Deo Prakash Vidyarthi
Biplab Kumer Sarker
Anil Kumar Tripathi
Laurence Tianruo Yang

Scheduling in Distributed Computing Systems

Analysis, Design and Models

Springer

Scheduling in Distributed Computing Systems

Analysis, Design & Models

(A Research Monograph)

Scheduling in Distributed Computing Systems

Analysis, Design & Models

(A Research Monograph)

by

Deo Prakash Vidyarthi
Jawaharlal Nehru University
New Delhi, India

Biplab Kumer Sarker
Primal Fusion Inc.
Waterloo, Canada

Anil Kumar Tripathi
Banaras Hindu University
Varanasi, India

Laurence Tianruo Yang
St. Francis Xavier University
Antigonish, Canada

Springer

Authors:
Deo Prakash Vidyarthi
Jawaharlal Nehru University
School of Computer & Systems Sciences
New Mehrauli Road
New Delhi-110067
India
dpv@mail.jnu.ac.in

Biplab Kumer Sarker
Primal Fusion Inc.
Research and Development
7–258 King Street North
Waterloo, Ontario N2J 2Y9
Canada
biplab.sarker@gmail.com

Anil Kumar Tripathi
Banaras Hindu University
Institute of Technology
Department of Computer Engineering
Varanasi-221005
India
anilkt@bhu.ac.in

Laurence Tianruo Yang
St. Francis Xavier University
Dept. Computer Science
PO Box 5000
Antigonish NS B2G 2W5
Canada
ltyang@gmail.com

ISBN-13: 978-0-387-74480-3 e-ISBN-13: 978-0-387-74483-4

Library of Congress Control Number: 2008935404

Printed on acid-free paper

springer.com

ACKNOWLEDEGEMENT

We would like to acknowledge all individual and institution that helped in any form in the contribution of this book. It will not be out of place to pay sincere thanks to Prof. V.V.Menon, (Retired Professor, Department of Applied Mathematics, Institute of Technology, Banaras Hindu University, Varanasi, India) for his nice suggestions and accomplishments throughout our research activities. We would also like to acknowledge Prof. A.N.Mantri (Ex- Head, Department of Computer Science, Banaras Hindu University, Varanasi, India) for his sincere advice towards our research. Our students Mr. Alok Singh, Mr. Neeraj Asthana has provided technical support towards the completion of this book.

Our sincere thanks to Mr. Lutfi M. Omer Khanbary, a Ph.D. student, for typesetting the whole manuscript as per the specifications.

Finally, we would like to thank our family for their understanding and support while writing this book.

PREFACE

The rapid growth of network technologies, processor architecture and software development has facilitated meaningful attempts to exploit the capabilities of a collection of computers for speeding up computations and services. A distributed system consists of various servers integrated in such a manner so as to appear as one system, whereas a distributed computing system (DCS), also appearing as one system to the user, aims at distributing the parts of a task submitted to it, to various participating nodes of the system. Thus one may view a distributed computing system as one that tries to minimize the execution time of tasks submitted to it by exploiting as many computing nodes as possible and plausible. A distributed system may also have computing nodes that may be known as compute servers, and co-operatively execute various modules of tasks submitted; apart from the services that it runs e.g. print, mail, name etc. In this book, distributed system has been used quite frequently to refer to the distributed computing system, because the objective is scheduling of the computational load.

The distribution of a computation load across processing nodes, forming a DCS, has been a challenging task. Many researchers have contributed to study of this problem during the last two decades. The problem consists of allocation of task modules to various processing nodes so as to incur as minimum as possible interprocessor communication overhead and thereby obtaining good execution speed

as opposed to a single processor execution. Many a times the inter-processor communication may be too substantive compared to the total execution time.

The approaches for task scheduling in operating system for a distributed computing system must consider the multiplicity of processing nodes with underlying interconnection network unlike the case of a single processor system. In the case of uniprocessor system, the objective is to make the processors busy executing jobs all the time by insuring that it does not idle and this serves the purpose.

In a distributed computing system, the scheduling of various modules on particular processing nodes may be preceded by appropriate allocation of modules of the different tasks to various processing nodes and then only the appropriate execution characteristic can be obtained. Thus task allocation becomes the important most and major activity in the task scheduling within the operating system of a DCS.

