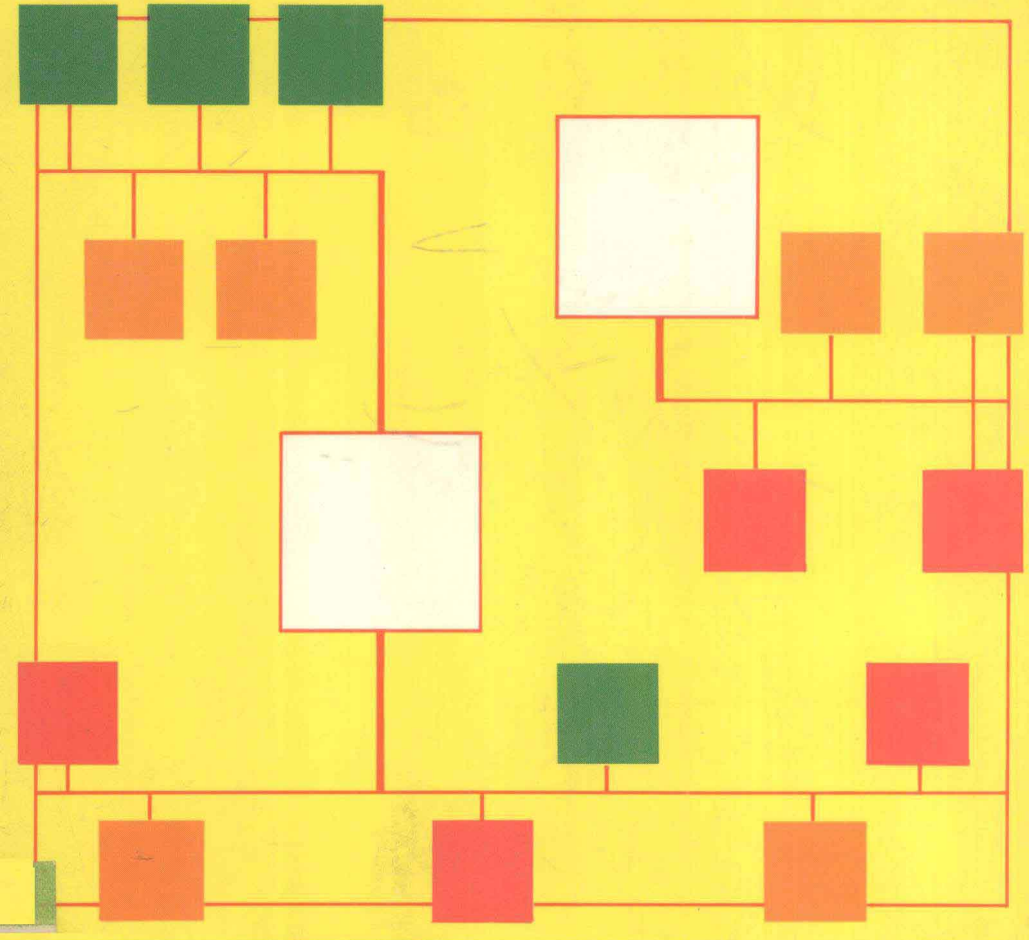
An Introduction to DISTRIBUTED AND PARALLEL COMPUTING



JOEL M. CRICHLOW

AN INTRODUCTION TO DISTRIBUTED AND PARALLEL COMPUTING

Joel M. Crichlow

The University of the West Indies

PRENTICE HALL

New York London Toronto Sydney Tokyo

First published 1988 by
Prentice Hall International (UK) Ltd,
66 Wood Lane End, Hemel Hempstead,
Hertfordshire, HP2 4RG
A division of
Simon & Schuster International Group

Printed and bound in Great Britain by
A. Wheaton and Co. Ltd, Exeter

Library of Congress Cataloging-in-Publication Data

Crichlow, Joel M., 1948–
An introduction to distributed and
parallel computing / Joel M. Crichlow.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-13-481094-5 (pbk.)
1. Electronic data processing – Distributed processing.
2. Parallel processing (Electronic computers) I. Title.
QA76.9.D5C75 1987
ISBN 0-13-481086-4

British Library Cataloguing in Publication Data

Crichlow, Joel M.
An Introduction to Distributed and Parallel Computing
1. Parallel processing (Electronic computers)
2. Electronic data processing – Distributed processing
I. Title
004'.35 QA76.6

