tutorial

DISTRIBUTED PROCESSING

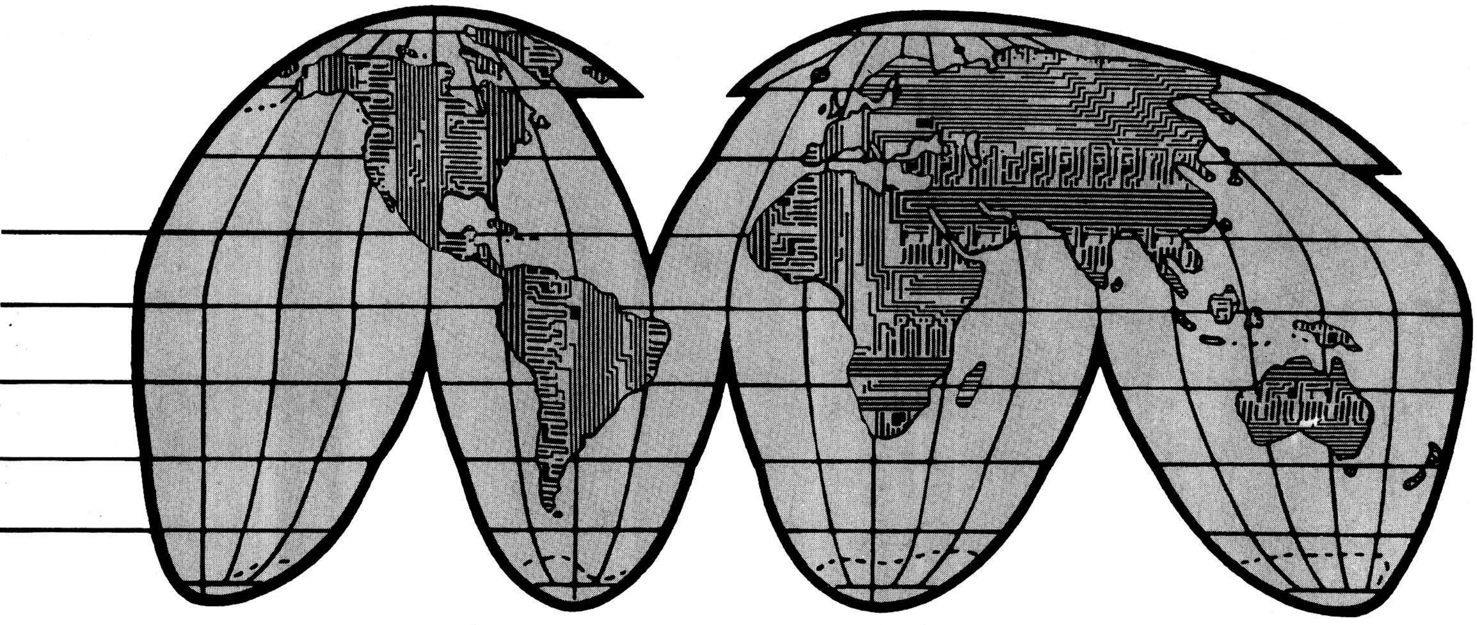
Second Edition

Burt H. Liebowitz
John H. Carson

IEEE Catalog No. EHO 127-1
Library of Congress No. 78-67757

IEEE COMPUTER SOCIETY

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS tutorial



DISTRIBUTED PROCESSING

Second Edition

Burt H. Liebowitz
International Computing Company
Bethesda, Maryland

John H. Carson
RLG Associates, Inc.
Reston, Virginia

IEEE Catalog No. EHO 127-1
Library of Congress No. 78-67757

IEEE COMPUTER SOCIETY

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS

Additional copies available from:

IEEE Computer Society	IEEE Service Center
5855 Naples Plaza, Suite 301	445 Hoes Lane
Long Beach, CA 90803	Piscataway, NJ 08854

The Institute of Electrical Electronics Engineers, Inc., New York, N.Y., Printed in USA

Table of Contents

1. Introduction

The past several years have seen an increasing interest in the application of distributed computing systems.

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

Distributed systems take many forms covering a diverse range of system architectures. The very term *distributed processing* may invoke radically different images of technology and problem solutions depending upon the user.

To some, a distributed system is a collection of multiple computers or processing elements working closely together in the solution of a single problem. An example might be an on-line banking system comprised of several minicomputers linked together by shared memory, communication lines, or common busses. Each minicomputer processes a subset of banking transactions and updates a portion of a common data base. Users of such systems are concerned with issues of hardware and software design, reliability, multicomputer operating systems, and how to optimally decompose programs and data bases.

To other users, a distributed system is a set of intelligent terminals located at the point of use to give local organizational elements more responsive computer support. These terminals perform most of the computing functions for the local group. When necessary the terminals communicate with remote host computers and each other for enhanced support. These users are concerned with optimizing their business functions, determining the best products available for building the system, reducing communication costs, and with what can be severe management control problems associated with decentralized computing facilities.

To yet another set of users, distributed systems may mean collections of geographically dispersed, independent computing centers linked together to allow the sharing of software and hardware resources. These systems are commonly referred to as computer networks. An example is a network developed by a collection of educational institutes each with its own computer facility. The network allows a user at one installation to access a program or a file resident at another facility. The network may also allow load leveling and backup of failed facilities. Designers of such networks are concerned with issues of security, maintaining local independence, appropriate charging algorithms, network management, optimum communications, user survivability, and linking together heterogeneous computers.

