**LESSON 3: DISTRIBUTED PROCESSING SYSTEMS**

**CONTENTS**

3.0 Aim and Objectives

1. Introduction
2. Pros and Cons of distributed Processing
3. Distributed Computing System Models
4. Distributed Operating System
5. Let us Sum UP
6. Lesson-End Activities
7. Points for Discussion

**3.0. AIM AND OBJECTIVES**

At the end of this Lesson, you will be able to understand

the advantages and Limitations of Distributed Processing, various types of Distributed Computing System Models, distributed Operating System

1. **INTRODUCTION**

The reasons behind the development of distributed systems were the availability of powerful microprocessors at low cost as well as significant advances in communication technology. The availability of powerful yet cheap microprocessors led to the development of powerful workstations that satisfy a single user’s needs. These powerful stand-alone workstations satisfy user need by providing such things as bit-mapped displays and visual interfaces, which traditional time-sharing mainframe systems do not support.

When a group of people works together, there is generally a need to communicate with each other, to share data, and to share expensive resources (such as high quality printers, disk drivers, etc). This requires interconnecting computers and resources. Designing such systems became feasible with the availability of cheap and powerful microprocessors, and advances in communication technology.

When a few powerful workstations are interconnected and can communicate with each other, the total computing power available in such a system can be enormous. Such a system generally only costs tens of thousands of dollars. On the other hand, if one tries

to obtain a single machine with the computing power equal to that of a network of workstations, the cost can be as high as a few million dollars. Hence, the main advantage of distributed system is that they have a decisive price/performance advantage over more traditional time-sharing systems.

**3.2. PROS AND CONS OF DISTRIBUTED PROCESSING**

**Resource sharing:** Since a computer can request a service from another computer bysending an appropriate request to it over the communication network, hardware and software resources can be shared among computers. For example, a printer, a compiler, a text processor, or a database at a computer can be shared with remote computers.

**Enhanced Performance**: A distributed computing system is capable of providing rapidresponse time and higher system throughput. This ability is mainly due to the fact that many tasks can be concurrently executed at different computers. Moreover, distributed systems can employ a load distributing technique to improve response time. In load distributing tasks at heavily loaded computers are transferred to lightly loaded computers, thereby reducing the time tasks wait before receiving service.

**Improved reliability and availability**: A distributed computing system providesimproved reliability and availability because a few components of the system can fail without affecting the availability of the rest of the system. Also, through the replication of data (e.g., files and directories) and services, distributed systems can be made fault tolerant. Services are processes that provide functionality (e.g., a file service provides file system management; a mail service provides an electronic mail facility).

**Modular expandability:** Distributed computing systems are inherently; amenable tomodular expansion because new hardware and software resources can be easily added without replacing the existing resources.

**3.3. DISTRIBUTED COMPUTING SYSTEM MODELS**

Various models are used for building distributed computing systems. These models can be broadly classified into five categories-minicomputer, workstation, workstation-workstations server processor-pool, and hybrid. They are briefly described below