ISBN 0-13-481086-4
ISBN 0-13-481094-5 pbk

1 2 3 4 5 92 91 90 89 88

AN INTRODUCTION TO DISTRIBUTED AND PARALLEL COMPUTING

Dedicated to
my wife Valerie,
my children Vaughn, Brook, Jamie and Amanda,
my father Cecil,
and to the memory of my dear mother Elmyra

PREFACE

This text introduces the reader to several different streams of activity that fall under the heading 'distributed and parallel computing'. Both hardware and software aspects are touched on. The material presented should be adequate for a university introductory level course in this subject.

The student is assumed to have had an introduction to programming, data structures, file design and operating systems. Some knowledge of computer organization would be helpful but no previous exposure to communications technology is required. Readers who do not have this background should still be able to appreciate some of the concepts that are discussed. Many of the technical terms are explained and are set in ***bold italic*** on their first appearance. There is a relatively extensive glossary to which one may refer in order to clarify the passages in which these terms are used subsequently.

There are seven chapters. Chapter One is an introduction to all the areas that are discussed in the following six chapters. Chapter Two presents the ways in which computer systems have been organized to facilitate distributed and parallel computing. Chapter Three gives an introduction to communications technology and computer networks. Chapter Four discusses aspects of operating systems design for distributed and parallel computing. Chapter Five deals with the popular client-server model of distributed computing. Chapter Six focuses on distributed database systems and Chapter Seven provides an informal introduction to some parallel programming languages.

There are questions at the end of each chapter to aid the reader in the assimilation of the concepts presented. For those who would like to explore any of these areas in greater detail there is a list of references following each chapter.

I must acknowledge the help of a number of people who have contributed in different ways to making this book possible. I am appreciative of the support of my colleagues at the University of the West Indies in

Computer science, Ken Cazabon, Noel Kalicharan, Krishna Brijpaul, Sheik Yussuff, Margaret Bernard, Graham Taylor, Roger Barnes and Wayne Sheppard; the entire staff in Mathematics, but, in particular, David Beckles, David Owen, Hydar Ali, Ed Farrel and Philbert Morris; and in Seismology, John Shepherd and William Aspinal.

Thanks to Professor Bruce Bolt of the University of California, Berkeley for introducing me to large scale seismic computations. I am also grateful to Professor Peter Brown, Professor Brian Spratt and others at the Computing Laboratory, University of Kent at Canterbury who contributed to making my stay there as a Research Fellow a productive one; and the editorial staff at Prentice Hall International for their guidance during the preparation of this text.

Permission was kindly granted by the following:

Addison-Wesley to take an abstract from Needham, R.M. and Herbert, A.J., 1982, *The Cambridge Distributed Computing System*, Addison-Wesley, London.

Dr R.P.A. Collinson to abstract from his paper 'Operating System Interfaces to LANs' presented at the University of Kent at Canterbury in September, 1984.

Mr Ian Utting to take an abstract from his paper 'Distributed High Quality Printer Servers' presented at the University of Kent at Canterbury in September, 1984.

John Wiley & Sons, Ltd to take an abstract from Turner, D.A., 'A new implementation technique for applicative languages', *Software – Practice and Experience*, **9**, 1979, 31–49, © 1979, John Wiley & Sons, Ltd.

John Wiley & Sons, Ltd to take an abstract from Brownbridge, D.R., Marshall, L.F. and Randell, B., 1982, 'The Newcastle Connection or UNIXes of the World Unite!', *Software – Practice and Experience*, **12**, 12, 1147–1162 © 1982, John Wiley & Sons, Ltd.

The following are trademarks: Ada (US Government, Ada Joint Program Office); Connection Machine (Thinking Machines Corporation); Cray (Cray Research, Inc.); DEC (Digital Equipment Corporation); DECnet (Digital Equipment Corporation); UNIX (AT & T Bell Laboratories).

I am indebted to my wife Valerie, my father and the other members of my family who provided all the encouragement; and thanks to God in whom 'we live, and move, and have our being'.

