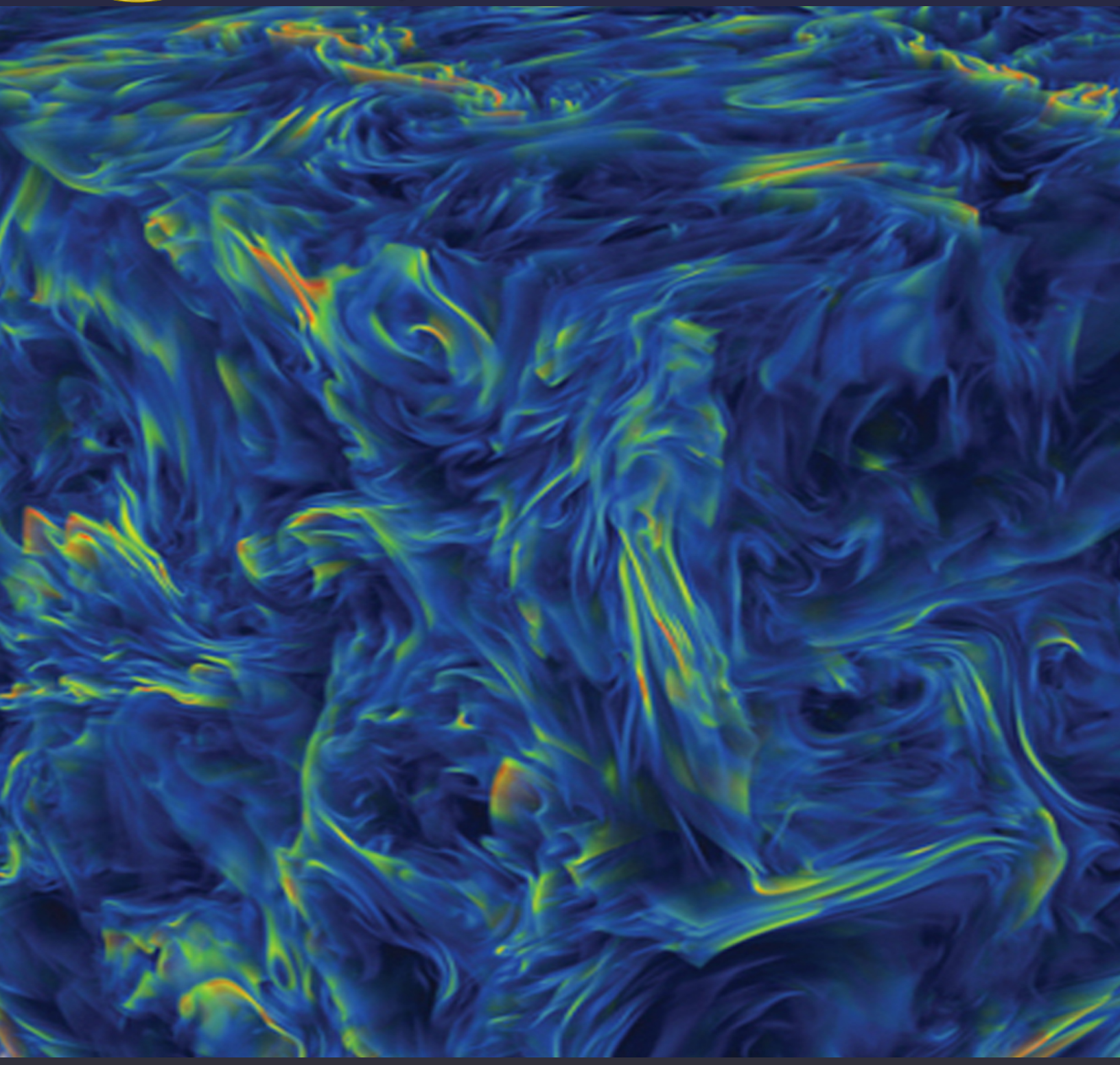
Top Ten Exascale Research Challenges

DOE ASCAC Subcommittee Report
February 10, 2014



U.S. DEPARTMENT OF ENERGY
Office of Science

Sponsored by the U.S. Department of Energy, Office of Science,
Office of Advanced Scientific Computing Research

ASCAC Subcommittee for the Top Ten Exascale Research Challenges

Subcommittee Chair
Robert Lucas (University of Southern California, Information Sciences Institute)

Subcommittee Members
James Ang (Sandia National Laboratories)
Keren Bergman (Columbia University)
Shekhar Borkar (Intel)
William Carlson (Institute for Defense Analyses)
Laura Carrington (UC, San Diego)
George Chiu (IBM)
Robert Colwell (DARPA)
William Dally (NVIDIA)
Jack Dongarra (U. Tennessee)
Al Geist (ORNL)
Gary Grider (LANL)
Rud Haring (IBM)
Jeffrey Hittinger (LLNL)
Adolfy Hoisie (PNLL)
Dean Klein (Micron)
Peter Kogge (U. Notre Dame)
Richard Lethin (Reservoir Labs)
Vivek Sarkar (Rice U.)
Robert Schreiber (Hewlett Packard)
John Shalf (LBNL)
Thomas Sterling (Indiana U.)
Rick Stevens (ANL)

Acknowledgements
The ASCAC Subcommittee for the Top Ten Exascale Research Challenges would like to thank the following for their contributions to this report.

Jon Bashor (LBNL)
Ron Brightwell (Sandia National Laboratories)
Paul Coteus (IBM)
Erik Debenedictus (Sandia National Laboratories)
Paul Franzon (North Carolina State University)
Scott Hemmert (Sandia National Laboratories)
Jon Hiller (Science and Technology Associates)
K.H. Kim (IBM)
Harper Langston (Reservior Labs)
James Laros III (Sandia National Laboratories)
Sven Leyffer (ANL)
Richard Murphy (Micron)
Rob Ross (ANL)
Clayton Webster (ORNL)
Stefan Wild (ANL)

Cover Image
Cover image was produced by a multidisciplinary team that includes LANL researcher Hussein Aluie of the Computational Physics and Methods group and the Center for Nonlinear Studies reports an alternative explanation, involving turbulence, for the fast magnetic reconnection.

Contents

List of Figures

The Top Ten Exascale Research Challenges

1 Executive Summary

Exascale computing systems are essential for the scientific fields that will transform the 21st-century global economy, including energy, biotechnology, nanotechnology, and materials science. Progress in these fields is predicated on the ability to perform advanced scientific and engineering simulations, and analyze the deluge of data. The Department of Energy's 2011 strategic plan and 2012 addendum calls out a Priority Goal: *Lead Computational Sciences and High-Performance Computing* and a Targeted Outcome: *Continue to develop and deploy high-performance computing hardware and software systems through exascale platforms.*

On July 29, 2013, ASCAC was charged by Patricia Dehmer, the Acting Director of the Office of Science, to assemble a subcommittee. This subcommittee was directed to return "a list of no more than ten technical approaches (hardware and software) that will enable the development of a system that achieves the Department's exascale goals, particularly the usability goals for the Department's mission-critical applications, as articulated in the attached presentation. That is, given the known technical barriers that could prevent the development of a computer that achieves the Department's exascale goals, what are credible technical approaches for overcoming these barriers? The subcommittee's report should provide compelling justifications for including each item in the list and describe the expected impact on overall system performance." This report, in response to DOE's charge, details key technological barriers to achieving productive exascale platforms, along with recommended approaches to overcome these barriers.

Numerous reports over the past five years have documented the technical challenges and the non-viability of simply scaling existing computer designs to reach exascale. The technical challenges revolve around energy consumption, memory performance, resilience, extreme concurrency, and big data. Drawing from these reports and more recent experience, this ASCAC subcommittee has identified the top ten computing technology advancements that are critical to making a productive, economically viable, exascale system. Note, the challenges are not rank-ordered by importance, but rather grouped by the subcommittee into related categories.

1.1 The Top Ten Exascale System Research Challenges

1. Energy efficiency: Creating more energy-efficient circuit, power, and cooling technologies.
2. Interconnect technology: Increasing the performance and energy efficiency of data movement.
3. Memory Technology: Integrating advanced memory technologies to improve both capacity and bandwidth.
4. Scalable System Software: Developing scalable system software that is power- and resilience-aware.
5. Programming systems: Inventing new programming environments that express massive parallelism, data locality, and resilience
6. Data management: Creating data management software that can handle the volume, velocity and diversity of data that is anticipated.
7. Exascale Algorithms: Reformulating science problems and redesigning, or reinventing, their solution algorithms for exascale systems.

8. Algorithms for discovery, design, and decision: Facilitating mathematical optimization and uncertainty quantification for exascale discovery, design, and decision making.

9. Resilience and correctness: Ensuring correct scientific computation in face of faults, reproducibility, and algorithm verification challenges.

10. Scientific productivity: Increasing the productivity of computational scientists with new software engineering tools and environments.

Innovative solutions to the challenges identified in these ten areas are on the critical path to delivering effective HPC systems for critical science and technology areas that underpin U.S. economic competitiveness for the 21st century. For example, advances in near-threshold-voltage circuit technology hold the promise of improving energy efficiency of systems by nearly an order of magnitude, but it will require a revolutionary change in programming environments and application design to tap into this opportunity. Without dramatic reformulation of hardware together with software, exascale systems are projected to consume 100s of megawatts, and few existing HPC applications will be able to scale beyond a tenth of an exaflop. Without the advances in programming models and productivity described in this report, it will not be feasible to redesign algorithms and application codes to provide efficient and correct answers. And without the tools to analyze the data from experiment and simulation, scientists will be overwhelmed by the data volume generated by next generation scientific instruments and advanced computing facilities. Each of the ten areas plays an essential role in enabling a practical and effective HPC technology for the next decade, which in turn impacts the DOE mission, and more broadly ensures U.S. leadership in these key science and technology fields. Unlike past systems, the next generation of supercomputers needs to be developed using co-design approaches where scientific problem requirements guide computer architecture and system software design. Doing so will require long-term partnerships among vendors, government agencies, and academia.

In the course of conducting this study, the subcommittee considered a number of related issues that add context to the top ten research challenges and the means to overcome them. To further assist the DOE in planning its future research directions, these issues are addressed in a series of findings and recommendations.

1.2 Summary of Findings

1.2.1 Exascale computing is critical for executing the DOE mission.

ASCAC reaffirms its findings from previous reports that leadership in high performance computing (HPC) is critical to achieving the DOE mission of ensuring U.S. leadership in science, engineering, and national security. In the last six years, this has been documented in many exascale reports from Office of Science programs, the National Nuclear Security Administration, and other U.S. government agencies.

1.2.2 U.S. national leadership is at risk.

Without aggressive investment and technical innovation in HPC, the U.S. risks falling behind rapidly emerging international competitors, not all of whom are friendly to U.S. interests. This in turn threatens to undermine the nation's intellectual leadership in a broad range of science, its economic position, and its security.

1.2.3 The U.S. has the technical foundation to create exascale systems.

The U.S. semiconductor and HPC systems industries are capable of developing the necessary technologies for an exascale computing capability by the early part of the next decade, based largely on evolving commercially driven component fabrication, systems integration, and software engineering capabilities. However, for a truly effective and productive exascale computing capability, the U.S. government will need to focus investments on the research, development, and integration of HPC technologies that will otherwise not be created solely by commercial drivers.

1.2.4 An evolutionary approach to achieving exascale will not be adequate.

The dramatic improvements essential to achieving effective exascale computing will not be satisfied by incremental extensions to today's conventional practices. Commercial market drivers do not provide a viable general path to delivering necessary scalability, time and energy efficiency, and user productivity including performance portability to future generation exascale class computers.

1.2.5 The U.S. government's continuous leadership and investment are required to create exascale systems.

The U.S. computing industry is unlikely to develop effective exascale computer systems without U.S. government investment and focused mission goals. Innovation, sometimes of an incremental nature, and in other areas revolutionary, will be required under DOE direction to enable U.S. leadership in advanced HPC.

1.3 Summary of Recommendations

1.3.1 DOE should invest in a program of continuous advancement in HPC.

Exascale is only the next milepost in a half-century of continuous progress towards increasing capability in computational science. The U.S. government requires a stable, long-term investment strategy to ensure continuous U.S. leadership in HPC beyond today's petascale performance regime, extending to exascale and beyond. In the immediate future, much of that research investment should be focused on the top ten challenges identified within this report.

1.3.2 DOE should invest in the U.S. industrial base to catalyze the foundation for exascale systems.

DOE should invest in extending commercial semiconductor, communications, systems integration, and software technologies to prepare the U.S. industrial base for its role and contributions in future HPC scientific, engineering and national security missions. All of these exascale components must be developed by and be available from U.S. sources, otherwise the supply chain is vulnerable to interdiction by foreign powers, which in turn could threaten the nation's security.

1.3.3 DOE should invest in exascale mathematics and system software responsive to DOE missions and other U.S. government requirements.

The mathematical algorithms needed for many DOE missions are unique, and must be reinvented to function at exascale. As with today's Leadership-class systems, much of the software infrastructure of an exascale system will be unique to its scale and the missions for which DOE will deploy it. Therefore, DOE must invest in robust and scalable mathematical algorithms, operating systems, runtime systems, and tools for the management of the data that will be generated and/or processed.

1.3.4 DOE should create an Open Exascale System Design Framework to enable cooperative hardware and software advancement.

DOE's understanding of government mission drivers provides a critical foundation to coordinate requirements for and development of interoperable system components. Thus DOE should establish a co-design framework for collaboration and system integration of crosscutting component layers. Such a framework will serve as the conceptual scaffolding for the development of new programming language and hardware architecture, runtime software and operating system, application algorithms along with their data and workflows, and management policies for future exascale computing.

2 Introduction

Scientific progress has been, and will continue to be, a notable source of American prosperity and leadership in world affairs. High performance computing (HPC), has a fundamental role to play in enabling scientific progress. The DOE mission elevates the importance of leadership in this space for national security and economic reasons. This spans a broad range of DOE missions, including new energy solutions, ecological sustainability, global climate change, next generation nuclear power plants, scientific discovery, and global security. This is reflected in the Department of Energy's 2011 strategic plan. The 2012 addendum calls out both a Priority Goal: *Lead Computational Sciences and High-Performance Computing,* and a Targeted Outcome: *Continue to develop and deploy high-performance computing hardware and software systems through exascale platforms.* Based on this fact alone, one would expect continued and substantial investment in HPC.

It is remarkable to observe that computing is becoming central to progress even for the most fundamental research in the sciences. Maintaining U.S. leadership in computational science requires the best tools. This includes a succession of computer resources with world-class capability as well as the data storage and analysis tools needed to effectively use the results. As is stated in the DOE Office of Science (DOE SC) strategic plan, "Each of the [scientific] goals, and progress in many other areas of science, depends critically on advances in computational modeling and simulation. Crucial problems that we can only hope to address computationally require us to deliver effective computing power orders-of-magnitude greater than we can deploy today." That implies exascale, 10^{18} operations per second, the next major milestone in a process of exponential improvement that has continued for over half a century. For the purposes of this report, the definition of exascale is more than just a peak rate of sustained arithmetic operations, bytes of storage, or the familiar High Performance Linpack (HPL) [117] Rmax floating-point operations per second (Flop/s). Exascale is a relative term suggesting 1000-fold better capability than representative from the petascale, which started circa 2009. This definition encompasses 1000-fold improvements in the usable performance of systems at all scales relative to the 2009 baseline, not just the largest. As such, exascale suggests a practical delivered capability within the scope of achievable power, reliability, size, cost, programmability, and other constraints. Such a system cannot be produced today, and identifying research required to achieve this capability is the purpose of this report.

While DOE is able to maintain its position of leadership today, that position is threatened in the near future [93]. Underlying technology is changing dramatically now. The models of scaling that formed the basis of ever increasing computer power are ending [73, 113]. Dennard scaling [32] is over, and while Moore's Law [84] continues, its demise is in sight too. Therefore, the future of high performance computing (HPC) is unknown. This in turn threatens to compromise DOE's ability to perform the leadership aspect of its mission. This is not the first time this has happened. Circa 1990, CMOS device technology overtook TTL (Transistor Transistor Logic) and ECL (Emitter Coupled Logic). DOE led the way to a new stable point in HPC, communicating sequential processes (CSP) [60] integrated from commercial, off-the-shelf technology (COTS). DOE needs to do it again, as nobody else in U.S. is stepping up. The alternative is for the United States to cede leadership in HPC to Europe, Japan, and/or China

2.1 Challenges

Given the importance of exascale computing to DOE, on July 29, 2013, ASCAC was charged by Patricia Dehmer, the Acting Director of the Office of Science, to assemble a subcommittee. This subcommittee was directed to return "a list of no more than ten technical approaches (hardware and software) that will enable the development of a system that achieves the Department's ex-

ascale goals, particularly the usability goals for the Department's mission-critical applications, as articulated in the attached letter. That is, given the known technical barriers that could prevent the development of a computer that achieves the Department's exascale goals, what are credible technical approaches for overcoming these barriers? The subcommittee's report should provide compelling justifications for including each item in the list and describe the expected impact on overall system performance."

This report, written in response to the charge, describes ten key technical areas that need substantial research and development in order create a productive, economically viable, broadly useful, exascale system, and enable DOE to perform its mission. The report also discusses the impact that advances in each of the ten areas will have on the exascale system's performance and its ability to solve the science problems critical to the nation. Numerous reports over the past five years have documented the technical challenges and the non-viability of existing computer designs to simply scale to exascale. The technical challenges revolve around energy consumption, memory performance, resilience, and extreme concurrency. These challenges are the drivers for the research that must be done.

To determine the top ten research and development areas, a group of 23 experts in high performance computing hardware and software was assembled. These experts were from industry, academia, and government-funded research laboratories. They met dozens of times, both in person and by telephone, analyzed the state of the technology upon which an exascale system will be developed, and came up with the following top ten challenges. The challenges are not rank-ordered by importance, but rather grouped by the subcommittee into related categories.

1. *Energy efficiency:* The goal is to achieve exascale using 20 MW of power, yet existing circuits consume an order of magnitude too much power to meet this goal. Without much more energy-efficient circuits, architecture, power conversion, power delivery and cooling technologies, the total cost of ownership for exascale systems could be 10 times higher than today.

2. *Interconnect technology:* The performance of the interconnect is key to extracting the full computational capability of a computing system. Without a high performance, energy-efficient interconnect, an exascale system would be more like the millions of individual computers in a data center, rather than a supercomputer.

3. *Memory technology:* Many new memory technologies are emerging, including stacked memory, non-volatile memory, and processor-in-memory. All of these need to be evaluated for use in an exascale system. Minimizing data movement to this memory and making it more energy efficient are critical to developing a viable exascale system. Science requirements for the amount of memory will be a significant driver of overall system cost.

4. *Scalable system software:* Present system software was not designed to handle the exponentially growing scale of leadership-class systems. Overall management of the power and resilience of the millions of nodes in an exascale system will be the responsibility of the system software.

5. *Programming systems:* The present CSP model doesn't include resilience and puts all the burden of locality and parallelization on the application developers. Exascale systems will have billion-way parallelism, and frequent faults. More expressive programming models are needed that can deal with this behavior and simplify the developer's efforts.

6. *Data management:* There is an explosion in both the amount and complexity of the data being generated by experiments and simulations. Without significant improvements in data management, the answers in the data will never be found.

7. *Exascale algorithms:* There are many thousands of man-years invested in the scientific and engineering codes now in use. Changing them to run with billion-way parallelism will require redesigning, or even reinventing, the algorithms used in them, and potentially reformulating the science problems. Understanding how to do these things efficiently and effectively will be key to solving mission-critical science problems at exascale.

8. *Algorithms for discovery, design, and decision:* It is anticipated that the need for methods and software to efficiently carry out uncertainty quantification and optimization on complex multi-physics problems will be a key need at exascale.

9. *Resilience and correctness:* Getting the correct answer on exascale systems with frequent faults, lack of reproducibility in collective communication, and new mathematical algorithms with limited verification will be a critical area of investment. Getting the wrong answer really fast is of little value to the scientist.

10. *Scientific productivity:* Programming tools, compilers, debuggers, and performance enhancement tools will all play a big part in how productive a scientist is when working with an exascale system. Without increasing programming productivity, an application may run in a few hours or days at exascale, but it may take months for the scientist to get it ready to run.

2.2 Integrating the Results

For the last two decades, the HPC community has successfully used a strategy based on the large-scale integration of commercial-off-the-shelf (COTS) microprocessors into massively parallel supercomputing systems. In recent years, issues with the level of performance realized at scale have led to questions about the continued viability of this strategy, and the need for research to address this problem. Addressing the top ten exscale research challenges should alleviate many of these concerns, but it is not enough.

All exascale research activity will need to be organized, coordinated, and funded in the context of, and in support of, an integrated system design that is focused on meeting the performance, productivity, and energy requirements associated with the anticipated mission-driven exascale workflows of the institutions that will ultimately acquire these systems. The general principle must be that one can't design in isolation. A collection of separate, disjointed research activities, each aimed at optimizing a single design aspect or technology, will likely result in a suboptimal overall solution. The realization of an exascale system will involve a complex, multidimensional tradeoff between hardware (processors, memory, energy efficiency, reliability, interconnectivity), software (programming models, scalability, data management, productivity) and algorithms. Thus, a total systems approach is necessary, implying co-design of hardware and software, and such a total systems approach must inform the research programs for exascale [33, 34, 114].

Advancing to exascale will require innovations in architecture to integrate the many research results from diverse fields including silicon, memory, packaging, system software, interconnect, signaling and optical technologies, mathematics, and programming models. Exascale architecture requires finding the right balance of these technologies to achieve density, cost, performance and power efficiency at the appropriate levels. Research in exascale architecture will leverage the learning and experience gained in developing and using intermediate systems, and will result in improvements to some of the key elements of architecture, programming models, and system software. Exascale system design will be an optimization problem involving a set of constraints, a set of metrics for evaluating those constraints, and a methodology for evaluating them.

2.3 Overview

This report is focused on identifying and explaining the top ten research challenges that must be overcome to create an effective exascale computing system. The next ten sections of the report each address one of the challenges. They include why the challenge is thought to be in the top ten, credible technical approaches for overcoming them, and the impact these results will have on the overall exascale system. The structure of the sections varies a bit, reflecting the personal styles of the experts who contributed them. While progress on the top ten research challenges may initially proceed independently, an effective exascale system will require balance and integration of the results. Issues related to a execution models, architecture, co-design methodology, modeling, and system integration are discussed in section 13.

Finally, in the course of conducting this study, the subcommittee considered a number of related issues that add context to the top ten research challenges and the means to overcome them. To further assist the DOE in planning its future research directions, this report elaborates on these additional findings and makes recommendations that, in the aggregate, contribute to advancing U.S. capacity to develop and apply exascale computing.

3 Energy Efficiency

The Tianhe-2 system in China, which is number one in the current TOP500 list, consumes 17.8 megawatts (MW) to deliver 33.86 PFlops (10^{15} floating point operations per second) on the HPL benchmark [117]. The current top supercomputers typically have an energy efficiency of 2–4 GFlops/W [51]. Thus, an exascale computing system ($\sim 10^{18}$ Flops) with a power budget of 20 MW will need at least an order-of-magnitude improvement in energy efficiency as compared to today's systems. To address this challenge, the energy efficiency of basic compute blocks, interconnections between the blocks, and the memory subsystem all need to be considered. Successful energy management at exascale will require research that cuts across all aspects of the design and must be integral to all design elements. Energy management will need to be practiced from the very concept of the system, and driven into base technology elements. It impacts silicon technology, circuit and chip design as well as design tools, power delivery subsystems throughout the design, system interconnect, memory and storage design, system packaging and system control software. As discussed in this section, research is needed to guide the evolution of power management from the largely static and hardware-focused environments of today to a much more dynamic process, requiring significant software support and interoperation with the underlying hardware.

3.1 Aggressive versus Revolutionary Technologies

An exascale program will accelerate the introduction of advanced technology for research and for commercial use. Given the challenges in energy efficiency, aggressive use of existing (or evolutionary) low power technologies and design techniques will be required. However, some revolutionary approaches are likely to be necessary as well.

A balanced approach is required between such "aggressive evolutionary" and any "revolutionary" technologies. Exotic technologies typically have the following properties:

- They almost always cost more and take longer to go from laboratory demonstrations to system readiness than initially predicted;
- They require significant application and system software restructuring efforts and costs;
- They require significant efforts in new design tools, qualification processes, test methodology and equipment, etc.;
- They take much longer to be commercially viable (vs building a single system) and hence can be economically unviable.

A balanced approach will apply radically new technologies only where necessary, and with strict management of the above risks. The long term exploration of such technologies should be encouraged as part of advanced research programs, independent of important product or mission critical systems development. When, at a future decision checkpoint, some revolutionary technologies look viable for exascale application, further investment would be required in hardware and software tools, qualification, bring-up and test issues in preparation for the introduction of these technologies. When organized this way, an exascale program should be flexible enough to introduce radically new technologies, should they mature early.