Various research papers have addressed this problem during the last two decades. As the problem is quite difficult, most of the solutions proposed had simplifying assumptions. The very first assumption has been: consideration of a single task only, second no consideration of the status of processing nodes in terms of the previously allocated modules of various tasks and third the capacity and capability of the processing nodes. The solutions reported in the beginning even assume that the precedence constraints amongst the modules of a task are non existent or negligible. Nevertheless many good algorithms were proposed for the purpose.

This book consists of various proposed algorithms for task allocation as part of scheduling in an operating system for DCS. It starts with analyzing the existing propositions, considering the precedence constraint and improving the known algorithms, proposing a solution for minimizing intermodule communication apart from the main and important contribution, made in this book, in the form of the allocation algorithms that aim at distribution of computational modules belonging to multiple tasks onto the various processing nodes considering there status in terms of previous allocations and capacity. As the problem is NP-Hard, the techniques of A*, GA etc. have been purposefully used to propose the algorithmic solutions.

The meaningful contributions have been organized in the chapters as given below:

Chapter 1 discusses the possible performance improvement in computing system. It also addresses how the distributed computing system has evolved over the years and the issues in DCS research.

Chapter 2 briefs about the distributed computing system. It discusses various architectural models of DCS. Transparency is one of the biggest issues in the design of a DCS that gives the DCS a single system image. Chapter 2 points out transparency issues of the DCS. Fault tolerance and synchronization in the DCS has also been briefed in chapter 2.

Chapter 3 defines the scheduling problem and identifies the characteristic parameter for a scheduler. It indicates the task allocation issues. Assumptions and notation, used in this book, have been kept together at one place in chapter 3.

Chapter 4 addresses the load balancing problem in a DCS. It defines load distribution, load balancing methodology, migration and also the conflict between load balancing and task allocation.

Chapter 5 briefs the earlier task allocation models. Propositions, which consider the precedence amongst the modules of the task and multiprogramming of the individual nodes, have been proposed in section 5.3 of chapter 5 using list schedule. In the same chapter an inter module communication reduction model is also proposed (sec. 5.4), which incurs a heavy penalty in total turnaround time of any task.

Chapter 6 proposes a few Load Balancing Task Allocation (LBTA) models. It discusses the LBTA strategy and its solution for single and multiple tasks.

Chapter 7 uses the important search technique of GA in proposing two allocation models, one, which incorporates problem specific knowledge for quick convergence and the other to maximize the reliability of the DCS with allocation.

Chapter 8 considers the important proposition of multiple tasks allocation and proposes a few models. In section 8.1, an allocation algorithm based on A* is proposed and in section 8.2 a new and novel idea of cluster based allocation is proposed. Cluster based allocation model avoids the priori requirement of execution

time of modules of the task on the processing nodes and thus can be proved to be very useful model of task allocation. Sections 8.3 and 8.4 deal with the LBTA strategy using A* and GA respectively.

Chapter 9 proposes few other approaches for task allocation models. These are hybrid and object oriented models.

Computational Grid is an emerging form of distributed computing. Chapter 10 concentrates on Grid Computing systems and discusses the scheduling problem for a computational grid. “What are the various issues in Scheduling for Grid Computing systems?” finds place in chapter 10.

Finally concluding remarks are made in chapter 11. This chapter also discusses the structure and place of a scheduler in a DCS.

TABLE OF CONTENTS

CHAPTER 1

Introduction

A Distributed Computing System (DCS) falls in the category of disjoint memory multiple processor architecture with an underlying processor-to-processor interconnection network. Such a private memory-processor interconnection network is known to constitute a multi-computer system only if the programmers need to consider the multiplicity of the machines, in programming a solution to the problem. In case of a distributed computing system the entire system appears as a centralized system to the user submitting a task; meaning thereby that it is the responsibility of the system to distribute the computational modules of the given task to various processing nodes for their efficient execution unlike the case of multi-computer system as stated above.

