

Architectural Simulation for Exascale Hardware/Software Co-design

Curtis Janssen[†], Dan Quinlan[§], John Shalf^{*}

^{*}*National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Laboratory, Berkeley, CA, USA*

[†]*Scalable Modeling and Analysis Department, Sandia National Laboratories, Livermore, CA, USA*

[§]*Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, CA, USA*

Abstract—The rapid and disruptive changes anticipated in hardware design over this next decade necessitate a more agile development process, such as the hardware-software co-design processes developed for rapid product development in the embedded space. This article will describe the structure of the co-design process as applied to supercomputing systems, introduce the role of architectural simulation and code analysis to enable co-design, and describe the CoDEX project that is developing tools to accelerate the iterative co-design cycle for the DOE exascale computing program.

I. INTRODUCTION

Over the years users of HPC systems have observed dramatic increases in claimed peak performance without the commensurate improvements in effective performance delivered to applications. Furthermore, as we approach exaflop-scale HPC systems, we are confronted by their anticipated enormous electrical power requirements. The cost of power is expected to exceed the procurement costs of such systems, and will thus ultimately limit the practicality of future state-of-the-art HPC platforms. These trends will lead to a crisis in HPC in the not-too-distant future, unless we work aggressively with vendors and the scientific community to develop more energy-efficient solutions. HPC system design methodologies on which we have relied so far never had to consider power constraints or parallelism of the scale being contemplated for exascale systems. Furthermore, the programming model and software environment anticipated for future extreme-scale systems is anticipated to be substantially different than the current practice. The designers of HPC hardware and software components have an urgent need for a systematic design methodology that reflects future design concerns and constraints.

The Department of Energy’s Exascale Computing initiative has identified hardware/software co-design as a central strategy to overcome this concerning situation. The strategy is based on development partnerships with computer vendors to engage application scientists to participate in a highly collaborative and iterative design process well before a given system is available for commercial use. The process is built around identifying leading edge, high-impact scientific applications providing concrete optimization targets rather than focusing on speeds and feeds (FLOPs and bandwidth) and percent of peak. Rather than ask “what kind of scientific applications can run on an exascale system after it arrives,” this application-driven design process instead asks “what kind of system

should be built to meet the needs of the most important science problems.” This leverages DOE’s deep understanding of application requirements and broad-based computational science portfolio.

Whereas a traditional pipelined design optimization process primarily optimizes isolated metrics that are specific to individual teams, the co-design process requires collaboration and compromise from all teams to achieve the common optimization target. The integrated design teams formed for co-design will include application experts working in concert with computer architects and algorithm designers, all using sophisticated tools to accelerate the design loop. The methodology depends on a bi-directional optimization of design parameters where software requirements drive hardware design decisions and hardware design constraints motivate changes in the software design to better fit within those constraints. Deep analysis of application requirements drives key design decisions for the overall system architecture. The analysis provides quantitative measures of application requirements and relates them back to architectural parameters such as on-chip memory, memory bandwidth requirements and interconnect requirements. However, there are many hardware design choices that are too costly to implement or do not improve energy efficiency enough to justify the cost, which motivate changes to the software implementation. Cycle accurate simulation tools enable quantification of the performance impact on the applications when they are subjected to specific hardware constraints. Analysis of the hardware constraints, in terms of cost, area, and power consumption, is fed back to the application to motivate changes in the application design and algorithms to better fit within hardware constraints. An accelerated iterative design cycle will enable orders of magnitude improvement in energy efficiency and usable performance.

