Dr. Priyesh Kanungo

Scheduling in Distributed Computing Environment Using Dynamic Load Balancing

Anchor Academic Publishing

disseminate knowledge

Kanungo, Priyesh: Scheduling in Distributed Computing Environment Using Dynamic Load Balancing, Hamburg, Anchor Academic Publishing 2016

PDF-eBook-ISBN: 978-3-96067-546-4
Druck/Herstellung: Anchor Academic Publishing, Hamburg, 2016

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Bibliographical Information of the German National Library:
The German National Library lists this publication in the German National Bibliography. Detailed bibliographic data can be found at: http://dnb.d-nb.de

SILENT FEATURES OF THIS BOOK

This book illustrates distributed computing concepts and the steps involved in processor management in computing cluster, server cluster and grid. The problem of poor resource utilization due to uneven processing load in distributed systems is studied and techniques of solving the problem using dynamic load balancing have been suggested. It describes detailed algorithms for scheduling using dynamic load balancing. Various theoretical concepts, experiments and examples enable students in understanding the process of dynamic load balancing.

The book is suitable for the students of Distributed Computing, Operating Systems and Advance Operating Systems subjects of B.E., M.C.A., M. Tech. and Ph.D courses.

PREFACE

This century has presented new challenges for distributed systems. These challenges include manifold increase in the number of information sources and the number of users. With the growing demand of resource intensive distributed computing applications, the need of using sophisticated techniques to improve the performance has also increased. Distributed systems suffer from uneven process arrivals which causes load imbalance, where some nodes are overloaded while other nodes are underloaded, or even idle. Dynamic load balancing is a distributed scheduling technique which may be used to improve reliability and overall throughput not only on a cluster of nodes and workstations, but also on a server cluster. It distributes processing workload evenly to improve response time and to maximize resource utilization.

In this book, the problem of poor resource utilization due to uneven processing load in distributed systems is studied and techniques of solving the problem using dynamic load balancing have been suggested. We have addressed the issue of dynamic load balancing in terms of large amount of status information and heavy network traffic. We have presented algorithmic infrastructure for load balancing in a cluster of nodes and workstations as well as a server cluster. Various load indices for load measurement and parameters for performance measurement in a distributed system have been explored. Performances of various load balancing algorithms have been compared using these load indices and parameters. The impacts of load balancing on individual hosts and servers as well as the factors affecting load balancing performance are investigated. For achieving dynamic load balancing, we have presented both non-preemptive as well as preemptive process migration methodologies. We have compared two strategies and suggested parameters to calculate process migration cost. Performance studies with respect to web servers have been carried out and techniques for improving performance of a server cluster have been suggested. New challenges favouring further need of dynamic load balancing in Information Technology applications have also been highlighted.

Dynamic load balancing is found to significantly improve mean response time under unbalanced workload conditions. Load balancing is found to be very effective for small as well as large networks. All nodes, even underloaded nodes, are benefited from load balancing. Similarly all types of jobs get better average response time. Many of the above results are likely to be applicable in general to cluster nodes and workstations, network and web servers and even to networking devices like routers. Dynamic load balancing is cost effective, flexible and reliable strategy to support distributed scheduling even without modifying the system kernels or application programs and without deploying costly powerful servers and nodes.

This book is organized into eight chapters that reflect the stages of DLBs. In Chapter1, we have provided a general overview of the field along with introduction to related areas. We have also mentioned the objective of the proposed research work in this chapter. Rest of the thesis is organized as f+ollows:

Chapter 2 describes the process of load balancing in details. A number of load balancing techniques are defined and studied. The process of collecting the current state of the system, identifying underloaded and overloaded nodes, identifying processes to be transferred and mechanism of transferring processes from underloaded nodes to overloaded nodes has been described. The algorithms for selecting destination node have been described and compared. We also describe an overall methodology for carrying out DLB.

