Lecture Notes in Computer Science 805

Michel Cosnard Afonso Ferreira
Joseph Peters (Eds.)

Parallel and Distributed Computing

Theory and Practice

First Canada-France Conference
Montréal, Canada, May 1994
Proceedings

Springer-Verlag

TP301.6-53
P222.3
1994

9561558

Michel Cosnard Afonso Ferreira
Joseph Peters (Eds.)

Parallel and Distributed Computing

Theory and Practice

First Canada-France Conference
Montréal, Canada, May 19-21, 1994
Proceedings

Centre Jacques Cartier

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

LIP

Université de Montréal
Concordia University
CRIM - Montréal

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
D-76131 Karlsruhe, Germany

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

Volume Editors

Michel Cosnard
Afonso Ferreira
CNRS - Laboratoire de l'Informatique du Parallélisme
Ecole Normale Supérieure de Lyon
46, Allée d'Italie, F-69364 Lyon Cédex 07, France

Joseph Peters
School of Computing Science, Simon Fraser University
Burnaby, B.C. V5A 1S6 Canada

CR Subject Classification (1991): D.4.2-4, D.4.7, C.2.1, C.2.4, F.1.2, G.1, I.3.5

ISBN 3-540-58078-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-58078-6 Springer-Verlag New York Berlin Heidelberg

CIP data applied for

Typesetting: Camera-ready by author
SPIN: 10131120 45/3140-543210 - Printed on acid-free paper

Preface

With the arrival of the 90's, a successful series of small annual workshops have been organized in Vancouver (Canada) in the summers of 1990 and 1992 and in Marseille (France) in the summers of 1991 and 1993. The main subject of such workshops was information dissemination in interconnection networks for distributed memory parallel computers.

The first Canada-France Conference on Parallel Computing grew out of such workshops, along with the well established and very productive collaboration between Canadian and French researchers, aiming to foster collaboration between theoreticians who study the design and analysis of parallel and distributed algorithms and networks, and practitioners who apply, adapt, and extend the theoretical results to solve real-world problems.

This Conference, that is open to all researchers from the international community working in the area, is a complement to the annual workshops. Its scientific program consists of four sessions composed by one invited speaker followed by five contributed talks each. Their themes are parallel algorithms and complexity, interconnection networks and distributed computing, algorithms for unstructured problems, and structured communications. The contributed papers in this proceedings were selected by the Program Committee on the basis of referee reports. Each paper appearing in these proceedings was reviewed by at least two referees who judged the papers for originality, quality, and consistency with the themes of the conference. Nick Pippenger's invited paper was also refereed. We wish to thank all of the authors who responded to the call for papers, our invited speakers, and all of the referees and Program Committee members who reviewed papers.

We are grateful to our partner, the Centre Jacques Cartier, and to the PRC C3 of the French CNRS for financial support, to Centre de Recherche en Informatique de Montréal, to Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, and Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon for secretarial support, and to Concordia University for providing the facilities for the Conference. Also, we especially thank Valerie Roger, Jean-Louis Duclos, Eliane Fleury, and Jocelyne Richerd, responsible for most of the secretarial duties related to the conference, as well as the members of the Organizing Committee for doing triple duty. In addition to handling organizational details including local arrangements, publicity, and funding, they served as referees and as Program Committee members.

March 1994 M.Cosnard, A.Ferreira, J.Peters

Préface

Avec l'arrivée des années 90, une série réussie de petits séminaires de recherche ont été organisés à Vancouver (Canada) pendant les étés 1990 et 1992 ainsi qu'à Marseille (France) pendant les étés 1991 et 1993. Le sujet principal de ces séminaires était l'étude des communications dans les réseaux d'interconnexion des ordinateurs parallèles à mémoire distribuée.

La première conférence Canada-France sur le Calcul Parallèle est issue de ces séminaires, couplée avec la collaboration bien établie et très productive entre chercheurs Canadiens et Français, dans l'objectif de développer la collaboration entre les théoriciens qui étudient la conception et l'analyse d'algorithmes parallèles et distribués et les réseaux, et les praticiens qui appliquent, adaptent et étendent la recherche théorique pour résoudre des problèmes du monde réel.

