Chapter 1

Introduction: Basic Concepts

Objective of this chapter:

Chapter one will provide an introduction to the Distributed Systems and how to characterize them. In addition, the chapter will describe the evolution of distributed systems as well as the research challenges facing the design of general purpose high performance distributed systems.

Key Terms

Complexity, Grid structure, High performance distributed system.

1.1 Introduction

The last two decades spawned a revolution in the world of computing, a move away from central mainframe-based computing to network-based computing. Today, workstation servers are quickly achieving the levels of CPU performance, memory capacity, and I/O bandwidth once available only in mainframes at a cost order of magnitude below that of mainframes. Workstations are being used to solve computationally intensive problems in science and engineering that once belonged exclusively to the domain of supercomputers. A distributed computing system is the system architecture that makes a collection of heterogeneous computers or workstations to act and behave as being one single computing system. In such a computing environment, users can uniformly access and name local or remote resources, and run processes from anywhere in the system, without being aware of which computers their processes are running on. Many claims have been made for distributed computing systems. In fact, it is hard to rule out any desirable feature of a computing system that has not been claimed to be offered by a distributed system [Comer et al, 1991]. However, the recent advances in computing, networking and software have made feasible to achieve the following advantages:

- **Increased Performance:** The existence of multiple computers in a distributed system allows applications to be processed in parallel and thus improve the application and system performance. For example, the performance of a file system can be improved by replicating its functions over several computers; the file replication allows several applications to access that file system in parallel. Also, file replication results in distributing the network traffic to access that file over different sites and thus reduces network contention and queuing delays.

- **Sharing of Resources:** Distributed systems **enable** efficient access for all the system resources. Users can share special purpose and sometimes expensive hardware and

software resources such as database server, compute server, virtual reality server, multimedia information server and printer server, just to name a few.

- **Increased Extendibility:** Distributed systems can be designed to be modular and adaptive so that for certain computations the system will configure itself to include a large number of computers and resources while in other instances, it will just consist of a few resources. Furthermore, the file system capacity and computing power can be increased incrementally rather than throwing all the system resources to acquire higher performance and capacity systems.

- **Increased Reliability, Availability and Fault Tolerance:** The existence of multiple computing and storage resources in the distributed system makes it attractive and cost-effective to introduce redundancy in order to improve the system dependability and fault-tolerance. The system can tolerate the failure in one computer by allocating its tasks to another available computer. Furthermore, by replicating system functions, the system can tolerate one or more component failures.

- **Cost-Effectiveness:** The performance of computers have been improving by approximately 50% per year while their cost is decreasing by half every year during the last decade [Patterson and Hennessy, 1994]. Furthermore, the emerging high speed optical network technology will make the development of distributed systems attractive in terms of price/performance ratio when compared to those of parallel computers. The cost-effectiveness of distributed systems has contributed significantly to the failure of the supercomputer industry to dominate the high performance computing market.

These advantages or benefits can not be achieved easily because designing a general purpose distributed computing system is several orders of magnitude more complex than the design of centralized computing systems. The design of such systems is a complicated process because of the many options that the designers must evaluate and choose from such as the type of communication network and communication protocol, the type of host-network interface, distributed system architecture (e.g., pool, client-server, integrated, hybrid), the type of system level services to be supported (distributed file service, transaction service, load balancing and scheduling, fault-tolerance, security, etc.) and the type of distributed programming paradigms (e.g., data model, functional model, message passing or distributed shared memory). Below is a list of the main issues that must be addressed.

- Lack of good understanding of distributed computing theory. The field is relatively new and to overcome that we need to experiment with and evaluate all possible architectures to design general purpose reliable distributed systems. Our current approach to design such systems is based on ad-hoc approach and we need to develop system-engineering theory before we can master the design of such systems**.** Mullender compared the design of a distributed system to the design of a reliable national railway system that took a century and half to be fully understood and mature [Bagley, 1993]. Similarly, distributed systems (which have been around for approximately two decades)

need to evolve into several generations of different design architectures before their designs, structures and programming techniques can be fully understood and mature.

- The asynchronous and independence behavior of the computers complicate the control software that aims at making them operate as one centralized computing system. If the computers are structured in a master-slave relationship, the control software is easier to develop and system behavior is more predictable. However, this structure is in conflict with the distributed system property that requires computers to operate independently and asynchronously.

- The use of a communication network to interconnect the computers introduces another level of complexity; distributed system designers need not only to master the design of the computing systems and their software systems, but also to master the design of reliable communication networks, how to achieve efficient synchronization and consistency among the system processes and applications, and how to handle faults in a system composed of geographically dispersed heterogeneous computers. The number of computers involved in the system can vary from a few to hundreds or even hundreds of thousands of computers.