With the proliferation of large-scale inter-networks, the idea of distributed computing system has been gaining importance. In a distributed computing system various computational and informational resources are dispersed over a wide geographical area with appropriate servers maintaining them at locations and providing services to clients hooked onto these systems. The idea is that a distributed computing system may receive a task that requires various named services from various servers and in this case the job of the operating system is to provide the

appropriate connectivity and the service mechanism. In case of a computational task, consisting of various modules, the requirement is that of identification of appropriate computing nodes in the distributed computing system for scheduling the executable modules of the task so as to achieve a good turnaround for such a task and possibly an increase in the throughput of the computing system. This problem has been studied as task scheduling or task allocation problem in the literature [1-7]. This book deals with the problem of task scheduling/ allocation in a distributed computing system.

The following section 1.1 reviews the various ways of performance improvement in computing system including parallel computing with multiprocessors, multi-computers and distributed computing environment for the sake of completeness. Section 1.2 discusses the role of distributed computing system in high speed computing. Section 1.3 takes a view how the DCS has evolved, as a computing system, over the time. Section 1.4 deals with the research issues in distributed computing systems. Final section 1.5, describes the organization of the book.

1.1 Performance Improvement in Computing System

Parallel processing has emerged as a key enabling technology in modern computers, driven by the increasing demand for higher performance, lower cost and sustained productivity in real-life applications. Concurrent events are taking place

in today's high performance computers due to common practice of multiprogramming, multiprocessing and multi-computing. Modern computers are equipped with powerful hardware facilities driven by extensive software packages [8].

Parallel processing and distributed processing are closely related. In some cases certain distributed techniques are used to achieve parallelism. As the communication technology advances progressively, the distinction between parallel and distributed processing becomes smaller and smaller. In this extended sense, we may view distributed processing as a form of parallel processing in a special environment [9].

It has long been recognized that the concept of computer architecture is no longer restricted to the structure of the bare machine hardware. It is an integrated system of machine hardware, system software, application programs and user interfaces. Depending on the nature of the problems, the solutions may require different computing resources. The rapid progress made in hardware technology has significantly increased the economic feasibility of building a new generation of computers adopting parallel processing. Two categories of parallel computers are architecturally modeled. These physical models are distinguished by having a shared common memory and unshared distributed memories. Multiprocessors are called tightly coupled systems due to the high degree of resource sharing (including memory). Symmetric multiprocessors are those in which all processors have equal access to all peripheral devices. In such system, all the processors are equally capable of running the executive programs, such as OS kernel and I/O ser-

vice routines etc. In contrast to this, in an asymmetric multiprocessor system only one or a subset of processors is of executive capable [8].

The distributed memory multi-computer consists of multiple computers, often called nodes, interconnected by a message-passing network. Each node is an autonomous computer consisting of a processor, local memory and sometimes attached disks or I/O peripherals.

Distributed computing system falls in the category of distributed memory parallel architecture and is characterized by resource multiplicity and system transparency. The advantage of the DCS is that they are capable of incremental growth[5] i.e. it is possible to gradually extend the power and functionality of a distributed computing system by simply adding additional resources (both hardware and software) to the system as and when the need arises. For example, additional processors can be easily added to the system to handle the increased workload of an organization that might have resulted from its expansion. With the rapidly increasing power and reduction in the price of microprocessors, DCS potentially have a much better price performance ratio than a single large centralized system. Moreover the existing microcomputers, minicomputers or even a workstation can be added to the DCS for its better utilization.

1.2 High Speed Computing and DCS

In practice, parallelism appears in various forms, such as look ahead, pipelining, vectorization, concurrency, simultaneity, data parallelism, partitioning, interleaving, overlapping, multiplicity, replication, time sharing, space sharing, multitasking, multiprogramming, multithreading and distributed computing at different processing levels. All forms can be attributed to levels of parallelism, computational granularity, time complexities, communication latencies, scheduling policies and load balancing [10]. DCSs are naturally attractive as existing interconnected computers can be used to assign them various parts of a computational task to achieve parallelism.

The definition of high speed computing has undergone many changes in recent years. Perhaps, the most notable development in the evolution from the industry, dominated by vector mainframe architectures, to one in which massively parallel computers have been the primary choice for solving computationally intensive problems. As an alternative to massively parallel computers, increasing interest has immerged in distributed computing in which networked collection of dedicated or general purpose workstations are treated as a parallel computer. Although this method has existed for many years, two developments have served as catalysts to the rapid growth in the use of such cluster-based computing. First, high performance workstations with microprocessors that challenge custom-made architectures are widely available at relatively low cost. Second, several software packages have been developed to assist the programmer in process management, inter-process communication and program monitoring/debugging in a distributed environment [11].

