Parallel Processing and Parallel Algorithms

Springer Science+Business Media, LLC

Seyed H. Roosta

Parallel Processing and Parallel Algorithms

Theory and Computation

With 194 Illustrations

Springer

Seyed H. Roosta
Department of Computer Science
State University of New York
Oswego, NY 13126
USA

Library of Congress Cataloging-in-Publication Data
Roosta, Seyed H.
Parallel processing and parallel algorithms: theory and computation / Seyed H. Roosta
p. cm.
Includes bibliographical references.
ISBN 978-1-4612-7048-5 ISBN 978-1-4612-1220-1 (eBook)
DOI 10.1007/978-1-4612-1220-1
1. Parallel processing (Electronic computers) 2. Parallel programming (Computer science)
3. Computer algorithms. I. Title.
QA76.58R66 1999
004'.35—dc21 99-13243

Printed on acid-free paper.

Production managed by Jenny Wolkowicki; manufacturing supervised by Jeffrey Taub.
Photocomposed copy prepared from the author's Microsoft Word files.

9 8 7 6 5 4 3 2 1

ISBN 978-1-4612-7048-5 SPIN 10708016

This book is dedicated with affection and respect to the memory of my father, Seyed Abolghasem Roosta. To my mother, Mahjabin, for her generosity and to my wife, Sima, for her affectionate support and encouragement. She endured the long hours I spent on this book. Without her understanding and support, this book would not be a reality.

Preface

Motivation

It is now possible to build powerful single-processor and multiprocessor systems and use them efficiently for data processing, which has seen an explosive expansion in many areas of computer science and engineering. One approach to meeting the performance requirements of the applications has been to utilize the most powerful single-processor system that is available. When such a system does not provide the performance requirements, pipelined and parallel processing structures can be employed. The concept of parallel processing is a departure from sequential processing. In sequential computation one processor is involved and performs one operation at a time. On the other hand, in parallel computation several processors cooperate to solve a problem, which reduces computing time because several operations can be carried out simultaneously. Using several processors that work together on a given computation illustrates a new paradigm in computer problem solving which is completely different from sequential processing. From the practical point of view, this provides sufficient justification to investigate the concept of parallel processing and related issues, such as parallel algorithms. Parallel processing involves utilizing several factors, such as parallel architectures, parallel algorithms, parallel programming languages and performance analysis, which are strongly interrelated.

In general, four steps are involved in performing a computational problem in parallel. The first step is to understand the nature of computations in the specific application domain. The second step involves designing a parallel algorithm or parallelizing the existing sequential algorithm. The third step is to map the parallel algorithm into a suitable parallel computer architecture, and the last step involves writing the parallel program utilizing an applicable parallel programming approach.

Parallel Processing

Parallel processing is making a tremendous impact in many areas of computer applications. A growing number of applications in science, engineering, business and medicine are requiring computing speeds that hardly can be achieved by the current conventional computers. These applications involve processing huge amount of data or performing a large number of iterations. Parallel processing is one of the approaches known today which would help to make these computations feasible. It includes the study of parallel architectures and algorithms.

Another reason to utilize parallel processing whose importance has been recognized can be illustrated by the real-time applications. In a real-time application a computer needs certain data from the other resources in order to solve a computational problem. The system receives data from several resources at the same time and it is not feasible to store data arriving from the resources for later processing, because the data become meaningless if not used immediately. The foregoing applications are representative of a host of situations in which the probability of success in performing a computational task is increased through the use of a parallel computer utilizing parallel processing. Many factors contribute to the performance of parallel systems, such as interaction among processors, hardware architecture and parallel programming approach.

Parallel Algorithms

The most important ingredient for parallel processing is parallel algorithms or parallel solution methods, and there has been considerable interest in the development of parallel algorithms. Given a problem to be solved in parallel, a parallel algorithm describes how the problem can be solved on a given parallel architecture by dividing the problem into subproblems, by communicating among processors, and by joining the partial solutions to produce the final result.

Several rough taxonomies of parallel architectures and parallel languages are beginning to emerge, but there remains a lack of understanding of how to classify parallel algorithms and applications and how a class of parallel algorithms relates to a given architecture or programming language, or even to another class of parallel algorithms.

A good parallel algorithm results from looking for parallelism that might be inherent in a sequential algorithm for a given problem. For example, algorithms based on divide-and-conquer strategy usually have an inherent parallel nature.

There are two approaches in designing parallel algorithms with respect to the number of processors available. The first is to design an algorithm in which the number of processors utilized by the algorithm is an input parameter, which means that the number of processors does not depend on the input size of the problem. The second approach is to allow the number of processors used by the parallel algorithm to grow with the size of the input, which means that the number of processors is not an input parameter but is a function of the input size. By using a division-of-labor scheme, an algorithm designed utilizing the second approach can always be converted into an algorithm suitable for the first approach.

A number of constraints arise in the design of parallel algorithms which are not present in the design of sequential algorithms. These constraints have to be highlighted together with the development of various performance measures for parallel algorithms.

Parallel Programming Languages

It has become clear in recent years that parallel programming is a subject of study in its own right, primarily because it is now recognized that the use of parallelism can be beneficial as much to the applications programmer as to the systems programmer.

The parallel programming languages are based on two categories, parallel programming abstractions that are based on mutual exclusion of access to an individual memory location and those that raise the level of abstraction to processes which communicate by sending messages to each other. Sending a message is a higher level action that can be implemented on physically distributed processors.