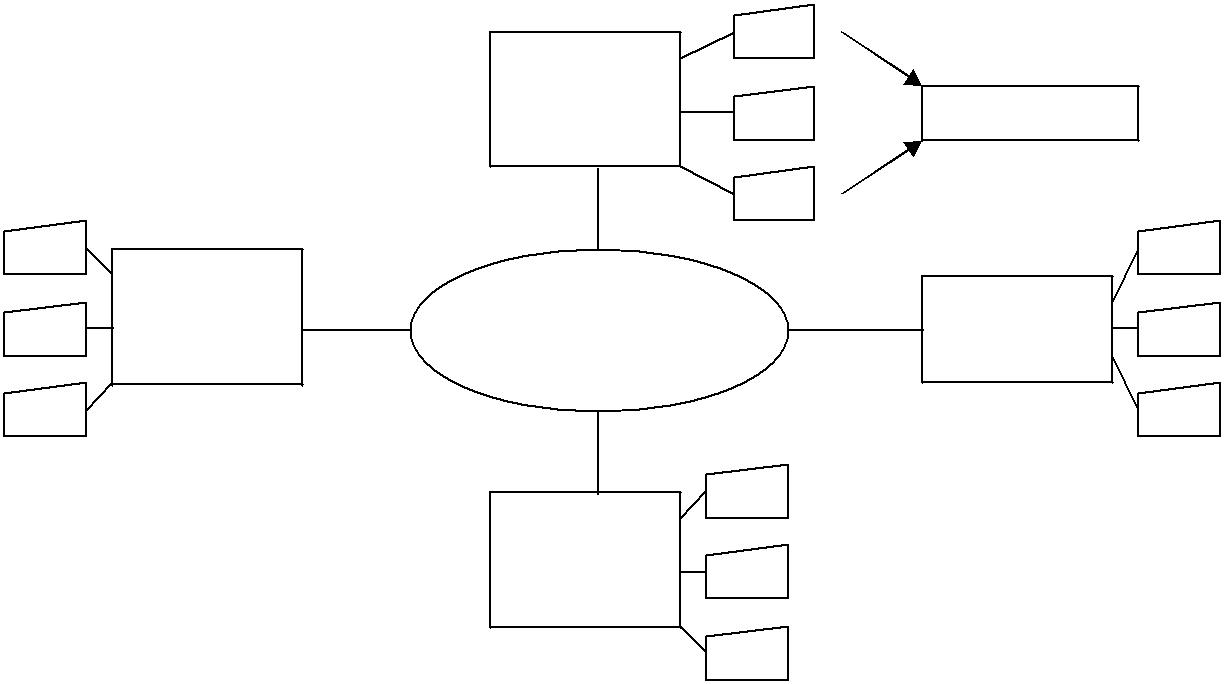
**Minicomputer Model**

The minicomputer model is a simple extension of the centralized time-sharing system. A distributed computing system based on this model consists of minicomputers (they may be large supercomputers as well) interconnected by a communication network. Each minicomputer usually has multiple users simultaneous logged on to it. For this, several interactive terminals are connected to each minicomputer each user is logged on to one specific minicomputer, with remote access to other Minicomputers. The network allows a user to access remote resources

that are available at some machine other than the one on to which the user is currently logged.

The minicomputer model may be used when resource sharing (such as sharing information databases of different types, with each type of database located on a different machine) with remote users is desired.

The early ARPANET is an example of a distributed computing system based minicomputer model.



|  |  |  |
| --- | --- | --- |
| Mini |  |  |
| computer | Terminals |  |
|  |  |

Mini

Communication Mini

computer

computer

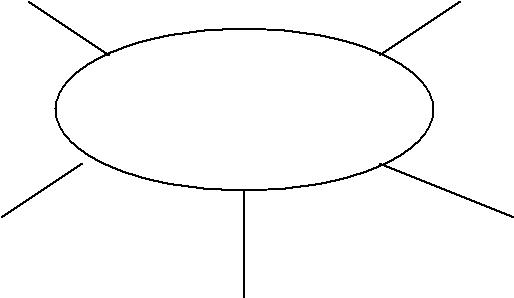
Mini computer

***Fig 3.1*** *Distributed-computing system based on the minicomputer model*

**Workstation Model**

The distributed computing system based on the workstation model consists of several workstations interconnected by a communication network. A company's office or a university department may have several workstations scattered throughout a building or campus, each workstation equipped with its own disk and serving as a single-user computer. It has been often found that in such an environment, at anyone time (especially at night), a significant proportion of the workstations are idle (not being used), resulting in the waste of large amounts of CPU time. Therefore, the idea of the workstation model is to interconnect all these workstations by a high-speed LAN so that idle workstations may be used to process jobs of users who are logged onto other workstations and do not have sufficient processing power at their own workstations to get their jobs processed efficiently.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Work - | |  |  |
| Work - |  | station | |  | Work - |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Work - |  | Communication | | Work - |
|  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Work - |  |  |  | Work - |  |
|  | Work- |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

