Available online at www.sciencedirect.com

ScienceDirect

Parallel Computing 33 (2007) 645–647

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Guest editorial

High-performance computing using accelerators

A recent trend in high-performance computing is the development and use of *heterogeneous* architectures that combine fine-grain and coarse-grain parallelism using tens or hundreds of disparate processing cores. These processing cores are available as *accelerators* or *many-core processors*, which are designed with the goal of achieving higher parallel-code performance. This is in contrast with traditional multicore CPUs that effectively replicate serial CPU cores. The recent demand for these accelerators comes primarily from consumer applications, including computer gaming and multimedia. Examples of such accelerators include graphics processing units (GPUs), Cell Broadband Engines (Cell BEs), field-programmable gate arrays (FPGAs), and other data-parallel or streaming processors. Compared to conventional CPUs, the accelerators can offer an order-of-magnitude improvement in performance per dollar as well as per watt. Moreover, some recent industry announcements are pointing towards the design of heterogeneous processors and computing environments, which are scalable from a system with a single homogeneous processor to a high-end computing platform with tens, or even hundreds, of thousands of heterogeneous processors.

This special issue on "High-Performance Computing Using Accelerators" includes many papers on such commodity, many-core processors, including GPUs, Cell BEs, and FPGAs.

GPGPUs: Current top-of-the-line GPUs have tens or hundreds of fragment processors and high memory bandwidth, i.e. 10× more than current CPUs. This processing power of GPUs has been successfully exploited for scientific, database, geometric and imaging applications (i.e. GPGPUs, short for General-Purpose computation on GPUs). The significant increase in parallelism within a processor can also lead to other benefits including higher power-efficiency and better memory-latency tolerance. In many cases, an order-of-magnitude performance was shown, as compared to top-of-the-line CPUs. For example, GPUTeraSort used the GPU interface to drive memory more efficiently and resulted in a threefold improvement in records/second/CPU. Similarly, some of the fastest algorithms for many numerical computations – including FFT, dense matrix multiplications and linear solvers, and collision and proximity computations – use GPUs to achieve tremendous speed-ups.

Cell Broadband Engines: The Cell Broadband Engine is a joint venture between Sony, Toshiba, and IBM. It appears in consumer products such as Sony's PlayStation 3 computer entertainment system and Toshiba's Cell Reference Set, a development tool for Cell Broadband Engine applications. When viewed as a processor, the Cell can exploit the orthogonal dimensions of task and data parallelism on a single chip. The Cell processor consists of a symmetric multi-threaded (SMT) Power Processing Element (PPE) and eight Synergistic Processing Elements (SPEs) with pipelined SIMD capabilities. The processor achieves a theoretical peak performance of over 200 Gflops for single-precision floating-point calculations and has a peak memory bandwidth of over 25 GB/s. Actual speed-up factors achieved when automatically parallelizing sequential code kernels via the Cell's pipelined SIMD capabilities reach as high as 26-fold.

Field-Programmable Gate Arrays (FPGAS): FPGAs support the notion of reconfigurable computing and offer a high degree of on-chip parallelism that can be mapped directly from the dataflow characteristics of an application's parallel algorithm. Their recent emergence in the high-performance computing arena can be attributed to a hybrid approach that combines the logic blocks and interconnects of traditional FPGAs with

0167-8191/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.parco.2007.10.001

embedded microprocessors to create a "complete system on a programmable chip." Examples of such an approach include the Virtex-II Pro, Virtex-4, and Xilinx devices, which include one or more PowerPC processors embedded within the FPGA's logic. The most recent success of FPGAs in high-performance computing came when FPGAs in the Tsubame cluster in Tokyo were used to improve performance by an additional 25%.

High-Performance Computing: The high-performance research community is increasingly leveraging the capabilities of these accelerators for various applications. Many supercomputer designs targeting PetaFlop performance and beyond now revolve around the use of accelerators. For instance, the Roadrunner machine designed at Los Alamos National Laboratory could use up to 16,000 Cell processors with a performance target in excess of 1 PF. The Tsubame cluster in Tokyo recently used 360 ClearSpeed accelerators to reach 47 TF on the Linpack benchmark, placing it 5th in performance worldwide.

Challenges and Paradigm Shift: The design and availability of these accelerators or many-core processors has given rise to many fundamental challenges. For example, these processors provide us concurrency, i.e. the ability to execute the sub-tasks of a given task in parallel. However, in order to take advantage of the available hardware resources, we need to develop new parallel programming paradigms in software technologies and applications. The new paradigm shift brings us many new challenges in terms of software engineering design as well as design of new algorithms and software tools that can handle heterogeneity. Finally, we need to design better programming models to take advantage of multiple cores within these accelerators, along with good debugging and profiling support.

Contents of this issue

Given this recent interest in accelerators and the new challenges, this special issue addresses some of the issues with architectures, programming models, tools and libraries, applications, and performance evaluation. Specifically, this special issue consists of six papers. We have organized them based on the specific accelerators that are used to accelerate the related hardware architectures and supporting graphics layers and application programming interfaces (APIs). The first three papers make use of computational capabilities of GPUs, in the form of new emerging area of GPGPUs. The first paper on "Scout: A data parallel programming language for graphics processors" by McCormick et al., describes a novel data programming language that provides a high-level API for various applications and hides the low-level details models for GPUs and also presents many extensions for visualization-centric operations. The second paper on "Cache-efficient numerical algorithms for graphics hardware" by Govindaraju and Manocha addresses the issue of designing cache-friendly GPGPU algorithms. This paper highlights the differences between the memory architectures of commodity CPUs and GPUs and uses the characteristics of GPU cache hierarchies to considerably improve the performance of many numerical and scientific algorithms. The third paper on "Exploring weak scalability for FEM calculations on a GPU-enhanced cluster" by Göeddeke et al. addresses many issues in developing cluster systems that utilize GPUs as co-processors. These include overall computational performance along with system integration and power consumption. They highlight the performance on a Poisson solver based on a 160 compute-node cluster.

The next two papers in this special issue deal with the Cell Broadband Engine (BE) architecture. The first Cell BE paper – "Runtime scheduling of dynamic parallelism on accelerator-based multi-core systems" – explores run-time mechanisms and policies for *scheduling* dynamic multi-grain parallelism on the Cell. Specifically, it investigates user-level schedulers that appropriately size the dimensions and degrees of parallelism on the Cell without requiring programmer intervention or sophisticated compiler support. The second Cell paper, entitled "High performance combinatorial algorithm design on the Cell Broadband Engine processor," studies the *mapping* of combinatorial algorithms to the Cell BE. In particular, the paper looks at the performance of the Cell on problems with non-uniform memory access patterns. Finally, the last paper of this issue – "Single pass streaming BLAST on FPGAs" – presents a new algorithm to emulate the seeding and extension phases of the BLAST algorithm in a single pass on an FPGA.

Acknowledgements

We would like to acknowledge several folks, who have helped us with the special issue. First of all, we thank Dan Reed for proposing the idea of coming up with a special use on use of GPUs and other accelerators for high-performance computing. Jeff Hollingsworth provided considerable support and guidance as a member of the editorial board of Parallel Computing. This issue would not have been possible without endless efforts and support of Jan Kastelein, the Journal Manager of Parallel Computing. Finally, we would like to thank all the authors who submitted papers in response to our call.

Wu-Feng
Departments of Computer Science and Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24060, USA
E-mail address: feng@cs.vt.edu

Dinesh Manocha
Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, USA
E-mail address: dm@cs.unc.edu

Available online 11 October 2007