The researches in the area of parallel computing have been indicating the availability of immense computing power, for execution of properly distributed and coordinated parts of jobs submitted to the system from time to time. The confluence of low-cost high performance processors and interconnection technologies has spurred a great interest in the advancement of computer architectures. The often-cited advantages of these architectures include high performance, availability, and extensibility at lower cost. As pointed out earlier, the computer architectures, consisting of interconnected multiple processors are of two types:

(i) Multiprocessors, known as tightly coupled systems, allow sharing of global memory by multiple processes running on their processors and communication amongst the processes is actuated by use of the shared variables. In such coupled systems, the number of processors that can be usefully deployed is usually small and limited by the bandwidth of shared memory.

(ii) Multicomputers and DCSs consist of a number of independent processors with private memory units and the IPC is done by message passing making use of the processors interconnection.

1.3 Evolution of the DCS

The processors in a DCS may vary in the size and the functionality. They may include small microcomputers, workstations, minicomputers and large general purpose computer systems. For a particular processor its own resources are local, whereas the other processors and their resources are remote. Together, a processor and its resources are usually referred to as a node or site or machine of the distributed computing systems. Resource sharing, computational speedup, reliability and communication over distances are the main reasons for building the DCS [12].

DCSs have become more and more attractive in recent years due to the advancement of VLSI and computer networking technologies. DCS not only provide the facility for utilizing remote computer resources and data but also increase the throughput by providing facilities for multiprogramming and parallel processing [13]. Furthermore, modularity, flexibility and reliability of the DCS make them attractive for many types of application.

The advent of time-sharing systems was the first step towards building the DCS because it provides us with two important concepts used in DCS; the sharing of computer resources simultaneously by many users and the accessing of computers from the different places. The centralized time-sharing systems had a limitation that their terminals/workstations could not be placed very far from the main computer room/system (like in minicomputers) since ordinary cables were used to connect the terminals to the main computer. But the advancement of computer networking technologies LAN (Local Area Network) and WAN(Wide Area Network) allow hundreds, even thousand of computers to be connected (may be resid-

ing in different cities or countries or continents) in such a way that the small amounts of information can be transferred between computers in a fraction of second or so. Recently there has been another major advancement in networking technology, the ATM (Asynchronous Transfer Mode) technology, which makes very high speed networking possible in both LAN and WAN environments. The availability of such high bandwidth networks allows DCSs to support a completely new class of distributed applications called multimedia applications that deal with the handling of a mixture of information, including voice, video and ordinary data [14].

The operating systems commonly used for DCS can be broadly classified into two types- Network Operating System (NOS) and Distributed Operating System (DOS) [14].

In NOS, the users are aware of the multiplicity of the machine and can access the resources either by logging into the appropriate remote machine or transferring the data from the remote machine to their own machine. On the other hand, in DOS the users would not be aware of the multiplicity of machines. It provides a single system image to its users. Users access remote resources in the same manner as they access local resources. A DOS dynamically and automatically allocates tasks to the various machines of the system for its processing.

In NOS, each computer of the system has its own local operating system (the operating systems of different computers may be the same or different) that functions

independently of the other machines meaning thereby that each one makes independent decisions about the creation and termination of their own processes and management of local resources. Due to the possibility of difference in local operating systems, the system calls for different machines of the same DCS may be different in this case. On the other hand, in DOS which is a single system wide operating system and each machine of the DCS runs a part of this global operating system. There is a single set of globally valid system calls available on all computers of the DCS.

The fault tolerance capability of a DCS is usually very high as compared to that of a networked system. If some computers fail in NOS, then several users are unable to continue with their work. On the other hand, in case of a DOS, most of the users can continue their work normally with only some percentage of loss in performance of the DCS.

