Q.c. 01 - 678

F. Meyer auf der Heide   B. Monien
A. L. Rosenberg (Eds.)

Parallel Architectures and Their Efficient Use

First Heinz Nixdorf Symposium
Paderborn, Germany, November 11-13, 1992
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editors

Friedhelm Meyer auf der Heide
Burkhard Monien
Heinz Nixdorf Institut and FB Mathematik-Informatik
Universität-GH Paderborn, Postfach 1621
W-4790 Paderborn, Germany

Arnold L. Rosenberg
University of Massachusetts, Computer Science Department
Lederle Graduate Research Center
Amherst, MA 01003, USA

CR Subject Classification (1991): F.1.2, B.2.1, B.6.1, C.1.2, F.2

6236

ISBN 3-540-56731-3 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-56731-3 Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera ready by author/editor
45/3140-543210 - Printed on acid-free paper

Preface

The papers in this volume were presented at the

First Heinz Nixdorf Symposium:
Parallel Architectures and Their Efficient Use

in Paderborn, November 11–13, 1992, organized by the Heinz Nixdorf Institute of the University of Paderborn.

The symposium is the first in a series of Heinz Nixdorf Symposia, intended to cover varying subjects from the research spectrum of the Heinz Nixdorf Institute. As intended by the founder of this Institute – Heinz Nixdorf – its research spectrum is interdisciplinary and ranges from computer science to engineering and economics, including basics from natural sciences and related areas from humanistic and social sciences. Currently it focuses on research in the field of parallel computation and its applications in manufacturing technology.

Research in the field of parallel computer architectures and parallel algorithms has been very successful in recent years, and further progress is to be expected. On the other hand, the question of basic principles of the architecture of universal parallel computers and their realizations is still wide open. The answer to this question must be regarded as most important for the further development of parallel computing and especially for user acceptance. The First Heinz Nixdorf Symposium has brought together leading experts in this field to discuss the state of the art, promising directions of research, and future perspectives.

The symposium was organized as a "public day" followed by a "closed workshop" restricted to a smaller audience. The public day was attended by more than 200 participants; talks were given by Wolfgang Paul, Franco Preparata, Marc Snir, Charles Leiserson, and Tom Leighton.

The closed shop was split into four sections:

1. Parallel Computation Models and Simulations
2. Existing Parallel Machines
3. Communication and Programming Paradigms
4. Parallel Algorithms

In these proceedings, we have integrated the five talks from the public day in the appropriate sections. Surveys of the papers included in each section are given on the next pages (with the speakers' names capitalized).

We are in debt to all the people who helped us to make the symposium to something we are happy with and proud of:

– The invited participants:
  Fred Annexstein, Sandeep Bhatt, Michel Cosnard, Lennart Johnsson, Tom Leighton, Charles Leiserson, Fabrizio Luccio, David May, Ernst Mayr, Kurt Mehlhorn, Wolfgang Paul, Franco Preparata, Abhiram Ranade, Larry Rudolph, Marc Snir, Lawrence Snyder, Hal Sudborough, Eli Upfal, Leslie Valiant, Uzi Vishkin, Jean Vuillemin.

– The organizing group:
Bernhard Bauer, Astrid Burger, Michael Figge, Matthias Paul, Uta Schneider.

We are particularly indebted to the Westfalia Foundation for its generous support of the symposium.

Paderborn and Amherst, February 1993

Friedhelm Meyer auf der Heide
Burkhard Monien
Arnold Rosenberg

Survey of Papers

Section 1: Parallel Computation Models and Simulations

The major obstacle for achieving wide acceptance of parallel computers is the lack of a standard programming model for parallel computation, like the Von Neumann model for sequential computation. This section contributes to the discussion about the form such a parallel model could take.

LESLIE VALIANT has earlier proposed his bulk synchronous machine as a standard programming model, this model allows both (via simulations) the use of shared memory and direct interprocessor communication. In his paper he supports this model by presenting a combining mechanism that allows simulation of concurrent shared memory access in his model.

