Exascale Computing Technology Challenges

John Shalf[1], Sudip Dosanjh[2], and John Morrison[3]

[1] NERSC Division, Lawrence Berkeley National Laboratory,
1 Cyclotron Road, Berkeley, California 94611
[2] Sandia National Laboratories,
New Mexico 87185
[3] Los Alamos National Laboratory,
Los Alamos, New Mexico 87544
jshalf@lbl.gov, sudip@sandia.gov, jfm@lanl.gov

Abstract. High Performance Computing architectures are expected to change dramatically in the next decade as power and cooling constraints limit increases in microprocessor clock speeds. Consequently computer companies are dramatically increasing on-chip parallelism to improve performance. The traditional doubling of clock speeds every 18-24 months is being replaced by a doubling of cores or other parallelism mechanisms. During the next decade the amount of parallelism on a single microprocessor will rival the number of nodes in early massively parallel supercomputers that were built in the 1980s. Applications and algorithms will need to change and adapt as node architectures evolve. In particular, they will need to manage locality to achieve performance. A key element of the strategy as we move forward is the co-design of applications, architectures and programming environments. There is an unprecedented opportunity for application and algorithm developers to influence the direction of future architectures so that they meet DOE mission needs. This article will describe the technology challenges on the road to exascale, their underlying causes, and their effect on the future of HPC system design.

Keywords: Exascale, HPC, codesign.

1 Introduction

Node architectures are expected to change dramatically in the next decade as power and cooling constraints limit increases in microprocessor clock speeds (which are expected to remain near 1 GHz). Consequently computer companies are dramatically increasing on-chip parallelism to improve performance. The traditional doubling of clock speeds every 18-24 months is being replaced by a doubling of cores, threads or other parallelism mechanisms. During the next decade the amount of parallelism on a single microprocessor will rival the number of nodes early massively parallel supercomputers that were built in the 1980s.

Applications and algorithms will need to change and adapt as node architectures evolve. They will need to manage locality and perhaps resilience to achieve high performance. In addition, hardware breakthroughs will be needed to achieve useful Exascale computing later this decade, at least within any reasonable power budget. A

J.M.L.M. Palma et al. (Eds.): VECPAR 2010, LNCS 6449, pp. 1–25, 2011.

key element of the strategy as we move forward is the co-design of applications, architectures and programming environments as shown in Figure 1. Much greater collaboration between these communities will be needed to overcome the key Exascale challenges. There is an unprecedented opportunity for application and algorithm developers to influence the direction of future architectures so that they meet DOE mission needs.

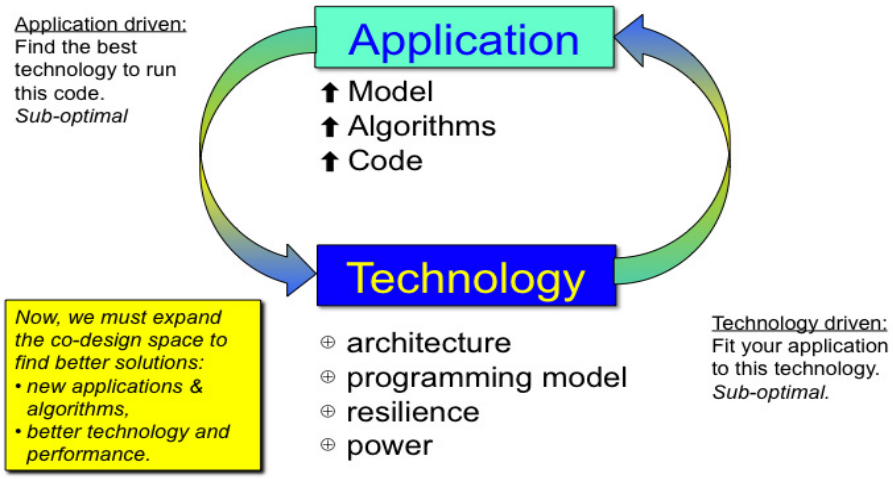


Fig. 1. Schematic of application-driven hardware/software co-design process

These trends are illustrated in Figure 2, which shows the energy cost of moving data to different levels of system memory relative to the cost of a floating point operation. The cost of data movement will not improve substantially whereas the cost of performing a floating -point operation will likely improve between 5x to 10x. Past attempts to exploit intra-node parallelism did not show significant benefits primarily because the cost of data movement within a node was not substantially lower than the cost of moving data across the interconnect because the cost of moving data off-chip dominated the energy costs. However, modern chip multiprocessors have CPU's co-located on the same chip. Consequently, there is a huge opportunity to capture energy-efficiency and performance benefits by directly taking advantage of intra-chip communication pathways.

2 Metrics, Cost Functions, and Constraints

For Exascale systems, the primary constraints are (for the purposes of discussion) platform capital costs under $200M, less than 20MW power consumption, and delivery in 2018. All other system architectural choices are free parameters, and are optimized to deliver maximum application performance subject to these very challenging constraints.

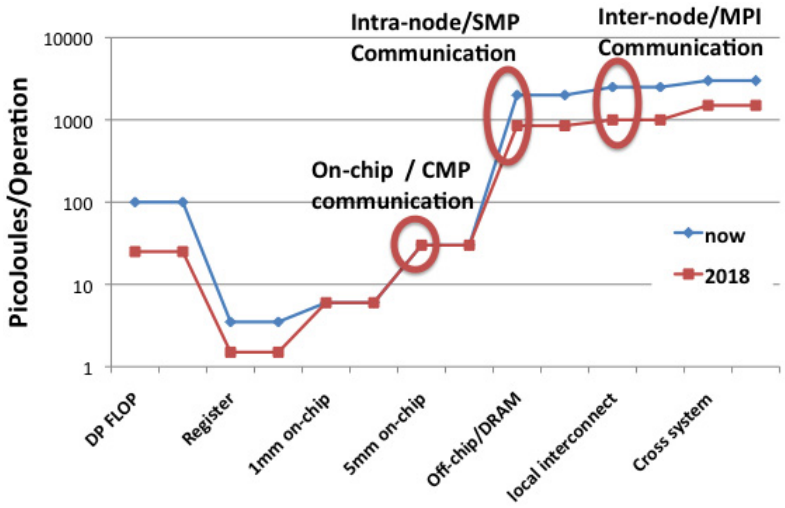


Fig. 2. Energy cost of data movement relative to the cost of a flop for current and 2018 systems (the 2018 estimate is conservative and doesn't account for the development of an advanced memory part). The biggest change in energy cost is moving data off-chip. Therefore, future programming environments must support the ability of algorithms and applications to exploit locality which will, in turn, be necessary to achieve performance and energy efficiency.

In an ideal world, we would design systems that would never subject applications to any performance constraints. However, component costs and power usage force system architects to consider difficult trade-offs that balance the actual cost of system components against their effect on application performance. For example, if doubling floating point execution rate nets a 10% gain in overall application performance, but only increases system costs by 5%, then it is a net benefit despite degrading system balance. It is important to have an open dialog to fully understand the cost impacts of key design choices so that they can be evaluated against their benefit to the application space.

Cost Functions

The Cost of Power: Even with the least expensive power available in the US, the cost of electricity to power supercomputing systems is a substantial part of the Total Cost of Ownership (TCO). When burdened with cooling and power distribution overheads, even the least expensive power in the U.S. (< 5cents/KWH) ultimately costs $1M per Megawatt per year to operate a system. To keep the TCO manageable DOE's Exascale Initiative Steering Committee adopted 20MW as the upper limit for a reasonable system design [1,2]. This limit is movable, but at great cost and design risk.