Each parallel programming approach suggests a particular hardware configuration which best matches its language primitives. In designing primitives for programming, a wide variety of issues must be considered, Asynchronous programming can be used for programming multiprocessors or distributed systems whereas synchronous parallel solutions are suitable for use on array and vector processors.

Book Organization

This book advocates the concept of parallel processing and parallel algorithms. It aims to cover well one aspect of the analysis of parallel computers, which is the essence of the architectures. It attempts to cover the relationship between parallel programming approaches and machines, or algorithms and architectures.

Chapter 1 (Computer Architecture) establishes a hardware theme that runs through the book. Since a knowledge of architecture features helps in the explanation of some of the programming concepts, a broad hardware overview is given in this chapter which covers the basic parallel computer architectures.

Chapter 2 (Components of Parallel Computers) introduces primitives of parallel computers, including memory, interconnection networks, compilers, operating systems and input/output constraints.

Chapter 3 (Principles of Parallel Programming) examines the principles of parallel programming with regard to issues that are involved, such as message passing compared with shared-address-space parallelism, mapping the algorithms into specific architectures, level of parallelism and granularity problems.

Chapter 4 (Parallel Programming Approaches) covers different parallel programming languages for transforming sequential programs into parallel forms. The parallel programming approaches are used in different parallel systems.

Chapter 5 (Principles of Parallel Algorithm Design) focuses on the structure of the parallel algorithms with regard to design, performance measures and complexities. The theory of parallel algorithms is becoming very important in the area of parallel processing. So, from the theory's point of view, this issue provides a challenging range of problems with new rules for the design and analysis of algorithms.

The next three chapters are dedicated to a variety of parallel algorithms. We have selected the important area of interest in the field of parallel processing which concentrates on determining and taking advantage of the inherently parallel nature of the problems. In this regard we essentially concentrate on problems which are known to have efficient parallel solutions.

Chapter 6 (Parallel Graph Algorithms) treats a variety of parallel graph algorithms. The chapter is devoted to a discussion of a number of basic graph problems and presents the design and analysis of efficient parallel algorithms for them.

Chapter 7 (Parallel Search Algorithms) focuses on the most prominent search methods. In this chapter, we deal with two basic problems concerned with finite lists of elements in the context of parallel processing. These are the problems of selection and searching.

Chapter 8 (Parallel Computational Algorithms) deals with computational algorithms that are of fundamental importance in scientific computations. We informally classify parallelism in computational algorithms demonstrating various types of parallelism such as matrix multiplication and systems of linear equations.

Chapter 9 (Data flow and Functional Programming) describes data flow computing, which is a fundamentally different issue in parallel processing and can be achieved by data flow languages. The purpose of this contribution is to deal with the architecture of data flow computers. SISAL is a functional language with no explicit parallel control constructs, but its data types and its constructs for expressing parallelism and operations are specifically chosen to produce efficient code for large-scale scientific computations.

Chapters 10 (Asynchronous Parallel Programming) focuses on asynchronous computations when all program fragments are initially regarded as parallel, independent and unordered. Any constraint on their interactions is formulated as explicit or implicit individual conditions associated with fragments. The chapter introduces three different languages, of which Modula-2 is used in hybrid systems of SIMD-MIMD computers.

Chapter 11 (Data Parallel Programming) introduces a high-level notation that simplifies parallel programming and enhances portability. This provides compilers with enough information to allow them to generate efficient parallel code for both shared-memory multiprocessors and distributed-memory multicomputers. The languages presented are Dataparallel C, a variant of the original

C* language developed by Thinking Machines Corporation and Fortran 90, which has vector and array operations.

Chapter 12 (Artificial Intelligence and Parallel Processing) discusses knowledge processing, which is a fast-growing area of computer science and engineering and how the processing speed of rule-based expert systems may be increased by utilizing parallel processing. We discuss Concurrent Prolog as a parallel logic language and Multilisp, which is a modification of an existing language (Lisp). The basic idea behind Multilisp is parallel expression evaluation.

Audience

The intended audience includes advanced undergraduate and beginning graduate students as well as anyone interested in the wonder of parallel processing. Some knowledge of design and analysis of algorithms is helpful but is not necessary. The book provides many references from which such background can be obtained.

Features

Some of the book's most distinctive features are:

- It covers the majority of the important approaches in parallel processing and algorithms.
- The book raises a number of research issues. This indicates that the book may serve as a source of inspiration for students and researchers who want to.be engaged in the area of parallel processing.
- One of the most important features provided is to enhance the reader's ability to create new algorithms or to modify existing algorithms in order to make them suitable for parallel processing.
- Since parallel algorithms and parallel processing are treated in separate sections, one is able to study as much of the parallel material as desired and use the book in the design and analysis of algorithms or principles of parallel processing.
- A comparison of the different parallel programming languages and architectures is presented to address the efficiency and applicability of the languages with regard to the processing environment.

Acknowledgments

I want to express my gratitude to the staff of Springer-Verlag Publishing Company for an excellent job with editing and production of this book. On the production side, I have benefitted by the diligence and help of Jenny Wolkowicki and Hal Henglein. I am grateful for their valuable assistance and professional editing support. A special thanks is reserved for Dr. Bill Sanders and Mr. John Kimmel, my editors, for encouragement and support of this project.

I would like to send personal thanks to my family members, my wife Sima, my children Nasim, Maryam, and Mahsa, my mother, my brother, my sisters, my father- and mother-in-law, and my brothers-in-law, for their reliable and unconditional love and understanding. The State University of New York at Oswego and the Department of Computer Science provided an active and nurturing environment to conduct this project.

State University of New York
Oswego, NY
September 1999

Seyed H. Roosta

Contents