Workshop 03

Automatic Parallelization and High-Performance Compilers

Workshop 03: Automatic Parallelization and High-Performance Compilers

Yves Robert

Presentation

This workshop deals with all topics concerning automatic parallelization techniques and the production of parallel programs using high performance compilers. Topics of interest include the traditional fields of compiler technology (static analysis, program transformations, scheduling, allocation, mapping, communication optimization, code generation, ...).

Of the 26 submissions to this workshop, 6 were accepted as regular papers, 6 as short papers, and 14 were rejected. One regular paper has been moved from Workshop 11 to this workshop.

Organization

This workshop is divided into three sessions. The first session is devoted to the tuning of automatic parallelization according to features of target architectures (cache policies, memory systems, mapping onto regular arrays, etc). The first paper by Clauss introduces new methods for counting the number of points in a non-linear mapping of a polytope, which has potentially many applications in compiling for parallel machines. The second paper by Ahmad describes a software tool that contains a number of scheduling algorithms for the purpose of program parallelization. The third paper by Wyrzykowski and Kaniewski (moved from Workshop 11) aims at mapping iterative sparse matrix algorithms onto regular processor arrays. In the fourth paper, Zhu and Watson present an interprocedural analysis to optimize the execution of C programs on virtual shared memory systems. Data redistribution is the subject of the fifth paper by Beckmann and Kelly, in connection to parallel library issues.

The second session is devoted to compiling analysis techniques. It is composed of five papers. The first paper by Kotlyar, Pingali and Stodghill addresses the automatic generation of sparse matrix code from dense DO-ANY loop nests via a relational algebra approach. The second paper by Loechner and Mongenet deals with the problem of finding communication-optimal allocation mappings together with their corresponding additional constraints on schedules. The third paper by Besch and Pohl introduces a new algebraic handling of data alignment, with the help of group theory. In the fourth paper, Barrado and Labarta present a new approach for modulo scheduling. The problem is formulated as scheduling Hamiltonian recurrences and solved by dynamic programming techniques. Finally, the fifth paper by Amme and Zehendner describes a data-dependence technique to deal with pointers and list structures in imperative languages.

The third session is devoted to code-generation optimization techniques. It is composed of two papers only. In the first paper, Lefebvre and Feautrier present a technique for parallelizing static control programs with only partial data expansion. In the second paper, Knoop and Mehofer deal with the optimization of (unavoidable) data redistributions in HPF-like programs.

Comments

The typical target programming style in this workshop obviously remains data-parallel imperative languages (HPF), despite the fact that the Call for Papers explicitly welcomed compiler optimizations for other kinds of languages.

We also encouraged submissions on practical experiments to assess the benefits and the limits of current automatic parallelization techniques and programming styles. The bad news are that no application at all is reported upon! But the good news are that sparse matrix computations are addressed in this workshop. Slowly but inexorably, the scope of automatic parallelization techniques is extending, maturing, and gaining practical significance.

Remember, compilers always win in the end! :-)

Acknowledgement

The workshop chair would like to deeply thank the other three PC members, Jean-François Collard, Bill Pugh and Jingling Xue. Their contribution to the success of the workshop has been essential.