To some, distributed processing means the use of small computers to off-load supporting functions from large mainframes to extend the life of the mainframe. Front-end computers have been used for years to handle data communications. Recently, back-end computers have been developed to support data-base management functions. Designers of such systems are concerned with methods for coupling small computers to large computers, operating system interactions, and the development of efficient software for the support systems.

There are some users whose distributed systems are composites of the ones mentioned above.

The common thread linking the different types of distributed systems is the use of multiple cooperating computing elements to perform jobs that were heretofore performed on large computers or not performed at all. Distributed systems have evolved as alternatives to large centralized facilities because, in many circumstances, centralized facilities have not provided optimum or effective problem solutions.

The large computer might prove costly for applications which do not need extensive computing capability. An individual user may get lost in the shuffle of a large computing center and receive poor service. The complex hardware and operating system software required for a large facility may fail more often than is acceptable for some time-critical jobs. Redundancy may be too expensive to consider as an approach to improving reliability. Response times for real-time jobs may become unpredictable and uncontrollable due to the wide variation in load handled by a multiuser installation. Telecommunication costs may become excessive in an organization with geographically dispersed users. A local facility may not have the resources required for a specific job.

A network of cooperating processors can, under the proper circumstances, alleviate the above problems. In the past, a major deterent to the distributed approach has been cost. The technology of computing has been such that a collection of small computers equivalent in capability to a large computer cost more than the large computer. However, the situation has changed in the past few years. Low-cost computing power in the form of minicomputers and microcomputers presents an attractive alternative to large computers for many computing tasks. This trend will increase in the future. It can even be argued that economy of scale now favors the small computer. The cost per unit of computing power should be less in an assembly line of a large number of small computers than in an assembly line of a small number of large computers.

A further potential of cost reduction exists in systems which are geographically dispersed. Local processing capability can be used to reduce the volume of data transmission between sites. The savings in communication costs for lines and modems could more than pay for the cost of local computers or intelligent terminals.

With cost no longer a barrier the inherent advantages of distributed computing can be applied to a wide array of computing problems. These advantages include the ones discussed below.

(1) Reliability—redundancy can be achieved in a relatively inexpensive manner in a distributed system. The entire system does not have to be replicated as is the

case with a single computer. Only an incremental number of processors must be added to insure the required degree of availability. Simpler, and hence, more reliable software structures may be achievable in a collection of small computers.

(2) Responsiveness—the distributed system can be more responsive because direct access to a computer can be provided to smaller user communities. This responsiveness can take the form of reduced turn-around time in a batch environment and faster response times in a real-time environment.

(3) Incremental growth—a distributed system in danger of overload can be expanded incrementally at low cost by the addition of more processors. A centralized system in danger of overload can be preserved by offloading functions onto small processors.

(4) Correspondence to organizational patterns—many organizations are decentralized in nature. Centralized computing facilities may present an unnatural restriction to the efficient operation of the organization. Dispersed computing alleviates this restriction by allowing a decentralized group access to its own computer which can be optimized to the group's needs. A network arrangement will allow central management access to summary information for all groups.

(5) Resource sharing—a distributed network of computing systems allows users at one location to take advantage of resources that are available at other locations. These resources could consist of programs, data bases, and computational power. Resource-sharing networks allow load balancing, backup, and reduced duplication of effort.

Of course distributed processing can create its own problems. Multiple-computer systems can be quite complex to build and validate. Decentralization of computing capability could lead to uncontrolled growth, local empires, and diffusion of an organization's computing personnel. Poor design can lead to loss of reliability and responsiveness. Some types of problems are most amenable to solution on large number-crunching systems. There is not yet available in the marketplace a wide array of software capable of supporting distributed systems.

The advantages and disadvantages of centralized versus distributed processing must be carefully weighed before the decision is made. There is, however, reason to believe that the trend to distributed processing will grow, particularly as the cost of computing power drops.

Purpose and scope of this tutorial

The purpose of this tutorial is to provide an introduction to distributed processing concepts. These concepts can be used to determine the relevance of a distributed processing approach to the reader's computing needs.

Those considering the use of distributed systems will be confronted with managerial and technical issues that may be new and unfamiliar. The managerial issues relate to a struggle that has persisted since the infancy of computing, that is, whether to centralize or decentralize the control of data processing. The advent of low-cost distributed systems has made decentralization more feasible and thus has rekindled the debate.

The technological issues involve communications, intercomputer coupling, executive software structures, system architecture, component selection, and allocation of functions and data files to multiple processors.

This tutorial will emphasize the technological issues. This emphasis reflects the greater availability of literature dealing with technical issues and the authors' opinion that an understanding of technical issues is a prerequisite to successful management decision making. It also reflects the observation that management issues often are resolved more for political, personnel, or historical reasons than by the application of well-established analytical principles. For those interested in management issues, References 1 and 2 are recommended for further reading.

The broad scope of distributed processing is reflected in the organization of this tutorial. There are eight chapters, including this introduction. Chapters 2 through 4 discuss basic building blocks of distributed systems, computers, communications, and terminals. The chapter on computers emphasizes minicomputers, microcomputers and methods of interconnecting colocated computers. The chapter on communications discusses issues associated with the interconnection of computers using serial communication lines. The chapter on terminals emphasizes intelligent terminals and point-of-use applications, such as data entry, point-of-sale, and local business processing.