Chapter 3 considers an important issue of load estimation and performance measurement of load balancing algorithms. We have explored various parameters to measure load on the nodes in the system and evaluated various load balancing policies. We have also discussed architecture, implementation and performance evaluation of indices and parameters for capturing and distributing the load using DLB technique.

DLB can not be achieved without process migration. In **Chapter 4**, we discuss about this important phase in DLB. We have compared non-preemptive and preemptive migration methods and described framework for process migration. Technique of

transferring process address space from source node to destination node has been explored. We have discussed mechanism for calculating process migration cost and presented methodology for process migration.

A critical problem of performance improvement of network and web servers is highlighted in **Chapter 5.** In this chapter, we have studied the method of performance improvement in server cluster with the help of DLB. Web servers are facing the problem of constantly increasing network traffic and diverse load levels. It is not feasible to use a single powerful server. A cluster of replicated servers can be used and clients' requests can be distributed evenly among the servers in this cluster. We have described the problem of server load balancing and compared various load balancing policies for the cluster. The objective is to identify the algorithm that produces good overall performance.

In **Chapter 6**, we have identified new challenges posed by IT application, which are causing overload in the web based applications and necessitate the use of DLB. We have mainly raised the issues of public domain software, information overload, lack of optimization algorithms in routers, heterogeneity of servers and incompatibility problem of servers. Objective of this chapter is to explore the IT domains where the DLB techniques can be effectively implemented. To meet these challenges only few solutions are available and more solutions are possible. These problems can be tackled by the solutions provided in Chapter 2 to Chapter 5. Possible areas of research have also been mentioned.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION TO DISTRIBUTED COMPUTING ENVIRONMENT

1.1 PREAMBLE

Modern operating systems provide access to a large number of resources and facilities including communication and resource sharing. In distributed computing environments, effective scheduling of jobs and efficient resource utilization are critical issues. Hence, there is a great deal of work to be done by an operating system (OS hereafter) as far as scheduling of jobs on various processing elements is concerned. Our thesis addresses this important issue of processor scheduling in a distributed computing environment and emphasizes the need of dynamic load balancing (DLB hereafter) to solve the problem in a cost effective manner.

A conventional OS on a centralized computer manages all the systems' resources viz. processor, memory, devices and information. It provides all the related services like processor allocation, memory management, device management and information management to the users . It may also provide some simple communication services e.g. message passing and file transfer from one computer to other as shown in Fig 1.1.

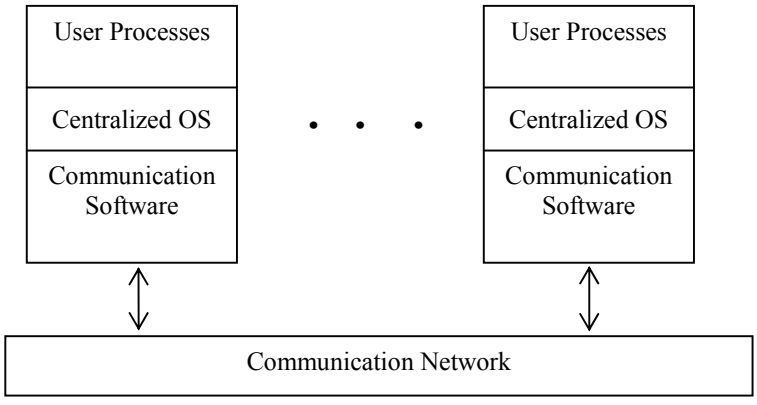

Fig. 1.1: Communication in centralized OS

Network operating system (NOS hereafter) is intended to provide users with global access to resources beyond simple communication available in a centralized system as shown in Fig. 1.2. Major limitation of NOS is that it does not take global control over the resources in the network. The NOS provides access to remote resources by using the facilities and mechanisms supported by local OS. Each computer in the network is managed locally, independent of the other computers. NOS merely provides communication infrastructure to the users. A user must have the knowledge of existence of a remote resource and privileges to access this resource. He must explicitly request NOS to provide connectivity to the remote resource.

