MODULE – 1

1. *What is Distributed System ?*

- a distributed computing system is basically a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network. 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 system.

2. *What is Tightly and loosely coupled systems?*

1. Tightly coupled systems. In these systems, there is a single systemwide primary memory (address space) that is shared by all the processors [Fig. l.1(a)]. If any processor writes, for example, the value 100 to the memory location x, any other processor subsequently reading from location x will get the value 100. Therefore, in these systems, any communication between the processors usually takes place through the shared memory.

2. Loosely coupled systems. In these systems, the processors do not share memory, and each processor has its own local memory [Fig. l.1(b)]. If a processor writes the value 100 to the memory location x, this write operation will only change the contents of its local memory and will not affect the contents of the memory of any other processor. Hence, if another processor reads the memory location x, it will get whatever value was there before in that location of its own local memory. In these systems, all physical communication between the processors is done by passing messages across the network that interconnects the processors.

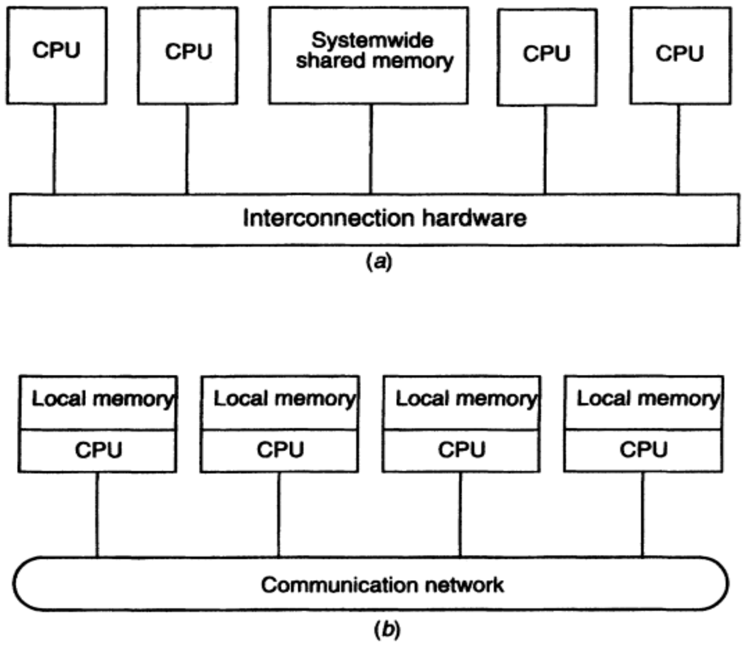

Fig. 1.1 Difference between tightly coupled and loosely coupled multiprocessor systems: (*a*) a tightly coupled multiprocessor system; (*b*) a loosely coupled multiprocessor system.

Usually, tightly coupled systems are referred to as parallel processing systems, and loosely coupled systems are referred to as distributed computing systems, or simply distributed systems.

3. ***Distribution Transparency:***

- The eight forms of transparency identified by the International Standards Organization's Reference Model for Open Distributed Processing [ISO 1992] are **access transparency, location transparency, replication transparency, failure transparency, migration transparency, concurrency transparency, performance transparency, and scaling transparency**.
- **Access Transparency :** Access transparency means that users should not need or be able to recognize whether a resource (hardware or software) is remote or local. This implies that the distributed operating system should allow users to access remote resources in the same way as local resources. That is, the user interface, which takes the form of a set of system calls, should not distinguish between local and remote resources, and it should be the responsibility of the distributed operating system to locate the resources and to arrange for servicing user requests in a user-transparent manner.
- ***Location Transparency:***

  The two main aspects of location transparency are as follows:

  1. Name transparency: This refers to the fact that the name of a resource (hardware or software) should not reveal any hint as to the physical location of the resource. That is, the name of a resource should be independent of the physical connectivity or topology of the system or the current location of the resource. Furthermore, such resources, which are capable of being moved from one node to another in a distributed system (such as a file), must be allowed to move without having their names changed. Therefore, resource names must be unique systemwide.

  2. User-mobility: This refers to the fact that no matter which machine a user is logged onto, he or she should be able to access a resource with the same name. That is, the user should not be required to use different names to access the same resource from two different nodes of the system. In a distributed system that supports user mobility, users can freely log on to any machine in the system and access any resource without making any extra effort. Both name transparency and user mobility requirements call for a systemwide, global resource naming facility.

- ***Replication transparency:*** Replication transparency is a form of transparency in distributed computing that refers to the ability of a distributed system to hide the fact that a resource or service is replicated across multiple nodes in the network. Replication is a technique used in distributed systems to improve performance, availability, and fault tolerance by duplicating resources or services across multiple nodes. However, the complexity of replication can introduce additional challenges and requirements for the system.

To achieve replication transparency, a distributed system must provide mechanisms for handling replication, such as consistency protocols, update propagation, and load balancing. The system must also ensure that the replicas are consistent and up-to-date, and that any updates or modifications are propagated to all replicas in a timely and efficient manner.