We are assembling a comprehensive hardware/software co-design environment, called CoDEX (Co-Design for Exascale) that will enable an unprecedented opportunity for application and algorithm developers to influence the direction of future architectures so that they meet DOE mission needs. CoDEX combines highly-configurable, cycle accurate simulation of node architectures, developed through the LBNL Green Flash project, with novel automatic extraction and exascale extrapolation of memory and interconnect traces using the LLNL ROSE compiler framework, and scalable simulation of massive interconnection networks using the Sandia-developed

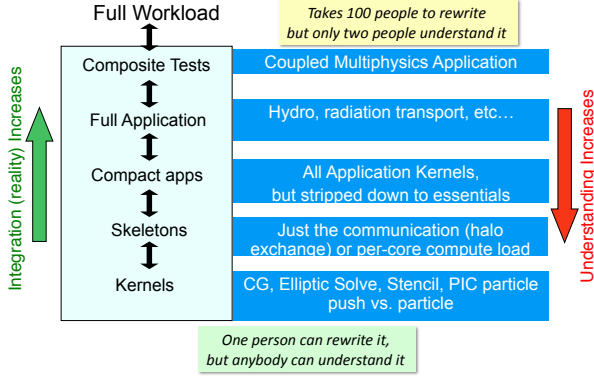


Fig. 1. Shows interplay between the hierarchy of reduced applications that we employ in the co-design process. Kernels enable rapid exploration of new languages and algorithms because of ease of rewriting while full applications are harder to rewrite, but ensure adherence to original application requirements.

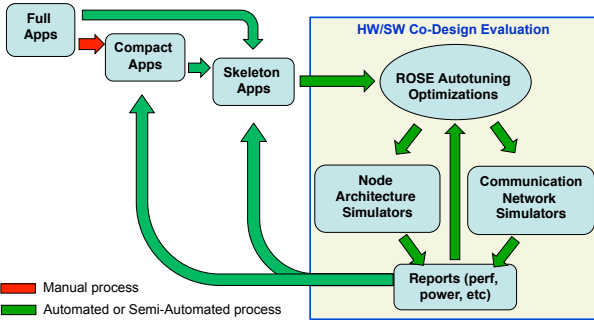


Fig. 2. The co-design process employs the hierarchy of simplified surrogate code representations to provide actionable detailed information that hardware designers need, but ensure the context for those insights is still faithful to the requirements of the full-application.

SST/macro coarse-grained simulator. These tools will enable a tightly-coupled software/hardware co-design process can be applied effectively to the complex HPC application space.

II. TOOLS TO ENABLE CO-DESIGN

A successful co-design process requires models for both hardware and software components of an exascale system which can be employed in a co-design process. The software models consist of a variety of surrogates for the application software. Different types of surrogates are required to implement different aspects of the co-design process. Table I shows several surrogates and their definitions. Figure 1 shows the role of the hierarchy of reduced applications in code analysis while Figure 2 shows the co-design workflow using these different application representations. Among these, compact and mini-applications as well as mini-drivers and kernels run on actual hardware, while skeleton applications are explicitly designed to run only in a simulated hardware environment. Tools that allow automated conversion between these surrogates are a key feature of the CoDEX environment and are discussed in the next section.

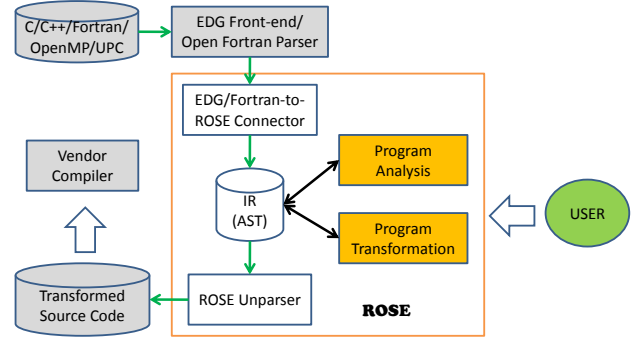


Fig. 3. Overview of how ROSE based tools are constructed and used to perform analysis and transformations to build skeletons of input code and feed them to other tools (e.g. vendor compilers)

A. Tools for Deep Code Analysis

The ROSE compiler infrastructure [7] forms a central part of the tool chain for automating the extraction of skeleton applications to support the inputs to simulators for co-design. Starting with a source code application a specialized tool (shown in figure 3), built using ROSE, reads the source code into an Abstract Syntax Tree (AST) using an intermediate representation. This tool then performs analysis of the source code (indirectly using the AST), and then transforms the AST to cut away parts of the application’s representation to reduce the complexity but preserve specific features (e.g. MPI communication). From the modified AST a new less complex version of the source code can be generated; this new version is the skeleton suitable for input into hardware simulators. Research work to support this work has been a collaboration with both the ROSE project at LLNL and Galois Inc. Existing and future work is released iteratively as part of the ROSE project at www.roseCompiler.org.

