

# Hardware Acceleration Prospects and Challenges for High Performance Computing

Gregory B. Newby  
Arctic Region Supercomputing Center  
University of Alaska Fairbanks  
Fairbanks, Alaska, United States  
newby@arsc.edu

*High performance computing (HPC) has often benefited from special-purpose hardware. This paper examines the potential roles for several different approaches to hardware acceleration that are currently being deployed in HPC systems. Because each technology has different performance characteristics, as well as practical considerations (such as electrical consumption, physical interface, and cost), a match of these characteristics to the desired HPC workload is desirable. Technologies discussed include multicore processors, chip multithreading, graphics processing units, field-programmable gate arrays, Cell processors, and vector processors.*

## I. INTRODUCTION

The basic design of computing systems for high performance computing, which are also known as supercomputers, emphasizes scaling out to large numbers of processors. These processors are supported by associated hardware and communication infrastructure to enable rapid exchange of data among them. The application programs that run most efficiently on supercomputers are those that keep the processors busy, without spending too much time waiting for communication among processors or other events. In this paper, we examine emerging processor technologies that can accelerate some portions of applications, but are not targeted to replace general-purpose microprocessors at the heart of most contemporary systems.

Per Amdahl's law, as more time is spent computing (versus other activities), applications will be more able to achieve the theoretical peak performance of the system. However, the new HPC technologies described here can make it difficult to assess the role of Amdahl's law [1], or even to determine how to gauge fundamentals such as inter-processor communication time. In addition to dividing the computational work of an application across a number of available processors, these new technologies offer acceleration to just some of the computational workload – for example, multiplication of matrices, or evaluation of search strings, or digital signal processing.

In the following sections, we discuss several types of acceleration technologies that are of great interest for next-

generation HPC systems. While some, such as the Cell processor, have already been deployed in HPC systems, others have yet to be adequately demonstrated for large-scale systems. Technologies discussed include multicore processors, chip multithreading, the Cell Broadband Engine, graphics processing units, FPGAs, and vector processing units. Approaches to dealing with non-homogeneous systems are discussed. A concluding section recommends the approach of choosing HPC hardware, including options for acceleration technologies, based on the profile of the application, as well as practical matters such as cost, electrical consumption, and interface.

## II. MULTICORE PROCESSORS

It's only been a few years since microprocessor manufacturing processes enabled placement of more than one microprocessor on a single physical socket. Today, dual-core processors are inexpensive enough to be found in consumer computers, as well as HPC systems. Within a computer, whether it's a notebook or a node in a supercomputer, there is some number of physical CPU sockets. HPC systems built from large numbers of commodity parts, known as clusters, are often made of two or four-socket individual systems – known in this context as nodes. Given a two-socket node, for example, outfitting the node with single-core processors yields only two processors per node. Quad-core processors yield eight processors per node. In this way, we see that more cores provides for increased processor density within the same number of sockets.

The increased processor density comes at a cost, however: if a given node has some number of communication buses (i.e., PCI-E or other input/output buses), and a given bandwidth to main memory, and a given communication interconnect bandwidth, then that will be shared by more processors as the core count increases. Furthermore, today's quad-core processors operate at decreased clock frequency compared to dual- or single-core processors. On the socket, the processors might share cache memory.

Six, eight, and twelve-core processors will be available quite soon. Intel has showcased an 80-core chip providing a teraflop of computing power, but this is not expected to be a

---

This work was supported by the Arctic Region Supercomputing Center at the University of Alaska Fairbanks, with additional support from the US Department of Defense High Performance Computing Modernization Program Office under contract G00003435 and others.

mainstream product in the near future (see Intel Teraflops Research <http://www.intel.com/pressroom/kits/Teraflops/>).

For HPC workloads, multicore processors offer the challenge that the communication latency and bandwidth on the same socket will generally be less than between sockets on a node, which in turn will be less than communication among nodes. This lack of homogeneity in sharing of resources at the core, socket, and node level is generally unaccounted for by programming techniques and libraries – instead, such techniques, notably the message passing interface (MPI), assume that, all things being equal, processors are interchangeable.

