The New Landscape of Parallel Computer Architecture

John Shalf
NERSC Division, Lawrence Berkeley National Laboratory
1 Cyclotron Road, Berkeley California, 94720, USA

E-mail: jshalf@lbl.gov

Abstract. The past few years has seen a sea change in computer architecture that will impact every facet of our society as every electronic device from cell phone to supercomputer will need to confront parallelism of unprecedented scale. Whereas the conventional multicore approach (2, 4, and even 8 cores) adopted by the computing industry will eventually hit a performance plateau, the highest performance per watt and per chip area is achieved using manycore technology (hundreds or even thousands of cores). However, fully unleashing the potential of the manycore approach to ensure future advances in sustained computational performance will require fundamental advances in computer architecture and programming models that are nothing short of reinventing computing. In this paper we examine the reasons behind the movement to exponentially increasing parallelism, and its ramifications for system design, applications and programming models.

1. Introduction

The past few years has seen a sea change in computer architecture that will impact every facet of our society as every electronic device from cell phone to supercomputer will need to confront parallelism of unprecedented scale. Whereas the conventional multicore approach (2, 4, and even 8 cores) adopted by the computing industry will eventually hit a performance plateau, the highest performance per watt and per chip area is achieved using manycore technology (hundreds or even thousands of cores). However, fully unleashing the potential of the manycore approach to ensure future advances in sustained computational performance will require fundamental advances in computer architecture and programming models that are nothing short of reinventing computing.

Recent trends in the microprocessor industry have important ramifications for the design of the next generation of High Performance Computing (HPC) systems as we look beyond the petaflop scale. The need to switch to a geometric growth path in system concurrency is leading to reconsideration of interconnect design, memory balance, and I/O system design that will have dramatic consequences for the design of future HPC applications and algorithms. The required reengineering of existing application codes will likely be as dramatic as the migration from vector HPC systems to Massively Parallel Processors (MPPs) that occurred in the early 90's. Such comprehensive code reengineering took nearly a decade, so there are serious concerns about undertaking yet another major transition in our software infrastructure.

A group of University of California researchers with backgrounds ranging from circuit design, computer architecture, CAD, embedded hardware/software, programming languages, compilers, applied math, to HPC, met for a period of two years to consider how current constraints on device physics at the silicon level would affect CPU design, system architecture, and programming models for future systems. The results of the discussions are documented in the University of California Berkeley Technical Report entitled "The Landscape of Parallel Computing Research: A View from Berkeley"[1]. Whereas that report was concerned mostly with the broader microprocessor and consumer electronics industry, this paper applies the reasoning laid out in the "Berkeley View" to the future of HPC system design.

This paper explores the fundamental device constraints that have led to the recent stall in CPU clock frequencies. It examines whether multicore (or manycore) is in fact a reasonable response to the underlying constraints to future IC designs. Then it explores the ramifications of these changes in the context of computer architecture, system architecture, and programming models for future HPC systems.

2. The End of Traditional Sources for CPU Performance Improvements

One important discontinuity that has developed in system architecture is motivated by changes in device physics below the 90nm scale. The changes at the chip level create a cascade of design choices that affect every aspect of system design and result in some of the most daunting challenges for software design on future trans-petaflop systems.

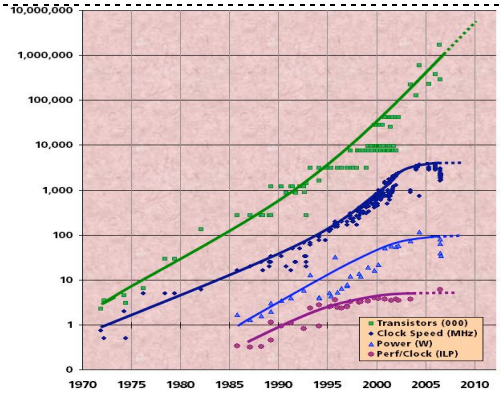

Figure 1. This graph (from Kunle Olukotun and Herb Sutter) shows that Moore's law is alive and well, but the traditional sources of performance improvements (ILP and clock frequencies) have all been flattening.

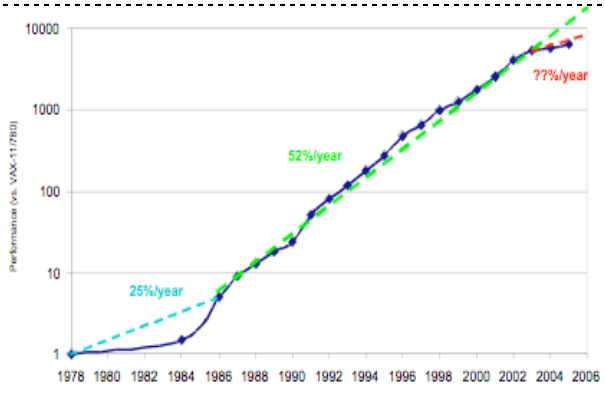

Figure 2. The performance of processors as measured by SpecINT has grown 52% per year with remarkable consistency, but improvements tapered off around 2003. [2]