Significant internal compiler analysis is required to support this work. Current work has used data dependence and control flow information (def-use analysis) to support generation of backward slices rooted in MPI calls in the application. Future work will expand on this to support generation of skeletons that capture a range of behaviors separate from just MPI communication. Future work will also incorporate interprocedural analysis based on the System Dependence Graph (SDG) to track usage of variables across function interfaces and provide more aggressive levels of simplification of large scale applications.

These techniques to automate generation of skeleton applications are central to how co-design can incorporate current scientific software into the process of evaluating modern numerical techniques (see figure 2) on proposed future exascale architectures.

B. Coarse-grained Interconnect Simulations

Architecture simulation provides an avenue to experiment with hardware configurations, programming models, and algorithms on exascale class machines before full implementations of the hardware and software components of a system are available. Models of the hardware, the runtime system, and the application can be integrated into to the simulation to

Surrogate	Description
Compact app.	Small app. having fewer features and simplified boundary conditions relative to full apps.
Mini-app.	Small, self-contained program that embodies essential performance characteristics of key applications.
Skeleton app.	Captures control flow and communication pattern of app. Can only be run in a simulator.
Proxy app.	General term for all the above "app" approaches.
Mini-driver	Small programs that act as drivers of performance-impacting library packages.
Kernel	Captures node-level aspects of an algorithm

TABLE I
APPLICATION SURROGATES AND THEIR DEFINITIONS.

evaluate the trade-offs between design choices imposed on each aspect of the integrated system. Coarsened-grained simulations allow large-scale systems to be studied in a way that captures the complex interactions between various hardware components, including those interactions which only arise at the largest scales. Examples of simulators specifically designed for studying exascale class machines include the Structural Simulation Toolkit macroscale components (SST/macro) [4] and the Warwick Performance Prediction (WARPP) simulator [3]. Here we focus on the SST/macro simulator. The simulator software is modular, permitting the study of various computation and communication models. SST/macro is distributed under an open-source license and can be downloaded from the internet [8].

There are two primary techniques for driving the simulator: a trace of a previously run MPI application can be replayed through the simulator, allowing its execution time to be estimated on new hardware, or a skeleton-application [1], [9] can be provided which mimics the control-flow and messaging pattern of a real application with the computational and message passing costs replaced by values obtained from performance models. The tracing approach provides a easy way to understand existing application performance on new architectures, while the skeleton application approach allows studies at extreme scales and permits new programming model ideas and notional algorithms to be evaluated without the need for a complete implementation.

C. Rapid Design Synthesis

The embedded processor market has refined co-design processes by developing tools that make hardware-software co-design productive, cost-effective, and beneficial (see Sidebar "Co-Design in Embedded Systems"). We have adopted many of the tools designed for rapid synthesis of embedded designs and have extended them so that they can be used for the exploration of the HPC design space. One key tool that we use is the Tensilica Inc. Xtensa Processor Generator (XPG) toolchain, which provides an end-to-end solution for quickly creating simple, semi-custom processors out of optimized, power efficient building blocks. The customizable instruction set, communication interfaces, and memory hierarchy of the Xtensa design tools makes it ideally suited for exploration of novel chip multiprocessor designs. In combination with the ability to extend the instruction set to add application specific functionality produces a streamlined processor with scratchpad memories, advanced communication features, and

custom opcodes to facilitate advanced communication and synchronization as part of our design space exploration. XPG's ability to automatically generate C/C++ compilers, debuggers, and functional models enables fast software porting and testing for a new architecture. It also speeds language development by enabling us to target source-to-source translation technology (targeting the compilers that are automatically generated by Tensilica), which is substantially easier than crafting a new compiler from scratch.