The next three chapters discuss major categories of distributed systems, distinguished by the degree of coupling between the processing elements and the purpose for which the system was developed. In this tutorial we define three categories of coupling: (1) loose, in which the processors are connected by serial lines of relatively low bandwidth ($\leqslant$50K b/s); (2) moderate, in which the processors are connected by higher speed serial lines, parallel busses, or shared peripherals; and (3) tight, in which the processors are connected by shared memory.

Chapter 5 describes resource-sharing networks. These networks connect geographically dispersed general purpose computing centers in a loosely coupled manner. The network allows users at one location to take advantage of resources available at other locations. Significant software and management issues arise if the computers are heterogeneous and operate under differing management structures.

Chapter 6 discusses multiple-processor systems (MPS). The MPS consists of moderately coupled processors, is usually comprised of homogeneous processors, and is dedicated to a limited and well-defined set of jobs. The processors can either be colocated or geographically dispersed. MPSs comprised of minicomputers have the potential of outperforming more expensive, large general purpose computers for a diverse category of dedicated applications. Some MPSs consist of large computers supported by small computers to offload critical functions. One example discussed in Chapter 6 is the back-end processor for data-base management.

Chapter 7 discusses multiprocessors which are tightly coupled multiple-processor systems. The multiprocessor provides the potential for high reliability, high performance, geographically centralized computing. Tight coupling provides significant challenges in software and processor coordination.

Chapter 8 describes the problems and potential benefits associated with distributed data bases. A data base is comprised of two or more processing elements each of which has a set of files attached. These files can be divided on a geographical or functional basis or for reasons of redundancy. Distributed data bases may arise in applications that exist in all the major categories of distributed systems discussed in this tutorial.

Each chapter will contain an introductory discussion of the subject material and a set of reprints of current articles containing more detailed information on the subject matter. Where appropriate, references are provided for material not reprinted within this tutorial.

Preview

The three papers in this section present general discussions of distributed processing systems. The paper by Booth describes several major categories of distributed systems. Scrupski provides some specific examples of distributed systems and provides an overview of manufacturer offerings in the distributed processing marketplace. Joseph discusses types, functions, architectures, and trends in distributed systems. ■

Distributed information systems

by GRAYCE M. BOOTH
Honeywell Information Systems
Phoenix, Arizona

ABSTRACT

Distributed information systems represent an increasingly important trend to computer users. Distributed processing is a technique for implementing a single logical set of processing functions across a number of physical devices, so that each performs some part of the total processing required. Distributed processing is often accompanied by the formation of a distributed database. A distributed database exists when the data elements stored at multiple locations are interrelated, or if a process (program execution) at one location requires access to data stored at another location. Examples of how these techniques are being used are provided, with comments on the advantages and disadvantages of the distribution of processing and databases in the current state-of-the-art.

INTRODUCTION

This paper discusses distributed information systems—what they are, and why they will become increasingly important to computer users. Distributed systems—also called distributed database systems—are in the early stages of development as yet, so much of this discussion will be theoretical. Whenever possible, however, practical examples will be supplied from systems now being planned, implemented, or in use.

DEFINITIONS

Information network

A distributed database system always exists within an information network environment. An information network is a combination of information processing facilities, data communications facilities, and endpoint facilities. Together these support the movement and processing of files, programs, data, messages, and transactions. An example of a complex information network is shown in Figure 1.

Data communications facilities include the transmission ("telephone") lines, coupling devices such as modems, and concentration devices such as multiplexors and terminal concentrators. The endpoint facilities include terminal devices, and also satellite processors used both as endpoints and for localized information processing.

Distributed processing

Distributed processing exists when a single logical set of processing functions is implemented across a number of physical devices, so that each performs some part of the total processing required. Two types of distributed processing have been defined, horizontal and hierarchical (or vertical).

In horizontal distribution all devices cooperate at an equal level, logically, to perform a set of tasks. There is no hierarchical relationship among the devices.

In vertical distribution the interconnected devices form a hierarchy, sharing tasks in a structured way with each component to some degree controlled by the higher-level member(s) of the hierarchy.

In a complex system, such as the Figure 1 example, both types of distributed processing may be present. Horizontal distribution is shown in the example be-

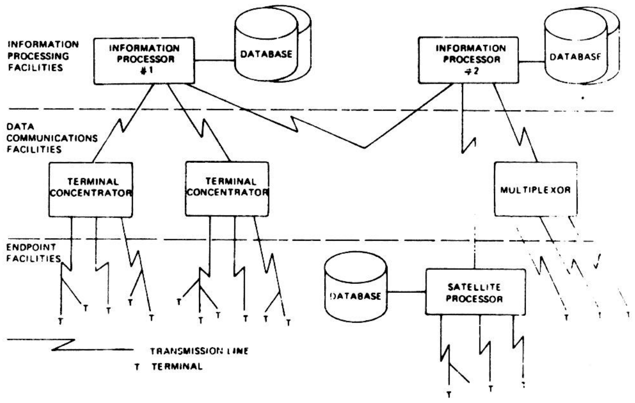


Figure 1—Information network

tween the two information processors, and hierarchical distribution is shown between Information Processor #2 and its satellite processor.

