CS 3551 DISTRIBUTED COMPUTING

UNIT I INTRODUCTION 8

Introduction: Definition-Relation to Computer System Components – Motivation – Message - Passing Systems versus Shared Memory Systems – Primitives for Distributed Communication – Synchronous versus Asynchronous Executions – Design Issues and Challenges; A Model of Distributed Computations: A Distributed Program – A Model of Distributed Executions – Models of Communication Networks – Global State of a Distributed System.

UNIT II LOGICAL TIME AND GLOBAL STATE 10

Logical Time: Physical Clock Synchronization: NTP – A Framework for a System of Logical Clocks – Scalar Time – Vector Time; Message Ordering and Group Communication: Message Ordering Paradigms – Asynchronous Execution with Synchronous Communication – Synchronous Program Order on Asynchronous System – Group Communication – Causal Order – Total Order; Global State and Snapshot Recording Algorithms: Introduction – System Model and Definitions – Snapshot Algorithms for FIFO Channels.

UNIT III DISTRIBUTED MUTEX AND DEADLOCK 10

Distributed Mutual exclusion Algorithms: Introduction – Preliminaries – Lamport's algorithm – Ricart- Agrawala's Algorithm — Token-Based Algorithms – Suzuki-Kasami's Broadcast Algorithm; Deadlock Detection in Distributed Systems: Introduction – System Model – Preliminaries – Models of Deadlocks – Chandy-Misra-Haas Algorithm for the AND model and OR Model.

UNIT IV CONSENSUS AND RECOVERY 10

Consensus and Agreement Algorithms: Problem Definition – Overview of Results – Agreement in a Failure-Free System(Synchronous and Asynchronous) – Agreement in Synchronous Systems with Failures; Checkpointing and Rollback Recovery: Introduction – Background and Definitions – Issues in Failure Recovery – Checkpoint-based Recovery – Coordinated Checkpointing Algorithm -- Algorithm for Asynchronous Checkpointing and Recovery

UNIT V CLOUD COMPUTING 7

Definition of Cloud Computing – Characteristics of Cloud – Cloud Deployment Models – Cloud Service Models – Driving Factors and Challenges of Cloud – Virtualization – Load Balancing – Scalability and Elasticity – Replication – Monitoring – Cloud Services and Platforms: Compute Services – Storage Services – Application Services

Book

TEXT BOOKS

1. Kshemkalyani Ajay D, Mukesh Singhal, “Distributed Computing: Principles, Algorithms and Systems”, Cambridge Press, 2011.

LIKE

COMMENT

SHARE

SUBSCRIBE

Unit 1

UNIT I INTRODUCTION **8**

Introduction: Definition-Relation to Computer System Components – Motivation – Message - Passing Systems versus Shared Memory Systems – Primitives for Distributed Communication – Synchronous versus Asynchronous Executions – Design Issues and Challenges; A Model of Distributed Computations: A Distributed Program – A Model of Distributed Executions – Models of Communication Networks – Global State of a Distributed System.