In spite of these difficulties, there has been a limited success in designing special purpose distributed systems such as banking systems, on-line transaction systems, and point-of-sale systems. However, the design of a general purpose reliable distributed system that has the advantages of both centralized systems (accessibility, management, and coherence) and networked systems (sharing, growth, cost, and autonomy) is still a challenging task [Stankovic, 1984]. Klienrock [Tannenbaum, 1988] makes an interesting analogy between the human-made computing systems and the brain. He points out that the brain is organized and structured very differently from our present computing machines. Nature has been extremely successful in implementing distributed systems that are far cleverer and more impressive than any computing machines humans have yet devised. We have succeeded in manufacturing highly complex devices capable of high-speed computation and massive accurate memory, but we have not gained sufficient understanding of distributed systems and distributed applications; our systems are still highly constrained and rigid in their construction and behavior. The gap between natural and man-made systems is huge and more research is required to bridge this gap and to design better distributed systems.

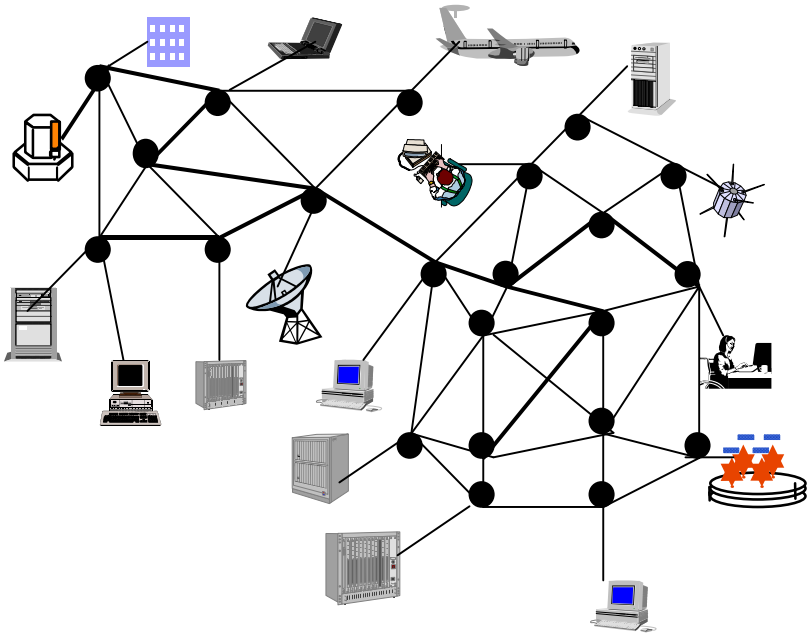

Figure 1.1 An Example of a Distributed Computing System.

The main objective of this book is to provide a comprehensive study of the design principles and architectures of distributed computing systems. We first present a distributed system design framework to provide a systematic design methodology for distributed systems and their applications. Furthermore, the design framework decompose the design issues into several layers to enable us to better understand the architectural design issues and the available technologies to implement each component of a distributed system. In addition to addressing the design issues and technologies for distributed computing systems, we will also focus on those that will be viable to build the next generations of wide area distributed systems (e.g. Grid and Autonomic computing systems) as shown in Figure 1.1.

1.2 Characterization of Distributed Systems

Distributed systems have been referred to by many different names such as distributed processing, distributed data processing, distributed multiple computer systems, distributed database systems, network-based computing, cooperative computing, client-server systems, and geographically distributed multiple computer systems [Hwang and Briggs, 1984]. Bagely [Kung, 1992] has reported 50 different definitions of distributed systems. Other researchers feel acceptable to have many different definitions for distributed systems and even warren against having one single definition of a distributed system [Liebowitz, and Carson, 1985; Bagley, 1993]. Furthermore, many different methods have been proposed to define and characterize distributed computing systems and distinguish them from other types of computing systems. In what follows, we present the important characteristics and services that have been proposed to characterize and classify distributed systems.

- **Logical Property.** The distributed system is defined as a collection of logical units that are physically connected through an agreeable protocol for executing distributed programs [Liebowitz and Carson, 1985]. The logical notion allows the system components to interact and communicate without knowing their physical locations in the system.

- **Distribution Property.** This approach emphasizes the distribution feature of a distributed system. The word "distributed" implies that something has been spread out or scattered over a geographically dispersed area. At least four physical components of a computing system can be distributed: 1) hardware or processing logic, 2) data, 3) the processing itself, and 4) the control. However, a classification using only the distribution property is not sufficient to define distributed systems and many existing computing systems can satisfy this property. Consider a collection of terminals attached to a mainframe or an I/O processor within a mainframe. A definition that is based solely on the physical distribution of some components of the system does not capture the essence of a distributed system. A proper definition must also take into consideration component types and how they interact.