Figure 1 shows that Moore's law, which states that you can integrate twice as many components onto an integrated circuit every 18 months at fixed cost, still holds. However, the traditional sources of performance improvements such as instruction level parallelism (ILP) and clock frequency scaling have been flattening since 2003. Figure 2 shows the improvements in processor performance as measured by the SPEC benchmark over the period from 1975 to present. Since 1986 performance has improved by 52 percent per year with remarkable consistency. During that period, as process geometries scaled according to Moore's law, the active capacitance of circuits scaled down so that supply voltages could be kept constant or even dropped modestly in order to allow manufacturers to increase clock-speeds. This

approach, known as "constant electric field" frequency scaling [3] fed the relentless increases in CPU clock-rates over the past decade and a half. However, below the 90nm scale for silicon lithography, this technique began to hit its limits as static power dissipation from leakage current began to surpass dynamic power dissipation from circuit switching. Power density has now become the dominant constraint in the design of new processing elements, and ultimately limits clock-frequency growth for future microprocessors. The direct result of power constraints has been a stall in clock frequency that is reflected in the flattening of the performance growth rates starting in 2002. In 2006, individual processor cores are nearly a factor of three slower than if progress had continued at the historical rate of the preceding decade. Other approaches for extracting more performance such as Instruction Level Parallelism (ILP) and out-of-order instruction processing have also delivered diminishing returns (see Figure 1). Having exhausted other well-understood avenues to extract more performance from a uniprocessor, the mainstream microprocessor industry has responded by halting further improvements in clock frequency and increasing the number of cores on the chip. Patterson and Hennessy estimate the number of cores per chip is likely to double every 18-24 months henceforth. Therefore, new algorithms and programming models will need to stay ahead of a wave of geometrically increasing system concurrency – a tsunami of parallelism.

The stall in clock frequencies and the industry's comparatively straightforward response of doubling cores has reinvigorated study of more radical alternative approaches to computing such as Field Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPU), and even dataflow-like tiled array architectures such as TRIPS [4]. The principle impediment to adapting a more radical approach to hardware architecture is that we know even less about how to program efficiently such devices for diverse applications than we do parallel machines composed of multiple CPU cores. Kurt Keutzer has put this more elegantly when he states "The shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures." To get at the heart of Keutzer's statement, it is necessary to deconstruct the most serious problems with current CPU core designs.

2.1. New IC Design Constraints

As mentioned in the previous section, due to the leakage current, a number of recent industry studies have concluded that power dissipation ultimately limits continued performance improvements based on clock frequency scaling [3][5]. Some chip designs, such as the Intel Tejas, were cancelled due to power consumption issues [6]. The other problem facing the industry is the extremely high cost of new logic designs. In order to squeeze more performance out of chips at high frequencies, a considerable amount of surface area is devoted to latency hiding technology such as deep execution pipelines and out-of-order instruction processing. The phenomenally complex logic required to implement such features has caused design costs of new chips to skyrocket. It has also become impractical to verify all logic on new chip designs containing hundreds of millions of logic gates due to the combinatorial nature of the verification process. Finally, with more complex chip logic designs, a single defect on the chip surface can render the entire CPU core non-functional. As we move to smaller feature sizes for future chips, the likelihood of such defects will continue to increase – dramatically lowering chip yields. These key problems ultimately conspire to limit the performance and practicality of extrapolating past design techniques to future chip designs, regardless of whether the logic implements some exotic non-von Neumann architecture or a more conventional approach.

The following remedies represent the consensus opinion of the contributors to the "Berkeley View" report.

- Power: Parallelism is an energy efficient way to achieve performance [7]. Many simple cores offer higher performance per unit area for parallel codes than a comparable design employing smaller numbers of complex cores.
- Design Cost: The behavior of a smaller, simpler processing element is much easier to predict within existing electronic design-automation workflows and more amenable to formal verification. Lower complexity makes the chip more economical to design and produce. [8]
- Defect Tolerance: Smaller processing elements provide an economical way to improve defect tolerance by providing many redundant cores that can be turned off if there are defects. For example, the Cisco Metro chip [9] has 4 redundant processor cores per die. The STI Cell processor has 8 cores, but only 6 are enabled in its mainstream consumer application in the Sony Playstation 3 in order to provide additional redundancy to better tolerate defects.

2.2. Response to Constraints: Manycore vs. Multicore

The new industry buzzword "multicore" captures the plan of doubling the number of standard cores per die with every semiconductor process generation starting with a single processor. Multicore will obviously help multiprogrammed workloads, which contain a mix of independent sequential tasks, and prevents further degradation of individual task performance. But how will individual tasks become faster? Switching from sequential to modestly parallel computing will make programming much more difficult without rewarding this greater effort with a dramatic improvement in power-performance. Hence, multicore is unlikely to be the ideal answer.

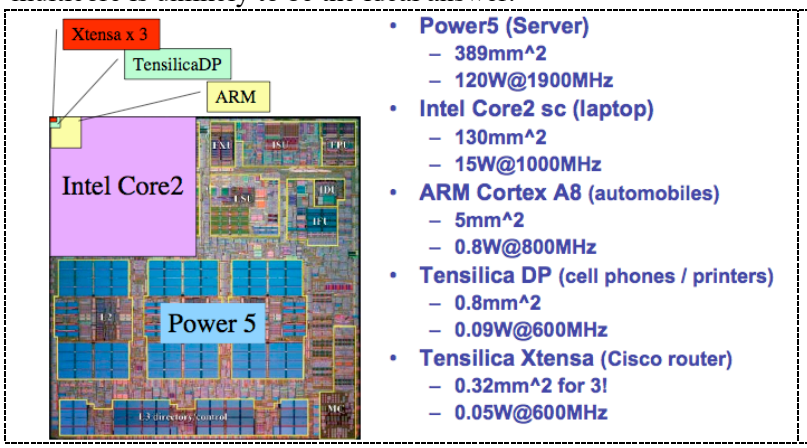

Figure 3. The diagram to the left shows the relative size and power dissipation of different CPU core architectures. Simpler processor cores require far less surface area and power with only a modest drop in clock frequency. Even if measured by sustained performance on applications, the power efficiency and performance per unit area is significantly better when using the simpler cores.

The alternative approach moving forward is to adopt the 'manycore' trajectory, which employs simpler cores running at modestly lower clock frequencies. Rather than progressing from 2 to 4, to 8 cores with the multicore approach, a manycore design would start with hundreds of cores and progress geometrically to thousands of cores over time. Figure 3 shows that moving to a simpler core design results in modestly lower clock frequencies, but has enormous benefits in power consumption and chip surface area. Even if you presume that the simpler core will offer only 1/3 the computational efficiency of the more complex out-of-order cores, a manycore design would still be an order of magnitude more power and area efficient in terms of sustained performance.

