Introduction

Barbara Chapman*, Bart Kienhuis*, Eduard Ayguadé*, François Bodin*, Oscar Plata*, and Eric Stotzer*

During the past few years, mainstream computing has rapidly embraced multi-core architectures and there is now considerable research into the design of such systems and into all aspects of their utilization. In particular, much work is needed in order to improve their programmability. Limitations on the amount of memory, bandwidth constraints, multiple layers of parallelism and heterogeneous components all introduce new challenges for application development and execution. In recognition of the importance of this architectural shift, multi-core programming was added to the collection of Euro-Par topics this year. The focus of the topic covered general-purpose multi-core programming techniques, models and languages, as well as those for multi-core embedded systems, and included related work on compilers, run-time systems, and performance and scalability studies.

The topic attracted 31 submissions covering a broad variety of research into languages, compilers, runtime libraries, algorithms and application experiences, many of which targeted heterogeneous multi-core platforms. Out of these, 12 were accepted for presentation at the conference.

In their paper "Tile Percolation: an OpenMP Tile Aware Parallelization Technique for the Cyclops-64 Multicore Processor", researchers at the University of Delaware propose a set of OpenMP pragmas that enable the application developer to control data movement on platforms with a software-managed memory hierarchy, such as the IBM Cyclops-64. They propose tile percolation, a technique using these pragmas, and demonstrate that it improves the efficiency of OpenMP codes on the Cyclops.

Ayguade, Badia, Igual, Labarta, Mayo, and Quintana-Orti present an extension of the Star Superscalar programming model for expressing the parallelism in applications targeting a general purpose CPU connected to multiple GPGPUs in their paper "An Extension of the StarSs Programming Model for Platforms with Multiple GPUs". Their approach requires few modifications to application code in order to adapt it to heterogeneous systems with multiple memory spaces.

"StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures", from Augonnet, Thibault, Namyst, and Wacrenier at the University of Bordeaux describes a novel runtime system that provides a unified execution model and data management library for heterogeneous systems. StarPU can support the generation of parallel tasks, as well as runtime techniques for scheduling them on heterogeneous systems in a manner that directly exploits the heterogeneity.

* Topic chairs.

H. Sips, D. Epema, and H.-X. Lin (Eds.): Euro-Par 2009, LNCS 5704, pp. 837–838, 2009.

XJava is a Java extension for object-oriented parallel programming. It is described, along with its implementation, in "XJava: Exploiting Parallelism with Object-Oriented Stream Programming" by Otto, Pankratius, and Tichy. It relies on tasks as a central construct for achieving parallelism and composability while preventing synchronization bugs.

Yan, Grossman, and Sarkar propose a means for Java programmers to invoke CUDA kernels in "JCUDA: A programmer-friendly interface for accelerating Java Programs with CUDA". Their approach relies on the compiler to generate the code needed to invoke the kernels and to transfer data between host and accelerator.

"Fast and Efficient Synchronization and Communication Collective Primitives for Dual Cell-based Blades" by Gaona, Fernandez, and Acacio describes the design of collective operations that take into account the structure of heterogeneous computing systems requiring explicit data transfer between the hardware components. They show results based upon their implementation on dual Cell-based blades.

A group of researchers at Simon Fraser University and Electronic Arts Blackbox describe requirements and techniques for parallelizing video game engines in "Searching for Concurrent Design Patterns in Video Games". They also discuss a parallel programming environment that they have created specifically targeting video game engines.

The paper entitied "Parallelization of a Video Segmentation Algorithm on CUDA-enabled Graphics Processing Units:" presents experiences gained by Gomez-Luna, Gonzalez-Linares, Benavides and Guil using CUDA to parallelize highly compute-intensive video frame feature calculations. They compare results with an OpenMP code version running on a multi-core platform.

"A Parallel Point Matching Algorithm for Landmark Based Image Registration Using Multicore Platform" by Yang, Gong, Zhang, Nosher, and Foran reports on a new algorithm for fast point matching in the context of landmark-based medical image registration, which has near real-time requirements. They have shown significant performance improvements on a Cell platform.

The number of cores configured on future architectures is likely to be much higher than those on current platforms. In "High Performance Matrix Multiplication on Many Cores", Yuan, Zhou, Tan, Zhang, and Fan discuss the Godson-T many-core processor and explain how they have adapted a dense matrix multiplication algorithm to exploit this architecture, achieving 97.1% of peak performance.

The paper "Parallel Lattice Basis Reduction using a Multithreaded Schnorr-Euchner LLL Algorithm" introduces a new, multithreaded variant of this algorithm which is of great importance in cryptography. The authors, Backes and Wetzel, discuss performance obtained by a Pthreads-based implementation on two dual-core Opteron processors for both sparse and dense lattice bases.

Researchers at the University of Strasbourg have developed a parallel evolutionary algorithm that exploits a GPGPU for the computationally intensive portions. The results they present in "Efficient Parallel Implementation of Evolutionary Algorithms on GPGPU Cards" are based on extensions to the EASEA specification language and compiler, and indicate up to a 100-fold performance increase over execution on a standard CPU.