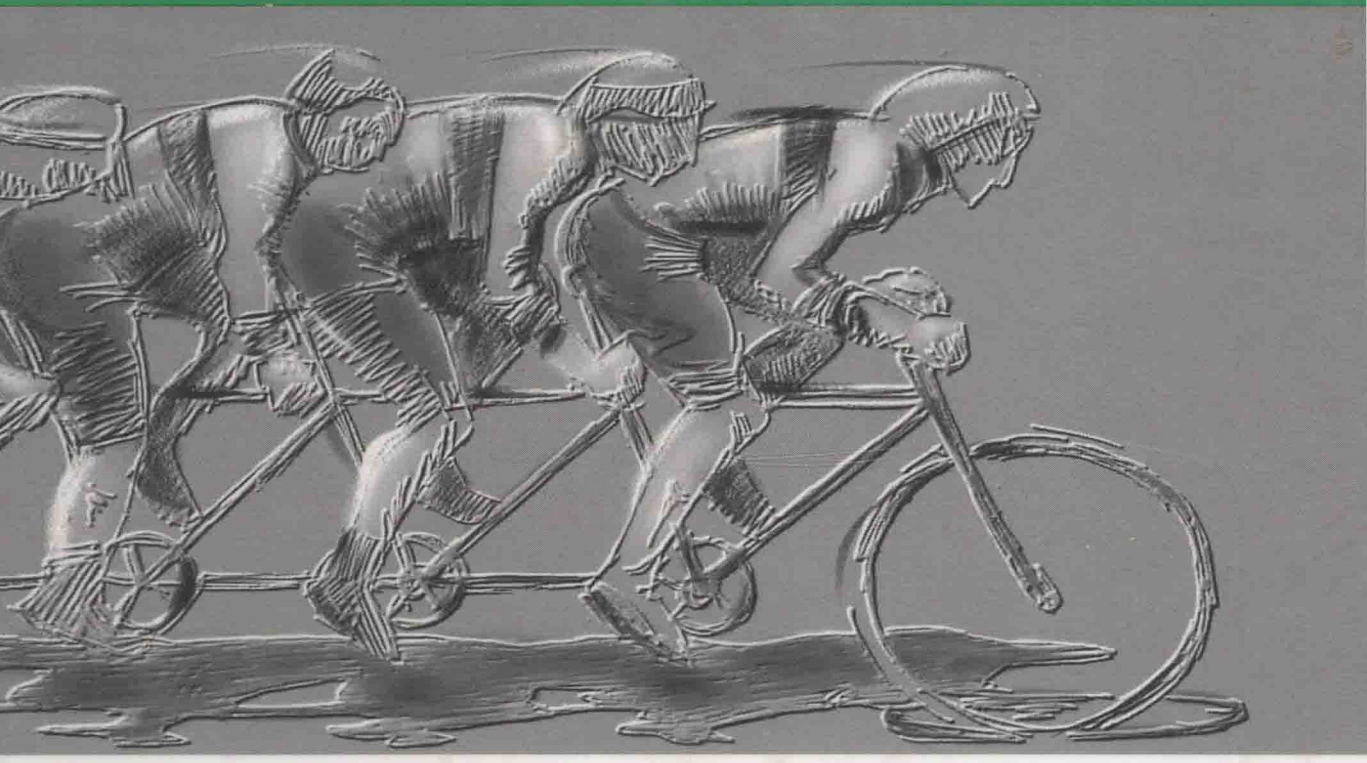
INTRODUCTION TO PARALLEL COMPUTING



Ted G. Lewis
Hesham El-Rewini

INTRODUCTION TO PARALLEL COMPUTING

Ted G. Lewis and Hesham El-Rewini
with In-Kyu Kim

PRENTICE HALL, *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging-in-Publication Data

Lewis, T. G. (Theodore Gyle)
Introduction to parallel computing / by Ted G. Lewis & Hesham El-Rewini.
p. cm.
Includes index.
ISBN 0-13-498924-4
1. Parallel processing (Electronic computers) I. El-Rewini, Hesham. II. Title.
QA76,58.L48 1992
004'.35--dc20 91-39589
CIP

This book was acquired, developed, and produced by Manning Publications Co.

