On The Path to Exascale

Ken Alvin*, Brian Barrett*, Ron Brightwell*, Sudip Dosanjh*, Al Geist°, Scott Hemmert*, Michael Heroux*, Doug Kothe°, Richard Murphy*, Jeff Nichols°, Ron Oldfield*, Arun Rodrigues*, Jeffrey S. Vetter°

**Sandia National Laboratories[1], °Oak Ridge National Laboratory, USA*

ABSTRACT

There is considerable interest in achieving a 1000 fold increase in supercomputing power in the next decade, but the challenges are formidable. This paper discusses some of the driving science and security applications that require exascale computing (a million, trillion operations per second). Key architectural challenges include power, memory, interconnection networks and resilience. The paper summarizes ongoing research aimed at overcoming these hurdles. Topics of interest are architecture aware and scalable algorithms, system simulation, 3D integration, new approaches to system-directed resilience and new benchmarks. Although significant progress is being made a broader international program is needed.

Keywords: Architectural Features, Architecture Types, Computer Architecture, Computer Size, Networking Technology, Processor Architecture, Supercomputers, Operating Systems.

INTRODUCTION

In 1997 Intel's ASCI Red broke the Teraflops barrier, achieving over 1 trillion floating point operations per second (Heermann, 1998). Last year both IBM's Roadrunner and Cray's Jaguar system surpassed 1 PetaFLOPS or 1,000 TeraFLOPS (Feldman, 2008). The next factor of 1000 improvement in supercomputing performance (Exascale) will be even more challenging. The primary driver for the architectural change underway is that clock speeds are increasingly constrained by power and cooling limits. All of the major chip manufactures are moving to multicore architectures, which results in the addition of hierarchical parallelism to supercomputers. Oak Ridge National Laboratory's Cray Jaguar system has 18,688 nodes, each comprised of two quad core AMD Opterons. Roadrunner is a one of a kind supercomputer composed of 6480 dual core AMD Opterons, each connected to two IBM PowerXCell 8i processors, which are similar to the processor used in Sony's Playstation 3 (Kahle, 2005). Both of these systems demonstrate that the transition to multicore is already a key design challenge for supercomputer architectures.

The memory wall is defined as the mismatch between CPU and memory performance (latency, chip I/O capabilities, etc.), and will continue to plague processor design. Today's

---

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

applications are primarily limited by data movement, represented by the statement "FLOPS are free" (Shiva, 2005). Power budgets continue to increase, which will result in power requirements in excess of 100 MW if existing petascale design methodologies are used for such a system. It is unlikely that such a system would be built unless power demands can be decreased significantly. Additionally, resilience will limit the availability of such systems. The current method for recovering from faults is checkpointing – at various times during a calculation a restart file is written to disk. Given weak scaling, I/O bandwidth requirements scale with the memory size and inversely with the mean time between interrupts (which decreases with increasing parts counts). Since exascale will require millions of nodes, the mean time between interrupts will decrease and checkpointing will become an impractical mechanism for fault recovery.

These challenges require new approaches to applications, algorithms, system software, and computer architecture, which has been noted by the numerous workshops and reports devoted to the problem (Kogge et al., 2008; Simon et al., 2007). The remainder of the paper discusses science and security applications that require exascale computing, architectural challenges, and ongoing work aimed at overcoming some of these obstacles.

APPLICATIONS

Scientific computing is essential to the advancement of numerous fields of study. Never before have we been able to accurately anticipate, analyze, and plan for complex events that have not occurred—from the operation of a reactor running at 100 million degrees to future changes in climate. Combined with the more traditional approaches of theory and experiment, scientific computing provides a profound tool for insight as we look at complex systems containing billions of components.

Science

Key areas of scientific research, including materials science, Earth science, energy assurance, fundamental science, biology and medicine, engineering design, and security can benefit from continued growth in high performance computing (to exascale and beyond). Table 1 summarizes scientific opportunities that can be enabled by exascale computing, key application areas, and the goals and associated benefits.

TABLE 1: THE PROMISE OF EXASCALE COMPUTING

Opportunity	Key application areas	Goal and benefit
Materials science	Nanoscale science; material lifecycles, response and failure; and manufacturing	Design, characterize, and manufacture materials, down to the nanoscale, tailored and optimized for specific applications
Earth science	Weather, carbon management, climate change mitigation and adaptation, environment	Understand the complex biogeochemical cycles that underpin global ecosystems and control the sustainability of life on Earth
Energy assurance	Fossil, fusion, combustion, nuclear fuel cycle, chemical catalysis, renewables (wind, solar, hydro), bioenergy, energy efficiency, energy storage and transmission, transportation, buildings	Attain, without costly disruption, the energy required for economically viable, and environmentally benign ways to satisfy residential, commercial, and transportation requirements
Fundamental science	High energy physics, nuclear physics, astrophysics, accelerator physics	Decipher and comprehend the core laws governing the universe and unravel its origins
Biology and	Proteomics, drug design, systems biology	Understand connections from individual proteins through whole cells into ecosystems and

medicine		environments
National security	Disaster management, homeland security, defense systems, public policy	Analyze, design, stress-test, and optimize critical systems such as communications, homeland security, and defense systems; understand and uncover human behavioral systems underlying asymmetric operation environments
Engineering design	Industrial and manufacturing processes	Design, deploy, and operate safe and economical structures, machines, processes, and systems with reduced concept-to-deployment time

The system and application-wide advances required to reach exascale are not inevitable, and require a fundamental rethinking across all aspects of High Performance Computing (HPC).

Material Science

Materials science drivers, objectives, and impacts that are enabled by Exascale leadership platforms have been identified in Table 2 (Department of Energy, 2007).

TABLE 2: SELECT MATERIALS SCIENCE DRIVERS FOR LEADERSHIP APPLICATOINS AT EXASCALE

Application area	Science driver	Science objective	Impact
Nanoscale science	First principles design of increasingly complex materials with specific, targeted properties	Understand and use isolated nanostructures to design materials made out of nano-building blocks	Smart materials for nanoelectronics, photo voltaics, information technology, and medicine
	Predictive description of microscopic behavior of water to understand systems in aqueous environments	Perform molecular dynamics with forces found with Quantum Monte Carlo computations	Detailed understanding of the structure of water—fundamental understanding of biological systems.
	Understand synthesis of alloy nanoparticles with potential impact for design of new catalysts	Define the thermodynamics of compositions of alloy nanoparticles	Magnetic data storage Economically viable ethanol production Energy storage via structural transitions in nanoparticles
	Physics of strongly correlated electron materials	Explain the fundamental mechanism of high-temperature superconductivity, including materials specificity and inhomogeneities	New materials for practical applications in oxide electronics and next-generation power transmission

Earth Science

Earth science and climate change research will focus on two principal activities in the decade ahead:

- **Mitigation**: Evaluating strategies and informing policy decisions for climate stabilization; and
- **Adaptation**: Preparing for committed climate change with decadal forecasts and region impacts.

Simulations of 100–1,000 years will be typical for mitigation activities, while shorter simulations of 10–100 years will be used for adaptation. Each set of simulations must be

predictive and quantifiable in order to reliably inform policy makers. The requirements for computing can be tied to these activities and goals, as shown in Table 1. For example, estimates call for compute factors 10^{10}–10^{12} greater than those available today to meet goals for spatial resolution, model completeness, simulation times, and breadth and depth of ensembles and scenarios.

Climate models are currently more reliable at short times scales and long, asymptotic scales. Since many of the questions to be answered are targeted in the 20–50 year range, the ability of models to provide reliable forecasts will be challenged.