2.3. Convergence Between High End Servers and Consumer Electronics Core Design

The approach of using simpler lower-frequency core designs has been used for many years by the embedded-computing industry to improve battery life, lower design costs, and reduce time to market for consumer electronics. In the past the design targets for embedded applications were nearly opposite of the performance-

driven requirements of high-end computing. However, the needs of the high-end-computing market have converged with the design motivation of the embedded-computing industry as a result of their common need to improve power efficiency and reduce design costs. With regard to power efficiency, the embedded-computing industry has the most accumulated expertise and technology. Whereas past design innovations in high-end-computing, such as superscalar execution and out-of-order instruction pipelines, trickled down to consumer applications on PCs, we are starting to see innovations that emerged in the embedded space trickling up into high-end-server designs. This flow of innovation is likely to increase in the future.

There are already examples of the convergence between embedded computing and HPC in the design of the BlueGene and SiCortex supercomputers, which are based on embedded processor cores that are more typically seen in automobiles, cell-phones and toaster ovens. Parallelism with concurrencies that have formerly been associated with HPC applications are already emerging in mainstream embedded applications. The Metro chip in new Cisco CRS-1 router contains 188 general-purpose Tensilica cores, and has supplanted Cisco's previous approach of employing custom Application Specific Integrated Circuits (ASICs) for the same purpose [9]. Surprisingly, the performance and power efficiency of the Metro are far higher than could be achieved using FPGAs (dimming hopes that FPGAs offer a more power efficient approach to computation). The Motorola Razor cell phone also contains 8 Tensilica cores. The NVidia G80 (CUDA) GPU replaces the semi-custom pipelines of previous generation GPUs with 128 more general-purpose CPU cores.

The G80 in particular heralds the convergence of manycore with mainstream computing applications. Whereas traditional GPGPUs have a remarkably obtuse programming model involving drawing an image of your data to the framebuffer (the screen), the G80's more general purpose cores can be programmed using more conventional C code and will soon support IEEE standard double precision arithmetic. The motivation for the more general-purpose cores is the increasing role of GPUs for accelerating commonly required non-graphical game calculations such as AI, object interactions, and even models for physical processes. Companies such as AISeek's Intia (http://www.aiseek.com/index.html) and Ageia's PhysX (http://www.ageia.com/) have implemented game physics acceleration on GPUs that use algorithms that are very similar to those used for scientific simulation and modelling. ATI (recently acquired by AMD) will be offering GPUs that share the same cache-coherent HyperTransport fabric of their mainstream CPUs. Intel's experimental Polaris chip uses 80 simpler CPU cores on a single chip to deliver 1 Teraflop of peak performance while consuming only 65 Watts [10] and may feed into future GPU designs from Intel. Both Intel and AMD roadmaps indicate that tighter integration between GPUs and CPUs is the likely path toward introducing manycore processing to mainstream consumer applications on desktop and laptop computers.

2.4. Consumer Electronics Supplanting Desktop PC as Driver for CPU Innovation

Taking a historical perspective on HPC system design, Bell's Law is likely as important to the development of HPC system design as Moore's law. Moore's law says that you can double the number of components that can be integrated onto a chip every 18 months for the same cost. Bell's law is the corollary that says that by maintaining the same design complexity you can halve the size and costs of the chip every 18 months. The Top500 project [11] has documented a steady progression from the early years of supercomputing, from exotic and specialized designs towards clusters composed of components derived from consumer PC applications. The enormous volumes and profits of desktop PC technology led to huge cost/performance benefits for employing clusters of desktop CPU designs for scientific applications despite the lower computational efficiency. As we move in to the new century, the center of gravity for the market in terms of unit volume and profit has shifted to handheld consumer electronics. This movement has some significant implications for the future of the HPC.

Figure 4 shows the market share of the consumer electronics applications for CPUs surpassed that of the desktop PC industry in 2003. Shortly thereafter, revenue in the PC business flattened (Figure 5), and IBM subsequently divested itself of its desktop and portable personal computer units. During that same period of time, Apple moved into the portable electronic music player market with the iPod and more recently into the cellular phone business. This may be the most dramatic twist yet for the HPC industry if these trends continue (and they likely will). Although the desktop computing industry has been leading a major upheaval in HPC system design over the past decade (the movement to clusters based on desktop technology), that industry segment is no longer in the driver's seat for the next revolution in CPU design. Rather, the market volume, and hence design targets, are now driven by the needs of handheld consumer electronics such as digital cameras, cell phones, and other embedded devices.

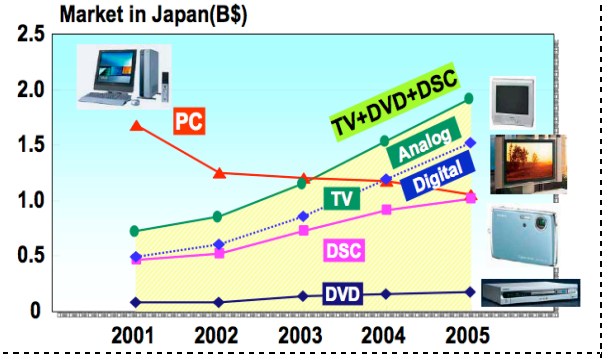

Figure 4. The graph above shows the declining influence of PC microprocessors on the overall revenue share in microprocessor-based electronic designs (from Tsugio Makimoto: JEITA).