The Cost of a FLOP: Floating point used to be the most costly component of a system both in terms of design cost and power. However, today, FPUs consume a very small fraction of the area of a modern chip design and a much smaller fraction of the

power consumption. On modern systems, a double-precision FMA (fused multiply add) consumes 100 picoJoules. By contrast, reading the double precision operands from DRAM costs about 2000 pJ. By 2018 floating point operations will consume about ~10.6pJ/op on 11nm lithography technology [3], and the cost of reading from DRAM will only improve modestly to 1000pJ unless more energy-efficient memory technology is developed.

With these figures of merit, it would only consume 100W to put 10 Teraflops on a chip, which is easily achievable. However, it would require 2000W of power required to supply memory bandwidth to those floating point units at a modest memory bandwidth to floating point ratio of 0.2. The consequence is that we can engineer far more floating point capability onto a chip than can reasonably be used by an application. Engineering FLOPs is not a design constraint – data movement presents the most daunting engineering and computer architecture challenge.

The Cost of Moving Data: Memory interfaces and communication links on modern computing systems are currently dominated by electrical/copper technology. However, wires are rapidly being subsumed by optical technology because of the limits of bit rate scaling as we shrink wires length scales as observed by David A. B. Miller of Stanford [4-5]. Miller observes that for a conventional electrical line (without repeaters or equalization) can be modeled as a simple RC circuit by virtue of the simplified Telegrapher's equation for lossy transmission line. The wire must be charged and discharged at a rate governed by the RC time constant, which is given by equation 1 where R_l is the resistance of the wire, C_l is the capacitance and l is the length of the wire. As the wire length increases, the risetime (given by the RC time constant) increases by the square of the length – thereby reducing the bit-rate.

$$\text{risetime}=R_l C_l\, l^2 \tag{1}$$

Miller observes that if you shrink the wire proportionally in all dimensions by a factor of *s*, the resistance (R_l) increases proportionally to the reduced wire aspect ratio, which reduces by a factor of s^2, but capacitance (C_l) remains the same. The consequence is that for constant voltage, the bit-rate carrying capacity of an RC line scales proportional to $B \approx A/l^2$, where B is the bandwidth of the wire and A is the cross-sectional area of the wire and l^2 is the length of the wire. The consequence of this observation is that natural bit rate capacity of the wire depends on the aspect ratio of the line, which is the ratio of the length to the cross-sectional area for a constant input voltage and does not improve as we shrink the wires down with smaller lithographic processes. We can push to a higher bitrate by increasing the drive voltage to the wire, but this also increases power consumption. These effects are summarized in equation 2, which assumes a simple RC model of the wire and no re-amplification *(long-haul wires on-chip are normally re-amplified at regular intervals to maintain a linear power profile as a function of length, but at a cost of more power consumption).*

$$\text{Power} \approx B \times l^2 / A \tag{2}$$

This has the following consequences to system design [6, 16]:

- Power consumed increases proportionally to the bit-rate, so as we move to ultra-high-bandwidth links, the power requirements will become an increasing concern.

- Power consumption is highly distance-dependent (quadratically with wire length without re-amplification), so bandwidth is likely to become increasingly localized as power becomes a more difficult problem.
- Improvements in chip lithography (making smaller wires) will not improve the energy efficiency or data carrying capacity of electrical wires.

In contrast, optical technology does not have significant distance-dependent energy consumption. It costs nearly the same amount of energy to transmit an optical signal 1 inch as it does to transmit it to the other end of a room. Also, signaling rate does not strongly affect the energy required for optical data transmission. Rather, the fixed cost of the laser package for optical systems and the absorption of light to receive a signal are the dominant power costs for optical solutions.

As the cost and complexity of moving data over copper will become more difficult over time, the cross-over point where optical technology becomes more cost-effective than electrical signaling has been edging closer to the board and chip package at a steady pace for the past 2 decades. Contemporary short-distance copper links consume about 10-20 pJ/bit, but could be improved to 2pJ/bit for short-haul 1 cm length links by 2018. However, the efficiency and/or data carrying capacity of the copper links will fall off rapidly with distance (as per equation 2) that may force a movement to optical links. Contemporary optical links consume about 30-60pJ/bit, but solutions that consume as little as 2.5pJ/bit have been demonstrated in the lab. In the 2018 timeframe optical links are likely to operate at 10pJ/bit efficiency [7]. Moreover, silicon photonics offers the promise of breaking through the limited bandwidth and packaging constraints of organic carriers using electrical pins.

Another serious barrier to future performance growth is cost of signals that go off-chip as we rapidly approach pin-limited bandwidth. Due to the skin effect [19], and overheads of more complex signal equalization, it is estimated that 10-15GHz is likely the maximum feasible signaling rate for off-chip differential links that are 1-2cm in length. A chip with 4000 pins would be a very aggressive, but feasible design point for 2018. If you consider that half of those pins (2000) are power and ground, while the remaining 2000 pins are differential pairs, then the maximum feasible off-chip bandwidth would be ~1000 × 10GHz, which comes to approximately 1 Terabyte/second (*10 Terabits/sec with 10/8 encoding*). Breaking through this 1 TB/s barrier would require either more expensive, exotic packaging technology (ceramics rather than organic packages), or migration to on-chip optics, such as silicon-photonic ring-resonator technology [20, 21].

Without major breakthroughs in packaging technology or photonics, it will not be feasible to support globally flat bandwidth across a system. Algorithms, system software, and applications will need to aware of data locality. The programming environment must enable algorithm designers to express and control data locality more carefully. The system must have sufficient information and control to make decisions that maximally exploit information about communication topology and locality. Flat models of parallelism (e.g. flat MPI or shared memory/PRAM models) will not map well to future node architectures.

3 Memory Subsystem

Ultimately, memory performance is primarily constrained by the dynamics of the commodity market. One key finding of DOE's Architecture and Technology workshop [8]

was that memory bandwidth is primarily constrained by power & efficiency of the memory interface protocols, whereas memory capacity is primarily constrained by cost. Early investments in improving the efficiency of DRAM interfaces and packaging technology may result in substantially improved balance between memory bandwidth and floating point rate. Investments in packaging (mainly chip-stacking technology) can also provide some benefit in the memory capacity of nodes, but it is unclear how much the price of the components can be affected by these investments given commodity market forces.

3.1 Memory Bandwidth

The power consumed by data movement will dominate the power consumption profile of future systems. Chief among these concerns is the power consumed by memory technology, which would easily dominate the overall power consumption of future systems if we attempt to maintain historical bandwidth/flop ratios of 1 byte/flop. A 20 MW power constraint on an Exascale system will limit the breadth of applications that can execute effectively on such systems unless there are fundamental breakthroughs in memory and communications technologies.

For example, today's DDR-3 memory interface technology consumes about 70picoJoules/bit, resulting in approximately 5000 pJ of energy to load a double-precision operand (accounting for ECC overhead). If we extrapolate the energy-efficiency of memory interfaces to DDR-5 in 2018, the efficiency could be improved to 30pJ/bit. A system with merely 0.2 bytes/flop of memory bandwidth would consume > 70Megawatts of power, which is not considered a feasible design point. Keeping under the 20MW limit would force the memory system to have less than 0.02 bytes/flop, which would severely constrain the number of applications that could run efficiently on the system as illustrated in Figures 4 and Figure 5.