The Earth Science community has also done well in articulating what it believes to be attainable biogeochemical objectives over the next decade (Department of Energy, 2007):

- Integrated models and measurements of biogeochemical cycles;
- Development of next-generation ecological models; and
- Better theory for and quantification of uncertainty.

Climate science opportunities at the exascale are abundant (Hack and Bierly, 2007):

- Decadal prediction on regional scales (accuracy in global models);
- Climate extremes (heat waves, drought, floods, synoptic events, etc.);
- Climate variability (low-frequency variability);
- Water cycle (particularly in the tropics);
- Human-induced impacts on carbon cycle;
- Sea-level rise (melting of the Greenland and Antarctic ice sheets); and
- Abrupt climate change.

The rate limiters above are decadal prediction, abrupt climate change, and climate variability.

Security

Nuclear weapons provide an application driver for high performance computing and advanced simulation in the security. The complexity of nuclear weapons design, certification and assessment requires a combination of the most advanced computational and experimental science, even during the era of underground nuclear testing. Presently, under the nuclear testing moratorium and the U.S. Stockpile Stewardship Program (SSP), advanced computational simulation has come to play a significant and foundational role in stewardship of the U.S. nuclear stockpile, as led by the Department of Energy's Advanced Simulation and Computing Program (ASC)

The most significant application needs for Exascale simulation are the assessment and certification of nuclear explosive package performance, full system safety, and weapon survivability in nuclear environments. These simulations involve highly coupled, nonlinear multi-physics with three-dimensional features, multi-scale phenomena, and inherent variability in materials and manufacturing processes. By the end of the first decade of ASC, it was possible to move from two-dimensional to three-dimensional nuclear performance simulations with "standard" mesh resolutions and calibrated physics. This required achieving the goal of 100 tera-flops of integrated hardware and software capability. It is now estimated that, to replace existing ad-hoc physics models, we must increase average mesh spacing in each spatial dimension by an order of magnitude, incorporate more sophisticated, multi-scale model-based physics, and quantify uncertainties in those physical models. The increase in spatial resolution increases computational needs by at least a factor of 1000, while improved physics models are estimated to increase computational needs by an additional factor of 100. Finally, the costs of estimating uncertainties can be expected to results in suites of computations, ranging from 100s to 1000s of realizations of the uncertain system parameters. This final class of computational

demand may not directly multiply the previous contributions, with the development of more sophisticated sampling and model reduction techniques. However, even conservative estimates of needed model evaluations increases computational needs by a least an additional order of magnitude.

Additionally, we see emerging application areas in Informatics, where the goal is to examine very large data sets (from direct observation or the output of simulations) to ask new questions, form hypotheses, or generate new understanding. These emerging Informatics applications show significantly more difficult data movement properties than do traditional 3D Physics applications, and represent a critical challenge for the Exascale era.

ARCHITECTURAL CHALLENGES

The architectural challenges for reaching exascale are dominated by power, memory, interconnection networks, and resilience. There are also significant software challenges including the need for new programming models, latency and bandwidth tolerant techniques, fault-tolerant methods, and algorithms that are scalable to millions and perhaps billions of threads (see the discussion of architecture aware algorithms under ongoing research).

Power

Power is the dominant constraint for exascale computing. Already, current installations consume tens of megawatts. Many large organizations have relocated their datacenters to areas providing cheap electricity. Though shrinking feature size helps to alleviate these requirements, high-end HPC machines have outpaced Moore's law, resulting in an upward trend in power requirements. The sheer scale of a machine performing one exa-operation per second (i.e. 10^{18} operations/second) means that if individual operations each require a single picojoule (1 pJ=10^{-12} J) of energy, a total of one megawatt of power will be required.

Extrapolating current power consumption trends into the Exascale timeframe (2018 for the purposes of this article) yields startling trends. It is estimated that in the 2018 time frame a single floating point operation will require 10 pJ of energy (Kogge et al., 2008). This assumes the operands are already present at the floating point unit and does not include moving the data to or from a register file. Using the Cacti 5.1 tool (Thoziyoor et al., 2008), and extrapolating to the 18nm technology node, a 32K L1 cache would consume 31 pJ and a 512K L2 96 pJ per access. In addition to floating point and cache access, a processor core must also perform instruction fetch and decode, integer operations for bookkeeping and data management, and drive its control path. This can require substantial power. A MIPS64 5Kc processor, designed for low power consumption still consumes 490 pJ per cycle at 90nm (MIPS Technologies, Inc., 2007). A 32-bit ARM11 processor consumes 180 pJ/bit at 130nm. Adjusting to 64bit and assuming a linear extrapolation, these processor cores would require 50-90pJ/cycle in an 18nm process.

Based on HPC workload analysis (Murphy et al., 2009) and the above estimates, we consider two possible cases of instruction mix and cache performance to calculate processor power consumption (see Table 3) to be 118-125 pJ per average operation. For an Exa-op machine, this would become 118-125 MW.

TABLE 3: HPC WORKLOAD CHARACTERISTICS AND PROCESSOR POWER

% FP instructions	% Load/Store	L1 Hit Rate	Energy/op
10%	50%	90%	69.8 pJ
5%	60%	80%	116.9 pJ

DRAM requirements for an Exascale machine will also be large. In 2018 it is estimated that DRAM density will be 16GB/chip (ITRS 2007). 300 Petabytes of DRAM storage would require 18 to 21 million chips depending on ECC. If we assume DRAM chip power to be at 260-450mW – based on current chips (Micron technology, 2007) and assuming a drop to 1.0V – an Exascale memory system would require 5.5-9.5MW. Again, this assumes an extrapolation of current technologies, which will likely not provide sufficient bandwidth to future massively multicore processors.

Network requirements are discussed in more detail as part of the discussion of Architectural Challenges. If we assume 8,192 to 32,768 routers, each capable of routing 77.8 Tb/sec and 128K-512K 1.2Tb/sec network endpoints, we have a unidirectional system bandwidth of 790.9 to 3,163.8 Pb/sec. Current long distance electrical interconnect can operate at 30 pJ/bit (bidirectional)(Kogge et al., 2008). It should be quite feasible to reach 10 pJ/bit by 2018, resulting in a total network power of 4.0-15.6MW

TABLE 4: TOTAL SYSTEM POWER REQUIREMENTS

	Low (MW)	High (MW)
Processor	69.8	116.9
DRAM Memory	5.5	9.5
Network	4.0	15.6
Total	**79.3**	**142.0**

This results in a total system power of 79.3 to 142.0 MW (See Table 4). It should be noted that this does not include archival storage, cooling, a RAS monitoring system, the internal router power and many other necessary system components.

From 2007 to 1987 commercial power rates ranged from 9.2-12.0 cents per KWH in real 2008 dollars (DOE, 2007). Thus, one Watt-year costs $0.80 to $1.05, or one megawatt-year is $800,000 to $1,050,000. This would mean that a conventionally constructed exascale machine would cost between $63.4 million and $149.1 million a year simply to keep powered. Again, this ignores cooling, power conversion, and other probably inefficiencies. Clearly, power will become a limiting factor in designing an exascale machine.

Memory

Data movement has been the dominant performance bottleneck for all electronic computer systems since their invention in the mid 1940s. Today's HPC systems are no different (Murphy, 2007; Murphy and Kogge, 2007). As discussed in the Power and Packaging Section, data movement, particularly "far" remote accesses will dominate the power requirements for all levels of the memory hierarchy. Within a node, analysis of applications shows that even for scientific codes, most of the instructions executed are not floating point, but memory and integer instructions, and that most of the integer instructions are computing memory addresses (See Figure 1). Furthermore, today's dominant MPP architecture does a poor job of exposing the memory hierarchy beyond a single node to fine-grain data access. While Partitioned Global Address Space (PGAS) mechanisms in software and hardware may exist in modern supercomputers, these mechanisms provide very little capability to do anything other than copy data throughout the system. Worse, most node-to-node copies must be performed in a coarse grained fashion, creating bursty communication patterns and forcing the programmer to explicitly manage copies (even in a PGAS environment).