Distributed database

Whenever multiple processing devices are configured in an information network the possibility of a distributed database exists. In the broadest sense the data stored at all locations could be considered to form a distributed database. However, for practical purposes a distributed database exists only when the data elements at multiple locations are interrelated, or if a process (program execution) at one location requires access to data stored at another location.

A distributed database may consist of a single copy of a set of information, divided into increments which reside at multiple locations. This form is called a partitioned database.

A distributed database may also consist of a set of information, all or selected parts of which is copied at two or more locations. This form is called a replicated database. A single distributed system may make use of both of these forms of database.

EXAMPLE SYSTEMS

This section presents a small sample of distributed database systems which are now in the planning and/or implementation process.

Example #1

The first example, shown in Figure 2, is a horizontally distributed system. Two large-scale information processors, located remotely from each other, are linked using communications facilities.

The two computers communicate by exchanging files. A file may be a source-level program; in the company shown most program development takes place in City A, and copies of new programs are transmitted to City B for use there.

A file may also represent a job (an executable program and its accompanying input/output data), which is transmitted from one information processor to the other for load leveling purposes.

This example represents a "loosely-coupled" distributed system. The two information processors are largely independent, and exchange data only occasionally. A distributed database is not required to support this mode of operation.

Example #2

A manufacturing control system (Figure 3) is the second example. In this system the large-scale information processor is not directly involved in real-time process control. It maintains the master database, which is used for overall scheduling and control of the manufacturing process.

At the next level in the hierarchy there are several satellite processors, implemented using minicomputers. Each satellite handles a part of the manufacturing process, and maintains its local database, which is a subset of the central database and contains only the data applicable to the local task. This is an example of a replicated database.

The next lower level of the hierarchy consists of terminal concentrators. These minicomputers link the satellites to the lowest-level devices, concentrating data from many devices onto a few communications links and/or direct cables to each satellite processor.

The lowest-level minis (in some cases microcomputers are used) monitor and control the factory equip-

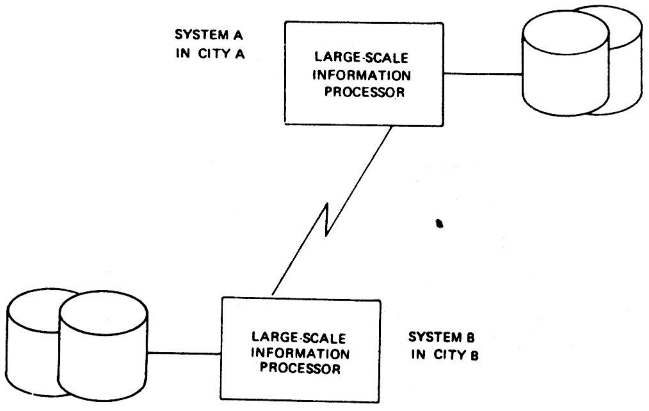

SYSTEM A
IN CITY A
LARGE-SCALE INFORMATION PROCESSOR
LARGE-SCALE INFORMATION PROCESSOR
SYSTEM B
IN CITY B

Figure 2—Horizontally distributed system

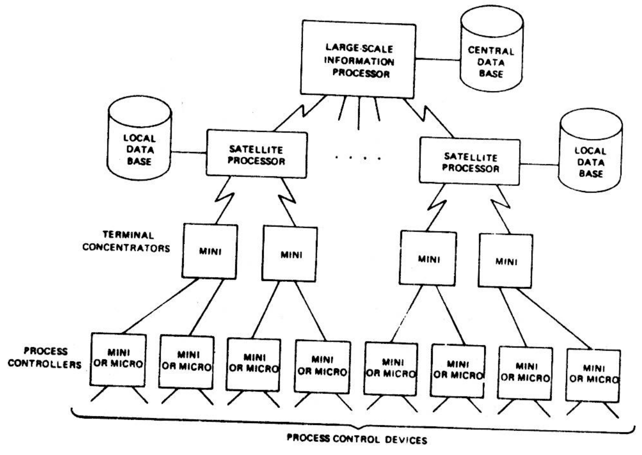


Figure 3—Hierarchically distributed system

ment. Process status information is sent up to the satellite processors, while control commands are received from the satellites to guide the manufacturing process.

Distributed system architecture

The two major forms of distributed system architecture are horizontal distribution and vertical (or hierarchical) distribution. In a very complex system both forms may be used.

HORIZONTALLY DISTRIBUTED PROCESSING

The horizontal distribution of processing functions involves the interconnection of two or more components which are logically equal. Note the term "logically"; the components may be physically unlike and of different capacity and power. The important aspect is their logical relationship within the distributed system.

In a horizontally distributed system multiple information processors most often cooperate to exchange jobs and/or transactions so that the total workload is suitably distributed. An example is shown in Figure 4.

The example shows three information processors, geographically distributed but interconnected via data communications facilities.

In a configuration of this type each information processor will normally handle jobs/transactions which originate locally. The interconnections can be used for load leveling among the three information processors. Candidates for load leveling could range all the way from compilations to complex jobs which require that the referenced files be transferred between cities with the jobs.

This illustrates one trend among current users; the interconnection of hitherto independent computers to gain the advantages of load leveling and common access to all files (which may form a distributed database).