Cette conférence, qui est ouverte à tout chercheur de la communauté internationale travaillant dans le domaine du calcul parallèle, est donc une suite logique aux séminaires annuels. Son programme scientifique est constitué de quatre sessions comportant chacune un conférencier invité suivi de 5 intervenants. Les thèmes de ces sessions sont respectivement les algorithmes parallèles et la complexité, les réseaux d'interconnexion et l'informatique distribuée, les algorithmes pour des problèmes non-structurés, et les communications structurées. Les articles qui sont publiés dans ces actes ont été sélectionnés par un Comité de Programme sur la base des rapports des relecteurs. Chaque article a été jugé par au moins deux relecteurs sur la base de son originalité, de sa qualité et de sa pertinence avec les thèmes de la conférence. La contribution invitée de Nick Pippenger a aussi été examinée par le comité de lecture.

Nous souhaitons remercier tous les auteurs qui ont répondu à l'appel à contribution, les conférenciers invités, tous les relecteurs et les membres du comité de programme qui ont jugé de la qualité des articles.

Nous remercions également notre partenaire, le Centre Jacques Cartier, le CRIM à Montréal et le PRC C3 du CNRS Français pour le support financier, le Département d'Informatique et de Recherche Opérationnelle de l'Université de Montréal, le Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon pour le support de secrétariat, ainsi que l'Université de Concordia qui a fourni les facilités pour la conférence.

Nous voulons remercier particulièrement Valérie Roger, Jean-Louis Duclos, Eliane Fleury, et Jocelyne Richerd, qui ont assumé la plupart des tâches administratives et de secrétariat liées à la conférence, ainsi que les membres du comité d'organisation pour leur triple responsabilité. En plus de la prise en charge de l'organisation incluant les préparatifs sur place, la publicité et la recherche de financement, ils ont également accepté d'être relecteurs et membres du comité de programme.

Mars 1994 M.Cosnard, A.Ferreira, J.Peters

Committees

Steering Committee

Organising Committee

Program Committee

Contents

Juggling Networks

Nicholas Pippenger

University of British Columbia, Vancouver, BC V6T 1Z4, Canada

Abstract. Switching networks of various kinds have come to occupy a prominent position in computer science as well as communication engineering. The classical switching network technology has been space-division-multiplex switching, in which each switching function is performed by a spatially separate switching component (such as a crossbar switch). A recent trend in switching network technology has been the advent of time-division-multiplex switching, wherein a single switching component performs the function of many switches at successive moments of time according to a periodic schedule. This technology has the advantage that nearly all of the cost of the network is in inertial memory (such as delay lines), with the cost of switching elements growing much more slowly as a function of the capacity of the network. In order for a classical space-division-multiplex network to be adaptable to time-division-multiplex technology, its interconnection pattern must satisfy stringent requirements. For example, networks based on randomized interconnections (an important tool in determining the asymptotic complexity of optimal networks) are not suitable for time-division-multiplex implementation. Indeed, time-division-multiplex implementations have been presented for only a few of the simplest classical space-division-multiplex constructions, such as rearrangeable connection networks. This paper shows how interconnection patterns based on explicit constructions for expanding graphs can be implemented in time-division-multiplex networks. This provides time-division-multiplex implementations for switching networks that are within constant factors of optimal in memory cost, and that have asymptotically more slowly growing switching costs. These constructions are based on a metaphor involving teams of jugglers whose throwing, catching and passing patterns result in intricate permutations of the balls. This metaphor affords a convenient visualization of time-division-multiplex activities that should be of value in devising networks for a variety of switching tasks.

1. Introduction

In this paper we will present a metaphor for describing the construction and operation of time-division-multiplex networks, and use it to present a new time-division-multiplex implementation of an explicit construction for expanding graphs, which are an essential component in many constructions for switching networks. Both the new metaphor and the main techniques for construction of time-division-multiplex networks will be illustrated in Sect. 2 by a well known construction for rearrangeable connection networks. This construction was described in the context of space-division-multiplex networks by Beneš [6] in 1964. The time-division-multiplex implementation was first described by Marcus [13] in 1970, and has recently been rediscovered by Ramanan, Jordan and Sauer [23]. The resulting implementation is "time-slot interchanger" in the sense of Inose [9]. In Sect. 3 we indicate how these methods can be adapted to other types of switching networks. The main obstacle for such applications is the requirement for "expanding graphs" (and related objects) presented by many constructions for switching networks. In Sect. 4 we present a time-division-multiplex implementation of a well known construction for expanding graphs (and, more generally, for graphs with a prescribed "eigenvalue separation ratio"). This construction was first proposed by Margulis [14] in 1974. Quantitative estimates essential for its application were provided by Gabber and Galil [8] in 1981, and improvements to these estimates have been given by Jimbo and Maruoka [10], whose version of the space-division-multiplex construction we follow. In Sect. 5 we present some open problems prompted by this work.