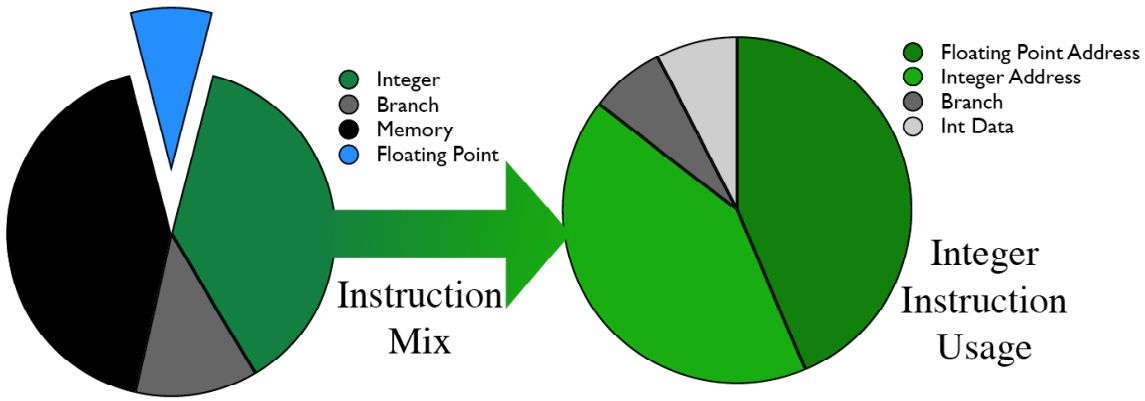

Figure 1: Instruction characterization and use of a portfolio of Department of Energy Applications

Fundamental technology imbalances between processor and memory systems, combined with large memory footprints required by many applications typically produce low Instruction Per Cycle (IPC) measures on modern processors, often significantly less than one. This inefficiency is primarily dominated by a lack of concurrency and overabundance of latency in modern memory systems (Murphy, 2007). Creating a more balanced system is one of the key challenges in reducing power to enable exascale machines.

Finally, while mechanisms for synchronization, atomic operations, and transactions may exist in small-scale cache coherent environments, there are no lightweight analogues to these mechanisms available on MPPs. Performing operation at a distance, minimizing unnecessary data copies, and enabling fine-grained operation will be required to optimize both power and performance.

Figure 2 summarizes the results of Murphy and Kogge (2007), and measures three key data movement and memory access pattern properties for applications: the Spatial Locality (or use of data "near" data already used in memory), the Temporal Locality (or resuse of already used data), and unique data set size normalized over an instruction interval (represented by the relative size of the points on the graph). The application space can be thought of as consisting of two classes: first, Physics Applications, which are the core of traditional HPC; and second, Informatics Applications that represent increasingly important new codes. Whereas Physics applications are generally simulations, have a 3D spatial decomposition, and are computing floating point results, the Informatics applications tend to be unstructured, integer oriented, and are used to form hypotheses from large data sets (either sensor data or the results of simulations).

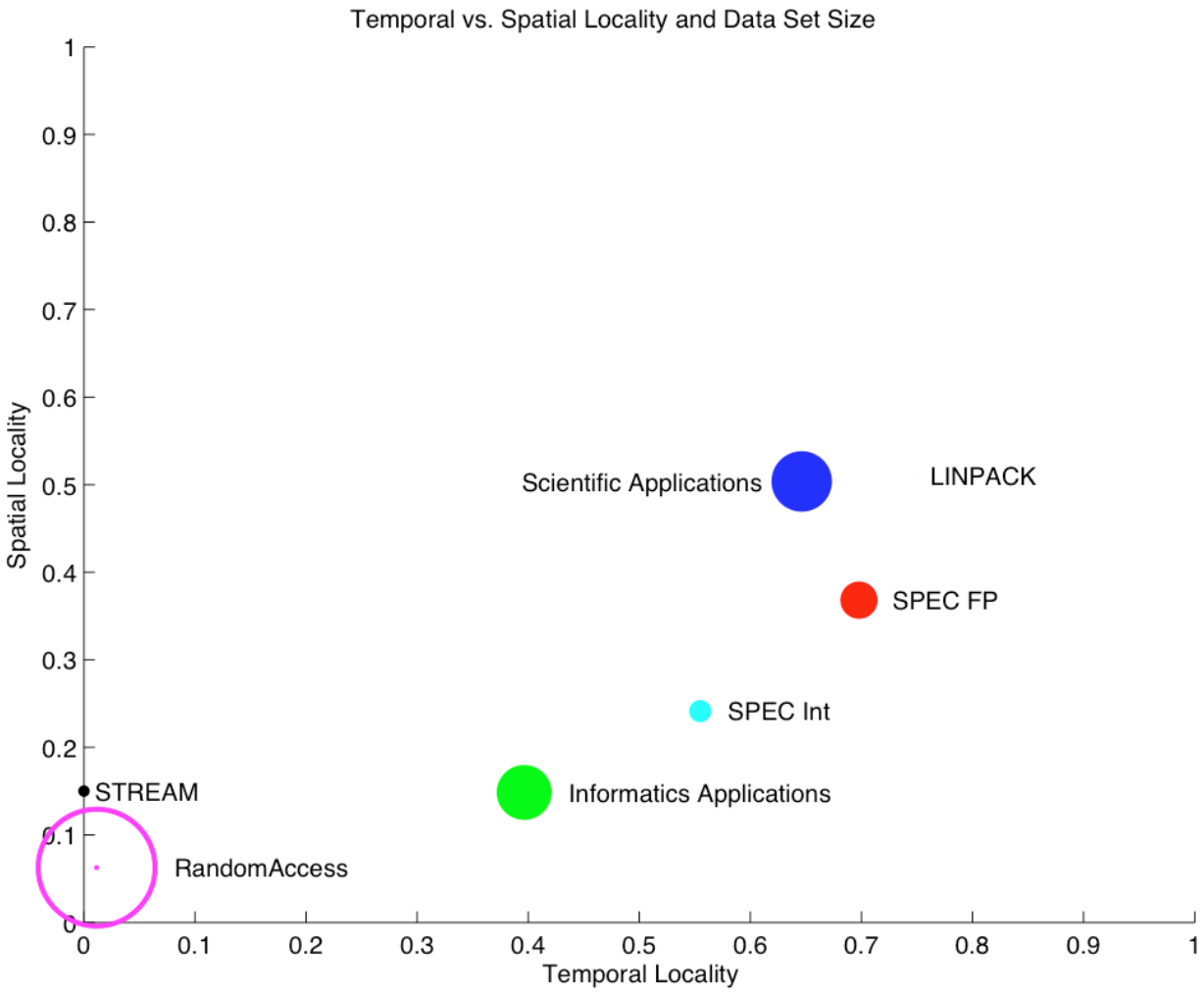

Figure 2: Application Temporal Locality, Spatial Locality, and Data Set Size (adapted from Murphy & Kogge (2007))

There are two key points that increase the challenges of data movement on an exascale system. First, the Informatics applications are much more data movement oriented than the Physics applications. For a comparable data set size, they exhibit only 61% of the temporal locality and 30% of the spatial locality. And second, neither application set is well represented by the benchmark suites used to tune the performance of machines. LINPACK has an extremely small data set and much more temporal locality than the typical Physics code it is meant to represent (and looks nothing like an informatics code), and neither SPEC benchmark suite (Int or FP) is representative of anything, despite the fact that SPEC typically dominates CPU design optimizations. An environment where the critical measures of success are demonstrably unrepresentative, results inevitably in additional power consumption that we can no longer afford.