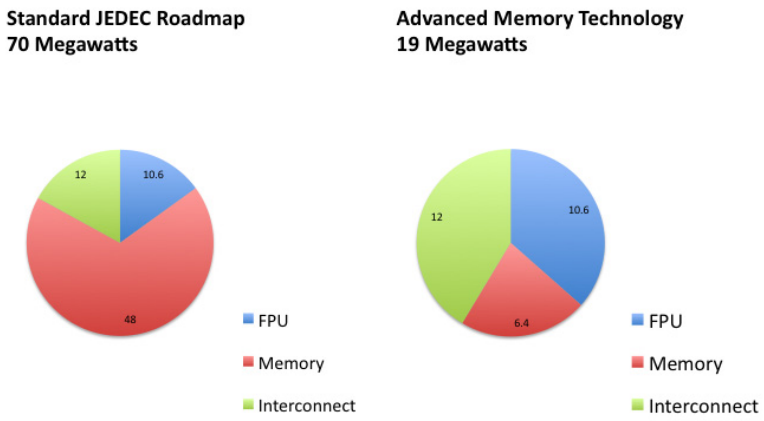


Fig. 3. If we follow standard JEDEC memory technology roadmap, the power consumption of a feasible Exascale system design (using 0.2 bytes/flop memory bandwidth balance) will be >70Megawatts due to memory power consumption, which is an impractical design point. Keeping memory power under control will either require substantial investments in more efficient memory interface protocols, or substantial compromises in memory bandwidth and floating point performance (< 0.02 bytes/flop).

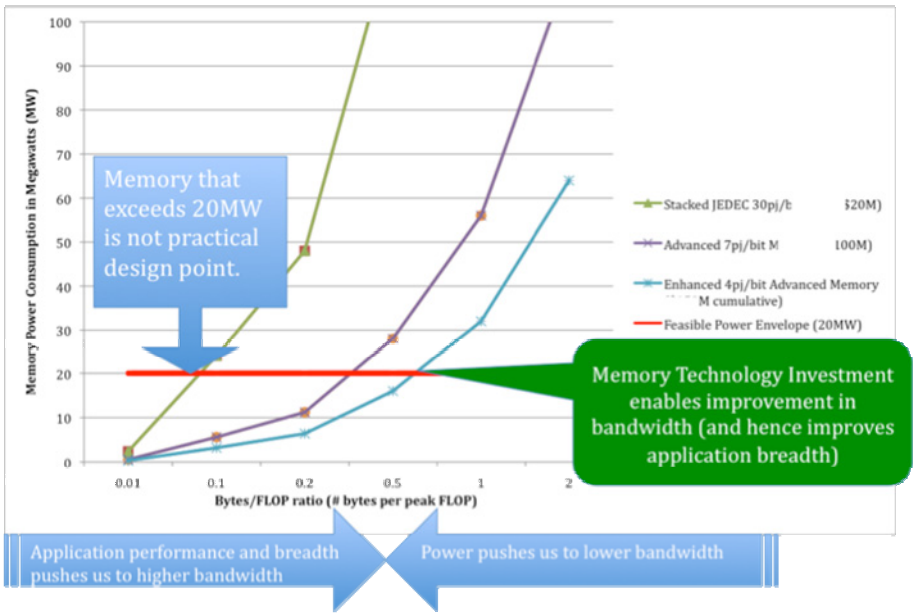


Fig. 4. This figure illustrates the trade-offs between memory power consumption and the desire for a more broadly applicable Exascale system design under different assumptions about investments in advanced memory technology.

We cannot reach reasonable memory energy efficiency by following the JEDEC roadmap. Getting to a reasonable energy efficiency requires development of new, more efficient interface designs and memory protocols. Advanced memory technology can get to about 7 pJ/bit with investments to bring the technology to market. The limit of this new technology is estimated to be 4pJ/bit (excluding memory queues and controller logic). Therefore, in order to maintain 0.2 byte/flop system balance and stay under a 20MW design limit for power requires either substantial investments in advanced memory technology, or a substantial degradation in system memory balance, as illustrated in Figure 5. As always, these ratios are movable. For example, the power limit could be relaxed, but would put the feasibility of fielding siting such a system in jeopardy and increase the total cost of ownership.

3.2 Memory Capacity

One figure of merit for improvements to HPC systems is the total memory capacity. More aggregate memory enables systems to solve problems that have either proportionally higher resolution, or more physics fidelity/complexity – or both. However, cost considerations may limit an exascale system to a memory capacity that improves only by a factor of 100x in comparison to the system peak floating point rate which will improve by 1000x. This is a movable parameter in the design space of the machine, but the consequence of moving this parameter is increased cost for the memory subsystem and the total cost of the system.

The DRAM capacity of a system is primarily limited by cost, which is defined by the dynamics of a broad-based high-volume commodity market. The commodity

market for memory makes pricing of the components highly volatile, but the centroid of the market is approximately $1.80/chip. Figure 4 illustrates that the rate of memory density improvement has gone from a 4x improvement every 3 years to a 2x improvement every 3 years (a 30% annual rate of improvement). Consequently the cost of memory technology is not improving as rapidly as the cost of Floating Point capability. Given the new rate of technology improvement, 8 gigabit memory parts will be widely available in the commodity market in the 2018 timeframe and 16 gigabit parts will also have been introduced. It is unclear which density will be the most cost-effective in that timeframe.

If we assume that memory should not exceed 50% of the cost of a computer system, and that the anticipated capital cost of an Exascale system is $200M, then Table 5 shows that the memory capacity we could afford lies somewhere between 50 and 100Petabytes. Again, these are not hard limits on capacity, but they do have a substantial effect on the cost of the system, and the trade-off between memory capacity and other system components must be considered carefully given a limited procurement budget.

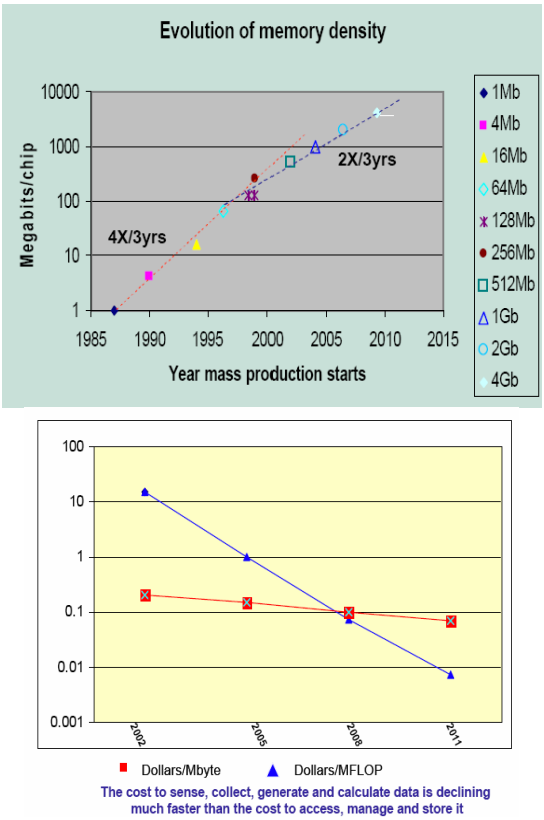


Fig. 5. The rate of improvement in memory technology improving at slower rates -- now approaching 30% per year. (Figure courtesy of David Turek, IBM).

3.3 Latency

Off-chip latencies are unlikely to improve substantially over existing systems. With a fixed clock rate of 2 GHz, the distance to off-chip memory on modern systems is approximately 100ns (200 clock cycles away), and will potentially improve to 40-50ns (100 clock cycles away from memory) in the 2018 timeframe. A modern interconnect has a messaging latency of 1 microsecond. Most of that latency is on the end-points for the message (message overhead of assembling a message and interrupt handling to receive it). By 2018, this could improve to as little as 200-500ns for message latency, which is at that point limited by the speed of light (0.75c in optical fiber) and comes to about 5ns latency per meter of cable.

Lastly, the message injection rates of modern systems (an indirect measure of the overhead of sending messages) is approximately tens of thousands of messages/second on leading-edge designs. If the interconnect NIC is moved on-chip, it may be feasible to support message injection rates of hundreds of millions of messages per second for lightweight messaging (such as one-sided messages for PGAS languages).

With no substantial improvements in off-chip and cross-system latency, the bandwidth-latency product for future systems (which determines the number of bytes that must be in flight to fully saturate bandwidth) will be large. This means there must be considerable attention to latency hiding support in both algorithms and in hardware designs. The approach to latency hiding has not yet been determined.

