simics: A full system simulation platform". *Computer* 35 (2): 50–58.

Martin, M. M., D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. 2005. "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset". *ACM SIGARCH Computer Architecture News* 33 (4): 92–99.

Miller, J. E., H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. 2010. "Graphite: A distributed parallel simulator for multicores". In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, 1–12. IEEE.

Mubarak, M., C. D. Carothers, R. B. Ross, and P. Carns. 2014. "Using massively parallel simulation for MPI collective communication modeling in extreme-scale networks". In *Simulation Conference (WSC), 2014 Winter*, 3107–3118.

Pai, V. S., P. Ranganathan, and S. V. Adve. 1997. "RSIM: An execution-driven simulator for ILP-based shared-memory multiprocessors and uniprocessors". In *Proceedings of the Third Workshop on Computer Architecture Education*, Volume 178. Citeseer.

Patel, A., F. Afram, S. Chen, and K. Ghose. 2011. "MARSS: a full system simulator for multicore x86 CPUs". In *Proceedings of the 48th Design Automation Conference*, 1050–1055. ACM.

Pellauer, M., M. Adler, M. Kinsy, A. Parashar, and J. Emer. 2011. "HAsim: FPGA-based high-detail multicore simulation using time-division multiplexing". In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, 406–417.

Perumalla, K. S. 2010. "$\mu\pi$: a scalable and transparent system for simulating MPI programs". In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 62.

Perumalla, K. S., and A. J. Park. 2014. "Simulating billion-task parallel programs". In *Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014), International Symposium on*, 585–592.

Rodrigues, A. F., K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls et al. 2011. "The structural simulation toolkit". *ACM SIGMETRICS Performance Evaluation Review* 38 (4): 37–42.

Rosenfeld, P., E. Cooper-Balis, and B. Jacob. 2011. "DRAMSim2: A cycle accurate memory system simulator". *Computer Architecture Letters* 10 (1): 16–19.

Sanchez, D., and C. Kozyrakis. 2013. "ZSim: fast and accurate microarchitectural simulation of thousand-core systems". In *ACM SIGARCH Computer Architecture News*, Volume 41, 475–486.

Santhi, N., S. Eidenzenz, and J. Liu. 2015. "The Simian concept: parallel discrete event simulation with interpreted languages". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti.

Shende, S. S., and A. D. Malony. 2006. "The TAU parallel performance system". *International Journal of High Performance Computing Applications* 20 (2): 287–311.

Spafford, K. L., and J. S. Vetter. 2012. "Aspen: a domain specific language for performance modeling". In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 84.

Tallent, N. R., and A. Hoisie. 2014. "Palm: easing the burden of analytical performance modeling". In *Proceedings of the 28th ACM international conference on Supercomputing*, 221–230.

Tan, Z., A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanović. 2010. "RAMP gold: an FPGA-based architecture simulator for multiprocessors". In *Proceedings of the 47th Design Automation Conference*, 463–468.

Tsay, B. 2013. "The Tianhe-2 Supercomputer Less than Meets the Eye". *SITC Bulletin Analysis*.

Tullsen, D. M., S. J. Eggers, and H. M. Levy. 1995. "Simultaneous multithreading: Maximizing on-chip parallelism". In *ACM SIGARCH Computer Architecture News*, Volume 23, 392–403. ACM.

Wang, D., B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob. 2005. "DRAMsim: a memory system simulator". *ACM SIGARCH Computer Architecture News* 33 (4): 100–107.

Wang, H., V. Sathish, R. Singh, M. J. Schulte, and N. S. Kim. 2012. "Workload and power budget partitioning for single-chip heterogeneous processors". In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, 401–410. ACM.

Wang, J., J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao et al. 2014. "Manifold: A parallel simulation framework for multicore systems". In *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, 106–115.

Waterman, A., Y. Lee, D. A. Patterson, and K. Asanovic. 2011. "The risc-v instruction set manual, volume i: Base user-level isa". *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62*.

Wenisch, T. F., R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. 2006. "SimFlex: statistical sampling of computer system simulation". *Micro, IEEE* 26 (4): 18–31.

Wilton, S. J., and N. P. Jouppi. 1996. "CACTI: An enhanced cache access and cycle time model". *Solid-State Circuits, IEEE Journal of* 31 (5): 677–688.

Zamora, R. J., A. F. Voter, D. Perez, N. Santhi, S. M. Mniszewski, S. Thulasidasan, and S. J. Eidenbenz. 2016. "Discrete event performance prediction of speculatively parallel temperature-accelerated dynamics". *Simulation* 92 (12): 1065–1086.

Zheng, G., G. Kakulapati, and L. V. Kalé. 2004. "Bigsim: A parallel simulator for performance prediction of extremely large parallel machines". In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 78.

AUTHOR BIOGRAPHIES

KISHWAR AHMED is a Ph.D. candidate at the School of Computing and Information Sciences, Florida International University. He received his B.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology. His research interests include high-performance computing, performance prediction, and optimization. His email address is kahme006@cis.fiu.edu.

JASON LIU is an Associate Professor at the School of Computing and Information Sciences, Florida International University. He received a Ph.D. degree from Dartmouth College in Computer Science. His research focuses on parallel simulation and high-performance modeling of computer systems and communication networks. His email address is liux@cis.fiu.edu.

ABDEL-HAMEED BADAWY is an Assistant Professor at the Klipsch School of Electrical and Computer Engineering, New Mexico State University. He received a Ph.D. degree from the University of Maryland in Computer Engineering. His research interests include computer architecture, performance modeling, compiler-architecture interactions, and high performance computing. His email address is badawy@nmsu.edu.

STEPHAN EIDENBENZ is the Director of the Information Science and Technology (ISTI) institute at Los Alamos National Laboratory. He obtained a PhD from the Swiss Federal Institute of Technology, Zurich (ETHZ) in Computer Science. His research interests include cyber security, computational codesign, communication networks, scalable modeling and simulation, and theoretical computer science. His email address is eidenben@lanl.gov.