Interconnection Networks

One of the critical challenges for Exascale computing will be providing sufficient interconnect performance to facilitate application performance and scaling. Traditionally, three primary metrics have been used to characterize interconnection networks: bandwidth, latency and

message rate. All of these metrics have seen slowing performance gains in the last several years, and, going forward, latency is unlikely to improve much given the physical limitations (i.e. speed of light); however, this trend must be reversed for bandwidth and message rate if we are to see well balanced exascale systems. It will also be vital for networks to provide mechanisms for overlapping communication with computation, as this tends to reduce the bandwidth requirements for applications that take advantage of it. This implies that providing independent progress and high message rate in the network hardware and software will become increasingly important. However, in the exascale timeframe, two other characteristics will jump to the forefront: power and resilience. The main challenge of exascale networks will be in providing sufficient interconnect performance while maintaining reasonable power and resilience.

To illustrate the power problem with respect to bandwidth, consider an Exascale machine that maintains the system balance found in the Cray XT4 system. The XT4 configuration considered here has a network byte per FLOP ratio of 0.25 and a memory bandwidth to network bandwidth ratio of 3:1. We assume that both of these ratios are maintained, but believe that the memory to network bandwidth ratio is generally more important, as it typically drives on-node performance for most of our applications. The analysis is topology agnostic, but assumes a direct network where 1/4 of the switch bandwidth is used for host connections, while the remaining 3/4 is used to build the network topology. This ratio is consistent with those used in recently developed network topologies, such as dragonfly (Kim, 2008) and flattened butterfly (Kim, 2007).

If these requirements are projected forward to the Exascale, then total bidirectional node bandwidth becomes 250 PB/s. It is generally felt that the use of direct to package optics and WDM (wave division multiplexing) will readily lead to 10 pJ (Kogge, 2008) per bit transmission energy requirements[2], and a similar amount of energy required to move the data through the internal switch logic. This leads to a total power requirement for this example Exascale interconnect of 100 MW (including NIC and switch power) – which is an unreasonable number for even the largest, contemporary computer data center.

There are two approaches to reducing the interconnect power requirements: reduce the application bandwidth requirements and/or reduce the energy per bit transferred. We believe pursuing both of these opportunities will be crucial. First, we note that there are ways to improve the efficiency with which the network performance can be utilized. For example, our applications have generally run slower on a per core basis on a Cray XT5 system than on an XT4 system. The difference between these two systems is that the FLOP rate and memory bandwidth of an XT5 node is roughly twice that of an XT4, while the network performance remained constant. However, we believe much of the application performance could be recovered by facilitating higher message rate and true independent progress in the network. These features would allow the applications to be modified to overlap computation and communication. Enabling these capabilities will require research into both NIC architecture and the network stack software. Enabling computation/communication overlap could allow us to reduce the bandwidth requirements by a factor of 2 or more, depending on the application. However, the cost may be the requirement for more memory bandwidth to provide enough memory performance to complete both activities simultaneously. A factor of 2 reduction would bring the network power requirements to 50MW and further reductions may be possible based on application

---

[2] It should be noted that WDM increases the energy requirements per bit, but will be necessary to keep the cable count to a reasonable level.

requirements. Part of the challenge of exascale will be in understanding the application requirements so that the system can be properly balanced without wasting energy.

Further reductions can be found in fundamental device technologies. It has been suggested (Kogge, 2008) that data transmission energy could be reduced to 2 pJ/bit. However, substantial research investment will need to be made to reach this goal. A similar research effort will be required to reduce the energy consumption of the core switch, which could include fundamental circuit research and/or optical switching technology. A 5X power reduction in both of these areas would lower the interconnect power requirements to 10MW, which is finally in a reasonable range.

We feel the primary challenge to building reasonably balanced exascale systems will be power consumption, though reliability will also be an important area to consider. Current trends will not lead us to viable interconnect options and substantial investment in various research areas will be necessary. These include, but are not limited to: fundamental device technologies, particularly in the field of silicon photonics; mechanisms to provide high message throughput and true independent progress; applications; and network topologies to take advantage of advances in these areas.

Resilience

As massively parallel processing (MPP) systems continue to grow in size, complexity, and component count, the ability of applications to endure failures and make effective use of increasing computational power is rapidly decreasing. The average time that the largest platforms can run without incurring a failure fatal to a full-system application was once measured in weeks. That time is now being measured in hours, and, barring any change to current practices, it will soon be measured in minutes. A recent study led by Sandia concluded that, as systems grow beyond 100 thousand components, a combination of factors lead to a situation where more than half of the available computational resources will be wasted (Oldfield, 2006). Figure 3 shows the minimum checkpoint overhead as a percentage of the overall execution time. This calculation is overly optimistic because it assumes data transfers at

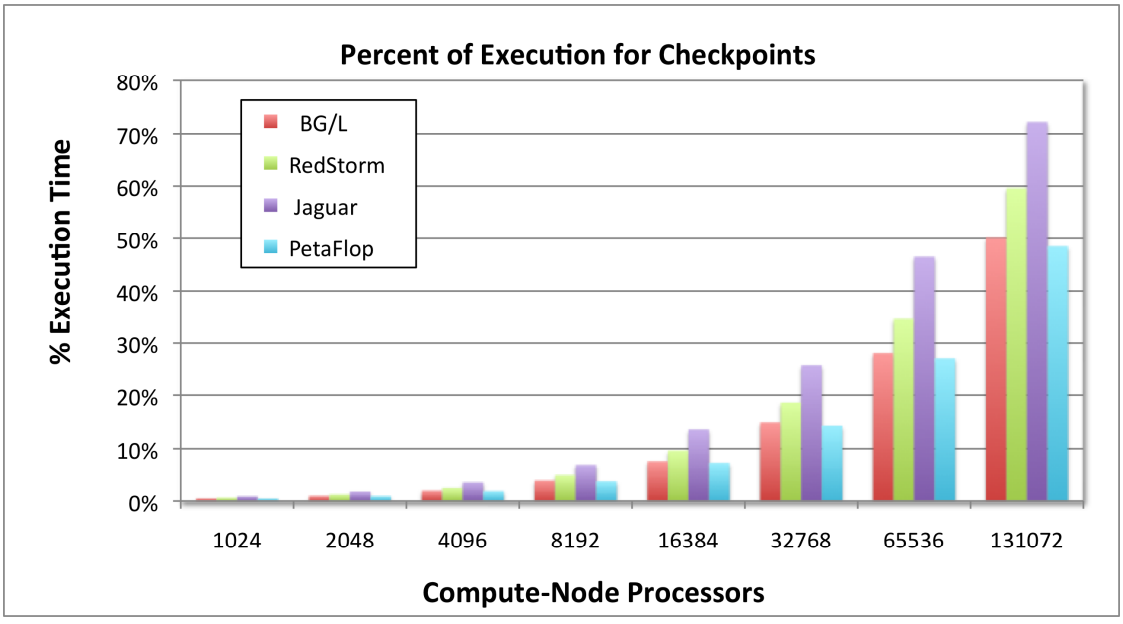

Figure 3: Approximation of checkpoint overheads for HPC systems.

theoretical network and storage rates, and that the application writes a checkpoint at the "optimal" checkpoint period (Daly 2006). In practice, these overheads will be worse. In the remainder of this section, we discuss the issues related to application resilience for exascale systems and outline potential solutions.

In some cases faults are automatically detected and corrected by hardware and system software. For uncorrectable errors, resilience on Massively Parallel Processing (MPP) systems has traditionally been the responsibility of the application, with the primary tool being application-directed checkpoints to secondary storage. There are several reasons that this approach will not be sufficient in the near future.