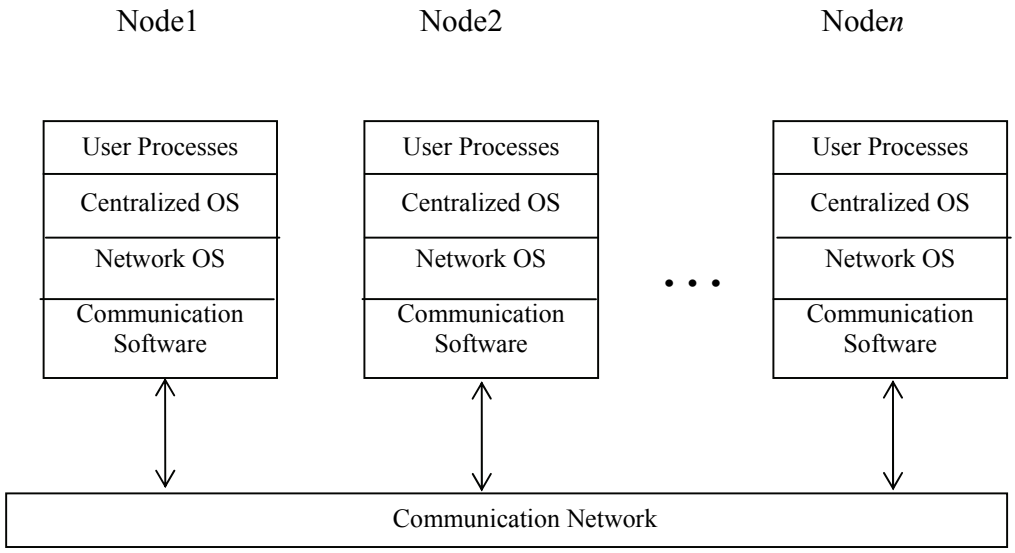

Fig. 1.2: A typical microkernel

Distributed operating system, on the other hand, considers the resources across multiple computer systems (including all resources on all sites) to be globally owned. The system controls and management are based on a single system-wide policy. Contrary to NOS, a distributed operating system is built on a bare machine, not just as an add-on to existing software. The distributed operating system determines the resource requirements of a process and decides how best to execute this process based on best guess or knowledge about the total system as shown in Fig. 1.3.

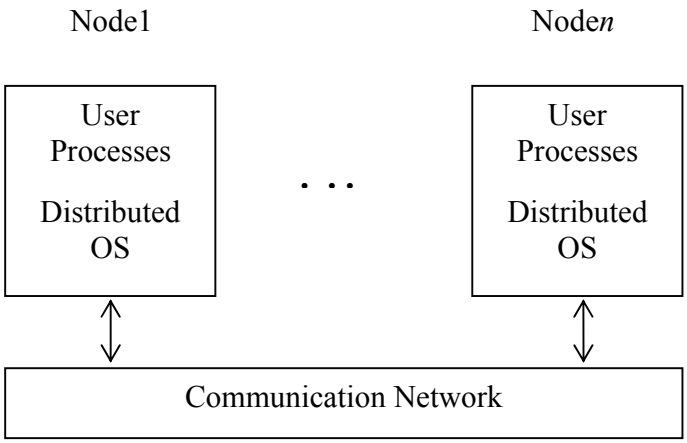

Fig. 1.3: Resource sharing in distributed OS

As the distributed operating system considers various resources available on a computer network to be globally owned, it provides resource sharing in a user transparent way. Thus, it makes the collection of computers to act like a virtual uni-processor system. The system is perceived as a whole and the existence of separate components of the system is concealed from the users and application programmers. A single system-wide policy manages the access to the resources effectively and efficiently. The system determines the processes' resource requirement and allocates the resources in a global way by collecting the information about current status of the total system. In this manner, the functionality of centralized system is made available in managing the resources in a computer network. For a given node, local and remote processes are executed in an identical way [Shiraji,1995].