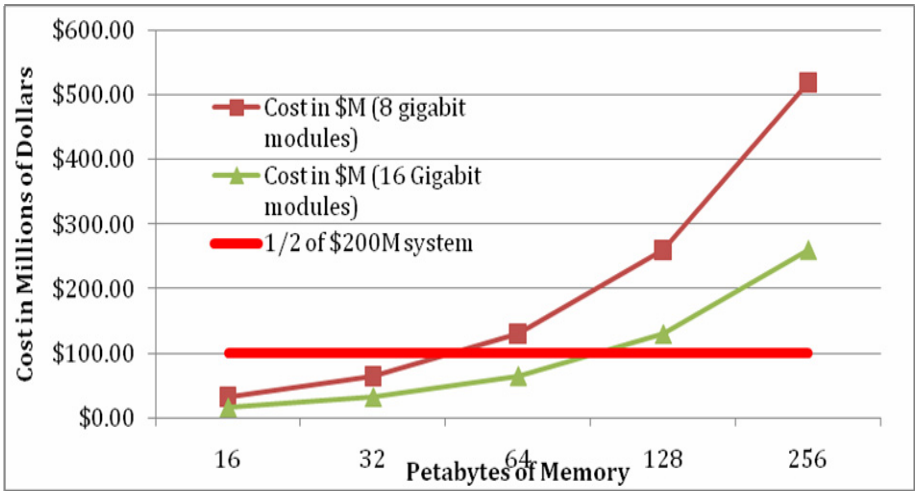


Fig. 6. There are two possible memory chip densities in the 2018 timeframe. It is less certain which option will be most cost-effective.

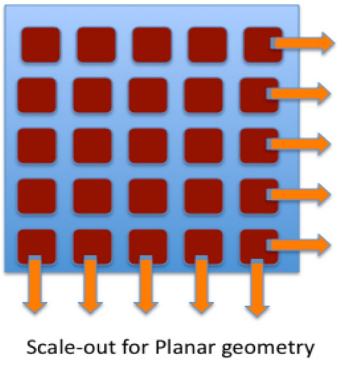
4 Node Architecture Projections for 2018

There are many opportunities for major reorganization of our computation model to take better advantage of future hardware designs. However, much of the discussion

to-date of inter-processor communication semantics and node organization has focused on evolutionary rather than revolutionary features.

4.1 Clock Rate

Technology projections[1,2,3,9] indicate that clock-speeds will not change appreciably by 2018 and will remain near 1-2 GHz. This sets clear design constraints for the number of floating point functional units that will be on a future chip design. In order to keep component counts for future systems within practical limits (< 1M nodes), a node must perform between 1-10 Teraflops. At 1 GHz, that means there will be between 1000 and 10,000 discrete Floating Point Units on a chip.



- ~1000-10k simple cores
- 4-8 wide SIMD or VLIW bundles
- Either 4 or 50+ HW threads
- On-chip communication Fabric
  - Low-degree topology for on-chip communication (torus or mesh)
  - *Scale cache coherence?*
  - cache coherent clusters
  - HW msg. passing
  - Global (nonCC memory)
  - Shared register file (clusters)
- Off-chip communication fabric
  - Integrated directly on an SoC
  - Reduced component counts
  - Coherent with TLB (no pinning)

Fig. 7. Schematic of a future node architecture. The number of functional units on the chip will need to scale out in a 2-D planar geometry and communication locality between the functional units will be increasingly important for efficient computation.

4.2 Instruction Level Parallelism

Up until recently, microprocessors depended on Instruction Level Parallelism and out-of-order execution to make implicit parallelism available to a programmer and to hide latency. Power and complexity costs make it clear that we cannot depend on out-of-order instruction streams to hide latency and improve performance. Instead, we must move to more explicit forms of exposing parallelism such as SIMD units and chips with many independent CPUs.

4.3 Instruction Bundling (SIMD and VLIW)

One way to organize floating point functional units to get implicit parallelism is to depend on grouping multiple operations together into SIMD or VLIW bundles. The benefit of such bundling is that they enable finer-grained data sharing among the

instructions, which lowers energy costs and controls complexity. Although SIMD is the most popular approach to organizing FPUs today, there may be movement towards a VLIW organization because it is more flexible in instruction mixing.

The number of SIMD units on x86 chips has doubled in recent years, but the ability to fully exploit evern greater widths is questionable. GPUs also depend on very wide SIMD units, but the semantics of the GPU programming model (CUDA for example) make it easier to automatically use SIMD or VLIW lanes. Currently, Nvidia uses 32-wide SIMD lanes, but there is a pressure to shrink down to 4-8. Current CPU designs have a SIMD width of 4 slots, but will likely move to 8 slots. Overall, this indicates a convergence in the design space towards 4-8 wide instruction bundles (whether it be SIMD or VLIW).

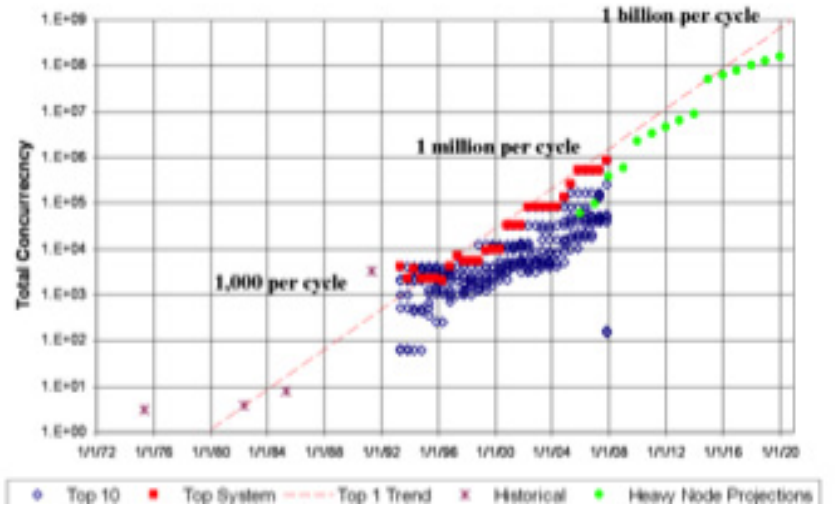


Fig. 8. Due to the stall in clock speeds, future performance improvements will be from increased explicit parallelism. 2018 systems may have as much as 1 billion way parallelism (from DARPA Exascale Report)[2].

4.4 Multithreading to Hide Latency

Little's Law (equation 3) is derived from general information theory, but has important application to understanding the performance of memory hierarchies.

$$\#\text{outstanding\_memory\_requests} = \text{bandwidth} * \text{latency} \tag{3}$$

In order to fully utilize the available bandwidth of a memory interface, this equation must be balanced. If you have a high bandwidth memory interface, bandwidth will be underutilized if there are not enough outstanding memory requests to hide the latency term of this equation (latency limited). Since we will no longer be depending on complex out-of-order instruction processors to hide latency in the memory hierarchy, there will be increased dependence on hardware multithreading to achieve latency hiding. The latency to local memory is 100ns, but typically you don't have to hide all of the time due to cache reuse.

In swim-lane #1, manycore chip architectures currently support 2-4-way multithreading, and this may increase to 4-8 way multithreading in future architectures depending on energy cost. GPUs currently depend on 48-64-way hardware multithreading and will likely support these many threads in the future.

The consequence for programming models is that the baseline expression of parallelism will require 1 billion-way parallelism to achieve an Exaflop if a 1 GHz clock-rate is used. Additional hardware threading required to hide latency will increase the amount of parallelism by a factor of 10-100x.

