Multiple Processor Systems for Real-Time Applications

BURT H. LIEBOWITZ

JOHN H. CARSON

Multiple Processor Systems for Real-Time Applications

BURT H. LIEBOWITZ
The Sombers Group, Inc.
Wilmington, Delaware

JOHN H. CARSON
Management Science Department
The George Washington University
Washington, D.C.

PRENTICE-HALL, Inc., Englewood Cliffs, New Jersey 07632

Preface

This book presents a pragmatic approach to the design and construction of a class of computing system we call the "locally distributed multiple processor system (MPS)." These systems consist of closely located independent CPU–memory pairs linked together by a communication subsystem. These pairs, which we call processors, cooperate closely to solve a problem. For brevity, we use the acronym MPS to describe this architecture.

The processors in the MPS are located in the same room or same building and thus can be interconnected by relatively high-bandwidth communication media. This means that processors can communicate quickly and reliably so that a problem can be effectively partitioned between them.

It is our contention that this architecture can, for many applications, provide a more cost-effective solution than the more conventional mainframe or multiprocessor approach. This is especially true for real-time systems which demand high reliability and high throughput. By high reliability we mean systems that cannot be down for the time it takes to fix a failure. Hence all critical components must be redundant. By high throughput we mean a work load that exceeds the capacity of a single processor.

The emergence of the MPS is due, in some degree, to the decreasing cost of minicomputer and microcomputer technology and the increase in the number of programmers and engineers who can work with these devices. This trend will continue and hence we expect more interest in the MPS as time progresses.

In this book we are concerned primarily with the technological aspects of the systems engineering process for the MPS. In being so, we do not mean to minimize the importance of management control. Rather, we are reflecting our observation that the management process for the MPS is mostly similar to that for on-line systems using conventional mainframes. Although this process is not trivial, it has been well covered in the literature. Whatever management tools are considered uniquely important to MPS development are discussed as appropriate throughout the book.

What is unique to the design process is the need for a system architectural role. The MPS is made up of building blocks: computers, peripherals, intercomputer links, software. There is a bewildering array of system structures that can be constructed from these blocks. The search for an optimum or even good solution requires knowledge of these building blocks and a "systems approach" toward putting them together. It is the purpose of this book to provide this knowledge, and buttress it with actual examples of systems that have been built using the MPS architecture.

We assume that the reader has a basic knowledge of computer architecture and is somewhat familiar with computer programming. There is no need to be expert in any one type of computer or language. Although the book is not written as a text, it could be used in an advance undergraduate course or a graduate course in computer systems engineering. However, the book is intended primarily as a reference work for the practicing engineer or system analyst.

The book consists of 12 chapters, which divide into four distinct parts. The first two chapters, forming Part 1, are introductory in nature. Chapter 1 provides a definition and overview of the MPS. Chapter 2 discusses major design issues associated with the MPS. Most issues are stated as an introduction to the remaining parts of the book, which provide more detailed discussion. However, the subject of problem decomposition, which is fundamental to the MPS design process, is covered in depth in this chapter.

Chapters 3 through 6 form Part 2 of the book. These chapters describe the major building-block components of the MPS. We start in Chapter 3 with a description of the computer component; minicomputer and microcomputer technology are emphasized. Both hardware and software aspects of computer technology are discussed. Chapter 4 describes the interprocessor communications system. Physical links and network structures are discussed. The emphasis in this chapter is on techniques for interconnecting locally distributed systems but draws heavily on concepts developed for long-haul networks.

Chapter 5 discusses requirements for the MPS operating system. Several different concepts of operating systems are presented. Issues unique to distributed operating systems are emphasized, including the coordination of processes that are located in separate processors, and communication between processes.

Chapter 6 discusses design issues associated with locally distributed data bases. Such issues as why to distribute data, feasible distribution schemes, and problems associated with having multiple copies of data that require coordination are discussed.

Chapter 7 discusses reliability and recovery within the MPS. Concepts of error detection, fault isolation, and recovery in multiple processor systems are described. Several architectures that facilitate the use of on-line spares for high reliability are introduced. Operating system issues associated with high-reliability systems are discussed.

Part 3 of this book, comprised of Chapters 8, 9, and 10, provides design tools for the MPS. Chapter 8 presents simplified reliability calculations that allow the system architect to determine the level of redundancy needed to meet specified reliability goals. Chapter 9 provides a cookbook approach to the calculation of response times by use of elementary queuing theory. The concept of response time is essential in the design of a real-time system. This chapter provides an easy method for first-level estimations of response times.

