Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

317

Timo Lepistö Arto Salomaa (Eds.)

Automata, Languages and Programming

15th International Colloquium
Tampere, Finland, July 1988
Proceedings

Springer-Verlag

Lecture Notes in
Computer Science

Edited by G. Goos and J. Hartmanis

317

Timo Lepistö Arto Salomaa (Eds.)

Automata, Languages and Programming

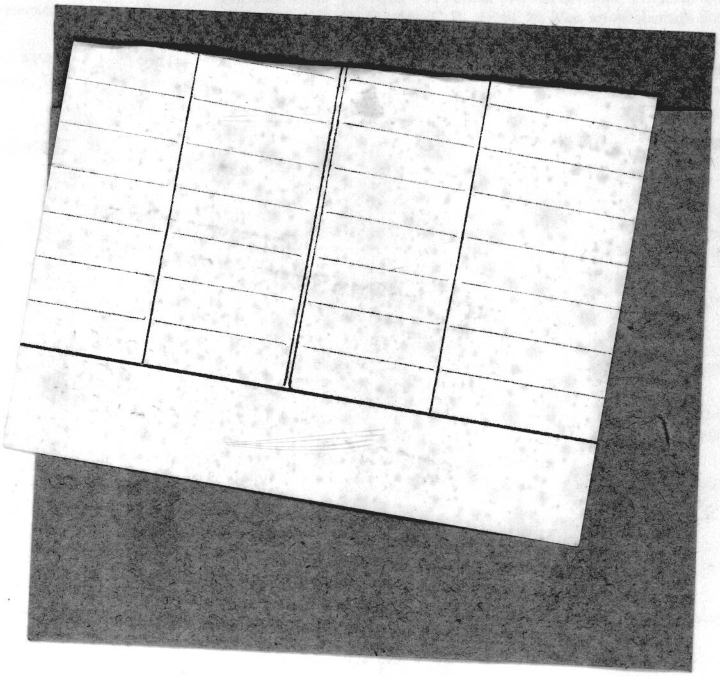
15th International Colloquium
Tampere, Finland, July 11–15, 1988
Proceedings

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo

Editors

Timo Lepistö
Tampere University of Technology
P.O. Box 527, SF-33101 Tampere, Finland

Arto Salomaa
Department of Mathematics, University of Turku
SF-20 500 Turku 50, Finland



CR Subject Classification (1987): C.1, E.1, F, G.2, I.1, I.2.3

ISBN 3-540-19488-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-19488-6 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210

PREFACE

ICALP 88 is the 15th International Colloquium on Automata, Languages and Programming in a series of meetings sponsored by the European Association for Theoretical Computer Science (EATCS). It is a broadly based conference covering all aspects of Theoretical Computer Science including topics like Computability, Automata, Formal Languages, Analysis of Algorithms, Computational Complexity, Data Types and Data Structures, Theory of Data Bases and Knowledge Bases, Semantics of Programming Languages, Program Specification, Transformation and Verification, Foundations of Logic Programming, Theory of Logical Design and Layout, Parallel and Distributed Computation, Theory of Concurrency, Symbolic and Algebraic Computation, Term Rewriting Systems, Cryptography, Theory of Robotics.

ICALP 88 was held at the campus of the Tampere University of Technology from July 11 to July 15, 1988. The organizing committee consisted of T. Lepistö, P. Järvinen, R. Kurki-Suonio, K. Ruohonen, K.-J. Räihä, M. Tienari, T. Granroth, P. Kaila and E. Hirvonen.

The program committee selected 46 papers from 195 extended abstracts and draft papers submitted. Each submitted paper was evaluated by at least four members of the program committee. The final selection was made during a two-day selection meeting in which the following program committee members participated: S. Abramsky, G. Ausiello, J. Díaz, J. Gruska, W. Kuich, J. van Leeuwen, M. Nivat, J. Paredaens, A. Paz, A. Salomaa, S. Skyum, E. Ukkonen and D. Wood. It is a pleasure to thank all those who have submitted papers for consideration, the members of the program committee for their help in the evaluation of the papers and the many who assisted in this process.

We also gratefully acknowledge all the institutions and corporations which supported this conference.

Finally, we would like to thank K.-J. Räihä, K. Ruohonen and P. Kaila, who did a beautiful job in all organizational matters related to the conference.