HIERARCHICALLY DISTRIBUTED PROCESSING

Probably the most basic rule in designing hierarchically distributed systems is to spread the processing load up and down the hierarchy by locating functions where they can be performed with the best cost/performance ratio. Functions which are required repeatedly and with quick response are moved out (or down) into the hierarchy as far as feasible. Other functions less often executed and/or with less stringent response requirements tend to move upward toward the center of the hierarchy.

As functions are relocated within the hierarchy the data which supports these functions must move with them, leading to the formation of distributed databases.

An example three-level hierarchy is shown in Figure 5. At the top of the hierarchy is a central computing complex, consisting of one or more large-scale information processors. At the next level there are several satellite processors, placed in factory locations, warehouses, or bank branches, depending on the type of application. At the lowest level there is a potentially large number of controllers, to which the terminals and/or other input/output devices (such as process-control equipment) are attached.

It is easy to visualize hierarchical systems with fewer levels or with more levels, although three is probably most typical today. A two-level hierarchy might omit the terminal controllers and handle the terminal/device interfaces directly from the satellite processors. A four-level hierarchy might separate each satellite processor into a local information processor and a terminal concentrator.

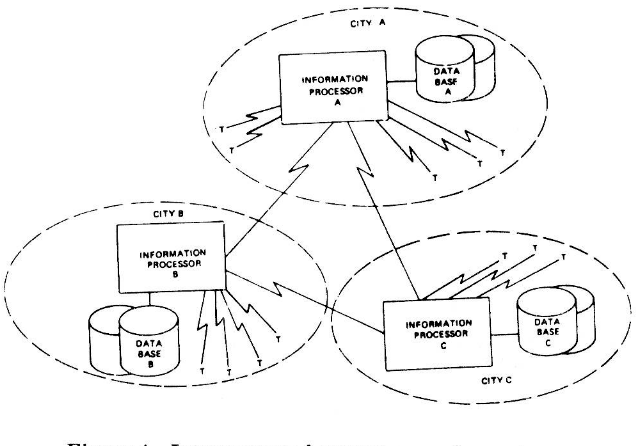


Figure 4—Interconnected computer service centers

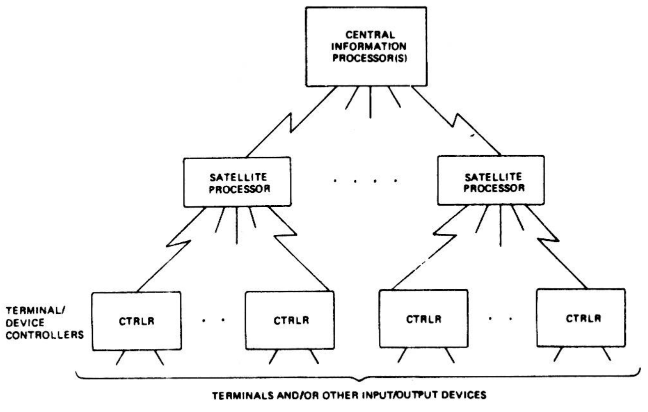

CENTRAL INFORMATION PROCESSOR(S)
SATELLITE PROCESSOR
SATELLITE PROCESSOR
TERMINAL/ DEVICE CONTROLLERS
CTRLR
TERMINALS AND/OR OTHER INPUT/OUTPUT DEVICES

Figure 5—Computer hierarchy

Distributed database architecture

In many distributed processing systems there is a strong probability that a distributed database will be required. Distributed databases can be separated into two categories:

- Partitioned databases
- Replicated databases

PARTITIONED DATABASES

The first category of distributed database exists when a conceptual database is separated into sections and spread across multiple computers. The term "conceptual" is used because in general a single database is not created and then partitioned; instead, the database is designed as a logical entity but actually only created in the form of its partitions. The separate sections, because of their interrelationship, logically form a single database.

Database access partitioning

Database partitioning often follows the natural distribution of database access requirements.

As an example, consider a wholesale company with two information processors. Their total database will probably be split into two partitions, and one attached to each computer. This separates both the processing load and the database accesses between the two information processors. If the company operates nationwide within the United States, the information processors might be located as shown in Figure 6.

In the example system it is logical to locate database partitions heavily accessed by east coast users at the eastern information processor and locate those most accessed by west coast users at the western information processor. Accesses from intermediate locations must be equitably distributed between the two computers.

An important reason for geographical grouping of data is the cost of transmitting data to/from remote locations. In general, the shorter the transmission distance the lower the cost.

In the Figure 6 example provision must be made for at least some use from outside of the area where the database partition is located. The east coast database partition will no doubt sometimes be accessed from west coast locations, and vice versa. In a distributed database system such accesses must be allowed. However, the great majority of the accesses to each section of the database will originate locally. If this is not the case, the database has not been partitioned correctly.

Vertical hierarchy partitioning

Another example of the use of a partitioned database is shown in Figure 7. In this case the database is partitioned over a hierarchy, rather than across a horizontally distributed system.

This example shows a processing hierarchy which corresponds to the organizational hierarchy—a fairly typical case. The large-scale information processor handles corporate-level processing, with a corporate database attached. Each division has its own satellite processor—two of these are shown. Each divisional satellite processor has associated with it a satellite database. This example assumes that there is no duplication of information among the databases shown. In effect, therefore, the data elements at all locations form a corporate database, which has been partitioned across the hierarchy.

