Revolutionary Technologies for Acceleration of Emerging Petascale Applications
Guest Editors' Introduction

Rupak Biswas, Leonid Oliker, Jeffrey Vetter

As we enter the era of billion transistor chips, computer architects face significant challenges in effectively harnessing the large amount of computational potential available in modern CMOS technology. Chip designers have been moving away from maximizing single-thread performance via exponential scaling of clock frequencies toward chip multiprocessors (CMPs) in order to better manage trade-offs among performance, energy efficiency, and reliability. Because this design approach is relatively immature, the community is exploring a vast diversity of CMP architectures. System designers and application programmers are confronted with a myriad of architectural features, including multicore, simultaneous multithreading, core heterogeneity, and unconventional memory hierarchies, often combined in novel arrangements. Given the current flux in CMP design, it is unclear which architectural philosophy is best suited for a given class of algorithms. Likewise, this architectural diversity leads to uncertainty on how to re-configure existing algorithms and tune them to take the maximum advantage of existing and emerging platforms. Understanding the most efficient design and utilization of these increasingly parallel multicore systems is one of the most challenging questions faced by the computing industry in several decades.

Today's CMP designs can be separated into several architectural categories. The most straightforward approach for CMP design integrates numerous copies of existing superscalar processors on to a single chip, and is exemplified by platforms such as the Intel Xeon and AMD Opteron. Utilizing monolithic processor technology as building blocks maximizes serial thread performance and leverages the architectural optimization techniques of the past twenty years. However, the vendors of these commodity processors conceived of their design in an era in which Moore's Law scaling allowed continuous improvement in logic density, which was typically accompanied by increases in clock frequencies. Chip architects leveraged this continuous improvement to increase the complexity of their designs in order to improve sequential processing performance. Due to practical constraints on power and thermal design, however, the sequential performance of the last several generations of high-end processors has improved very little due to the plateau of clock frequencies. Moreover, commodity processors remain grossly inefficient for many scientific codes, achieving only a small fraction of the system's advertised peak performance.

The move to CMP designs and away from increasing sequential performance has also triggered investigations of programming environments. For nearly two decades, users could write applications that benefited directly from increasing clock speeds, oftentimes without modifications to their source code. The CMP architectural revolution is forcing users to refactor and optimize their applications to exploit these new architectures by exposing parallelism at multiple levels (e.g., instructions, threads, thread groups). For example, applications targeting graphics or Cell processors must rewrite their main kernels in NVIDIA's Compute Unified Device Architecture (CUDA) or Cell's programming environment, respectively. Making this task more difficult is the lack of standards across CMP designs, forcing users to invest considerable time and skill in rewriting their applications. Recent standardization efforts for Open Compute Language (OpenCL) appear as a promising step in order to reduce the labor of using and evaluating these new devices.

In this special issue, we examine two unconventional designs of CMP architectures with the potential to revolutionize high-end computing: the Sony Toshiba IBM (STI) Cell and the recent NVIDIA GPGPUs (general-purpose graphics processing units). These platforms have the promise of effectively leveraging the continued advances in silicon technology to achieve significant improvements in performance- and power-efficiency compared with conventional superscalar-based CMPs. Clearly, these benefits are appealing to the computational science community whether the CMPs are used as standalone computational devices or accelerator offload engines.

The heterogeneous STI Cell processor is the heart of the Sony PlayStation 3 (PS3) video game console, whose aggressive design is intended to meet the demanding computational requirements of video games. Cell adopts a heterogeneous approach to multicore, with one conventional processor core to handle OS and control functions, combined with up to eight simpler cores for the computationally intensive work. These simple cores differ considerably from conventional core architectures due to their use of a disjoint software-controlled local memory instead of the conventional hardware-managed cache hierarchy. This approach allows more efficient use of available memory bandwidth than is possible with standard prefetch schemes on conventional cache hierarchies, but also makes the programming model more complex. Combined, the heterogeneous cores of the Cell processor allow the potential of high performance for appropriately multithreaded applications as well as the ability to efficiently execute unmodified single-threaded codes.

The NVIDIA GPGPU is a homogeneous multicore architecture comprised of many simple in-order cores and is designed primarily for high-performance 3D graphics rendering. Currently, it is available only as discrete graphics units on PCI-Express cards. Its recent inclusion of double precision datapaths makes it a particularly interesting target for high performance scientific applications. This architectural approach clearly focuses on multithreaded throughput instead of singe thread performance, and thus requires application design to focus on maximizing thread-level parallelism. Like the Cell processor, the NVIDIA GPGPU utilizes local-store technology, requiring application reengineering; however, the newly developed C-like CUDA programming language interface promises a significantly simpler and much more general-purpose programming paradigm than on previous GPGPU platforms.

The first paper in this special issue by Bader, Agarwal, and Kang investigates the performance of two different discrete transform kernels on the Cell architecture. Their innovative Fast Fourier Transform algorithm for 16K complex input samples achieves a single-precision performance of 18.6 Gflop/s, outperforming the Intel Xeon (Woodcrest) chip. The Discrete Wavelet Transform implementation utilizes a novel data decomposition strategy to improve direct memory access and enhance scalability for both lossless and lossy transforms, running significantly faster than on the AMD Opteron (Barcelona). The authors highlight the superiority of Cell over general-purpose multicore processors for bandwidth-intensive applications.

The paper by Kurzak, Alvaro, and Dongarra describes the canonical implementation of single-precision matrix multiplication micro-kernels on the short-vector synergistic processing element (SPE) of the Cell processor. A sustained performance of 99.8% of peak is reported for matrices of size 64 x 64 elements while using less than 6 KB of memory for the code and supporting data structures. However, this was achieved by manually optimizing the code. The challenge is to achieve similar results via auto-vectorization combined with heuristic techniques for more substantial application codes.

Meredith, Alvarez, Maier, Schulthess, and Vetter conduct a case study of a quantum Monte Carlo application of critical importance to the Department of Energy on NVIDIA G80, and report on its accuracy and performance for the entire application across a cluster of GPUs. These issues are important because GPUs were originally developed for real-time rendering where accuracy is a secondary concern relative to throughput. Results indicate good performance and excellent accuracy, including for double-precision calculations on a pre-production sample of the GT200. However, a more detailed investigation with a variety of codes is necessary to establish the viability of GPUs for large-scale scientific applications.

The paper by Hardy, Stone, and Schulten report on the use of a GT200 GPU and the CUDA programming toolkit to accelerate the multilevel summation process for computing electrostatic potentials for a system of interacting atoms. These calculations constitute a key component in biomolecular modeling applications. Results indicate a 26x speedup on a single GPU when computing the electrostatic potential for a system containing more than 1.5 million atoms. The ability to create potential maps of such systems in a few seconds enables interactive analysis previously unavailable. The challenge however is to run efficient calculations on systems larger than 100 million atoms on multi-GPU platforms.

The final paper in this special issue is authored by Williams, Oliker, Vuduc, Shalf, Yelick, and Demmel. They examine the performance of a sparse matrix-vector multiplication kernel on several multicore architectures including those from AMD, Intel, and Sun, and compare it against that on the Cell. Effective optimization strategies designed explicitly for multicores demonstrate significant performance improvements on these platforms, but architectural differences determine the optimal implementation. The authors conclude by providing valuable insights into the hardware design trade-offs for memory-bound scientific applications.