- **Distributed System Components.** Enslow [Comer, 1991] presents a "research and development" definition of distributed systems that identifies five components of such a system. First, the system has a multiplicity of general-purpose resource components, including both hardware and software resources, that can be assigned to specific tasks on a dynamic basis. Second, there is a physical distribution of the hardware and software resources of the system that interact through a communications network. Third, a high-level operating system that unifies and integrates the control of the distributed system. Fourth, system transparency that permits services to be requested by name only. Lastly, cooperative autonomy that characterizes the operation of both hardware and software resources.

- **Transparency Property.** Other researchers emphasize the transparency property of the system and the degree to which it looks like a single integrated system to users and applications. Transparency is defined as the technique used to hide the separation from both the users and the application programs so that the system is perceived as one single system rather than a collection of computers. The transparency property is provided by the software structure overlaying the distributed hardware. Tanenbaum and VanRenesse used this property to define a distributed system as one that looks to its users like an ordinary centralized system, but runs on multiple independent computers [Bagley, 1993; Halsall, 1992] The authors of ANSA reference Manual [Borghoff, 1992] defined eight different types of transparencies:

1. Access Transparency: This property allows local and remote files and other objects to be accessed using the same set of operations.

2. Location Transparency: This property allows objects to be accessed without knowing their physical locations.

3. Concurrency Transparency: This property enables multiple users or distributed applications to run concurrently without any conflict; the users do not need to write any extra code to enable their applications to run concurrently in the system.

4. Replication Transparency: This property allows several copies of files and computations to exist in order to increase reliability and performance. These replicas are invisible to the users or application programs. The number of redundant copies can be selected dynamically by the system, or the user could specify the required number of replicas.

5. Failure Transparency: This property allows the system to continue its operation correctly in spite of component failures; i.e. it enables users and distributed applications to run for completion in spite some failures in hardware and/or software components without modifying their programs.

6. Migration Transparency: This property allows system components (processes, threads, applications, files, etc.) to move within the system without affecting the operation of users or application programs. This migration is triggered by the system software in order to improve system and/or application desired goals (e.g., performance, fault tolerance, security).

7. Performance Transparency: This property provides the system with the ability to dynamically balance its load and schedule the user applications (processes) in a transparent manner to the users in order to optimize the system performance and/or application performance.

8. Scaling Transparency: This property allows the system to be expanded or shirked without changing the system structure or modifying the distributed applications supported by the system.

In this book, we assume a distributed computing system resources might include a wide range of computing resources such as workstations, PC's, minicomputers, mainframes, supercomputers, and other special purpose hardware units. The underlying network interconnecting the system resources can span LAN's, MAN's and even WAN's, can have different topologies (e.g., bus, ring, full connectivity, random interconnect, etc.), and can support a wide range of communication protocols. In high performance distributed computing environments, computers communicate and cooperate with latency and throughput comparable to that experienced in tightly coupled parallel computers. Based on these properties, we define a distributed system as a networked (loosely coupled) system of independent computing resources with adequate software structure to enable the integrated use of these resources toward a common goal.

1.3 Evolution of Distributed Computing Systems

Distributed computing systems have been evolving for more than two decades and this evolution could **be** described in terms of four generations of distributed systems: Remote Execution Systems (RES), Distributed Computing Systems (DCS), and High Performance Distributed Systems (Grid computing systems), and Autonomic Computing. Each generation can be distinguished by the type of computers, the communications networks, the software environments and applications that are typically used in that generation.

1.3.1 Remote Execution Systems (RES): First Generation

The first generation spans the 1970's era, a time when the first computer networks were being developed. During this era, computers were large and expensive. Even minicomputers would cost tens of thousands of dollars. As a result, most organizations had only a handful of computers that were operated independently from one another and were located in one centralized computing center. The computer network concepts were first introduced around the mid 1970s and the initial computer network research was funded by the federal government. For example, the Defense Advanced Research Projects Agency (DARPA) has funded many pioneered research projects in packet switching including the ARPANET. The ARPANET used conventional point-to-point leased line interconnection. Experimental packet switching over radio networks and satellite communication channels were also conducted during this period. The transmission rate of the networks was slow (typically in the 2400 to 9600 bit per second (bps) range). Most of the software available to the user for information exchange was in providing the capability of terminal emulation and file transfer. Consequently, most of the applications were limited to remote login capability, remote job execution and remote data entry.