A Simon and Schuster Company
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-498924-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To Molly, Woofer, and Fuji
Ted Lewis

To my mother, who taught me to give!
Hesham El-Rewini

Preface

Parallel computing is one of the most exciting technologies to achieve prominence since the invention of electronic computers in the 1940s. The shift toward multiple processing units per computer takes its place alongside other major shifts in computing since the Dawning Age of the 1950s; the Age of Mainframes in the 1960s; the Age of Minis in the 1970s; the Age of Personal Computers in the 1980s; and finally, the Age of Parallel Computers in the 1990s. With such a profound influence on all of computing, it is important to understand the fundamentals of parallel computing, and how it changes the way we think about computers. Fundamentals is what this book is about.

New technologies go through several distinct stages on their way to mainstream acceptance. First, there is the experimental stage, when researchers study ways to best apply technology to certain problems. Then, there quickly follows an entrepreneurial stage, when products are manufactured and sold to "early adopters." In stage three, a rapid climb up the logistics growth curve is experienced, while the new technology moves into the mainstream. At the time of this writing, we are at the base of this logistics curve. By the mid-1990s, we will reach its mid-point, and by the year 2000, parallel computing will be as mainstream as personal computers were in 1989.

Information dissemination plays an important role in moving a new idea into the mainstream. Research articles in journals are quickly followed by research monographs, of keen interest to a limited number of researchers. But, for the technology to begin its climb along the logistics growth curve, readable explanations of the new ideas must find their way into the hands of non-experts. Hopefully, this book will serve that purpose.

This book is appropriate for a one-semester first course on parallel computing. Each of the 12 chapters covers a different portion of parallel computing in approximately one week of class meetings. Exercises are given at the end of each chapter to stimulate discussion, and a number of references are listed so the student can dig deeper into each subject. In most American universities, this course will precede other, more detailed courses on parallel processing, parallel programming, and applications of parallel computers.

This book is also appropriate for an introduction to parallel computing for practitioners—experts in the field of computing who are curious to learn more about parallelism, but who do not want to plow through large volumes of research to extract the basic ideas. In other words, this book is also written for the knowledgeable computer engineer, scientist, and student who has not participated in the parallel comput-

ing revolution. For this group of people, we have tried to relate the ideas to specific products and techniques that existed in 1990. We have also attempted to survey the emerging techniques and products that we think will become pivotal by the mid-1990s. (As in any crystal ball exercise, we may be totally wrong in our estimation of where parallel computing is headed.)

Our first goal in writing this book has been to provide a survey of all available technologies for parallel computing and to relate them to applications. Therefore, in Chapter 1, we cover the entire field of parallel computing at a very introductory level. In the most succinct manner possible, we describe the different styles or paradigms of parallel computing using several analogies. This establishes a common terminology that we rely on throughout the book.

Parallelism has been touted as a solution to the problem of making computers faster and faster. Indeed, it would seem that there is no limit to the performance of electronic devices as we hear about increases in speed on a daily basis. But, in fact, there are physical limits to switching speeds of components. Sooner or later, these limits will be reached, and parallelism will be the only recourse. However, even before the speed limit is reached, there is an economic motivation to use parallel processing in place of faster and more expensive single-processor systems. Indeed, the economic advantage of low-cost, multiple processing systems was realized in the mid-1980s. Hence, the 1990s were poised for the decade of parallelism simply due to economic forces. But, what do we mean by "performance"? Chapter 2 develops both theoretical and practical measures of performance and shows that there is more to measured performance than raw MFLOPS (million floating point operations per second).

This book is not about parallel processing hardware—the application of parallel hardware to construct parallel computers—but instead, on the essence of parallel computing: hardware plus software plus applications. To gain a solid understanding of the hardware devices used in parallel computing, we provide an overview of parallel processors in Chapter 3. This survey describes the many ways to connect multiple processors together, and shows how each might be used to solve certain problems.

We are especially concerned with the paradigm shift that takes place in software when the target machine is a parallel processor. The traditional programming style and way of thinking about a problem no longer works when the solution can be computed by a collection of parallel parts all cooperating to achieve a final result. Parallel computing is an opportunity to rethink programming from an entirely fresh perspective. Such "new perspectives" are called paradigms. In Chapters 4 through 8, we elaborate on a number of paradigms and show how parallelism is employed in a variety of programming styles. The shared-memory paradigm is perhaps most like traditional programming; the distributed-memory paradigm introduces the concept of message passing; the object-oriented paradigm introduces the concept of a server; the data parallel paradigm relies on the single-instruction-multiple-data (SIMD) model; and the functional dataflow paradigm introduces the concept of a side-effect-free MIMD (multiple-instruction-multiple-data) model.