J.M.C.
St. Augustine
1987

CONTENTS

CHAPTER ONE

INTRODUCTION

Computers can be physically linked via a communications channel to facilitate the sharing of hardware and software resources, and to allow the immediate and accurate transfer of information over distances ranging from less than a meter within a single room to thousands of kilometers across continents and over oceans. Such an arrangement also allows jobs and processes to be distributed to separate computers where they can be executed in parallel resulting in an increase in the number of jobs done in unit time. Issues which pertain both to the hardware and software design of such an interconnection of autonomous computers can generally be classified as ***distributed computing***.

The demand for significant increases in computer processing speeds over those provided by uniprocessor systems has resulted in the design of single computer configurations containing many thousands of processing units. The parallelism in program execution is now not only at a job or process level, but has reached down to the point where individual machine instructions can be allocated to separate processors. The hardware and software issues pertaining to such single computer multiprocessor arrangements can generally be classified as ***parallel computing***.

In this book, we will be looking at hardware organization and software design for distributed and parallel computing. We have made a distinction between distributed and parallel computing in order to reflect what seems to be the dominant view among the computer fraternity. One will still find instances, here and elsewhere, where these terms are used interchangeably. Indeed, it can be safely said that the computation can be distributed within a parallel computer environment, or that the distributed computing environment can be used to exploit the parallelism in some computation.

1.1 A BRIEF HISTORY

Computer technology developed comparatively slowly after Blaise Pascal's calculating machine appeared in 1642. Almost two centuries elapsed before Charles Babbage's difference engine (1812–1832) and analytical engine (1833–1871) were built. It took another century before the electromechanical computers of the late 1930s and early 1940s appeared. However, the pace of development quickened considerably after the creation of the first electronic computers in the 1940s and early 1950s. These initial electronic wonders were bulky, used large amounts of power and were often unreliable (Randell, 1975; Spencer, 1983).

These early systems could only be used by one person at a time. The development of the transistor and, shortly thereafter, integrated circuitry meant that less bulky, more powerful and more reliable computers could be built. It subsequently became apparent that single user uniprogramming operation was grossly inefficient.

Together with the development of the hardware, there was the development of software systems to manage these hardware resources with a view towards making more efficient use of the available computing power. Operating systems soon came on the scene. The earliest operating systems afforded only ***batch stream operation***, where a number of jobs were placed in a ***batch*** of data cards which were then input to the computer. The operating system would then handle the batch by executing each job in turn (see Figure 1.1).

However, these batch systems, operating in a uniprogramming environment, did not make efficient use of the computer resources. The