1.3.2 Distributed Computing Systems (DCS): Second Generation

This era spans approximately the 1980s, where significant advances occurred in the computing, networking and software resources used to design distributed systems. In this period, the computing technology introduced powerful microcomputer systems capable of providing computing power comparable to that of minicomputers and even mainframes at a much lower price. This made microcomputers attractive to design distributed systems that have better performance, reliability, fault tolerance, and scalability than centralized computing systems.

Likewise, network technology improved significantly during the 1980s. This was demonstrated by the proliferation and the availability of high-speed local area networks that were operating at 10 and 100 million bits per second (Mbps) (e.g., Ethernet and FDDI). These systems allowed dozens, even hundreds, of computers that varied from mainframes, minicomputers, and workstations to PCs to be connected such that information could be transferred between machines in the millisecond range. The wide area network's speed was slower than LAN' speed and it was in the 5600 bps to 1.54 Mbps range.

During this period, most of the computer systems were running the Berkeley UNIX operating system (referred to as the BSD UNIX) that was developed at the University of California's Berkeley Software Distribution. The Berkeley software distribution became popular because it was integrated with the Unix operating system and also offered more than the basic TCP/IP protocols. For example, in addition to standard TCP/IP applications (such as ftp, telnet, and mail), BSD UNIX offered a set of utilities to run programs or copy files to or from remote computers as if they were local (e.g., rcp, rsh, rexe.). Using the Berkeley socket, it was possible to build distributed application programs that were running over several machines. However, the users had to be aware of the activities on each machine; that is where each machine was located, and how to communicate with it. No transparency or support was provided by the operating system; the responsibility for creating, downloading and running the distributed application was solely performed by the programmer.

However, between 1985-1990, a significant progress in distributed computing software tools was achieved. In fact, many of the current message passing tools were introduced during this period such as Parallel Virtual Machine (PVM) developed at Oak ridge National Laboratory, ISIS developed by ISIS/Cornell University, Portable Programming for Parallel Processing (P4), just to name a few. These tools have contributed significantly to the widespread of distributed systems and their applications. With these tools, a large number of specialized distributed systems and applications were deployed in the office, medical, engineering, banking and military environments. These distributed applications were commonly developed based on the client-server model.

1.3.3 High-Performance Distributed Systems: Third Generation

This generation will span the 19
90's, which will be the decade where parallel and distributed computing will be unified into one computing environment that we refer to in this book as high performance distributed system. The emergence of high-speed networks and the evolution of processor technology and software tools will hasten the proliferation and the development of high performance distributed systems and their applications.

The existing computing technology has introduced processor chips that are capable of performing billions of floating point operations per second (Gigaflops) and are swiftly moving towards the trillion floating point operation per second (Teraflops) goal. A current parallel computer like the IBM ASCI White Pacific Computer at Lawrence Livermore National Laboratory in California can computer 7 trillion math operations a second. Comparable performance can now be achieved in high performance distributed systems. For example, a Livermore cluster contains, 2,304 2.4-GHz Intel Pentium Xeon processors have a theoretical peak speed of 11 trillion floating-point operations per second. In HPDS environment, the computing resources will include several types of computers (supercomputers, parallel computers, workstations and even PC's) that collectively execute the computing tasks of one large-scale application.

Similarly, the use of fiber optics in computer networks has stretched the transmission rates from 64 Kilobit per second (Kbps) in the 1970s to over 100 Giga bit per second (Gbps) as shown in Figure 1.2. Consequently, this has resulted in a significant reduction in the transmission time of data between computers. For example, it took 80 seconds to transmit a 1 Kbyte data over a 100 Kbps network, but it now takes only 8 milliseconds to transmit the same sized data over a 1 Gbps network. Furthermore, the current movement towards the standardization of terabit networks will make high-speed networks attractive in the development of high performance distributed systems.