One of the major differences between serial computing and parallel computing is the notion of optimal resource allocation. In a parallel computer, multiple processors are resources to be carefully allocated and used to advantage. A poorly designed parallel computer application may run slower in parallel than on a serial machine! Thus, careful planning and design are needed to utilize the processors in the most efficient and "optimal" manner. Chapter 9 develops the theory of optimal processor utilization more fully. This is the general problem of scheduling parts of a parallel program onto the processors in such a pattern that the application runs as fast as possible. Chapter 10 continues along this line of reasoning by concentrating on perhaps the greatest opportunity for parallelism—loops. Loops consume much time in serial processing, so they naturally become targets for parallelization.

Facing the prospect of converting billions of lines of serial program code into parallel code has deterred many from using parallel computing technology. Unfortunately, it is not possible to gain the maximal benefit of parallelism without major restructuring of existing software. Chapter 11 describes the technology of program restructuring and parallelization. This difficult problem remains stuck in the world of research, but some progress has been made. Chapter 11 discusses both theory and practical tools for conversion of serial programs into parallel equivalents.

Finally, we would like to combine all that we know about parallelism into one "dream environment" that makes application development simple and easy. Such a parallel programming environment does not exist, but progress is being made along several fronts. Chapter 12 surveys what has been done, what is available, and what is still needed in this vital area.

We have attempted to provide a complete reference to the field of parallel computing without overly burdening the reader with details. Large pieces of parallel program code, benchmarks, and other details are placed in the appendices.

We owe much to the people who have contributed to this work. In particular, to Inkyu Kim, who wrote Chapter 11 and provided many suggestions for other chapters. Youfeng Wu, Shala Arshi, Gary Graunke, Mike Quinn, and others provided sample programs. Norris Smith, Marjan Bace, Mike Evangelist, Bruce Shriver, David Padua, Doug DeGroot, Janice Cuny, John Gustafson, Bruce Boghosian, Shreekant Thakkar, Gregory Riccardi, and Marilynn Livingston all contributed to many improvements in the original manuscript. We are indebted to them for many suggestions and insightful conversations. Responsibility for errors and inconsistencies rests with us.

Ted Lewis, lewis@cs.orst.edu
Hesham El-Rewini, rewini@unocss.unomaha.edu

Contents

Chapter 1

What Is Parallel Computing?

In the first computing wave, scientific and business computers were more or less identical—big and slow. This was the "prehistory of computing," where computing had to be employed at any cost. And, even if early electronic computers were not very fast, they achieved speeds that easily exceeded human computers.

The second and third waves brought on mainframes, minis, and finally micros. This diversity of computing caused a number of niches to develop which broadened and deepened the computer industry. Scientific and business computing went their separate ways, and there seemed to be a computer in just about everyone's price range.

But the original power users who pioneered computing continued to emphasize speed above all else. Single-processor supercomputers achieved unheard of speeds beyond 100 million instructions per second, and pushed hardware technology to the physical limits of chip building. But soon this trend will come to an end, because there are physical and architectural bounds which limit the computational power that can be achieved with a single-processor system.

We are now enjoying the Parallel Wave of computing, where performance is enhanced by using multiple processors. What is parallel computing and how does it work? In this chapter we survey the (somewhat overlapping) paradigms of parallel computing, touching on synchronous versus asynchronous, SIMD, MIMD, SPMD, vector/array, systolic array, and dataflow. In each case, a fundamental parallel computing paradigm is illustrated using a hypothetical bank as an analogy with a parallel computer (tellers are parallel processors and customer transactions are tasks to be performed).

Finally, we show that MIMD is the most general form of parallel computing, but that there are certain performance advantages to each of the other forms. As a consequence, it is important to know when and why to apply each one. SIMD and SPMD forms of parallelism appear to be very good at scientific problem solving where speed can be achieved because the data are regular and the calculations are uniform and repetitious. MIMD appears to be appropriate for medium-grained parallelism where communication overhead is not too great. While some scientific computing may benefit from the MIMD style of parallelism, medium-grain problems are typical of business transaction processing applications.

1.1 THE NEXT REVOLUTION IN COMPUTING

Modern society is particularly susceptible to changes in computer technology. The insurance and banking industries were forever changed by the mainframe data processing computer; science and engineering will never be the same after the impact of minicomputers and workstations; and our personal lives have been enriched by personal computers. Computers, and the advances they make, affect everyone. So revolutions in computing make front-page news.