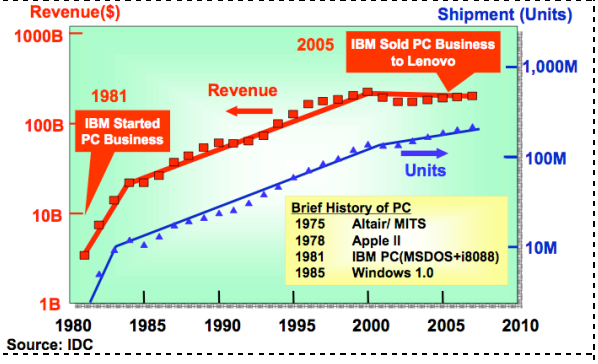

Figure 5. This graph shows how revenues for IBM's PC business have flattened in response to the increasing dominance of consumer electronics applications in the electronics industry (from Tsugio Makimoto: from IDC)

The next generation of desktop systems, and consequent HPC system designs, will borrow many design elements and components from the consumer electronics devices. Namely, the base unit of integration for constructing new chip designs targeted at different applications will be the CPU cores derived from embedded applications rather than the transistor. Simpler CPU cores may be combined together with some specialization to HPC applications (eg. different memory controllers and double precision floating point), just as transistors are currently arranged to form current CPU designs. Indeed, within the next 3 years, there will be manycore chip designs that contain greater than 2000 CPU cores, which is very close to the number of transistors that we used in the very first Intel 4004 CPU. This led Chris Rowen, from Tensilica, to describe the new design trend by saying "The processor is the new transistor". This is a brave new world, and we don't know where it will take us.

3. Ramifications for the HPC Ecosystem

Given current trends, petaflop scale systems in 2011 are projected to have the following characteristics.

- Systems will contain between 200,000 and 1,500,00 processors (50,000 to 200,000 "sockets"). Each "socket" in the system will be a chip that contains multiple cores (a Chip MultiProcessor or CMP).
- In 2011 these multicore chips will contain between 8 and 32 conventional processor cores per chip. Technology based on "manycore" will employ 100s to 1000s of CPU cores per chip. Consider that the Cisco CRS-1 router currently employs a chip containing 188 processor cores

using conventional 90nm process technology, so "1000 cores on a chip" is not as far away as one might expect. A System on Chip (SoC) design, such as BlueGene or SiCortex, may still use the simpler embedded cores (manycore design point), but sacrifice raw core count to integrate more system services onto the chip (eg. interconnect fabric, memory controllers).

- As microprocessor manufacturers move their design targets from peak-clock-rate to reducing power consumption and packing more cores per chip, there will be a commensurate trend towards simplification of the core design. This is already evidenced by the architecture of the Pentium-M derived Intel Core-Duo processors that utilize pipelines that are half the length of its predecessor, the Pentium4. The trend towards simpler processor cores will likely simplify performance tuning, but will also result in lower sustained performance per core as out-of-order instruction processing is dropped in favor of smaller, less-complex less-power-hungry in-order core designs. Ultimately this trend portends a convergence between the manycore and multicore design points.
- As the number of cores per socket increase, memory will become proportionally more expensive and power hungry in comparison to the processors. Consequently, cost and power-efficiency considerations will push memory balance (in terms of the quantity of memory put on each node) from the current nominal level of 0.5 bytes of DRAM memory per peak flop, down below 0.1 bytes/flop (possibly even less than 0.02 bytes/flop).
- Cost scaling issues will force fully-connected interconnect topologies such as the fat-tree and crossbar to be gradually supplanted at the high-end by lower-degree interconnects such as the n-dimensional torii, meshes or alternative approaches such as hierarchical fully-connected graphs. The consequence will be that HPC software designers must take interconnect topology and associated increased non-uniformity in bandwidth and latency into account for both algorithm design and job mapping. At this point in time, the interconnect topology is typically ignored by mainstream code and algorithm implementations. BlueGene/L programmers already have to grapple with the topology mapping issues – it is merely a harbinger of the broader challenges facing programmers all HPC programmers at some point in the near future.

Whether you are convinced or not that the future is in 'multicore' or 'manycore' technology, the industry has already retooled to move in the direction of geometrically scaling parallelism. If you think the levels of concurrency that result from a transition directly to 'manycore' are daunting, the trends on the Top500 list in Figure 6 show that you need only wait 3 years before 'multicore' will take you to the same breathtaking levels of parallelism. With either path, the HPC community faces an unprecedented challenge to existing design practices for system design, OS design, and our programming model. The following sections examine the following concerns about how the multicore approach destabilizes practices that have been established and solidified over the past 15 years of relative stability in the HPC ecosystem.

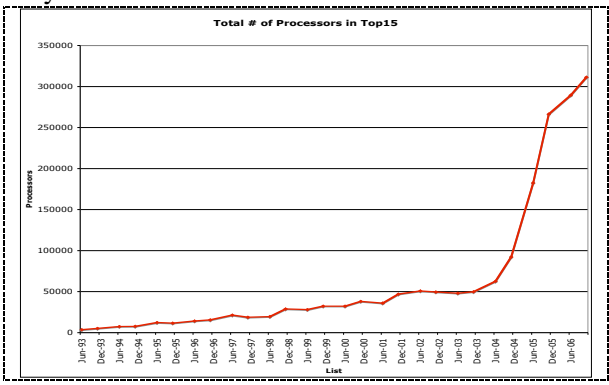

Figure 6. The graph shows the dramatic increase in system concurrency for the Top 15 systems in the annual Top500 list of HPC systems. Even if BG/L systems are removed from consideration, the inflection point of system concurrency is still quite dramatic.

The concerns are summarized as follows;

- System Balance: As we move to integrate more CPU cores on a chip, future system designs will become increasingly unbalanced with respect to memory and interconnect performance.
- Reliability: As we move to higher concurrencies, the systems will become increasingly prone to failures.
- Programmability: How on earth will we stay abreast of exponentially increasing parallelism? If we cannot program a manycore computer system productively, efficiently, and correctly, then there is little benefit to this approach for maintaining historical per-socket performance improvements for future HPC systems.