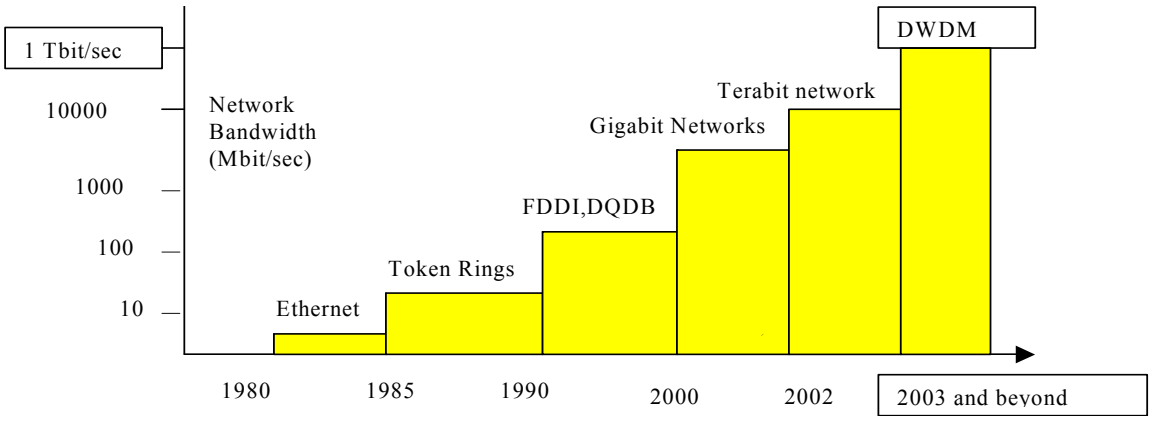

Figure1.2 Evolution of network technology

The software tools used to develop HPDS applications make the underlying computing resources, whether they are parallel or distributed, transparent to the application developers. The same parallel application can run without any code modification in HPDS. Software tools generally fall into three groups on the basis of the service they provide to the programmer. The first class attempts to hide the parallelism from the user completely. These systems consist of parallelizing and vectorizing compilers that exploit the parallelism presented by loops and have been developed mainly for vector computers. The second approach uses shared memory constructs as a means for applications to interact and collaborate in parallel. The third class requires the user to explicitly write parallel programs by message passing. During this period, many tools have been developed to assist the users developing parallel and distributed applications at each stage of the software development life cycle. For each stage, there exist some tools to assist the users with the activities of that stage.

The potential application examples cover parallel and distributed computing, national multimedia information server (e.g., national or international multimedia yellow pages server), video-on-demand and computer imaging, just to name a few [Reed, and Fujimoto, 1987]. The critical performance criteria for the real-time distributed applications require extremely high bandwidth and strict requirements on the magnitude and variance of network delay. Another important class of applications that require high performance distributed systems is the National Grand Challenge problems. These

problems are characterized by massive data sets and complex operations that exceed the capacity of current supercomputers. Figure 1.3 shows the computing and storage requirements for the candidate applications for high performance distributed systems.