2. Connectors

Imagine a *juggler* who can with complete reliability throw balls to a fixed height, so that they always return a fixed amount of time after they are thrown. All amounts of time considered in this paper will be multiples of some fixed unit of time that will be called the *pulse*. Suppose that our juggler can take a ball at each pulse from an external agent, the juggler's *source*, and can give a ball at each pulse to another external agent, the juggler's *sink*. Suppose further that at each pulse the juggler can execute either of two *moves*, which will be called the *straight* and *crossed* moves. In the straight move, the juggler rethrows the ball that returns from the air (if any such ball returns), and gives the ball taken from the source to the sink (if any such ball is taken). In the crossed move, the juggler throws the ball taken from the source (if any is taken), and gives the ball that returns from the air to the sink (if any returns).

Now imagine a *chain* of jugglers; that is, a finite sequence of jugglers $J_1, \ldots, J_\mu$ in which J_λ is the source of $J_{\lambda+1}$, and $J_{\lambda+1}$ is the sink of J_λ, for $1 \leq \lambda < \mu$. (The source of J_1 and the sink of J_μ are external to the chain. They will be called the *source* and *sink* of the chain.) We assume that the jugglers may have different "spans" (where the *span* of a juggler is the amount of time between the throw of a ball and its return), but that all of these are multiples of a common pulse. Depending on the spans of the various jugglers, and on the sequence of straight and crossed moves executed by each juggler, the sequence of balls passed by the source of the chain (and empty pulses during which no ball is passed) will be rearranged in some way before being passed to the sink of the chain.

In what follows, we shall regard the span of each juggler as a fixed and unchanging attribute of the juggler, while we regard the sequence of moves as being variable. How does each juggler decide what sequence of moves to execute?

Our assumption will be that each juggler has a partner, called the juggler's *cox*, who calls out the name, "straight" or "crossed", of the move to execute at each pulse. How does the cox decide what sequence of moves to call? Our assumption will be that each cox is also a juggler who juggles a fixed sequence of balls. A cox has no source or sink, and always executes straight moves, rethrowing each ball as it returns from the air. We shall assume that a ball returns at each pulse (there are no empty pulses), so that the number of balls being juggled by the cox is equal to the cox's span (which may be different from the cox's partner's span). Finally, we shall assume that each ball juggled by the cox has one of two *colors*, say *red* for "straight" and *blue* for "crossed", and that the cox calls out the move corresponding to the color of each ball as it is rethrown. Thus each cox calls for a periodic sequence of moves, corresponding to the cyclic sequence of colors of balls in the cox's pattern, with a period that is equal to the cox's span.

We can now give a simple example showing how a coxed chain of jugglers can serve as a model for a time-division-multiplex rearrangeable connection network. Let $n = 2^\nu$ be an integral power of 2. Consider a chain of $2\nu - 1$ jugglers $J_1, \ldots, J_{2\nu-1}$. Suppose that jugglers $J_1, \ldots, J_\nu$ have spans $2^0 = 1, \ldots, 2^{\nu-1} = n/2$, respectively, and that jugglers $J_{\nu+1}, \ldots, J_{2\nu-1}$ have spans $2^{\nu-2} = n/4, \ldots, 2^0 = 1$, respectively. Suppose further that all $2\nu - 1$ coxes have span $2^\nu = n$.

Suppose that the source of the chain just described passes it a sequence of balls at successive pulses. Let us divide the pulses into a sequence of *frames*, with each frame comprising n successive pulses. The sequence of balls passed by the source to the chain may be broken into frames, with each frame of balls comprising the balls passed to the chain during a frame of pulses. The sequence of balls passed by the chain to its sink may be broken into frames in a similar way. Furthermore, we may establish a correspondence between source frames and sink frames in the following way. Imagine that each juggler in the chain executes only crossed moves, so that the stream of balls from the source is passed on to the sink after a fixed delay, equal to the sum of the spans of the jugglers in the chain (which is in this case $3n/2 - 2$). Thus each source frame corresponds to a sink frame that is the series of n pulses during which the balls of the source frame emerge from the chain in this situation. The positions of the n balls within their frame will be called *slots*. We shall index the slots of each frame from $1, \ldots, n$ (slot 1 is the earliest, and slot n the latest, slot of its frame).