Chapter 10 provides a design and implementation methodology, utilizing the technological concepts developed in the early chapters of the book. It ties this knowledge together in a methodology for resolving issues and optimizing solutions for the multiple processor system. Especially difficult issues such as testing and specification of the MPS are emphasized. An example of a design built according to the principles of this chapter is given to illustrate the design concept.

Part 4 provides examples of MPSs that have either been built, are in the design phase, or are anticipated in the future. Chapter 11 provides actual case histories of several systems for use in both government and industry. Chapter 12 concludes the book with a discussion of future technology and where we see the industry going.

This book has evolved from a series of courses given by the authors, both jointly and separately, at The George Washington University in Washington, D.C., and to several industry groups. The original notes for this book were part of a five-day course that has been taught for several years at GWU. These notes have been modified over the years based on feedback from students, some of whom are quite sophisticated practitioners of the art.

The original course taught by us at GWU was broader in scope than this book. It covered all aspects of distributed processing, including both locally distributed systems and geographically distributed systems. We found that at regular intervals, both private and government organizations asked for specialized versions of the course, emphasizing locally distributed systems. It was from such requests that we recognized the subject of locally distributed systems as deserving of a course and a book unto itself. Of special impact were requests by the Naval Air Development Center, Warminister, Pennsylvania, Univac Defense Systems Division, St. Paul, Minnesota, and the IBM Federal System Division, in the Washington, D.C., area. We gratefully acknowledge the contributions of students from these organizations in shaping the structure of the book and its method of presentation.

The authors, collectively, have lectured and taught distributed processing to over 2000 students. The one truth that we see from this exposure is that the use of distributed processing systems, including the locally distributed MPS, is evolving and growing. We believe that the basic concepts set forth in this book will stand the test of time. However, we do recognize that specific technologies described in this book will change and improve, and that the literature of case histories will also expand. We hope that in some way this

book will be a catalyst and help increase the rate of improvements and expansion in this aspect of computer technology.

We gratefully acknowledge the help and contributions of the authors' colleagues at both International Computing Company (now part of Contel Information Systems), with whom Burt Liebowitz shared many years of experience in building systems of the type described in this text, and George Washington University for support in refining this material. Specific thanks go to Ernie Forman, Bill Hardgrave, Lois Graff, and Nozer Singpurwalla for "volunteering" to review portions of this material. Special thanks to Bill Highleyman of The Sombers Group for permission to incorporate into this book portions of his excellent paper on survivable systems. We sincerely thank Bernard Goodwin, our editor at Prentice-Hall, for putting up with us and Raeia Maes, our production editor, for turning our manuscript into a professional text. Our typists—Ann Kovacks, Kati Rogan, Donna Carson, and John Carson—were a great help. We also thank Sherry Clanney for preparing the index. The final draft of this text was prepared with an APPLE II^+ computer using the Screenwriter II word processing package—an excellent combination. Finally, the tolerance of our families, who put up with more than should be asked of anyone during this project, reflects a sacrifice that could come only from love—we thank them.

Burt H. Liebowitz
John H. Carson

Contents

原书缺页

原书缺页

Part 1

Introduction

1 *Introduction and Overview*

There is increasing interest in the application of distributed processing systems to real-time computing problems. These problems are characterized as having critical response time and throughput requirements. Response time may be defined as the time it takes a system to react to a given input, and throughput as the number of transactions per unit time the system can handle. To handle the demands of a real-time situation, a distributed system architecture is usually preferred over a monolithic one. We will define distributed processing as follows [LIEB80a]:

> A distributed system is one in which the computing functions are dispersed among several physical computing elements. These elements may be co-located or geographically separated.

This definition is broad and is not universally accepted. Other definitions exist [DIST79]. The problem of definitions typifies the difficulties of working in a new field. This is especially so in an area such as distributed systems, where perceptions of what a distributed system is vary broadly with users. In the broadest sense, an organization uses distributed computing if it uses more than one processor to handle its computing requirements. This leads to a wide variety of architectures that people call distributed processing systems.

In this book we emphasize one type of distributed system, which we call the locally distributed multiple processor system (MPS). The locally distributed MPS (Figure 1.1) consists of physically independent CPU–memory pairs (called processors) interconnected by an interprocessor communication system (IPCS) composed of physical communication links. The processors are in the same room or building so that the links can be built with high bandwidth and high reliability. The processors can cooperate closely to solve a single problem or a closely related group of problems (i.e., a mission-oriented system). The processors tend to be in the mini- or microcomputer class.