D. FPGA Accelerated Hardware Emulation

FPGA-based architectural emulation platforms are commonly used in the industry for the rapid prototyping and evaluation of ASIC designs. Design synthesis tools produce a list of gates with the circuit connections between them called "gate level Register Transfer Language" (gate-level RTL). This can be laid out into a chip design, which in turn can be mass produced at a chip fab. However, you can instead take the gate-level RTL for a potential processor design and load it onto an FPGA for full cycle-accurate emulation. Once the gateware components are mapped onto the FPGA's, the resulting system looks, for all practical purposes, like the actual hardware, except that it runs at a much lower clock frequency. The emulated system can boot conventional OS's such as Linux and use actual compilers that are targeted at the final development platform. This is hundreds to thousands of times faster than typical software emulation environments.

This speed advantage does not come at the expense of accuracy; to the contrary, FPGA emulation is arguably much more accurate than a software simulation environment as it truly represents the hardware design. The direct mapping to FPGAs on the hardware emulation platform and copious performance data provide a fast, accurate performance emulation environment allowing the benchmarking of real codes ensuring the application developers are intimately involved in the co-design process. Lastly, the gate-level RTL design also provides accurate feedback on the power consumption of the design because it is the actual gates that would be used to produce the *real* chip design. In contrast, power estimation much more difficult to verify for a software-only simulation environment.

For our work, we have adopted the Research Accelerator for Multi-Processors (RAMP) is a cooperative effort between six universities to build a new standard emulation system for parallel processors. RAMP emulation technology uses the Berkeley Emulation Engine 3 (BEE3) boards, using Xilinx FPGAs. Using inter-FPGA links between FPGAs on the same

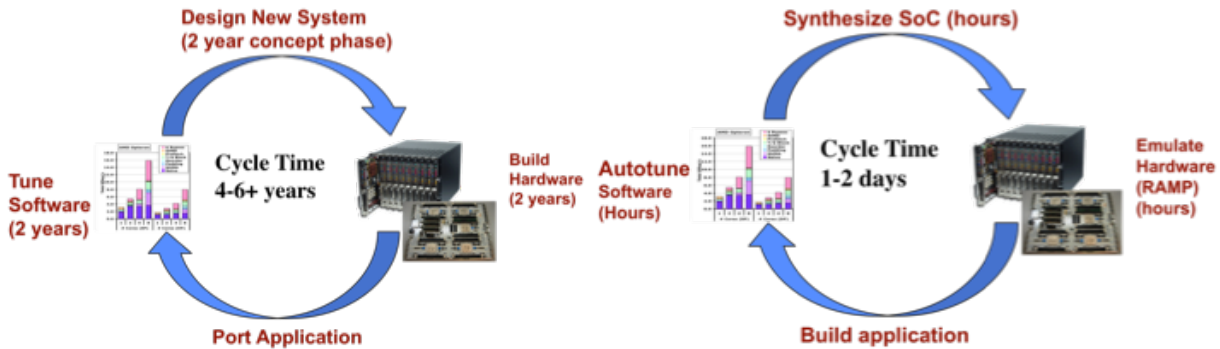


Fig. 4. Conventional design cycles last 4-6 years. With rapid synthesis tools to generate prototype designs, FPGA-accelerated emulation and software auto-tuning, we hope to get the co-design cycle time down to 1-2 days.

board, and between BEE boards, once can scale up simulations to simulate a single manycore socket, or even larger-scale clusters. One recent example was demonstrated by the Berkeley RAMP Blue project where over 1,000 cores were emulated using a stack of sixteen BEE2 boards.

With FPGA-accelerated emulation capabilities, one can generate a new system design (ie. “tape out”) every day and the “build time” of a new system is measured in minutes rather than years. This accelerates the rate that we can evaluate alternative design scenarios in comparison a typical design cycle that takes months or even years using conventional development processes. Overall, the performance and increased confidence in the timing model of FPGA-accelerated hardware emulation platforms enhance interaction at all levels of system development. These capabilities can facilitate and ultimately accelerate the iterative process of science-driven system design (see Figure 4).