This rest of this section will examine these three areas of concern.

3.1. System Balance

One of the very first concerns that arise when considering manycore designs is that packing so many cores onto a single chip will destabilize system balance as described by the Amdahl ratios [12]. Namely, the memory bandwidth will be so constrained as to completely eliminate the benefit of adding additional cores to the chip (the so called "memory wall") [13].

3.1.1. Memory Balance

While the memory wall is certainly a worthy concern, it is important to note that it isn't a new concern. The term was first coined in 1992 to describe the effects of clock frequency scaling -- multicore simply continues this historical trend along the same lines. Unlike the historical trend, the stall in clock frequencies has halted the latency gap given that chip clock frequencies are no longer growing at historical rates. Finally, we note that GPUs and the Cisco Metro chip show that even today's technology can deliver 12 times more bandwidth onto the chip than currently used for mainstream processor designs. We choose not to use high bandwidth memory in desktop systems due to cost considerations. If bandwidth would ultimately limit performance, it is possible to apportion more of the cost of a new system design to the cost of the memory.

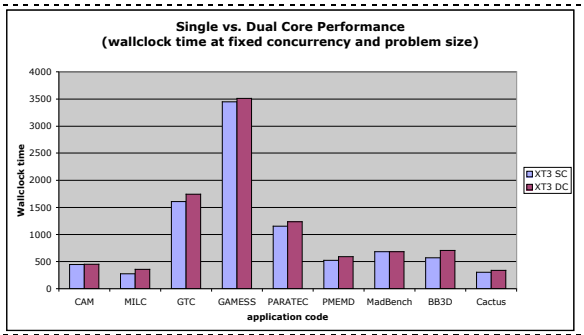

Figure 7. This graph shows the wallclock time consumed by full applications run at their native concurrency on the Cray XT3 sited at Oak Ridge National Laboratory. The blue bar shows the wallclock time when running on one processor per socket and the red bar shows the time spent running on both processors in each socket at the same concurrency.

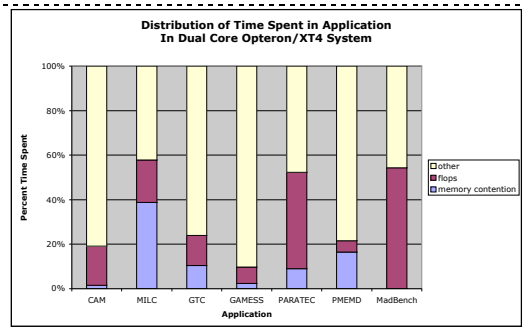

Figure 8. Thee graph above shows a breakdown of where time was spent in a subset of those application codes. Memory bandwidth contention is not as significant a factor as the "other" category, which includes integer operations and latency stalls.

A recent joint study with the National Energy Research Supercomputing Center (NERSC) and Cray Inc. [14] examined the current dual-core AMD processor on Cray XT3 and XT4 systems to assess the current state of system balance and to determine when to invest more of our monetary resources to improve memory bandwidth. The study used the NERSC SSP [15], which is a diverse array of full-scale applications that represent a significant fraction of the NERSC workload. As shown in Figure 7, the surprising result is that application performance showed very little degradation when moving from single-core to dual core (running same concurrency, but using one verses two cores per socket). The memory bandwidth seen by the application was effectively halved, but the average performance impact was less than 10%! This was certainly not what one would expect based on the conventional wisdom regarding the necessity of bandwidth for HPC applications.

Examining a breakdown of time spent in various components of the code (Figure 8), very little time could be attributed to memory contention. The largest fraction of time is attributed to either latency stalls or integer/address arithmetic (the "other" category). So, although in an ideal world, these applications should be memory-bandwidth bound, existing CPUs are constrained by other microarchitectural bottlenecks. Multicore can actually help in this area by balancing Little's Law, which states that the number of concurrent memory accesses in flight at any given time must equal the product of latency and bandwidth of the memory interface in order to ensure available memory bandwidth is utilized. Multicore is more effective than scaling the clock frequency of uniprocessors because it is capable of using the cores to launch more concurrent accesses to memory. Vector architectures were effective at using memory bandwidth for similar reasons as the vector register loads greatly increased the number of concurrent load operations presented to the main memory.

Perhaps more chip real-estate should be devoted to latency hiding features such as vector-like register sets or software controlled memories. Early data collected on the STI Cell processor indicates that the leading reason for its high performance relative to other CPU implementations can be traced back to the efficiency of its software managed memory hierarchy rather than the high FLOP rates and memory bandwidth the platform offers [16]. At the moment multicore is not memory bandwidth bound, but ideally it *should* be!

3.1.2. Interconnects

The other concern that arises as we move toward the phenomenal concurrencies shown in Figure 6 is that the interconnect topologies will be inadequate to the task of handling the communication requirements of applications. It is becoming impractical to build fully-connected networks such as crossbars or CLOS networks for ever-increasing concurrencies because such implementations have component costs that scale exponentially with port count. Lower degree interconnects such as Torii and hypercubes are characterized by linear scaling in component costs, but application performance on these networks may ultimately become constrained by the limited bisection bandwidth. In order to better understand the application communication requirements, we have embarked on a 3-year study of application communication requirements documented in a series of papers [17][18][19].