Theorem 0. *For every permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, there exist patterns for each cox that cause each ball that is passed by the source to the chain in slot i of a frame to be passed by the chain to its sink in slot $\pi(i)$ of the corresponding frame.*

The proof of this theorem, which is implicit in the work of Marcus [13] in 1970, is based on the construction of Beneš for rearrangeable connection networks. This space-division-multiplex construction employs $(2\nu - 1)2^{\nu-1}$ switching elements (2×2 crossbar switches), arranged in $2\nu - 1$ *stages*, with each stage comprising

$2^{\nu-1}$ crossbars. In the time-division-multiplex implementation of this construction, each of the $2\nu - 1$ jugglers in the chain will simulate the $2^{\nu-1}$ crossbars of the corresponding stage.

The space-division-multiplex construction is usually described recursively. In the drawing resulting from this description, crossbars are depicted as "boxes" and the wires interconnecting them are depicted as "lines" that follow "perfect shuffle" interconnection patterns. It is possible to redraw the this picture, however, so that the wires that carry the signals from the inputs to the outputs remain parallel to each other, with the crossbars of each stage conditionally exchanging the signals on wires separated by a fixed distance (depending upon the stage). This can in fact be done so that the distance in each stage is just the span that we have assigned to the corresponding juggler.

When this redrawing has been done, we see that the task of a juggler for each pair of slots (separated by the span of the juggler) is either to leave them unaffected, or to exchange the balls in these two slots. In the latter case, we need to "delay" the contents of the earlier slot by a number of pulses equal to the span, and to "advance" the contents of the later slot by the same amount. Since we cannot implement negative delays, we add a constant delay, equal to the span, to all slots of the frame. With this adjustment, each juggler's task is either to delay both slots by the span (which can be accomplished by three crossed moves at appropriate pulses), or to delay the earlier slot by twice the span and the later slot not at all (which can be accomplished by a crossed move, followed by a straight move, followed by another crossed move). Thus in any case the juggler can be instructed to perform the appropriate sequence of moves by a suitable pattern for the cox.

We may summarize the import of Theorem 0 by saying that a time-division-multiplex rearrangeable connection network with $n = 2^\nu$ slots can be implemented by a *juggling network* with $2\nu - 1 = O(\log n)$ jugglers, overall delay $3n/2 - 2 = O(n)$, and total memory $(3n/2 - 2) + (2\nu - 1)2^\nu = O(n \log n)$. (In the expression for the total memory, the term $(3n/2 - 2)$ represents the memory for the principal jugglers in the chain, while the term $(2\nu - 1)2^\nu$ represents the memory of the coxes.) This yields an extremely attractive time-division-multiplex implementation, since the only aspect of the cost that grows as fast as the size of the corresponding space-division-multiplex network (as $O(n \log n)$) is the total memory, which can be furnished by relatively inexpensive technology (inertial delay lines), whereas the number of high-speed switching elements (represented by the jugglers) grows much more slowly (as $O(\log n)$).

In our description of juggling networks, we have assumed that jugglers execute their moves instantaneously, so that a ball received by a juggler executing a straight move is passed on at the same pulse. In practice there would be a fixed overhead time for a juggler, which might be a large fixed multiple of the pulse. In the chain of jugglers we have described, and more generally in any juggling network in which all balls are processed by the same number of jugglers, this overhead delay can be ignored in the analysis of the network, and it results merely in the addition of a constant delay per juggler being added to the overall delay. Even in more complicated juggling networks, with different numbers of jugglers on various paths between the source and the sink (as is necessary, for example, for the efficient construction of superconcentrators), this overhead delay can be taken into account by setting up "time zones" for the various jugglers, and introducing extra delays to compensate for differences in time zones. Thus we shall maintain the convenient fiction that jugglers act instantaneously, as it will have no effect on our conclusions and will simplify our analysis.