Examples of replicated resources or services include replicated databases, replicated files, and replicated web servers. Replication transparency can improve the usability, reliability, and scalability of a distributed system by providing a transparent and efficient way to handle replication.

- ***Failure Transparency:*** Failure transparency deals with masking from the users' partial failures in the system, such as a communication link failure, a machine failure, or a storage device crash. A distributed operating system having failure transparency property will continue to function, perhaps in a degraded form, in the face of partial failures. For example, suppose the file service of a distributed operating system is to be made failure transparent.
- ***Migration Transparency:*** For better performance, reliability, and security reasons, an object that is capable of being moved (such as a process or a file) is often migrated from one node to another in a distributed system. The aim of migration transparency is to ensure that the movement of the object is handled automatically by the system in a user-transparent manner.

  Three important issues in achieving this goal are as follows:

  1. Migration decisions such as which object is to be moved from where to where should be made automatically by the system.

  2. Migration of an object from one node to another should not require any change in its name.

  3. When the migrating object is a process, the inter process communication mechanism should ensure that a message sent to the migrating process reaches it without the need for the sender process to resend it if the receiver process moves to another node before the message is received.

- ***Concurrency Transparency:*** In a distributed system, multiple users who are spatially separated use the system concurrently. In such a situation, it is economical to share the system resources (hardware or software) among the concurrently executing user processes. However, since the number of available resources in a computing system is restricted, one user process must necessarily influence the action of other concurrently executing user processes, as it competes for resources.

  For providing concurrency transparency, the resource sharing mechanisms of the distributed operating system must have the following four properties:

  1. An event-ordering property ensures that all access requests to various system resources are properly ordered to provide a consistent view to all users of the system.

  2. A mutual-exclusion property ensures that at any time at most one process accesses a shared resource, which must not be used simultaneously by multiple processes if program operation is to be correct.

  3. A no-starvation property ensures that if every process that is granted a resource, which must not be used simultaneously by multiple processes, eventually releases it, every request for that resource is eventually granted.

  4. A no-deadlock property ensures that a situation will never occur in which competing processes prevent their mutual progress even though no single one requests more resources than available in the system.

- ***Performance Transparency:*** The aim of performance transparency is to allow the system to be automatically reconfigured to improve performance, as loads vary dynamically in the system. As far as practicable, a situation in which one processor of the system is overloaded with jobs while another processor is idle should not be allowed to occur. That is, the processing capability of the system should be uniformly distributed among the currently available jobs in the system

- ***Scaling Transparency***: The aim of scaling transparency is to allow the system to expand in scale without disrupting the activities of the users. This requirement calls for open-system architecture and the use of scalable algorithms for designing the distributed operating system components.

Distributed Shared Memory :

DSM is basically an abstraction that integrates the local memory of different machines in a network environment into a single logical entity shared by cooperating processes executing on multiple sites. The shared memory itself exists only virtually. Application programs can use it in the same way as a traditional virtual memory, except, of course, that processes using it can run on different machines in parallel. Due to the virtual existence of the shared memory, DSM is sometimes also referred to as Distributed Shared Virtual Memory (DSVM).

In a DSM system, each node has its own physical memory, which is mapped to a global address space that is shared among all the nodes in the system. When a process on one node accesses a memory location in the global address space, the DSM system ensures that the data is fetched from the appropriate node and returned to the requesting process. This allows the process to access shared data as if it were stored in local memory, without the need for explicit communication with other nodes.

There are two basic approaches to implementing DSM systems: software-based and hardware-based. Software-based DSM systems rely on software libraries and runtime systems to manage the distributed memory and handle communication between nodes. Hardware-based DSM systems use specialized hardware to manage the distributed memory and provide high-speed communication between nodes.