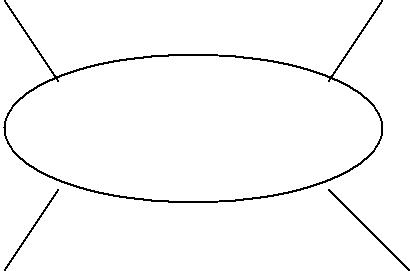
***Fig. 3.2*** *A distributed computing system based on the workstation model.*

**Workstation – Server Model**

The workstation model is a network of personal workstations, each with its own disk and a local file system. A workstation with its own local disk is usually called a diskful workstation and a workstation without a local disk is called a diskless workstation. With the proliferation of high-speed networks, diskless workstations have become more popular in network. Environments than diskful workstations, making the workstation-server model more popular than the workstation model for building distributed computing systems.

A distributed computing system based on the workstation server model consists of a few minicomputers and several workstations (most of which are diskless, but a few of which may be diskful) interconnected by a communication network.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Work - | |  |  |  |  |  |  |
|  |  |  |  |  | station | |  |  |  |  |  |  |
|  | Work - | | |  |  | Work - | | |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |
|  | station | | |  |  |  |  | station | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |  |  |  | |  |
| Work - | |  |  | Communication | | | |  |  | Work - | |  |
| station | |  |  |  |  | station | |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

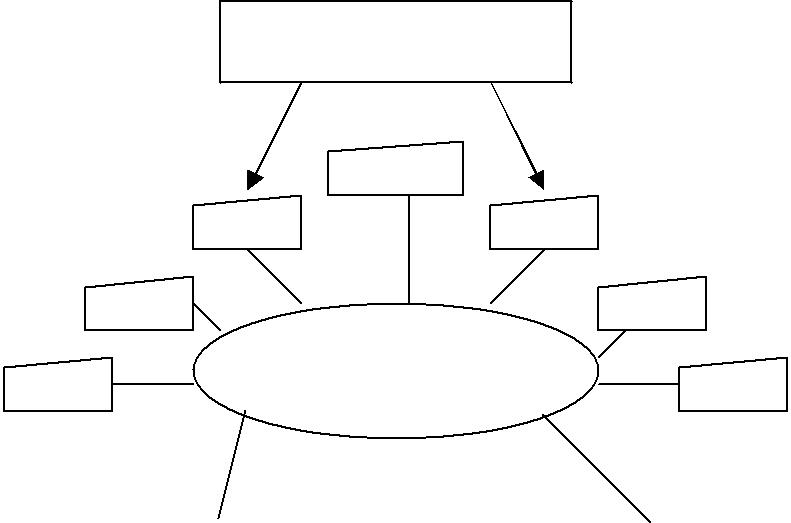


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Mini |  | Mini | |  | Mini |
| computer |  | computer | |  | comput |
| used as |  | used as | |  | er used |
| file |  | database | |  | as |
| server |  | server | |  | printer |
|  |  |  |  |  |  |

**Fig 3.3.** Distributed computing system based on the workstation server model.

**Processor – Pool Model**

The processor-pool model is based on the observation that most of the time a user does not need any computing power but once in a while he or she may need a very large amount of computing power for a short time (e.g., when recompiling a program consisting of a large number of files after changing a basic shared declaration). Therefore, unlike the workstation-server model in which a processor is allocated to each user, in the processor pool model the processors are pooled together to be shared by the users as needed. The pool of processors consists of a large number of microcomputers and minicomputers attached to the network. Each processor in the pool has its own memory to load and run a system program or an application program of the distributed computing system.



Terminals

Communication

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| Run |  |  |  |  |  |  |  |  |  | File |
| server |  |  |  |  |  |  |  |  |  | server |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

**Fig 3.4.** A distributed computing system based on the processor - pool model.

**Hybrid Model**

Out of the four models described above the workstation – server model, is the most widely used model for building distributed computing system. This is because large number of computer users only perform simple inter acting tasks such as editing jobs, sending electronic mails, and executing small programs. The workstation server model is ideal for such simple usage. However in a working environment that has groups of users who often perform jobs needing massive computation, the processor model is more attractive and suitable.

