Topic 04
Compilers for High Performance

Samuel P. Midkiff, Barbara Chapman, Jean-François Collard, and Jens Knoop

Topic Chairpersons

We would like to welcome you to the Euro-Par 2000 topic on High Performance Compilers. The presentations, papers, and interactions with fellow researchers promise to be both enjoyable and useful. We also hope that you enjoy your visit to Munich.

The High Performance Compilers topic is devoted to research in the areas of static program analysis and transformation, mapping programs to processors (including scheduling, allocation and mapping of tasks), code generation and compiling for heterogeneous systems. The topic is distinguished from the other compiler oriented topics — #03, #07 and #14 — in Euro-Par by its focus on the application of these techniques to the automatic extraction and exploitation of parallelism.

This year 27 papers, from three continents and seven countries, were submitted. The quality and range of the submitted papers was impressive, and testifies to the continued vibrancy and importance of the field of compilation for high performance computing. All papers were reviewed by three or more referees. Using the referees' reports as guidelines, the program committee picked eleven of the submitted papers for publication and presentation at the conference. Ten papers were selected as regular papers, and one as a research note. These will be presented in four sessions, one of which is a combined session with topic 14, Instruction Level Parallelism and Processor Architecture.

The paper in the combined session with topic 14 (Session 14-B.1 on Thursday afternoon), by Kevin D. Rich and Matthew K. Farrens, describes an implementation of a compiler that automatically partitions code between data read/write operations and data manipulation operations to target processors that support independent data fetch and write instruction streams. This allows more flexibility and parallelism in the scheduling of memory references and computation.

The first full session of the High Performance Compilers topic focuses on automatic parallelization. Reflecting the increasing importance of sparse computations in both industrial and basic science applications, the first two papers describe methods for improving compiler parallelization of sparse code. The first of these, by Gerardo Bandera and Emilio L. Zapata, uses information from high-level directives to perform sparse privatization and thereby parallelize the sparse code. The second paper, by Roxane Adle, Marc Aiguier and Franck Delaplace, uses symbolic analysis to perform more precise dependence analysis on sparse structures to parallelize the sparse codes. The third paper, by Rashindra Manniesing, Ireneusz Karkowski and Henk Corporaal, targets problems at the equally

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 329–330, 2000.
© Springer-Verlag Berlin Heidelberg 2000

important, but opposite end of the computation spectrum — SIMD parallelization of embedded applications using a pattern matching based code generation phase.

The first three papers of the second full session are concerned with program restructuring. The first paper describes the management of temporary arrays arising from the distribution of loops with control dependences. In particular, Alain Darte and Georges-André Silber describe a new type of dependence graph and how it is used to limit the number of temporaries used to store conditionals. The second paper, by Nawaaz Ahmed and Keshav Pingali, describes how block-recursive codes can be automatically generated from iterative versions of a kernel to better exploit data locality across the entire memory hierarchy. The third paper, by Nikolay Mateev, Vijay Menon and Keshav Pingali, describes how to transform linear algebra computations between eager (or right-looking forms) and lazy (or left-looking forms) to enhance later compiler optimizations. Finally, the last paper of the session, by Diego Novillo, Ron Unrau and Jonathan Schaeffer, examines the problem of validating irregular mutual exclusion synchronization in explicitly parallel programs.

The third, and final full session focuses on problems of data layout and parallelism specification. In the first paper, R. W. Ford, M. F. P. O'Boyle and E. A. Stohr show how to place a minimal number of coherence operations in such a way as to eliminate all invalidation traffic in programs with statically decidable control flow. The second paper, by Alain Darte, Claude Diderich, Marc Gengler and Frédéric Vivien, presents a technique to consider together the mapping of loop iterations to processors, and the order of execution (schedule) of those iterations, for better exploitation of parallelism than can be achieved by using strategies which independently arrive at mappings and schedules. Finally, the third paper, by Felix Heine and Adrian Slowik, describes how to use Ehrhart polynomials to precisely determine the amount of static locality, and to use this information to guide data transformations and distributions to increase the quality of a program's data distribution.

In closing, we would like to thank the authors who submitted a contribution, as well as the Euro-Par Organizing Committee, and the scores of referees, whose efforts have made this conference, and the High Performance Compilers track possible.