1.4 Issues in DCS Research

The hardware issues of building a distributed computing system were fairly well understood, the major stumbling block is the availability of adequate software for making these systems easy to use and exploit its power fully. Therefore, since 1970, a significant amount of research work was carried out in the area of distributed operating system. Designing a distributed operating system is more difficult than a centralized one mainly because of the non-availability of complete informa-

tion about the system environment [4-7, 12, 14]. There is no common clock and various resources are physically separated in DCS in contrast to a centralized system. Despite these, the users of the DCS are to be provided all the advantages of the system. To meet these challenges the researchers, in the DCS discipline, must deal with several important issues. Some of these key issues are identified and discussed below.

The distributed computing system is designed in such a way that the collection of various machines, connected by an interconnection network, appears as a virtual uniprocessor system. Achieving complete transparency is a difficult task and research is still continuing on this issue. Of the several transparency issues identified by the ISO Reference Model for Open Distributed Processing, location transparency, migration transparency and concurrency transparency are very important [6-7, 14].

The often-advocated advantage of the DCS, in comparison to the centralized system, is the reliability due to the existence of multiple resources. However, only the multiple instances of resources cannot increase the reliability of the DCS, rather the various processes of the distributed operating system (viz. memory manager, task scheduler etc.) must be designed properly to increase the reliability by extracting the characteristic features of the DCS.

Another important issue is flexibility. It is more required for open distributed system [4-5, 14].

Performance improvement of an application running on the DCS than that of single processor system is another desired feature. To achieve this though the various components of the distributed operating system are taken into account, but the most important role is that of a scheduler or task allocator. The turnaround and throughput are the two important characteristic measures for the performance improvement.

Another issue in DCS research is scalability that refers to the capability of the system to adapt to an increase in the service load. A distributed computing system should be designed to cope with the growth of the processing nodes and the users as well. How to design a system so that such growth should not cause any serious disruption of services is very important research issue in the DCS.

Growth, in the number and types, of processing nodes introduces another dimension that is inevitable to have dissimilar hardware or software. Many users often prefer heterogeneity because it provides the flexibility of different computer platforms for different applications. Designing heterogeneous system is far more difficult than a homogeneous one.

In order that the users can trust the system, the various resources must be protected against destruction and unauthorized access. Enforcing security in a DCS is another important research area and is much more difficult than in a centralized one.

The book discusses one of the research issues that of task scheduling/allocation thoroughly. The problem is as such an NP-Hard problem and thus various feasible solutions are possible. The authors present and discuss all those task scheduling models that have been proposed by the authors themselves.

1.5 Organization of the Book

The book is organized in ten chapters. The current chapter, which is the first one, is an introductory chapter. Second chapter defines the task scheduling problem of the DCS. Chapter 2 takes a cursory look over the distributed system. What is exactly expected out of scheduling and how it has been addressed in this work, is detailed in chapter 3. Load balancing is an important aspect of the scheduling problem and is pursued in the chapter 4. Some of the earlier task allocation models, their limitations and few proposed trivial models for task allocation have been discussed in chapter 5. A precedence constrained task allocation model, in which the emphasis is on the precedence of the modules [15] and that minimizes the turn-around time of the given task is discussed in sec 5.3 of chapter 5. The effect of already allocated modules of unrelated tasks, on the processing elements comprising the system, is considered (assuming round robin scheduling) in this model.

Communication amongst the modules adds to the cost of overall execution of the task, for the allocation being considered, if its modules are to execute onto the

distant processing nodes of the DCS. An IMC cost reduction model (section 5.4), an aid in allocation algorithms, uses fuzzy logic to consider high and low communicating modules. The same fuzzy function is applied to determine near or distant nodes of the DCS. This IMC cost reduction model can be introduced in any task allocation algorithms at minimum cost [16].

Load balancing task allocation models find place in chapter 6. This chapter considers load balancing strategies and discusses the LBTA solutions for both the single and multiple task allocation.

As the task allocation problem is an NP-hard problem, Genetic Algorithms (GA) is found to be suitable to solve it. Two task allocation models, based on GA, have been proposed in chapter 7. First one aims at minimizing turn-around time of a task (sec. 7.1) and the second (sec. 7.2) gives an allocation that maximizes the reliability of the DCS as desired in some systems. The TA model proposed in section 7.1 [17], is based on a finding that the incorporation of some problem specific knowledge in construction of the GA, improves its performance and solution converges quickly [18]. This algorithm considers the inclusion of all possible constraints in the model, and as suggested in [19] will converge quickly.