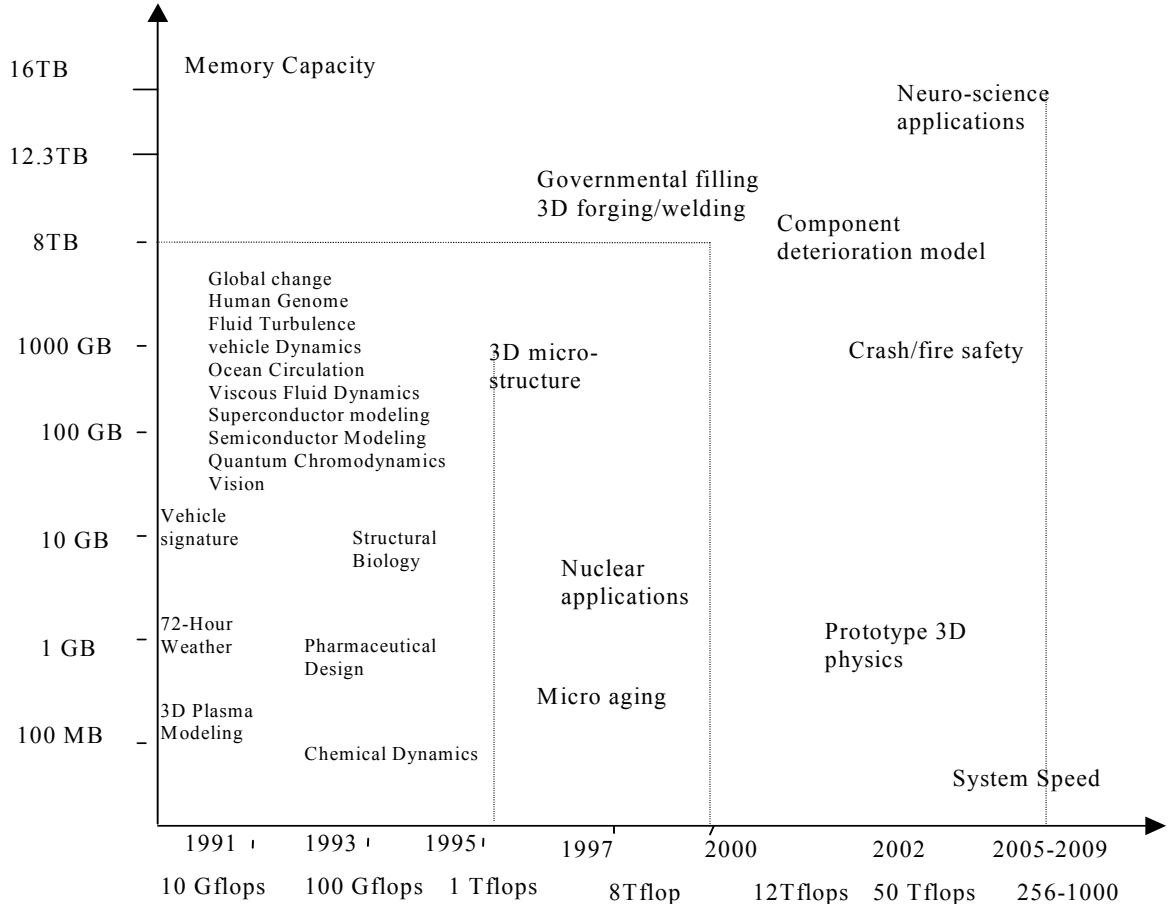

Figure 1.3 Computing and Storage Requirements of HPDS Applications

1.3.3 Autonomic Computing Systems: Fourth Generation

The autonomic computing concept was introduced in the early 2000 by IBM [www.research.ibm.com/autonomic]. The basic approach is to build computing systems that are capable of managing themselves; that can anticipate their workloads and adapt their resources to optimize their performance. This approach has been inspired by the human autonomic nervous system that has the ability to self-configure, self-tune and even repair himself without any human conscience involvement. The resources of autonomic systems include a wide range of computing systems, wireless and Internet devices. The applications cover a wide range of applications that touch all aspects of our life such as education, business, government and defense. The field is still in its infancy and it is expected to play an important role in defining the next era of computing.

Table 1.1 summarizes the main features that characterize computing and network resources, the software support, and applications associated with each distributed system generation.

Table 1.1 Evolutions of Distributed Computing Systems

Distributed System Generation	Computing resources	Networking resources	Software/ Application
Remote Execution Systems (RES)	Mainframe, Minicomputers; Centralized; Few and expensive	Packet switched networks; Slow WAN (2400-9600bps); Few networks	Terminal emulation; Remote login; Remote data entry
Distributed Computing Systems (DCS)	Workstation & PCs Mainframe, Minicomputers; Distributed; Not expensive	Fast LANs & MANs, 100Mbps, FDDI, DQDB, ATM; Fast WANs (1.5Mbps) Large number	Network File Systems (NFS); Message-passing tools (PVM,P4 ,ISIS); On-line transaction systems -Airline reservation, - online Banking
High-performance Distributed Systems (HPDC)	Workstations & PCs; Parallel/super computers; Fully distributed Explosive number	High-speed LANs, MANs, WANs; ATM, Gigabit Ethernet; Explosive number	Fluid turbulence; Climate modeling; Video-On-Demand; Parallel/Distributed Computing
Autonomic Computing	Computers of all types (PCs, Workstations, Parallel/supercomputers), cellular phone and Internet devices	High-speed LANs, MANs, WANs; ATM, Gigabit Ethernet; wireless and mobile networks	Business applications, Internet services, scientific, medical and engineering applications,

1.4 Promises and Challenges of High Performance Distributed Systems

The proliferation of high performance workstations and the emergence of high-speed networks (Terrabit networks) have attracted a lot of interest in high performance distributed computing. The driving forces towards this end will be (1) the advances in processing technology, (2) the availability of high speed networks and (3) the increased research directed towards the development of software support and programming environments for distributed computing. Further, with the increasing requirements for computing power and the diversity in the computing requirements, it is apparent that no single computing platform will meet all these requirements. Consequently, future computing environments need to adaptively and effectively utilize the existing heterogeneous computing resources. Only high performance distributed systems provide the potential of achieving such an integration of resources and technologies in a feasible manner while retaining desired usability and flexibility. Realization of this potential requires advances on a number of fronts-- processing technology, network technology and software tools and environments.

1.4.1 Processing Technology

Distributed computing relies to a large extent on the processing power of the individual nodes of the network. Microprocessor performance has been growing at a rate of 35--70 percent during the last decade, and this trend shows no indication of slowing down in the current decade. The enormous power of the future generations of microprocessors, however, cannot be utilized without corresponding improvements in the memory and I/O systems. Research in main-memory technologies, high-performance disk-arrays, and high-speed I/O channels are therefore, critical to utilize efficiently the advances in processing technology and the development of cost-effective high performance distributed computing.