III. HARDWARE/SOFTWARE CO-DESIGN STRATEGY

Co-design also depends on integrated design teams that include both application experts working in concert with computer architects and algorithm designers and sophisticated tools to accelerate the design loop. The optimization target for the process is the delivered application performance for the combined hardware and software environment, rather than tangentially related metrics such as *peak flops* or bytes-to-flop ratios. Therefore, it is essential to clearly identify the application up-front to act as the common metric for success for all aspects of the system design during the iterative co-design process.

The iterative design methodology shown in Figure 4 depends on a bi-directional optimization of design parameters where software requirements drive hardware design decisions and hardware design constraints motivate changes in the software design to better fit within those constraints. Deep analysis of application requirements drives key design decisions for the overall system architecture. The analysis provides quantitative measures of application requirements and relates them back to architectural parameters such as on-chip memory, memory bandwidth requirements and interconnect requirements. However, there are many design choices that are too costly to implement or do not offer energy efficiency benefits, which motivate changes to the software implementation. Cycle accu-

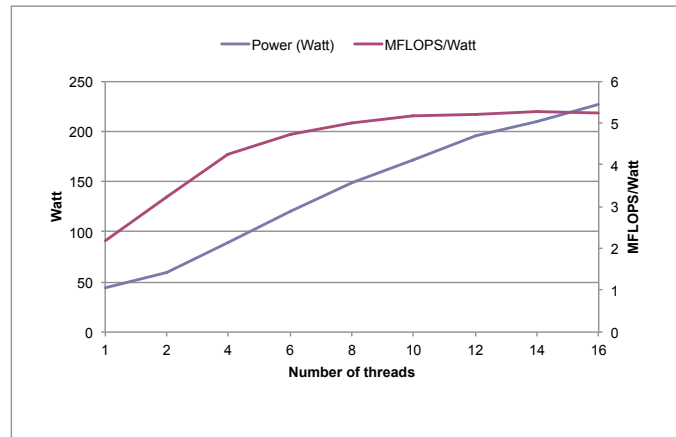


Fig. 5. OpenMP transformations using ROSE and their evaluation using a simulator on a numerical kernel (Jacobi Iteration) for both power and MFLOP/Watt.

rate simulation tools enable quantification of the performance impact on the applications when they are subjected to specific hardware constraints. Analysis of the hardware constraints, in terms of cost, area, and power consumption, is fed back to the application to motivate changes in the application design and algorithms to better fit within hardware constraints. This tightly integrated design cycle will enable orders of magnitude improvement in energy efficiency and usable performance.

IV. RESULTS

The co-design process that includes tight collaboration between an applications team, code analysis, and architectural simulation has been to work for applications ranging from Climate Modeling (Green Flash) [2], Seismic Imaging (Green Wave) [5], and an automated co-tuning process that includes both auto-tuning of hardware and software in the same process [6].

Our first co-design project, described in the November 2009 IEEE Computer article, [2] was a machine design co-designed for kilometer scale climate modeling, which is called Green Flash. The three central features of the Green Flash design process were fast FPGA-based hardware emulation, rapid design synthesis tools, and software auto-tuning technology. Green Flash has benefitted greatly from FPGA-based emulation to facilitate a tightly coupled co-design processes to develop an application-driven manycore chip design targeted at

Architecture	Intel Nehalem	NVIDIA Fermi	Green Wave
Total Nodes	127,740	66,823	75,968
MPoints/Watt	4.27	6.28	32.63
Comm overhead	9.5%	43%	16%
Total MWatt	38.2	26.1	5.0

TABLE II

Extrapolated node configuration, sustained energy efficiency and power consumption for 8th order forward and backward model of 30k x 20k x 10k survey with 12,000 timesteps within a week using 512³ subdomains arranged as subclusters of 256 nodes to achieve that throughput.

scientific applications. The project demonstrated how a multi-disciplinary hardware/software design process that facilitates close interactions between applications scientists, computer scientists, and hardware engineers can be used to develop a system tailored for the requirements of scientific computing. The Green Wave project, in collaboration with Fraunhofer ITWM, extended the co-design process study to Reverse Time Migration problems in Seismic Imaging, which are among the most demanding commercial applications for large-scale HPC clusters. Green Wave extended the modeling to the full node design including the memory subsystem and to the performance of the interconnect design. The project demonstrated how a hardware software co-design process can achieve an order of magnitude increase in energy efficiency over comparable GPU and CPU-based systems.