4.5 FPU Organization

Floating point used to be the most costly component of a system both in terms of design cost and power. However, today, FPUs use a very small fraction of the area of a modern chip design and consume an even smaller fraction of power. On modern systems, a double-precision FMA (fused multiply add) consumes 100 pJ per FMA in 65nm lithography. For 11nm technology anticipated for a 2018 system, a double precision FMA will consume approximately 10.6pJ/op and take 0.02 mm^2 of chip surface area. The FPUs of modern CPUs consume a very small fraction of chip surface area (5-10% of a 400 mm^2 die), whereas GPUs see a larger fraction of their surface area developed to FPUs and general ALUs. A CPU design that consists of many lightweight cores (a manycore chip akin to Larrabee, or Tilera) would likely see the fraction of die area devoted to FPUs close to that observed on modern GPUs.

In order to reduce failure rates and component counts, it is desirable to build a system that reduces the total number of nodes by maximizing the performance of each node. Placing 10,000 FPUs on a chip would only consume 100Watts in this timeframe, and is entirely reasonable in terms of area and power consumption. However supplying memory bandwidth and capacity to a 10Teraflop chip is the primary barrier to this design point. Without advanced packaging technology and substantial improvements in DRAM interface energy efficiency, the upper limit for per-chip performance will likely be 1-2 Teraflops/chip.

We consider two design points to represent this range.

- Swim Lane 1: 1,000 FPUs per chip
- Swim Lane 2: 10,000 FPUs per chip

To support full floating point performance, the on-chip register file bandwidth would need to supply 64 bytes per op. Therefore, a 10 Teraflops chip requires 320TB/s of register file bandwidth and 64TB/s register file bandwidth is needed for a 1TF chip. The upper limit of feasible off-chip memory bandwidth will be 4TB/s. Therefore, the design point for Swim Lane 2 would require O(100) data reuse on chip and the design point for Swim Lane 1 would require O(10) data reuse on chip if a 4TB/s memory interface is used. In both cases, the assumed quantity of on-chip memory is on the order of 0.5-1GB/chip, so all temporal recurrences necessary to achieve on-chip data reuse would need to be captured within this memory footprint.

For node organizations that use more than one chip for a node, the bandwidth would likely be more on the order of 0.5 to 1TB/s to remote DRAM (1/4 to 1/8 of local DRAM BW). Therefore, NUMA effects on a multi-chip node will have a substantial performance impact.

4.6 System on Chip (SoC) Integration

To reduce power, and improve reliability it is useful to minimize off-chip I/O by integrating peripheral functions, such as network interfaces and memory controllers, directly onto the chip that contains the CPUs. There are fringe benefits, such as having the communication adaptor be TLB-coherent with the processing elements, which eliminates the need for expensive memory pinning or replicated page tables that is required for current high-performance messaging layers. It also reduces exposure to hard-errors caused by mechanical failure of solder joints. From a packaging standpoint, the node design can be reduced to a single chip surrounded by stacked memory packages, which increases system density. SoC integration will play an increasingly important role in future HPC node designs.

4.7 Alternative Exotic Functional Unit Organizations

Accelerators and Heterogenous Multicore Processors: Accelerators and heterogeneous processing offers some opportunity to greatly increase computational performance within a fixed power budget, while still retaining conventional processors to manage more general purpose components of the computation such as OS services. Currently, such accelerators have disjoint memory spaces that are at the other end of a PCIe interface, which makes programming them very difficult.

There is a desire to have these accelerators fully integrated with the host processor's memory space. At low end, accelerators already are integrated in a unified memory space, but such integration is difficult at the high-end because of differences in the specialized memory technology used for the accelerator and the host processor. By 2015 it will be feasible from a market standpoint to integrate scalar cores with accelerators to obviate the need to copy data between disjoint memory spaces. This was true for NVidia GPU solutions and possibly for heterogeneous manycore architectures like Intel's Larrabee/Knight's Corner[10].

FPGAs and Application-Specific Accelerators: Application specific functional unit organizations may need to be considered to tailor computation and power utilization profiles to more closely match application requirements. However, the scope of such systems may be limited and therefore impact the cost-effectiveness of the resulting system design. FPGAs enable application-tailored logic to be created on-the-fly, but are currently too expensive. Otherwise, FPGA's could be used to implement application-specific primitives.

There is some evidence that power considerations will force system architects to rely on application-tailored processor designs in the 2020 timeframe. Economics will likely constrain the number of application tailored processor designs to a small number and the high performance computing marketplace may not be of sufficient size to warrant its own application-tailored processor.

5 Cache Hierarchy

5.1 Levels of Cache Hierarchy

There has been general agreement among computer companies that there will be 2-4-levels of on-chip hierarchy that can be managed explicitly or flipped to implicit

state. The reason for a multi-level hierarchy is mostly governed by the cost of data movement across the chip. Moving data 1 mm across a chip costs far less than a floating point operation, but movement of 20mm (to the other end of the chip) costs substantially more than a floating point operation. Consequently, computation and memory hierarchy on a chip will likely be grouped into clusters or hierarchies of some form to exploit spatial locality of data accesses. There will need to be more effort to create Hardware Block Transfer support to copy data between levels of the memory hierarchy with gather/scatter (multi-level DMA).

5.2 Private vs. Shared Caches

Most codes make no use of cache coherence. So it is likely the cache hierarchy will be organized to put most of the on-chip memory into private cache. Performance analysis indicate less sharing is best (ie. Code written in threads to look like MPI generally performs better).

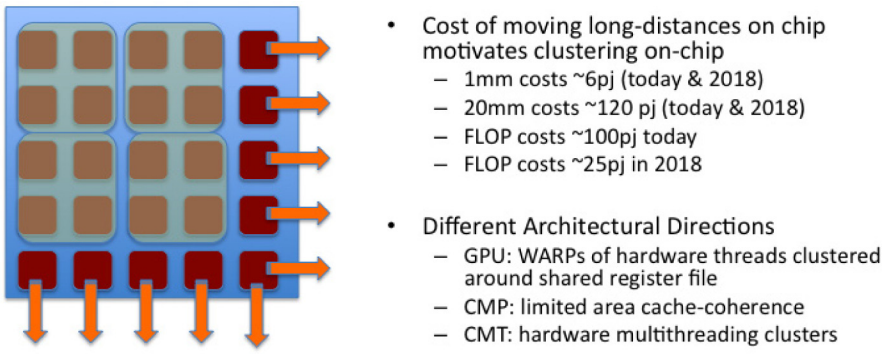


Fig. 9. Processor cores or functional units will likely be organized into groups or a hierarchy in order to exploit spatial locality of data accesses

5.3 Software Managed Caches vs. Conventional Caches

Automatically managed caches virtualize the notion of on-chip and off-chip memory, and are therefore invisible to current programming models. However, the cost of moving data off-chip is so substantial, that virtualizing data location in this manner wastes energy and substantially reduces performance. Therefore, there has been increasing interest in explicit software management of memory, such as the Local-stores used by the STI Cell processor and by GPUs. Over the next decade, explicitly managed on-chip memory will become mainstream in conventional CPU designs as well.

However, we have not found the right abstraction for exposing software-controlled memories in our existing programming models. To support an incremental path for existing applications, these explicitly managed memory hierarchies will need to co-exist with conventional automatically managed caches. These software-managed caches may depend on the ability to switch dynamically from automatically managed caches to software-managed caches. Switchable, and dynamically partitionable caches

are already demonstrated in the Fermi GPUs, but will likely be seen in conventional multicore architectures as well.

When data is placed into an explicitly controlled cache, it can be globally visible to other processors on the chip, but is not visible to the cache-coherence protocol. Therefore, if the path to higher performance involves keeping more data in these explicitly managed caches, then it means cache-coherence (and the notion of an SMP with it) cannot be part of the high-performance path. Programming language designers must consider how to enable expression of on-chip parallelism without an SMP/cache-coherent model.