1.4.2 Networking Technology

The performance of distributed algorithms depends to a large extent on the bandwidth and latency of communication among the network nodes. Achieving high bandwidth and low latency involves not only fast hardware, but also efficient communication protocols that minimize the software overhead. Developments in high-speed networks will, in the future, provide gigabit bandwidths over local area networks as well as wide area networks at a moderate cost, and thus increasing the geographical scope of high performance distributed systems.

The problem of providing the required communication bandwidth for distributed computational algorithms is now relatively easy to solve, given the mature state of fiber-optics and opto-electronic device technologies. Achieving the low latencies necessary, however, remains a challenge. Reducing latency requires progress on a number of fronts: First, current communication protocols do not scale well to a high-speed environment. To keep latencies low, it is desirable to execute the entire protocol stack, up to the transport layer, in hardware. Second, the communication interface of the operating system must be streamlined to allow direct transfer of data from the network interface to the memory space of the application program. Finally, the speed of light (approximately 5 microseconds per kilometer) poses the ultimate limit to latency.

In general, achieving low latency requires a two-pronged approach:

1. **Latency Reduction:** Minimize protocol-processing overhead by using streamlined protocols executed in hardware and by improving the network interface of the operating system.

2. **Latency Hiding:** Modify the computational algorithm to hide latency by pipelining communication and computation.

These problems are now perhaps most fundamental to the success of high-performance distributed computing, a fact that is increasingly being recognized by the research community.

1.4.3 Software Tools and Environments

The development of high performance distributed applications is a non-trivial process and requires a thorough understanding of the application and the architecture. Although, an HPDS provides the user with enormous computing power and a great deal of flexibility, this flexibility implies increased degrees of freedom which have to be optimized in-order to fully exploit the benefits of the distributed system. For example, during software development, the developer is required to select the optimal hardware configuration for the particular application, the best decomposition of the problem on the selected hardware configuration, the best communication and synchronization strategy to be used, etc. The set of reasonable alternatives that have to be evaluated in such an environment that is very large and selecting the best alternative among these is a non-trivial task. Consequently, there is a need for a set of simple and portable software development tools which can assist the developer in appropriately distributing the application computations to make efficient use of the underlying computing resources. Such a set of tools should span the software life-cycle and must support the developer during each stage of application development starting from the specification and design formulation stages through the programming, mapping, distribution, scheduling phases, tuning and debugging stages up to the evaluation and maintenance stages.

1.5 Summary

Distributed systems applications and deployment have been growing at a fast pace to cover many fields, such as education, industry, finance, medicine and military. Distributed systems have the potential to offer many benefits when compared to centralized computing systems that include increased performance, reliability and fault tolerance, extensibility, cost-effectiveness, and scalability. However, designing distributed systems is more complex than designing a centralized system because of the asynchronous behavior and the complex interaction of their components, heterogeneity, and the use of communication network for their information exchange and interactions. Distributed systems provide designers with many options to choose from and poor designs might lead to poorer performance than centralized systems.

Many researchers have studied distributed systems and used different names and features to characterize them. Some researchers used the logical unit concept to organize and characterize distributed systems, while others used multiplicity, distribution, or transparency. However, there is a growing consensus to define a distributed system as a collection of resources and/or services that are interconnected by a communication network and these resources and services collaborate to provide an integrated solution to an application or a service.

Distributed systems have been around for approximately three decades and have been evolving since their inception in the 1970s. One can describe their evolution in terms of

four generations: Remote Execution Systems, Distributed Computing Systems, High Performance Distributed Systems, and Autonomic Computing. Autonomic Computing Systems and High performance distributed systems will be the focus of this book. They utilize efficiently and adaptively a wide range of heterogeneous computing resources, networks and software tools and environments. . These systems will change their computing environment dynamically to provide the computing, storage, and connectivity for large scale applications encountered in business, finance, health care, scientific and engineering fields.

1.6 PROBLEMS

1. Many claims have been attributed to distributed systems since their inception. Enumerate all these claims and then explain which of these claims can be achieved using the current technology and which ones will be achieved in the near future, and which will not be possible at all.

2. What are the features or services that could be used to define and characterize any computing system? Use these features or properties to compare and contrast the computing systems built on the basis of
   - Single operating system in a parallel computer
   - Network operating system
   - Distributed system
   - High performance distributed system.

3. Describe the main advantages and disadvantages of distributed systems when they are compared with centralized computing systems.

4. Why is it difficult to design general-purpose reliable distributed systems?

5. What are the main driving forces toward the development of high performance distributed computing environment. Describe four classes of applications that will be enabled by high performance distributed systems. Explain why these applications could not run on second-generation distributed systems.