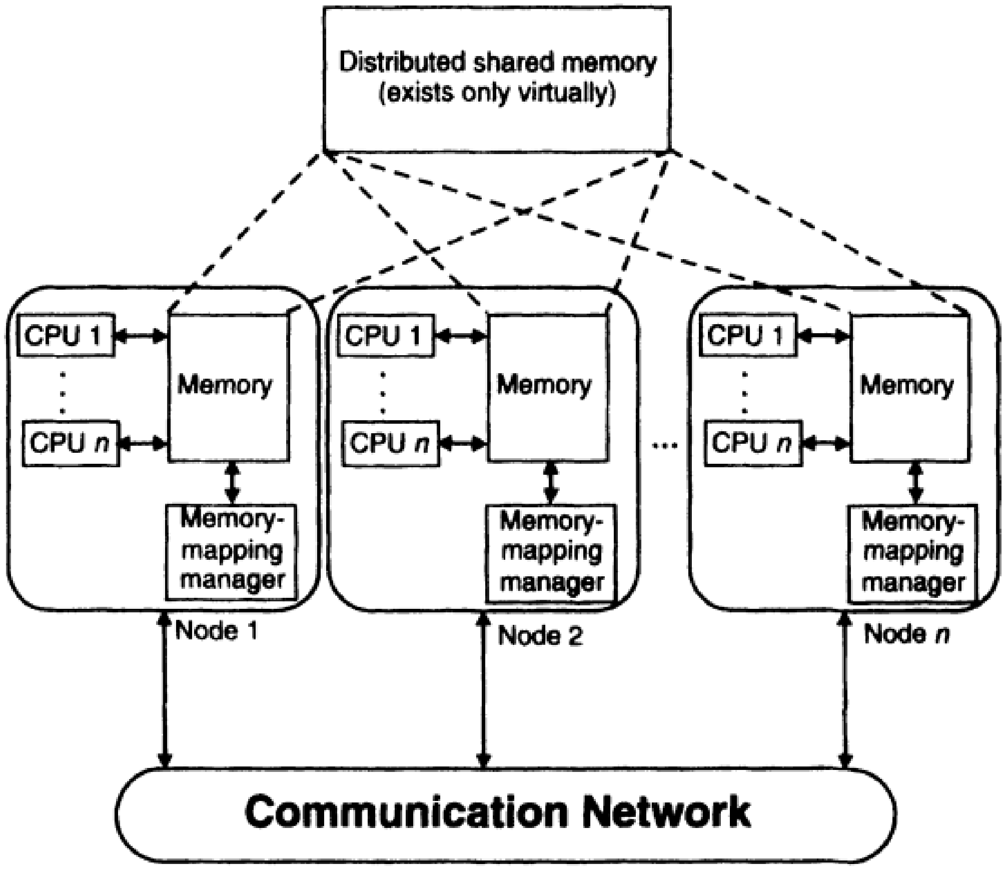

Fig. 5.1 Distributed shared memory (DSM).

Question: Differentiate among the following types of operating systems by defining their essential properties: (a) Time sharing (b) Parallel processing (c) Network (d) Distributed

Ans:-

(a) Time sharing operating systems are designed to allow multiple users to share a single computer system simultaneously. The essential property of time-sharing operating systems is that they provide a mechanism for each user to interact with the system as if they had the entire system to themselves. The operating system manages the allocation of system resources such as CPU time, memory, and peripherals between multiple users and processes.

(b) Parallel processing operating systems are designed to take advantage of multiple processors or cores in a single computer system to perform computations simultaneously. The essential property of parallel processing operating systems is that they provide mechanisms for breaking down a computation into smaller, independent parts that can be executed in parallel on multiple processors. The operating system manages the allocation of tasks to the processors and coordinates their interactions to achieve high performance.

(c) Network operating systems are designed to manage the resources and services of a computer network. The essential property of network operating systems is that they provide mechanisms for sharing resources such as files, printers, and applications across multiple computers connected by a network. The operating system manages the allocation of network resources and provides security, authentication, and other services required for efficient and reliable network communication.

(d) Distributed operating systems are designed to manage resources and services across multiple computers connected by a network, with the goal of providing a single, coherent view of the system to users. The essential property of distributed operating systems is that they provide mechanisms for coordinating the interactions of multiple computers to provide a unified view of resources and services. The operating system manages the allocation of resources and provides mechanisms for communication, coordination, and fault tolerance to ensure that the system behaves as if it were a single, integrated system.

Question: In what respect are distributed computing systems better than parallel processing systems? Give examples of three applications for which distributed computing systems will be more suitable than parallel processing systems.

Ans:-

Distributed computing systems are designed to manage resources and services across multiple computers connected by a network, while parallel processing systems are designed to take advantage of multiple processors or cores in a single computer system to perform computations simultaneously. Distributed computing systems have several advantages over parallel processing systems, including:

1. Scalability: Distributed computing systems can scale to a much larger number of computers than parallel processing systems, allowing them to handle larger computations and workloads.
2. Fault tolerance: Distributed computing systems are designed to be fault-tolerant, meaning that they can continue to operate even if one or more computers in the network fail. Parallel processing systems may not have the same level of fault tolerance.
3. Geographic distribution: Distributed computing systems can handle computations that involve data or resources that are geographically distributed, such as large-scale data analytics or simulations.

Examples of applications for which distributed computing systems are more suitable than parallel processing systems include:

a. Large-scale data processing: Distributed computing systems such as Apache Hadoop or Apache Spark are designed to handle large-scale data processing tasks by distributing the data and computations across a large number of computers.
b. Cloud computing: Cloud computing services such as Amazon Web Services or Microsoft Azure are built on distributed computing systems, allowing customers to deploy and manage applications across a distributed network of computers.
c. Internet of Things (IoT): IoT applications typically involve a large number of devices that generate data and need to be managed and analyzed in real-time. Distributed computing systems are well-suited to handle the processing and analysis of this data across a network of devices and computers.