6 Intra-node Communication (Networks-on-Chip)

The primary area of growth in parallelism is explicit parallelism on-chip. Whereas the number of nodes in an Exascale system is expected to grow by a factor of 10x over the next decade, on-chip parallelism is expected to grow by a factor of 100x. This requires reconsideration of on-chip organization of CPU cores, and the semantics of inter-processor communication.

6.1 Cache Coherence (or Lack Thereof)

It is likely that cache-coherence strategies can scale to dozens of processing elements, but the cost and latency of data movement on chip would make cache-coherence an inefficient method for interprocessor communication for future chip designs. In all likelihood cache-coherence could be used effectively in clusters or sub-domains of the chip (as illustrated in figure 7), but is unlikely be effective if extended across a chip containing thousands of cores. It is more likely that global memory addressing without cache-coherence will be supported with synchronization primitives to explicitly manage memory consistency.

It is unlikely that cache-coherence will be eliminated completely, but there will need to be careful consideration of the trade-offs of the size of the coherency domain with the magnitude of NUMA (Non-Uniform Memory Access) effects. For a fixed power budget, you can offer users a cluster of cache-coherent domains that have minimal NUMA effects, or very large numbers of cores in the cache-coherent domain that expose the programmer to large NUMA effects. A chip with minimal NUMA effects and small coherence domain could be programmed without substantial attention to data locality, but would derive less benefit from surface-to volume ratios if the coherence-domain is small. There is some opportunity in language support for better implicit locality management in both cases. Creating a chip that has a large coherence domain and minimal NUMA effects would require a substantial increase in power budget to over-design the on-chip interconnection network.

6.2 Global Address Space

Partitioned Global Address Space (PGAS) programming models, including the HPCS programming languages benefit from Global Address Space (GAS) to ensure a compact way to reference remote memory across the machine. PGAS models are willing to accept global addressing without SMP cache-coherence on the node. Therefore, there will likely be support for incoherent global addressing for small-scale systems,

but will require hardware investment to scale to larger systems. It is not clear how many address bits will be supported in mainstream implementation. From a technology standpoint, it is entirely feasible to support global addressing within context of Exascale. However, larger scale global addressing schemes will not naturally occur without investment. Global addressing only makes sense with hardware support for sync, which is also investment dependent.

6.3 Fine Grained Synchronization Support

Future programming models will need much finer-grained synchronization features that could directly map to programming language primitives. These features could greatly improve the efficiency of fine-grained on-chip parallelism.

One option is moving atomics memory operations (AMOs) to memory controllers and full empty bits on-chip. Moving atomics as close to memory as possible makes sense from a power and performance standpoint, but would force us to give up some temporal recurrences since the data operated on by the atomics would not pass through the cache hierarchy.

An alternative approach to supporting these atomics is to use an intermediate level of the memory hierarchy where synchronization constructs get enforced/resolved. For example, you could imagine an L2 cache on-chip that is specifically dedicated to fine-grained inter-processor synchronization and atomic memory operations. This approach would potentially encode synchronization state information or other coordinating state using the ECC words of the memory system, because it cannot be held on-chip. All of these options are feasible, but would require close interaction with application developers and programming model designers to determine which approach will be most effective.

7 Power Management

Thermally limited designs force compromises that 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 and greatly limit the scope and effectiveness of such systems.

From an applications perspective, active power management techniques improve application performance on systems with a limited power budget by dynamically directing power 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 — forcing design compromises that limit the memory bandwidth to 0.01 bytes/flop according to the DARPA projections. However, in this thermally limited case you can deliver higher memory bandwidth to the application for the 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 the period of time when the application is bandwidth limited by shifting the power away from floating point (or other components that are under-utilized 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. In a practical application code that contains many solvers, the power modes cannot switch fast enough to be of use. Technology that would enable power management systems to switch to low-power modes within a single clock cycle may emerge in the 2015 timeframe. However, there is still a lot of work required to coordinate switching across a large-scale HPC system. Without system scale coordination of power modes, this approach will not be effective.

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 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 latency for changing between different power modes. 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 current 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 for accomplishing this kind of optimal control for power management is fundamentally broken. Predictive models for response to control decisions are generally hand-crafted (a time-consuming process) for the few examples that currently[11]. There is no common expression of policy or objective. 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.

7.1 Node-Scale Power Management

Operating systems must support Quality-of-Service management for node-level access to very limited/shared resources. For example, the OS must enable coordinated/fair sharing of the memory interface and network adaptor by hundreds or even thousands of processors on the same node. Support for local and global control decisions require standardized monitoring interfaces for energy and resource utilization (PAPI for energy counters). Standard control and monitoring interfaces enable adaptable software to handle diversity of hardware features/designs. Future OS's must also

manage heterogeneous computing resources, and manage data movement and locality in memory hierarchy [13].

7.2 System-Scale Power Management

We need to develop power Performance monitoring and aggregation that scales to 1B+ core system. System management services require standard interfaces to enable coordination across subsystems and international collaboration on component development. Many power management decisions must be executed too rapidly for a software implementation, so must be expressed as a declarative policy rather than a procedural description of actions. Therefore, policy descriptions must be standardized to do fine-grained management on chip. This requires standards for specifying reduced models of hardware power impact and algorithm performance to make logistical decisions about when and where to move computation as well as the response to adaptations. This includes analytical power models of system response and empirical models based on advanced learning theory. We must also develop scalable control algorithms to bridge gap between global and local models. Systems to aggregate sensor data from across the system (scalable data assimilation and reduction), make control decisions and distribute those control decisions in a coordinated fashion across large scale machines are needed. Both online and offline tuning options based on advanced search pruning heuristics should be considered.

7.3 Energy Aware Algorithms

New algorithms must base order of complexity on energy cost of operations rather than FLOPs. A good example of this approach is communication-avoiding algorithms, which trade-off FLOPS for communication to save energy. However, the optimal trade-off is very context specific. There would need to be some methodology to annotate code with a parameterized model of energy consumption for different architectures so that the trade-offs could be computed analytically for different systems. Alternatively, a persistent database could collect runtime information to build up an empirical model of energy consumption for each basic-block of code. Standardizing the approach to specifying or building lightweight analytical models to predict response to resource adjustment will be important to this effort.

7.4 Library Integration with Power Management Systems

Library designers need to use their domain-specific knowledge of the algorithm to provide power management and policy hints to the power management infrastructure. This research agenda requires performance/energy efficiency models and power management interfaces in software libraries to be standardized. This ensures compatibility of the management interfaces and policy coordination across different libraries (horizontal integration) as well as supporting portability across different machines (vertical integration).

7.5 Compiler Assisted Power Management

Compilers and code generators must be able to automatically instrument code for power management sensors and control interfaces to improve the programmability of

such systems. Compiler technology can be augmented to automatically expose "knobs for control" and "sensors" for monitoring of non-library code. A more advanced research topic would be to find ways to automatically generate reduced performance and energy consumption models to predict response to resource adaptation.

7.6 Application-Directed Power Management

Applications require more effective declarative annotations for policy objectives and interfaces to coordinate with advanced power-aware libraries and power management subsystems.

7.7 System "Aging"

Today's systems operate with clock rates and voltages in guard margins to account for chip "wear-out". By employing slight clock speed reduction over the lifetime of the system, a 5% power savings can be achieved instead of using guard bands to account for silicon aging effects.

7.8 Voltage Conversion and Cooling Efficiency

Another key area for power reduction is to design hardware to minimize the losses in voltage regulation and power conversion components. For example, the D.E. Shaw system had 30% efficiency loss just from the power conversion stages going from 480V to lowest voltage level delivered to chips.

There are opportunities to use smart-grid strategies to reduce energy consumption. Improve data center efficiencies (5-10% savings in total power consumption) have been demonstrated using this approach [13]. Smart grid technology can rapidly shift power distribution to balance power utilization across the system.

Exascale systems should be water cooled (some may be warm water cooled) because it is substantially more efficient that air cooling.