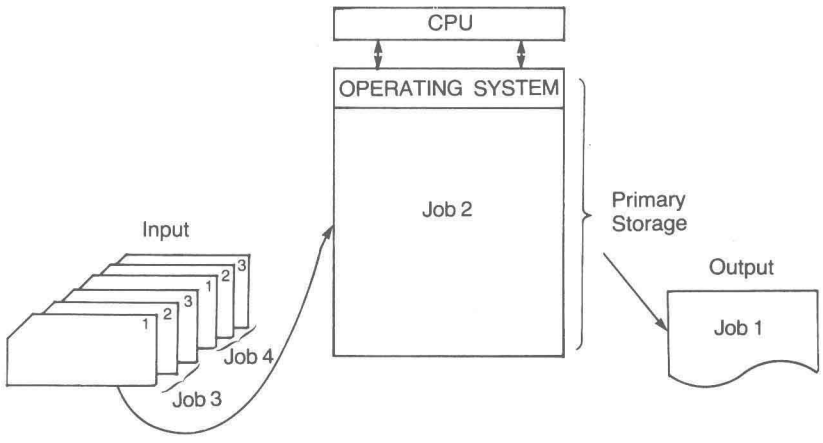


Fig. 1.1 Batch processing in a uniprogramming environment

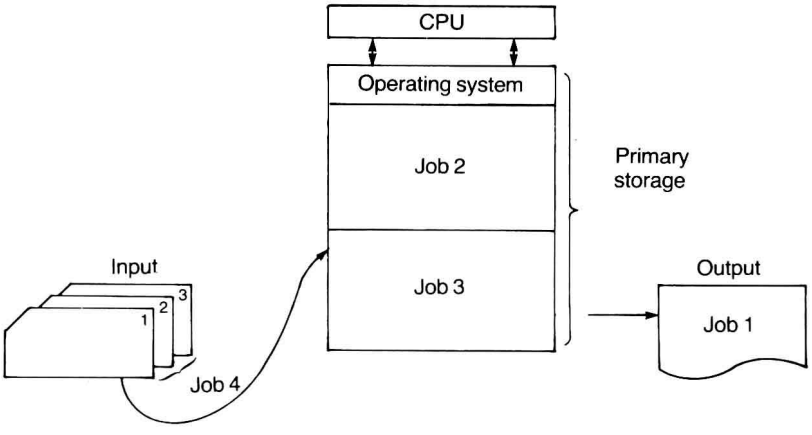


Fig. 1.2 Batch processing in a multiprogramming environment

great difference in speeds between the central processing unit (CPU) and the input/output peripherals resulted in poor use of the CPU, since there would often be significant CPU idle time while waiting on a slow peripheral. This led to the development of multiprogramming which allowed more than one program to be in execution at the same time (see Figure 1.2).

In the early single-user hands-on environment, ***interactive*** operation was possible, i.e. the programmer could examine and modify elements of the program during processing. Batch stream processing did not provide this interactive facility. However, there are applications which are better handled in an interactive mode. Therefore, with the development of key-input terminals and the necessary interrupt-driven operating systems, interactive programming was reintroduced. This soon led to another dimension in shared computer facilities.

Time-sharing systems, where more than one user can run programs on the computer from separate terminals at the same time were developed (see Figure 1.3). Such systems were in relatively common use by the early 1970s. It was soon possible to support ***real-time*** systems, where the computer was expected to respond to a query immediately, e.g. airline reservation systems, or monitor the operation of a physical process, e.g. temperature levels in a turbine. Although most of the computers used had only one CPU, multiprocessor systems were introduced into some environments to provide even more time-sharing and real-time capabilities.

Towards the end of the 1960s and during the early 1970s, two significant developments in the history of computing occurred. These developments: microprocessor technology and computer networking,

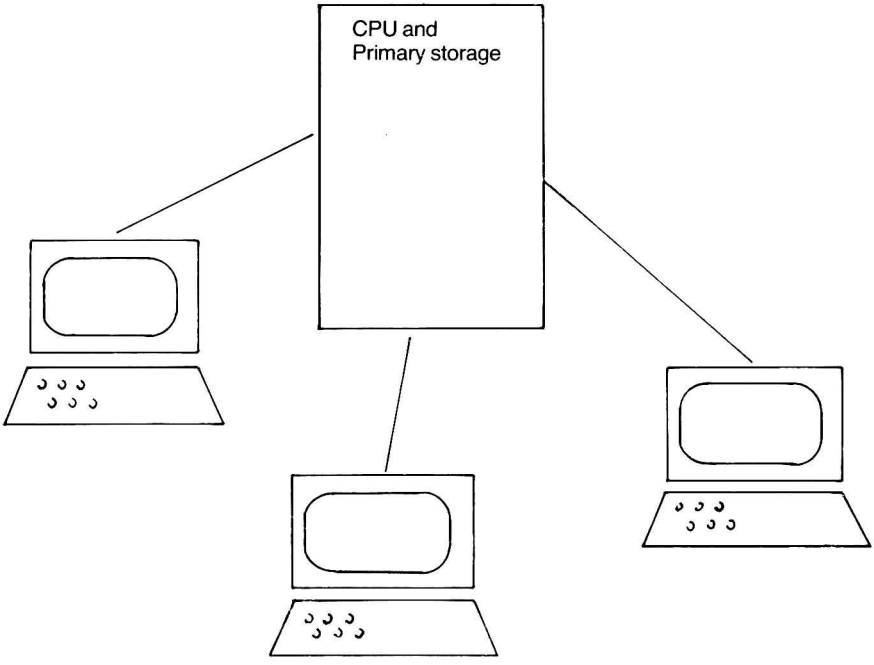


Fig. 1.3 Interactive terminal environment

opened up tremendous opportunities in the fields of computing and communication.

The great reduction in the size of computer processors, made possible by very large scale integrated circuitry, brought about a significant decrease in the cost of computer hardware. There has also been a vast increase in reliability. The versatility provided by these smaller components has made it possible to use microprocessors in many areas. Computer systems with thousands of processing elements constitute one of these areas. These systems now offer very many opportunities for parallel computing.