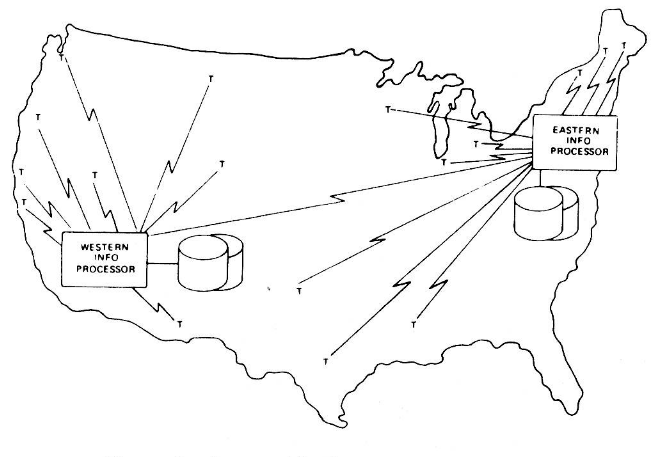
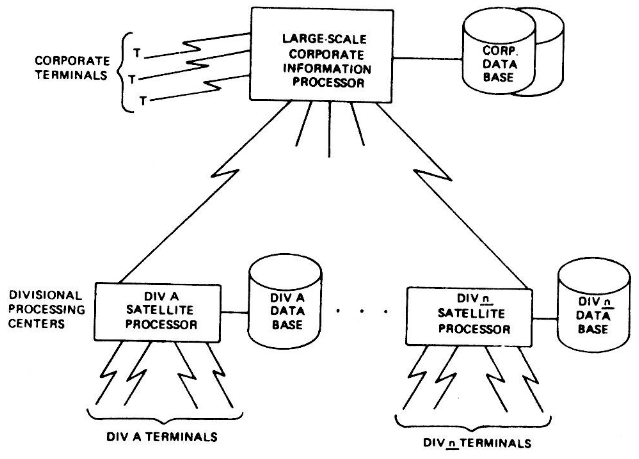


Figure 6—Geographically partitioned database


CORPORATE TERMINALS
T
LARGE-SCALE CORPORATE INFORMATION PROCESSOR
CORP. DATA BASE
DIVISIONAL PROCESSING CENTERS
DIV A SATELLITE PROCESSOR
DIV A DATA BASE
DIV n SATELLITE PROCESSOR
DIV n DATA BASE
DIV A TERMINALS
DIV n TERMINALS
Figure 7—Vertically partitioned database

Information is exchanged in both directions across the hierarchy. Corporate-level reports require information from the divisional database for consolidation with corporate data. Data also travels down the hierarchy, as goals and budgets are established at the corporate level and are passed down to the divisional level.

REPLICATED DATABASES

The replication of all or part of a database at two or more locations is another way to create a distributed database.

An example of the use of a replicated database is shown in the Figure 8 illustration of a complex banking system formed of the interconnection of two hierarchies.

In this system the central information processors maintain the master database of customer accounts, with the total set of customer records partitioned between the two computers. Each satellite processor maintains a database containing the accounts for its local customers. Each local database is created by copying the necessary information from one of the partitions of the central database.

The local databases are essentially work files, refreshed each night from the master database. On-line activity during the day is posted to the local databases; these are used for withdrawal authorization and similar functions. At night all activity is batched and used to update the master database, from which new local databases are then created, and the cycle repeats.

Distributed database system use

The two forms of distributed system architecture—horizontal distribution and hierarchical distribution—and the two categories of distributed database—replicated database and partitioned database—can be combined in a variety of ways.

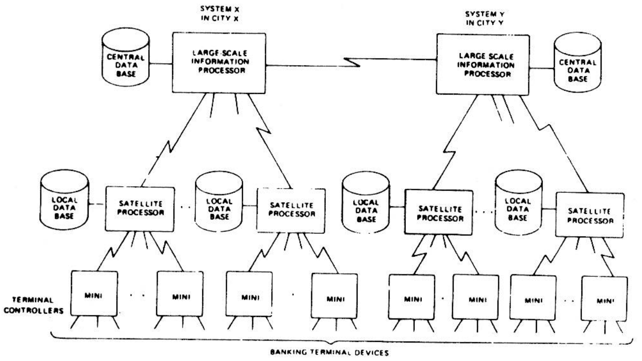


Figure 8—Complex banking system

DYNAMIC LOAD LEVELING

A distributed system can be used to dynamically spread the total processing/database access load across the available computers. In general, this mode of use is associated with horizontally distributed systems.

In a relatively straightforward situation an organization might have three identical information processors, as shown earlier in the Figure 4 example.

If all three information processors have the same capabilities, then jobs/transactions can be moved freely among them. It may be desirable to execute a specific job in a particular computer because it requires access to a database associated with that computer. On other occasions it may be desirable to move jobs, possibly accompanied by their data, from an overloaded information processor to one which has available capacity. The choices for load leveling can be summarized as follows:

- Move the process to the data
- Move the data to the process

In a batch-oriented environment the tendency is to transmit a job, or a subdivision of a job, to the information processor where the database is located. Because a batch job often does extensive processing against the database, and database size is typically large, it is more economical to transmit the job and its input/output data than to transmit the database.

In the Figure 4 example, if a job is entered at City A but requires access to Database B, then the job will normally be transmitted to Information Processor B, executed there, and output returned to City A if necessary.