To determine the next step in computing, we must look to the past, because, like most progress in technology, computing evolves through time in an orderly fashion. Once we understand the pattern, we can extrapolate to surmise what will most likely happen in the future.

1.1.1 Modern Prehistory

The Alwac 3E computer was typical of the state of computing in 1963. It could store 32,000 numbers, each with 32 bits, and read punched paper tape at an unheard of 100 frames per second. The Alwac was less powerful than a 1980 personal computer, but it was operated by one person at a time much like a personal computer.

Early computers such as the Alwac had one major disadvantage compared with personal computers: they were expensive. Because of high hardware costs, the first generation of computers had to be shared by a lot of users to justify their cost. It would take 20 years before these simple, easy-to-use machines were to reappear as inexpensive alternatives to centralized computing.

1.1.2 The Age of Dinosaurs

By 1965 the Alwac "personal computer" and its contemporaries had been pushed aside by the radically new IBM System/360 mainframe. Mainframes boosted IBM from medium-sized office equipment manufacturer to global master of the computing industry, and established large centralized computers as the standard form of computing for decades.

The IBM System/360 was the right computer in the right place at the right time. It was in harmony with the instincts of most programmers of the mid-1960s and early 1970s. It had a real operating system, multiple programming languages, and incredibly large disks capable of 10 megabytes of storage! This was the first wave of modern computing, and the world quickly jumped on the mainframe bandwagon (see Figure 1.1).

The System/360 filled a room with boxes and people to run them. Its transistor circuits were reasonably fast. Power users could order magnetic core memories with up to one megabyte of 32-bit words. This machine was large enough and expensive enough to support many programs in memory at the same time, even though the central processing unit had to switch from program to program as if it were juggling balls in a circus. It quickly became the workhorse of business and made "IBM" a household word for "computer."

1.1.3 The Second Wave

The mainframes of the first wave were firmly established by the late 1960s when advances in semiconductor technology made the solid state memory and integrated circuit feasible. These advances in hardware technology spawned the minicomputer era. They were small, fast, and inexpensive enough to distribute throughout the company. Minicomputers made by DEC, Prime, and Data General led the way in defining a new kind of computing: departmental.

By the 1970s it was clear that there existed two kinds of commercial or business computing: (1) centralized data processing mainframes, and (2) decentralized transaction processing minicomputers. The minis expanded the usefulness of computing into engineering, scientific, and non-data processing applications. Computing got broader, and in the process, touched more people's lives (see Figure 1.2).

1.1.4 The Third Wave

When personal computers were introduced in 1977 by Altair, Processor Technology, North Star, Tandy, Commodore, Apple, and many others, they were largely ignored. But then a strange program called VisiCalc suddenly catapulted the original "personal" computing idea of the 1960s into orbit. By 1981, personal computing was becoming so pervasive that IBM entered the "billion dollar baby" market.

Personal computers enhanced the productivity of individuals, and in turn departments. Because big companies are made up of individuals, the productivity improvements of individuals using stand-alone computers was too compelling to ignore. PCs soon became pervasive (see Figure 1.3).

Networks of powerful personal computers and workstations began to replace mainframes and minis by 1990. The power of the most capable "big" machine could be had in a desktop model for one-tenth the cost. But, these individual desktop computers were soon to be connected into larger complexes of computing by networking.

One of the clear trends in computing is the gradual substitution of networks in place of central computers. These networks connect inexpensive, powerful desktop machines to one another to form unequaled computing power. Networks are an early form of parallel computing.

Clearly, there is a limit to the power of a single computer. Even networking has limitations. Within the decade of the 1990s, the maximum switching speed of silicon will be reached and the rapid progress in achieving greater computing speed will level off.

1.2 THE PARALLEL WAVE

What is the next wave of computing? How can machines continue to operate faster and faster in the face of fundamental limits to the hardware? Parallel computing is the answer to both of these questions. The 1990 decade is to parallel computing what the