For nearly all systems, secondary storage comes in the form of disk system that is accessible by each computing element, typically through a parallel file system. Writing checkpoint data from tens of thousands of processes in a parallel application simultaneously places great strain on the parallel file system in terms of both performance and reliability. The parallel file system and the underlying disk components are arguably the most fragile pieces of software and hardware in any system. Depending on the most unreliable part of a system to ensure application resilience is disconcerting. Currently, the only method of increasing I/O performance is to add more disks, which in turn increases the likelihood of a disk failure. As the amount of checkpoint data increases with system size, disk-based parallel file systems will not be sufficient.

In addition, the hardware components from which systems are composed are becoming more fragile. In an effort to continue to increase performance and lower energy costs, hardware components are becoming more susceptible and sensitive to uncorrectable errors. For example, reducing the feature size of an integrated circuit to increase the number of processing elements per area also increases the probability of errors stemming from naturally occurring radiation sources, such as cosmic rays. While the probability of such errors can be extremely low for commodity components for mass consumer markets, the massive scale at which these parts are used in large parallel computing systems increases the probability of error to a statistical certainty. Current methods of using MPP systems are not sufficient to deal with an environment where there is at least one failed component throughout the lifetime of the application.

ONGOING RESEARCH

This section summarizes ongoing research within the DOE Institute for Advanced Architecture aimed at overcoming some of the challenges discussed in the previous section. A key strategy is to co-evolve architectures and algorithms as we strive to reach exascale computing. Key facets of this work are the development of:

1. Architectural-aware and highly scalable algorithms;
2. Simulation tools to quantify the impact of architectural changes on algorithms and applications, to guide the development of future supercomputers, and to identify algorithm bottlenecks and hence aid in the development of new methods;
3. 3D-Packaging techniques to reduce power consumption and increase memory bandwidth;
4. New approaches to enable resilience; and
5. New benchmarks that better characterize HPC applications.

Each of these is discussed in turn in the following sections.

Architecture Aware Algorithms and Scalability

For many years, science and engineering application developers have enjoyed a stable programming environment in the context of standard programming languages and message passing on a nearly homogeneous network of serial processors. However, the path to exascale

computing poses several new challenges to application developers. Exascale performance will require the use of multicore nodes such that MPI-only will not be sufficient. Furthermore, optimal node performance will be challenged by reduced memory bandwidth per core, the need for SIMD expressions and underdeveloped programming environments.

Although there is much that hardware and system software developers can do to aid application performance on new systems, more than ever before scalable algorithms and implementations will need to be architecture-aware and developed for scalability. It is especially important that algorithms be designed and implemented with knowledge of the node architecture, and that new modeling and analysis methods such as time-parallel methods, advanced mathematical optimization and uncertainty quantification be cultivated to expose additional parallelism.

Parallel Programming Transformation

The first and foremost challenge to optimal use of Exascale computers is the required transformation of parallel programming strategies. There is mounting evidence that optimal parallel applications for scalable multicore computer systems will rely on MPI for inter-node parallelism, but will need to introduce large-volume functional parallelism and SIMD vectorization to effectively use the multicore node. Vectorization is the job of the compiler, with a limited help from the programmer via pragmas and directives, so the real issue is that presently there is no obvious parallel programming model for implementing the middle layer of parallelism. Current standards such as OpenMP, Pthreads and UPC are not designed for multicore nodes. CUDA, RapidMind and related products target multicore nodes but are proprietary. OpenCL is an emerging standard but is not really a user-oriented interface, and will likely not provide optimal performance (e.g., in comparison to CUDA on GPUs).

However, even without an emerging programming model for multicore, there is a vast amount of work required to prepare existing applications for multicore nodes. Two major tasks are reducing bandwidth requirements as much as possible, primarily by introducing the use of mixed precision, storing data in 32-bit arrays wherever possible; and rewriting low-level kernels as stateless functions with large enough granularity to keep a SIMD core busy, and small enough that there is a large volume of simultaneous function calls to execute.

Beyond the Forward Problem

In many areas of science and engineering, solving a single problem with given input conditions, often called the forward problem, is sufficiently challenging, and higher forward problem fidelity is the highest priority for scalable computing. However, as the fidelity of the forward problem becomes sufficiently good, it becomes possible and imperative to study parameter sensitivities, quantify uncertainties and automatically compute an optimal solution over a range of parameter values.

All of these advanced modeling and simulation techniques quickly increase problem size and parallelism—often by orders of magnitude—and large problems can easily exceed the computing capacity of our largest systems. The simplest of these approaches are “black box” in nature and do not require a true Peta/Exascale system (instead requiring a cluster of Tera/Petascale systems). However, more advanced methods (often called embedded methods) rely on a tightly coupled aggregation of forward problems and require a true peta/exascale system. The challenge with embedded methods is that they require the transformation of an application into a “subroutine” because embedded methods need to call the forward solve as a function. Most applications were not designed with this mindset, so this transformation will be challenging.

Furthermore, many of these approaches assume a smoothly varying nonlinear function, which is often not the case in practice. Some functions are inherently non-smooth. Others are implemented in such a way that function evaluations involve table lookups, or ad hoc evaluation techniques. Such functions can often be rewritten to improve smoothness.

An additional dimension of potential parallelism is in time. Traditionally, we have restricted our parallelism to spatial dimensions, but there are promising new algorithms that can extract parallelism by considering multiple time steps simultaneously. Such approaches can greatly improve parallel execution, especially in situations where the spatial resolution cannot be practically increased, either due to stability constraints or sufficient spatial resolution.

Robust multi-precision algorithms

Floating point computation has always been faster for single precision (SP) data and computation than for double precision (DP). However, presently, SP computations are even more attractive because bandwidth-intensive calculations can severely limit effective core use on a multicore node. Figure 4 shows the impact of SP vs. DP for an implicit finite element miniapplications called MiniFE on the Intel Nehalem and AMD Barcelona processors. Use of SP is not only faster than DP on a single core, but also allows much more effective use of additional cores.

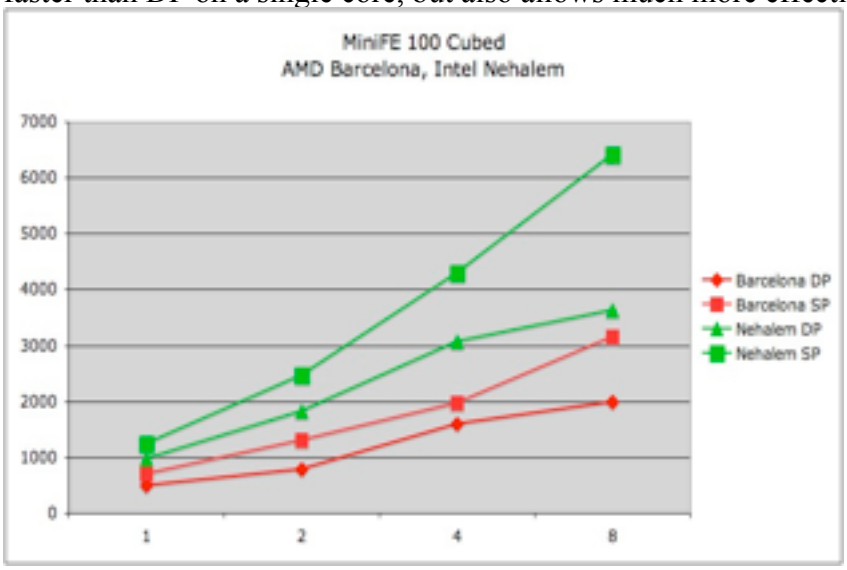

Figure 4: Single precision (SP) vs. Double precision (DP) performance for a finite element miniapplications. Nehalem performance from 4 to 8 cores goes up by 50% for SP, 15% for DP. SP is nearly twice as fast as DP at 8 cores.