Both the Green Flash and Green Wave co-design projects relied on this rapid synthesis to rapidly prototype new interprocessor communication and synchronization services to support advanced execution models. For example, we synthesized direct word-granularity interprocessor communication services to support fine-grained synchronization primitives for non-cache-coherent global-address space languages. The CoDEX environment automatically exposes this new capability in the compiler to make it immediately accessible to our software stack developers for experimentation. These automated rapid prototyping tools will enable us to make substantial progress on developing novel hardware support for locality management, resilience, security, and interprocessor communication on a limited budget, which work together in a unified design flow as shown in Figure 4.

Advanced compiler environments also play a key role in our architectural analysis. For example, we use ROSE to analyze example applications and extract the MPI usage as an input code to support evaluation of message passing to support the SST/macro simulator. This work is part of the CoDEX project to automate how co-design can be applied to support the evaluation of DOE applications. The simulator represents a range of proposed future node architectures and ROSE is able to automate complex code transformations necessary to evaluate the potential of each design point. Figure 5 shows the evaluation of both power and MFLOP per Watt using a node simulator and for different numbers of threads. The automation of both the hardware and software design space exploration are both essential for accelerating the design-cycle for the iterative co-design process.

Overall, the hardware simulation and compiler-drive code

analysis for model development enabled consideration of complete application codes rather than proxy kernels thereof. Deep code analysis using compiler-assisted model development, interconnect simulation, and FPGA-based hardware emulation systems with rapid design synthesis tools played a central role in each of these projects. The co-tuning environment enabled rapid search of an enormous design space to find a more optimal hardware/software solution that could not be determined using the typical decoupled design processes. As such, this presents a vision of a new design process that could be used to develop effective exascale-class HPC systems.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy’s Office of Advanced Scientific Computing Research. Lawrence Berkeley National Laboratory is supported by the DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

V. SIDEBARS

begin sidebar: Co-design in Embedded Systems

Hardware/software co-design has long been a feature of power-sensitive embedded system designs. In this context, co-design refers to a design methodology to create integrated processor + software solutions that are optimized to deliver more useful work per watt than an unmodified off-the-shelf design. The embedded processor market has refined co-design processes over the past 20 years to meet the demanding cost and power efficiency requirements of battery-powered consumer electronics applications (smart phones, MP3 players, etc.), and power-sensitive high-performance embedded applications like avionics systems in aircraft. What has made it so successful is a continuing focus on developing tools that make hardware-software co-design productive, cost-effective, and beneficial, such as automated processor synthesis tools, cycle accurate simulators, and automated generation of software tools (compilers and debuggers) from hardware specifications. These tools have broken through the slow pace of system design that held back the embedded industry for many years, and continues to hold back necessary advances in the HPC space.

end sidebar: Co-design in Embedded Systems

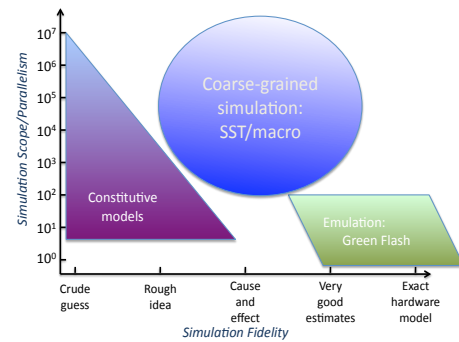


Fig. 6. **Multiple modeling approaches are required to cover both the scale and accuracy required to understand system design trade-offs. The two key axes for simulation and modeling techniques are fidelity of the model (horizontal axis) and the scale of system you can simulate.**

begin sidebar: Multiresolution Modeling

Modeling, simulation, and compiler analysis all play synergistic roles in the co-design process to cover a broad space of design parameters. Figure 6 shows that a multi-resolution approach using multiple modeling methodologies must be employed to cover both the scale of exascale systems and the fidelity required to have confidence in our design choices. Tools such as cycle accurate hardware simulation (in green) offer extreme detail in their modeling capability, but limit the scale of system that can modeled to node size or small clusters. Software simulators, such as SST macro, can expand the scale of system that can be modeled, but must neglect some detail to achieve that scalability. Lastly, the constitutive models and other empirical modeling methods can cover much larger systems, but by definition only model effects included as parameters in the model. The majority of modeling is done by empirical models because they are faster to construct and evaluate, but the software-based, and cycle-accurate models are used to verify that the simpler model has included all important effects, and has not neglected anything any essential (but unanticipated) effects.