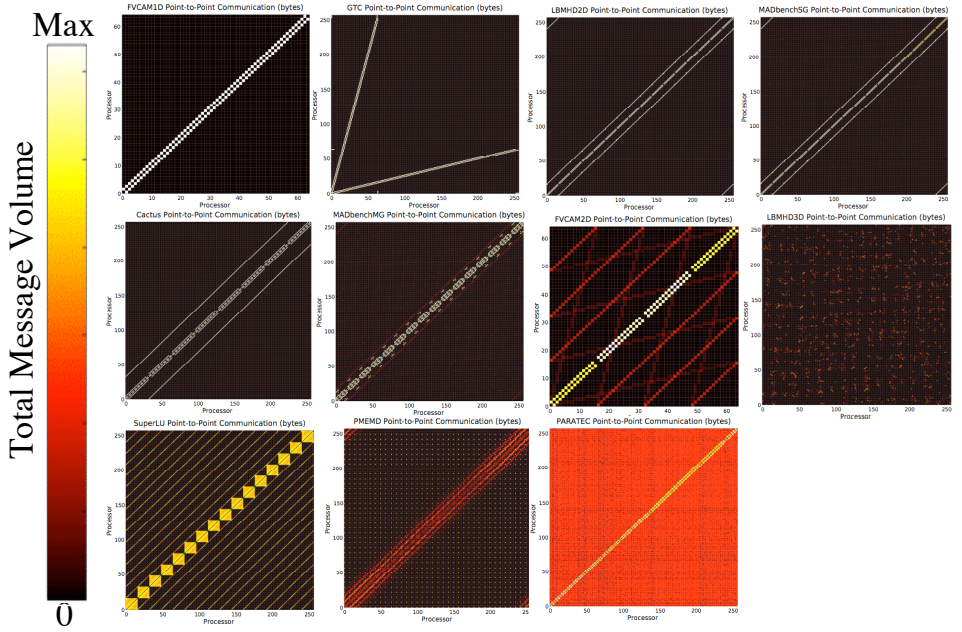

Figure 9. Communication Requirements for HPC Applications [18].

Figure 9 shows the communication patterns of a number of major Department of Energy (DOE) HPC applications captured non-invasively by IPM [20] as they ran at scale on the IBM SP at NERSC. The matrix of cells describe the communication volume between processors over the course of code execution. The conclusion of these studies was that the point-to-point messages tended to be very large (bandwidth bound), but the pattern of communication was very sparse (typically less than 12 communicating partners for many applications). Although the bandwidth requirements for the interconnect remain high, an interconnect topology supporting a large bisection bandwidth would be overkill for most applications. Even for the 3D FFT (bottom right matrix in Figure 9), which requires a bisection-bandwidth-limited transpose, as you move to a 2-dimensional domain decomposition required for high concurrency, the number of communicating partners reduces from P processes to the square root of P processes – thereby relaxing the bisection bandwidth constraints. Also, the FFT tends to use messages so small the ability to efficiently overlap computation with communication using very low-overhead one-sided messages is very effective at mitigating the effects of a low bisection bandwidth interconnect [21].

The second finding was that the collective communications used for synchronization and global reductions is strongly bound by latency. Rather than routing collective operations over the same interconnect used for high-bandwidth point-to-point messaging, it may be more effective to use a dedicated interconnect for latency-sensitive operations. This approach has proven useful on the Thinking Machines CM-5 and IBM BG/L systems.

These very same considerations will be as important for the design of interconnects between CPUs on a chip as they are for interconnections between chips in a large scale MPP. Once chip architectures move to 1000s of cores, the conventional crossbar switch will no longer be feasible for on-chip interconnects, and the same issues for mapping application communication patterns to lower-degree communication topologies will arise. Ultimately, future applications will need to be increasingly aware of the interconnect

topology and resulting hierarchical communication characteristics – both on-chip and off-chip. It may be possible to employ circuit switches to adapt the communication topology to application requirements, or alternatively the OS could be involved in intelligent task migration so as to map the communication topology as closely as possible to the constrained topology of the interconnect.

3.2. Reliability

It is generally agreed that as feature sizes shrink down to atomic scale, there will be an increasing number of both hard and soft errors in chip design [22][23]. While this is true, the move to multicore has nothing to do with this progression. In fact, the availability of additional redundant cores on chip will make recovery from manufacturing defects more tractable by building in redundant cores that can replace cores that were disabled by manufacturing defects (hard errors). Cores can also run in tandem, in a manner similar to banking systems, to better detect and protect against soft errors. So the manycore appoach offers many paths to actually improve reliability on-chip.

The largest source of chip errors comes from the sockets where IC's are plugged in to system boards. Pin failure rates on sockets tend to be as high or a higher source for failures than soft-errors in many system implementations. The design of BG/L offers some insight into how to keep such points of failure under control. In order to minimize such errors, IBM chose to solder memory onto the system board for the BG/L system in order to improve reliability. In addition, rather than devoting chip surface area to putting more cores on board or more memory, the BG/L ASIC integrates a large number of system services such as the interconnect and memory controller, which would otherwise have been implemented as discrete components on the system board. By employing the SoC (system on chip) integration to reduce the number of components on the board, and by minimizing the number of socketed chip connections in the system, the BG/L has a better MTBAF (Mean Time Between Application Failure) than the ASCI Purple system despite having twelve times as many CPUs. So the benefits of employing simpler core design for the manycore design point are not limited to simply using chip area for more CPUs – using chip surface area for increased system integration (and fewer components) can greatly increase the cost-effectiveness and reduce failure rates for future HPC system implementations.

3.3. Programmability

Keeping abreast of geometrically increasing concurrency is certainly the most daunting challenge as we move beyond the petaflop in the next few years. Most algorithms and software implementations are built fundamentally on the premise that concurrencies would continue to increase modestly, just as they have over the past 15 years. The exponentially increasing concurrency throws all of those assumptions into question. At a minimum, we will need to embark upon a campaign of reengineering our software infrastructure that is as dramatic and broad in scope as the transition from vector systems to MPPs that occurred in the early 90's. However, the heir apparent to our current programming practice is not obvious. If we cannot succeed in selecting a scalable programming model to carry us for the next 15 years, we will be forced to reckon with multiple phases of ground-up software rewrites, or a hard limit to the useable performance of future system. Consequently, the programming model for manycore systems is the leading challenge for future systems.