Using preconditioned iterative methods as an example, new basic kernels such as preconditioners for multi-core processors must support multiple scalar data types, including single and double precision. These kernels will enable the use of mixed precision algorithms and multi-core processors. They will provide the foundation for optimal preconditioners on scalable multi-core systems. In addition, we will need production quality metrics of condition estimates, and accuracy and precision estimates that can help determine the required precision for storing a given data object and the required precision for a given computational step.

Simulation

To reach the goal of exascale, it will be necessary to make substantial leaps in a number of technologies and architectures. Unfortunately, it will not always be practical to construct prototype systems of sufficient size to fully evaluate the impact of these new technologies at

scale. Therefore we will have to rely on simulation to guide many design decisions. Currently, the architecture community lacks the tools needed for such evaluations.

To meet this need, we are constructing a simulation environment for simulating large-scale HPC systems. The Structural Simulation Toolkit (SST) will allow parallel simulation of machines at multiple levels of detail (from cycle-accurate instruction-based to message-driven simulation). It will incorporate models for processors, memory, and network subsystems. Some key features:

- Scalable Parallel Simulation: The simulation framework allows large parallel simulations of even larger parallel machines. This will allow us to use the supercomputers of today to design the supercomputers of tomorrow. Efficient parallel simulation requires built-in support for automatic partitioning, checkpointing, and event serialization.
- Multiscale: Different simulation models allow either abstract or detailed evaluation of system components. This will allow different system characteristics to be evaluated at the necessary level of detail and accuracy, while still allowing other parts of the simulation to be performed in a faster but more abstract manner.
- Holistic: Raw performance is only one of several challenges for an exascale system. An exascale simulator should provide a unified interface to a variety of technology models, allowing components to easily estimate power, energy, area, cost, and reliability.

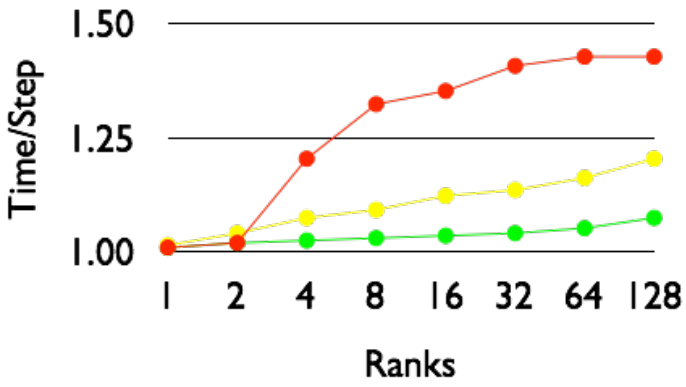

Figure 5: Preliminary weak scaling performance of parallel DES Algortihm

Currently, Sandia has completed a serial implementation of the SST. This version has been used to explore interconnect (Hemmert, et all, 2007) and perform application analysis (Rupnow, et all, 2006). Currently, Sandia is expanding the serial SST into a parallel version. Already, we have constructed a parallel core that demonstrates many of these features and have integrated basic processor and network models and a detailed memory models (Jacob, 2009) into the framework. Preliminary scaling studies of the (See Figure 5) conservative distance-based parallel discrete event simulation algorithm under a variety of levels of detail indicate good scaling characteristics. A consortium of other academic and laboratory partners has formed to continue to improve this core and add new components. We are actively soliciting input from potential users and partners on the structure and requirements of this simulation toolkit.

3D Packaging

The power analysis in presented above assumes a machine constructed along highly conventional lines. To efficiently reach exascale, it will be necessary to harness a number of unconventional

and emerging technologies. In particular, advanced 3D packaging may provide a solution to the power dilemma.

3D packaging involves integrating IC chips with vertical connections extending through the substrate of one chip to connect the next. These connections through the silicon, or Through-Silicon Vias (TSVs) allow information to be moved from one chip to another with very little power – 1-11 fJ/bit (Kogge et al., 2008). TSVs may be used to allow close connections between a processor, DRAM, and network chips.

Currently, a DDR3 DRAM chip consumes 400mW-450mW, with about 180-200 mW of that used for communication across the memory bus to the processor. A stacked DRAM would only require about 10 fJ/bit to drive a Through-Silicon Vias, or only a single mW to reach a similar bandwidth as the DDR3. This would save 40-50% of the DRAM's power. Additionally, with more signal lines to access the DRAM it will be possible to reduce the number of bits charged in each DRAM activation from the 8K bits per device in current DRAMs to only the 512 needed to fill a cacheline. This could save another 15-20%. Total savings would then be on the order of 55% to 70%, or $2.4M to $7.0M a year.

The processor could also benefit from close integration with memory. Currently, I/O across the memory bus can account for 10-15% of a processor's energy (Laudon, 2007). This could be reduced substantially with the use of TSVs instead of power-hungry bus-based communication. Additionally, the proximity of stacked memory could make it possible to remove the L2 cache from processors, reducing the size of the clock tree and saving an additional 10-20%. Total savings of 20-35% would translate into $11.2 to $43.0M a year.

Lastly, the network could be improved by enabling silicon photonic interconnect. By using optical instead of electrical signaling, silicon photonic networks have the potential to reduce network power requirements to 0.2 to 1.5 pJ/bit (Kogge et al., 2008; Watts, 2009). However, to function effectively, they require extremely low capacitance connections between the optical devices and the communicating processor. 3D TSV integration could provide the necessary connections. This would reduce the network power to .01 to 2.34 MW a savings of $1.3 to $16.4M per year.

Additionally, 3D packaging may reduce the cost of systems by allowing more efficient integration of silicon pieces fabricated in heterogeneous processes and by amortizing packaging costs.

TABLE 5: ESTIMATED DIE PRODUCTION COSTS

Device	Date	Size (mm^2)	Good Die Cost ($)	G.D. Cost/mm^2	Packaging Cost
DRAM 1Gb	June 17	45-90	1.07-1.63	$.01-$.03	18%
Core Duo 45nm	09Q1	143	$13.50	$.094	46%
Nehalem 45nm	09Q1	263	$36.80	$.140	47%
Barcelona 65nm	07Q4	285	$41.92	$.147	54%

Using an IC Cost estimation tool (ICKnowledge LLC, 2009) and current prices for DRAM (DRAM Exchange, 2009), it is possible to estimate the production cost for known good die for various devices (See Table 5). From this it can be seen that not all ICs are equal. The cost of a DRAM die may be as low a penny per mm^2, while a processor may be over ten times this, due to

the extra layers of metal and more exotic materials and processes required. While it is beneficial to integrate processing and memory in close proximity, doing this on the same die would either be wasteful (running a simple DRAM part through a processor fab line) or detrimental to performance (implementing a processor in a DRAM fab process). With 3D integration, it is possible to choose the fabrication line which best suits each intended device and still provide tight integration with other dissimilar devices.

Package and post-package test of processors can account for 50% or more of their total gross cost. The cost per pin (including test, packaging yield, redistribution layers, etc…) can be 2-4 cents. Additionally, each pin requires more that complexity be added to the socket, and a new trace on the board. Though 3D integration introduces new difficulties and cost into the cost packaging stage it may reduce overall cost by amortizing the cost of a package over several chips and by replacing expensive pins with inexpensive VIAs.

In summary, advanced packaging will have to be a critical part of any exascale strategy. The prohibitive cost of powering an exascale system can be reduced to more manageable levels ($48.5M to $82M/year) by allowing tighter integration of processor, memory, and interconnect.