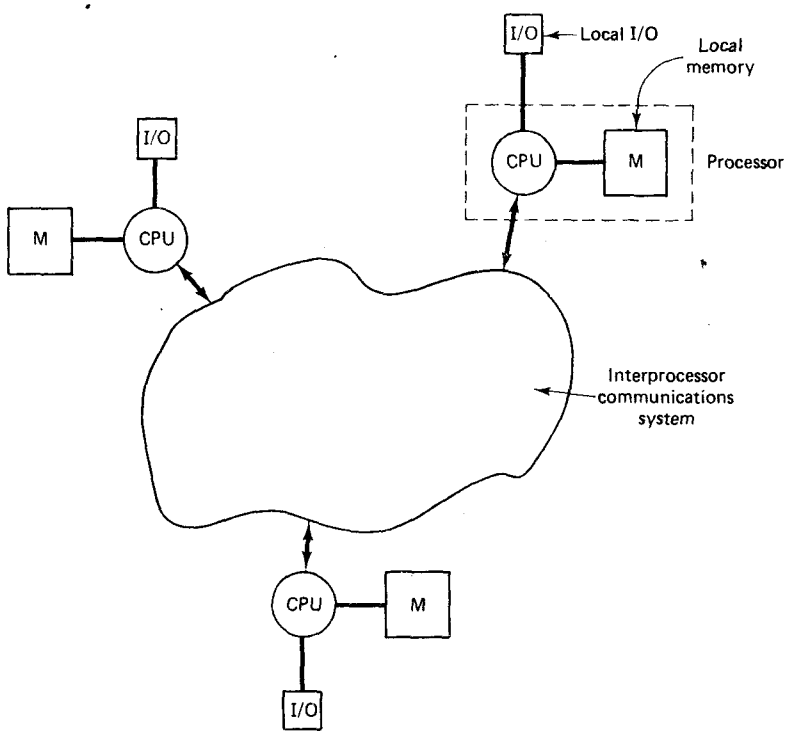

Figure 1.1 Multiple processor system.

The locally distributed MPS has attracted increasing interest because it has potential advantages of cost-effectiveness and performance when applied to real-time computing systems, especially those that require high reliability. This chapter defines the locally distributed MPS, to distinguish it from other distributed processing architectures and mainframe systems. The chapter emphasizes the potential advantages of the MPS in environments that demand high throughput, fast response time, high reliability, and attention to costs.

Some brief case histories will be presented to enhance the claim that the MPS is, in many cases, a cost-effective alternative to mainframe systems. By necessity the case histories are dated—the technology has indeed changed—however, the basic concepts still hold. When these case histories were built or studied, the designers compared the mainframe and the MPS. They found the MPS architecture attractive and the same holds true today. The designer must compare alternatives to strive for the best solution to the real-time application. The purpose of this chapter is to motivate the reader to recognize that the process which evaluates the distributed architecture is worth pursuing. It is not our intention to claim that this process will always lead to the MPS as the best solution—only that it will in many cases.

The locally distributed MPS is one of several identifiable types of

distributed processing systems. To clarify its definition it might prove useful first to describe the matrix of distributed processing system types, of which the locally distributed MPS is one type. We then continue with a more detailed comparison of the locally distributed MPS with other computer system architectures.

1.1 TYPES OF DISTRIBUTED PROCESSING SYSTEMS

The spectrum of distributed processing applications can be depicted as the two-dimensional matrix shown in Table 1.1. One axis represents geographical separation of processors. (We will use the word "processor" to represent a computing element that contains, at a minimum, a CPU–memory pair and hardware for communicating with other processors. A processor can be a complete computing system, including I/O devices.) The second axis represents the degree of cooperation between the processors.

Although both dimensions are continuous, for convenience we can group each dimension in finite categories. On the axis representing geographical separation we recognize two categories: distant and close. By "close" we mean processors that are in the same room, building, or at most in buildings that are located within thousands of meters. This closeness allows us to use low-cost high-bandwidth interprocessor links (50 kbps to 50 Mbps). We can also avoid the use of services provided by common carriers.

For greater distances we must normally use common carrier services. These allow data to be sent almost anywhere, but usually at much lower bandwidths than those available for closely located systems. Common carrier services are provided by twisted-pair wire, coaxial cable, microwave, and satellite. Propagation delay, especially for satellite links, can be significant relative to the length of the message. Elaborate networks may have to be developed to ensure reliable message delivery to large numbers of interconnected computers.