Task allocation models for maximizing reliability of a DCS have appeared in the past [20-22]. We applied GA to maximize reliability of the DCS with task allocation and the same is discussed in section 7.2. The algorithm not only gets the advantage of GA for quick convergence but also produces better solutions in terms

of allocation with improved reliability [23]. The result is compared with that of Shatz [21] and it shown that the proposed model performs better. Many more inferences are drawn.

The TA models, proposed by researchers in the past, have considered the modules of a single task and assume that processing nodes have enough memory to accommodate unlimited modules. In a realistic situation multiple tasks arrive and at any instance of time, the Processing Element (PE) has modules of earlier tasks allocated and the memory occupied by it. In chapter 8, multiple tasks allocation in DCS is deliberated. Multiple task allocation, using A*, appears in sec. 8.1 [24]. To implement this, the concept of Global Table is introduced. Section 8.2 proposes a new idea of cluster-based approach of load partitioning and allocation in DCS. Cluster of the modules, based on communication requirement and cluster of PEs based on interprocessor distance is formed. Allocation is decided from task cluster to processor cluster. This model has the advantage that it does not require the priori knowledge of execution time of the modules of the tasks onto nodes of the DCS. As the allocation algorithms, in this chapter, consider multiple tasks and status of PEs due to previous allocations, these are not comparable with other models proposed in the literature. Section 8.3 discusses the load balancing task allocation for multiple tasks execution in DCS using A* and section 8.4 discusses the same using GA.

Chapter 9 makes a comparative analysis of scheduling models based on A* and GA and proposes a hybrid model using both A* and GA. This chapter also discusses the object allocation as most of the system are going object oriented.

Grid Computing is another form of Parallel and Distributed Computing. Computational grid is an emerging computing system so chapter 10 is dedicated to the discussion on the scheduling in computational grid. This chapter details various research issues in Grid scheduling.

Chapter 11 is the concluding chapter that summarizes the whole book. Structure and place of scheduler in Distributed Operating System is briefed in sec. 11.2 of this chapter.

The abbreviations used in the book are listed at one place for quick reference.

Finally, we have listed few programs written to carry out the experiments in the appendix given is last.

BIBLIOGRAPHY

[1]V.Rajaraman, C.Siva Ram Murthy, *Parallel Computer: Architecture and Programming*, Prentice Hall of India Ltd, 2000.
[2]M. Sasikumar, Dinesh Shikhare, P. Ravi Prakash, *Introduction to Parallel Processing*, Prentice Hall of India Ltd, 2000.
[3]Sanjeev K. Setia, Mark S. Squillante, Satish K. Tripathi, "Analysis of Processor Allocation in Distributed –Memory Parallel Processing Systems", *IEEE Trans. on Parallel & Distributed Systems*, Vol. 5, No. 4, April 1994, pp. 401-420.
[4]A.S.Tanenbaum, *Distributed Operating Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[5]A.S.Tanenbaum, R. Van Ressen, "Distributed Operating Systems," *ACM Computing Surveys*, Vol. 7, No.4, 1985, pp. 419-470.

