JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 5, 331–333 (1988)

Special Issue on Parallel Architectures and Algorithms

Guest Editor's Introduction

SARTAJ SAHNI

University of Minnesota, Minneapolis, Minnesota 55455

Received March 15, 1987

1. THE CONFERENCE

The International Conference on Parallel Processing is an outgrowth of the Sagamore Computer Conference on Parallel Processing. The first Sagamore Conference was held in 1972. The name of the conference was changed, in 1975, to The International Conference on Parallel Processing. Thus, 1987 was the 16th year for the conference. In the 16 years since its inception, the conference has grown considerably. In 1972, there were 17 technical presentations and about 90 participants [1]. All of these were invited. In 1987, there were 174 technical presentations. All were selected through a rigorous reviewing process. The papers on which these presentations were based are included in the proceedings of the conference [3]. There were over 600 participants at the 1987 conference. The tremendous growth in the research activities related to parallel processing is evident in the increased number of submissions to the conference and in the increased attendance at the conference.

2. THE PAPERS IN THIS ISSUE

The conference included numerous technical sessions on compiler techniques, parallel algorithms and data structures, logic programming systems, parallel architectures, programming languages, software support, interconnection networks, image processing, systolic computing, and hypercube computing. Because of space limitations, it is not possible to demonstrate, in this special issue, the breadth of topics covered at the conference. This special issue contains only 5 of the 174 papers presented at the conference. Some of

0743-7315/88 $3.00

the other papers will appear in later issues of this journal. The papers included in this special issue were refereed in accordance with the strict refereeing standards established by the journal. The topics covered by these papers are, in order, DO loop transformation techniques for pipelined processors, evaluation of a hybrid data flow and control flow architecture, multistage interconnection networks, matching the problem size and the number of processors to get optimal speedup, and an environment to prototype parallel algorithms.

For over a decade, researchers have explored techniques to transform DO loops so as to obtain efficient code for pipelined processors. Loop unrolling, fusion, interchange, distribution, etc., are some of the techniques that have been advanced. In their paper "Estimating interlock and improving balance for pipelined architectures," Callahan, Cocke, and Kennedy, introduce the metrics of machine balance and loop balance. These metrics may be used to estimate the performance gains from a particular loop transformation. Using these metrics, the authors show that while loop unrolling does not reduce pipeline interlock, loop fusion does. Consequently, loop fusion can reduce the optimal running time of a DO loop while loop unrolling cannot. Their analysis leads to the formulation of two new transformations: redundant load elimination and unroll-and-jam. These may be used to obtain better run times on a pipelined processor.

Data flow and control flow represent two different approaches to computer design. While almost all commercially available computers employ control flow, data flow promises significant improvements in execution times as it eliminates the sequencing constraints of a control flow computer. Hence, only those sequencing constraints that are essential to a correct computation need be enforced. Carlson and Fortes, in their paper "On the performance of combined data flow and control flow systems: Experiments using two iterative algorithms," report an experimental evaluation of an architecture that combines concepts of both control flow and data flow computers. The matrix multiplication and iterative relaxation problems are used for the experiments. Their experiments study the effects of changing the relative costs of computation, synchronization, and sequencing.

A multistage interconnection network comprises several stages of $k \times k$ switching elements. The network connects $N = 2^n$ inputs to N outputs. All such networks are required to provide at least one path between every input–output pair. This is called the *full access* property. In a *minimal full access* (MFA) network, the minimum possible number of switching elements is used and each stage has the same number of switching elements. An MFA in which the connection pattern between adjacent stages is the same is called a *uniform* MFA (UMFA). Several UMFAs have been proposed for use in multiprocessor computers. All of these are known to be topologically equivalent. The paper "Uniform-full minimal access networks" by Sridhar and Raghavandera examines the conjecture that all UMFAs are, in fact, topologically equivalent. They resolve this conjecture in the negative. For the case of 2×2 switch

networks, they show that there are at least $\Omega(2^{N/32})$ nonequivalent UMFAs when $N \geq 32$.

It is well known that the time required by a multiprocessor computer to solve a problem instance of a particular size is seldom a nonincreasing function of the number of processors. As the number of processors is increased, one reaches a point beyond which a further increase in the number of processors results in an increase in the execution time. This is a result of the increased communication and synchronization overheads. In the paper "Problem size, parallel architecture, and optimal speedup," Nichol and Willard develop an analytical model to predict the performance of an algorithm for the elliptical partial differential equation problem. The model is able to predict the optimal number of processors to use in the solution of this problem. Additionally, the model may be used to evaluate the effects of using different grid sizes, different architectures, and changes in the processor and communication speeds.

The last paper in this special issue is "Environments for prototyping parallel algorithms" by Purtilo, Reed, and Grunwald. This paper describes a design system, POLYLITH, that allows one to specify a parallel algorithm independent of its implementation on a target architecture. This separation of the specification task from the implementation task allows an algorithm to be quickly prototyped and results in a specification that is portable across architectures that run the POLYLITH support system. The paper develops POLYLITH specifications for two applications: simulation of parallel systems and a parallel solution of the Poisson equation.

3. Acknowledgments

To begin with, thanks are due to Professor Tse-yun Feng for giving me the opportunity to manage the technical program for this conference and also for handling the review of the papers submitted by faculty and students at the University of Minnesota. Next, thanks are due to the 453 reviewers of the 487 papers submitted to the conference. They are to be credited for providing their informed and valuable evaluations in a most timely manner. Special thanks go to the subset of these reviewers who were asked to provide a more thorough review of the papers considered for this special issue. In the interest of preserving anonymity of the reviewing process, their names are omitted. Mr. William Ellis spent countless hours cataloging the submitted papers, reviewer evaluations, and accepted papers. His able, responsible, and conscientious handling of all administrative aspects of the reviewing process made it possible to complete this task on schedule. The administrative support of Kathy Boyer, Brent Gregoire, Cindy Olson, Julie Payne, Michelle Torres, and Robin Toy is also appreciated. Finally, Ms. Elaine Smales is to be thanked for collating the camera-ready copies and putting the conference proceedings together.

4. References

1. Feng, Tse-yun. Preface. *Proc. 1975 Sagamore Conference on Parallel Processing.* IEEE Computer Society Press, New York, 1975.
2. Kuck, D. J. *The structure of computers and computation.* Wiley, New York, 1978, Vol. 1.
3. Sahni, Sartaj (Ed.). *Proc. 1987 International Conference on Parallel Processing.* Penn. State Univ. Press, University Park, 1987.