While the multicore processors of today are fully featured general-purpose microprocessors, we see in on- versus off-socket latency an indication that the lack of homogeneity described for other technologies in following sections are evident even in widely available multicore processors.

### III. CHIP MULTITHREADING

The Sun UltraSPARC T1 and T2+ processor are good examples of chip multithreading (CMT) technology, in which the processor maintains separate threads, managed in hardware. To the operating system (and its human users), a single T2+ processor with 8 cores and 8 threads per core appears as 64 separate processors. These processors have a somewhat unbalanced allocation of other resources, though – with different levels of shared cache, and only one floating-point unit (FPU) per core, they are not suited for all application workloads.

All processors (and the operating systems that run on them) do multitasking and context switching, in order to run many programs with the illusion that each program is getting the full attention of the system. With multithreaded processors, the idea is that the separate program execution threads, which are to some extent under the control of the programmer, will be ready for execution only part of the time. So, if one thread is waiting for access to memory, and another is waiting for disk input, and a third is awaiting communication from another node, a fourth thread might be ready to be executed.

For enterprise computing and transaction processing, this division of labor among threads may work quite well. But for HPC applications that perform very well on modern processors, threads might have enough work ready for execution that chip multithreading does not yield much improvement.

For example, ARSC examined the weather research and forecasting (WRF) application, which tends to scale well to many thousands of processors. On the T2+ processor, we obtained linear increases in speed as we added cores, from one to eight. But beyond eight, where multithreading plays a role, there was little increase in performance.

### IV. CELL BROADBAND ENGINE

The Cell Broadband Engine processor (often, simply called the Cell) is at the heart of the Playstation3 gaming

console. Sony, IBM and Toshiba jointly developed the processor, which obtained recent fame as the main computational base of Roadrunner, the first petaflops-capable supercomputer. In Roadrunner, nodes are paired, with one node based on AMD Opteron processors, and the other based on Cell BE processors. This hybrid approach allows the Opterons to act as host and communications gateway for the Cells.

The architecture of the Cell BE is unique, and while it was designed for gaming it has good applicability to HPC workloads. The newly released PowerXCell 8i variant of the Cell processor, which is different than the processor found in Playstations, provides enhanced floating-point performance over its predecessor.

Within the Cell, general-purpose PowerPC cores host the OS, and interface with the rest of the system. Up to eight synergistic processing elements (SPEs) provide additional processing power, each with a theoretical peak performance of over 25 GFLOPS for single precision mathematics. These SPEs have their own shared and local memory, a communication bus, and an interface through direct memory access (DMA) to the PowerPC and the rest of the outside system.

The Cell presents some challenges for programmers [2]. Even without having the Opteron nodes of Roadrunner, a Cell-based system has the PowerPC, plus the SPEs. Deciding which components of a workload will run best on the SPEs, versus on the PowerPC or (in the case of Roadrunner) elsewhere, can be challenging at compile time or runtime.

In the Cell BE we see that the DMA model and PowerPC “front end” to the SPEs incurs a computational cost for accessing the SPEs. Issues such as memory alignment, word size, and choices of single versus double precision mathematics can make the choice to utilize the SPEs a difficult one. The software development kit (SDK) provided by IBM, as well as third party SDKs from RapidMind and others, can make use of the SPEs more transparent. Optimized libraries for linear algebra (via BLAS) and other common functions can make it easier for existing programs to benefit from the Cell.

With the Cell, for chip multithreading and multicore processors, we see that single-socket performance can, at least optimally, greatly exceed performance of a single microprocessor (with performance considered simply as a number of floating-point operations per second. There are many ways of looking at performance of an HPC system). We can also consider factors such as power consumption, physical form factor (how large the processor is, whether it fits in standard CPU sockets), and cost, in determining whether utilization will yield faster time to solution for applications of interest.

### V. GRAPHICS PROCESSING UNITS (GPUS)

While graphics processing units (GPUs) have utility in nearly every desktop and notebook computer, their usefulness for HPC has been limited until recently. This is changing as a result of GPU vendors providing general-purpose access to the hardware via hybrid programming languages, an SDK, and

associated application programmer interfaces (APIs). While these languages have been somewhat customized for particular GPU models and vendors, initiatives such as the Open Computing Language (OpenCL) are expected to provide a common programming platform for use of hybrid architectures.