Communications technology is another of the areas that have benefited from microelectronics. We now have not only the home microcomputer but also the facility to link that home computer to other computers via communication networks (Davies and Barber, 1973; Davies et al, 1979). This has presented many avenues for processing and transferring information and the sharing of computer resources.

In the following sections we will open small windows to the areas of distributed and parallel computing that will be discussed in this text. Each section is expanded as a chapter.

1.2 COMPUTER ORGANIZATION FOR PARALLEL AND DISTRIBUTED COMPUTING

On-line access to many processors at the same time presents many opportunities for parallel and distributed computing. The computational task can be divided into several subtasks and different processors can be allocated to execute these subtasks. There must be some control to ensure that the participating processors cooperate.

A large class of problems for which computer-aided solutions are sought lend themselves to distribution. Indeed, in the area of large-scale scientific computations – aeronautics, nuclear physics, etc. – parallelism in processing seems like the only feasible way (Parter, 1984).

Different computer organizations have materialized to meet the demand for parallel and distributed computing:

(a) There are ***multiprocessor systems*** with a few processors which share a common memory.
(b) There are ***massively parallel systems*** with thousands of processing elements, where each element has a dedicated memory module.
(c) There are ***multicomputer systems*** which include computers each with its own private memory. These computers are connected via a high-speed communication link and are localized within a relatively small area (usually a single room).
(d) There are ***computer networks*** which link autonomous computers via a ***communication network***.

Issues which arise when the computation is distributed include:

(a) how to achieve effective communication among the separate subtasks; and
(b) how to synchronize the subtask activity to ensure that the correct results are generated.

Parallel systems fall into a number of categories. There are systems which contain multiple processing elements capable of executing the *same* instruction on distinct sets of operands simultaneously. There are also systems with multiple processing elements each capable of executing a *different* instruction sequence simultaneously on distinct sets of operands. Furthermore, there are machines which can switch back and forth between these different modes of operation.

Multicomputers and computer networks permit many forms of distribution in the computation. The computational tasks can be categorized by type and different computers can be dedicated to perform particular types of functions. For example, I/O on slow peripherals could be distinguished from CPU intensive activity. Floating point computation could be distinguished from text editing, and so on.

On the other hand, processors may all handle the same type of job. The scheduling discipline may be designed so that it generates an equitable distribution of the processing load among the communicating processors.

In order to make good use of the parallel architecture, guidelines and tools for algorithm and program development must exist. We will look at this in Chapter 7 where we discuss parallel programming languages. However, we must observe here that the availability of techniques for constructing parallel algorithms serves as a strong motivation to build machines to perform the execution.

This has been the case in the design of some non von Neumann-type computers, namely ***data-flow*** and ***reduction machines*** (see section 2.5). A fundamentally different approach to program code formulation has stimulated research and development in this area.

1.3 COMMUNICATIONS AND COMPUTER NETWORKS

In Chapter 3 we will look at some communication principles and expand on the computer network, which was introduced in the previous section. Computer networks facilitate the exchange of information among computers. Expensive hardware resources, e.g. a high speed disk or high quality printer, and software resources, e.g. a database or large compiler can be shared among the users of the network.

There is the facility to access information at remote sites in negligible time. Processing tasks in a particular job may be distributed among sites thus using available computer power more efficiently. There is, too, the increased reliability provided by the availability of interconnected alternative resources in the event of an isolated breakdown. These capabilities all combine to produce better price/performance ratios in the operation of computer systems.

The computer network includes the computers on which the users run programs or applications. These computers are called ***hosts***. These hosts are connected by a ***communication subnet*** of ***communication processors***. The communication processors are usually referred to as ***IMPs*** (interface message processors) or ***PSNs*** (packet switch nodes) (see Figure 1.4). The subnet is the environment where the main communication functions are undertaken (Tanenbaum, 1981).

Networks are placed into two broad classes: wide area or long-haul networks (***WANs***) and local area networks (***LANs***). The wide area network covers a wide geographic area, even connecting continents whereas the LAN covers a small geographic area, e.g. an office block or university campus. LANs do not usually have a communication subnet.

Computer network development has benefited significantly from the pioneering work done by ***ARPA*** (Advanced Research Projects Agency) of the U.S. Department of Defense (now called DARPA). They gave us