Approaches to Resilience

In order to reduce the amount of data that needs to be preserved and managed, we are developing runtime support for high-reliability data storage and retrieval. By developing runtime system capabilities that provide user-requested high-reliability for a given data object, algorithm and application developers can make known to the system the critical subset of data needed by a computation.

In addition, we are exploring the ability for a user to declare a computational scope that needs to be free from soft data errors and/or computation errors. This approach allows the programmer to explicitly communicate the performance/reliability tradeoff to the system. These mechanisms would be used in the development of resilient iterative Krylov methods. By using flexible variants of these methods, and careful formulations of orthogonalization steps, we believe that we can still have reliable iterative results with only a fraction of the total data and computation.

From a system software perspective, we are investigating system-directed checkpointing strategies and transparent redundant computation. The goal of system-directed checkpointing is to provide efficient, application-transparent resilience through coordinated use of system resources. To efficiently quiesce and checkpoint a large-scale application not only requires cooperation among the individual processes, it also requires integration and cooperation with shared services like the network, scheduler, and storage system. This approach will have to deal with messages in transit, in-progress file system operations, and interactions with various other shared services. We expect to leverage the simplicity of a lightweight kernel environment and an integrated RAS system to reduce the complexity and increase the performance of extracting application state. For transparent redundant computation, we are providing the ability for an MPI application to run extra processes on a set of redundant nodes and automatically switch to using these extra processes should a failure be encountered. Our current approach is to provide this capability entirely at the application layer using MPI, in order to be as portable as possible.

We are also actively engaged in real-time statistical analysis of monitored systems (Brandt 2009) as well as detailed analysis of system log files (Oliner, 2007). This work provides a critical foundation for understanding the root cause of failures and could ultimately lead to mathematical models that enable prediction for certain types of system faults. For example, Figure 6 shows active memory normalized to the total system memory for a single node of one of Sandia's Tri-Lab Linux Capacity Clusters (TLCC) systems. Automated monitoring systems,

along with statistical models help identify anomalies that identify future faults. In this case, the anomaly identified in Figure 6 signals an impending memory fault.

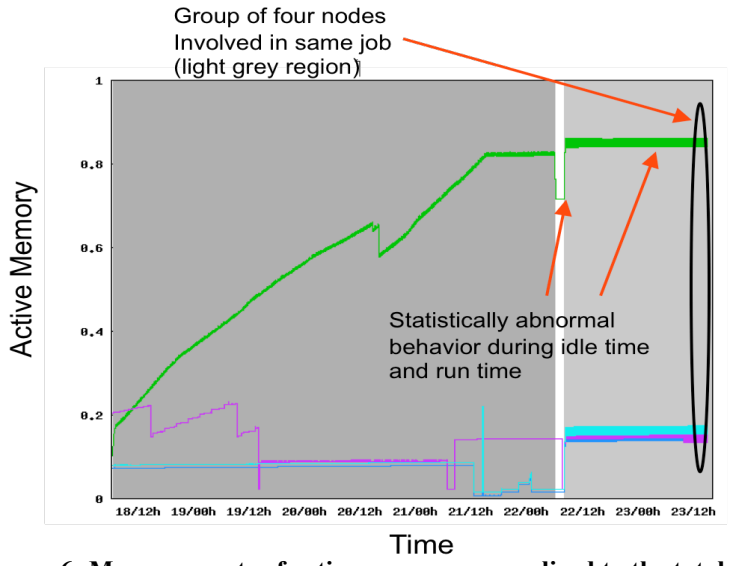

Figure 6: Measurements of active memory normalized to the total system memory help identify abnormal behavior that ultimately leads to a memory fault (Brandt, 2009).

Improved Benchmarks

Production-quality science and engineering applications are typically large, complicated and full-featured software products. As a result, they tend to be challenging to port to new computer platforms and require a well-trained user to do so. Although benchmarking of these applications on new platforms is essential as part of the design and implementation of a new computer system, the scope of this benchmarking is necessarily limited by the complexity of the software product, not to mention its demand for a full scope of system features that are only available after a new computer system reaches its near-production capabilities.

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 application performance.

To address these needs, our recent work in application performance analysis takes advantage of two important properties of many applications: Although an application may have one million or more source lines of code, performance is often dominated by a very small subset of lines; and, for the remaining code, these applications often contain many physics models that are mathematically distinct but have very similar performance characteristics.

To exploit the properties listed above, we have developed a growing collection of mini-applications (called *miniapps*). Miniapps take advantage of the above two application properties by encapsulating only the most important computational operations and consolidating physics capabilities that have the same performance profiles. The large-scale application developer, who is tasked with developing the miniapp, guides the decisions, resulting in a code that is a small fraction of the original application size, yet still captures the primary performance behavior.

There are many benchmarking efforts for scientific computing. The Top 500 High Performance Linpack (HPL) and the HPC Challenge benchmark suite are among the most popular. In addition, full-scale applications are often used for performance analysis, but usually on near-production systems. Between these two extremes there is a middle ground for small, self-contained programs that, like benchmarks, contain the performance-intensive computations of a large-scale application, but are large enough to also contain the context of those computations. The NAS Parallel Benchmarks fall into this category and have been commonly used, as have the compact or synthetic applications developed as part of the Department of Defense High Performance Computing Modernization Program. SWEEP3D also fits this category.

The Mantevo project (Heroux, et. al., 2009) has developed several miniapps that are available via the GNU Lesser General Public License (LGPL) and downloadable from the Mantevo website. The miniapps include implicit finite elements (MiniFE), molecular dynamics (MiniMD), contact detection (phdMesh) and electrical circuits (MiniXyce). The following sections discuss MiniFE and MiniMD.

MiniFE: Implicit Finite Elements

Many engineering applications require the implicit solution of a nonlinear system of equations where the vast majority of time--as problem size increases--is spent in some variation of a conjugate gradient solver. As a result, any miniapp focusing on this area will necessarily have a conjugate gradient solver as the dominant computational kernel.

MiniFE (also known as HPCCG) is a miniapp that mimics the finite element generation, assembly and solution for an unstructured grid problem. The physical domain is a 3D box with configurable dimensions and a structured discretization (which is treated as unstructured). The domain is decomposed using a recursive coordinate bisection (RCB) approach and the elements are simple hexahedra. The problem is linear and the resulting matrix is symmetric, so a standard conjugate gradient algorithm is used with a general sparse matrix data format and no preconditioning.

This simple code—which is not intended to be a true physics problem—is sufficiently realistic for performance purposes. Furthermore, it contains approximately 1,500 lines of C++ code.

MiniMD: Molecular Dynamics

The MiniMD application is miniature version of the molecular dynamics (MD) application LAMMPS (LAMMPS 2009). The source for MiniMD is less than 3,000 lines of C++ code. Like LAMMPS, MiniMD uses spatial decomposition MD, where individual processors in a cluster own subsets of the simulation box. And like LAMMPS, MiniMD enables users to specify a problem size, atom density, temperature, timestep size, number of timesteps to perform, and particle interaction cutoff distance. But compared to LAMMPS, MiniMD's feature set is extremely limited, and only one type of pair interaction (Lennard-Jones) is available. No long-range electrostatics or molecular force field features are available. Inclusion of such features is unnecessary for testing basic MD and would have made MiniMD much bigger, more complicated, and harder to port to novel hardware. The current version of LAMMPS includes over 130,000 lines of code in hundreds of files, nineteen optional packages, over one hundred different commands, and over five hundred pages of documentation. Such a large and complicated code is not ideally suited for answering certain performance questions or for tinkering by non-MD-experts.