To combine the advantages of both the models a hybrid model may be used to build a distributed computing system. The hybrid model is based on the workstation server model but with the addition of pool of processors. The processors in the pool can be allocated dynamically for computations that are too large for workstations or that require several computers concurrently for efficient execution and gives a guaranteed response to interactive jobs by allowing them to be processed on local workstation of the users.

**3.4 DISTRIBUTED OPERATING SYSTEM**

An operating system as a program that control, the resources of a computer system and provides its users with an interface or virtual machine that is more convenient to use than the bare machine. According to this definition, the two primary tasks of an operating system are as follows

1. To present users with a virtual machine that is easier to program than the underlying hardware.
2. To manage the various resources of the system. This involves performing such tasks as keeping track of who is using which resource, granting resource requests, accounting for resource usage, and mediating conflicting requests from different programs and users

The operating systems commonly used for distributed computing systems can be broadly classified into two types-network operating systems and distributed operating systems. The three most important features commonly used to differentiate between these two types of operating systems are system image, autonomy, and fault tolerance capability. These features are explained below.

1. **System image.** The most important feature used to differentiate between the twotypes of operating systems is the image of the distributed computing system from the point of view of its users. In case of a network operating system, the users view the distributed computing system as a collection of distinct machines connected by a communication subsystem. A distributed operating system hides the existence of multiple computers and provides a single-system image to its users. That is, it makes a collection of networked machines act as a virtual uniprocessor.
2. **Autonomy.** In the case of a network operating system, each computer of thedistributed computing system has its own local operating system (the operating systems of different computers may be the same or different), and there is essentially no coordination at all among the computers except for the rule that when two processes of different computers communicate with each other, they must use a mutually agreed on communication protocol. With a distributed operating system, there is a single system wide operating system and each computer of the distributed computing system runs a part of this global operating system. The distributed operating system tightly interweaves all the computers of the distributed computing

system in the sense that they work in close cooperation with each other for the efficient and effective utilization of the various resources of the system.

1. **Fault tolerance capability.** A network operating system provides little or no faulttolerance capability in the sense that if 10% of the machines of the entire distributed computing system are down at any moment, at least 10% of the users are unable to continue with their work. On the other hand, with a distributed operating system, most of the users are normally unaffected by the failed machines and can continue to perform their work normally, with only a 10% loss in performance of the entire distributed computing system. Therefore, the fault tolerance capability of a distributed operating system is usually very high as compared to that of a network operating system

**3.5. LET US SUM UP**

Distributed systems are classified into three broad categories, namely, the minicomputer model, the workstation model, and the processor pool mode.

In the minicomputer model, the distributed system consists of several minicomputers (e.g., VAXs). Each computer supports multiple users and provides access to remote resources. The ratio of the number of processors to the number of users is normally less than one.

In the workstation model, the distributed system consists of a number of workstations (up to several thousand). Each user has a workstation at his disposal, where in general, all of the user’s work is performed. With the help of a distributed file system, a user can access data regardless of the location of the data or of the user’s workstation. The ratio of the number of processors to the number of users is normally one. The workstations are typically equipped with a powerful processor, memory, a bit-mapped display and some cases a math co-processor and local disk storage.

In the processor pool model, the ratio of the number of processors to the number of users is normally greater than one. This model attempts to allocate one ore more microprocessors according to the user’s needs. Once the processors assigned to a user complete their tasks, they return to the pool and await a new assignment. Amoeba is an experimental system that is a combination of the workstation and the processor pool models. In Amoeba, each user has a workstation where the user performs tasks that require a quick interactive response (such as editing). In addition to the workstation users have access to a pool of processors for running applications that require greater speed (such as parallel algorithms performing significant numerical computations).