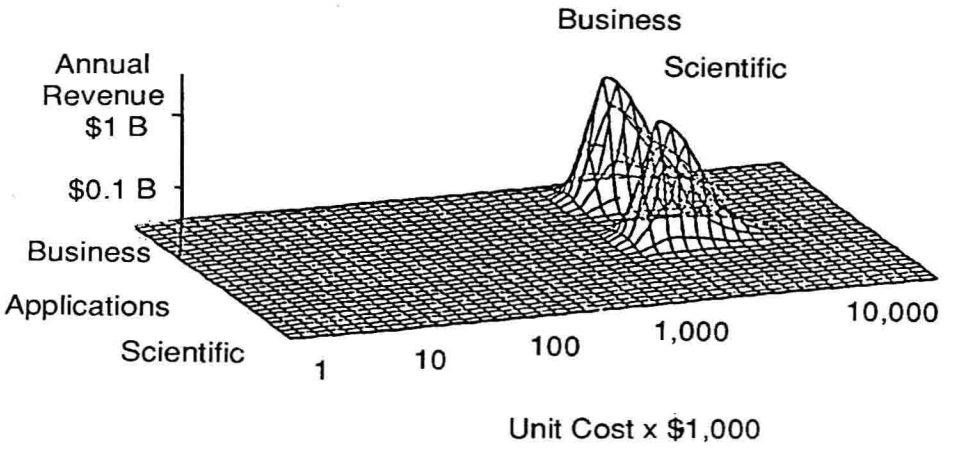


Figure 1.1 Profile of computing circa 1960. Annual revenues vs. unit cost, application type

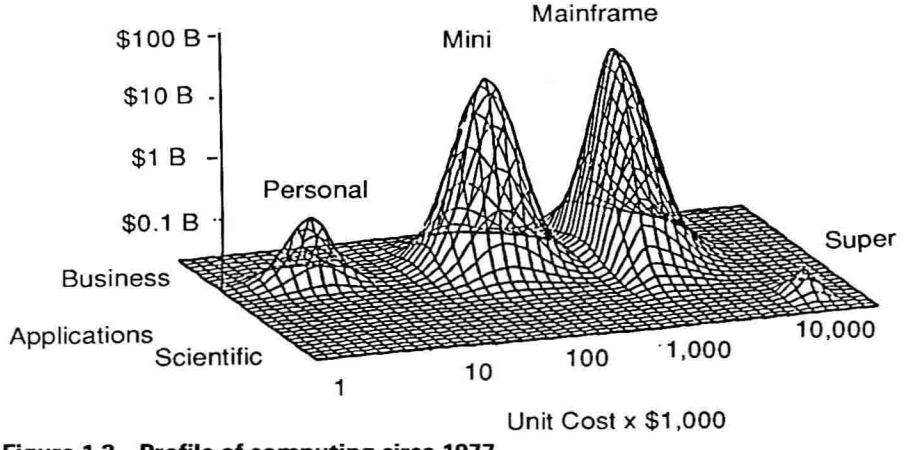


Figure 1.2 Profile of computing circa 1977

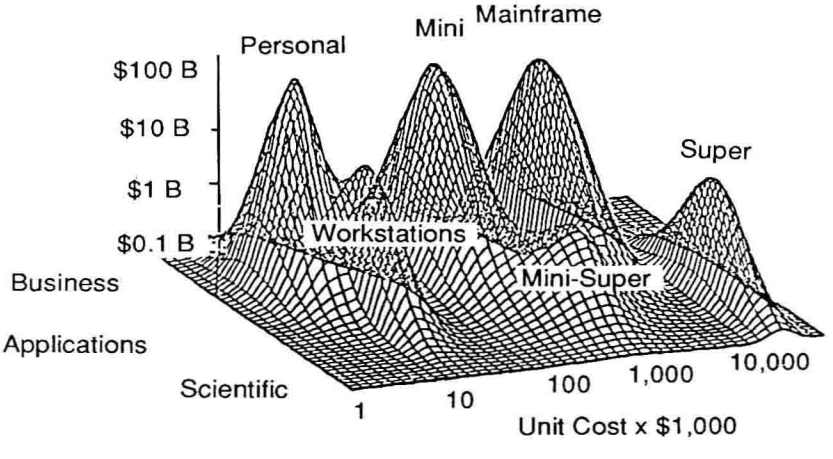


Figure 1.3 Profile of computing circa 1987

1980 decade was to personal computing. But we must first define parallel computing and explore its capabilities.

Parallelism is the process of performing tasks concurrently. To get a better feel for parallel computing, we need to study a number of parallel paradigms. Then we can classify these paradigms in some reasonable way, and use the classification as a basis for an informal theory. This informal theory will be of immense value in determining what approach is best to use in each application we want to put on a parallel computer.