8 Fault Detection and Recovery

There is a vibrant debate regarding how much responsibility for fault resilience will need to be handled by applications. As a baseline, nearly all applications running on extreme-scale platforms already incorporate some form of application-based defensive I/O (checkpointing). The discussion is primarily concerns shifting balance of responsibility between hardware and software, and its effect on how much additional burden beyond conventional application-driven checkpointing will be required. System architects are keenly aware that applications writers prefer not to have have additional burdens placed upon them.

The circuit hardening techniques required to handle resiliency entirely in hardware are well understood by industry circuit designers for milspec/radiation-hardened parts. Shifting the responsibility more toward the hardware will have a cost in performance or in power consumption (for example, if you add redundancy to harden critical data paths). However, the biggest concern is how far such parts will depart from high-volume mainstream components that will benefit from sharing NRE costs across a larger set of applications. The current failure rates of nodes are primarily defined by

market considerations rather than technology. So perhaps it is better to project reliability based on market pressure rather than technology scaling.

From the standpoint of technology scaling, the sources of transient errors will increase by a factor of 100 to 1000x. However, offering a laptop or even cell phone that fails at a 1000x higher rate than today is wholly and entirely impractical from the standpoint of a mainstream technology vendor. Therefore industry will be highly motivated to keep per-node soft error rates from degrading.

1. Moore's law die shrinks will deliver a 100x increase in processors per node in the 11 years between the debut of Petascale systems and the debut of Exascale in 2018.
2. We will need to increase the number of nodes by 10x to get to an Exaflop by 2018.
3. Therefore, market pressure will likely result in a system that is 10x worse than today's extreme-scale systems because of the increased node (and hence component) count.

With 10x, localized checkpointing techniques (such as LLNL's SCR[15]) may be sufficient. As long as users move to a standardized API for user-level checkpointing, these techniques would be comparatively non-invasive since most user codes already understand the importance of defensive I/O (and message logging/replay techniques are transparent to the application).

HPC traditionally acquires network switches, disks, etc from a marketplace that isn't as focused on reliability. We still need a better understanding of the reliability cost trade-offs of these choices. An MTTI of 1 day is achievable for an Exascale system in the 2018 timeframe if the FIT rate *per node* (Failures in time per billion hours of operation for transient uncorrectable errors) stays constant.

8.1 Hard (Permanent) Errors

Hard errors, which are also known as permanent errors, depend on a different mitigation strategy than soft errors. Hard errors might be partly accommodated by incorporating redundant or spare components. For example, building extra cores into a processor chip that can be pressed into service to replace any failed processors on chip. System on Chip designs, described in the Node Architecture section above, can greatly reduce the hard-error rate by reducing the number of discrete chips in the system. Both sockets and solder-joints are a large source of hard-failures – both of which are minimized if all peripheral components are integrated onto a single chip. This approach has been employed successfully on BlueGene systems to achieve a 10-15x lower hard-error rate than conventional clusters.

8.2 Soft (Transient) Errors

The soft (transient) error rate refers to transient errors that affect the Mean time between application interruption (MTTI). The MTTI is any failure that requires application remedial action as opposed to errors that are hidden from the application by resilience mechanism in the hardware or the system software. The MTTI can be much better, using mechanisms a supplier can provide.

It can be useful if the application does some self-checking with a common API to facilitate error detection. Defining a common API for error detection and resilience would help provide uniformity of semantics and innovation of mechanism across multiple vendor platforms. Software approaches for managing error detection and resilience can reduce dependence on hardware checking mechanisms, which can save on power and cost of the system. For example a code could run duplex calculations to self-check could be alternative approach to error detection.

8.3 Node Localized Checkpointing

Localized checkpointing to node-integrated non-volatile storage can accommodate O(10 day) uncorrectable soft errors, but failure characteristics of nonvolatile node-localized storage must be far lower that current commodity parts would support. Using increased redundancy and extensions to Reed-Solomon error correction encodings could make high-volume commodity NVRAM components suitable for node-localized checkpointing.

9 Interconnection Networks

The path towards realizing next-generation petascale and exascale computing is increasingly dependent on building supercomputers with unprecedented numbers of processors. To prevent the interconnect from dominating the overall cost of these ultra-scale systems, there is a critical need for scalable interconnects that capture the communication requirements of ultrascale applications. Future computing systems must rely on development of interconnect topologies that efficiently support the underlying applications' communication characteristics. It is therefore essential to understand high-end application communication characteristics across a broad spectrum of computational methods, and utilize that insight to tailor interconnect designs to the specific requirements of the underlying codes.

9.1 Topology

Throughout the 1990's and early 2000's, high performance computing (HPC) systems implementing *fully-connected networks* (FCNs) such as fat-trees and crossbars have proven popular due to their excellent bisection bandwidth and ease of application mapping for arbitrary communication topologies. However, as supercomputing systems move towards tens or even hundreds of thousands of nodes, FCNs quickly become unfeasibly expensive in terms of wiring complexity, power consumption, and cost[15]. The two leading approaches discussed at the meeting were multi-dimensional Torii and Dragonfly[17] as feasible scalable interconnect topologies. Both approaches present feasible wiring and cost-scaling characteristics for an exascale system. However, it is unclear what portion of scientific computations have communication patterns that can be efficiently embedded onto these types of networks.

The Dragonfly depends on availability of high-radix (radix 64 or greater) router technology to implement a tapered CLOS interconnect topology. The Dragonfly organizes the wiring pattern for the CLOS to localize the high-density wiring within individual cabinets and taper bandwidth for the longer-haul connections. The high-density wiring within a cabinet is amenable to lower-cost copper backplanes to minimize use of

discrete wires. Long-haul connections between cabinets would rely on optical transceivers. The tapering of bandwidth for the long-haul connections keeps wiring complexity & cost within practical limits, and results in power and bisection bandwidth characteristics that are similar to the Torus and hypercube.

Another viable technology option is low-radix torus and hypercube interconnects, which rely on low-degree (6-12 port) routers and exploit spatial locality in application communication patterns. The growth in system parallelism has renewed interest in networks with a lower topological degree, such as mesh and torus interconnects (like those used in the IBM BlueGene and Cray XT series), whose costs rise linearly with system scale. Indeed, the number of systems using lower degree interconnects such as the BG/L and Cray Torus interconnects has increased from 6 systems in the November 2004 list to 58 systems in the more recent Top500 list of June 2009[18]. Although there has been a move towards higher-dimensional torus and hypercube networks, in the 2018 timeframe computing system designs may be forced back towards lower-dimensional (4D or 3D) designs in order to keep control of wiring complexity & wire lengths (maximizing the use of wire paths that can be embedded into board designs.

9.2 Effect of Interconnect Topology on Interconnect Design

Practical wiring, cost and power constraints force us away from fully-connected networks. Both networks (Dragonfly[17] and Torus), will require algorithms and other support software that are more aware of the underlying network topology to make the most efficient use of the available network bandwidth at different levels of the network hierarchy. Both networks have similar bisection bandwidth characteristics when compared with similar link performance and message injection bandwidth.

10 Conclusions

Addressing the technology challenges discussed in this report and accelerating the pace of technology development will require focused investments to achieve Exascale computing by 2018. Achieving an Exascale level of performance by the end of the decade will require applications to exploit on the order of a billion-way parallelism provided by an envisioned exascale system. This is in sharp contrast to the approximately quarter million-way parallelism in today's petascale systems. Node architectures are expected to change dramatically in the next decade as power and cooling constraints limit increases in microprocessor clock speeds. Consequently computer companies are dramatically increasing on-chip parallelism to improve performance. The traditional doubling of clock speeds every 18-24 months is being replaced by a doubling of cores, threads or other parallelism mechanisms. Exascale systems will be designed to achieve the best performance within both power and cost constraints. In addition, hardware breakthroughs will be needed to achieve useful exascale computing later this decade, at least within any reasonable power budget. Applications and algorithms will need to change and adapt as node architectures evolve. They will need to manage locality and perhaps resilience to achieve high performance. A key element of the strategy as we move forward is the co-design of applications, architectures and programming environments as shown in Figure 1. Much greater collaboration between these communities

will be needed to overcome the key Exascale challenges. There is an unprecedented opportunity for application and algorithm developers to influence the direction of future architectures to reinvent computing for the next decade.