As evolutionary device/chip technologies we can count, for example, the aggressive use of clock gating, power gating, and dynamically adaptive voltage/frequency domains. Three-dimensional (3D) stacking is promising for integrated power regulation and distribution, and is promising for

memory density, energy efficiency, and processing-in-memory. The high device density may lead to power delivery and cooling challenges. For applications that do not critically require the integration advantages, 3D integration will initially be more costly than the same functionality on separate components, and large-scale commercial viability is a concern.

CMOS and related field effect technologies (FETs) are fundamentally limited by the "Boltzmann tail" of electrons tunneling through sub-threshold voltage barriers, which gives rise to sub-threshold leakage power. The room temperature sub-threshold slope of 60 mV/decade of current FETs implies that reliable CMOS operation, with high performance (i.e. high on-to-off current ratio), requires operation at voltages above 0.55-0.6V. Low-temperature operation increases the sub-threshold slope and enables lower voltage operation, but is costly to achieve, may not lead to overall energy savings at the systems level, and does not have a commercially viable outlook. Very low (near threshold) voltage operation is another such aggressive/evolutionary technology, that can be applied to current CMOS technologies, and is the subject of the next subsection.

In contrast, revolutionary technologies are centered on exploratory devices. Some (e.g., carbon nanotubes) are still subject to Boltzmann tail behavior and will help perhaps a generation of device scaling, and may help with further voltage scaling to perhaps 0.3V. Other emerging "steep slope" (non-Boltzmann) device technologies hold promise for true low voltage operation, (e.g., tunnel FETs, piezo FETs, and spin based devices). Different device characteristics may lead to a hybrid approach, e.g. fast C-tube/tunnel/piezo FETs for logic and slow spin-based devices for arrays. These new device technologies will face issues such as materials, device scalability, performance, and reliability. Research on these post-CMOS devices is organized under the auspices of the Nanoelectronics Research Initiative of the Semiconductor Research Corporation (NRI/SRC).

Moderate funding would be required to accelerate research and reach a decision checkpoint for a few of the most promising exploratory device candidates. This would fund device research, circuit/chip design tools to create test chips of appreciable complexity, and test chips. Then the market (i.e., mobile, internet-of-things, power harvesting applications) will take over commercial development. An exascale program should be ready to leverage such a development and provide funding to accelerate design tools and system design.

3.2 Near Threshold Voltage (NTV)

The concept of operating near the threshold voltage (NTV) of a transistor is well established, and can provide up to a 10-fold increase in energy efficiency especially for the basic compute blocks. However, it poses numerous implementation challenges needing maturity, rethinking of system architecture, and design technologies to successfully adopt it into future systems. Figure 1 shows experimental results of NTV operation and energy efficiency in 65 nm technology, which were subsequently confirmed on 45, 32, and 22 nm technologies [2, 63, 67, 68]. As the supply voltage is reduced, energy efficiency increases by an order of magnitude. However, it also poses the following system challenges:

1. Variability in the speed of operation of individual circuit blocks increases dramatically to almost 50%,
2. As the frequency of operation is lowered, logic throughput is reduced, which increases the demand for concurrency
3. Stability of small signal circuits such as SRAM reduces significantly, and
4. Sub-threshold leakage power becomes a substantial portion of the total power.

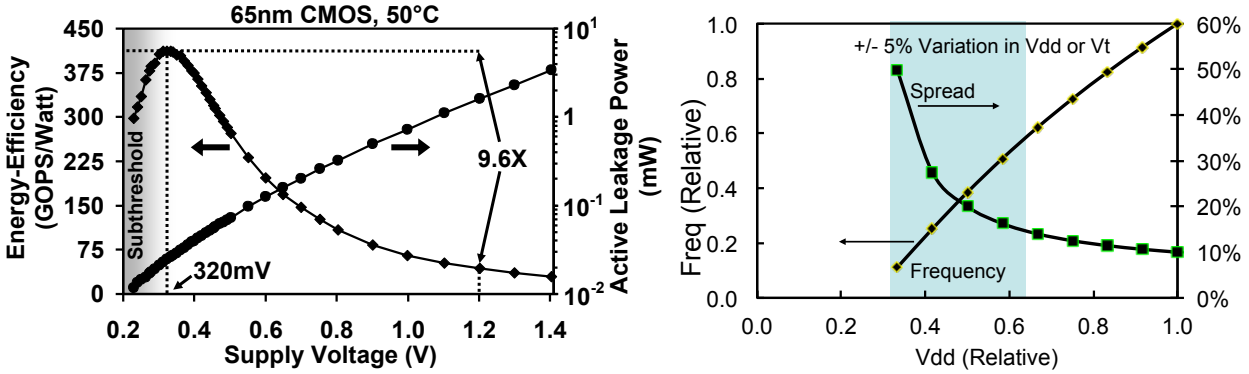


Figure 1: Energy efficiency and variability in NTV operation.

The increased variability in circuit speed poses both a circuit design and a system challenge. The gate-to-gate variability calls for the development of appropriate circuit design techniques and corresponding design tools. From a system perspective, variability in core-to-core or thread-to-thread speed will impact synchronization (barriers) in multi-threaded/multi-core processors, with further possible impact on system software.

Reduction in the logic throughput is addressed by increased parallelism, replicating logic blocks (cores) to maintain the throughput. However, this will compound an issue already encountered in the current petascale era, with about a million processor cores per system. At current core performance, the number of processor cores for exascale machines will grow to the order of a billion. However, if the number of cores will have to be increased additionally also to restore the logic throughput for a given workload running at very low frequency, then the software may need to deal with another order-of-magnitude growth in the number of cores. This approach carries the implied assumption that that any given exascale workload can be distributed over more cores without adverse effects. This assumption is valid for only a few HPC workloads today (and even in those cases, with limits to the scaling that can be achieved), and will severely limit applicability of this idea for general HPC workloads.

Stability of SRAM arrays can be addressed by: (1) replacing traditional 6T designs by 8T or 10T SRAM designs, or (2) not employing NTV to the SRAM arrays. The disproportionate sub-threshold leakage power issue is addressed by employing fine grain power and energy management using power gating, and integrated power delivery subsystems, providing fine grain spatial (functional block level) and temporal (10s of nanosecond) control. All of these techniques will need careful optimization at the system level, especially with hardware/software co-design principles, to ensure that the system software can indeed exercise control, and dynamically establish the optimum operating point.

3.3 Energy Efficient Architecture

Much of the "energy efficiency gap" that needs to be bridged to reach exascale goals can be achieved via more efficient architecture. Performing a double-precision floating point operation on a 28nm CMOS chip requires about 12pJ (25pJ for a double-precision fused multiply accumulate operation). A 32-bit integer operation requires only 3pJ. Yet executing one floating point instruction requires around 200pJ on a very simple, single-issue, in-order core, and upwards of 1nJ on a high-

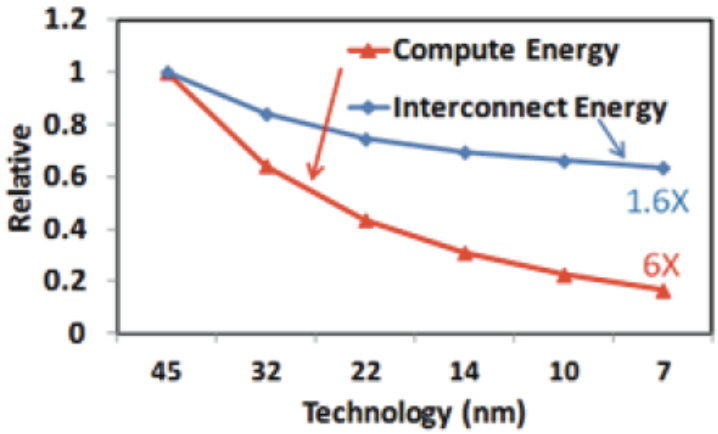


Figure 2: Compute and interconnect energy.

performance, multiple-issue, out-of-order core. Efficiency can be gained first by simplifying the core, eliminating complex logic that performs run-time, hardware scheduling, branch prediction, and other such functions that improve single-thread performance but do not affect throughput. Such simplification results in a simple, in-order core with no speculation.

Further gains are possible by reducing the overhead of instruction supply and data supply. Simply reading a 32-bit instruction from a typical 32KB instruction cache requires 20pJ - roughly the same energy as that needed to perform a 64-bit floating-point operation. Similar overheads are required by conventional data-supply mechanisms. New, streamlined instruction- and data-supply mechanisms hold the potential to dramatically improve the efficiency of even simple, in-order cores. Another promising approach is the use of compound operations, having a single instruction direct the execution of several arithmetic operations with intermediate results passed over dedicated wiring amortizes the instruction-supply overhead over many operations and eliminates the operand-supply overhead for the intermediate results.

3.4 Energy Efficient Interconnects

As semiconductor process technology scales from 22 nm today towards 7 nm and beyond, one can expect the compute (and local interconnect) energy to scale down, but as shown in Figure 2, the energy associated with longer "on-die" interconnects across-chip will not scale as much. This is because the size of the chip will remain constant across the generations, and the energy to move data across the chip will not scale much. An NTV approach will reduce the energy of computation and across-chip interconnects equally, but the energy disparity will continue to increase. We will need to adopt advanced interconnect techniques for on-die interconnects, such as low-swing differential signaling, twisted for noise immunity, and mature these technologies.

NTV, as described above, reduces compute energy, but requires more parallelism for a given throughput, demanding even more energy for data movement across the logic and the cores. Therefore, a critical balance of data movement and the extent of NTV will need to be established.

3.5 Memory and Register Files

Memory and register file arrays will likely use NTV modestly, and therefore their performance will not drop as much as that of logic. As a result, to balance the performance of the compute and

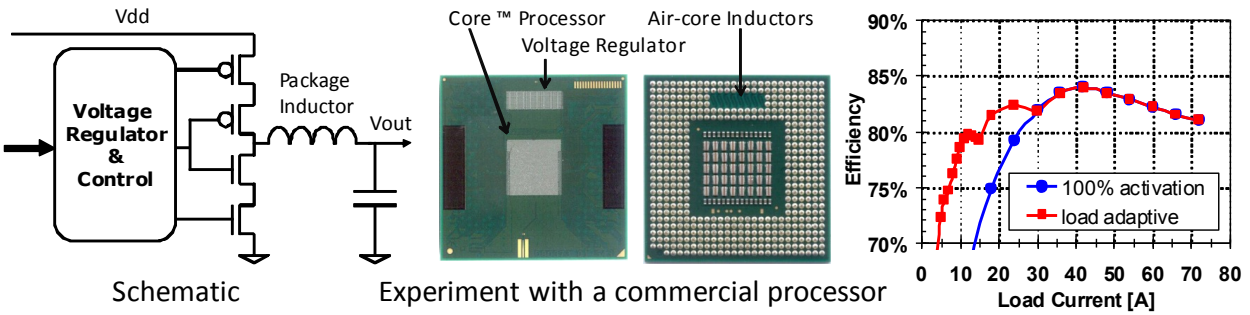


Figure 3: Integrated voltage regulator.

the memory, arrays will become disproportionately larger. On the one hand this is good for data locality, reducing the need for data movement and saving energy, but on the other hand, larger arrays will consume disproportionately larger energy. Almost half of the power consumed in the arrays is due to sub-threshold leakage and therefore application of leakage management techniques will be important. This paradigm shift needs to be considered in devising the architecture of the extreme scale system to exploit larger arrays, and also employ fine grain leakage power management techniques.

3.6 Efficient On-Chip Power Delivery and Management

Electrically distant power delivery solutions, such as voltage regulators on boards, are not effective in fine grain power management because (1) response time is large, due to inductive effects, (2) regulators are limited in number, and (3) efficiency is not constant across a broad range of loads. Figure 3 shows an experimental integrated voltage regulator which can be implemented on the processor die itself to create multiple supply voltage rails needed for NTV operation [111]. Several regulators can be ganged together, and even distributed across the die, to improve overall efficiency across broad range of loads to allow fine grain power and energy management by bringing power delivery closer to the load.

Local fine grain power and energy management can be accomplished by the well-known clock-gating techniques for active power reduction, and power gating techniques for leakage power management, as shown in Figure 4. Although these techniques are well known, the challenge lies in implementation at a fine grain level, as well as state preservation when power gating is used.

An effective exascale power distribution scheme will likely include integrated voltage regulators distributed across the die with a small number of power rails. The power gates of individual cores and logic blocks can be powered up using any of these rails to dynamically select the optimal voltage and frequency controlled by the system software. Of course, at a system level, a balance will need to be struck between the increased overhead for dynamic on-chip power regulation (both in terms of control circuitry and in terms of the higher voltage that will need to be applied) and its benefit in terms of reduction of leakage power, especially for temporarily inactive circuitry.

In implementing these on-module regulators, care must be taken to both remove the heat from these devices in such a way as to not raise the temperature of the processor itself (and thus limit its performance), and to find efficient means to provide the input power to these converters. Good progress is being made in both of these directions, but more research is needed, particularly into the efficiency of the local regulators.

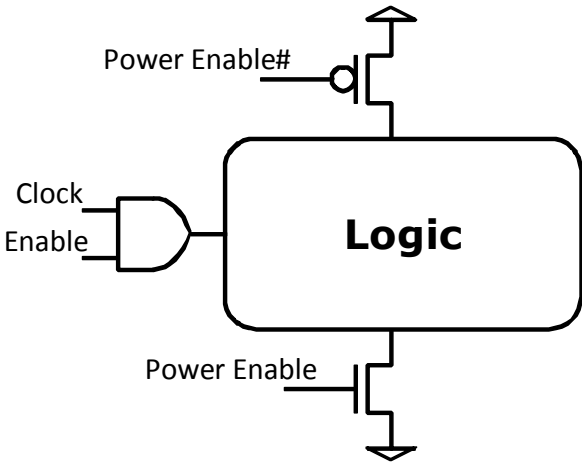


Figure 4: Clock and power gating.

3.7 System-Scale Power Management

Power Distribution Typical servers today deliver power from wall plug to processor chip pins at efficiencies of about 70%. With appropriate research an exascale machine could target 90% efficiency on this path. Areas to investigate include:

- Integrated, highly reliable single stage converters
- On-chip power conversion technology
- Monolithic micro-buck converters
- Low-resistance, high-current connectors
- Lower resistance packages.

Although 20 MW is a very low total power target for an exascale system, to efficiently move this power to the computer racks the datacenter will want to target at least 480 V AC or high voltage (400 V) DC distribution. For safety reasons this voltage should be transformed down to 48 V DC for distribution in the rack, which is high enough to keep distribution losses low but low enough to be safe for handling. Continued research on efficient AC rectification and DC isolation, as well as exploitation of wide band-gap semiconductor technology, could make this stage upwards of 98% efficient. Then DC-DC conversion from the 48 V to whatever voltage is required for the on-module or on-chip power converters, plus any residual electronics being powered directly, should be made with a single stage, highly reliable converter with at least 95% efficiency. Wide band-gap semiconductor technology, coupled with better magnetic materials for the transformers and lower resistance power connectors, will allow even higher efficiencies. These new power semiconductors will require precisely tailored gate control and new very low- inductance, low-resistance packages, requiring research in these areas. The switching speed should be considered in concert with the on-module converters, so that the system can react as efficiently as possible, and allow processors to be powered at the lowest possible voltage. In general higher-frequency, faster-response converters are desirable (as they can react faster to transient loads and reduce the storage requirements of downstream converters and local capacitors). This, in turn, will drive the need for research into lower-inductance power connectors which will allow the converters to be placed inductively and

resistively closer to the load. Finally the development of very compact and efficient monolithic buck-converters will allow the current trend of very specialized voltages for system-on-chip design to continue. All power converters should be made as reliable as possible, with the target of avoiding the use of redundant power supplies, which ultimately cost efficiency both through the use of ORing FETs for fault isolation and by causing the converter to operate in the less efficient low-current part of its range.

Cooling Almost certainly exascale systems will be water cooled to minimize floor space and reduce the cost (and energy) of cooling. A fault tolerant exascale system means nodes can be replaced while the machine runs. Hence, each node must be independently pluggable.

Designs would profit from research in

- Ultra-low thermal resistance separable thermal connections
- Improved thermal interface materials
- Very low cost, highly reliable quick-connect
- Flexible, fire-safe, highly durable polymer hoses
- Micro-channel cooling and other means for heat extraction at the component level.

Ultimately the cooling reservoir for the 20 MW Exascale system is the atmosphere. The thermal challenge can be thought of as the most energy efficient means to move the heat from the electrical and optical components of the system to this final atmospheric heat sink with minimal thermal loss along the way. Ambient water, in thermal equilibrium with the atmosphere but chemically non-corrosive, biologically inert, and without particulates that could clog small cooling orifices, is a excellent cooling fluid. However, connectors are needed to allow service of the hardware. Direct water cooling technologies (cold heads, micro-channel coolers, spray coolers and the like) will need quick-connects that allow the water hose to be disconnected and reconnected as connect circuit cards are today - at low cost, blind mate, low resistance, etc. Such quick-connects do not exist today and they impact all water cooled designs in a dramatic way. When using lower cost, but less thermally efficient indirect cooling means, such as vapor chambers, heat-pipes, thermal spreaders and the like, highly thermally conductive dry thermal interfaces are needed. Such technologies have been used for years in military applications but are still very expensive and too thermally insulating for broad use.

Finally local and national fire codes require use of fire-safe materials within the rack, even when carrying cooling fluids. Today this results in either the use of very thick, unwieldy hoses which detract from dense packaging, or very expensive formed and brazed tubing. Chemical research in polymer hoses can overcome these limitations and clear the way to very efficient water cooling direct to the thermal loads.

Packaging Large scale HPC systems benefit most from compact node designs indicating the need for research in:

- 3D technology
- Optical connectors

Three-dimensional chip structures can be used to dramatically lower power by allowing many more and electrically shorter connections between chips, but they carry with this even higher local power densities that drive power delivery and cooling concerns, as well as increased fragility owing to wafer thinning to expose chemically etched through-silicon vias (TSVs). Packaging research

aimed at overcoming these new problems will be well received. As these 3D structures are grouped into highly energy efficient, but still higher energy, enclosures, users must retain the ability to easily service them, preferably while the rest of the equipment is still running. A the shift from 12 V to 48 V DC power will reduce distribution loss, but research which allows this higher voltage to be safely accessed may be required.

Most importantly, the pervasive use of optics in the exascale machine will require several types of optical connectors. It is entirely possible that the most difficult technical challenge to the widespread use of optical technology is the development of the required connectors. For example, optical transceivers, whether VCSEL or silicon photonic based, once integrated into the system-on-chip processor require a very high I/O density optical connector to allow the processor to be socketed and field replaceable. Such connectors are currently unavailable. A second connector is required to allow optical connections to be made between servers. Alternatively, use of active optical cables, plugged close to the source, require currently unavailable midboard connectors. Use of high port count (48 ports or more) top-of-rack switches, or larger datacenter switches based on collections of these switch chips, will require compact and closely packed optical connectors for the passive cables that connect these components. Alternatively, new packaging approaches may be developed that allow the use of anticipated dense active optical cables plugged immediately adjacent the switch chips.

3.8 Impact

Following a business-as-usual approach in system design, an exascale machine is expected to consume gigawatts of power. The circuit, memory, interconnect, and platform technologies described above, both evolutionary as well as revolutionary, are expected to bring the power consumption closer to the goal of 20 MW. With hardware/software co-design approach, these technologies have potential to provide even further energy benefits to help meet the exascale power and energy goals.

4 Interconnect Technology

Perhaps the most critical barrier toward realizing exascale computing is the fundamental challenge of data movement. This is for both *vertical* data movement between processors and memory as well as *horizontal* data movement between processing nodes. With the extraordinary growth in parallelism, performance is increasingly determined by how data is communicated among the numerous compute resources, rather than the arithmetic operations performed. At exascale, these challenges are daunting, as energy consumption is increasingly dominated by the cost of data movement, and is thereby the main constraint on ultimate performance.

4.1 Data Movement Energy and Bandwidth Challenges

The fundamental limitations imposed by increasing energy consumption associated with moving vast data among the growing parallel compute resources has led to the so called "bandwidth taper" prevalent in current system architectures. To illustrate these interwoven bandwidth-energy challenges, Figure 5 shows the bandwidth taper (in blue) for conventional electronic interconnect technology. The units are Gbps/mm of horizontal cross-section, and bandwidth decreases by orders of magnitude as data propagates from on-chip, across the module, over the printed circuit boards, and onto the racks and the whole system. The energetic consequences of this are illustrated in Figure 6 which shows the relationship between the growing energy costs of data movement and system distance. For scaled conventional electronic interconnect technology (in blue), off-chip and inter-node communications face an order-of-magnitude energy wall associated with shrinking aggregate bandwidth and overall energy efficiency. Clearly, revolutionary data movement technology is required to create future exascale computing systems that are energy efficient and can truly scale in performance on real applications, and as Figures 5 and 6 show, photonics offers a potential solution.

Figure 5: Bandwidth vs system distance

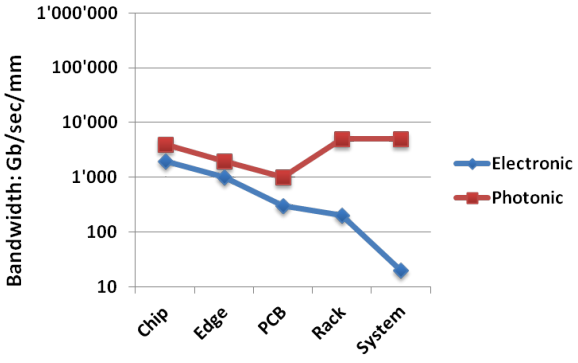
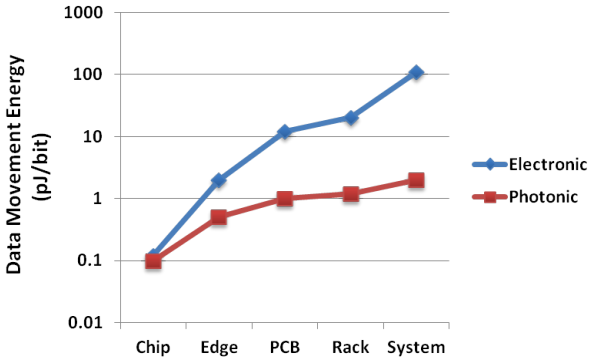


Figure 6: Energy vs system distance



The main challenge for interconnect technology has always been to provide high network bandwidth and message rates, while minimizing latency. Equally important is how effectively applications are able to utilize the network. [66] Historically, many interconnects have been built to provide these capabilities without regard to the communication mechanisms being used by the applications running on the system, leading to a bad semantic match between the communication primitives required by the applications and those provided by the network. The resulting poor utilization of the interconnect is bad for both application performance and energy efficiency.

4.2 On-Die Interconnect Fabric

An on-die interconnect fabric connects cores on the die to the memory subsystem on and off the die and to the external interconnect fabric. The challenge is that the die size of the processor die will remain constant across process technology generations, and therefore the on-die interconnect energy will not scale as much (as described in section 3.4).

Investigation is needed into various on-die signaling technologies, such as repeated single-ended, differential low-swing and twisted interconnects, to determine the best interconnect technology for the processor chip. These techniques are well known, but need to be examined and methodologies developed in the context of the extreme scale processor.

Several on-die network topologies and switches have been studied, [6, 58] such as a traditional bus, 2D mesh or torus with routers, crossbar switches, butterfly networks, and fat-trees, as well as circuit and packet switched networks. All of these show different characteristics, with each having benefits if used appropriately by the underlying software. The hardware choices have deep implications for the design of the communication libraries (e.g., MPI or Global Address Space), the runtime systems (scheduling and active message dispatch to the correct thread) and ultimately the applications and algorithms. The challenge is to develop network designs using these techniques that are applicable to the exascale processor, evaluate them in the context of different candidate software systems using simulations and emulations, and mature these for exascale systems.

We need to consider implementing a hierarchical network, with potentially different sub-networks at each level in the hierarchy, using the technique most suited for that level in the hierarchy, considering energy efficiency and bandwidth. For example, connecting cores in a local block is best achieved using a bus, since a bus is the most energy-efficient solution for short distances. Blocks can be connected together using a wider bus or a packet-switched network such as a mesh, or a crossbar, depending on the size and distance. It remains an open question as to whether current computing applications and algorithms can be redesigned to exploit these different potential hardware realizations. For this reason, hardware and software design must be closely coupled and not deferred until the machine arrives.

Thus, it is likely that for a processor die with O(1000) cores we will have to evolve from the conventional approach of a homogenous interconnect fabric across the chip with constant bisection bandwidth to a revolutionary approach of a hierarchical, heterogeneous, and tapered on-die interconnect fabric.

