# Topic 04
# Compilers for High Performance

Jens Knoop, Manish Gupta, Keshav K. Pingali, and Michael F. P. O'Boyle

Topic Chairpersons

It is a pleasure for us to welcome you to this year's Euro-Par conference in Manchester, and its topic on "Compilers for High Performance." As in previous years, this meeting provides an excellent opportunity to socialize with fellow researchers from all over the world, and to get hands on up-to-date research and the many activities going on in the field of parallel processing.

Within Euro-Par, the High Performance Compilers topic is devoted to research in all subjects concerning technology for compilation of programs for high performance systems, including, but not limited to, optimizing the utilization of system resources such as power consumption, code size, and memory requirements. Contributions were sought in all related areas, including the traditional fields of compiler technology such as static program analysis and transformation (including automatic parallelization), mapping programs to processors (including scheduling, and allocation and mapping of tasks), and code generation, but also dynamic and feedback directed optimization, compiling for embedded, hybrid and heterogeneous systems, and the interplay between compiler technology and development and execution environments. The focus on the application of these techniques to the automatic extraction and exploitation of parallelism distinguishes the topic from the other compiler oriented topics in Euro-Par (#03, #07, #13, and #20).

This year fifteen papers, contributed by an international mix of authors from three continents and eight countries, were submitted, one of which was moved from Topic 03 on "Scheduling and Load Balancing" to Topic 04. Each of these papers was reviewed by at least three referees, with an average of 3.66 reports per paper. Using the referees' reports as guidelines, the topic committee selected after intensive e-mail discussions eight of the submitted papers for publication and presentation at the conference. Of these, five papers were elected as regular papers, and three as research notes. These papers were presented in two consecutive sessions on Friday morning.

The topics covered by this year's submissions reflect the wide spectrum of current research in the field of high performance compilers. This heterogeneity is still visible in the papers accepted for presentation at the conference. However, there are some superordinate themes predominating in this year's submissions. Above all, these are novel uses of elsewhere developed and approved means and techniques for new applications and new application scenarios, and the integration of dynamic and static approaches to benefit from the best of both worlds. Additionally, much emphasis is given to the comparing exploration of the efficacy of competing approaches, and the extraction of their specific application profiles. Considering the accepted papers in more detail makes this evident.

R. Sakellariou et al. (Eds.): Euro-Par 2001, LNCS 2150, pp. 204–206, 2001.

The first session of the High Performance Compilers topic is devoted to program analysis and transformation. It begins with a paper by Sebastian Unger and Frank Mueller, in which they reconsider the handling of irreducible code in optimization, a classical problem getting new relevance with the growing dissemination of modern VLIW-like architectures supporting instruction-level parallelism such as Intel's IA-64. The authors present a novel approach for transforming irreducible loops into reducible ones, which is guided by a new heuristic for node splitting based on computing the dominator tree of a program. They explore the efficacy and adequacy of the new approach for subsequent optimizations in comparison to both the traditional node-splitting approach, and an approach using DJ-graphs for representing and optimizing irreducible code. Their findings will be useful for anyone (re-) designing a compiler. The next two papers are concerned with redundancy elimination. The first of these, by Manel Fernández, Roger Espasa, and Saumya Debray, targets the elimination of (partially) redundant loads in executable code by replacing expensive loads from memory by inexpensive register transfers. Conceptually similar to classical *partial redundancy elimination*, the specific constraints of the setting such as the fixed number of hardware registers, require original solutions, for which the authors develop three algorithms of different power and complexity. In the next paper, by Peter Faber, Martin Griebl, and Christian Lengauer, the elimination of loop-carried redundancies among computations involving arrays is targeted, a subject not considered by classical redundancy elimination techniques. Focusing on a typical setting for scientific code, where loop bounds and array subscripts are often expressible as affine functions, they show how to re-orchestrate and utilize techniques for scheduling and automatic parallelization to restructure such loops making them amenable to redundancy elimination involving array expressions. The final paper of this session, by Apan Qasem, David Whalley, Xin Yuan, and Robert van Engelen, describes an alternative approach to that of Fernández and colleagues to reducing the number of load and store instructions in a program. They explore the adequacy of using swap instructions, which are typically used for process synchronization, to coalescing load and store operations. Their results are promising, showing that both the number of executed instructions and of accesses to the memory system are reduced.

The second and final session of the High Performance Compilers topic is devoted to automatic parallelization and compiler support techniques. The first paper, by Daniel Chavarría-Miranda, John Mellor-Crummey, and Trushar Sarang, targets the automatic parallelization of data-parallel programs using multipartitioned data distributions. These are known to offer better parallel efficiency and scalability for multi-directional line-sweep computations than block unipartitionings. In their paper, the authors present a comprehensive study of the compiler techniques implemented in the dHPF compiler supporting the automatic generation of multipartitioned code. The reported performance results are impressive showing that the generated code is most competitive to that of hand-coded parallelizations using multipartitioning. The second paper, by Peter M. W. Knijnenburg, Toru Kisuki, and Kyle Gallivan, deals with iterative compilation, a

highly effective, but time-consuming approach searching for the best program optimizations by profiling many variants and selecting the most efficient one. The authors show that iterative compilation can effectively be guided by static models. Considering loop unrolling and tiling for illustration, they demonstrate that the usage of static cache models allows substantial reductions of the computation costs essentially without any performance deductions of the final code. The next paper, by Vincent Loechner, Benoît Meister, and Philippe Clauss, targets the efficient usage of the memory hierarchy. The reduction of cache misses is here an important source for getting performance improvements, addressed for example by methods for enhancing temporal locality. The authors present an architecture-independent generalization of these methods focusing on innermost loops to non-perfect loop nests. Essentially, this works by minimizing the number of iterations between consecutive accesses of the data in memory. In effect, this reduces cache misses as well as TLB (translation lookaside buffer) misses, while still allowing the additional application of specific architecture-dependent transformations such as blocking. The final paper of this session, by Vivek Sarkar and Stephen Fink, targets dependence analysis for arrays in Java programs that is efficiently enough for usage in just-in-time compilers in contemporary Java Virtual Machines. Features such as the dynamic allocation of arrays in Java requiring pointer-induced aliases of array objects make this a difficult task. The authors address this by a new approach based on sparse congruence partitioning representations built on the so-called array SSA form. Preliminary experimental results reported are encouraging, but underline also the importance of, e.g., enhanced interprocedural information for further improving the precision of the analysis results.

In closing, we would like to express our gratitude to the many people, whose contributions made this conference, and the High Performance Compilers track possible. Above all, we thank the authors who submitted a paper, the Euro-Par Organizing Committee, and the numerous referees, whose excellent work was an invaluable help for the topic committee. We hope you will enjoy the presentations, as well as the papers in the proceedings, finding (many of) them useful for your work and research.