In inquiry/response applications, in contrast to batch processing, usually only a small part of the database is accessed. It is therefore feasible to consider transmitting the data to the process and returning only updates (if any) to the computer where the data is stored. In this case the process, as specified by its process-state and program, may be considerably larger than the data involved.

STATIC LOAD LEVELING

In the second method of organizing a distributed database system the total load is leveled by preassigning functions and database segments to processors statically.

This method is most often used in a hierarchically distributed system, which may be formed of unlike computers; i.e., large-scale information processors, minicomputers, terminal/device controllers, and microcomputers.

In this situation control of a distributed database is somewhat less complex than in the preceding case. The functions which must be supplied are: (1) provide remote as well as local access to parts of the distributed database; and (2) move parts of the database up and down the hierarchy.

Referring to the complex banking system shown earlier in Figure 8, in this system enough information is included in each satellite processor's database to handle approximately 80 percent of the transactions which originate at the local terminals. Deposits and withdrawals can be handled by the satellite, provided that the customer's home branch is within the satellite processor's local area.

Conditions which the satellite cannot handle include withdrawals for a customer whose account is in a remote branch, and complex transactions such as loan applications and credit card applications. For this particular bank 80 percent of all transactions can be handled by application programs resident in the satellite processors, working with their local databases. The complex transactions are sent up to the large-scale information processors; these have a more complete set of application programs and a much larger database.

Perhaps the most complicated case to handle occurs when a customer enters a branch remote from the branch at which his account is maintained, and requests a withdrawal. Approximately 3 percent to 5 percent of all transactions are of this type.

In this case the withdrawal transaction is passed on by the satellite processor where entered, and travels up the hierarchy to the nearest central information processor. From there it is passed, via the other information processor if necessary, to the satellite which handles the customer's home branch. The withdrawal is processed by that satellite, using its local database, and the response is returned to the satellite where the transaction entered.

The banking system is a good illustration of the principles of distributing functions—application processing—and database information as close as feasible to the point of transaction origination. It also illustrates the principles of centralizing the handling of small numbers of complex transactions which cannot economically be handled by each of the satellite processors.

SUMMARY

Today the computer industry and its users are in the early stages of a developing trend toward the use of distributed database systems. Not all information system users will follow this trend; many will continue to be served satisfactorily by more conventional centralized system structures.

However, advances in both hardware and software technology will make distributed architectures attractive to an increasing number of users. Information networks will become more and more common in the next five to ten years. Accompanying these information networks and distributed system architectures will be an increased use of distributed databases.

These distributed database systems, although complex, will provide a degree of efficiency and cost/effectiveness often impossible to achieve in centralized system architecture.

ACKNOWLEDGEMENTS

I am indebted to many of my colleagues at Honeywell Information Systems, especially the Distributed Processing Technology Group under M. Canepa, for their significant contributions to this work.

Reprinted from *Electronics*, May 27, 1976; Copyright © McGraw-Hill Inc., 1976.

Distributed processing grows as its hardware and software develop

Large-scale integration and better interconnections of computers benefit both lower-level processing and organized data networks

by Stephen E. Scrupski, *Computers Editor*

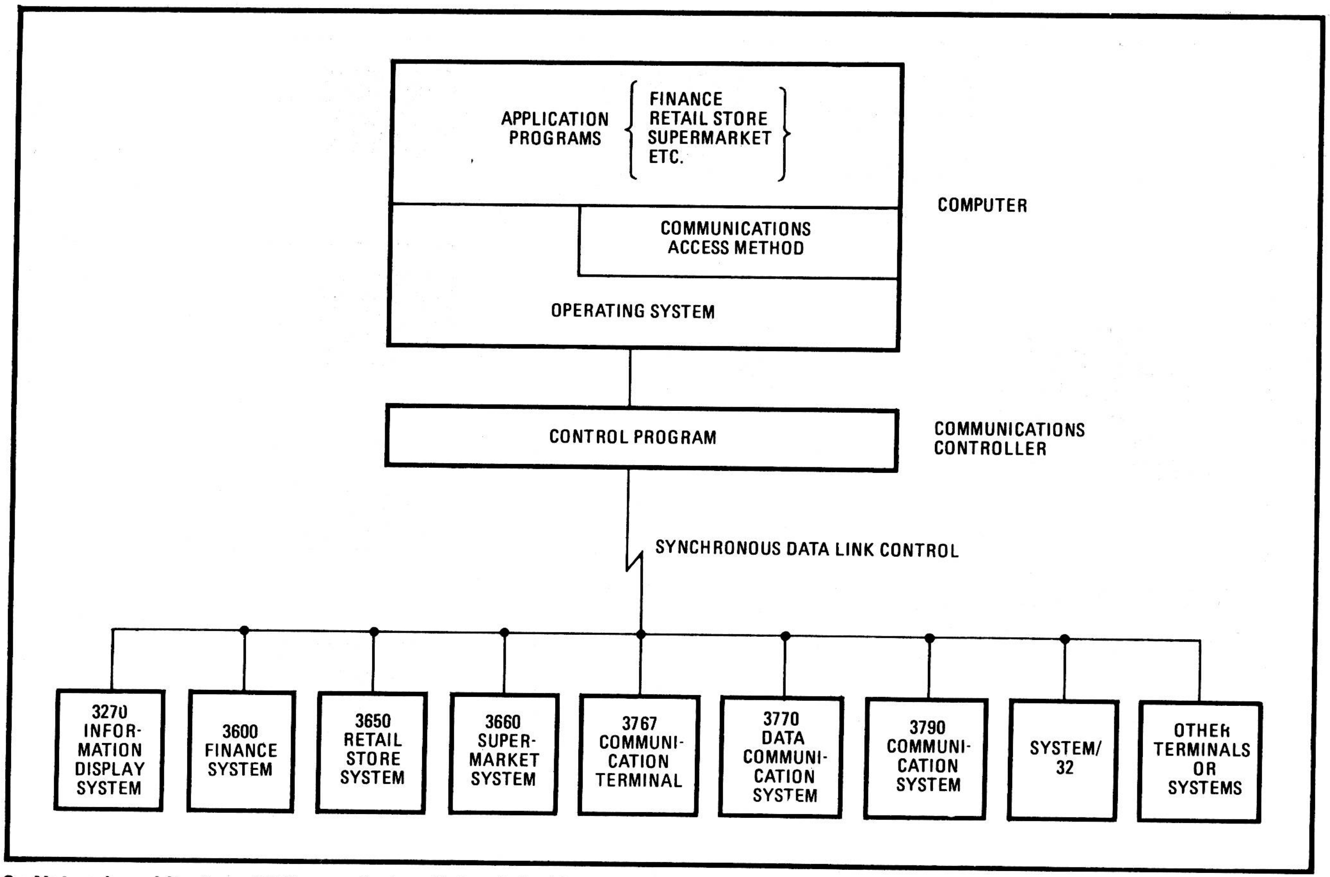
□ Distributed processing systems are still evolving, so some confusion may exist as to what they are and what they do. But there is no misunderstanding about their potential. "If you're looking for a wave of the future, you must look towards distributed systems," says C. W. Spangle, president of Honeywell Information Systems. There are several trends that back up his prediction.