4.3 Inter-chip Network Integration

The main function of the inter-chip interconnect is to provide the processors with access to remote memory. As such, the mechanisms used to integrate the network with the rest of the node architecture (processor and memory) can have a dramatic impact on how and what type of features can be effectively supported by the communication runtime. There are many available options for integrating the network. Logically, the network can be connected as an I/O device, or as a peer on the processor or memory network. Additionally, the network interface controller (NIC) can be either coherent or non-coherent with the processor cache hierarchy. Physically, the network can be integrated on-die, on-package, or on-board. Each integration choice comes with a set of tradeoffs.

As CMOS process technology continues to advance, it has already become feasible to integrate portions of the network on the same die or package as the processor. This tighter integration can provide significant energy savings, as well as opportunities for significantly enhanced functionality. By moving significant packet processing functions from software into the networking hardware (i.e, "processing-in-network"), this trend has also been contributing to reductions in network latency.

A major challenge with integrating the network for high performance computing will be identifying the proper function to include in the NIC, as well as identifying a suitable network protocol to enable cross-vendor compatibility. Commodity computing will inevitably drive this market toward integrated networks designed to work well for the general Internet. Such networks will provide no advantage to high end scientific computing.

On the other hand, careful selection of advanced network capabilities, such as remote direct memory access (RDMA), remote atomic memory operation support, virtual memory enhancements to reduce overhead of memory registration, integration with complex memory hierarchies, advanced message completion notification, communication protocol offload, in-network processing of collective operations, and active messaging support, could enable the extreme scalability required for exascale computing. Making the best tradeoffs in enabling these capabilities will require a deep understanding of the requirements of large scale scientific computing applications.

4.4 Photonics

Among the emerging technologies that could contribute to an energy-efficient interconnect, photonics is perhaps the most promising. Optical technologies can directly impact the critical communications challenges within computing systems through their remarkable capabilities to generate, transmit, and receive ultra-high bandwidth signals with both fundamentally superior power efficiencies as well as inherent immunity to noise and degradation. Unlike prior generations of photonic technologies, recent results in silicon photonics offer the possibility of creating highly integrated platforms with dimensions and fabrication processes compatible with electronic logic and memory devices [78]. During the past decade, a series of breakthroughs in silicon photonic devices have demonstrated that all the components that are necessary to build chip-scale photonic interconnect components (e.g., modulators, filters, switches, detectors) can be fabricated using common CMOS processes [14]. Figures 5 and 6 outline how photonic technologies can break through the bandwidth and energy consumption tapers that impose severe limitations on conventional system scaling.

It is important to point out that the key aspect of chip-scale photonics is that it is not related to the cross-chip communication-distance scale. With respect to optics, on-chip, board-scale, and cluster-scale distances are all equivalent in throughput performance. This fundamental property of photonics contrasts with electronic communications, which must adhere to highly constrained bandwidth-distance product restrictions. Optical interconnection networks employing these silicon photonic building blocks therefore present an unparalleled opportunity for transformative advances capable of fundamentally altering the roadmap to creating truly scalable future computational platforms.

Besides the well-known improvements that optical communication can offer in terms of transmission latency, energy-efficiency, and bandwidth density, the critical advantage enabled by chip-scale silicon photonics is the close proximity of electronic driver circuitry integration. This allows for the ultra-low energy transmission and reception of optically encoded data that is distance independent: once the data is encoded in the optical domain, the propagation distance, whether it's 1 cm, 10 cm, or even 100 m, is immaterial to the link performance [95].

The insertion of photonics into next-generation high performance systems is not a one-to-one replacement. The potential impact of chip-scale photonics on exascale computing systems can be realized only by re-evaluating the composition of computing systems and understanding how data movement in the optical domain can be leveraged to create new architectural paradigms. New network architectures include queuing structures and strategies that provide synergetic hardware/software support for collectives, embedded communications and computation functions, and reliability [120], [59].

The advanced network capabilities mentioned in Section 4.3 further complicate the insertion of photonics. Such capabilites are being introduced into current and near-future high performance computers, which benefit from the enhanced density of CMOS integration, allowing more and more function to be integrated in the NIC and/or the network switches. Many of these functions will be deemed essential to exascale computing. The extent to which the development of on-chip photonics will be able to implement or incorporate these functions, in a cost-effective and energy-effective manner, will limit its initial photonics deployment in pre-exascale systems unless there is substantial government investment to accelerate the insertion of this technology.

4.5 Impact

Data movement challenges impact the entire system stack: from the physical architecture and memory hierarchy to the algorithms and software development that must manage the immense growth in parallelism. High performance interconnection networks that are intimately co-designed with the hardware and software and take advantage of the transformational advantages of increased CMOS processing capabilities and photonics will directly impact the execution performance and scalability of exascale DOE applications. The co-integration of the network processing with the compute and memory node as well as the insertion of inter-node optical data movement will enable the creation of future classes of exascale systems with extreme system-wide bandwidth dense communications and scalable energy efficiency.

5 Memory Technology

Memory is crucial component in any computing system. To meet a variety of needs and uses, several memory types have found their way into today's systems: NOR Flash for boot storage, SRAM cache for high speed code and data storage, DRAM for high density and high speed code and data storage, NAND Flash for persistent, block-oriented read and write storage, and hard drives for very high density, cost-effective, low speed block-oriented storage. Volatile memory loses data when powered off; nonvolatile memory retains it, and can compete with disk for long-term storage. This section addresses the challenges facing both the volatile and nonvolatile memory technology needed for exascale systems.

Today, DRAM and NAND dominate in the two central roles: volatile, high-density, high-speed code and data memory (DRAM), and nonvolatile, low-latency, high-density, block-oriented data storage (NAND). Some new nonvolatile memory technologies, currently on the horizon, have the potential to make a disruptive impact in the exascale timeframe. Examples include spin-transfer torque (STT-RAM), phase-change (PCRAM), and metal-oxide resistive (ReRAM, or memristor). While these memory technologies are not in volume production today, the issues of exascale are sufficiently challenging that alternative technologies should be examined for their disruptive potential. Notably, one cannot ignore the possibility that a change in memory technology may have a profound effect on architecture and software, and may be the best path to a cost-effective, low energy, high performance system.

A lot has changed since the original DARPA Exascale report [71], including the consolidation of the memory industry and the emergence of NAND Flash as the predominant memory type. In examining a post-2020 deployment, the pace of Moore's Law will continue to slow as devices reach their physical limits. Before those limits are reached, process complexity and cell interference may impact memory scaling for economic reasons. The problem of cost-effective memory capacity is a central concern that is discussed below.

5.1 The Challenges

Memory Capacity: Capacity is critical to applications. It allows numerous problems to be solved in parallel, a powerful form of weak scaling; it allows in-memory checkpoints and message logging/replay for resilience; and it enables algorithms that buy performance by using data structures that may not be minimal in their memory footprint. There are numerous examples of this space/performance tradeoff. The primary challenge will be the continued scaling of memory density on a per-compute-operation basis (bytes per Flop/s).

Figure 7 shows that the machines at the top of the TOP500 do not have sufficient memory to match historical requirements, and the situation is getting worse. This is a big change from the traditional one byte per flop that the NNSA labs prefer in support of their application base. It places the burden increasingly on strong scaling of applications for performance, rather than the weak-scaling model which dominated the terascale era. Interestingly, the "tipping point" in the graph is in the 2003-04 timeframe, which coincided with the beginning of the end of Dennard scaling and the start of the multicore era.

One reason is economic, with users often prioritizing arithmetic performance over memory capability. This critical capacity challenge can be addressed by combining volatile and nonvolatile memories into a single programmer-addressable device. However, there are numerous architectural challenges that emerge from integrating the two. These include, but are not limited to, the relatively poor latency, especially write latency, of some of the nonvolatile memories; a lack of abstracted memory protocols which decouple timing and naming from the device interface; differences between

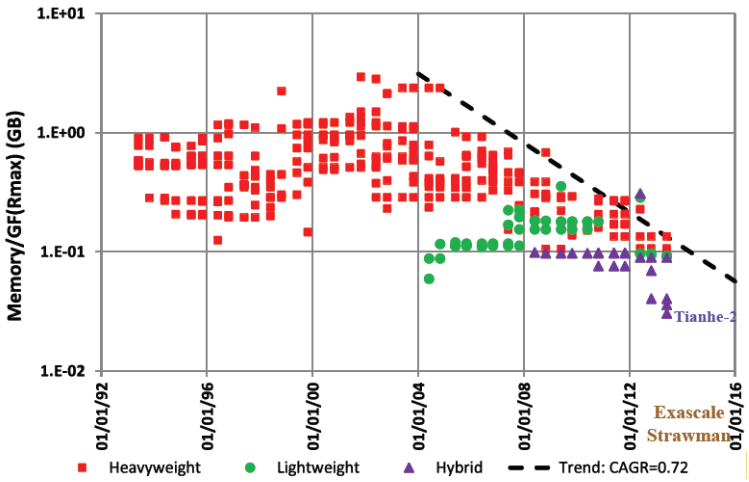


Figure 7: Memory capacity (Bytes) per gigaflop/s, courtesy of Peter Kogge.

the block-oriented access typically favored (for reasons of chip architecture and error management) in a Flash device and the word-oriented access typically favored by CPUs and volatile memory; and differences in read latency, wearout, soft error vulnerability, and other differences between any two classes of memory that are hybridized.

5.1.1 Energy

Energy is the primary technical challenge in supercomputing, but the least likely to be observed directly by the programmer. Memory faces three primary energy challenges:

1. Processor/Memory Transport: DDR-style memories will have reached the end of their useful lifespan by the exascale timeframe; more energy-efficient transport will be required.

2. Processor/Memory Protocol: Since the invention of DRAM, the interface between the CPU and the memory system has been slaved in both naming and timing to the CPU. This produced the least expensive devices possible (from an acquisition cost perspective) at the expense of centralizing a rigid memory-management model and limiting memory evolution. Memory energy consumption, error mitigation, and data resiliency are removed by this model from the memory, and are managed by non-memory vendors, a situation which is poorly considered in the complexity of DDRx specifications. Instead, distributed control with decoupled naming and timing must be enabled.

3. Memory Architecture: Typical commodity DRAMs are architected for the lowest possible cost per bit, which requires significant "over-fetching" of data for any given memory request. This refers to the granularity of access in the memory devices, in which a "row" of bits is destructively read from a memory array into a row buffer, in each DRAM chip involved in a cache line read or write. The data are fetched into these row buffers in chunks of 16 kilobits or more. Usually, most of these bits are neither read nor written by the processor and must simply be written back to the memory array at some energy cost. Furthermore, these memories often provide a weak banking structure for the same cost reasons. Enabling finer-grained access of a larger number of banks provides significantly improved performance and potentially lower energy profile [See, for example, Jung Ho Ahn, Norman P. Jouppi, Christos Kozyrakis,

Jacob Leverich, and Robert S. Schreiber. 2009. Future scaling of processor-memory interfaces. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*. ACM, New York, NY, USA.]

The Hybrid Memory Cube (HMC) roadmap provides examples of how to address each of these problems from a technology perspective, but it is not the complete answer. In particular, HMC demonstrates a path to manufacturable technology with sufficiently low transport energy, an abstracted, packet-based protocol, and a rearchitected DRAM structure designed for higher performance. This is one example of a potential future direction.

5.1.2 Scaling

Manufacturing at commodity scales and low cost continues to be a problem. Wafer process cost and fabrication capital cost are growing because of the slowing of Moore's Law. This has resulted in a significant slowdown in the building of new DRAM fabrication capacity. In fact, today there are no new fabrication facilities on the horizon.

None of the emerging alternative nonvolatile memory technologies has yet achieved a near-term, cost-effective, manufacturable path to replacing commodity memory. Further research in this area could be highly disruptive, which is a desirable outcome for the long term health of the high performance computing field.

5.1.3 Resilience

The increasing number of components required for an exascale system contributes to additional complexity in resilience, which will be compounded if memory capacity trends are addressed by a potential exascale program that would require increased memory density. In addition, today's slaved timing and naming require a centralized approach to failure recovery. This implies the following issues:

1. It is becoming difficult for memory controllers designed by CPU vendors to map out bad or problem rows. Recent "row hammer" disturb issues (in which repeated accesses to the same row may alter the state of that row's neighbors) demonstrate that an intimate understanding of internal part topology may be required to address memory error conditions. These kinds of issues will continue to present themselves as technology scales.

2. Just as NAND is a fundamentally lossy media, which requires significant error correction performed transparently to the user, opportunities exist to enable more resilient memory systems by performing distributed error recovery.

3. Variable Refresh Time (VRT) errors, in which some DRAM cells require unpredictable refresh times, point towards the benefits of a more distributed approach to resilience and recovery, as well as more intimate knowledge of individual DRAM parts and processes. Similar issues arise in alternative memory technologies.

On the positive side, the deeply buried capacitor structure used in modern DRAMS makes each generation of DRAM more immune to SEUs than prior generations. SRAM, on the other hand, has increasing vulnerability as it scales. This dichotomy has a potentially profound effect on cache architectures, which may be significantly less reliable (and require increased error correction) than main memory. Alternative memory technologies may address some of these problems.

Generous memory capacity can be a boon to system-level resilience through frequent checkpoints. Checkpointing exascale state to disk via a storage network connection will become impractically slow; in-memory distributed checkpoints are a scalable alternative.

5.1.4 Systems Tradeoffs and Integration:

The government is uniquely positioned to enable a whole-system approach with multi-vendor teams, academia, and the national laboratories to find the best solutions for entire classes of applications. The exascale problem is sufficiently challenging that a business-as- usual approach is not likely to work. Typical design patterns have forced a component-oriented approach to constructing systems, which is a challenge to the proposed DOE methodology of hardware/software co-design. Examining individual costs (e.g., "cost per bit of capacity", "cost per bit per second of bandwidth", etc.) rather than system costs (e.g., "how to design the least expensive system for a given aggregate bandwidth and capacity over the lifetime of the system") often leads to per-subsystem or per-component tradeoffs resulting in an overall inefficient design.

Device manufacturers are investing in the basic layer of new alternative memory and storage technologies – devices and media controllers. A co-design approach in which the opportunities and implications for the memory channel, cache hierarchy, processors, and algorithms is explored in tandem with the DOE's applications is required to best assess and then profit from this opportunity, should any of these technologies "make it" as a commercially viable, low-cost alternative.

5.2 Research Directions

Given the technology challenges in the 2020 timeframe and the likelihood of these challenges to intensify over the lifetime of exascale-class systems, it would be beneficial to invest in further development of emerging memory technologies as well as possible process extensions to DRAM. Several "DRAM replacement" candidates already exist in the laboratory. While the maturation of these to mass production remains uncertain, the possibility of creating valuable alternatives makes them a sound strategic research investment.

The new nonvolatile technologies offer an alternative attack on the capacity wall. The layering of a crossbar memory array on top of physical addressing circuits can produce a highly dense memory structure. Research directed at the optimization of this approach would be helpful.

It is important that several emerging nonvolatile memories will be byte-addressable, making them desirable candidates for main memory usage. For use as main memory, considerable durability is necessary. Research at the device and the circuit/media control levels is needed to achieve simultaneously the density, durability, performance, and energy efficiency needed for use as a main memory technology.

Compared to current memory systems, emerging technologies have some commonality, and some important distinctions in how they relate to exascale computing. First, they share the trend of abstracting the memory system from the CPU, with distributed control including media controllers clustered with NVM physical resources. They also share the benefits of advanced physical packaging such as through-silicon vias (TSVs) that result in increased volumetric density and intimate local interconnect channels. As with DRAM, some emerging memories can be realized across a range of access latencies and densities, depending on device technology and circuit architecture. Some candidates also generally have asymmetric read and write performance or endurance limitations.

Research is also needed to understand overall system optimization accounting for capacity (cost)/latency tradeoffs in order to guide direction in memory architecture. Within the memory module, many low-level aspects of the media controller/ physical media interaction deserve atten-

tion. Examples include coding, read/write scheduling, and wear-leveling or mitigation schemes to minimize power consumption and maximize endurance. Research should address the low level issues, such as manufacturability, and the overall system impacts of changes to the memory technology, which would have a significant ripple effect.

Given the importance of abstracted memory systems (with distributed control and decoupled timing) as outlined above, groups of vendors (processor, memory, integrators) should work on the potential of lowering overall system cost through memory systems that are abstracted, distributed, and more resilient. This would also serve as a vehicle to explore unified memory systems of heterogeneous memory types.

Finally, enabling even limited processing capabilities near memory has been demonstrated to significantly improve performance and lower energy by numerous industry, lab and academic research efforts. Multi-vendor team exploration of processing-near-memory architectures (e.g., on the logic base of a 3D stack) in the context of overall system performance for DOE applications has a significant potential to yield early fruit.

5.3 Impact

Decisions on memory provisioning, interfacing and architecture will ultimately determine the capability, cost, and energy profile of future computing systems. The best of today's memory interfaces and density need to be further improved if exascale class systems are to deliver the promised performance under real applications. This improvement must be done in an informed manner which maximizes the potential of the machine while simultaneously embracing the capability of new memory technology.

6 Scalable System Software

System software plays a critical role in creating a unified computing resource from a collection of hardware components and ensuring that applications are able to maximize the capabilities of the underlying hardware. The ever-increasing core count of processors means that there is a massive amount of on-node parallelism for the system software to manage. Operating system (OS) resource management strategies have focused on accommodating a small number of threads of execution within an address space. Most current HPC operating systems were optimized to allocate compute and memory resources to a few, fixed number of threads and then be as non-intrusive as possible, and the role of the runtime system was very limited. As cores become plentiful and more lightweight, applications are having to extract and expose as much on-node parallelism as possible. This approach leads to a situation where applications will have more threads than cores, threads will be created and destroyed much more often, and running threads will need to synchronize more often. This dynamic behavior creates new challenges for the operating system and the runtime system. Rather than trying to be as non-intrusive as possible and let the application implement its own resource management policies, the system software must now play a more active role in making decisions about how resources are allocated and managed, and the strategies for managing these resources can be very different for different applications. Applications are now dependent on the runtime system to achieve performance and scalability, and the runtime system is now dependent on the underlying operating system to provide the appropriate interfaces and functionality to support the needs of the runtime system.

In many ways, runtime systems are taking on tasks that humans perform today. Current system software was designed and optimized so that application performance and scalability are largely the responsibility of the application. Application developers identify performance issues by using performance analysis tools and iterate over the possible solution space until an acceptable approach is discovered. In order to achieve the goals of exascale, the runtime system will need to take on more responsibility for delivering performance from an application. The runtime system will need to inspect the state of the resources, analyze potential ways to use them more efficiently and effectively, and then respond appropriately. This challenge is compounded by the fact that hardware reliability is not improving. Hardware components may fail without warning, further complicating the ability to manage resources. An additional challenge is to design and implement a dynamic adaptive runtime system that is portable across multiple hardware platforms.

The other essential challenge for system software is that the fundamental cost model for resources is changing. Processors and memory systems have spent several decades trying to provide hardware support for determining how best to use memory resources including cache policies, TLBs, virtual memory, pre-fetchers, etc. In most cases, these mechanisms have induced overhead for HPC applications, but the costs could be mitigated. This is no longer the case. Architectures are providing more support for explicit versus implicit management of memory resources because the cost/benefits tradeoffs are changing. Minimizing data movement is increasingly important.

These changing costs have impacts throughout the system. For example, the usage model for today's HPC machines is evolving from sequences of simulation and analysis tasks, communicating via long-term storage, toward a more dynamic, "compositional" approach, wherein applications consist of complex combinations of coupled codes, data services, and tools. Projected limitations in moving data motivate solutions that integrate simulation and analysis, couple complex physics codes, and develop fully integrated application workflows. Expected power constraints motivate the need to co-locate applications to avoid data movement wherever possible. New HPC use cases for streaming and graph analytics require features of the OS/R such as global addressing, massive multithreading, and event-based processing that are not well supported on existing HPC systems.

6.1 Research Directions

The changing costs motivate a reexamination of the structure of operating systems and the hardware mechanisms needed to support applications. The cost of virtualizing memory resources, through mechanisms such as demand-paging and multiple levels of cache hierarchy, continues to increase for HPC applications. Applications software and system software go to great lengths to avoid the penalties associated with these mechanisms. Applications avoid using demand-paged virtual memory. Caches, which were designed to be invisible to applications, have instead motivated entire areas of research, such as autotuning. Other hardware mechanisms, such as interrupts and traps, were designed when compute resources were scarce and asynchrony was an abnormal occurrence. As cores become lightweight and plentiful, the operating system structure should evolve to embrace asynchrony and alternative hardware mechanisms should be explored. This co-design of hardware with system software will be critical to ensuring that operating systems and applications don't continue to focus on strategies that work around the capabilities of the hardware.

Researchers today are already exploring the interface between the operating system and the runtime system to ensure that the capabilities needed by an introspective and adaptive runtime system are supported by the operating system. As runtime systems become more dynamic, the operating system needs to provide a richer set of interfaces and mechanisms, and enable the composition of multiple applications to create more complex workflows. Investigators are already taking the first steps towards understanding the requirements of system services as different levels of the system, both node-level and system-wide [3, 19, 43]. In the process, they are creating the infrastructure needed to support the exploration of more complex issues, such as resilience, scheduling, and energy efficiency.

However, much work remains to explore and understand the hardware and software mechanisms needed to meet these requirements. System software for extreme-scale computing will have to deal with the challenges of billion-fold concurrency and associated locality, new architectural concepts, and emerging workloads. This will support the creation and evolution of new programming models, computational libraries and intelligent runtimes. A rich set of tools (interactive and visual) will also be needed to enable more productive application development and to allow users to understand and optimize the execution behavior of an application. Very low noise kernels and operating systems need to be developed, as well as support for the hybrid memory structures described above. Finally, research will be needed into robust system and resource management, and in monitoring and control systems at extreme scale. Aspects of these are discussed in the next few subsections.

6.2 Lightweight OS

Exascale system nodes, containing hundreds or even thousands of cores, will challenge current operating system practices. Many of the fundamental principles that underlie current OS technology are based on assumptions that will no longer be valid for an exascale processor. In the context of exascale system requirements, as machines grow in scale and complexity, techniques to make the most effective use of network, memory, processor, and energy resources are becoming increasingly important. In its role as gatekeeper to all these resources, the OS becomes a major obstacle in allowing the application to view the hardware, or the hardware state. A baseline challenge for the exascale software stack is how to get the OS out of the way without compromising the need to protect hardware state from errant (or malicious) software. Execution models that support more asynchrony will be necessary to hide latency. Such execution models will also require more carefully coordinated scheduling to balance resource utilization and minimize work starvation or resource contention. These execution models will also require extraordinarily low overhead fine-

grained messaging. However, the attributes required by the execution model are nearly impossible to achieve when the OS intervenes for every operation that touches its privileged domain. Today it must intervene for inter-processor communication operations, has exclusive and privileged control of scheduling policy, and exclusive ownership of resource management policies.

In its role as the gatekeeper to shared resources, the OS has traditionally been a major bottleneck in achieving scalability on SMPs. This is especially true for the open source Linux operating system. Over time, OSs have evolved into multifaceted and hugely complex software implementations that have accreted a broad range of capabilities. The challenge of breaking the OS apart based on separation of concerns is typically referred to as "deconstructing the OS." Below is a subset of leading issues that motivate the need to re-examine the underlying assumptions that are encoded in current OS implementations and fodder to imagine a re-implementation of the minimum necessary capabilities based around new realities of hardware design. High-end systems typically use specialized operating systems for compute and I/O nodes, and standard operating systems for service, front-end, and file-server nodes.

In the case of BlueGene, the specialized OS operates at the level of a processing set (pset) which consists of one I/O node and a collection of compute nodes. The design of the compute node kernel (CNK) was simplified by placing a number of restrictions on the application (e.g., absence of virtual paging, and support for only a handful of system calls in Linux). While this design approach achieved the design goals of the BlueGene systems, scaling such OS services to exascale systems with a thousand cores per chip, and dynamic, asynchronous, and irregular parallel structures remains a grand challenge. A co-design approach is recommended to discern whether the advantages associated with dynamic, asynchronous, and irregular parallelism balance against the fundamental design choices for exascale operating systems.

6.3 Runtime Systems

When considering the challenge of supporting billion-way parallelism for an exascale system, it is widely agreed that the bigger disruption will occur at the intra-node level rather than the inter-node level. This is because the degree of intra-node parallelism needs to increase by about three orders of magnitude relative to today's high-end systems on the path to exascale, while the degree of inter-node parallelism only needs to increase by at most one order of magnitude. Other challenges at the intra-node level include vast degrees of performance and functional heterogeneity across cores within a node, as well as severe memory and power constraints per core. Taken together, these challenges point to the need for a clean sheet approach to intra-node runtime systems. They also have a significant impact on inter-node runtime systems, because any successful inter-node runtime system for exascale systems must be able to integrate well with new intra-node runtime systems (hybridization).

Though there is a large variety of execution models under consideration for exascale computing, there is also a growing consensus in the community on key abstractions that will need to be present in an exascale execution model and that the runtime system (assisted by the compiler and architecture) must provide first-class support for these abstractions. These abstractions are expected to lead to research and development challenges in lightweight tasking, managing data movement in memory hierarchies, coordination of dynamic task teams, managing heterogeneity and in-situ runtime monitoring. These are discussed below.

