Parallel Processing Letters Vol. 7 No. 2 (1997) 113–115

EDITORIAL NOTE

As parallelism emerges as a viable and important concept of computer technology, automatic parallelization of loops is becoming increasingly important and receiving increased attention from researchers. The reasons are (1) that programming parallel computers by hand is impractical in all but the simplest applications and (2) that, by exploiting the parallelism in nested loops, potentially large numbers of processors can be easily utilized and a speed-up of orders of magnitude can be attained.

Methods of loop parallelization have been developed in two research communities: regular array design and parallelizing compilation.

Researchers in regular array design impose regularity constraints on loop nests in order to apply a geometric model in which the set of all parallelizations of the source loop nest can be characterized, the quality of each member of the set can be assessed and an optimal choice from the set can be made automatically. That is, regular array design methods identify optimal parallelizations of highly regular loop nests.

Researchers in parallelizing compilation are interested in faster methods than their colleagues in regular array design and, therefore, often apply heuristics to attain reasonable but not necessarily optimal parallelizations. Parallelizing compilation methods can often cope with less regular loops but do, in general, not produce provably optimal results.

This special issue of Parallel Processing Letters is the product of a seminar held at Schloß Dagstuhl, Wadern, Germany in April 1996. The seminar was organized by the four editors of this issue.

The primary goal of the seminar was to intensify communication between the two communities. In recent years, the methods used in both communities have increasingly converged on the theory of linear algebra and linear programming.

Questions discussed at the seminar included:

- Loop parallelization methods have yielded static parallelism in the past. How can they be made more dynamic, e.g., for handling dynamic dependences and loop bounds or run-time load balancing?

- Algorithms that yield optimal parallelizations are usually computationally complex (often NP-complete). For which applications is this (not) a serious restriction? For which applications do heuristic algorithms yield better performance and what are the heuristics?
- Which parallel programmable computer architectures should the research in loop parallelization aim at? Which inter-processor communication schemes are likely to be prevalent in 5–10 years – or does it matter?
- What do the users of parallelizing compilers expect from loop parallelization?
- Where do methods for deriving parallel hardware and software coincide/differ, and why?
- Which models and techniques in loop parallelization could be extended to a wider class of iterations/recursions/programs?

We have classified the 8 papers of this special issue (somewhat arbitrarily) along the following categories:

Mathematical background. One paper deals with the polyhedral theory used for space-time mapping loops:

- "On Manipulating Z-polyhedra" by Patrice Quinton, Sanjay Rajopadhye and Tanguy Risset.

Dependence analysis. One paper deals with the analysis of data and control dependences for parallelism:

- "Array dataflow analysis for explicitly parallel programs" by Jean-François Collard and Martin Griebl.

Mapping: Three papers investigate mapping techniques:

- "Parallelizing nested loops with approximation of distance vectors: a survey" by Alain Darte and Frédéric Vivien,
- "Reductions for Parallel Execution" by Chi-Chung Lam, P. Sadayappan and Rephael Wenger, and
- Modular transformations and data distribution independent computation" by Hyuk-Jae Lee and José A. B. Fortes.

Optimization: Two papers deal with the optimization of time and space resources:

- "Static analysis to reduce synchronization costs in data-parallel programs" by Manish Gupta and Edith Schonberg, and
- "Memory Reuse Analysis in the Polyhedral Model" by Sanjay Rajopadhye and Doran Wilde.

Code generation: One paper discusses code generation for communication:

- "Communication generation for block-cyclic distributions" by Arun Venkatachar, Jagannathan Ramanujam and Ashwath Thirumalai.

We hope that this issue will contribute to a further stimulation of research in loop parallelization techniques, and we would like to thank all authors and referees for their contribution.

Christian Lengauer
Universität Passau, lengauer@fmi.uni-passau.de

Lothar Thiele
ETH Zürich, thiele@tik.ee.ethz.ch

Michael Wolfe
The Portland Group Inc., mwolfe@pgroup.com

Hans P. Zima
Universität Wien, zima@par.univie.ac.at