6. What are the main differences between distributed systems and high performance distributed systems?

7. Investigate the evolution of distributed systems and study their characteristics and applications. Based on this study, can you identify the types of applications and any additional features associated with each generation of distributed systems as discussed in Section 3?

8. What are the main challenges facing the design and development of large-scale high performance distributed systems that have 100,000 of resources and/or services?

Discuss any potential techniques and technologies that can be used to address these challenges.

References

1. Mullender, S., Distributed Systems, First Edition, Addison-Wesley, 1989.

2. Mullender, S., Distributed Systems, Second Edition, Addison-Wesley, 1993.

3. Patterson and J. Hennessy, Computer Organization Design: the hardware/software interface, Morgan Kaufamm Publishers, 1994.

4. Liebowitz B.H., and Carson, J.H., ``Multiple Processor Systems for Real-Time Applications", Prentice-Hall, 1985.

5. Umar, A., Distributed Computing, PTR Prentice-Hall, 1993.

6. Enslow, P.H., "What is a "Distributed"Data Processing System?", IEEE Computer, January 1978.

7. Kleinrock, L., ``Distributed Systems", Communications of the ACM, November 1985.

8. Lorin, H., ``Aspects of Distributed Computer Systems", John-Wiley and Sons, 1980.

9. Tannenbaum, A.S., Modern Operating Systems, Prentice-Hall, 1992.

10. ANSA 1997, ANSA Reference Mnaual Release 0.03 (Draft), Alvey Advanced Network Systems Architectures Project, 24 Hills Road, Cambridge CB2 1JP, UK.

11. Bell, G.,``Ultracomputer A Teraflop Before its Time", Communications of the ACM, pp 27-47, August 1992.

12. Geist, A., ``PVM 3 User's Guide and Reference Manual", Oak Ridge National Laboratory, 1993.

13. Birman, K. K. Marzullo, ``ISIS and the META Project", Sun Technology, Summer 1989.

14. Birman, K., et al ISIS User Guide and Reference Manual, Isis Distributed Systems, Inc, 111 South Cayuga St., Ithaca NY, 1992.

15. Spragins, J.D., Hammond, J.L., and Pawlikowski, K., ``Telecommunications Protocols and Design", Addison Wesley, 1991.

16. McGlynn, D.R., ``Distributed Processing and Data Communications", John Wiley and Sons, 1978.

17. Tashenberg, C.B., ``Design and Implementation of Distributed-Processing Systems", American Management Associations, 1984.

18. Hwang, K., and Briggs, F.A., ``Computer Architecture and Parallel Processing", McGraw-Hill, 1984.

19. Halsall, F., Data Communications, Computer Networks and Open Systems", Third Edition, Addison-Wesley, 1992.

20. Danthine, A., and Spaniol, O., ``High Performance Networking, IV", International Federation for Information Processing, 1992.

21. Borghoff, U.M., ``Catalog of Distributed File/Operating Systems", Springer-Verlag, 1992.

22. LaPorta, T.F., and Schwartz, M., ``Architectures, Features, and Implmentations of High-Speed Transport Protocols", IEEE Network Magazine", May 1991.

23. Kung, H.T., ``Gigabit Local Area Networks: A systems perspective", ``IEEE Communications Magazine", April 1992.

24. Comer, D.E., Internetworking with TCP/IP, Volume I, Prentice-Hall. 1991.

25. Tannenbaum, A.S., Computer Networks Prentice-Hall, 1988.

26. Coulouris, G.F., Dollimore, J., Distributed Systems: Concepts and Design, Addison-Wesley, 1988.

27. Bagley, ``Dont't have this one", October 1993.

28. Stankovic, J.A., ``A Perspective on Distributed Computer Systems", IEEE Transactions on Computers, December 1984.

29. Andrews, G., ``Paradigms for Interaction in Distributed Programs", Computing Surveys, March 1991.

30. Chin, R. S. Chanson, ``Distributed Object Based Programming Systems", Computing Surveys, March 1991.

31. The Random House College Dictionary, Random House, 1975.

32. Shatz, S., Development of Distributed Software, Macmillan, 1993.

33. Jain, N. and Schwartz, M. and Bashkow, T. R., ``Transport Protocol Processing at GBPS Rates", Proceedings of the SIGCOMM Symposium on Communication Architecture and Protocols, August 1990.

34. Reed, D.A., and Fujimoto, R.M., ``Multicomputer Networks Message-Based Parallel Processing", MIT Press, 1987.

35. Maurice, J.B., ``The Design and Implementation of the UNIX Operating System", Prentice-Hall, 1986.

36. Ross, `` An overview of FDDI: The Fiber Distributed Data Interface," IEEE Journal on Selected Areas in Communications, pp. 1043--1051, September 1989.