It is widely accepted that the parallelism exposed by an exascale application must be over-provisioned by 1-2 orders of magnitude relative to the available parallelism in hardware on an exascale system. Even with lightweight OS kernels, it is unlikely to be practical to use OS threads for this over-provisioning, especially due to the memory overheads imposed by standard OS threads.

Thus, an exascale runtime system must support large numbers of lightweight tasks that are scheduled on a fixed number of "worker" OS threads (typically, one per core or hardware context). A number of recent multicore programming models (e.g., OpenMP 3.0, Cilk, TBB, and a multitude of DAG-parallelism schedulers) already assume the availability of lightweight task parallelism. In an exascale runtime, there will be additional challenges in supporting lightweight tasks that include support for heterogeneous processors and integration with asynchronous inter-node communications. Support for end-to-end asynchrony across tasks and communications remains a major challenge. Therefore, lightweight asynchronous tasks need to be developed as a fundamental primitive for computation and communication across software and hardware; further, general support for *task preconditions* can be used to include both computation and communication events (thereby encompassing event-driven and message-driven execution models).

While memory hierarchy details will vary from platform to platform, there is agreement that exascale software will have to deal with deep memory hierarchies combined with the need to minimize data movement due to energy limitations. Thus, the task scheduling runtime capabilities discussed above will need to be locality-aware and be capable of supporting function shipping and data shipping as interchangeable alternatives. In general, all data movement (memory accesses, DMA transfers, inter-node messages) should be abstracted as asynchronous tasks whose completion can trigger additional computation tasks and data movements.

Past approaches to coordinated parallelism such as thread groups and communicators involved a fixed number of threads (e.g., the set of threads or processes synchronizing on a barrier does not change after the first barrier operation is performed). However, dynamic task creation and termination is expected to be a frequent and common operation in future exascale systems due to large degrees of variability across processors and also due to resilience requirements. Thus, there's an urgent need for synchronization operations that can be performed on dynamically varying sets of tasks. These synchronizations can sometimes be manifested as task termination operations (e.g., sync, taskwait, finish, future), but additional synchronization (e.g., barriers, phasers, mutual exclusion) may be needed in the midst of task execution. Mutual exclusion has received a lot of attention recently with software and hardware approaches to transactional memory so as to avoid the scalability and programmability limitations inherent in locks. Additional approaches to consider include the use of accumulators (lightweight embodiment of map-reduce, with better scalability than locks) and the use of actors, both of which avoid data races in different ways (accumulators are well suited for race-free deterministic parallelism, and actors are well suited for race-free nondeterministic parallelism). For all of the above, there is a general need to determine which task coordination primitives will be best suited to supporting exascale applications on exascale hardware.

Exascale systems are anticipated to have an increased number of sources for non-uniform execution rates that contribute to load-imbalances, which include adaptive algorithms, non-uniform application of power management features, and recovery costs for software-based fault resilience mechanisms. It will be useful to have an API that enables a runtime system or an application to measure distances in a machine by comparing two memory addresses, or by comparing the physical placement of two process ranks. Such information would provide useful cost information that can be used by a runtime load-balancing algorithm to make rational decisions.

Finally, as discussed in section 7, monitoring tools will be critical for performance optimization and correctness debugging. There is a need for research and development to determine how best to use monitoring to provide performance and debugging feedback to users in a prompt and efficient manner.

6.4 Introspection

A self-optimizing adaptive control system requires a method for monitoring current operational conditions, a model for predicting the response to a control decision, and further monitoring to assess the results and apply corrections. Integrated performance monitoring and modeling will provide the intelligence necessary to control the behavior of the runtime. However, an exascale system may flood users with a deluge of performance, timing, and resource measurement information from all levels of the architecture. To make productive use of this data, it must be collected, distilled, analyzed, and presented in an efficient and unobtrusive manner. Doing so enables runtime adaption and decision making to be integrated into applications and directly into the software stack. An effective dynamic control system for extreme scale systems requires both low-level information to quantify detailed system characteristics, as well as high-level insights into expected behavior. The instrumentation informs the adaptation policy for the runtime system, and is also available to the application directly for application specific adaptations.

6.5 Energy Management

In the absence of energy management capabilities, thermal limits force compromises that may lead to highly imbalanced computing systems (such as reduced global system bandwidth). The design compromises required for power-limited logic will reduce system bandwidth and consequently reduce delivered application performance. Scalable system software can help address these challenges if appropriate interfaces are available for measurement and control of electrical energy and power use.

From an applications perspective, active power management techniques improve performance on systems with a limited power budget by dynamically directing usage only to the portions of the system that require it. For example, a system without power management would melt if it operated memory interfaces at full performance while also operating the floating point unit at full performance. This forces design compromises that limit the memory bandwidth to 0.01 bytes/flop according to the recent projections [73]. However, in this thermally limited case, higher memory bandwidth can be provided to the application for short periods of time by shifting power away from other components. Whereas the projected bandwidth ratio for a machine would be limited to 0.01 bytes/flop without power management, the delivered bandwidth could be increased to 1 byte/flop for a transient by shifting the power away from floating point (or other components that are underutilized in the bandwidth-limited phase of an algorithm). Therefore, power management is an important part of enabling better delivered application performance through dynamic adjustment of system balance to fit within a fixed power budget. Currently, changes between power modes take many clock-cycles to take effect. For current application codes, power modes cannot switch fast enough to be of use. While vendors have indicated that it is possible for power management systems to switch to low-power modes within a single clock cycle, there is still a lot of work required for scalable system software to coordinate switching across a large-scale HPC system.

Current power management features are primarily derived from consumer technology, where the power savings decisions are all made locally. For a large parallel system, locally optimal solutions can be tremendously non-optimal at the system scale. When nodes go into low-power modes opportunistically based on local decisions, it creates a jitter that can substantially reduce system-scale performance. For this reason, localized automatic power management features are often turned off on production HPC systems. Moreover, the decision to change system balance dynamically to conserve power requires advance notice because there is the latency for changing between different power modes. So the control loop for such a capability requires a predictive capability to make optimal control decisions. Therefore, new mechanisms that can coordinate

these power savings technologies at system scale will be required to realize an energy-efficiency benefit without a corresponding loss in delivered performance.

A complete adaptive control system requires a method for sensing real-time resource requirements, making a control decision based on an accurate model for how the system will respond to the control decision, and then distributing that control decision in a coordinated fashion. Currently the control loop to accomplish this kind of optimal control for power management does not exist. Predictive models for response to control decisions are generally hand-crafted (a time-consuming process) for the few examples that currently exist. There is no comprehensive monitoring or data aggregation. More importantly, there is almost no tool support for integration of power management into libraries and application codes. Without substantial investments to create system wide control systems for power management, standards to enable vertical and horizontal integration of these capabilities, and the tools to facilitate easier integration of power management features into application codes, there is little chance that effective power management technologies will emerge. The consequence will be systems that must compromise system balance (and hence delivered application performance) to fit within fixed power constraints, or systems that have impractical power requirements.

To address these significant challenges requires the capability to expose power, energy, temperature, and performance data at many levels throughout the HPC system in a standard way to the scalable system software and the application. Once obtained, acting on this data requires that control capabilities be exposed that are appropriate for each varying interface need. As systems evolve each level will play a role in statically or dynamically adjusting system parameters to application needs to maximize system efficiency. A widely scoped *power API* must be developed that addresses and specifies both the measurement and control interfaces necessary to accomplish these requirements. Finally, each system is part of a larger facility. Standard interfaces must be developed to both expose accurate and timely system metrics to the facility, and to allow facility input into system scheduling policies to enable dynamic energy aware resource utilization of multiple platforms within the larger facility.

6.6 Impact

Scalable system software provides DOE important options and opportunities to support the execution of mission applications on future hardware and system architectures. The critical component of the scalable system software is the runtime system. In contrast to current runtimes, the exascale runtime system will need the capability to manage asynchronous lightweight tasks, threads and multicore processors, with a deep multi-layered memory hierarchy integrated into exascale systems. This adaptive runtime will be required to extract application performance while responding to failures and managing power to control energy to solution. Co-design will play a key role in partitioning capabilities among the underlying hardware architecture, scalable system software stack, and DOE applications.

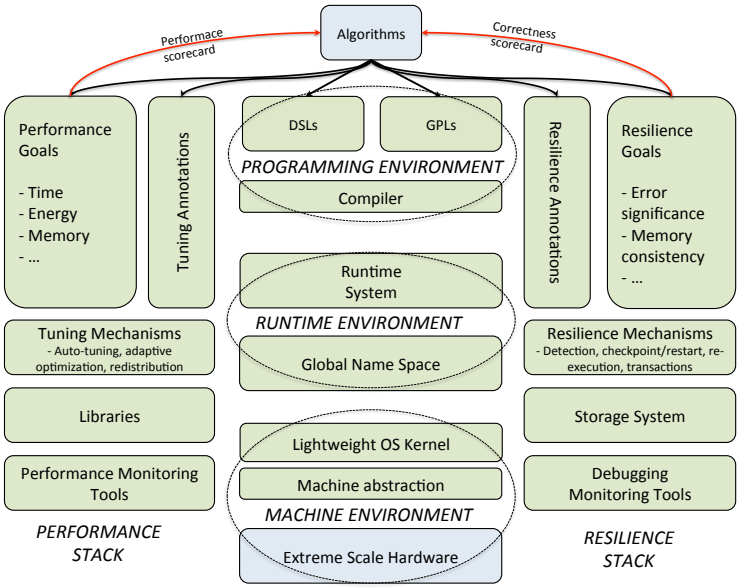


Figure 8: Notional programming system structure with programming, runtime, and machine environments, and performance and resilience "stacks."

7 Programming Systems

Programming systems encompass languages, compilers, runtime systems, and libraries, and serve as a bridge from high-level algorithms to lower-level hardware and operating-system platforms. Given the major changes underway on both algorithms and platforms, there is general agreement that advances in programming systems are necessary for application enablement on exascale systems. Further, advances in programming systems need to be coordinated with advances in hardware to ensure that application developers can achieve both correctness and performance on a wide range of exascale and post-exascale platforms without requiring heroic investments of skills and labor.

Figure 8 shows a notional structure of a programming system and its interfaces with algorithms and exascale hardware. The center column shows that the programming system includes a *programming environment* with DSLs (domain-specific languages), GPLs (general-purpose languages), and compilers, a *runtime environment* with a runtime system and a global name space, and a *machine environment* with a lightweight OS kernel, a machine abstraction, and real hardware that implements the abstraction. This structure supports a *separation of concerns* across application scientists, computational mathematicians, and computer scientists. Two important concerns for exascale computing are *performance*[1] and *resilience*, which are shown in the left and right columns of the figure. Each includes goals, annotations, mechanisms, libraries, and monitoring tools.

At a high level, the primary research and development need for programming systems is to provide a *declarative goal-driven* approach to application enablement on exascale systems, such as the notional structure shown in Figure 8. Some of the details behind this need are elaborated on in the following subsections[2].

[1] "Performance" is used as a broad term that encompasses usage of any system resource such as wall clock time, energy, memory, etc.

[2] Some of the text in this section was derived from the report on the Workshop on ASCR Programming Challenges

7.1 Programming Models and Notations

It is useful to distinguish between a *programming model*, which refers to fundamental concepts, and a *programming notation*, which refers to the actual syntax and semantics of the (general-purpose or domain-specific) language used to express programs in a given model. In general, a given programming model can be implemented in many programming notations.

There are significant research challenges that need to be addressed at both the programming model and notation levels. Large-scale DOE codes are typically written in C++ or Fortran, with heavy use of numerical frameworks and libraries that use MPI under the covers. However, recent trends with many-core processors and accelerators have caused a major disruption in this model due to their addition of significant new complexities at the intra-node level. These complexities can be addressed at the programming model level through new abstractions that are better aligned with the platform characteristics expected in exascale systems, and at the programming notation level by choosing notations that are more amenable to automated transformation (as in tools such as SPIRAL and FFTW) and that are more suitable for interoperability with C++ and Fortran.

New languages (notably Chapel and X10) can have a direct impact in the post-exascale era but cannot be relied on as the primary vehicle for programming exascale applications, given the planned schedule for achieving exascale capability. However, these languages are already having an indirect impact on existing programming models (MPI and OpenMP) that are trying to evolve to exascale by absorbing key ideas from these new languages, albeit without their notational benefits. Some examples of this indirect impact can be found in recent extensions to OpenMP for task parallelism, accelerators, and thread affinity.

Recently, there is growing interest in the use of *embedded DSLs* to achieve the research and development needs for programming models. Embedded DSLs represent a compromise between expressiveness and pragmatics by using an existing base-language (e.g., Fortran, C, and C++) as the host language for the DSL extensions. Embedded DSLs are a pragmatic way to exploit the sophisticated analysis and transformations capabilities of the compilers for standard languages.

7.2 Compilers

The challenge of automatic parallelization and optimization for today's supercomputers, with distributed memories, multi-core processors, and accelerators, is already keeping compiler researchers and developers very busy. Nevertheless, most application programmers today explicitly manage parallelism, communications, multicore, and accelerators within program notations designed to expose the hardware (e.g., MPI, OpenMP, CUDA, OpenCL, OpenACC). While they are not programming in assembly, they are programming "to the metal" in the sense that the structure of their code is strongly tied to the underlying system hardware and a particular execution model, primarily bulk synchronous.

Such "to the metal" programming exacerbates the compilation challenge because it obscures the underlying semantics of the application, making it impossible for a compiler to safely remap or optimize to new machines or architectures. Such applications are practically "baked" to specific machines, execution models and architectures. Taking applications forward to exascale requires a rewrite.

Thus, the research imperative for compilers for exascale goes beyond expanding the repertoire of optimizations and code generation techniques for new exascale hardware and execution models (discussed below). It includes determining ways of expressing applications in the declarative, goal-driven manner that maximizes the semantic content and minimizes the clutter of machine control.

for Exascale Computing held at the Information Sciences Institute in Marina del Rey. July 2011 [106].

It must maximize the span of potential mappings that a compiler can reach in order to reach the new levels of parallelism, locality, and explicit machine controls needed for exascale performance. Without such reform of programming practice, the value of automatic optimization for exascale architectures will be diminished.

7.3 Main Exascale Challenges for Compilers R&D Focus

The significant challenges posed to the compiler by exascale hardware and system software include:

- Additional concurrency (4-10X), to compensate for NTV-induced slowdown of individual cores.
- Agility to NTV-induced increase variability of throughput from core to core.
- Deep Machine Hierarchy: exascale machines will have on the order of 10 levels of hardware partitioning and memory hierarchy (core, tile, block, unit, socket, module, board, chassis, rack, section, ...), compared today's machines with 3-4. Mapped programs must specify data layouts and execution schedules to conform to this hierarchy.
- Nimble DVFS: Voltage regulation on die will present finer-grained and more numerous opportunities for power savings in exploitation of slack in depowering windows. Mapped applications will need to explicitly manage DVFS through software APIs.
- Resilience, induced by system scale and NTV: Traditional bulk checkpointing will yield to fine-grained local checkpointing. Mapped applications must somehow indicate local checkpointing and restore points.
- Dynamic dataflow-oriented runtimes: The quest for greater concurrency and greater agility (to both NTV induced variability and application imbalances) has revived macro dataflow concepts foundationally for all proposed exascale runtimes. Mapped programs must indicate explicit fine-grained synchronizations (inter-task dependences, continuations, etc.).
- Increased dispersion of "compute" resources through the machine: The relatively low cost of transistors for "compute" (vs. communication and memory) is leading architects to plan to sprinkle compute into non-traditional parts of the machine, specifically, in-network (e.g. implementing versatile programmable units for collective operations) and in-memory (e.g. Custom Memory Cube, and mass storage controllers for flash and HDD).
- Increased pressure to find more locality: Locality reduces communication, so directly attacks the dominant cost of exascale computation. Note that the locality optimization imperative (maximizing the grouping of data and computations) is in contention with the concurrency imperative (which implies spreading things out).
- Explicit management of memory and communications: Explicit communications, RDMA, scratchpad memories, software controlled caching, and collective memory operations will allow more precise control of storage and communication to minimize wasted data storage and motion.

Not only do compilers face challenges from below, from exascale hardware. They also face challenges from above, from the applications.

- New algorithms: In particular, communication avoiding linear algebra, multipole methods, randomized numerical linear algebra, and combinatorial approaches to linear algebra (e.g., support preconditioners) have control and data flows that differ significantly from current algorithms.

- Approximate algorithms: Applications and libraries will increasingly adopt the approach of tuning precision to reduce energy; building on the success of mixed precision libraries. Recent papers suggest vendors will be including variable precision floating point units to help support this. There will also be a growth in approximate algorithms that can trade excess precision to increase parallelism and/or reduce communication. This will challenge compilers to support complex reasoning about numerical algorithms in the context of trading alternative mappings in the optimization process.

Potential approaches to addressing these challenges are discussed below.

7.4 More Knobs for Programmers

These challenges raise the design question of whether to (1) simply design more features in the programming abstractions ("at the metal") to control the additional exascale system features, or (2) to design ways for the compiler to automatically generate controls in the service of the declarative goals.

The first approach, designing new controls within the programming abstraction, would have the benefit of letting the programmers accomplish their programming goals without a compiler in the way, but those programmers would face many more controls to manage, with potentially complex interactions: an overwhelming manual programming challenge. The number of programmers who could perform such optimization would be very limited. Also, such explicitly annotated programs would be even more opaque than usual for automated optimization, and thus even more specialized to specific machine instances. While the approach might yield some hard-won short-term successes (a few applications coded by a few hero-programmers on a few machines), it will lead to much long-term pain (trying to carry such specialized codes forward or broaden the base of programmers).

This is why the second approach, a declarative, goal-driven programming approach, is so attractive. However, how, exactly, to achieve automatic optimization addressing these new exascale challenges is a research challenge.

7.5 Compiler Research Directions

To achieve these exascale mapping objectives requires that the base of conventional automatic mapping technologies is solid. There are many gaps in current automatic optimization technology to fill. These topics alone could fill a long research report. Research in general automatic parallelization, auto-tuning, and machine modeling are all important.

Achieving additional concurrency requires a mix of explicit parallelism and implicit parallelism to be mapped by the compiler. The notation must facilitate the programmer's expression of maximal parallelism. This must be done in a way that supports mapping of the source to future machines, minimizing the binding to any specific machine or execution model. The compiler's job is to extract and maximize the "right" parallelism. Intermediate representations that are "more precise" in representing explicit and implicit parallelism are needed. Advanced code generation techniques to macro-dataflow runtimes then will avoid introducing unnecessary constraints on parallelism (e.g., due to overly conservative reasoning about memory capacity) and overly conservative synchronizations (e.g., barriers). Research should support declarative programming notations that

encourage expression of true dependencies (data and control) in a program, but minimize extraneous dependencies.

A side benefit of a dataflow programming model is that the "write once" logical abstraction is coherent with a resilience approach since the compiler in mapping the applications has control over the introduction of reused state and side effects that are needed for practical implementation. It ought to be possible to combine it with the management of local storage for checkpoints, and the creation of "roll-back" points in the code in the face of faults.

Research into the optimization of explicitly parallel declarative programs should also be supported. Most compiler optimizations and intermediate representations today are designed to represent and support optimization of sequential programs.

Supporting nimble DVFS requires introducing scheduling objectives that increase opportunities for windows of time in which voltage can be decreased or units disabled for brief windows to save leakage power. In essence this becomes an objective of "clumping" load in concentrated groups to free up other resources to be powered down. This clumping is in tension with objectives for concurrency (spreading things out), and so will likely need to be performed by the compiler in a close trade-off among other optimization objectives. With the schedule determined, the code generation section of compilation can emit calls to power control APIs.

With deep (e.g., 10 level) hardware hierarchies, the traditional tiling and blocking of iterations and data to achieve locality will become cumbersome. With one level of hardware hierarchy, a 3-deep nested loop (e.g., matrix multiply) becomes a 6-deep loop nest. With 10 levels, tiled, it becomes a 30-deep nested loop. This presents various code image size and optimization complexity problems. As an alternative to deep nesting, recursive program expressions have the benefit of compactness in the face of hierarchy; they form the hierarchy dynamically at runtime with the recursion. This suggests focusing on compiler techniques that favor generating recursive mappings, with runtime exposition of the mapping.

Increasing locality will demand new approaches to program mapping. The traditional way to increase locality is through loop fusion, which merges producers of data values closely in space and time with their consumers. Ideally, the producer and consumer are so scheduled that the communication occurs at the highest levels of the memory, even through the register file or the bypass network in the datapath. Such fused schedules also have the benefit of reducing storage requirements for intermediate values. Exascale will demand more aggressive fusion across the application. For multi-physics, aggressively fusing distinct parts of the simulation, even when they are written as separate program modules, presents itself as a source of significant potential locality. In-situ UQ, visualization, and analysis, which are logically distinct parts of an exascale computation, can also be closely fused. NTV architectures will have high ratios of register file capacity compared to current architectures, supporting even greater fusion. Programming notations must be designed so as to enable such global analysis and transformation, and compilers must be capable of taking advantage of this locality.

The increased importance of such global analysis and transformations has implications for libraries. Libraries must shift from being opaque, stand-alone objects, toward being clear expressions of their function, effects, and preconditions, that are optimized by the compiler *along with* application codes. This presents opportunities for global optimizations such as aggressive fusion between application and the library. Current practice is for library writers to develop silos of pseudo compiler and pseudo runtime functions (e.g., auto-tuning, task oriented runtimes) within libraries, instead of exploiting compiler versions of these features. New optimization techniques, such as communication-avoiding linear algebra (which are applicable to arbitrary loop nests), should not be implemented as one-offs in a library, but implemented within the compiler so they can be performed broadly across code bases and in concert with other compilation and optimization processes.

7.6 Development Model

The mode of development and collaboration within the compiler community is the subject of much conversation. Collaboration and integration of new techniques will increase the rate of success. However, a single common "national compiler infrastructure" or an exclusive reliance on one model of software exchange (strictly open source, or strictly commercial) will exclude important collaborators and technology. Past experience with an open source national compiler infrastructure, SUIF, is that useful research papers were produced, but not an adopted tool. Integration that brings the best of the different technology sources together through open APIs and standard descriptions (e.g., source to source compilers, standard tuning languages, machine model description languages and runtime APIs) will facilitate the integration of "best of breed" technologies from the many potential contributors. Without question, the strongest compiler infrastructures to build on for exascale capabilities reside in U.S. commercial companies. Funding efforts to duplicate these existing available technologies simply to have something open source is wasteful and should be discouraged. Strong academic support is encouraged to achieve creative research and innovation; commercial software companies can also provide innovation as well as product engineering and long-term support.

7.7 Impact

Exascale systems will be usable by very few scientists without advances in languages, compilers, runtime systems, and libraries, all of which are encompassed in programming systems that serve as the bridge from high-level algorithms to lower-level platforms. Addressing the programming system research challenge is key to ensuring that application developers can achieve both correctness and performance on a wide range of exascale and post-exascale platforms without requiring heroic investments of skills and labor. These advances in programming systems will also be necessary for application enablement on future departmental petascale systems.

8 Data Management

Management of large volumes of data can be categorized into defensive I/O (i.e., the use of data management systems to enable resilient calculation and operation) and offensive I/O (i.e., the use of data management systems for other uses, including data analysis, visualization, and archiving of results). Extracting scientific insight from simulations that run on large HPC supercomputing facilities is of crucial importance. Scientific advances are made only once the data produced by the simulations is processed into an output that is understandable by a scientist.

Data management is a foundational component of computational science today, serving as a management mechanism for experimental and simulation results and providing the primary fault tolerant component in HPC systems. Today's data management systems, which provide the backbone of large-scale computational environments, have been stretched to the scalability limits of their underlying architecture. Based on designs more suitable for today's mid-range HPC environment, leadership class systems across the DOE complex have struggled to achieve scalable performance on these storage and data management systems.

Looking ahead, there is an expectation that system mean time to interrupt (MTTI) will decrease, system scale will dramatically increase, while cost constraints and technology gaps mean that the bandwidth of storage systems will decrease relative to our ability to generate data. These three trends, together with an increased need to analyze the massive data produced by simulation, sensors, and instruments, is outstripping conventional data management and analysis techniques. The lack of credible solutions within the marketplace today cast a great deal of doubt on the success of a "business as usual" approach to future extreme-scale data management systems.

The disruptive changes imposed by a progressive movement towards the exascale in HPC threaten to derail the scientific discovery process. Today's successes in extracting knowledge from large HPC simulation output are not generally applicable to the exascale era, and simply scaling techniques to higher concurrency is insufficient to meet the challenge. [Scientific Discovery at the Exascale, p. 1, http://science.energy.gov/ /media/ascr/pdf/program-documents/docs/Exascale-ASCR-Analysis.pdf.]