UZI VISHKIN argues in favor of the PRAM as one standard programming model. He discusses his thesis in the light of simplicity of programming and of (at least theoretically) efficient shared memory emulations on realistic parallel machines.

FRIEDHELM MEYER AUF DER HEIDE surveys the state of the art of shared memory simulations on distributed memory machines. The results show that simple, fast, simulations exist. This supports both the PRAM and the bulk synchronous machine as a standard model that can be used at the cost of only moderate loss in efficiency.

ARNOLD ROSENBERG advocates the thesis that parallel machines should consist of a very large number of very simple processors. More complex machines or programming paradigms should be added by sophisticated software. He illustrates this thesis by an examination of multigauging algorithms for SIMD bit-serial processor networks.

MICHEL COSNARD and Pascal Koiran contribute to the parallel complexity theory of computations over the real numbers. They introduce the Real PRAM, compare corresponding parallel complexity classes, and present P-completeness results in this framework.

FRANCO PREPARATA pinpoints the physical limits of the design of very large parallel machines, where the speed of light becomes a significant factor in the communication time. He argues that moderate size machines can use complicated networks, but very large machines should be built of meshes of such moderate size networks.

Section 2: Existing Parallel Machines

Massively parallel computers promise top performance for a wide range of applications at a price which is only a fraction of the price of conventional supercomputers. Such high performance is obtainable for an application only if the architecture of the machine and the communication primitives allow an efficient implementation of the underlying parallel algorithm.

WOLFGANG PAUL has developed a formal method to measure the cost-effectiveness of hardware architectures. Formella, Massone, and Paul use this method to compare the data flow machine Monsoon with the vector machines Cray I and Sparc 2.0.

CHARLES LEISERSON describes the communication networks of the Connection Machine CM-5. The machine contains three such networks: a data network, a control network, and a diagnostic network. The structure of the data network is based on the Fat Tree model.

LENNART JOHNSSON discusses techniques for preserving locality, especially the use of multidimensional address spaces and the partitioning of irregular grids. He also describes the communication primitives which are implemented on the Connection Machine.

BURKHARD MONIEN, Reinhard Lüling and Falk Langhammer describe the interconnection network of the transputer system SC 320 which is organized as a 2-level Clos network. The paper introduces also a new type of network, the "fat mesh of Clos", which combines aspects of efficiency and of realizability.

Many algorithms can be described in the "loosely synchronous" model of parallel programming, where processors alternate between phases of local computation and global communications. LARRY RUDOLPH introduces a parallel architecture supporting this model of parallel programming.

P. Bertin, D. Roncin, and JEAN VUILLEMIN present a number of "programmable active memories" for several applications including long multiplication, RSA cryptography, and data compression. A programmable active memory is a universal hardware co-processor closely coupled to a standard host computer. It is made of a configurable array of up to 14K programmable active bits.

Section 3: Communication and Programming Paradigms

The realization of communication among processors by routers and the discussion of paradigms for the design of parallel algorithms are the topics of this section.

TOM LEIGHTON and Bruce Maggs examine networks and protocols for packet routing and propose adding (pseudo-)randomness to the design of networks in order to achieve fast and fault-tolerant routing networks.

LARRY SNYDER describes an adaptive router, i.e. a router which allows that paths of messages are modified during execution. His Chaos Router is compared with many known routers.

Sergio Felperin, Prabhakar Raghavan and ELI UPFAL study wormhole routing, a routing mechanism that is not well studied but that is widely used in practice. They present analytical and experimental results on the performance of several variants of this routing mechanism.

The next two papers deal with programming paradigms.

FABRIZIO LUCCIO, Linda Pagli, and Geppino Pucci describe three algorithmic scenarios in which parallel solutions can be made especially efficient.

MARC SNIR focusses on the important, but often ignored, issue of scalability of algorithms. He discusses parallel algorithms that are scalable over a wide range of machine sizes. Further, he discusses paradigms of parallel programming languages for expressing scalable parallel algorithms.

Section 4: Parallel Algorithms

During the short history of computer science, algorithms have been designed mainly for sequential computers. The availability of parallel machines presents new challenges. In this section, a few methods are described for using parallel machines efficiently.