July 1988	Timo Lepistö Chairman of the Organizing Committee	Arto Salomaa Chairman of the Program Committee

Sponsors of ICALP 88

The main sponsor is IBM, Finland
Other sponsors:
Academy of Finland
Administrative Unit of Häme
City of Pori
City of Tampere
Digital Equipment Corporation
The Ministry of Education, Finland
Nevanlinna Institute
Nokia Oy Ab
Springer-Verlag

EATCS Institutional Sponsors

Bull, Louveciennes, France
Dansk Datamatik Center/ DDC International,Lyngby Denmark
Nixdorf Computer AG, Paderborn, Federal Republic of Germany
Philips Research Lab. Eindhoven, The Netherlands
Siemens ZTI, Munich, West Germany

List of referees for ICALP 88

Abrial,J.R.
Afonte,M.V.
Astesiano,E.
Asveld,P.R.J.
Baaz,M.
Back,R.-J.
de Baer,D.
Baeten,J.C.M.
de Bakker,J.W.
Balcázar,J.L.
Bar Yehuda,R.
Baron,G.
Barringer,H.
Bentley,J.
Berry,D.
Berry,G.
Berstel,J.
Bertin,P.
Bertolazzi,P.
Bodlaender,H.L.
Bougé,L.
de Bra,P.
Brell
Bruynooghe,M.
Brüggemann-Klein,A.
Buss,J.F.
Casas,R.
Chan,E.P.F.
Cherkasova,L.
Chytil,M.
Codognet,P.
Cohen,E.
Cohn,E.R.
Colbourn,C.J.
Coquand,Th.
Costa,R.
Courcelle,B.
Cucker,F.
Curien,P.-L.
Damgaard,I.
Ben David,S.
Despecpoux,J.
Duris,P.
Dybjer,P.
Ehler
Eklund,P.
van Emde Boas,P.
Engelfriet,J.
Etzion,T.
Farinas del Cerro,L.
Feder,T.
Felgentreu, K.-U.
Floréen,P.
Francez,N.
Frandsen,G.
Freitag,B.J.
Frutos Escrig,D.
Gabarro,J.
Gambosi,G.
Ganzinger,H.
Gerth,R.
Geser
van Gestel,E.
Goldreich,O.
Grahne,G.
Grass,W.
Guessarian,I.
Guibas,L.J.
Gyssens,M.
Haddad,R.
Hájek,P.
Hankin,C.
Hardin,Th.
Harju,T.
Harrison,P.
van Hee,K.M.
Heinz,A.
Hennicker
Hesselink,W.H.
Hofri,M.
Holmström,L.L.
Honkala,J.
van Horebeek,I.
Hromkovic,J.
Hussmann
Icking,Chr.
Indermark,K.
Itai,A.
Italiano,G.F.
Janssens,D.
Johnson,D.S.
Johnson,J.H.
Jouannaud,J.P.
Kahn,G.
Kaminski,M.
Kari,J.
Karvi,T.
Katz,S.M.
Kelemenová,A.
Klein,R.
Klop,J.
Kolaitis,P.G.
Korach,E.
Korec,I.
Koskimies,K.
Koymans,R.
Kranakis,E.
Kröger,F.
Kuiper,R.
Küster,G.
Lafont,Y.
Larsen,K.G.
Laville,A.
Le Chenadec,Ph.

Lee,D.
Lenstra,J.K.
Lescanne,P.
Levy,J.J.
Lewi,J.
Linna,M.
Lippe,W.M.
Louchard,S.
Lubiw,A.
MacQueen,D.
Mäkinen,E.
Mannila,H.
Marchetti-Speccamela,A.
Martin,U.
Mascari,G.
Mauny,M.
Mayoh,B.
Mayr,E.
Meyer,J.-J.Ch.
Möncke,U.
Moran,S.
Morlock,M.
Mosses,P.D.
Nagl,M.
Nardelli,E.
Nickl
de Nicola,R.
Nielsen,M.
Niemi,V.
Nijholt,A.
Nurmi,O.
Olderog,E.-R.
Olivié,H.
Ong,C.-H.
Orejas,F.
Orponen,P.
Overmars,M.
Pachl,J.K.
Padawitz,P.
Penttonen,M.
Pepper,P.
Phillips,I.
Plambeck,T.
Poigné,A.
Privara,I.
Protasi,M.
Räihä,K.-J.
Rathmann,P.K.
Reisig,W.
Rem,M.
Richier,J.-L.
Ringwood,G.
Rodriquez Artalejo,M.
de Roever,W.P.
Rosenberg,A.
Ron,E.
Rozenberg,G.
Rovan,B.
Roy,S.
Ruohonen,K.
Rusinowitch,M.
Ruzicka,P.
Ryslinková,J.
Sadler,M.
Salomaa,K.
Saltor,F.
Sassano,A.
Schäffer,A.
Schmerl
Schmidt,E.M.
Schmidt,G.
Schmidt,U.
Schöning,U.
Schor,B.
Schwartzbach,M.
Shmueli,O.
Sifakis,J.
Silberman,G.M.
de Simone,R.
Smolka,G.
Smyth,M.
van de Snepscheut,J.L.A.
Soisalon-Soininen,E.
Soria,M.
Staunstrup,J.
Steinby,M.
Steyaert,J.-M.
Streicher
Sturc,J.
Sturtivant,C.
Subramanian,A.
Sykora,O.
Talamo,M.
Tamminen,M.
Taubner,D.
Thomsen,B.
Tirri,H.R.
Tomasta,P.
Torán,J.
Torras,J.
Traktenbrot
Urbanek,F.J.
Vaandrager,F.
Vajtersic,M.
Vardi,M.Y.
Vavasis,S.A.
Veldhorst,M.
Vickers,S.
Vitanyi,P.M.B.
Volger
Vornberger,O.
Vyskoc,J.
van der Weide,Th.P.
Widmayer,P.
Wiederman,J.
Wilhelm,R.
Wirsing,M.
Yannakakis,M.
Yu,S.
Zaks,S.
Zucker,J.