8.1 Exascale Data Management Challenges

Some challenges in data management are common to both offensive and defensive uses, while others are specific to the usage mode. These are discussed below, the common ones first, then the those specific to each mode of operation.

8.1.1 Challenges Common to both Offensive and Defensive I/O

Reliability, availability, and correctness of data management system and data transfer are vital areas in need of improvement. Today's data management systems are assumed to be completely reliable as the data management system is the resilience mechanism for large supercomputers. In the future, storage systems will become large enough that we must plan on failure and corruption in these subsystems on a regular basis, so next generation solutions in this are in desperate need. Recent work on the DOE Storage Fast Forward project is daring to explore how HPC storage systems might be allowed to fail [9].

Affordability of data management, both procurement and operational, is a huge challenge. It is clear that over the last twenty years of deploying supercomputing environments that we have moved from low single digit percentages for infrastructure versus the cost of the supercomputer itself. Today this percentage is easily above 20% percent, and growing. Some of this is due to the

power/cooling of HPC environments but the growth in comparative costs for data management systems has also risen dramatically. A recent study indicates that an all disk data management solution will be too costly due to bandwidth growth and an all tape archive will similarly be too costly for the same reason [55] [79].

Scalability of data management solutions in terms of compute, storage, and networking component counts, as well as with respect to science workload needs, is a continuing issue, as scales need to extend into the billion-way concurrence mapping to potentially millions of I/O devices at multiple tiers. The realization that we need to utilize lesser semantics for I/O was called out strongly by the HECFSIO community many years ago [7].

Scientific workflow management of data from memory to archive at next generation scales is not well understood. An example of this is scientific parallel data archives, which employ parallel tape striping today. If we stay on the path we are on in the archive area, we would need to stripe data to hundreds of tape drives for tens of hours to move the data sizes we anticipate. Even with erasure protection across these highly mechanically challenged devices, there is little chance this type of solution will work, nevertheless be affordable [55]. In order to deal with the massive data sets developed and processed by these new systems, most science areas have a need for end-to-end solutions for data management and analysis.

8.1.2 Challenges Specific to Defensive I/O

Enrichment of application interfaces targeted at defensive I/O workloads is required. The ability to deal with the eventuality that the data management system must be allowed to be in failure mode frequently and the ability to map to billion-way task parallelism are two of many drivers for the requirement for new data management interfaces.

Integration with other system management and resilience functionality is also necessary. There is some hope that other resilience mechanisms, besides utilizing the data management systems for checkpoint style resilience, may be achievable and it is necessary to coordinate the data management system with these potential resilience mechanisms. Further, data movement will be expensive in power and time and may need to be scheduled and avoided at all costs. It is important that data management systems be integrated well into the exascale systems and applications to take advantage of any efficiency possible. We must be careful to apply data locality where it makes sense. This, however, does not mean that all functions must maximize locality, but truly expensive data movement should be managed.

8.1.3 Challenges Specific to Offensive I/O

Data organizations and functions that cater to efficient output and also data analysis, visualization, archiving, indexing, search, reduction, transformation, provenance, and representation are necessary. It may not be possible to do analysis on data once it gets all the way to disk-based mass storage due to the times involved in dealing with multi-petabyte objects [36]. It will become vital that data are organized efficiently so that analysis and other downstream functions are even possible given the volumes of data anticipated.

Improved support for application data models to facilitate the above organizations and functions is necessary to allow applications to be able to take advantage of and more simply deal with the data management tasks.

The entire workflow for data analysis and visualization must change in the context of the high power cost of data movement and the worsening I/O bottleneck for exascale systems. Simply scaling existing techniques to higher concurrency is insufficient to meet the challenge. To extract knowledge

from exponentially increasing multi-scale, multi-physics data, focused research is required into effective *in situ* processing frameworks, new I/O middleware systems, fine-grained visualization and analysis algorithms to exploit future architectures, and co-scheduling analysis and simulation on HPC platforms. Approaches to data triage and/or compression must also be developed that preserve the capability for exploratory analysis to discover the unexpected.

8.2 Research Directions

In getting to terascale and on to petascale computing, the area of data management has had to work hard to keep up under the challenges of lagging storage device bandwidths and reliability compared to the meteoric growth in compute capability. This area of data management for HPC has produced many commercial products through government-funded and/or led initiatives such as most of the middleware for I/O, including MPI-IO and more recently Parallel Log Structured File Systems (PLFS) [115] [12]. In the parallel file system area, products like Panasas, IBM's GPFS, and Lustre [86] [110] [112] were all supported or shaped in some way by government future investment initiatives along with careful management to produce usable products. More recently the introduction of the "Burst Buffer" concept to utilize in-system solid state storage like Flash is all the rage, which is making its way into commercial products as well [79] [54].

It is clear that commercial products, with the help of government funding and guidance, have made current scale processing viable and will play a huge role in the future. Of course, other non-commercial efforts to fill in the gaps and research and development to lead the way for the HPC data management community, are also crucial ingredients for managing this important area to success. Looking towards exascale, five strategies are contemplated for managing the data management area of HPC.

First, develop, hone, and disseminate an understanding of leadership computing data management architectures, workloads, and issues both – current and future. Community understanding of current architectures and workloads, as well as trends moving forward, is essential to effectively analyzing gaps in research and development as well as the product space. This understanding can be refined through measurement of current systems, modeling and simulation, and the use of testbeds by researchers exploring new software capabilities. Needs in this area include support for large supercomputing centers to collect, model, and disseminate information about current and future-facing workloads, and deployment and management of large future-looking testbeds for the data management research and development community. A strong community that understands the problems we face is vital and must be nurtured [7].

Second, track ongoing research and development activities and identify areas where gaps create risk in terms of future exascale deployments. The HECFSIO activity provided a useful view of ongoing file systems and I/O research and development across multiple agencies, industry, and academia that aided in identifying gaps. A similar activity would provide needed information on current research and development activities of relevance to exascale data management.

Needs in this area include a rekindling of the HECFSIO concept, a multi-agency approach to research and development portfolio management, funded activities in the gap areas via carefully crafted and focused solicitations, and active management of the projects [7]. Of course, the gaps lie in the challenges mentioned above in the areas of reliability, affordability, scale, workflow management, interfaces, integration, data organization, and application-centric data models. The gaps span the I/O and data storage software stack from in-system solid state storage, through disk-based parallel file systems, all the way to parallel archives. Fundamental changes to the design concepts that storage systems are always available/reliable and minimizing data movement where necessary need to be made to the data management subsystems to enable exascale-class computing.

It is quite possible that the scientific HPC community can leverage technology used in the the big data/analytics and cloud storage worlds in providing some portions of the overall data management area, but the problems are not identical. For example, single scientific objects to be stored might be petabytes in size in scientific HPC, but the typical size of a cloud storage object might be a few megabytes. Another example is how tightly interconnected data analysis might be in scientific workloads where the bulk of commercial data analytics today at least is quite embarrassingly parallel. Investigation into how leverage of these industry trends might be accomplished is also needed.

Third, facilitate commercialization of successful prototypes that provide necessary capabilities for exascale data management systems. Commercialization provides a vehicle for long-term support of software needed for exascale and beyond. Activities could include, for example, the commercialization of prototypes developed under the DOE Storage Fast Forward activity, or the integration of successful prototype software into commercial products under SBIR funding.

Needs in this area primarily include funding along the lines of the DOE Path Forward [35] style investments to continue the successes of HPC pushing industry in the data management arena. Different challenges exist economically in todays world of technology funding/management though. For example the "Big Data Analytics" industry is creating technologies that might be leveraged to provide some portions of the data management solution for Exascale class computing with the proper investments to make some of those ideas useful to the HPC community. Further, it is quite likely that investments by the HPC community will help the U.S. "Big Data Analytics" industry as well. For reasons like this, it is important to engage top subject matter experts to carefully manage investment portfolios for leverage and effect. It is vital that interim supercomputer procurements on the road to exascale require the commercialized data management solutions to be provided and supported to further solidify these solutions in the market place.

Fourth, support data management technologies that have not been commercialized but provide needed solutions to exascale data management challenges. Not all critical technologies are on a path to commercialization at this time: they may fill a niche particular to exascale systems, or they may be disruptive technologies that do not fit well into the business models of current vendors. While they remain on the critical path, these technologies must be supported until they are adopted by vendors or commercial alternatives emerge. The need in this area is essentially some level of funding commitment for niche-filling technologies that are required because of lack of commercial appeal in the near term.

Fifth, communication/coordination with the data management community and with other HPC related communities is needed. As has been mentioned before, the HECFSIO mechanism for community involvement, gap determination, research and development road-mapping, and research and development/commercialization portfolio management that includes multiple government agencies, universities, and industry was an excellent model for vertical management within the high performance data management area [7]. A similar multi-agency effort needs to be spun up and maintained with outcomes of better community involvement, excellent coordination, and portfolio management assistance in the form of annual documentation. Additionally the data management community needs to continue to be engaged in co-design activities with other areas of the exascale pursuit such as OS, runtime/execution, architecture, applications, programming models, etc. via co-design exchange and activities.

The needs in this area are some continuous funding opportunities for the universities in this area and a respin of the HECFSIO concept, a multi-agency portfolio management/community communications body.

8.3 Impact

The HPC data management solutions deployed today were designed well over a decade ago. The target for those solutions was a decade, which is approximately now. It is clear that those solutions are running out of gas by looking at the evolution of I/O libraries to fill gaps in function such as locking, name space scaling, correctness, and data movement minimization. In a decade, when exascale systems present billion-way parallelism, severe power limitations, and mean time to interrupts in the few hours range, the current solutions with I/O libraries filling gaps will have long since run out of gas from the scalability, resilience, and power efficiency points of view. The impact of the work laid out above will simply make possible computing at the exascale from the data point of view. Without this important data management work, it is unclear that applications will be able to run to completion to get a correct answer, systems can be fielded within power constraints, or systems can even be afforded.

9 Exascale Algorithms

Advancing science requires development of novel physical models to satisfy the accuracy and fidelity needs for targeted simulations. The impact of the need to increase simulation fidelity on computational science is two-fold. First, more complex physical models must be developed to account for additional aspects of the physical phenomena being modeled. Second, for the physical models being used, increases in resolution for key system variables, such as numbers of spatial zones, time steps or chemical species, are needed to improve simulation accuracy, which in turn places higher demands on computational hardware and software.

Scientific models create the functional requirements that drive the need for numerical algorithms and software implementations. The choice of model is in part motivated by the science objectives, but it is also constrained by the computer hardware characteristics attainable in the relevant time frame. The choice and specification of system attributes (e.g., peak speed or node memory capacity) tend to constrain the functional attributes able to be employed in a given physical model on that system. Models and associated algorithms are not selected in isolation but must be evaluated in the context of the computer hardware environment. Furthermore, algorithms that perform well on one type of computer hardware may become less effective on newer hardware, so selections must be made carefully, and may change over time. As will be described below, it will be very challenging to devise algorithms and software that can effectively exploit the exascale systems that are being envisioned.

It is widely recognized that, to date, improved numerical algorithms have contributed as much to increases in computational simulation capability as have improvements in hardware. These algorithms are often provided to scientists in the form of numerical libraries, which isolate the users from the details and allow optimization by system vendors. Future developments in computer systems will throw even greater focus on algorithms as a means of increasing computational capability. Enhancing the national capabilities in advanced computing algorithms and software will have a major impact on the U.S.'s future scientific research capacity in the ever-increasing number of domains in which high performance computing is, or is set to become, a core activity. In addition to the domains within DOE's areas of research, it is absolutely essential that the DOE make strategic investments now in high performance computing algorithms and software to maintain the U.S.'s international leadership in scientific computing.

Significant new model development, algorithm re-design, and science application code re-implementation, supported by exascale-appropriate programming models, will be required to exploit the power of exascale architectures effectively. New algorithms will need to be designed to optimize not only for floating-point performance and accuracy, but also to minimize associated data movement, power, and energy costs. The transition from current sub-petascale and petascale computing to exascale computing is expected to be at least as disruptive as the transition from vector to parallel computing in the 1990s. An intensive co-design effort will be essential for success, where system architects, application software designers, applied mathematicians, and computer scientists work closely together to produce a computational science discovery environment that fully leverages the significant advances in computational capability that will be available at the exascale.

Moving to exascale will put heavier demands on algorithms in at least two areas: (1) the need for increasing amounts of data locality in order to perform computations efficiently; and (2) the need to obtain much higher factors of fine-grained parallelism as high-end systems support increasing numbers of compute threads. As a consequence, parallel algorithms must adapt to this environment, and new algorithms and implementations must be developed to approach the computational capabilities of the new hardware. Below is a discussion of several key research areas in numerical algorithm development for exascale computing.

9.1 Multicore-Friendly and Multicore-Aware Algorithms

Scalable multicore systems bring new computation/communication ratios. Within a node, data transfers between cores are relatively inexpensive, but temporal affinity is still important for effective cache use. Across nodes, the relative cost of data transfer is growing very large. New algorithms must be developed that take these issues into account, such as communication-avoiding algorithms that increase the computation/communication ratio, algorithms that support simultaneous computation/communication, and algorithms that vectorize well and have a large volume of functional parallelism.

9.2 Communication Avoiding Algorithms

Algorithmic complexity is usually expressed in terms of the number of operations performed rather than the quantity of data movement to memory. This is antithetical to the expected costs of computation at the exascale, where memory movement will be very expensive and operations will be nearly free. When solving very large problems on parallel architectures, the most significant concern becomes the cost per iteration of the method, typically because of communication and synchronization overheads. This is especially the case for preconditioned Krylov methods, which are the most popular class of iterative methods for large sparse systems.

To address the critical issue of communication costs, there is a need to investigate algorithms that minimize communication. New lower bounds on bandwidth and latency must be derived for various numerical algorithms on parallel and sequential machines, e.g., for dense and sparse linear algebra algorithms where the well-known lower bounds for the usual $O(n^3)$ matrix multiplication algorithm should be extended. In many cases, new algorithms that attain these lower bounds must be invented. Another example of needed research is in Krylov subspace methods like GMRES, CG, and Lanczos, where one should devise means to take k steps of these methods with the same communication costs as a single step.

Scientists also need to consider a broader range of numerical algorithms. An example is the Fast Multipole Method (FMM) [24, 52, 53, 104], which trades precision to achieve significant reductions in both computation (asymptotically, from $O(n^2)$ to $O(n)$) and communication. This algorithm has been named as one of the most important algorithms of the 20th [26] and 21st centuries [94], yet strangely has seen little adoption (perhaps due to its initial perceived complexity?). However, a growing community of algorithms experts (e.g., [8]) are finding that the future success of the exascale era may go hand-in-hand with the ability for researchers to develop new fast direct solvers and adaptations to the FMM. The route to this may involve using kernel-independent approaches [75, 76, 103, 119, 124, 125] that remove the need for FMM solvers to be hand crafted for particular applications. The breadth of application of FMM is not limited to such solvers. One important application will be its ability to reduce communication of large FFTs by close to three-fold [38, 74].

9.3 Synchronization Reduction

It is often necessary in an algorithm to synchronize the computation. A good example is the parallel computation of dot products. Synchronization is needed after such global reductions. However, synchronizations can become bottlenecks. Thus, it is necessary to design algorithms that synchronize as little as possible. Attempts have been made to restructure existing algorithms so that the number of synchronizations is reduced. An example is the conjugate gradient algorithm. By using some of the mathematical identities, it is possible to come up with alternative versions of the conjugate gradient algorithm that have just one synchronization rather than two in the conventional description of the algorithm. Parallel-in-time algorithms are a more extreme example, in which

additional concurrency is obtained by decomposing the solution in time and using an iterative, multi-level approach to enforce causality between the independently advanced time segments.

Restructuring algorithms to reduce the number of synchronizations, and in general the amount of communication, will become more important in the exascale era. However, it should be noted that it is not necessarily the case that all the variants of an algorithm have the same numerical behavior. In the case of conjugate gradient, it is known that some variants may not be numerically stable. Thus, in restructuring an algorithm to reduce synchronization and communication, it is important to investigate the stability of the variants.

9.4 Multi-Physics Algorithms

Exascale resources will provide the opportunity to increase the physical fidelity of scientific simulations, which presents several issues and opportunities that require more research. Many applications of interest to DOE involve coupled physical processes, e.g., nuclear reactors and magnetically confined reacting fusion plasmas. Coupling can either be volumetric or interfacial, and various algorithms exist to solve the resulting model, either as a fully-coupled or operator-split system. The former case results in the solution of a large, nonlinear, algebraic system, and the development of efficient preconditioners for such a system is critical. One approach may be to use split-physics solvers as a form of pre-conditioning, but there may be other advantages to split physics approaches, especially if the physical modules provide a level of coarse-grain concurrency. However, the mathematical properties of most operator-split codes in practice are not well understood. Research is needed to define the strength of coupling and how it should affect algorithm choices, into how synchronization could be weakened in split approaches, and into provably stable, high-order accurate splitting schemes. Finally, exascale resources may encourage attempts to use models that were previously considered infeasible, and this may require significant new algorithm development.

9.5 Multi-Scale Algorithms

As with multi-physics algorithms, the potential increase in available resources at the exascale will be used to address ever more challenging multi-scale problems. Multi-scale algorithms include adaptive mesh refinement, asymptotic-preserving discretizations, and simulation with models at different scales coupled through scaling-bridging techniques such as moment models, coarse-grained molecular dynamics, projection-based methods, homogenization, renormalization group methods, and the Mori-Zwanzig formalism. The treatment of multiple scales can be global (multiple-scale models are defined over the whole domain) or local (static or adaptive model refinement, often coupled with adaptive mesh refinement in the latter case). Multi-scale algorithms should provide additional opportunities for concurrency; often, the model results in a hierarchy of descriptions that should map effectively onto the hierarchical structure of exascale architectures. Generally, coarse models are also a reduction in the number of degrees of freedom of the finer models, and solutions at the coarse scale can be used effectively to accelerate the solution at the fine scales, perhaps in an asynchronous, iterative way. Adaptive multiscale algorithms are an important part of the DOE portfolio because they apply computational power precisely where it is needed. However, these algorithms also introduce challenging computational requirements because they introduce dynamically changing computation that results in load imbalances. Finally, reduction processes like the Mori-Zwanzig formalism produce, in general, stochastic PDEs, for which theoretical results and numerical solution techniques are underdeveloped. Exascale computing will open up opportunities for additional research in multi-scale analysis and the development of multi-scale models and algorithms suitable for exascale machines.

9.6 Adaptive Response to Load Imbalance

As we move towards systems with billions of threads, even naturally load-balanced algorithms on homogeneous hardware will present many of the same load balancing problems that are observed in current adaptive codes. For example, software-based recovery mechanisms for fault-tolerance or energy-management features will create substantial load-imbalances as tasks are delayed by rollback to a previous state or correction of detected errors. Dynamic scheduling based on directed acyclic graphs (DAGs) has been considered as a path forward, but this approach will require new approaches to optimize for resource utilization without compromising spatial locality.

9.7 Scheduling and Memory Management for Heterogeneity and Scale

Extracting the desired performance from environments that offer massive parallelism, especially where additional constraints (e.g., limits on memory bandwidth and energy) are in play, requires more sophisticated scheduling and memory management techniques that have heretofore been applied to numerical algorithms. Confronting the limits of domain decomposition in the face of massive, explicit parallelism introduces another form of heterogeneity. Feed-forward pipeline parallelism can be used to extract additional parallelism without forcing additional domain decomposition, but exposes the user to dataflow hazards. Ideas relating to a data-flow-like model, where parallelism is expressed explicitly in DAGs, allows for the scheduling of tasks dynamically, support of massive parallelism, and application of common optimization techniques to increase throughput. Approaches to isolating side effects include explicit approaches that annotate the input arguments to explicitly identify their scope of reference and implicit methods, such as using language semantics or strongly typed elements to render code easier to analyze for side effects by compiler technology. New primitives for memory management techniques are needed that enable diverse memory management systems to be managed efficiently and in coordination with the execution schedule.

9.8 Energy-Efficient Algorithms

It is widely recognized that emerging constraints on energy consumption will have pervasive effects on HPC. Energy consumption must now be added to the traditional goals of algorithm design, i.e., correctness and performance. The emerging metric of merit is performance per watt. Consequently, it is essential to build power and energy awareness, control and efficiency into the foundations of exascale algorithms and the numerical libraries in which they are often realized. First and foremost, this will require that standardized interfaces and APIs be developed for collecting energy consumption data, just as PAPI [20] has done for hardware performance counter data. Accurate and fine-grained measurement of power consumption underpins all tools that seek to improve such metrics. Anything that cannot be measured cannot be improved. Secondly, these tools must be used to better understand the effects that energy-saving hardware features have on the performance of numerical algorithms. Finally, parameters and alternative execution strategies must be identified for each numerical algorithm that can be tuned for energy-efficient executions, and to enhance schedulers for better low-energy execution.

9.9 Automatic Adaptation of Algorithms

Numerical algorithms and libraries need to have the ability to adapt to the possibly heterogeneous environment in which they operate. Such adaptation must deal with the complexity of discovering and applying the best algorithm for diverse and rapidly evolving architectures. An automated process would be best, both for the sake of productivity and for correctness. Here, productivity

refers both to the development time of the implementation and to the user's time to solution. The objective is to provide a consistent library interface that, independent of scale and processor heterogeneity, achieves good performance and efficiency by binding to different underlying code, depending on the configuration. The diversity and rapid evolution of today's platforms mean that such automatic adaptation of libraries, such as BLAS, will be indispensable to achieving good performance, energy efficiency, load balancing, etc., across the range of systems. In addition, such adaptation has to be extended to frameworks that go beyond libraries, such as optimizing data layout (e.g., blocking strategies for sparse matrix/SpMV kernels), stencil auto-tuners (since stencils kernels are diverse and not amenable to library calls), and even tuning of the optimization strategy for multigrid solvers (optimizing the transition between the multigrid coarsening cycle and bottom-solver to minimize runtime). Adding heuristic search techniques and combining these with traditional compiler techniques will enhance the ability to address generic problems extending beyond linear algebra.

9.10 Required Action

ASCR has started to direct more attention to the development of numerical algorithms for exascale computing. For several years, most funding opportunities from the ASCR Applied Math program, including the Early Career Research Program and the recent "Exploratory Research for Extreme-Scale Science" call, have required the proposed research to address future architectural challenges. More explicitly, the projects funded under the "Resilient Extreme-Scale Solvers" initiative, the three Exascale Co-Design Centers, and the FASTMath SciDAC Institute, are all directly addressing some aspects of the algorithmic challenges. The Exascale Mathematics Working Group, chartered by ASCR, will soon produce a detailed report on the needed research in applied mathematics for exascale computing, and this should be acted on by DOE with funding above and beyond the applied math base program[3].

Beyond building a more coherent applied mathematics research program for exascale computing, DOE must facilitate communication and co-ordination between numerical algorithms researchers and other activities within the exascale computing enterprise. The move to extreme-scale computing will require tools for understanding complex behavior and for performance optimization to be based on a knowledge-oriented process. Performance models and expectations will required for algorithm investigation and reasoning. The level at which tools interoperate and can be integrated with the application development and execution environment must be raised. The challenges for performance analysis and tuning will grow as performance interactions and factor analysis will involve a whole-system perspective. The co-design methodology is iterative, requiring frequent interactions among hardware architects, systems software experts, designers of programming models, and implementers of the science applications that provide the rationale for building extreme-scale systems.

9.11 Impact

Numerical algorithms are the core of all physical simulation codes. To fulfill the promise of extreme-scale science enabled by exascale computers, existing numerical algorithms must be redesigned and new numerical algorithms must be developed so that a significant percentage of the potential performance of these machines can be reached. An exascale effort without investment in advanced numerical methods will produce machines that will fail to meet the science goals of the DOE.

[3]Exascale computing is just one area of applied mathematics research needed by DOE, so exascale algorithms research must not consume a disproportionate amount of the available resources.

10 Algorithms for Discovery, Design, and Decision

Future supercomputer architectures will provide unprecedented computing power, but it is important to understand now how such capability will be used to the advantage of extreme-scale science. What is clear is that one-off, "heroic" forward simulations on the scale of a full machine contribute little, in isolation, to scientific knowledge. It is the aggregate of ensembles of simulations that ultimately informs scientific theory and engineering design. Exascale computing holds the promise of providing the resources necessary to determine not only solutions, but their sensitivity to inputs, their uncertainties, and their optimality under a given set of design or decision constraints.

To make predictive simulation for design and decision a reality in the exascale computing context, however, work must immediately begin to develop scalable, resilient methodologies for uncertainty quantification (UQ) and mathematical optimization. The coming exascale era provides not only an opportunity to build optimization and UQ capabilities into simulation codes, but these additional capabilities, which ultimately reflect the true use cases for scientific simulation, may in fact be key to achieving high utilization on exascale computers. In particular, techniques more tightly coupled to the forward solution strategy could provide opportunities for re-use, communication-hiding, and even vectorization across multiple solutions. To impact future codes, however, research must be done now to develop, explore, and understand myriad algorithmic design tradeoffs.