In the distributed environment, kernel manages only basic resources like processor, memory and inter-process communication (IPC hereafter). It is implemented as microkernel architecture replicated on each node to derive functionality and features of a conventional monolithic kernel. As shown in Fig.1.4, it contains modules for process management, IPC, memory management; interrupt processing, system calls, traps and exceptions. Shared resources and services of the OS are provided by open servers that are

implemented above the micro-kernel layer. Local and remote resources are accessed in identical way without the knowledge of their location. DCS are open and scalable. They are capable of detection and recovery of faults. Fault tolerance is achieved with the help of hardware redundancy and software recovery [Petri,1995].

The services provided by the open servers are distributed scheduling, distributed shared memory, distributed file system, name services, remote procedure calls, network servers etc. Apart from the functionality of conventional OS in a centralized environment, a number of other services are provided in a distributed environment as shown in Fig. 1.4.

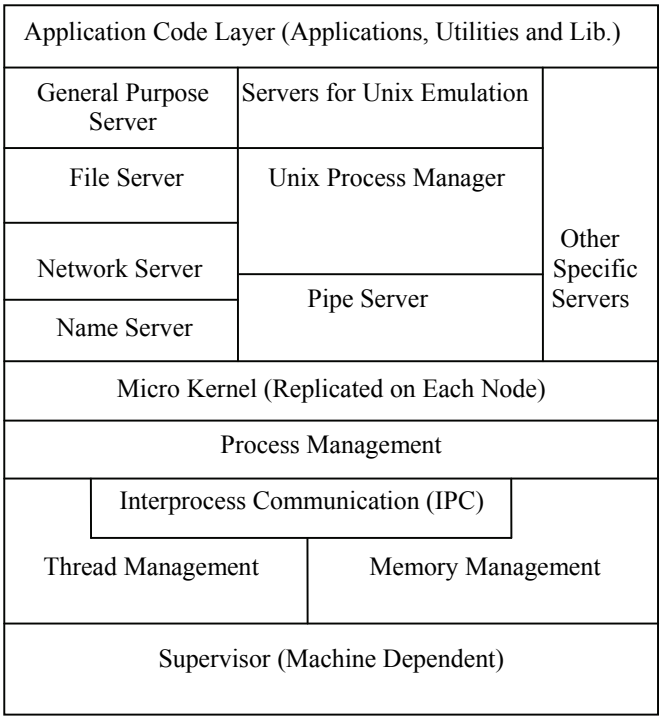

Fig. 1.4: A typical microkernel

The main services provided by distributed operating systems are**:**

1.1.1 Processor Allocation

Apart from processor scheduling on a specific node, process management tries to make optimal use of processing elements in a distributed environment and provides best possible services to a process by transferring the processes to remote processor, if necessary. Execution of a process is not bounded to local node. How best to execute a process using the resources available in the distributed environment depends on system's best guess about the current state of total system. If the node on which a process is waiting for execution has a long queue, then the process may be shifted to some other node which is either idle or having less number of processes. This ensures proper utilization of resources and improved response time of the process [Alnoso,1988; Ridge,1997].

Main design goals in processor scheduling are better resource utilization, improvement in response time of processes, minimizing network congestion and optimization of scheduling overheads. A number of techniques are used for distributed scheduling of processes. In task assignment approach, a process is treated as a collection of tasks which are scheduled on nodes by taking into consideration the cost of processing each task on every node and IPC cost between each pair of processors. An optimal weight is obtained by finding minimum weight cut-set using network flow algorithm or using heuristics if problem is NP-hard as in case of arbitrary number of processors [Sinha,2001; Barak,1993].

Multithreading can be used for implementing task assignment approach, where each task is organized as a thread. Peer threads can execute concurrently in multiprocessor as well as distributed computing systems (DCS hereafter). A thread is a unit of execution within a process and has its own program counter, register set and stack. However all threads share same address space. Threads also share open files, child processes, semaphores, signals and accounting information. Peer threads do not require protection among them. Threads may be supported at user level or at kernel level. User level threads can be implemented without OS support and allow users to use their own scheduling