GPUs are similar to multicore processors in that they make many computational cores available on a single system. They are similar to FPGAs (below) in that interaction with the GPU is via the system's communication bus, rather than being directly connected to the processor. The main difference is that the many cores (over 128 in current models) are somewhat limited in their computational abilities, and must be, to some extent, synchronized in their computation.

GPUs offer outstanding "flops per Watt," in that their large core count is balanced against the same power consumption as a general-purpose CPU (or a little more). While consumer graphics cards might not physically fit well in HPC nodes, this problem is likely temporary, as the major GPU manufacturers are looking at in-socket approaches, or other HPC-friendly form factors.

Like the Cell, programmers will benefit from application libraries for use of GPUs, so that they can avoid needing to program to the low-level differences between different devices. Current uses of the GPUs are limited to relatively small numbers of GPUs working together (perhaps two on a single node, or up to eight in a directly connected external system such as nVidia's Tesla).

Expanding the number of GPUs to accompany large numbers of nodes in a supercomputer is technically feasible, but due to form factor, power consumption, and lack of a single open programming model, has not been widely seen yet. Because of the immense potential speedups that GPUs offer for HPC workloads, favorable cost to special-purpose vector processors (below), and promise of consumer-driven enhancements over time, it is anticipated that GPUs will be more commonly seen in HPC systems in the near future.

## VI. VECTOR PROCESSORS

Historically, vector processors have played an important role in purpose-built supercomputers. Today, Cray's X2 module (which is an add-on blade for their XT5h product line) is the main exemplar in the US. NEC's SX series of supercomputers also provides vector processing capability.

The idea of vector processors, as found above for GPUs, is to process pipelines of mathematical computations. Increased speed over general purpose CPUs is achieved because the vector processors are optimized for common mathematical operations, such as matrix multiplication.

GPUs and Cell processors are both used for vector operations. Special-purpose vector processors tend to be costly, but their tight coupling with their host computers, along with efficient software libraries provided by the manufacturer, can make them appealing for HPC workloads.

## VII. FPGAS

Field programmable gate arrays (FPGAs) provide programmers the opportunity to run applications in hardware, rather than software. For well suited applications, performance close to that of a standalone application-specific integrated circuit (ASIC) may be achieved. These well suited applications are relatively small in number, however, due to the limited size of programs that fit on FPGAs, the bandwidth limitations in getting data to the FPGAs, and the limited ability of the FPGAs to fit multiple program elements simultaneously. Digital signal processing, some bioinformatics algorithms, and image processing are well suited to FPGAs, whereas many typical large-scale HPC applications (such as for climate modeling) are not as well suited.

Like special-purpose vector processors, FPGAs are sometimes found as part of purpose-built HPC systems. Experiments at ARSC and elsewhere found that the programming models for FPGAs, as well as the basic hardware limitations described above, are constraining factors for more widespread adoption [3].

## VIII. HETEROGENEOUS SYSTEMS: HARDWARE

Processor hierarchies, memory hierarchies, communication interconnect hierarchies, and storage hierarchies are all part of the emerging heterogeneous HPC landscape. Software approaches, discussed below, are needed to make this hardware more usable, and better able to approach peak efficiency across diverse HPC workloads.

The largest supercomputers in the world tend to be rather unique – Roadrunner is an excellent example. These unique designs yield results, later, which are incorporated into more mainstream systems. For the acceleration technologies discussed in this paper, we can anticipate that some technologies will join the mainstream, while others will either become of lesser interest, or be replaced by new advances.

Practical hands-on experience with such systems is a necessity to truly understand their utility. The Roadrunner system, for example, underwent numerous programmer years of application porting and testing, in order to insure the applications that the purchaser cared about most would run effectively. Even with this investment of effort, several applications of interest were abandoned due to a lack of perceived fit with the underlying hardware architecture.

For HPC, we need to consider the relative value of speedup from acceleration technology, as compared to the effort required to obtain that speedup. Perhaps more importantly, the continued pace of technology development, accompanied by decreasing costs on a per-FLOP level, means that a 10X speedup in application performance today, at great expense, might be less effective than simply either buying a system that is 10X larger, or waiting for mainstream computing components to get 10X faster.