end sidebar: Multiresolution Modeling

REFERENCES

- [1] Vikram S. Adve, Rajive Bagrodia, Ewa Deelman, and Rizos Sakellariou. Compiler-optimized simulation of large-scale applications on high performance architectures. *Journal of Parallel and Distributed Computing*, 62(3):393 – 426, 2002.
- [2] David Donofrio, Leonid Oliker, John Shalf, Michael F. Wehner, Chris Rowen, Jens Krueger, Shoab Kamil, and Marghoob Mohiyuddin. Energy-efficient computing for extreme-scale science. *Computer*, 42:62–71, 2009.
- [3] S. D. Hammond, G. R. Mudalige, J. A. Smith, S. A. Jarvis, J. A. Herdman, and A. Vadgama. WARPP: a toolkit for simulating high-performance parallel scientific codes. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools '09*, pages 19:1–19:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [4] Curtis L. Janssen, Helgi Adalsteinsson, Scott Cranford, Joseph P. Kenny, Ali Pinar, David A. Evensky, and Jackson Mayo. A simulator for large-scale parallel architectures. *International Journal of Parallel and Distributed Systems*, 1(2):57–73, 2010.
- [5] Jens Krueger, David Donofrio, John Shalf, Marghoob Mohiyuddin, Samuel Williams, Leonid Oliker, and Franz-Josef Pfreundt. Hardware/software codesign for energy-efficient seismic modeling. In *Proceedings of SC2011*, 2011.
- [6] Marghoob Mohiyuddin, Mark Murphy, Leonid Oliker, John Shalf, John Wawrzyniek, and Samuel Williams. A design methodology for domain-optimized power-efficient supercomputing. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, New York, NY, USA, 2009. ACM.
- [7] Rose compiler website. <http://www.rosecompiler.org>.
- [8] Structural Simulation Toolkit macroscale (SST/macro), website. <http://bitbucket.org/cljanss/sstmacro>.
- [9] R. Susukita, H. Ando, M. Aoyagi, H. Honda, Y. Inadomi, K. Inoue, S. Ishizuki, Y. Kimura, H. Komatsu, M. Kurokawa, K.J. Murakami, H. Shibamura, S. Yamamura, and Yunqing Yu. Performance prediction of large-scale parallel system and application using macro-level simulation. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1 –9, 2008.

John Shalf is a staff computer scientist at Lawrence Berkeley National Laboratory and leads the Advanced Technology Group at the National Energy Research Supercomputing Center. His research interests include computer architecture, programming models, and frameworks for large-scale scientific application development. Shalf received his degrees in Electrical and Computer Engineering from Virginia Tech. He is a member of the IEEE Computer Society. Contact him at jshalf@lbl.gov.

Dan Quinlan Dan Quinlan is the leader of the ROSE project in the Center for Advanced Scientific Computing. His research is in numerous areas that intersect computer science and numerical analysis. Research interests include object-oriented numerical frameworks, parallel adaptive mesh refinement, parallel multigrid algorithms, semantics-based source code transformations, C++ compiler tools/infrastructure/design, cache-based optimizations, parallel array classes, parallel data distribution mechanisms, and parallel load balancing algorithms. Dr. Quinlan earned his Ph.D. in Computational Mathematics from the University of Colorado.

Curtis Janssen is a Distinguished Member of Technical Staff at Sandia National Laboratories in Livermore, CA. He received a PhD in Theoretical Chemistry from the University of California at Berkeley. Dr. Janssen's interests span the various technologies required for high performance scientific computation, including areas such as programming models, performance analysis, and machine architecture.