10.1 Research Directions

Uncertainty quantification and mathematical optimization are extensive fields that have existed for some time, but their application to multi-scale and multi-physics scientific simulation is a more recent and evolving development. Uncertainties exist in all models, either due to intrinsic variability or lack of knowledge, and sources of uncertainty include inaccurate physical measurements, bias in mathematical descriptions, and numerical approximation errors in simulations. UQ is a broad term for a collection of activities that includes uncertainty characterization, *forward propagation*, and the inverse problem of *parameter estimation/model calibration*. As such, it is critical to organize and design simulations and physical experiments that ensure that the right types and amount of data are available not only to quantify uncertainties, but ultimately to reduce their effect on quantities of interest (QoIs) – the desired outputs of simulations.

Mathematical optimization, on the other hand, involves finding the best value(s) of an objective function, subject to constraint functions characterizing the feasible design/decision space. When physical or simulated phenomena are involved, these constraints necessarily include the space of realizable solutions from a simulation code. In one sense, both UQ and optimization can be seen as *outer loops* around the traditional forward simulation, and so both represent a substantial increase in needed computational resources. That need is amplified at the intersection of these two activities in areas such as optimization under uncertainty, robust optimization, or optimization-based model calibration. Exascale computing will begin to provide the necessary resources to facilitate UQ and optimization for complex multi-physics codes, but there is much research to do to achieve this goal.

10.1.1 Uncertainty Quantification

It is widely expected that rigorous uncertainty quantification over high-dimensional input spaces will play a crucial role in enabling extreme-scale science [34,90]. Indeed, a thousandfold increase in computing power would facilitate orders-of-magnitude more simulation realizations. A fundamental challenge in UQ is the *curse of dimensionality*, which is caused by the explosion in computational effort required to approximate the high-dimensional response of QoIs to input uncertainties. In the

last decade, this problem has gained considerable attention, and many *intrusive* and *non-intrusive* methods have been developed to alleviate it [4, 5, 31, 45, 48, 57, 82, 83, 89, 91, 92, 122, 123].

Considering the forward problem, a standard *non-intrusive* UQ approach is to sample from the probability distributions of input parameters, execute a simulation for each sample, and either to compute statistics of the ensemble (e.g., classic Monte Carlo), or to construct a interpolated response surface surrogate (e.g., stochastic collocation). On the other hand, similar to a stochastic collocation approach, an *intrusive method* also constructs a stochastic polynomial approximation, typically orthogonal polynomials with respect to the probability density function, and uses a Galerkin projection that couples both the deterministic and stochastic degrees of freedom. Both approaches have been effective at producing software and algorithms relying on modest numbers of simulations that scale well on existing petascale architectures. However, future exascale computers will provide enough concurrency for a thousand-fold increase in petascale sample evaluations for uncertainty propagation applied in this manner. The anticipated increase in node-level parallelism will favor increasing concurrent sample evaluations either by executing each sample on a smaller number of compute nodes or by executing multiple samples simultaneously on each compute node.

To leverage the increase in node performance, samples could be evaluated in parallel through multi-level embedded propagation schemes, whereby collections of samples are executed asynchronously and samples within each collection are propagated simultaneously at the node and processor core levels. Embedding portions of the "uncertainty loop" at the lowest levels of the simulation code requires replacing each scalar datum in a calculation with an array for the uncertainty representation of that datum, such as samples in a stochastic collocation-type method or polynomial coefficients in a stochastic Galerkin-type method. Operations on that datum can then be translated to parallel operations on the uncertainty array, which improves locality and exposes additional fine-grained parallelism. Since the messages for multiple realizations wil be incorporated into a single message for the ensemble, total communication time will be reduced. Finally, this approach will enable the design of new algorithms that reuse data and calculations across uncertainty representations to reduce aggregate simulation cost, (e.g., reuse of mesh calculations that are independent of uncertain input data) or of solvers and preconditioners across an ensemble [64, 65].

While such approaches could enable more concurrency in aggregate uncertainty propagation, there are considerable challenges that must be overcome to apply these ideas to practical scientific problems. First, redesigning the inner loops of simulation codes will require both considerable programming effort and knowledge of both simulation technologies and UQ methodologies. Code transformation techniques based on automatic differentiation could alleviate much of this difficulty [49, 97, 99]. Second, the concept of propagating multiple samples simultaneously at the scalar level of the simulation is predicated on the assumption that the code paths for these samples do not diverge greatly; otherwise, no benefit, or possibly even harm is achieved. However, many scientific simulation codes exhibit some form of non-smooth behavior that drives the simulation into different regimes depending on the values of the uncertain inputs. Careful research is needed to connect these behaviors to high-level adaptive uncertainty propagation methods that decide when and how to group samples for co-propagation. Finally, this form of uncertainty propagation generates linear and nonlinear problems with a special Kronecker-product structure. Little research has been undertaken to exploit this structure in solution and preconditioning algorithms. Effective parallel sparse partitioning, reordering, and balancing algorithms for the higher-order tensors generated by these approaches are also required.

Finally, in any UQ approach for simulations, the dominant cost lies in the solution of the underlying deterministic model. High-fidelity, multi-scale, multi-physics models can exhaust the resources on the largest machines with a single instantiation and, as such, are not practical for even the most advanced UQ techniques. Future increases in computational resources will be accompanied by con-

tinuing demands for increased resolution and physical fidelity. Consequently, new approaches are needed to decrease simulation costs in the UQ context. To exploit hierarchal exascale architectures, multi-level hierarchical sampling, interpolation, and polynomial approximations will enable the use of existing preconditioners [39–41, 47, 50, 96, 98, 101] at each level to inform solvers at the next level of the hierarchy and the efficient generation of new preconditioners to accelerate the convergence of additional sampling/collocation points or polynomial bases. Moreover, in contrast to principal orthogonal decomposition (POD)-type reduced order models [21–23, 56, 121], moment methods and variational techniques [77, 81, 109] could be used to develop multilevel formulations of the deterministic problem that would elucidate common solution structures across multiple UQ levels, including structures induced by multi-scale dynamics and scale separation. However, to minimize the total cost at a prescribed error level, a general strategy must be developed to balance the contributions from approximation error in the stochastic space with model error in the deterministic space. Such a strategy will require analysis of both the deterministic model errors and stochastic polynomial or sampling errors.

This is not an exhaustive list of the research challenges facing UQ at the extreme scale. However, because of potential effects on simulation code structure and composition, these examples demonstrate the immediate need to address the exascale challenges of UQ.

10.1.2 Optimization

The dramatic increase in computing capability at the exascale is expected to increase the role that mathematical optimization plays across the science and engineering landscape [34]. However, substantial barriers exist that prevent simple reuse of current optimization algorithms for application at the exascale. In traditional mathematical optimization, the optimization algorithm is represented by an outer loop around the forward evaluation of objective/constraint functions and their respective derivative/adjoint functions. These loops tend to be inherently sequential: the functions are evaluated at a single point in the design/decision space for each iteration. In such *single-point* methods, research has focused on parallelizing the underlying linear algebraic operations within an optimization step and/or parallelizing the forward evaluations [15, 17, 28, 85, 102]; for many problems, exascale computing will bring to an end to savings based on such parallelism. A fundamental challenge in optimization for the exascale is thus the reduction of the number of outer loop iterations, but exascale computing will also enable the solution of broader classes of optimization problems, such as optimization with integer decision variables [10, 46], robust optimization and optimization under uncertainty [13, 16], and global optimization [61, 88]. The structures of such problems allow for algorithms that are less sequential, which will provide additional opportunities for parallelism.

There are several possible approaches to reduce the number of optimization iterations. For instance, higher-order information, such as higher-order derivatives of the simulation output with respect to the design/decision parameters, could be exploited; current matrix-free algorithms tend to focus solely on efficient Jacobian- and Hessian-vector products. Research is needed on scalable methods to compute and/or apply higher-order information to ensure that higher arithmetic-intensity rates result. Efficient exploitation of sparsity structures, or more general partially separable structures [27], will become increasingly critical as higher-order derivatives are considered.

A less traditional approach will be through the development of *multi-point* or *concurrent-point* methods, which has been a long-standing challenge in optimization. Such methods determine multiple, distinct (but possibly related) design/decision points for concurrent evaluation either done asynchronously or at differing levels of fidelity. Current research has focused on derivative-free methods [62] or decomposition-based approaches [18, 44], but more general, derivative-based

concurrent-point methods need to be investigated as well; the *s*-step Krylov and related methods [25, 30], which solve a particular form of unconstrained optimization problem, also provide direction for further research. Classes of problems where concurrent-point methods can produce substantial savings will need to be established. Furthermore, as with UQ methods, the selection process of multiple points will need to account for associated performance implications, perhaps by grouping related forward simulations in order to exploit reuse and minimize data movement, in addition to traditional optimization considerations.

As with hierarchical UQ methods, research is also necessary on optimization algorithms that exploit multiple levels of fidelity. Such methods would embrace a strategy of avoiding expensive evaluations, due perhaps to high-fidelity or resilience considerations, when sampling far from the optimal solution. Such optimization algorithms will need to select point(s) for evaluation along with corresponding fidelity levels, possibly informed by architecture-based and/or performance-model-based knowledge of the expense associated with performing that set of evaluations. Applying multi-level optimization techniques to a complex algorithmic hierarchy is an open challenge, but preliminary work on simpler problems exists, e.g., when an underlying grid structure is present [87, 116]).

Finally, uncertainty can have deleterious effects on "optimal" solutions that fail to account for such uncertainty. Stochastic optimization [16] and robust optimization [11, 13] are two modeling paradigms that account for uncertainty in the optimization procedure; they differ in the way the uncertainty is incorporated in the design/decision process. Methods for stochastic optimization have natural tie-ins with scalable, ensemble-based UQ approaches. For example, evaluating many scenarios concurrently can mitigate variance or uncertainty [29, 70, 80], but overall savings may only result if such fidelity is truly useful and if the scenarios admit scalable, resource-constrained evaluation. Robust optimization, on the other hand, leads to more complex optimization problems with conic constraints that have higher linear algebra complexity than standard optimization problems. Provable convergence and error bounds for such methods are needed, as will general strategies with looser dependence on and/or adaptive refinement of the number of scenarios needed to enable scalable decision making/design under uncertainty.

As before, the possible research directions for mathematical optimization for exascale computing discussed here are meant to be illustrative and not exhaustive. The point is to demonstrate that mathematical optimization at the exascale will need to be enabled not only by advances in optimization techniques, but also through additional functionality in the *inner loops* – solvers, adaptive meshing, UQ, *etc.* – such as the ability to be run at different levels of fidelity, to provide information about their resource usage/contention for ensembles of related inputs, to propagate errors/resilience-handling, and to obtain savings from reuse and concurrent evaluations.

10.2 Impact

Advances in algorithms for design and decision under uncertainty will directly impact the science to be performed on exascale systems. The quantification of uncertainty in simulation results is just the extension to computational science of the best practices of the experimental scientific community. UQ for complex multi-scale, multi-physics models is a requirement for truly predictive simulation capability. It will provide confidence in the extreme-scale science conducted on the future exascale computers and aid in the validation and development of scientific models. Predictive simulation capability enables decision making and the use of complex simulations in design optimization.

11 Resilience and Correctness

Computing an incorrect answer quickly is of no use to a scientist. Yet computing with exascale hardware poses several challenges in assessing the correctness of numerical simulation results. First, both hard and soft faults are expected to occur with much greater frequency than on previous hardware. Uncorrected soft faults have the potential to corrupt computed solutions. Hard faults will need to be handled on the fly; halting and restarting an entire application due to the loss of a node, for instance, will be prohibitively expensive at the exascale. Dynamically recovering from either type of fault will introduce non-deterministic variability in resource usage, as will dynamic scheduling of tasks. Due to the non-associativity of floating-point arithmetic, such non-determinism will make bit-wise reproducibility difficult at best and will complicate code correctness testing procedures, including code verification.

Nevertheless, DOE's Office of Science and NNSA have several critical mission deliverables, including annual stockpile certification and safety assurance for NNSA and future energy generation technologies for Office of Science. Computer simulations are key to meeting these deliverables and must be resilient enough to complete in time and correctly in order to meet the respective critical mission need. Such mission-critical codes must undergo code verification tests to provide confidence in the computed results, even before considering additional concerns such as validation and uncertainty quantification. In many cases, these mission-critical simulations can take days, weeks, or even months to compute, which increases the computation's exposure to faults. Many of these applications possess complex, nonlinear, multi-physics models, so silent faults, which do not cause obvious failure, have the potential to alter results significantly. In some cases, such as in climate and nuclear reactor design code, great effort has been expended to ensure bit-reproducibility as a measure of code correctness, either by the scientific community or by regulatory agencies.

Therefore, resilience to faults has been identified as a critical need for future HPC systems [71]. The thousand-fold increase in computational capabilities expected over the next decade, along with incorporation of techniques for reducing energy consumption, are predicted to increase the error rate of the largest systems to a point where present checkpoint/restart methods will no longer be viable. Without research into new fault management techniques and the development of supporting resilience technologies in the hardware and system software, DOE's mission-critical applications may not be able to run to completion, or worse, will complete but get incorrect results due to undetected errors. Without suitable verification techniques, it will be difficult to put faith in computed results.

It will not be possible to eliminate all faults from exascale machines, and non-deterministic execution is likewise unavoidable without potentially severe performance penalties. Fault management will require developments in hardware, programming environments, runtime systems, and programming models. The issue of code correctness is ultimately a mathematical one in this context and will require mathematics-informed solutions. Research will be required to devise efficient application-level fault-tolerance mechanisms and new procedures to verify code correctness at scale.

Developing scientific applications that are fault tolerant on exascale computers will require methods that go beyond traditional checkpointing. All aspects of exascale computing, from hardware up to the application, can and must contribute to create viable resilience strategies. In addition, research into new techniques and tools is required to develop confidence in results computed at the exascale.

11.1 Hardware Support and Performance Modeling

Industry trends indicate that it will become increasingly difficult to be confident in calculations. The reasons for these trends are varied but effects from terrestrial neutrons, naturally occurring alpha particles, electro-magnetic interference, temperature, and voltage fluctuations are seen today and expected in greater levels in the future. Usually these faults are transient in nature but permanent faults from these sources are not unheard of. Certainly, any cost-effective measures that can increase the mean time between failures and that can detect, if not correct, errors at the hardware level will be beneficial. Parallel file systems, in particular, need to become more reliable and provide better performance. Research must be done into the use of alternative technologies, such as memory and solid state disks, to enable fast, hierarchical checkpointing techniques.

In taking the next steps beyond checkpoint/restart, resilience techniques will require an understanding of the actual errors seen on HPC systems, the rate of these errors, and, ideally, the most common causes. This knowledge is presently unknown, but is required to develop models that are needed to reason about fault detection and recovery. Fault injection tools are needed to test fault tolerance techniques, and these tools will also need accurate statistical models of faults.

11.2 Programming Models and Environment

New application resiliency and fault management techniques require resiliency features and support to be provided by programming models and environments. All levels of the software stack should be involved in the detection, notification, and recovery of faults, and holistic frameworks and standardized APIs must be developed to allow a coordinated response to both hard and soft faults. Such coordination will allow certain types of faults and recovery to be handled by the system software while others are handled by application-specific methods. Interfaces and semantics must be developed that allow programmers to identify critical computations within a code and to specify application-specific recovery techniques beyond global checkpointing. Compiler support could also insert fault-checking mechanisms, like redundancy, into programmer-specified critical code sections.

11.3 Algorithmic-Based Fault Tolerance

Applications, and the numerical engines that drive them, will need to take a more active role in the detection and/or recovery from errors. Research is needed to determine those portions of an algorithm requiring more reliability and those where less reliability is required. In this way, unnecessary error correction can be avoided; lower-level correction mechanisms would be unable to identify such savings. Skeptical programming techniques, using problem meta-knowledge such as conservation or orthogonality properties, can be used to detect errors and force recovery. More numerical algorithms need to be identified and designed that exhibit inherent resilience capability, e.g., algebraic multigrid and row augmentation of dense linear algebra problems, which allow the algorithm to obtain the correct answer even in the presence of faults. Algorithm properties, such as domain of dependence, might be useful to define containment domains for local recovery, and reconstruction techniques based on surrounding, compressed, or multi-level data (e.g., in an AMR hierarchy) could be used to reconstruct local state to reduce synchronization in recovery or possible to avoid local restart altogether. More analysis is needed to understand what errors can be tolerated and the generality of these approaches.

11.4 Correctness

Today, most DOE applications do not employ sophisticated ways of checking the integrity of their results. Generally a subject matter expert is involved in verifying application outputs, but obviously this is challenging and highly impractical. While modules and components are typically verified in isolation, verification of integrated applications is less frequent. Often developers and scientists use other surrogates, such as benchmarking (code-to-code comparison) and reproducibility, to develop confidence in codes.

Reproducibility will be expensive if not impossible to achieve on exascale machines. Requirements for bit-wise reproducibility will need to be relaxed. This will necessitate research into characterizations of acceptable variability in computed results both to enable debugging at scale and to satisfy regulatory constraints.

Formal code verification should be done where possible, but this is in general not a tractable approach for complex multi-physics codes. Code (order) verification is the preferred approach to demonstrating correctness, but this requires a known solution and therefore often fails to test the more complex interactions in the code. The method of manufactured solutions can generate more complex, integrated tests, but tools are needed to automate its use and codes must be developed with the necessary infrastructure to support this approach. Even once a suitable set of test problems is established, the error still needs to be measured, and this is often done by mesh convergence studies where all other code behavior is meant to be held fixed. The execution of such studies at scale is difficult now, but at the exascale will be even worse because of dynamic scheduling, dynamically changing multi-physics modules and models, fault handling, etc., and results will be difficult to interpret. Tools to asses code coverage should be considered to understand dynamic code behavior and the effectiveness of test problems. Certainly, studies can be done at smaller scale and on individual components, but this will not necessarily characterize the behavior of the full code with the many complexities meant to optimize exascale performance, in particular, the effects of coupling errors.

New approaches to verification must be considered. Theoretically justifiable statistical approaches to convergence studies may be necessary. The exascale-motivated rewrite of applications is an opportunity to build *a posteriori* error estimation techniques into application codes, but these techniques need further research to be applicable to the anticipated multi-scale, multi-physics models. The effects of possible faults in these error estimators must also be investigated.

11.5 Impact

Put simply, ensuring correctness and fault tolerance will be necessary to make exascale machines *useful*. Without new fault-management techniques and methods to demonstrate code correctness, it is unlikely that exascale machines will be useful for any significant science simulations; if simulations run to completion, there will be little confidence in the validity of the solution. The strategic risk is not meeting the DOE mission-critical needs. The specific risk is that insufficient resilience will lead to application crashes, hangs, delays, or worst of all, wrong answers.

12 Scientific Productivity

For the scientists who will utilize them, exascale systems will pose many challenges ranging from multiple levels of concurrency to a highly dynamic environment with performance, power, and reliability fluctuating across the system both in space and time. The challenge in accomplishing exascale computations lies in enabling the scientist to use these systems productively. Just because a system is capable of exascale calculations doesn't necessarily mean that scientists will be able to utilize it in a timely manner to perform transformative science. They have to be able to port existing codes, develop new ones, prepare their inputs, and collect and analyze the outputs. Historically, transitions in execution model, such as are anticipated for exascale, have been very disruptive to computational scientists, and it is imperative that DOE endeavor to prevent a repeat of this.

Productivity can have a lot of different metrics when you are discussing HPC and exascale. A common metric is time-to-solution. This relates to the speed at which the scientific calculation can be performed on a given system. Another metric that is just as important is time-to-develop. This is associated with the overall development time of a scientific application, as well as the architecture-specific changes that are required to execute the application on a given system. Each metric is important in the overall productivity of the scientist. Unfortunately, a solution that improves one of them is often detrimental to the other. For example, the effort required to optimize a large-scale application with architecture-specific modifications in order to improve the time-to-solution comes at a great cost to the time-to-develop. Therefore, in scientific productivity we seek solutions to each metric that don't impose a burden on the other.

The initial exascale system's users will likely come from amongst today's cadre of computational scientists. The science being pursued by these investigators addresses the mission of DOE labs and makes up a significant portion of their current HPC workload. This workload comprises large-scale HPC applications that have sometimes been developed over decades and some of their code artifacts are over a million lines of source code. The development and support of these codes equates to an investment of hundreds of full-time equivalent years of labor, often spanning decades [100]. These existing application codes comprise a massive capital investment and this must not be overlooked. To protect this investment we need to enable these large-scale applications to evolve towards exascale in such a way that they can utilize the system effectively and efficiently in a timely manner.

12.1 Research Directions

Improving time-to-solution in HPC tends to come in the form of hardware enhancements. Clearly the importance of this is substantiated by the fact that three of the top ten exascale technology research areas involve hardware (see Sections 5, 4, and 3). The remaining areas link to hardware via the algorithms and software necessary to utilize it efficiently. The hardware solutions addressing data movement improve scientific productivity by reducing the effort required by the scientist to achieve the desired time-to-solution. In addition, these solutions begin to address another challenge placed on the scientist, which is the power constraint for exascale computations.

While hardware advances can aid in the time-to-solution improvements, they often create their own challenges for productivity via the time-to-develop metric. This is a result of the more specialized hardware solutions that historically tend to require the application to utilize architecture-specific performance optimizations in order to compute within multiple constraints. Programming models often try to address these performance challenges (e.g., latency, resource contention, and overhead) and at the same time ease the burden on the scientist (e.g., programmability). A suitable programming model can assist large-scale applications in moving towards exascale systems

in a manner that minimizes the time-to-develop. Programming models that incorporate features about the application's existing programming model (e.g., MPI) stand to accomplish this.

In addition, programming models need to allow easy expression of parallelism, data locality, and resilience, something which is not possible in current models. They need to allow for new abstractions that are better aligned with the platform characteristics (e.g., billion-way parallelism and heterogeneity). Adding to these challenges in productivity is the fact that many of the application codes targeted for exascale utilize existing libraries and software packages built to reflect today's primary execution model, communicating sequential processes [60]. In order to improve scientific productivity, an exascale programming model must ease the transition of both existing application codes and supporting libraries.

Exascale systems will have hardware that is constantly changing during execution due to temperature fluctuations, component failure, and performance variability. This compounds the productivity challenges for scientists because they will need to account for this environment in the development of their applications, something they are not burdened with today. Runtime systems can help address this issue of productivity by disentangling architecture-specific runtime optimizations/adjustments from the algorithmic implementation. For exascale, the objective to be optimized is not only performance but also power, energy and reliability. Applications will need to compute within a power budget and component failure rate, and performance will fluctuate among components and throughout the execution. Enabling the application to make dynamic optimizations with multiple objectives (power, performance, and reliability) will be crucial to scientific productivity.

Among the main productivity advantages offered by runtime systems is their ability to provide resource management and task scheduling without user intervention. Such dynamic control can be adaptive through introspective means, exploiting continuous system status and application demand requirements to provide adaptive guidance of system operation for best behavior. This transition from static to dynamic control increases execution efficiency, especially for highly dynamic and irregular problems like adaptive mesh refinement algorithms. Runtime systems can provide active response to growing asynchrony due to the varying latencies of remote access to data and services. It also permits parallelism discovery such as from the meta-data of large irregular and time-varying graph problems for informatics to deliver greater scalability.

Next-generation runtime systems are under development that support different mixes of several classes of dynamic adaptive functionality. Lightweight multithreading exposes medium granularity parallelism for greater scalability and adaptive scheduling to achieve greater utilization of resources. Message-driven computation (e.g., active messages) provides adaptation to asynchrony and both reduces and hides latency by moving work to the data when appropriate rather than always having to move data to fixed work locations. Synchronization concepts such as dataflow and futures reduce amortized overhead and deliver continuation migration as a new layer of parallelism state. These runtime systems will be able to support legacy codes via conventional programming interfaces. For new code development, they will deliver unprecedented performance and efficiency on future exascale systems while enhancing user productivity, including superior programmability, performance portability, and generality.

Exascale systems are going to provide many challenges to productivity, from disruptive new technology to new programming environments, all of which pose difficulties in getting large scale applications to execute efficiently on the system. The right set of tools to overcome these difficulties is essential, and scalable tools are needed to help the scientists isolate issues with performance, power, resiliency, and correctness within the application's execution. This requires both static and dynamic analysis where the tool can capture the interactions between the application, the runtime, and the hardware to aid in the identification of the problem. There are many research challenges in providing this level of scalable tool support. These tools will need to address some of

the same challenges that the application will have as well, such as dealing with multiple levels of concurrency, a dynamic environment, fault tolerance, awareness of thermal and power constraints, data movement on and off the node, and heterogeneous processors. These tools must understand the hardware of exascale systems to be able to capture their performance and power data in a way that it can overcome data movement challenges and provide the information to the scientists in a timely fashion. Some of this will need to be built into the runtime system to ensure self-awareness to ease the burden of the productivity challenges faced by the scientists.