As central processing and memory hardware have dropped in cost—in large measure because of advances in large-scale integration—it was natural that computer-system designers would look for ways to take advantage of such trends. Also, minicomputer manufacturers, continually improving their products, are now able to offer more capability for the same price. Technology thus is inexorably pushing the manufacturers up into a market area where they will compete with the mainframe makers.

User management, also, is helping the development of distributed systems by decentralizing operations and allowing individual installations or departments to become more self-sufficient both in records keeping and data processing.

Another trend indicating the growth of distributed processing techniques is the increasing presence of

1. Business terminal. The Control Data Cyber 18-10 operates as a small, stand-alone computer and as an intelligent terminal connected to a computer network. The unit also can emulate IBM 2780 and 3780 terminals, allowing it to be connected to those types of systems.


APPLICATION PROGRAMS
FINANCE
RETAIL STORE
SUPERMARKET
ETC.
COMPUTER
COMMUNICATIONS ACCESS METHOD
OPERATING SYSTEM
CONTROL PROGRAM
COMMUNICATIONS CONTROLLER
SYNCHRONOUS DATA LINK CONTROL
3270 INFORMATION DISPLAY SYSTEM
3600 FINANCE SYSTEM
3650 RETAIL STORE SYSTEM
3660 SUPERMARKET SYSTEM
3767 COMMUNICATION TERMINAL
3770 DATA COMMUNICATION SYSTEM
3790 COMMUNICATION SYSTEM
SYSTEM/32
OTHER TERMINALS OR SYSTEMS

2. Network architecture. IBM's new System Network Architecture covers the interconnection of several different types of terminal systems. Communication is through the company's SDLC protocol. A central 370 computer can run applications programs to support the terminals.

many small computers in one organization. How can their aggregate power be made available to all the users? Might users be connected into some network to allow them to multiply the processing power available through their individual minicomputers?

Although hardware has been making big jumps in performance, the necessary software is evolving only slowly. "A number of companies have combined limited local processing with remote-batch and data-inquiry capabilities," Spangle says. "But when you talk about completely distributed systems—systems with processing and information storage resident within the various operating components of an organization and with each application program accessible by programs at other sites—you are talking about things that are still largely experimental."

Thus, the problem with distributed processing, is, first of all, to define it. According to Neil Gorchow, Sperry Univac vice president for product strategy and requirements, distributed processing means whatever you want it to mean. "Are we talking about distributing the functions that the computer system performs into the auxiliary elements, or are we talking about distributing the processing of data back toward the source where the data is generated? The latter is my definition of distributed processing. Where I capture the data, I also begin to process it and then forward the results to some central source."

As Spangle points out, two types of computer networks are laying claims to the name of distributed processing. Remote locations are being equipped with lower-level processing systems that perform the functions needed there, sending only important results back to headquarters. In this category—by far the more prevalent today because of rapidly decreasing processor and memory costs—are the intelligent terminal systems offered by such manufacturers as Datapoint and Sycor. They can be used for order entry and preliminary processing of other transactions.

The other type is one in which network users have full access to all of its scattered resources. The Government's Arpanet [*Electronics*, May 2, 1974, p. 98] is the best example of this type—anyone connected can use the computing power available anywhere else in the network.

Software is the problem

However, the actual processing hardware does not impose as significant problems as does the software necessary to run the network. Individual computers installed at network nodes generally represent a variety of standard units and have their own operating systems and applications programs. The problem is to overlay some high-level network software that allows any one computer automatic access to the resources elsewhere in the network.

One way to view the intelligent terminals in the first type, in typical hierarchical applications, is as somewhat