1.2.1 Paradigms of Parallelism

The definition of paradigm has taken on a specific meaning among computer experts. A *paradigm* is a model of the world that is used to formulate a computer solution to some problem. Paradigms give us a context in which to understand and then solve a real-world problem. Because a paradigm is a model, it abstracts the details of the problem from the reality, and in doing so, makes it easier to solve. However, like all abstractions, the model may be inaccurate because it is only an approximation to the real world.

Paradigms are especially important in parallel computing, because we need something to anchor our thoughts when thinking about concurrent processing. In particular, we need a clear way to think about parallelism so we can control the complexity of details that tends to overwhelm parallel programmers.

Some of the more popular paradigms for thinking about parallel computing are shown in Figure 1.4. This is not a complete classification system, but one that includes the major approaches taken by scientists, engineers, and researchers in a variety of fields, who apply parallel computing.

To understand Figure 1.4, we fabricate a purposely contrived example to illustrate the unique and similar features of each paradigm. To the author's knowledge, no real bank operates this way.

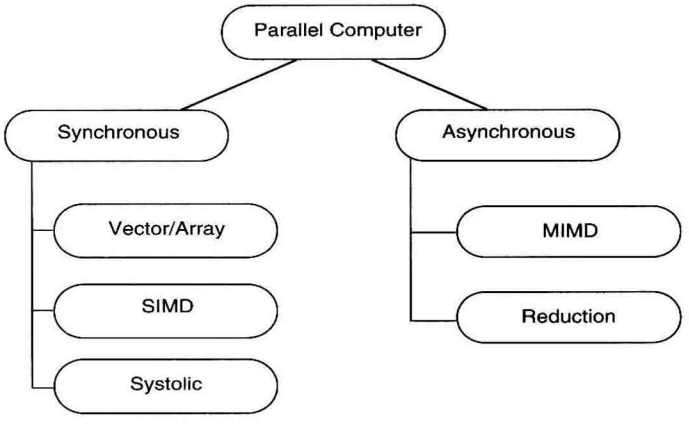


Figure 1.4 Taxonomy of parallel computing paradigms

A banking analogy

We make an analogy between a bank and a parallel computing system (see Figure 1.5). The bank tellers are like computer processors, and customers play the role of program tasks that need to be executed. Having only $n = 1$ teller to help customers is analogous to a uniprocessor system where program tasks are executed serially, one after another. When the number of available tellers is greater than one, the bank operates somewhat like a multiprocessor system.

Suppose there are m customers each waiting to perform a transaction that takes t_i units of time for a single teller to process. If there is only one teller, the customers are served sequentially so the time to serve all customers is $\sum_{i=1}^{m} t_i$.

Now suppose additional tellers return from their break and begin serving more than one customer at a time. If the number of working tellers is equal to the number of customers, $n = m$, then each teller can serve exactly one customer. In this case, the time for serving all customers is $t_{max} = \text{Max}(t_1, t_2, \ldots t_m)$.

Of course this is the best case because all customers are served concurrently. We call this form of parallelism *trivially parallel*, because it is trivial to find and use the parallelism intrinsic to the problem. In general, parallelism is much more difficult to exploit than we have indicated in this simple example.

If the number of tellers happens to be less than the number of customers, n < m, then some customers end up waiting on others to finish. Suppose for simplicity that there are twice as many customers as there are tellers. Each teller would, on the average, serve two customers. In general, it may be necessary for one teller to do more work than the others. Clearly, if all tellers work at the same rate, then it is best to spread the work evenly across all tellers. Sharing the burden equally is a form of *load balancing*—one of the goals of a well-tuned parallel processor system.

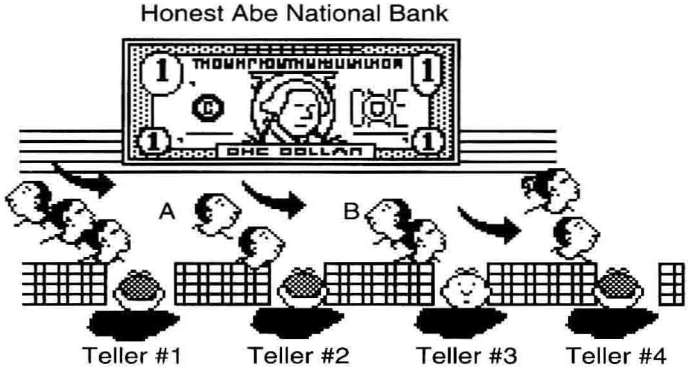


Figure 1.5 The bank analogy: Tellers are like parallel processors, customers are like tasks, transactions are like operations, and accounts are like data in a parallel computer