Topic 4: High-Performance Architecture and Compilers

Alex Veidenbaum, Nectarios Koziris, Toshinori Sato, and Avi Mendelson

Topic Committee

High-performance architecture and compilation are the foundation on which the modern computer systems are built. The two sub-topics are very strongly related and only in combination can deliver performance levels we came to expect from systems. The topic is quite broad, with sub-areas of interest ranging from multi-core and multi-threaded processors to large-scale parallel machines, and from program analysis, program transformation, automatic discovery and management of parallelism, programmer productivity tools, concurrent and sequential languages, and other compiler issues.

This year four papers were accepted after a thorough review and discussion. These papers are summarized below. We are grateful to all reviewers who helped us in this process, as we obtained at least three reviews per submitted paper.

It is clear that the remaining papers proposed interesting ideas, but this year's competition was really tough. We thank all for their submissions and hope everyone will continue to support the conference. We also thank the Euro-Par Organizing Committee for their guidance and their useful comments.

The paper "Dynamic Last-Level Cache Allocation to Reduce Area and Power Overhead in Directory Coherence Protocols" by Mario Lodde, Jose Flich, and Manuel E. Acacio proposes the reorganization of the Last Level Cache (LLC), where the storage or not of the cache blocks' data will depend on their characterization as private or shared blocks. More specifically, if a block is private (i.e. used only by one core), then the LLC will hold only its tag and any information needed by the coherence protocol. The motivation behind this proposal is the observation that a large percentage of the actions performed by the LLC concerns private blocks as they are forwarded straight to the L1 caches and do not involve the data portion of the LLC. By "eliminating" the storage of the private blocks in the LLC, the authors achieve area and power savings with a negligible impact on the performance.

The paper "A Practical Approach to DOACROSS Parallelization" by Priya Unnikrishnan, Jun Shirako, Kit Barton, Sanjay Chatterjee, Raul Silvera, and Vivek Sarkar presents a new approach for automatic parallelization of DOACROSS loops. It is based on a compiler and runtime optimization ("dependence folding") which bounds the number of synchronization variables needed to control cross-iteration dependences. Furthermore, the authors present a cost analysis for determining the profitability of parallelization, and additional techniques (unrolling, chunking) that increase granularity and reduce synchronization overhead. These characteristics render their approach practical, compared to prior similar efforts. Their approach was evaluated using 4 benchmarks on a 32-core machine. The auto-parallelization of DOACROSS loops offered

C. Kaklamanis et al. (Eds.): Euro-Par 2012, LNCS 7484, pp. 204–205, 2012.

significant speedups (compared both to sequential execution and DOALL automatic parallelization) but only when cost analysis and granularity control was enabled.

The paper "Exploiting Semantics of Virtual Memory to Improve the Efficiency of the On-Chip Memory System" by Bin Li, Zhen Fang, Li Zhao, Xiaowei Jiang, Lin Li, Andrew Herdrich, Ravishankar Iyer, and Srihari Makineni proposes two hardware-based mechanisms that exploit stack memory's characteristics to optimize on-chip memory. The first mechanism reduces TLB misses by $10\% - 20\%$ by automatically creating large pages ("superpages") to host stack memory contents. The second technique treats stack accesses in a distributed shared cache in a different way than regular ones, by routing them to each core's local cache slice. The benefit of this approach is reduced interconnect power consumption by more than 14%. Both techniques are evaluated using a simulation framework and the SPEC CPU 2000 benchmarks.

Finally, the paper "From Serial Loops to Parallel Execution on Distributed Systems" by George Bosilca, Aurelien Bouteiller, Anthony Danalis, Thomas Herault, and Jack Dongarra, presents a compiler front-end for the DAGuE runtime system, to analyze annotated serial loops of tiled dense linear algebra algorithms, in order to provide symbolic information to the runtime system for the efficient execution on distributed memory machines.