Most applications that run well on existing machines do so because they have locality, which can be exploited in algorithms. ABHIRAM RANADE introduces a framework for analyzing locality. His measure of locality allows him to show that several simple problems are inherently nonlocal. His paper also lists several problems for which fast network implementations can be designed.

The divide-and-conquer approach is one of the most successful programming paradigms. ERNST MAYR and Ralph Werchner show how to implement divide-and-conquer algorithms without undue overhead on a wide class of networks; in particular, they give an optimal implementation on hypercubes.

One promising way to take advantage of the communication pattern of an algorithm during the compilation process is to have software that can reconfigure the physical architecture according to the specified logical interconnection topology. FRED ANNEXSTEIN shows that many networks can be efficiently embedded into a hypercube using simple parallel algorithms.

Pancake networks have better diameter and vertex degree than the popular hypercube. Mohammad Heydari and HAL SUDBOROUGH study the diameters of pancake networks. They give new bounds for sorting some conjectured hard "stacks of pancakes."

Contents

Section 1: Parallel Computation Models and Simulations

A Combining Mechanism for Parallel Computers 1
Leslie G. Valiant

A Case for the PRAM As a Standard Programmer's Model 11
Uzi Vishkin

Hashing Strategies for Simulating Shared Memory
on Distributed Memory Machines 20
Friedhelm Meyer auf der Heide

Better Parallel Architectures via Emulations 30
Arnold L. Rosenberg

Relations between Models of Parallel Abstract Machines 37
Michel Cosnard and Pascal Koiran

Frontiers of Parallel Computing 47
Franco P. Preparata

Section 2: Existing Parallel Machines

Cost Effectiveness of Data Flow Machines and Vector Processors 48
Arno Formella, Werner Massonne and Wolfgang J. Paul

The Networks of the Connection Machine CM-5 66
Charles E. Leiserson

Massively Parallel Computing: Data Distribution and Communication 68
S. Lennart Johnsson

A Realizable Efficient Parallel Architecture 93
Burkhard Monien, Reinhard Lüling and Falk Langhammer

Hardware Support for Collective Communication Operations 110
Larry Rudolph

Programmable Active Memories: A Performance Assessment 119
Patrice Bertin, Didier Roncin and Jean Vuillemin

Section 3: Communication and Programming Paradigms

The Role of Randomness in the Design of Interconnection Networks 131
Tom Leighton and Bruce Maggs

Chaos Router: Finally, a Practical Adaptive Router? 146
Lawrence Snyder

An Experimental Study of Wormhole Routing in Parallel Computers 156
Sergio Felperin, Prabhakar Raghavan and Eli Upfal

Three non Conventional Paradigms of Parallel Computation 166
Fabrizio Luccio, Linda Pagli and Geppino Pucci

Scalable Parallel Computers and Scalable Parallel Codes:
From Theory to Practice 176
Marc Snir

Section 4: Parallel Algorithms

A Framework for Analyzing Locality and Portability Issues
in Parallel Computing 185
Abhiram Ranade

Optimal Implementation of General Divide-and-Conquer
on the Hypercube and Related Networks 195
Ernst W. Mayr and Ralph Werchner

Parallel Implementations of Graph Embeddings 207
Fred S. Annexstein

On Sorting by Prefix Reversals and the Diameter of Pancake Networks 218
Mohammad H. Heydari and I. Hal Sudborough

A Combining Mechanism for Parallel Computers*

Leslie G. Valiant

Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138

Abstract. In a multiprocessor computer communication among the components may be based either on a simple router, which delivers messages point-to-point like a mail service, or on a more elaborate combining network that, in return for a greater investment in hardware, can combine messages to the same address prior to delivery. This paper describes a mechanism for recirculating messages in a simple router so that the added functionality of a combining network, for arbitrary access patterns, can be achieved by it with reasonable efficiency. The method brings together the messages with the same destination address in more than one stage, and at a set of components that is determined by a hash function and decreases in number at each stage.