3.3.1. Unfamiliar Programming Target

The multicore or manycore CMP is an unfamiliar programming target. The most popular approach to programming multicore systems is to adopt the methods used to program familiar targets such as MPPs or SMPs. The Cray XT4 and BG/L systems are currently programmed using the same flat MPI programming model used for conventional MPPs. However, many datastructures in our application codes

and even the vendor MPI implementations grow O(N) or O(N^2) with the number of processors in the system – ultimately consuming most of the available memory with redundant copies of shared variables or with code segments.

As a result, there has been reconsideration of the hybrid OpenMP+MPI programming model that has shown more failures than successes over the past decade. Even if the deficiencies in implementation that have limited its effectiveness can be overcome, the barrier synchronizations and serialized setup of shared verses thread-private variables that occur at the preamble of OpenMP parallel loop structures will be increasingly stymied by Amdahl's law at higher on-chip concurrencies. Also the SMP (shared memory processor) target of OpenMP has dramatically different characteristics from the CMPs that we are currently considering. Whereas the SMP target for which OpenMP was designed has communication pathways between CPUs that have latencies and bandwidths that are very similar to the communication paths to memory, the CMP target offers 10-100 times lower latencies and 10-100 times higher bandwidths between the CPU cores on chip. OpenMP does not sufficiently exploit the much finer granularity afforded by the CMP design, and failure to adopt an execution model that exploits this feature will fail to fully unlock the potential of CMPs.

Some examples of novel execution models and programming primitives that better exploit the capabilities of the CMP compared to an SMP model are as follows

- CMPs could offer new lightweight coherency and synchronization primitives that only operate between cores on the same chip. Some examples include the streaming directives used to coordinate action between Single-Streaming Processors (SSPs) on the Cray X1, split phase transactions, or fine-grained memory fences.
- Transactional Memory offers a pathway to be more fault resilient and tolerant of programmer errors with respect to reasoning about concurrency and dataflow hazards [24].
- Standard coherence protocols are inefficient for certain data communication patterns (e.g., producer-consumer traffic), and these inefficiencies will be magnified by the increased core count and the vast increase in potential core bandwidth and reduced latency of CMPs. More flexible or even reconfigurable data coherency schemes will be needed to leverage the improved bandwidth and reduced latency.

If we simply treat a multicore chips as a traditional SMP—or worse still, treat it as a flat MPI machine -- then we may miss opportunities for new architectures and algorithm designs that can exploit these new features.

3.3.2. Manycore Opportunities

Most of our consternation regarding the move to multicore reflects our uncertainty that we will be able to extract sufficient parallelism from our existing algorithms. Our community is slowly rediscovering some of the beneficial properties of the coarse-grained dataflow execution paradigm (much studied in the 80's), which enables conservation of off-chip bandwidth by implementing feed-forward dataflow pipelines between cores on the CMP. This approach of course exposes us to many more potential side-effects and dataflow hazards than the conventional approach. Such approaches have resulted in reconsideration of functional programming styles using more familiar modern languages. Examples can be found in CILK [25] and the DAG scheduling explored by Parry Husbands [30] and Jack Dongarra [25].

Rather than applying all of the cores on a CMP towards parallelism, they could be applied to other tasks such as background handling of System calls, Interrupts and other OS services as demonstrated by side-core methods [27]. It could be used for asynchronously monitoring execution and background handling of

load balancing and remeshing for adaptive applications. The cores could be used for background code analysis [28], code optimization, and even code-rewriting to facilitate aspect oriented programming [29]. Finally, the additional cores could be used in place of DMA engines for efficient background/one-sided communication. This was certainly the original vision for the Intel Paragon's communication co-processor mode and BG/L's Coprocessor mode of operations, were it not for the lack of L1 cache-coherency between the CPUs on the node. If we begin by imagining hundreds of processors on each node rather than 2 or 4, then it opens up our imagination to a much broader set of possibilities for apportioning computational capability on each socket of a large-scale system.

4. Conclusion

An enormous transition is underway that affects all sectors of the computing industry. We believe that our current conservative path toward multicore will ultimately stall, and in the long term, the industry will move rapidly towards a manycore design point containing hundreds or thousands of cores per chip. The underlying motivation for adopting manycore is driven by power limits for future system designs. The changes in architecture are also driven by the movement of market volume and capital that go with new CPU designs, from the desktop PC industry towards the consumer electronics applications.

All of these changes precede emergence of the parallel programming model. This is as much a concern for the consumer electronics industry as it is for us in the HPC world. We have more familiarity with parallel algorithms though, but our approach to parallel programming may prove unacceptable to that industry. Consequently, the desktop and consumer electronics industry are working hard towards an in-socket parallel programming model that can be adopted by non-expert programmers that populate their workforce. We may find ourselves forced to use the programming model they develop unless we engage actively in the development process. This transition is not just about HPC!

These transitions will lead to new era of architectural exploration given uncertainties about programming and execution model (and we MUST explore!). The HPC community needs to get involved now, given that it takes 3-5 years for new hardware designs to emerge, 3-5 years lead for new software ideas necessary to support new hardware to emerge, and 5+ MORE years to general adoption of new software. We are already too late to have an impact on the petaflop and are likely to be too late for the Exaflop if we don't raise the priority for research into programming models and system design immediately.

5. Acknowledgements

I would like to gratefully thank David Patterson, and all of the participants in the Berkeley View report for organizing 2 years of thought provoking discussions that fed into this work. This work was supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05\-CH\-11231.