A rewrite of a full application code base would be a daunting task, but a complete rewrite of a miniapp to test a new idea can be achieved fairly quickly. We have used MiniFE, MiniMD and the other Mantevo miniapps to test numerous software performance questions and ideas. Explorations included changing from double to single precision to investigate how much performance would improve. We have also developed performance models that provide a mathematical description of performance. For example, MiniMD has been used to test the scaling performance of the spatial decomposition algorithm as the number of processors increased towards infinity. It was found that the fraction of time spent on computation did not approach unity (the fraction of time spent on communication did not approach zero). This finding demonstrated a limitation of the spatial decomposition algorithm for performing MD (MiniMD 2009).

CONCLUSIONS

There is currently considerable interest in Exascale computing. This paper discussed several application drivers, technological challenges and ongoing research aimed at overcoming some of these hurdles. However, a collaborative international effort will be needed to overcome all the key research challenges.

REFERENCES

Brandt, J., Gentile, A., Mayo, J., P´ebay, P., Roe, D., Thompson, D., & Wong, M. (2009). Methodologies for advance warning of compute cluster problems via statistical analysis: a case study. In *Resilience '09: Proceedings of the 2009 workshop on Resiliency in high performance* (pages 7–14). ACM.

Brown, J.L., Goudy, S., Heroux, M.A., Huang, S.S. & Wen, Z. (2006). An evolutionary path towards virtual shared memory with random access. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures* (page 117). ACM.

Daly, J. (2006). A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps. *Future Generation Computer Systems* (pages 303–312).

Department Of Energy (2007). "Annual Energy Review 2007." Energy Information Administration, 2007. Retrieved on June 16, 2009 from www.dramexchange.com.

Department of Energy (2007). *Modeling and Simulation at the Exascale for Energy and the Environment*, Office of Science, http://www.mcs.anl.gov/~insley/E3/E3-draft-2007-08-09.pdf, Washington, D.C.

DRAM Exchange (2009). Retrieved June 16, 2009 from http://www.dramexchange.com.

Feldman, M. (2008). ORNL's "Jaguar" Leaps Past Petaflop. *HPCWire.* On Jul5, 2009 from http://www.hpcwire.com/blogs/ORNLs_Jaguar_Leaps_Past_Petaflop_34282109.html.

Hack, J., & Bierly, E. (2007). *Computational and Informational Technology Rate Limiters to the Advancement of Climate Change Science*, presentation given to the DOE Advanced Scientific Computing Research Advisory Committee, November 6–7, 2007. http://www.sc.doe.gov/ascr/ASCAC/presentationpage1107.html.

Heermann, P. (1998). Production Visualization for the ASCI One TeraFLOPS machine. In *Proceedings of Visualization '98* (pp. 459-462). IEEE.

Hemmert, S., Underwood, K., & Rodrigues, A. (2007). An Architecture to Perform NIC Based MPI Matching. In *2006 International Conference on Cluster Computing (Cluster 2007)*.

Heroux, M. (2003). Trilinos Home Page, 2003. http://trilinos.sandia.gov.

Heroux, M. (2009). Mantevo Home Page, 2009. http://software.sandia.gov/mantevo.

Heroux, M., et. al. (2009). Improving Application Performance via Mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, 2009.

ICKnowledge LLC. "IC Cost Model 0904a." June 2009.

ITRS International Roadmap Committee (2007). International Technology Roadmap for Semiconductors.

Jacob, B. (2009) DRAMSim: University of Maryland Memory-System Simulation Framework. *University of Maryland Memory-Systems Research.* 2009. Accessed on June 19, 2009 from http://www.ece.umd.edu/dramsim/#version2.

Kahle, J. (2005). The Cell Processor Architecture. In *Proceedings of the 38th Annual IEEE/ACM International Symposium or Microarchitcture*. IEEE Computer Society.

Keiter, E.R., Mei, T., Russo, T.V., Rankin, E.L., Pawlowski, R.P., Schiek, R.L., Santarelli, K.R., Coffey, T.S., Thornquist, H.K., & Fixel, D.A. (2008). Xyce Parallel Electronic Simulator: Users' Guide, Version 4.1. Technical Report SAND2008-6461, Sandia National Laboratories, 2008.

Keyes, D., A. Kritz, & W. Tang, *Fusion Simulation Project (FSP): Workshop Report*, presentation at the DOE Advanced Scientific Computing Research Advisory Committee (ASCAC) meeting, November 2007.

Kim, J., Dally, W.J., & Abts, D. (2007). Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture* (pp.126-137).

Kim, J., Dally, W.J., Scott, S., & Abts, D. (2008). Technolgy-driven, highly-scalable, Dragonfly topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture* (pp.77-88).

Kogge, P. et al. (2008). ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. Department of Computer Science, University of Notre Dame, Notre, Dame.

LAMMPS Molecular Dynamics Simulator (2009). http://lammps.sandia.gov/index.html.

Laudon, James. "UltraSPARC T1: A 32-threaded CMP for Servers." Berkeley CMP presentation, 2007.

Mann, R. (2007). *BioEnergy Science Center: A DOE Bioenergy Research Center*, presentation at the 2007 Fall Creek Falls Workshop, Nashville, Tennessee, September 2007. Information available at http://www.iter.org.

Micron technology (2007). Calculating Memory System Power for DDR3. Boisie, Idaho.

MIPS Technologies, Inc. (2007). MIPS64® 5Kc® Processor Core Data Sheet.

Murphy, R.C. (2007). "On the Effects of Memory Latency and Bandwidth on Supercomputer Application Performance." *Proc. of IEEE Internaional Symposium on Workload Characterization 2007 (IISWC07)*, September 27-29, 2007.

Murphy, R.C., & Kogge, P.M. (2007). "On the Memory Access Patterns of Supercomputer Applications: Benchmark Selection and Its Implications." *IEEE Transactions on Computers 56*, no. 7 (July 2007): 937-945.

Murphy, R., Rodrigues, A., Kogge, P., & Underwood, K. (2009). The Implications of Working Set Analysis on Supercomputing Memory Hierarchy Design. *International Conference on Supercomputing.* Cambridge.

Numrich, R.W. & Heroux, M.A. (2009). A performance model with a fixed point for a molecular dynamics kernel. *In Proceedings International Supercomputing Conference '09*.

Oliner, A., Stearley, J., What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 575–584, Edinburgh, UK, June 2007. IEEE Computer Society Press.

Oldfield, R.A., Arunagiri, S., Teller, P.J., Seelam, S., Riesen, R., Varela, M.R., & Roth, P.C.. Modeling the impact of checkpoints on next-generation systems. In *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, San Diego, CA, Sept. 2007.

Qthreads (2009) Sandia National Laboratories: Qthreads, 2009. http://www.cs.sandia.gov/qthreads.

Rupnow, K., Rodrigues, A., Underwood, K., & Compton K. (2006). "Scientific Applications vs. SPEC-FP:A Comparison of Program Behavior", In *Proceedings of the International Conference on Supercomputing*.

Shiva, S. (2005). *Advanced Computer Architectures (page 7)*. Boca Raton, FL: CRC Press.

Simon, H., Zacharia, T., Stevens, R. et al. (2007). Modeling and Simulation at the Exascale for Energy and the Environment. Department of Energy Technical Report. http://www.sc.doe.gov/ascr/ProgramDocuments/TownHall.pdf.

Thoziyoor, S., Ahn, J.H. , Muralimanohar, N., & Jouppi, N. (2008). Cacti 5.1. HP Labs.

Watts, M (2009). Microphotonic Circuits, Networks, and Sensors. *Center For Integrated Photonic Systems Annual Meeting*. Massachusetts Institute of Technology.

Weigand, G., *Energy Assurance and High Performance Computing*, Supercomputing 2007, presentation given at the ORNL booth, November 2007.