[6]G.Colouris, J. Dollimore, T. Kindberg, *Distributed Systems: Concepts and Design*, Addison-Wesley Publishers Ltd., 1994.
[7]G.Nutt, Operating Systems: A Modern Perspective, Addison Wesley, 2000.
[8]Kai Hwang, *Parallel Computer Architecture*, Mc-Graw Hill International Edition, 1995.
[9]Kai Hwang, F.A.Briggs, *Computer Architecture and Parallel Processing*, McGraw Hill International Edition, 1995.
[10]Kai Hwang, Advanced Computer Architecture: Parallelism Scalability and Programmability, Mc-graw Hill, 1993.
[11]M.Hamdi and C.K.Lee, "Dynamic Load Balancing of Image Processing Applications on Clusters of Workstations," *Parallel Computing*, 22(1997), pp.1477-1492.
[12]A.Silberschatz and P.B. Galvin, *Operating Systems Concepts*, Addison- Wesley, 1998.
[13]C.C.Shen and W.H.Tsai , "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using a Minimax Criterion," *IEEE Trans. Computers*, Vol. c-34, no.3, March. 1985. pp. 197-203.
[14]P.K.Sinha, Distributed Operating Systems: Concepts and Design, Printice-Hall of India, 1997.
[15]D.P.Vidyarthi, A.K.Tripathi, "Precedence Constrained Task Allocation in Distributed Computing System", *Int. J. of High Speed Computing*, Vol. 8, No. 1, 1996, pp. 47-55.
[16]D.P.Vidyarthi, A.K.Tripathi, " A Fuzzy IMC Cost Reduction Model for Task Allocation in Distributed Computing Systems", *Proceedings of the Fifth International Symposium on Methods and Models in Automation and Robotics*, Vol. 2, Szczecin, Poland, August 1998, pp. 719-721.
[17]A.K.Tripathi, D.P.Vidyarthi, A.N.Mantri, "A Genetic Task Allocation Algorithm for Distributed Computing System Incorporating Problem Specific Knowledge", *International J. of High Speed Computing*, Vol.8, No.4, 1996, pp. 363-370.
[18]John J. Grefenstelle, "Incorporating Problem Specific Knowledge into Genetic Algorithm", *Genetic Algorithm and Simulated Annealing*, Morgan Kaufrman Publisher, California, 1987.
[19]K.Efe, "Heuristic Models of Task Assignment Scheduling in Distributed Systems", *IEEE Computer*, Vol. 15, June 1982, pp. 50-56.
[20]S.Kartik, C.S.Ram Murthy, "Task Allocation Algorithms for Maximizing Reliability of Distributed Computing Systems", *IEEE Trans. on Computers*, Vol.46, No.6, June1997, pp. 719-724.
[21]Sol.M.Shatz, Wang Goto, "Task Allocation for Maximizing Reliability of Distributed Computing Systems", *IEEE Trans. on Computers*, Vol.41, No.9, September 1992, pp. 1156-1168.
[22] Karthik, C. Siva Ram Murthy, " Improved Task Allocation Algorithms to Maximize Reliability of Redundant Distributed Systems", *IEEE Trans. on Reliability*, Vol. 44, No. 4, Dec. 1995, pp. 575-586.
[23]D.P.Vidyarthi, A.K.Tripathi, "Maximizing Reliability of Distributed Computing Systems with Task Allocation using Simple Genetic Algorithm", *J. of Systems Architecture*, Vol. 47, 2001, pp. 549-554.
[24]D.P.Vidyarthi, A.K.Tripathi, B.K.Sarker, "Multiple Task Management in Distributed Computing Systems", The *Journal of the CSI*, Vol. 31, No. 1 Sep. 2000, pp. 19-25.

CHAPTER 2

An Overview of a Distributed System

This chapter takes a cursory view of a distributed system. It discusses various architectural models of a distributed system and various types of transparencies involved. It also discusses fault tolerant, one of the very important property of a distributed system. A clock of the distributed system makes its computing nodes independent. How the system will be synchronized, in spite of the distribution of the clock, has been discussed in this chapter. The objective in this chapter is to introduce briefly the important properties of a distributed system to its readers, before moving to the scheduling aspect in a distributed computing system. The discussion, in general, is on a distributed system as the distributed computing system differs from the distributed system in its objective of handling the computational load as mentioned at other places as well.

2.1 DCS Architecture Models

Various models for the design of a distributed computing system have been proposed. We discuss here few models.

2.1.1 Workstation model

The workstation model is the simplest one and is consists of several workstations connected by a common communication network (Fig. 2.1). This is the most popular model also as it uses the available legacy systems in designing a distributed computing system. It has been observed that not often all the workstation in an organization are used all the time, whereas they are on and can be used for the execution of the jobs belonging to the other user. Thus by connecting all the workstation of an organization, all the workstation can be utilized fully and this will result in parallel execution of the jobs reducing the overall completion time.

The issue may arise that what happens when a workstation was executing a job of some other workstation. These issues have been addressed by Tanenbaum [2].

A user logs onto his/her workstation and submits the job to be executed. This job will be exploited for available parallelism and thus various concurrent modules of the job will be allocated onto any free workstation. This way the cooperation amongst the workstation will result in the concurrent execution of the job.