The other axis represents cooperation between processors. By "cooperation" we mean the level of interaction between the processors. We distinguish four such levels in our model.

TABLE 1.1 TYPES OF DISTRIBUTED SYSTEMS

Level of cooperation	Geographical separation	
	Close	Distant
None	Centralized independent processors	Decentralized system
Modest	Local area network	Distributed network
Close	Locally distributed multiple-processor system	Geographically distributed multiple-processor system
Intimate	Multiprocessor	None

At the first level there is no cooperation at all. Each processor is an independent computer. On occasion a magnetic tape might be used to transport data from one computer to another, but no direct level of communication exists. To some, this category should not be considered a distributed system. In fact, many definitions of distributed processing include an explicit statement of communications [DIST79,THUR79a]. However, to many organizations, the issue of whether or not to off-load stand-alone applications from mainframes to minicomputers is significant, so we include this category for completeness.

The next level in the spectrum is that of modest cooperation. Here the processors do communicate; however, each processor is relatively independent and can perform major functions by itself. In general, each processor performs a completely defined user function by itself and communicates summary results to one or more of its neighbors. Communication could be batch oriented. The processors could be connected by very low-speed lines, and in the case of geographically distant systems, by dial-up lines. The failure of any one processor would affect only its area of operations.

At the next level we find closely cooperative systems. Here each processor regularly, perhaps on a transaction-by-transaction basis, communicates with other processors and is dependent on their support. All communications require real-time results and high-bandwidth links may be needed. A failure in any processor could seriously affect the mission of the entire system.

The highest level of cooperation we call intimate. Here the processors share memory, I/O, programs, data, and tasks. These systems are commonly called multiprocessors [LIEB80a,ENSL74]. Their level of cooperation is so high that they can exist only in a local configuration.

From Table 1.1 we see that the matrix leads to seven categories of distributed systems:

1. *Centralized independent processor (CIP):* The CIP is a collection of independent processors located in close proximity. An organization would choose this approach if it determines that it is more cost-effective to off-load applications to an additional computer (typically, a minicomputer) than to augment or upgrade its mainframe system. Multiple computers also allow limited amounts of backup in the case of failure. Management, operational, and maintenance problems could arise because of the use of different computer types and languages.

2. *Decentralized system:* Here computers are geographically distributed throughout an organization to provide computing power at the point of need. The organizational subunits are independent, so the computers need not be linked. This approach would be an alternative to a centralized mainframe

with terminals at the local sites. The major advantage of this approach is that local computers might be more responsive to local needs than would a shared centralized system. The disadvantage of this approach relative to the mainframe is the potential for duplication of effort and resources. These problems can be minimized by the establishment of data processing standards and the application of effective management control.

3. *Local area network:* The processors in the local area network are located in the same room or building. They operate independently in that they have separate missions, but communicate with each other on occasion. The network mechanism could be a message switch, shared cable, or a collection of serial lines. In general, the computers and terminals are under the control of one organization, thereby easing management problems. If the number of processors and terminals is large, the design of the local network can become a complex task. This task can be even more difficult if the computers are heterogeneous. Strict communication standards are required to ensure that the computers can process data passed among or between them, and a network control center may be necessary to monitor the health and performance of the system. However, the local area network may be able to provide significant advantages in reliability, growth modularity, and cost-effectiveness relative to a mainframe.

4. *Distributed network:* The computers are geographically dispersed. They operate independently and communicate infrequently. The computers could be operated by different organizations, and could be quite heterogeneous in nature. A major motivation for this type of network is the desire to share resources [ENSL74]. In this situation, significant management activity is needed to ensure that effective communication can take place between the independent organizations. Technical issues concerning communication, translation of data, network operations, and operational procedures could also arise. Of course, a distributed network could be under the control of one organization, thereby simplifying management and technical problems.

5. *Locally distributed multiple-processor system:* We have previously defined the locally distributed MPS. The system architecture of the multiple-processor system (MPS) was shown in Figure 1.1. Individual CPU–memory pairs are identified as processors. Each processor is linked to an interprocessor communications system (IPCS). A processor may contain its own peripherals or use the peripherals attached to another processor. No single processor has a complete view of all system resources; the operating system is distributed and overall control is cooperative. The processors are located in the same room or building and cooperate to handle a specific mission. The