TABLE OF CONTENTS

Communication Complexity of PRAMs

(Preliminary Version)

Alok Aggarwal
Ashok K. Chandra

IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218, Yorktown Heights, New York 10598

Abstract: We propose a model for the concurrent read exclusive write PRAM that captures its communication and computational requirements. For this model, we present several results, including the following:

Two $n \times n$ matrices can be multiplied in $O(n^3/p)$ computation time and $O(n^2/p^{2/3})$ communication delay using p processors (for $p \leq n^3/\log^{3/2} n$). Furthermore, these bounds are optimal for arithmetic on semirings (using $+$, $\times$ only). For sorting and for FFT graphs, it is shown that communication delay of $\Omega(n \log n/(p \log(n/p))$ is required for $p \leq n/\log n$. This bound is tight for FFT graphs; it is also shown to be tight for sorting provided $p \leq n^{1-\varepsilon}$ for any fixed $\varepsilon > 0$.

Given a binary tree, τ, with n leaves and height h, let $D_{opt}(\tau)$ denote the minimum communication delay needed to compute τ. It is shown that $\Omega(\log n) \leq D_{opt}(\tau) \leq O(\sqrt{n})$, and $\Omega(\sqrt{h}) \leq D_{opt} \leq O(h)$, all bounds being the best possible. We also present a simple polynomial algorithm that generates a schedule for computing τ with at most $2D_{opt}(\tau)$ delay.

It is shown that the a communication delay-computation time tradeoff given by Papadimitriou and Ullman for a diamond dag can be achieved for essentially two values of the computation time. We also present DAGs that exhibit proper tradeoffs for a substantial range of time.

1. INTRODUCTION

It is becoming abundantly clear that much of the complexity in parallel computing is due to the difficulty in communication rather than the computation itself. This issue is likely to become more severe as the number of processors increase. Researchers have recognized this fact, and studied the communication complexity of special purpose interconnection networks [A80, DGS83, PS81, T84, Y79] and of VLSI chips [AUY83, JK84, T79]. However, except for the paper by Papadimitriou and Ullman [PU87] which explores the communication complexity in rather broad terms, very little is known regarding the communication complexity of PRAMs. The purpose of this paper is to study several aspects of this issue, and explore a few techniques that are useful in a better understanding of the communication requirements of PRAMs.

To capture the communication delay in multiprocessor systems, we model both the computational problem to be solved as well as the multiprocessing machine that is solving it. The model is similar to that used by Papadimitriou and Ullman [PU87] but since it differs in some essential features, it is described below.

The multiprocessing machine is taken to be a concurrent-read, exclusive write (CREW) PRAM in which each processor is provided with an unlimited amount of local random access memory. Processors can read simultaneously from the same location in the *global* memory, but two or more are not allowed to write simultaneously into the same location. The input variables are initially stored in the global memory, and the final outputs must be eventually stored in the global memory. The processors are MIMD, but work synchronously. In order to model the communication delay and computation time, it is convenient to restrict the machine such that at every time step, the processors do one of the following:

(i) In one communication step, a processor can write and then read a word from the global memory, or it may do nothing.

(ii) In a computation step, a processor can perform an operation (which we describe below) on at most two words that are present in its local memory.