12.2 Impact

We need to correctly address scientific productivity by ensuring enough resources (e.g., both hardware and software) to minimize both time-to-solution and time-to-develop for scientists needing to perform extreme-scale science on these future exascale computers. Without this investment, scientists will not be able to achieve their desired transformative science.

13 Co-Design and Integration Framework

For the last two decades, the HPC community has successfully used a strategy based on the large-scale integration of commercial-off-the-shelf (COTS) microprocessors into massively parallel supercomputing systems. In some cases, the HPC community has developed technologies to improve the scalability of these systems, e.g., interconnection networks and lightweight kernel operating systems. In recent years, issues with the performance realized at scale have led to questions about the continued viability of this strategy, and the need for research to address this problem. This section builds on the lessons DOE's HPC community has learned in its efforts to influence future commodity processors and other critical technologies such as memory and networking, to extend the viability of the COTS strategy. These lessons inform any future exascale research and development program. The ultimate goal is commercial delivery of a small number of extremely large-scale, integrated systems, and a much, much larger number of smaller-scale commercial systems.

Foremost of these lessons is that all exascale research activity will need to be organized, coordinated and funded in the context of, and in support of, an integrated system design that is focused on meeting the performance, productivity, and energy requirements associated with the anticipated mission-driven exascale workflows of the institutions that will ultimately acquire these systems. The general principle must be that one can't design in isolation. A collection of separate, disjointed research activities, each aimed at optimizing a single design aspect or technology, will likely result in a suboptimal overall solution. The realization of an exascale system will involve a complex, multidimensional tradeoff between hardware (processors, memory, energy efficiency, reliability, interconnectivity), software (programming models, scalability, data management, productivity), and algorithms. Thus, a total systems approach is necessary, implying co-design of hardware and software, and such a total systems approach must inform the research programs for exascale.

Advancing to exascale will require innovations in architecture to integrate the many research results from diverse fields including silicon, memory, packaging, system software, interconnect, signaling and optical technologies, mathematics, and programming models. Exascale architecture requires finding the right balance of these technologies to achieve density, cost, performance and power efficiency at the appropriate levels. Research in exascale architecture will leverage the learning and experience gained in developing and using intermediate systems, and will result in improvements to some of the key elements of architecture, programming models, and system software. Exascale system design will be an optimization problem involving a set of constraints, a set of metrics for evaluating those constraints, and a methodology for evaluating them. The remainder of this section will discuss the architectural constraints, performance metrics, co-design integration framework, and modeling and simulation technology required to achieve a general purpose exascale system by the end of the decade.

13.1 Execution Model

An execution model defines a strategy for performing computation. Also referred to as a "model of computation," an execution model is a crosscutting paradigm that specifies a system holistically, determining the functionality of all system component layers, their interrelationships, and their interoperability. A new execution model is devised to take advantage of advances in technology in order to achieve superior operational properties (e.g., efficiency, scalability, power) while employing methods to correct for potential limiting weaknesses. An execution model establishes the invariants of the semantics and organization that are shared by all systems of its class. It nonetheless permits great flexibility of the specifics of form, function, and scale of underlying implementations.

High performance computing has evolved over seven decades through a succession of mainstream execution models. Their transitions were driven by progress in underlying technologies and the opportunities they enabled. Although differing, sometimes dramatically through successive phase transitions, all HPC execution models address a common set of challenges imposed by the physics of time, work, concurrency, and energy. Each provides different solutions depending on the available enabling technologies (and concepts) at the time they were adopted. Examples of historical execution models include the original von Neumann architecture, vector processors, data parallel systems, shared memory multiprocessors, and communicating sequential processes.

Since 2005, increased device density, power limitations, and increasing relative latencies have forced a transitional period of multicore, heterogeneity, and hybrid programming to sustain continued delivered performance beyond the effectiveness of previous execution models. From this will emerge one or more new parallel execution models to guide future system development and usage methodology. Whether derivatives of prior mainstream models or a distinct departure towards a revolutionary paradigm will prevail is yet to be determined. As a consequence, there is a need now for renewed exploration of innovative execution models to drive future architecture designs, programming interfaces, and cooperating system software.

13.2 Architecture

The goal of delivering an exascale system around the turn of the decade that will provide world-class computational capabilities for workflows requires the integration of modeling and simulation, large datasets, and analytics. This will be a very challenging undertaking for computer architects, who must realize any exascale execution model. Such an exascale system will need to balance desires such as:

1. good sustained performance over a wide range of DOE workloads
2. a floating point capability of 10^{18} operations per second
3. power consumption in the neighborhood of 20 MW
4. mean time between service of at least 24 hours
5. less than 15,000 sq. ft. of floor space, implying that the number of cabinets be under 300, and
6. an interconnect developed in accordance with DOE application requirements.

This balancing act must be performed at a time when the computational science paradigm is shifting from a compute-centric model to a data-centric model expected to be necessary for future scientific workflows. In a compute-centric model, data lives on disk and tape, moving to the CPU as needed through a deep storage hierarchy. This model is sufficient when the computational aspects of the workflow dominate the data movement aspects. In contrast, in the data-centric model, data lives in persistent memory, with many CPUs surrounding it, and the data moves as little as possible through a shallow/flat storage hierarchy.

A first-order constraint for exascale is that the system will have be designed to be commercially viable. The system's architecture and components will be designed to enable it to scale from small commercially competitive solutions to specialized solutions at the exascale range and beyond. The system will need to be of sufficient prominence in the HPC industry to drive independent software vendors and ecosystems. However, as exascale requirements are more aggressive than commercial

markets demand, additional research and development, i.e., additional investment, will be required to implement the innovations which will enable an exascale-class system that meets DOE needs in the next decade. Moreover, in order to make exascale systems economically feasible, and to truly ensure U.S. leadership in this arena, any exascale development program must be carried out in the context of a commercially viable systems roadmap. The criterion for an outlook on commercial viability, which implies viability in a broader market than exascale computing, imposes a severe constraint on the realizability of various technology directions, and curtails the prospects of narrow research applicable to supercomputing only.

Each of the top ten exascale research challenges that we have identified address some subset of these crucial design challenges that hold the key to future performance growth. At a high level the architectural design challenges that must be overcome to achieve the required 1000x growth in delivered application performance for mission-critical applications of all scales are as follows.

1. Execution models that enable dynamic adaptive methods in runtime system software to guide computing resource management and task scheduling to achieve high efficiency, portability, and programmability.

2. A data-centric paradigm to control and minimize data movement to overcome the extreme cost of data movement relative to the cost of computation. The exascale solution will include compute, network, storage, software and I/O components, and will be targeted to meet important DOE and commercial workload requirements.

3. Introspection for active control of energy usage, fault responsiveness, and locality management, and

4. Hardware mechanisms and structures to support finer-granularity parallelism and control of data locality.

Together these approaches will provide a new approach to provide the required improvements in performance and scalability to achieve exasclae within the context of current and future technology trends.

13.3 Performance Metrics

Research into execution models, architectures that realize them, and the co-design process that controls it need to be guided by a performance framework that enables investigators to understand the implications of their design choices. Performance growth in the last two decades, the CSP or MPI era, has been largely though scaling problem sizes, so called *weak scaling*. As discussed above in section 5, the volume of memory will scale more slowly than computing power, and performance growth in the next decade will require more attention to *strong scaling* in which processes interact more frequently. An example of such a performance framework is provided here. There are others.

Fundamental properties of physics, structure, and order determine the performance and other operational properties delivered by any HPC system. The basic measures of time, work, energy, and concurrency define the base-level parameter space within which computation evolves and performance is achieved. Execution models derived to guide the development of HPC systems have differed, sometimes dramatically, in response to the changes of their respective enabling technologies. However, throughout the history of HPC, the execution models applied have shared the same underlying physical challenges to realizing best performance within practical limitations of these technologies. These physical challenges can be codified as a performance framework within the definition of which the strategies of all parallel execution models can be assessed and compared.

A performance framework is derived from a relationship among basic parameters as described above. A formulation for (average) performance is given in the following expression:

$$P = S \times e(L,O,W) \times \mu(E) \times \alpha(R)$$

Where:

$$P = \text{performance}$$
$$S = \text{scaling}$$
$$e = \text{efficiency}$$
$$L = \text{latency}$$
$$O = \text{overhead}$$
$$W = \text{contention}$$
$$\mu = \text{speed normalization}$$
$$E = \text{energy}$$
$$\alpha = \text{availability}$$
$$R = \text{reliability}$$

Implicit in this relationship and its factors are time and work from which performance is derived. Efficiency and availability range between 0 and 1, exclusive. Speed is the sequential rate of operation execution and is a function of the rate of energy consumed (more power allows for a higher clock rate). Overhead is the amount of work required to manage parallel resources and concurrent tasks (e.g., synchronization) that would not be required in a purely serial version of the computation. Latency is the distance (often measured in time or clock cycles) of remote object access or distant service request (e.g., main memory access, message passing). Reliability is a measure of mean time to failure that contributes to the determination of a system's effective availability.

The key properties that all HPC execution models must address are identified through this expression. Cast as sources of performance degradation, this SLOWER framework exposes the set of challenges to be overcome by execution models employed to guide the development and operation future supercomputers. These are:

- *Starvation:* insufficiency of computational work to keep all functional resources utilized either because the total work is less than all of the system resources or because the workload is imbalanced, causing some elements to have to much to do while others have too little.
- *Latency:* the time it takes to move information between two points within a system such as network latency or memory access latency usually measured in processor core cycles or nano/micro/milliseconds.
- *Overhead:* the amount of additional work required to manage parallel resources and concurrent task scheduling among other functions not required for pure sequential execution.
- *Waiting:* for contention of shared logical (e.g., synchronization) and physical (e.g., network links) resources.
- *Energy:* and the rate at which it is consumed (power) determine the amount of resources used to perform computation and the rate at which they operate (e.g., clock rate).
- *Reliability:* determines the mean time to interrupt of system operation caused by hardware and software system faults.

13.4 Co-Design Integration Framework

Co-design is the over-arching framework used to evaluate the trade-space in the context of a performance framework (e.g., SLOWER). Hardware/software co-design has long been a feature of power-sensitive embedded system designs, but thus far has seen very little application in the HPC space. Co-design refers to a design methodology to partition work in integrated processor/software solutions that are optimized to minimize energy consumption to perform a given task. The embedded processor market has refined co-design processes over the past twenty years to meet the demanding cost and power efficiency requirements of battery-powered consumer electronics applications, as well as 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 tools inlcude automated processor synthesis tools, cycle accurate simulators, and automated generation of software tools (compilers and debuggers) from hardware specifications. Now that power has become the leading design constraint of HPC systems, co-design and other application-driven design processes are expected to play a central role in the development of energy-efficient exascale computing systems that serve a diverse range of applications.

Today's DOE co-design efforts are investigating a diverse set of technical areas: architecture-aware algorithms, programming models, system software, hardware architectures, resiliency, power management, etc. The key tools applied for these investigations are:

- Proxy Applications
- HPC Architectural Simulators
- Experimental, advanced architecture testbeds

The naive projection of existing co-design processes onto the HPC design space exposes many gaps in the methodology. There is no tool or methodology to extrapolate design requirements to computing systems of unprecedented size that are yet to be built. Embedded co-design tools never conceived of simulating interconnects with millions of interacting endpoints, where each endpoint is itself a complex system. Furthermore, the programming model and software environment anticipated for future extreme-scale systems is anticipated to be substantially different than current practice. In order to make predictive modeling of large-scale systems tractable, we have moved towards using both simplified *proxy applications* together with simplified (Proxy-Architecture) models for the *important* aspects of future systems as shown in Figure 9.

Characteristics that impact performance should be understood as early as possible in the analysis and design of new computers. Furthermore, it is often the case that there are multiple ways to design and implement the algorithms used in an application, and the choice can have a dramatic impact on the application's performance. The proxy applications fill this gap, representing critical algorithmic kernels, yet being small enough to be reengineered to experiment with different execution models and their realizations, whether in modeling or system prototypes. Hardware architectural choices, if properly exposed through the system software to the application developers, can likewise have a tremendous impact on performance. Architectural simulations with enough fidelity to illuminate the impact of algorithmic and implementation design decisions allow the co-design process to advance concurrently with development of the exascale system's hardware and software.

A more application-centric view of the co-design cycle is provided by the Materials Co-Design Center [1] in Figure 10. Accurate modeling and simulation tools enable quantification of the

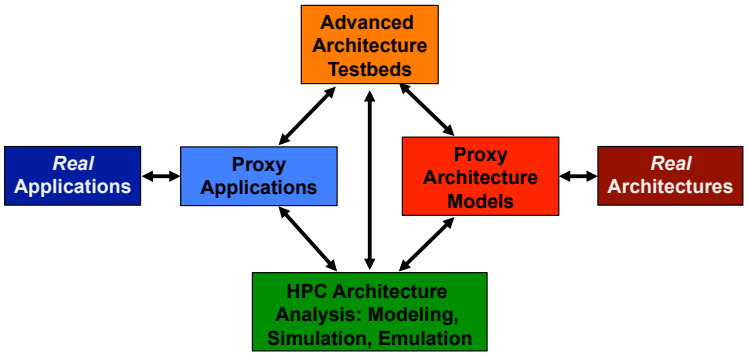


Figure 9: Simplified versions of the target hardware and target application enable rapid prototyping and assessment of future hardware/software interactions. Accurate proxy models for the hardware and software make the size and complexity of the modeling problem tractable. Accurate predictive modeling and simulation tools for future hardware are essential for enabling crucial design decisions for very expensive system designs before they are built (or written in the case of software).

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. The application of this design cycle in the embedded space has enabled a revolutionary advancement of the technology in that space, and the success in that space gives us strong confidence that it will enable orders of magnitude improvement in energy efficiency and usable performance in the HPC space.

It is also critical to have a diverse set of experimental architecture test beds to verify the co-design process and guide technology investment decisions. As a community, access to and experience with these experimental architecture testbeds will allow DOE to become more informed collaborators in co-design processes, more adaptable to changes in hardware, and have a stronger basis for making programming model changes. Perhaps more importantly, this experience will provide a foundation for decision makers to determine the path to exascale while the program continues to meet mission obligations.

Looking forward, it is likely that some revolutionary technologies will be required, particularly if one considers the low fraction of peak performance achieved by many HPC applications today, along with the need to avoid prohibitive power costs. It is also likely that many technologies for HPC will continue on an evolutionary track. DOE has historically adopted revolutionary technology changes, up to and including rewriting its applications, if that was required to meet mission needs. Recall that the ASCI program made sustained, substantial investments to transform its application code base from vectors to MPI, thus to creating the current NNSA application portfolio. In much the same way, any DOE exascale program must undertake the challenge of transforming current multi-scale, multi-physics applications to incorporate additional or different programming models. Due to the DOE investment in its application code base, there is tremendous incentive to pursue revolutionary hardware and system software technologies, if it enables application software to stay on an evolutionary path.

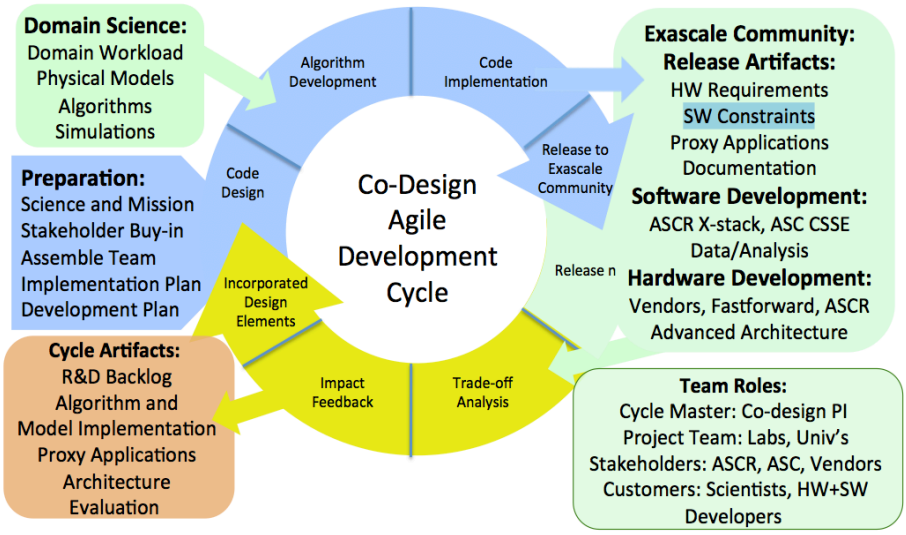


Figure 10: Typical co-design process for the design of co-tuned hardware and software systems.

13.5 Modeling and Simulation of Exascale Systems and Applications

In the co-design process discussed above, two or more factors are optimized in concert to achieve a better solution. Exascale systems and applications create a multidimensional space for optimization. To date, the DOE co-design process has not had the modeling and simulation tools to cover all of this space in a comprehensive fashion. That is, given a performance framework such as the one described above, investigators cannot yet evaluate alternative design choices, and rigorously identify the most profitable ones. Hence, achieving exascale will require the development of suitable co-design tools, including modeling and simulation, that address all aspects of the exascale design space. This spans the algorithms chosen, the scientific and engineering applications, the programming models used to create them, the runtime environment they execute in, and the architectures they run on.

Figure 11 illustrates the scale of the exascale design space. It shows a continuum of activities undertaken by the HPC community to connect the nation's overall scientific and engineering challenges to the computations that support them. Each box in the figure notionally represents a different area of emphasis and a different scale of the HPC workflow. For exascale it is first necessary to articulate key scientific goals such as energy independence or understanding combustion. The goals are extraordinary complex computational problems that need be solved on an exascale system. The computational problems need to be tackled via appropriate application, machine, and programming model designs. The executing systems and applications will need to be continuously monitored and optimized to ensure continued high performance in the face of changing application requirements and system status.

Modeling and simulation research is a vital component of every aspect of computational planning and execution as it enables community leaders and individual system and application developers to predict the outcomes of various decisions undertaking during design, implementation, and maintenance project phases. The widespread applicability of modeling and simulation is clearly shown by the red asterisks in Figure 11, indicating the specific areas in which modeling and simulation are

applied, makes this point amply clear. This capability is instrumental for enabling the computational science and HPC communities to design computing strategies as well as concrete systems and applications that provide a balance of the above productivity metrics and enable new breakthrough science that will provide high value in a cost-effective manner [69].

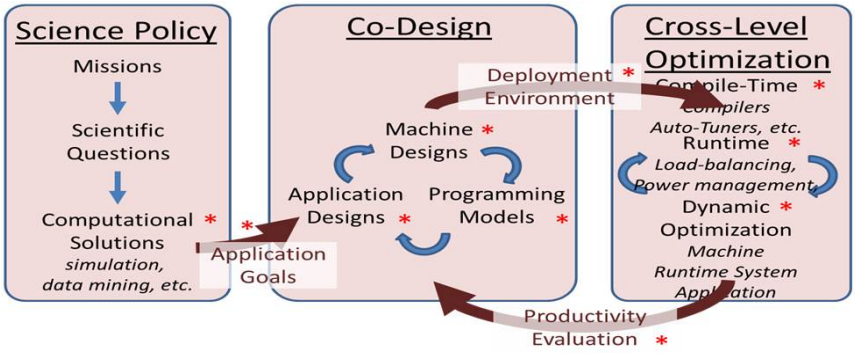


Figure 11: Notional HPC workflow. The * represent domains for modeling and simulation

The current state-of-the-art in modeling methodologies has developed in response to the architectures and application structures typically found in today's HPC landscape. For some time now, the complexities inherent in HPC architectures have meant that increases in peak performance have not always translated into improvements in application performance. Models have served as an ideal tool for quantifying the performance impact expected from changes in application and system architectures. As we move towards exascale, however, many of the assumptions built into these methods will no longer hold true.

Models and modeling methodologies will need to adapt to an increasingly complex landscape. Models will be key tools in the area of application/system co-design. As applications and systems both evolve simultaneously, models must be able to track ongoing complex changes and predict the impact of developments in both software and hardware design. While today's methods tend to focus on application performance as the metric of concern, exascale modeling methods must evolve to consider performance, power consumption, and reliability in concert. Construction of models needs to be further automated, both to reduce the cost of labor involved as well as to make it feasible to dynamically adapt them as application requirements or system status changes. Ideally, they become actionable tools, available at runtime, that enable an introspective exascale system to reason about how to best full manage its resource to maximize scientific throughput. While such objectives represent a significant step forward beyond today's techniques, achieving them is important for future large-scale design as power becomes a primary concern and system complexity continues to increase.

There are significant technical challenges facing the modeling and simulation of the performance, power consumption and reliability of both exascale applications and systems. Given the centrality of modeling and simulation in the co-design process, and its time criticality, new resources need to be made available for this activity in an overall exascale effort. Such resources need to materialize through funded projects in methodology and modeling and simulation tool development, which are tightly integrated with similar activities in architecture, applications, and system software. There is also the need for specialized testbeds to enable testing, verification and validation, and scale-up of the modeling and simulation tools-of-the-trade.

14 Findings and Recommendations

On July 29, 2013, ASCAC was charged with developing a list of the top ten principal research challenges that must be overcome to develop a practical exascale computing system. These have been enumerated in the introduction and described in ten sections of this report. In the course of conducting this study, the subcommittee considered a number of related issues that add context to the top ten research challenges and the means to overcome them. The findings and recommendations that arose from this are presented here.

14.1 Findings

14.1.1 Exascale computing is critical for executing the DOE mission.

ASCAC reaffirms its findings from previous reports that leadership in high performance computing is critical to achieving the DOE mission of ensuring U.S. leadership in science, engineering, and national security. Not only a DOE mission, leadership in the broad spectrum of HPC technologies and applications is also a foundation supporting American economic competitiveness. In the last six years, this has been documented in many exascale reports from Office of Science programs, the National Nuclear Security Administration, and other U.S. government agencies [33, 34, 42, 72, 90, 105, 107, 108, 118].

14.1.2 U.S. national leadership is at risk.

Without aggressive investment and technical innovation in HPC, the U.S. risks falling behind rapidly emerging international competitors, not all of whom are friendly to U.S. interests [93]. This in turn threatens to undermine the nation's intellectual leadership in a broad range of science, its economic position, and its security. As was true during the height of the nuclear arms race, the U.S. cannot slow down its competitors, but DOE can invest to stay ahead of them.

14.1.3 The U.S. has the technical foundation to create exascale systems.

The U.S. semiconductor and HPC systems industries are capable of developing the necessary technologies for an exascale computing capability by the early part of the next decade, based largely on evolving commercially driven component fabrication, systems integration, and software engineering capabilities. However, for a truly effective and productive exascale computing capability, the U.S. government will need to focus investments on the research, development, and integration of HPC technologies that otherwise will not be created solely for commercial ends. Furthermore, a focused exascale research program will mobilize mathematicians, scientists, and engineers in national laboratories and academia who are capable of creating the necessary algorithms and software, critical to DOEs mission among others, to augment those furnished by industry, and make a truly effective and productive exascale system.

14.1.4 An evolutionary approach to achieving exscale will not be adequate.

The dramatic improvements essential to achieving effective exascale computing will not be satisfied by incremental extensions to today's conventional practices. In the five years since Roadrunner first exceeded a PFlop/s of performance at Los Alamos National Laboratory, significant progress in pushing peak capability has continued with China's Tianhe-2 recently achieving a HPL throughput of 33.86 PFlop/s. However, this has not been accomplished without encountering the concerns that motivate the DOE exascale initiative. Tianhe-2 draws 17.8 MW, and simply following today's

technology trends to exascale would require hundreds of megawatts – enough to power a small city. Memory per core is dropping. Bandwidth, whether to memory or across the interconnect, is not keeping up with increasing processor performance. Furthermore, heterogeneous machines such as Tianhe-2 are difficult to program and serve a small subset of the applications needed to support DOE's broad mission. Therefore, we find that commercial market drivers do not provide a viable general path to delivering necessary scalability, time and energy efficiency, and user productivity, including performance portability for future exascale-class computers.

14.1.5 The U.S. government's continuous leadership and investment are required to create exascale systems.

The U.S. computing industry is unlikely to develop effective exascale computer systems without U.S. government investment and focused mission goals. This is due to disparities between commercial roadmaps targeted at emerging markets and the necessary revolutionary innovations needed for exascale computing. An evolutionary exascale system, built through incremental changes to today's conventional practices, will be of limited utility to DOE computational scientists. We find that innovation, sometimes of an incremental nature and in other areas revolutionary, will be required under DOE direction to continue U.S. leadership in advanced HPC.

14.2 Recommendations

14.2.1 DOE should initiate a program of continuous advancement in HPC.

Exascale is only the next milepost in a half-century of continuous progress towards increasing capability in computational science. The U.S. government requires a stable, long-term investment strategy to ensure continuous U.S. leadership in HPC beyond today's petascale performance regime, extending across exascale to zetascale and beyond. In the immediate future, much of that research investment should be focused on the top ten challenges identified within this report. That focus will change with time as those challenges recognized today are overcome, and new ones emerge.