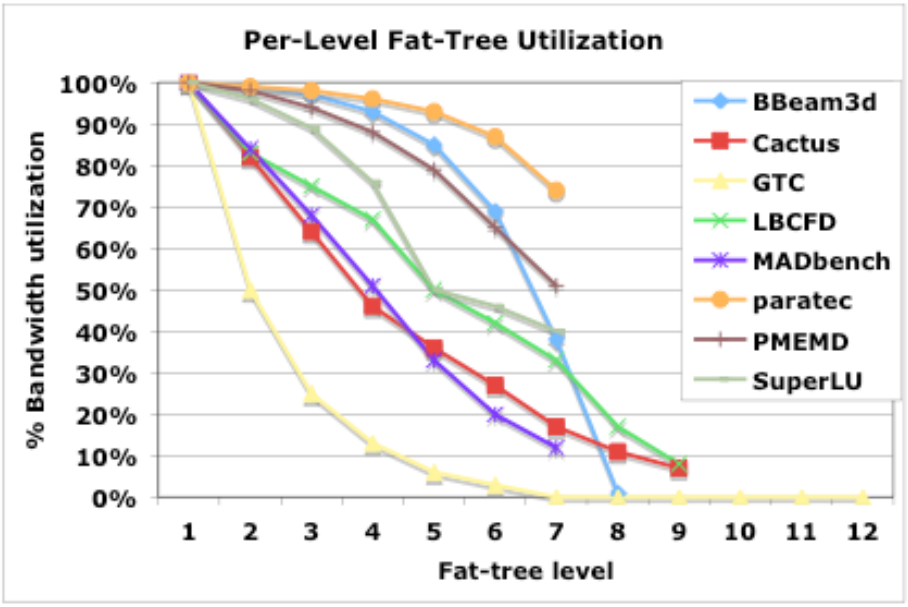


Fig. 10. This figure illustrates the bandwidth tapering characteristics of the communication patterns of 8 key DOE applications when mapped optimally to a multi-layer hierarchical network. Many applications do not fully utilize the upper-layers of the interconnect, meaning that full bisection is not required. [15]

Table 1. Overview of technology scaling for exascale systems. Swimlane 1 represents an extrapolation of manycore system design point whereas swimlane 2 represents scaling of a GPU design point.

Systems	2009	2018 Swimlane 1	2018 Swim-Lane 2
System peak	**2 Peta**	**1 Exa**	**Same as Swim-lane 1**
Power	**6 MW**	**~20 MW**	**Same as SL1**
System memory	0.3 PB	32 - 64 PB	Same as SL1
Node performance	125 GF	1,2TF	10TF

Table 1. (*continued*)

Systems	2009	2018 Swimlane 1	2018 Swim-Lane 2
Interconnect Latency (for longest path)	1-5usec (limited by overhead at endpoints)	0.5-1usec (speed of light)	Same
Memory Latency	150-250 clock cycles (~70-100ns)	100-200 clock cycles (~50ns)	same
Node memory BW	25 GB/s	0.4TB/s	4-5TB/s
Node concurrency	12	O(1k)	O(10k)
Total Node Interconnect BW	3.5 GB/s	100-400GB/s (1:4 or 1:8 from memory BW)	2TB/s
System size (nodes)	18,700	O(1M)	O(100,000)
Total concurrency	225,000	O(100M)*10 for latency hiding	O(100M)*100 for latency hiding
Storage	15 PB	500-1000 PB (>10x system memory is min)	Same as SL1
IO	0.2 TB	60 TB/s	Same as SL1

References

[1] DOE E3 Report, http://www.er.doe.gov/ascr/ProgramDocuments/ProgDocs.html
[2] A Platform Strategy for the Advanced Simulation and Computing Program (NA-ASC-113R-07-Vol. 1-Rev. 0)
[3] DARPA Exascale Computing Study (TR-2008-13), http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf
[4] Miller, D.A., Ozaktas, H.M.: Limit to the bit-rate capacity of electrical interconnects from the aspect ratio of the system architecture. J. Parallel Distrib. Comput. 41(1), 42–52 (1997), DOI http://dx.doi.org/10.1006/jpdc.1996.1285
[5] Miller, D.A.B.: Rationale and challenges for optical interconnects to electronic chips. Proc. IEEE, 728–749 (2000)
[6] Horowitz, M., Yang, C.K.K., Sidiropoulos, S.: High-speed electrical signaling: Overview and limitations. IEEE Micro. 18(1), 12–24 (1998)
[7] IAA Interconnection Network Workshop, San Jose, California, July 21-22 (2008), http://www.csm.ornl.gov/workshops/IAA-IC-Workshop-08/

[8] Architectures and Technology for Extrame Scale Computing Workshop, San Diego, California, December 8-10 (2009), http://extremecomputing.labworks.org/hardware/index.stm
[9] Asanovic, K., et al.: The Landscape of Parallel Computing Research: A View from Berkeley, Electrical Engineering and Computer Sciences. University of California at Berkeley, Technical Report No. UCB/EECS-2006-183, December 18 (2006)
[10] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., Hanrahan, P.: Larrabee: a many-core x86 architecture for visual computing. ACM Trans. Graph. 27(3), 1–15 (2008)
[11] Liu, Y., Zhu, H.: A survey of the research on power management techniques for high-performance systems. Softw. Pract. Exper. 40(11), 943–964 (2010)
[12] Colmenares, J.A., Bird, S., Cook, H., Pearce, P., Zhu, D., Shalf, J., Hofmeyr, S., Asanović, K., Kubiatowicz, J.: Resource Management in the Tesselation Manycore OS. In: HotPar 2010, Berkeley (2010), http://www.usenix.org/event/hotpar10/final_posters/Colmenares.pdf
[13] U.S. Department of Energy, DOE Data Center Energy Efficiency Program (April 2009)
[14] Moody, A., Bronevetsky, G., Mohror, K., de Supinski, B.R.: Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. In: IEEE/ACM Supercomputing Conference (SC) (November 2010)
[15] Kamil, S., Oliker, L., Pinar, A., Shalf, J.: Communication Requirements and Interconnect Optimization for High-End Scientific Applications. IEEE Transactions on Parallel and Distributed Systems (2009)
[16] Balfour, J., Dally, W.J.: Design tradeoffs for tiled CMP on-chip networks. In: Proceedings of the 20th Annual International Conference on Supercomputing, ICS 2006, Cairns, Queensland, Australia, June 28-July 01, pp. 187–198. ACM, New York (2006)
[17] Kim, J., Dally, W., Scott, S., Abts, D.: Cost-Efficient Dragonfly Topology for Large-Scale Systems. IEEE Micro. 29(1), 33–40 (2009)
[18] Top500 List Home, http://www.top500.org/
[19] Hayt, W.H.: Engineering Electromagnetics, 7th edn. McGraw Hill, New York (2006)
[20] Guha, B., Kyotoku, B.B.C., Lipson, M.: CMOS-compatible athermal silicon microring resonators. Optics Express 18(4) (2010)
[21] Hendry, G., Chan, J., Kamil, S., Oliker, L., Shalf, J., Carloni, L.P., Bergman, K.: Silicon Nanophotonic Network-On-Chip Using TDM Arbitration. In: IEEE Symposium on High Performance Interconnects (HOTI) 5.1 (August 2010)