1. **LESSON END ACTIVITIES** 
   1. Explain the various reasons for designing applications in Distributed Processing system

1. **POINTS FOR DISCUSSION**

1. Differentiate between Centralised approach and Fully Distributed Approach

**LESSON 4: LOADING FACTORS**

**CONTENTS**

4.0 Aim and Objectives

1. Introduction to Distributed Databases
2. Challenges of Distributed Data
3. Distributed Resource Management System
   1. Overview
   2. Main Concepts
   3. Evolution of DRMS
4. DRMS Responsibility

4.5.Let us Sum UP 4.6.Lesson-End Activities 4.7.Points for Discussion

**4.0. AIM AND OBJECTIVES**

At the end of this Lesson, you will be able to understand

the need of Distributed Database, challenges of Distributed Data,

distributed Resource Management System, and DRMS Responsibilities

1. **INTRODUCTION TO DISTRIBUTED DATABASES**

The assumption has been made up to this point that the data base under consideration is centrally managed, although of course the users might be geographically dispersed and various transactions might be executed concurrently. In actual fact many systems exist involving several different computers and a number of different databases located in a variety of different places. In such a circumstances one speaks of a distributed data base system. Possibly the best known operating distributed data base system is the world wide airline reservations system: each participating airline uses its own data base located near its particular headquarters location; common protocols are, however, used which control the operations when information involving more than one database is needed to answer a given query.

The usefulness of distributed database operations has become increasingly obvious because of the popularity of many small minicomputer systems. These systems can obviously be used to control local databases and to perform operations of interest in local environments. Furthermore, when data are need that is not locally available, the local computers can address request to a network of another machines and other databases located in various remote places.

**4.2. CHALLENGES OF DISTRIBUTED DATA**

A distributed data base environment complicates the systems organization and the resulting data base operations. A decision must first be made about the allocation of the files to the various locations, or nodes, of the data base network. A particular file could be kept in some unique, central place; alternatively allocating the various file portions to several different nodes could partition it. Finally, the file or certain file portion could be replicated by number of messages and request circulating from node to node assuming that the file portions of interest at a particular site are locally stored. When the data files are replicated, the message traffic between nodes may be substantially reduced. In fact, a tradeoff exists between the extra storage used by the data replication and the increased speed of operations resulting from the reduced communications load between the nodes.

In a distributed database system the need for the basic physical and logical data independence is expanded to include also location and replica transparency. Location transparency implies that the user programs are independent of the particular location of the files, while replica transparency extends the transparency to the use of an arbitrary number of copies.

Once a particular file environment is created, procedures must be available for executing the various transactions and furnishing results to the requesting parties. A given transaction might be run locally; alternatively, various remote points might be asked to carry out the operations followed by the routing of responses to the originating points. The latter strategy involves a goods deal of overhead in handling the message queues that may be formed at various points in the network.

It goes without saying that all operations must be carried out in a distributed environment in such a way that data integrity and consistency are maintained. This implies that special locking and update strategies must be used to ensure that all copies of a given database are properly updated. Specifically, all files must be locked before updating, the locks must be held until the end of a given transaction, and file alterations must be broadcast through the network to all replicas before the end of the transaction. Special transaction commit policies have been invented for this purpose in distributed systems. Specifically, a transaction coordinator is named for each transaction, and the coordinator alone is empowered to commit a given transaction after querying the participating sites concerning their individual readiness to commit.

It is obvious from what has been said that the distributed data base environment creates a host of complications. Because of the current widespread use of computer networks and the increasing availability of on-line access to computational facilities by a wide range of users, the organization and operations of distributed database systems have become popular areas of investigation for researchers in the computer field.

**4.3. DISTRIBUTED RESOURCE MANAGEMENT SYSTEM**

A **DRMS** is an enterprise software application that is in charge of unattended background executions, commonly known for historical reasons as batch processing.

Synonyms are **batch system**, **job scheduler**, and **Distributed Resource** **Manager** (DRM). Today's job schedulers typically provide a graphical user interface anda single point of control for definition and monitoring of background executions in a distributed network of computers. Increasingly job schedulers are required to orchestrate the integration of real-time business activities with traditional background IT processing, across different operating system platforms and business application environments.