3. Applications

The great economy and elegance of the construction given in Sect. 2 leads us to seek other applications for these ideas. The natural starting point is the class of switching networks with interconnection patterns similar to that of the Beneš network. Some prominent members of this class are (1) the spider-web interconnection networks (see Pippenger [19,20]), (2) the Cantor non-blocking network [7], and (3) the Batcher bitonic sorting network [5]. The first two of these are externally controlled interconnection networks analogous to the Beneš network, and require no further comment. The Batcher bitonic sorting network, however, is based on comparators, and we should say something about how these devices can be realized by jugglers.

As described by Batcher [5], a comparator is a finite automaton that sorts two records received at its inputs, producing the same two records in sorted order at its outputs. To do this, it receives the records one bit at a time, with the bits of the keys by which the records are to be sorted preceding any other data in the records, and with the bits of the keys being received in order of decreasing significance. As long as the bits of the two input records remain identical, these identical streams of bits are reproduced at the outputs. Once the bits of the input keys differ, the correct sorted order is established, and the remainders of the records are reproduced at the outputs in this order. Viewed as a finite automaton, a comparator requires two bits of state information to keep track of whether or not the sorted order has been established and, if so, what that order is.

A time-division-multiplex implementation of a comparator entails three jugglers: a principal juggler who juggles balls representing the successive bits of the records, an assistant who juggles balls representing the state of the comparator (these balls will be of three distinct colors, representing the three possible states of the automaton), and a cox who instructs the other two jugglers as to which of the larger and smaller records should appear in the earlier and later output slots. In this way one can easily construct a time-division-multiplex implementation of Batcher's bitonic sorting network [5] with $O\big((\log n)^2\big)$ jugglers, overall delay $O(n)$ and total memory $O\big(n(\log n)^2\big)$.

To go beyond these simple applications, however, it is necessary to employ one of the essential tools of the theory of switching networks: expanding graphs (or, more generally, graphs with favorable eigenvalue separation ratios). Armed with an efficient time-division-multiplex implementation of this tool, we can explore the possible time-division-multiplex analogs of the following kinds of networks: (1) concentrators and superconcentrators, as introduced by Pinsker [16] and Valiant [24] (see also Pippenger [21]), (2) non-blocking connection networks, following Bassalygo and Pinsker [4] (see also Pippenger [17]), (3) sorting networks, following Ajtai, Komlós and Szemerédi [1,2] (see also Pippenger [18]), and (4) self-routing networks, as introduced by Arora, Leighton and Maggs [3] and Pippenger [22]. We shall not delve further into any of these applications here, but will describe in Sect. 4 a time-division-multiplex implementation for expanding graphs that should be of use in attacking all of them.

4. Expanders

This section is devoted to the time-division-multiplex implementation of expanding graphs. Our implementation will be based upon a particular explicit construction for expanding graphs, originated by Margulis [14], with improvements due to Gabber and Galil [8] and Jimbo and Maruoka [10].

We shall construct a basic expanding graph, which is a regular bipartite multigraph $G = (A, B, E)$, in which every vertex (in $A \cup B$) has degree 8 (meets 8 edges in E), and in which A and B each contain n vertices, where $n = m^2$ is a perfect square, and $m = 2^\mu$ is a perfect power of 2 (so that $n = 4^\mu$ is a perfect power of 4).

We shall do this by describing 8 perfect matchings $E_1, \ldots, E_8 \subseteq A \times B$, the union $E_1 \cup \cdots \cup E_8$ of which is E. To describe these matchings, we let $\mathbf{Z}_m$ denote the ring of integers modulo m, and identify both A and B with the direct product $\mathbf{Z}_m \times \mathbf{Z}_m$, which we shall regard as having for its elements the 2-element columns of elements from $\mathbf{Z}_m$. Each of the matchings E_i will then have the form

$$E_i = \{(z, \pi_i(z)) : z \in \mathbf{Z}_m \times \mathbf{Z}_m\},$$

where π_i is a permutation of $\mathbf{Z}_m \times \mathbf{Z}_m$ defined by an affine mapping of the form

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u \\ v \end{pmatrix}.$$

Thus it will suffice to specify, for each $i \in \{1, \ldots, 8\}$, the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and the column $\begin{pmatrix} u \\ v \end{pmatrix}$.

For one particular construction given by Jimbo and Maruoka [10], the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is one of the matrices $\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$ or their inverses $\begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}$, and the column $\begin{pmatrix} u \\ v \end{pmatrix}$ is one of the columns $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ or their negatives $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$. Thus it will suffice to show how the permutations corresponding to each of these matrices and columns can be implemented by juggling