6. References

[1] Krste Asanovic, 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. (http://view.eecs.berkeley.edu/)

[2] J. L. Hennessy, D.A. Patterson, "Computer Architecture: A Quantitative Approach; fourth edition," *Morgan Kaufmann*, San Francisco, 2006.

[3] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, Jul.–Aug. 1999, pp. 23–29.

[4] D. Burger, S.W. Keckler, K.S. McKinley, et al., "Scaling to the End of Silicon with EDGE Architectures," *IEEE Computer,* 37 (7), pp. 44-55, July, 2004.

[5] P.P. Gelsinger, "Microprocessors for the new millennium: Challenges, opportunities, and new frontiers," in *Proceedings of the International Solid State Circuits Conference (ISSCC)*, 2001, pp. 22–25.

[6] A. Wolfe, "Intel Clears Up Post-Tejas Confusion," *VARBusiness,* May 17, 2004. (http://www.varbusiness.com/sections/news/breakingnews.jhtml?articleId=18842588)

[7] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, 1992, pp. 473–484.

[8] D. Sylvester and K. Keutzer, "Microarchitectures for systems on a chip in small process geometries," *Proceedings of the IEEE*, Apr. 2001, pp. 467–489.

[9] W. Eatherton, "The Push of Network Processing to the Top of the Pyramid," *keynote address at Symposium on Architectures for Networking and Communications Systems*, Oct. 26–28, 2005. Slides available at http://www.cesr.ncsu.edu/ancs/slides/eathertonKeynote.pdf

[10] Intel Polaris Chip: http://www.intel.com/research/platform/terascale/teraflops.htm

[11] Top500 Project: http://www.top500.org/

[12] G. Bell, J. Gray, A. Szalay, "Petascale Computational Systems," *IEEE Computer*, Volume: 39, Issue: 1 Jan. 2006, pp 110-112.

[13] W. Wulf and S. McKee, "Hitting the memory wall: Implications of the obvious," *Computer Architecture News,* 23(1), 1995.

[14] J. Carter, H. He, J. Shalf, E. Strohmaier, H. Shan, and H. Wasserman, "The Performance Effect of Multi-Core on Scientific Applications," *Cray User Group (CUG2007)*, Seattle Washington, May 7–10, 2007.

[15] William T.C. Kramer, John Shalf, and Erich Strohmaier, "The NERSC Sustained System Performance (SSP) Metric," *LBNL Technical Report LBNL-58868,* September 2005.

[16] S. Williams, J. Shalf, L. Oliker, P. Husbands, S. Kamil, K. Yelick, "The Potential of the Cell Processor for Scientific Computing", *International Journal of Parallel Programming (IJPP),* DOI 10.1007/s10766-007-0034-5, April 2007.

[17] J. Shalf, S.A. Kamil, L. Oliker, and D. Skinner, "Analyzing Ultra-Scale Application Communication Requirements for a Reconfigurable Hybrid Interconnect," *in Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC '05),* Seattle, WA, Nov. 12–18, 2005. (LBNL-58052)

[18] S. Kamil, J. Shalf, L. Oliker, and D. Skinner, "Understanding Ultra-Scale Application Communication Requirements," *in Proceedings of the 2005 IEEE International Symposium on Workload Characterization (IISWC),* Austin, TX, Oct. 6–8, 2005, pp. 178–187. (LBNL-58059)

[19] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, J. Shalf, "Reconfigurable Hybrid Interconnection for Static and Dynamic Scientific Applications", *ACM International Conference on Computing Frontiers*, 2007.

[20] IPM Homepage: http://ipm-hpc.sourceforge.net/

[21] C. Bell, D. Bonachea, R. Nishtala, K. Yelick, "Optimizing Bandwidth Limited Problems Using One-Sided Communication and Overlap," *20th International Parallel & Distributed Processing Symposium (IPDPS),* 2006. (LBNL-59207)

[22] S. Borkar, "Designing Reliable Systems from Unrealiable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro,* Nov.–Dec. 2005, pp. 10–16.

[23] S.S. Mukherjee, J. Emer, and S.K. Reinhardt, "The Soft Error Problem: An Architectural Perspective," *in Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA-11 2005)*, Feb. 2005, pp. 243–247.

[24] C. Kozyrakis and K. Olukotun, "ATLAS: A Scalable Emulator for Transactional Parallel Systems," in Workshop on Architecture Research using FPGA Platforms, 11th International Symposium on High-Performance Computer Architecture (HPCA-11 2005), San Francisco, CA, Feb. 13, 2005.

[25] S.S. Mukherjee, J. Emer, and S.K. Reinhardt, "The Soft Error Problem: An Architectural Perspective," *in Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA-11 2005),* Feb. 2005, pp. 243–247.

[26] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julien Langou, Piotr Luszczek, and Stanimire Tomov. "The impact of multicore on math software," *In PARA 2006,* Umea Sweden, June 2006.

[26] Sanjay Kumar, Himanshu Raj, Karsten Schwan, Ivan Ganev, "Re-architecting VMMs for Multicore Systems: The Sidecore Approach," *to appear in WIOSCA 2007,* in conjunction with ISCA 2007.

[27] Robert W. Wisniewski, Peter F. Sweeney, Kartik Sudeep, Matthias Hauswirth, Evelyn Duesterwald, Calin Cascaval, and Reza Azimi, "Performance and Environment Monitoring for Whole-System Characterization and Optimization", *PAC2 (Conference on Power/Performance interaction with Architecture, Circuits, and Compilers),* 2004.

[28] Kiczales, Gregor; John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin (1997). "Aspect-Oriented Programming", Proceedings of the European Conference on Object-Oriented Programming, vol.1241, pp.220–242.

[30] P. Husbands, K. Yelick, "Multi-threading and One-Sided Communication for Matrix Factorization", *Proceedings of SC2007,* (*to appear)* 2007.