**4.3.1. Overview**

Basic features expected of job scheduler software are:

Interfaces to define workflows and/or job dependencies Automatic submission of executions

Interfaces to monitor the executions

Priorities and/or queues to control the execution order of unrelated jobs

If software from a completely different area includes all or some of those features, this software is considered to have job scheduling capabilities.

Most operating system platforms such as Unix and Windows provide basic job scheduling capabilities, for example Cron. Many programs such as DBMS, backup, ERPs, and BPM also include relevant job scheduling capabilities. Operating System (OS) or point program supplied job scheduling will not usually provide the ability to schedule beyond a single OS instance or outside the remit of the specific program. Organizations needing to automate highly complex related and un-related IT workload will also be expecting more advanced features from a job scheduler, such as:

Real-time scheduling based on external, un-predictable events Automatic restart and recovery in event of failures

Alerting and notification to operations personnel Generation of incident reports

Audit trails for regulatory compliance purposes

These advanced capabilities can be written by in-house developers but are more often provided by solutions from suppliers that specialize in systems management software.

**4.3.2. Main concepts**

There are many concepts that are central to almost every job scheduler implementation and that are widely recognized with minimal variations:

Jobs

Dependencies Job Streams Users

Beyond the basic, single OS instance scheduling tools there are two major architectures that exist for Job Scheduling software.

Master/Agent architecture — the historic architecture for Job scheduling software. The Job Scheduling software is installed on a single machine (Master) while on production machines only a very small component (Agent) is installed that awaits commands from the Master, executes them, and returns the exit code back to the Master.

Cooperative architecture — a decentralized model where each machine is capable of helping with scheduling and can offload locally scheduled jobs to other cooperating machines. This enables dynamic workload balancing to maximize hardware resource utilization and high availability to ensure service delivery.

**4.3.3. Evolution of DRMS**

Job Scheduling has a long history. Job Schedulers are one of the major components of the IT infrastructure since the early mainframe systems. At first, stacks of punch cards were processed one after the other, hence the term “batch processing.”

From a historical point of view, we can distinguish two main eras about Job Schedulers:

1. The mainframe era

* 1. Job Control Language (JCL) on IBM mainframes. Initially based on JCL functionality to handle dependencies this era is typified by the development of sophisticated scheduling solutions forming part of the systems management and automation toolset on the mainframe.

1. The open systems era
   1. Modern schedulers on a variety of architectures and operating systems. With standard scheduling tools limited to such as Cron, the need for mainframe standard job schedulers has grown with the increased adoption of distributed computing environments.

In terms of the type of scheduling there are also distinct eras:

1. Batch processing - the traditional date and time based execution of background tasks based on a defined period during which resources were available for batch

processing (the batch window). In effect the original mainframe approach transposed onto the open systems environment.

* 1. Event-driven process automation - where background processes cannot be simply run at a defined time, either because the nature of the business demands that workload is based on the occurrence of external events (such as the arrival of an order from a customer or a stock update from a store branch) or because there is no / insufficient batch window.
  2. Service Oriented job scheduling - recent developments in Service Oriented Architecture (SOA) have seen a move towards deploying job scheduling as a reusable IT infrastructure service that can play a role in the integration of existing business application workload with new Web Services based real-time applications.

1. **DRMS RESPONSIBILITY**

Various schemes are used to decide which particular job to run. Parameters that might be considered include:

Job priority

Compute resource availability

License key if job is using licensed software Execution time allocated to user

Number of simultaneous jobs allowed for a user Estimated execution time

Elapsed execution time

Availability of peripheral devices Occurrence of prescribed events

1. **LET US SUM UP**

In this Lesson we discussed about Distributed Database Management System and where it is used. And we also discussed about Managing Distributed Resources and DRMS responsibility

1. **LESSON END ACTIVITIES** 
   1. List down various Loading factors with appropriate application example:
2. **POINTS FOR DISCUSSION**

1. Discuss various Distributed Resource Management System Funcitons