1 Introduction

A general purpose parallel computer needs to have a mechanism for realizing concurrent memory accesses efficiently. Several or all of possibly thousands of processors may wish to read the same memory address at the same time. Alternatively, several or all may wish to write a value into the same address, in which case some convention needs to be adopted about the desired outcome. In either case, the requests will have to converge from the various parts of the physical system to the one location.

If each request is sent directly to the component containing the relevant address, then this component will require time to handle them proportional to the number arriving there. In general, this becomes unacceptable if the number of processors p is large. This overhead, potentially linear in p, can be overcome by implementing the requests in more than one stage. In the first stage, for example, the requests to any one ultimate destination will converge in groups at various intermediate components, where each group is *combined* into a single request to be forwarded in the next stage. In the last stage all extant requests to an address finally converge to the chosen location. Thus the requests can be viewed as flowing from the leaves of a tree to the root. In some instances, as when the concurrent requests implement a read statement, a flow of information in the reverse direction, from the root to the leaves, needs to follow. In these instances, whenever requests are combined at a component, the sources of the requests in that stage are stored at that component, so that a complete record of the structure of the tree is maintained. In a general pattern of requests, accesses to several memory addresses may be present. In that case a combining tree has to be maintained for each address.

What is the most efficacious way of providing a multiprocessor computer with a combining facility that is acceptably efficient for the widest range of concurrent

* This research was supported in part by a grant from the National Science Foundation, NSF-CCR-89-02500.

access patterns? In this paper we shall describe one proposed solution, and provide some analytic, experimental and, also, qualitative arguments in its favor.

In [12] we proposed the BSP model of parallel computation in which the basic medium of inter-processor communication is a *router*, that delivers messages among the components point-to-point, but performs no combining or ther computational operations itself. It was shown that shared memory with arbitrary concurrent accesses could be simulated on a p-processor BSP machine with only constant loss in efficiency asymptotically, if the simulated program had $p^{1+\varepsilon}$ fold parallelism for some positive constant ε. One important advantage of having as the communication medium this simplest option is that it makes possible a competition for the highest throughput router among the widest possible range of technologies, including optical ones. In contrast, a medium that is required to perform more complex operations, imposes more constraints on the technology. The crucial question is whether the extra capabilities of more complex hardware can be simulated in practice on the simple router, with acceptable loss of efficiency.

In this paper we lend support to the position that simple routers can indeed implement concurrent accesses efficiently, by describing an algorithm for this that is more efficient and practical than previous solutions [8], [13]. Experiments suggest, for example, that for $p = 4096$ and with each processor making 32 requests, the cost of arbitrary concurrency patterns as compared with no concurrency is no more than a factor of about 3.5, even if nothing is known about the pattern. If the degree of concurrency is known then this factor is even smaller.

We conclude that it is indeed efficacious to invest the bulk of ones resources for communication in building a simple router with maximal throughput. Although every general purpose parallel machine needs to have mechanisms for implementing arbitrary patterns of concurrent accesses, if, as it appears, difficult access patterns occur rarely enough, then our proposed mechanism for dealing with them is efficient enough that substantial investments in combining networks are not warranted.

2 Multi-Phase Combining

We consider a system consisting of p *components*, each of which has some memory and processing capabilities. A *(q,r)-pattern* among the p component is a set of communication requests in which each component sends at most q requests, each request has a destination that is a memory *address* in a particular component, and at most r requests are addressed to any one component. Distinct components contain disjoint memory addresses. Several requests may share the same address (and therefore by implication also the same component.) We call the set of requests sharing the same address a *group*. The *degree* of a request is the number of elements in its group, and the degree of the pattern is the maximum degree of its elements. Thus if the pattern consists of n groups, of respective sizes $d_1, ..., d_n$ and destinations $t_1, ..., t_n$, then the *degree* of the pattern is $d = \max\{d_1, ..., d_n\}$. Although various alternatives may be considered, in this paper we shall charge $\max\{q, r\}$ units for executing directly a (q, r)-pattern on a router (as in the variant of the BSP model considered in [2].)

The proposed *multi-phase combining* algorithm implements patterns of high degree by decomposing them into a sequence of patterns of low degree. In each phase