Establishing consistency of direction and commitment will engage industry partners, ensure government resources, and encourage laboratory and academic researchers to conduct sustained research in achieving critical capabilities over the coming decades. Towards that end, DOE must derive and establish minimum enabling technology requirements for the U.S. trans-exascale performance regime and devise a plan of partnership and commitment with industry and academic researchers to address these needs over sufficient duration to be achievable and timely.

This U.S. government research investment in exascale technology needs to be relatively large now, as the end of Dennard scaling and the looming end of Moore's Law portend a significant change in HPC architectures. Change of this magnitude has not been seen in two decades, since CMOS technology eclipsed ECL. Once a new execution model suitable for trans-exascale systems has been derived, then we anticipate that a smaller, sustained investment will enable DOE to effectively and efficiently use its HPC applications on multiple generations of exascale systems.

14.2.2 DOE should invest in its industrial base to catalyze the foundation for exascale systems.

DOE should invest in extending commercial semiconductor, communications, systems integration, and software technologies to prepare the U.S. industrial base for its role and contributions in future HPC scientific, engineering and national security missions. As discussed in this report, of particular concern is the energy consumption (section 3) and resilience (section 11) of anticipated exascale

systems, due to their size and complexity. There are also both opportunity and concern regarding the memory capacity and capability that will be affordable (section 5), as well as the means of the integration of these new technologies. Finally, suitable networks and communication protocols (section 4) must exist to enable effective exascale computing for new modalities of information exchange and distributed control. All of these exascale components must be developed by and be available from U.S. sources, otherwise the supply chain is vulnerable to interdiction by foreign powers, which in turn could threaten the nation's security [37]. Such investments would be natural successors to DOE's investment in the DARPA High Productivity Computing Systems Program, and DOE's PathForward, FastForward, and DesignForward initiatives.

14.2.3 DOE should invest in exascale mathematics and system software responsive to DOE missions and other U.S. government requirements

The mathematical algorithms needed for many DOE missions are unique and must be revised or re-designed to function at exascale. As discussed in this report (sections 9,10,11), these challenges include exposing billion-fold concurrency, overcoming challenges due to asynchrony, multi-tiered memory, increasing fault rates, and quantifying the uncertainty of the computations being performed.

As with today's leadership-class systems, much of the software infrastructure of an exascale system will be unique to its scale and the missions for which DOE will deploy it. Therefore, as discussed in this report, DOE must invest in scalable operating systems (section 6), runtime systems, and tools for the management of the data that will be generated and/or processed (section 8).

New programming constructs, perhaps even a new language, will likely be required for exascale computation and applications. As discussed in this report (section 7), this parallel programming system must expose and exploit billion-way concurrency, support application performance portability across different hardware platforms, and interface to dynamic adaptive mechanisms from runtime system software and hardware architecture for high efficiency and parallelism discovery. DOE should oversee and sponsor the community-wide process for the development of such a parallel program capability.

The productivity of DOE computational scientists must be significantly improved relative to current practices in order for exascale HPC to prove an effective means of achieving strategic DOE mission goals. Productivity demands that applications realize most of the potential performance opportunity without undue programmer effort. Furthermore, performance portability must be attained such that effective performance is gained as applications move across system classes and scales without program redefinition in each case. As discussed in this report (section 12), new methods of runtime introspection and programming semantics will have to be developed. Furthermore, to ensure that the next generation of these computational scientists is available, DOE should adjust investments in programs such as fellowships, career awards, and funding grants, to increase the pool of computational scientists trained in both exascale and applied mathematics. Opportunities should also be provided for the next generation computational scientists to work closely with domain scientists in the development of scientific applications for exascale platforms.

We also note that the mathematics and software developed for exascale should be useful on systems of all scale, from workstations, to departmental servers, to leadership class supercomputers. It is important that DOE make its research investments with an eye to delivering portability and performance to scientists and engineers regardless of the scale of the computing infrastructure available to them.

14.2.4 DOE should create an Open Exascale System Design Framework to enable cooperative hardware and software advancement

To guide U.S. investments in exascale research, DOE should assume a leadership role in fostering innovative strategies to establish a new discipline capable of exploiting billion-way parallelism, in the context of asynchrony and to support of a new generation of parallel algorithms that will be derived for the new architectures being developed. DOE's understanding of mission drivers provides a critical foundation to coordinate requirements for and development of interoperable system components, establish a co-design framework for collaboration and system integration of crosscutting component layers. Such a context will serve as the conceptual scaffolding for the development of new programming language and hardware architecture, runtime software and operating system, and application algorithms and management policies for future exascale computing. The principles underlying such a framework were discussed in section 13 of this report.

DOE should conduct one or more point design studies, based on co-design principles, to establish the opportunity and challenges of the design space prior to committing to either an incremental or revolutionary path forward. Such studies should be concepts-driven, addressing fundamental technology challenges and application requirements. Other government agencies and HPC stakeholders should be invited to participate in these studies. The quantitative results produced will provide confidence in future directions, permit open evaluation and comparative analysis, and deliver engineering specifications for detailed designs. Such studies must be crosscutting from device technology foundations to applications with high-level programming interfaces, and include system architecture and software layers. They should complement and benefit from on-going U.S. government-sponsored research programs in programming models, system software, applications along with their data and workflows, component technologies, and architecture.

15 Bibliography

References

[1] Materials Co-Design Center. http://www.exmatex.org.

[2] A. Agarwal, S. Mathew, S. Hsu, M. Anders, H. Kaul, F. Sheikh, R. Ramanarayanan, S. Srinivasan, R. Krishnamurthy, and S. Borkar. A 320mv-to-1.2v on-die fine-grained reconfigurable fabric for dsp/media accelerators in 32nm cmos. In *ISSCC*, pages 328–329, 2010.

[3] http://www.mcs.anl.gov/project/argo-exascale-operating-system.

[4] I. M. Babuška, F. Nobile, and R. Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM J. Num. Anal.*, 45(3):1005–1034, 2007.

[5] I. M. Babuška, R. Tempone, and G. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.*, 42(2):800–825, 2004.

[6] J. Balfour and W. Dally. Design tradeoffs for tiled CMP on-chip networks. In *International Conference on Supercomputing*, 2006.

[7] M. Bancroft, J. Bent, E. Felix, G. Grider, J. Nunez, S. Poole, R. Ross, E. Salmon, and L. Ward. Hec fsio 2008 workshop report. high end computing interagency working group (heciwg) sponsored file systems and io. In *IO Workshop HEC FSIO*, 2009.

[8] L. Barba and R. Yokota. CSE 2013: How will the fast multipole method fare in the exascale era? *SIAM News (Society for Industrial and Applied Mathematics)*, 46(6):2, 2013.

[9] E. Barton. Fast Forward IO and Storage @ONLINE, Nov. 2012.

[10] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

[11] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[12] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate. Plfs: a checkpoint filesystem for parallel applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, page 21. ACM, 2009.

[13] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[14] A. Biberman and K. Bergman. Optical interconnection networks for high-performance computing systems. *Reports on Progress in Physics*, 75(4):046402, 2012.

[15] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders. *Large-Scale PDE-Constrained Optimization*. Springer Berlin Heidelberg, 2003.

[16] J. R. Birge and F. V. Louveaux. *Introduction to stochastic programming*. Springer, 1997.

[17] G. Biros and O. Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. part i: The Krylov–Schur solver. *SIAM J. Sci. Comput.*, 27(2):687–713, 2005.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[19] R. Brightwell, R. Oldfield, A. B. Maccabe, and D. E. Bernholdt. Hobbes: Composition and virtualization as the foundations of an extreme-scale os/r. In *In Proceedings of the Workshop on Runtime and Operating Systems for Supercomputers, Eugene, OR*, June 2013.

[20] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *Int. J. High Perf. Comput. App.*, 14(3):189–204, 2000.

[21] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders. Goal-oriented model-constrained optimization for reduction of large-scale systems. *Journal of Computational Physics*, 224:880–896, 2007.

[22] J. Burkardt, M. Gunzburger, and H.-C. Lee. Centroidal Voronoi tessellation-based reduced-order modeling of complex systems. *SIAM J. Sci. Comput.*, 28:459–484, 2006.

[23] J. Burkardt, M. Gunzburger, and C. G. Webster. Reduced order modeling of some nonlinear stochastic partial differential equations. *International Journal of Numerical Analysis and Modeling*, 4(3-4):368–391, 2007.

[24] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.*, 155:468–498, 1999.

[25] A. T. Chronopoulos. *s*-step iterative methods for (non) symmetric (in) definite linear systems. *SIAM Journal on Numerical Analysis*, 28(6):1776–1789, 1991.

[26] B. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM News (Society for Industrial and Applied Mathematics)*, 33(4):2, 2000.

[27] A. Conn, N. Gould, and P. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. *Computing Methods in Applied Sciences and Engineering*, pages 42–54, 1990.

[28] F. E. Curtis, O. Schenk, and A. Wächter. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations. *SIAM J. Sci. Comput.*, 32(6):3447–3475, 2010.

[29] T. H. de Mello. Variable-sample methods for stochastic optimization. *ACM Trans. Model. Comput. Simul.*, 13(2):108–133, 2003.

[30] E. de Sturler and H. A. van der Vorst. Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. *Applied Numerical Mathematics*, 18(4):441–459, 1995.

[31] M. K. Deb, I. M. Babuška, and J. T. Oden. Solution of stochastic partial differential equations using Galerkin finite element. *Comput. Methods Appl. Mech. Engrg. 2*, 190:6359–6372, 2001.

[32] R. Dennard et al. Design of ion-implanted mosfets with very small physical dimensions. *IEEE Journal of Solid State Circuits*, SC-9(5):256–268, October 1974.

[33] Scientific Grand Challenges: Architectures and Technology for Extreme Scale Computing. US Department of Energy Workshop Report, December 2009.

[34] Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale. US Department of Energy Workshop Report, February 2010.

[35] Department of Energy ASCR/ASC Fast Forward Program. http://science.energy.gov/~/media/_/pdf/news/in-the-news/2012/extreme_scale_supercomputers.pdf, Oct. 2012.

[36] DOE ASCR Workshop on Exascale Data Management, Analysis and Visualization . http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Exascale-ASCR-Analysis.pdf, Feb. 2011.

[37] Defense Science Board Task Force On High Performance Microchip Supply. http://www.acq.osd.mil/dsb/reports/ADA435563.pdf, Feb. 2005.

[38] A. Edelman, P. McCorquodale, and S. Toledo. The future fast Fourier transform? In *PPSC*. SIAM, 1997.

[39] H. C. Elman, O. G. Ernst, and D. P. O'Leary. A multigrid method enhanced by Krylov subspace iteration for discrete helmholtz equations. *SIAM Journal on scientific computing*, 23(4):1291–1315, 2001.

[40] H. C. Elman, C. W. Miller, E. T. Phipps, and R. S. Tuminaro. Assessment of collocation and galerkin approaches to linear diffusion equations with random data. *International Journal for Uncertainty Quantification*, 1(1), 2011.

[41] O. G. Ernst and E. Ullmann. Stochastic galerkin matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1848–1872, 2010.

[42] Report on the Workshop on Extreme-Scale Solvers: Transition to Future Architectures. http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/reportExtremeScaleSolvers2012.pdf, 2012.

[43] http://xstack.sandia.gov/xpress.

[44] M. Ferris and O. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

[45] G. Fishman. *Monte Carlo*. Springer Series in Operations Research. Springer-Verlag, New York, 1996. Concepts, algorithms, and applications.

[46] C. A. Floudas. Mixed-integer nonlinear optimization, 1995.

[47] R. G. Ghanem and R. M. Kruger. Numerical solution of spectral stochastic finite element systems. *Computer Methods in Applied Mechanics and Engineering*, 129(3):289–303, 1996.

[48] R. G. Ghanem and P. D. Spanos. *Stochastic finite elements: a spectral approach*. Springer-Verlag, New York, 1991.

[49] R. Giering and M. Voßbeck. Increasing memory locality by executing several model instances simultaneously. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 93–101. Springer Berlin Heidelberg, 2012.

[50] A. Gordon and C. Powell. On solving stochastic collocation systems with algebraic multigrid. *IMA Journal of Numerical Analysis*, 32(3):1051–1070, 2012.

[51] The Green 500 List. http://www.green500.org, 2013.

[52] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, Aug. 1987. This paper is credited as the origin of the fast multipole method, with an $O(N)$ algorithm. It was reprinted in the same journal, vol. 135, pp. 280–292, August 1997.

[53] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numer.*, 6:229–269, 1997.

[54] G. Grider. Exascale fsio: Can we get there? can we afford to. In *Proceedings of the HEC FSIO R&D Workshop 2010*, 2010.

[55] G. Grider. The curse of the burst - extreme hpc i/o and storage futures @ONLINE, Dec. 2013.

[56] M. Gunzburger, J. Peterson, and J. Shadid. Reduced-order modeling with multiple parameters. *Comp. Meth. Appl. Mech. Engrg.*, 196:1030–1047, 2006.

[57] J. Helton and F. Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering and System Safety*, 81:23–69, 2003.

[58] G. Hendry, S. Kamil, A. Biberman, J. Chan, B. Lee, M. Mohiyuddin, A. Jain, K. Bergman, L. Carloni, J. Kubiatowicz, L. Oliker, and J. Shalf. Analysis of photonic networks for a chip multiprocessor using scientific applications. In *International Symposium on Networks-on-Chip (NOCS)*, 2009.

[59] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L. P. Carloni, N. Bliss, and K. Bergman. Time-division-multiplexed arbitration in silicon nanophotonic networks-on-chip for high-performance chip multiprocessors. *J. Parallel Distrib. Comput.*, 71(5):641–650, May 2011.

[60] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, Aug. 1978.

[61] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization.* Kluwer, Boston, 2nd edition, 2000.

[62] P. D. Hough, T. G. Kolda, and V. J. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1):134–156, 2001.

[63] S. Hsu, A. Agarwal, M. Anders, M. K. S. Mathew, H. Kaul, F. Sheikh, R. Krishnamurthy, and S. Borkar. 2.8ghz 128-entry ? 152b 3-read/2-write multi-precision floating-point register file and shuffler in 32nm cmos. In *Symposium on VLSI Circuits (VLSIC)*, June 2012.

[64] C. Jin and X. Cai. A Preconditioned Recycling GMRES Solver for Stochastic Helmholtz Problems. *Communications in Computational Physics*, 6(2):342–353, 2009.

[65] C. Jin, X. Cai, and C. Li. Parallel Domain Decomposition Methods for Stochastic Elliptic Equations. *Siam Journal On Scientific Computing*, 29(5):2096–2114, 2007.

[66] S. Kamil, L. Oliker, A. Pinar, and J. Shalf. Communication requirements and interconnect optimization for high-end scientific applications. *IEEE Transactions on Parallel and Distributed Systems*, 21:188–202, 2010.

[67] H. Kaul, M. Anders, S. Mathew, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar. A 320mv 56w 411gops/watt ultra-low voltage motion estimation accelerator in 65nm cmos. In *ISSCC*, pages 316–317, 2008.

[68] H. Kaul, M. Anders, S. Mathew, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar. A 300mv 494gops/w reconfigurable dual-supply 4-way simd vector processing accelerator in 45nm cmos. In *ISSCC*, pages 260–261. IEEE, 2009.

[69] D. Kerbyson, A. Vishnu, K. Barker, and A. Hoisie. Codesign challenges for exascale systems: Performance, power, and reliability. *Computer*, 44(11):37–43, 2011.

[70] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502, 2002.

[71] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, and K. Yelick. Exascale computing study: Technology challenges in achieving exascale systems. Technical report, DARPA, 2008.

[72] P. Kogge et al. Exascale computing study: Technology challenges in achieving exascale systems. http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf, 2008.

[73] P. Kogge and J. Shalf. Exascale computing trends: Adjusting to the new normal for computer architecture. *Computing in Science and Engineering*, November/December 2013.

[74] M. H. Langston, M. M. Baskaran, B. Meister, N. Vasilache, and R. Lethin. Re-introduction of communication-avoiding fmm-accelerated ffts with gpu acceleration. In *IEEE Conference on High Performance Extreme Computing (HPEC), Waltham, MA, USA*. IEEE, September, Sep 2013.

[75] M. H. Langston, L. Greengard, and D. Zorin. A free-space adaptive FMM-based PDE solver in three dimensions. *Comm. in Applied Mathematics and Comp. Science*, 6(1):79–122, 2011.

[76] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. S. Sampath, A. Shringarpure, R. W. Vuduc, L. Ying, D. Zorin, and G. Biros. A massively parallel adaptive fast-multipole method on heterogeneous architectures. In *SC*. ACM, 2009.

[77] C. D. Levermore. Moment closure hierarchies for kinetic theory. *J. Stat. Phys.*, 83:1021–1065, 1996.

[78] M. Lipson. Silicon photonics: The optical spice rack. In *Photonics in Switching, 2009. PS '09. International Conference on*, pages 1–1, 2009.

[79] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn. On the role of burst buffers in leadership-class storage systems. In *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*, pages 1–11. IEEE, 2012.

[80] J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

[81] R. McClarren and C. Hauck. Simulating radiative transfer with filtered spherical harmonics. *Physics Letters A*, 374(22):2290–2296, 2010.

[82] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[83] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[84] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, page 4, 1965.

[85] T. Munson, J. Sarich, S. M. Wild, S. Benson, and L. Curfman McInnes. TAO 2.0 users manual. Technical Memorandum ANL/MCS-TM-322, Argonne National Laboratory, Argonne, Illinois, 2012.

[86] D. Nagle, D. Serenyi, and A. Matthews. The panasas activescale storage cluster: Delivering scalable high bandwidth storage. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 53. IEEE Computer Society, 2004.

[87] S. G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000.

[88] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.

[89] H. Niederreiter. Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Amer. Math. Soc*, 84(6):957–1041, 1978.

[90] Scientific Grand Challenges for National Security: The Role of Computing at the Extreme Scale. US Department of Energy Workshop Report, October 2009.

[91] F. Nobile, R. Tempone, and C. G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2411–2442, 2008.

[92] F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008.

[93] O. of Net Assessment. 2012 national defense authorization act net assessment of high performance computing capability possessed by foreign countries. Technical report, US Department of Defense, 2012.

[94] C. on the Mathematical Sciences in 2025; Board on Mathematical Sciences, T. A. D. on Engineering, and P. S. N. R. Council. *Fueling Innovation and Discovery: The Mathematical Sciences in the 21st Century*. The National Academies Press, 2012.

[95] N. Ophir, C. Mineo, D. Mountain, and K. Bergman. Silicon photonic microring links for high-bandwidth-density, low-power chip i/o. *Micro, IEEE*, 33(1):54–67, 2013.

[96] M. Parks, E. De Sturler, G. Mackey, D. Johnson, and S. Maiti. Recycling krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[97] R. P. Pawlowski, E. Phipps, and A. G. Salinger. Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part I: Template-based generic programming. *Scientific Programming*, 20:197–219, 2012.

[98] M. F. Pellissetti and R. G. Ghanem. Iterative solution of systems of linear equations arising in the context of stochastic finite elements. *Advances in Engineering Software*, 31(8):607–616, 2000.

[99] E. Phipps and R. Pawlowski. Efficient expression templates for operator overloading-based automatic differentiation. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, 2012.

[100] D. E. Post and R. P. Kendall. Software project management and quality engineering practices for complex, coupled multiphysics, massively parallel computational simulations: Lessons learned from asci. *International Journal of High Performance Computing Applications*, 18(4):399–416, 2004.

[101] C. E. Powell and E. Ullmann. Preconditioning stochastic galerkin saddle point systems. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2813–2840, 2010.

[102] E. E. Prudencio, R. Byrd, and X.-C. Cai. Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems. *SIAM J. Sci. Comput.*, 27(4):1305–1328, 2005.

[103] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[104] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, 60:187–207, 1985.

[105] Rosner et al. ASCAC Nov 2010 Exascale Report.

[106] S. Sachs et al. Exascale programming challenges: Report of the 2011 workshop on exascale programming challenges marina del rey, july 27-29, 2011. http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/ProgrammingChallengesWorkshopReport.pdf, 2011.

[107] Sarkar et al. Synergistic Challenges in Data Intensive Science and Exascale Computing, March 2013.

[108] V. Sarkar et al. ExaScale Software Study: Software Challenges in Extreme Scale Systems. http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS_20report_20101909.pdf, September 14, 2009.

[109] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences*, 2013.

[110] F. Schmuck and R. Haskin. Gpfs: A shared-disk file system for large computing clusters. In *Proceedings of the First USENIX Conference on File and Storage Technologies*, pages 231–244, 2002.

[111] Schrom and et. al. A 100mhz 8-phase buck converter delivering 12a in 25mm2 using air-core inductors. In *APEC*, 2007.

[112] P. Schwan. Lustre: Building a file system for 1000-node clusters. In *Proceedings of the 2003 Linux Symposium*, volume 2003, 2003.

[113] J. Shalf, S. S. Dosanjh, and J. Morrison. Exascale computing technology challenges. In J. M. L. M. Palma, M. J. Daydé, O. Marques, and J. C. Lopes, editors, *VECPAR*, volume 6449 of *Lecture Notes in Computer Science*, pages 1–25. Springer, 2011.

[114] J. Shalf, D. Quinlan, and C. Janssen. Rethinking hardware-software codesign for exascale systems. *IEEE Computer*, 44(11):22–30, 2011.

[115] R. Thakur, W. Gropp, and E. Lusk. On implementing mpi-io portably and with high performance. In *Proceedings of the sixth workshop on I/O in parallel and distributed systems*, pages 23–32. ACM, 1999.

[116] P. L. Toint, D. Tomanos, and M. Weber-Mendona. A multilevel algorithm for solving the trust-region subproblem. *Optimization Methods and Software*, 24(2):299–311, 2009.

[117] The Top 500 List. http://www.top500.org, 2013.

[118] Trivelpiece et al. Scientific grand challenges: Exascale workshop panel report.

[119] S. K. Veerapaneni, A. Rahimian, G. Biros, and D. Zorin. A fast algorithm for simulating vesicle flows in three dimensions. *J. Comput. Phys.*, 230(14):5610–5634, June 2011.

[120] H. Wang, Y. Xia, K. Bergman, T. E. Ng, S. Sahu, and K. Sripanidkulchai. Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient *-cast connectivity. *SIGCOMM Comput. Commun. Rev.*, 43(3):52–58, July 2013.

[121] K. Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & Fluids*, 35(2):208–226, February 2006.

[122] D. Xiu and J. Hesthaven. High-order collocation methods for differential equations with random inputs. *Siam J Sci Comput*, 27:1118–1139, Jan 2005.

[123] D. Xiu and G. E. Karniadakis. Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. *Comput. Methods Appl. Mech. Engrg.*, 191(43):4927–4948, 2002.

[124] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole method in two and three dimensions. *J. Comput. Phys.*, 196(2):596–626, 2004.

[125] L. Ying, G. Biros, D. Zorin, and M. Langston. A new parallel kernel-independent fast multipole method. In *SC'2003 Conference CD*, Phoenix, AZ, Nov. 2003. IEE/ACM SIGARCH.

A Charge Letter

The next page contains the charge letter from Dr. Dehmer to ASCAC, which prompted the formation of this subcommittee, and the writing of this report.

Department of Energy
Office of Science
Washington, DC 20585

July 29, 2013

Professor Roscoe Giles, ASCAC Chair
Department of Electrical & Computer Engineering
Boston University
8 St. Mary's Street
Boston, MA 02215

Dear Professor Giles:

Thank you for the recent Advanced Scientific Computing Advisory Committee (ASCAC) report on the synergistic challenges in data-intensive science and exascale computing. The report was both informative and timely.

As we move forward with exascale computing during a time of significant budget restrictions, it is important that we focus our efforts on the principal research challenges required to develop a practical exascale computing system. We are at a point in our planning where input from the community would be enormously useful, but only if gathered quickly.

By this letter, I am charging the ASCAC to assemble a subcommittee to gather this information from the community. Specifically, we request a list of no more than ten technical approaches (hardware and software) that will enable the development of a system that achieves the Department's exascale goals, particularly the usability goals for the Department's mission-critical applications, as articulated in the attached presentation. That is, given the known technical barriers that could prevent the development of a computer that achieves the Department's exascale goals, what are credible technical approaches for overcoming these barriers? The subcommittee's report should provide compelling justifications for including each item in the list and describe the expected impact on overall system performance.

To inform our budget process, I would appreciate receiving the committee's preliminary comments by October 15, 2013 and a final report by November 30, 2013. I appreciate ASCAC's willingness to undertake this important assignment.

If you have any questions regarding this request, please contact either Barbara Helland, the acting Associate Director of the Office of Science for Advance Scientific Computing Research or Christine Chalk, the Designated Federal Official for the ASCAC.

Sincerely,

Patricia M. Dehmer

Patricia M. Dehmer
Acting Director, Office of Science