The computation problem to be solved is usually presented as a directed acyclic graph (DAG) with its nodes corresponding to operations (on at most two variables) and its arcs corresponding to the values computed by performing such operations. Thus, the in-degree of each node is bounded by two, and if the out-degree of any node is more than one, then all its outgoing edges contain the same associated value. The nodes of the DAG with in-degree zero (i. e., the leaves) are available in the global memory at time $t = 0$, and likewise, the nodes of the DAG with out-degree zero (i. e., the roots), have to be eventually written into the global memory. Any processor correctly *computes* the given node if and only if the following conditions are satisfied:

(a) The node is computed only after all its children have been computed.

(b) A processor can perform the operation given at the node only if the corresponding values of its children reside in its local memory; it performs the operation on these values in one computation step. A processor can write a computed value into the global memory in a communication step, and can then read a value from the global memory in the same step.

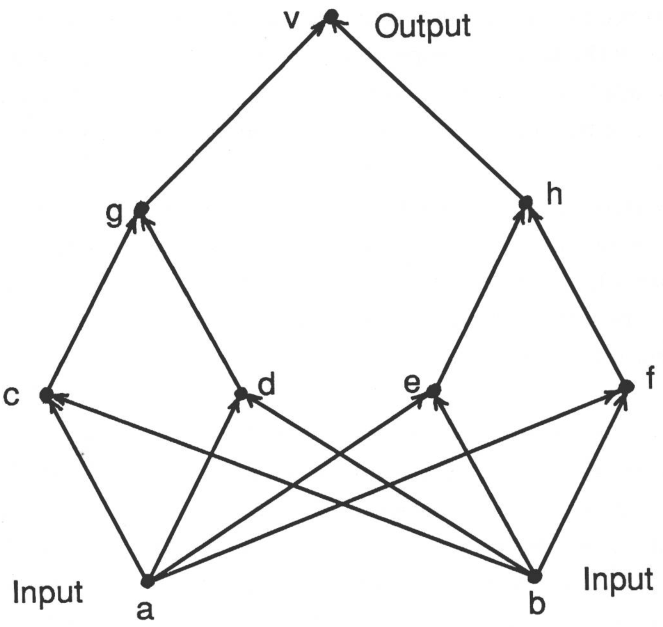
For the computation of a given DAG, each of its nodes is assigned to one or more processors that *compute* that node. Hence, the problem of computing the DAG reduces

to that of finding a schedule for performing the operations given at the nodes which ensures that each node of the DAG is computed correctly. Figure 1 demonstrates a nine node DAG and a schedule that computes it in 5 communication steps and 3 computation steps. This DAG can also be computed in 4 (resp. 3) communication steps and 4 (resp. 7) computation steps.
Remark: Since many algorithms for computing a given problem can be converted into a DAG, one can usually obtain a plethora of DAGs for the same problem. Usually, we will be concerned with a fixed DAG for a given problem, although we will sometimes deviate and give lower bounds for a class of DAGs that can be used to solve problems such as matrix multiplication, or sorting.

Three kinds of resources are relevant -- the number of processors used in the multiprocessing machine, the overall computation time, and the overall communication delay. The computation time (communication delay, resp.) of a schedule, S, is the number of computation (communication, resp.) steps used by this schedule. The computation time for the DAG is the minimum computation time over all schedules that correctly compute the given DAG, and likewise, for communication delay. Note that the minimum communication delay and the minimum computation time for a given DAG may not be achievable by the same schedule.

2. MATRIX MULTIPLICATION AND RELATED RESULTS

Consider the computations of DAGs that are obtained when two $n \times n$ matrices are multiplied using scalar multiplications and additions only. We establish an optimal bound of $\Theta(n^2/p^{2/3})$ on communication delay where p denotes the number of processors and $p \leq n^3/\log^{3/2} n$. Indeed, the communication delay of $\Theta(n^2/p^{2/3})$ may be contrasted with the optimal speed up of $\Theta(n^3/p)$ with respect to the number of computation steps for the same problem. The lower bound for matrix multiplication given in Theorem 2.3 can be extended to Boolean matrix multiplication (over a semi-ring containing AND and OR operations), to all pair shortest path problem (where only the semi-ring operations of MIN and + are allowed), and to transitive closure of matrices (if the closure is computed over a closed semi-ring with "+" and "$\times$" as two operations). Furthermore, the upper bound of Theorem 2.1 can be generalized to demonstrate the optimality of the two bounds for Boolean Matrix Multiplication (within a constant factor) and to those for computing the all-pair shortest paths and transitive closure within $\log n$ factors. For $p \leq n^2/\log n$, a PRAM with p processors can multiply an $n \times n$ matrix with an n-dimensional vector in $O(n^2/p)$ computation time and with $O(n^2/p)$ communication delay. Since all entries of matrix are initially stored in



