Scientific Programming 17 (2009) 283–284
DOI 10.3233/SPR-2009-0288
IOS Press

Introduction to the *Scientific Programming* Special Issue: Software Development for Multi-core Computing Systems

The improvement of single-processor performance by increasing clock rate and instructions-per-clock number has reached its technological limits. Increasing instead the number of processing cores per chip is an alternative that can yield better overall system performance and alleviate problems related to energy consumption, heat dissipation and design complexity. As a result multi-core processors are now rapidly emerging in all markets ranging from laptops and game consoles to servers and supercomputers. However, from the software development perspective exploiting the full potential of multi-core computing systems is a highly complex and essentially open domain, at the moment forcing application developers to deal with numerous low-level details. Major research efforts are required to streamline the process of software development for multi-core computing systems. The intention of this special issue was to give researchers an opportunity to offer an outlook on possible approaches and solutions to some of these problems.

The special issue includes four papers, that together address a representative set of issues related to software development for multi-core computing systems. Each submitted paper was thoroughly evaluated by at least three reviewers considering the significance to the call for papers, novelty, technical quality, and clarity of presentation. The final selection was based on the recommendations of the reviewers.

The first paper – "Heterogeneous multicore parallel programming for graphics processing units" by Francois Bodin and Stephane Bihan – is an invited paper that addresses the difficult issue of programming accelerated computing systems, with some emphasis on GPU-accelerated systems. The authors present the Heterogeneous Multicore Parallel Programming (HMPP) environment, which is a step towards enabling portable programming of GPU-accelerated systems. The HMPP programming environment includes C and FORTRAN compilers and a corresponding runtime system. The usefulness of HMPP is demonstrated with a set of experiments on a system that comprises an Intel Core 2 Duo and an NVIDIA Tesla C1060. Codes for experimentations include matrix multiplication, convolution, Black–Scholes and Sobel.

In the second paper – "Evaluating multicore algorithms on the unified memory model" by John E. Savage and Mohammad Zubair – the Unified Multicore Model (UMM) is introduced, which is an extension of the memory hierarchy game (introduced by one of the authors in the 1990s) to multi-core processors. Limitations of existing related models typically include the inability to model varying degrees cache sharing at different levels. Lower bounds on communication traffic between levels of cache are established. This permits authors to design a cache-efficient algorithm for an option pricing problem. They also demonstrate their methodology for the matrix multiplication, binomial pricing and FFT.

The third paper – "Exploiting fine-grain thread parallelism on multicore architectures" by Panagiotis Hadjidoukas, Giorgos Philos and Vassilios Dimakopoulos – describes a thread programming package called PSTHREADS with a PTHREADS-like programming interface. PSTHREADS can be used stand-alone for manual thread programming or as (low-level) component in the implementation of a runtime system of an OpenMP compiler. The authors present such a use in their own OpenMP compiler, called OMPi. The improvements in PSTHREADS address support for fine-grained threads that can occur when exploiting nested parallelism in OpenMP programs. The authors evaluate their approach using an extension of the EPCC microbenchmarks and a face-detection application.

In the fourth paper – "ePRO-MP: A tool for profiling and optimizing energy and performance of mobile multiprocessor applications" by Wonil Choi, Hyunhee Kim, Wook Song, Jiseok Song and Jihong Kim – a tool that can assist software developers in analyzing performance and energy consumption is presented. The tool uses hardware counters to extract data, and is only in the very first calibration phase dependent on external power measurement equipment. Performance is measured directly, while energy is estimated using the hardware counters information and a regression model. The authors include profiling and evaluation results for an ARM11 MPCore system.

We are grateful to all authors who submitted papers to this special issue and to reviewers for their efforts in evaluating the submissions, and for doing so in a timely way. Furthermore, we would like to acknowledge the encouragement and support of Ewa Deelman (Associate Editor of the *Scientific Programming*) and Boleslaw Szymanski (Editor-in-Chief of the *Scientific Programming*).

Sabri Pllana and Jesper Larsson Träff
Special Issue Editors