## IX. HETEROGENEOUS SYSTEMS: SOFTWARE

Many current applications make use of the message passing interface (MPI) to divide work among the different

available CPUs. Because MPI assumes approximately homogenous nodes, it attempts to amortize differences among node performance by assigning tasks equitably. This approach might not achieve optimal performance in systems with a variety of different node types. For the current discussion, an important limitation is that acceleration hardware might not be ubiquitous in a system. Instead it might be tied only to particular nodes.

Parallel global address space (PGAS) languages, such as Unified Parallel C [4] and CoArray Fortran, offer some promise for hiding the complexity of underlying hardware from the programmer. One challenge is that porting an MPI application to UPC is likely to be non-trivial for large-scale applications. Another is that the PGAS languages' attempt to hide complexity might need to be tempered somewhat, in order to give the programmer opportunity, whether at compile time or run time, to deploy the application so that it makes optimal use of the available acceleration hardware. Moreover, PGAS languages are likely to rely on software libraries such as BLAS to use acceleration hardware, and those libraries are not yet sufficiently sophisticated to make effective choices about what work will be faster on acceleration hardware, and what work will be faster if kept on the CPU.

Such libraries are suitable for existing applications as well. The largest and most complex HPC applications are often written in Fortran, C, or a combination. In addition to relying on MPI (or OpenMP, or other approaches to parallelization), the reliance on BLAS and other libraries offer opportunities to automatically, or semi-automatically, add support for new acceleration hardware such as GPUs, Cell processors, and so forth. Challenges occur, though, because for some workloads the acceleration hardware might actually yield slower performance. Additionally, many deployed systems do not pair acceleration hardware with every general-purpose CPU – thus necessitating a hybrid approach. Even with software libraries, this is likely to require significant programming effort.

## X. APPLICATION CHOICES

Not all applications are highly parallel, but those that tend to push leading-edge design in supercomputers are. These applications tend to push performance limits of CPUs, as well as interconnects among different components of the underlying systems. These include the memory subsystem and caches, buses among the CPUs and to peripheral devices, communication fabrics between nodes, and pathways to external storage and networks.

The prevalence of multicore processors is pushing application developers, whether they are individuals, communities of developers, or software vendors, to take heterogeneous systems into account – for example, by leveraging enhanced on-socket performance versus off-node. These heterogeneous approaches will be necessary for the other acceleration hardware discussed above.

## XI. CONCLUSION

Choose your hardware to match your budget and other constraints. The choice of acceleration hardware for the next

few years will be made difficult by the lack of maturity in the underlying software, as well as the lack of integrated vendor support. But tremendous opportunities await those who have a well-suited application, or the time and expertise to devote to get enhanced performance from acceleration hardware.

FPGAs, chip multithreading, Cell processors, GPUs, and vector processors are all available today, but are relatively immature technologies for use in HPC systems. Multicore processors will help to focus development efforts on heterogeneous systems, but there are few instant solutions to moving a particular application of interest to these new technologies.

## ACKNOWLEDGMENT

Thanks to the many colleagues, including coworkers, interns, postdoctoral fellows, students and faculty members, who contributed ideas, benchmarks, results and discussion reflected in this paper. Special thanks to Tarek El-Ghazawi, Mohamed Abouellail, Don Bahls, Chris Fallen, Abdullah Kayi, Ed Kornkven, Lenny Wang, and Ying Yu.

## REFERENCES

- [1] M.D. Hill, M.R. Marty, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33-38, Jul., 2008
- [2] J. A. Turner, "Roadrunner Applications Team: Cell and Hybrid Results to Date." Los Alamos National Laboratory. Online document accessed September 29, 2008 at <http://www.lanl.gov/roadrunner/rperfassess.shtml>
- [3] G. B. Newby, "Promise and prospects for multicore and emerging processor technologies." Arctic Region Supercomputing Center. Online document accessed September 29, 2008 at <http://www.arsc.edu/science/multicore/multicore-whitepaper.pdf>
- [4] T. El-Ghazawi, W. Carlson, T. Sterling, *UPC: Distributed Shared Memory Programming*. Hoboken: John Wiley. 2005.