Schedule: Comm. Step 1: P_1, P_2, P_3, P_4 read a.
Comm. Step 2: P_1, P_2, P_3, P_4 read b.
Comp. Step 1: $P_1, P_2, P_3,\ P_4$ compute $c,\ d,\ e,\ f$ respectively.
Comm. Step 3: P_2, P_4 write $d,\ f$ respectively. P_1, P_3 read $d,\ f$ respectively.
Comp. Step 2: P_1 computes g, P_3 computes h.
Comm. Step 4: P_3 writes h, P_1 reads it.
Comp. Step 3: P_1 computes v.
Comm. Step 5: P_1 writes v.

Figure 1: Example of a DAG and a schedule that computes it

the global memory, every PRAM algorithm would require $\Omega(n^2/p)$ communication delay. Furthermore, since a total of $\Omega(n^2)$ multiplications are required by every sequential program for matrix-vector multiplication (see [AHU74], pp. 428-435, for details), every PRAM algorithm would require $\Omega(n^2/p)$ computation steps and this establishes optimal bounds for matrix-vector multiplication.

Theorem 2.1: Two $n \times n$ matrices can be multiplied by p processors in $O((n^3/p) + \log n)$ computation time and $O((n^2/p^{2/3}) + \log n)$ communication delay using only $(\times , +)$.

Proof: Partition the two $n \times n$ matrices, A and B, in a natural manner, into $p^{2/3}$ submatrices of sizes $n/p^{1/3} \times n/p^{1/3}$ each, and for $1 \le i, j \le p,^{1/3}$ let these matrices be denoted by $A_{i,j}$ and $B_{i,j}$ respectively. For $1 \le i, j, k \le p,^{1/3}$ let the p processors read $A_{i,j}$ and $B_{j,k}$ such that each processor gets a unique pair of $(A_{i,j}, B_{j,k})$. Note that this global memory access takes $O(n^2/p^{2/3})$ communication delay. Now, compute the matrix $C_{i,j,k} = A_{i,j} \times B_{j,k}$ in $O(n^3/p)$ computation time. If $C = A \times B$ and if C is also partitioned, in a natural manner, into $p^{2/3}$ submatrices, $C_{i,k}$, for $1 \le i, k \le p^{1/3}$, then it is easy to see that $C_{i,k} = \Sigma C_{i,j,k}$. Now, summing these submatrices naively would require $O(n^2 \log n/p^{2/3})$ communication delay. However, using a *pipelining strategy,* which we give below, these submatrices can be added in $O((n^2/p^{2/3}) + \log n)$ communication delay. This yields an overall communication delay of $O((n^2/p^{2/3}) + \log n)$ and an overall computation time of $O((n^3/p) + \log n)$. For $p \le n^3/\log^{3/2} n$, these bounds are $O(n^2/p^{2/3})$ and $O(n^3/p)$, respectively.

To sum the matrices $C_{i,j,k}$ for $1 \le j \le p^{1/3}$, and fixed i, k, use $2p^{1/3} - 1$ processors such that $p^{1/3}$ of these processors contain the submatrices $C_{i,j,k}$ for $1 \le j \le p^{1/3}$ and the $2p^{1/3} - 1$ processors compute a DAG in the form of a complete binary tree that has $p^{1/3}$ leaves and whose internal nodes are '+'. These processors can now add any $p^{1/3}$ elements present at the leaves in $O(\log p)$ time and, in fact, they can *pipeline* $n^2/p^{2/3}$ elements (that reside in the processors corresponding to each leaf) so that if at any instant, the processors that correspond to, say the l-th level of the tree are adding the elements which would eventually yield, say, the s-th entry of $C_{i,k}$ then at the same instant the processors which correspond to the $(l-1)$-th level, are adding the elements which would eventually yield the $(s+1)$-th entry of $C_{i,k}$. ■

For the lower bound on the communication delay for matrix multiplication, we need the following technical Lemma; the proof of this Lemma can be found in [HK81]:

Lemma 2.2: Let a PRAM compute the matrix multiplication of two $n \times n$ matrices, A, B using scalar additions and multiplications only. During this computation, if any processor reads at most s elements of A and B, and computes at most s partial sums of the product C, then this processor can compute a total of at most $2s^{3/2}$ multiplicative terms for these partial sums. ■

Theorem 2.3: Let G be any DAG with in-degree two that corresponds to an algorithm for multiplying two $n \times n$ matrices using $(+, \times)$ only and p processors. Then, any