LIKE

COMMENT

SHARE

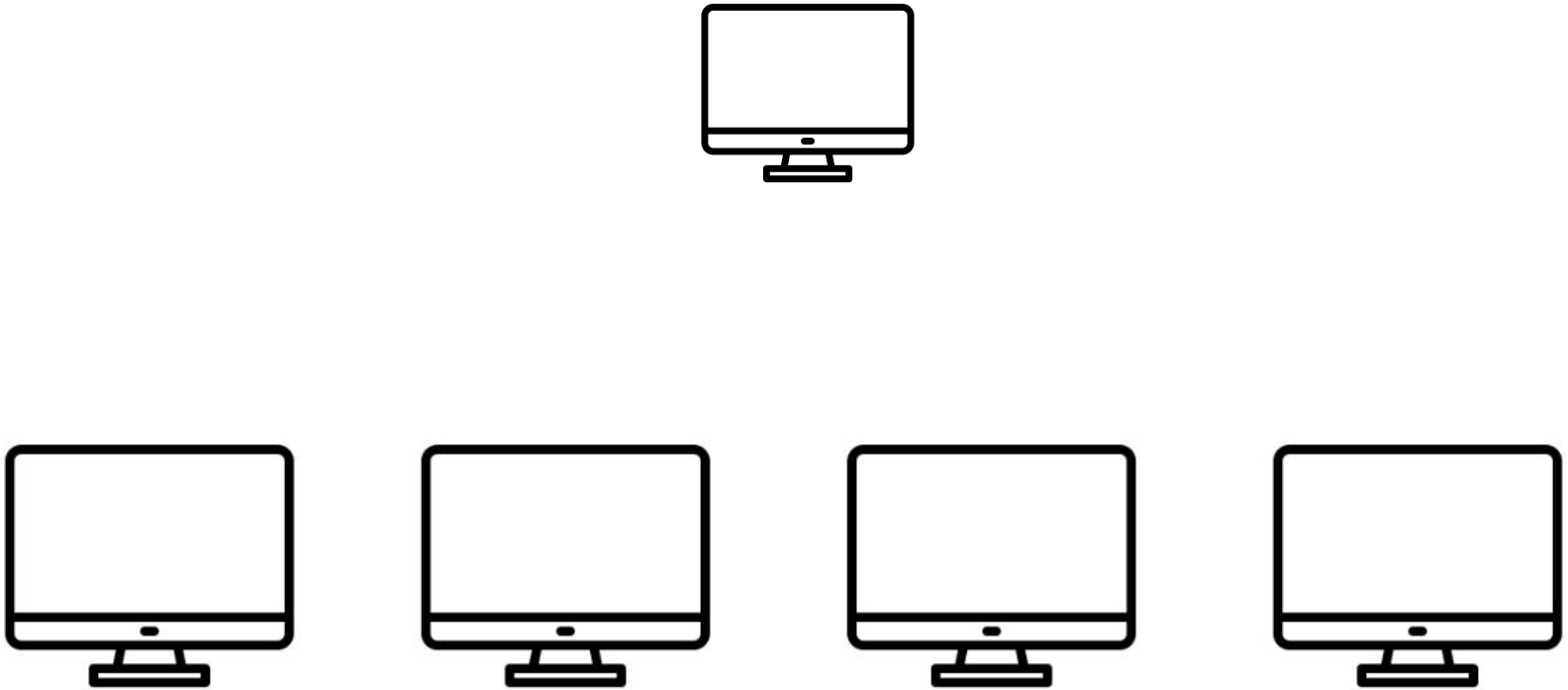
SUBSCRIBE

What is distributed Computing?

HOD wants to assign paper correction (500 papers in total)

Scenario 1 Assign paper to single faculty Arun Arun => 500 papers => 5 days	**Scenario 2** Assign paper to 5 faculties Arun => 100 papers => Gopal => 100 papers => Ashif => 100 papers => Hari => 100 papers => Murali => 100 papers =>

Eg - Online banking transaction for 10 million users



Eg 2 Image Rendering - Resize, Filter, Color, Effects

1. **Divide the Image**: The large image is divided into smaller, equally sized portions or "chunks." Each chunk represents a portion of the image that can be processed independently.
2. **Distribute Chunks**: These image chunks are distributed to multiple computers within a network. Each computer receives one or more chunks to work on.
3. **Parallel Processing**: Each computer processes its assigned image chunk independently and applies the desired effects simultaneously. This means that multiple parts of the image are being worked on at the same time, greatly reducing the time required for rendering.
4. **Combine Results**: Once all the computers have completed processing their respective chunks, the processed image chunks are combined or stitched together to create the final, fully rendered image.

Applications Area of Distributed System:

- **Finance and Commerce:** Amazon, eBay, Online Banking, E-Commerce websites.
- **Information Society:** Search Engines, Wikipedia, Social Networking, Cloud Computing.
- **Cloud Technologies:** AWS, Salesforce, Microsoft Azure, SAP.
- **Entertainment:** Online Gaming, Music, youtube.
- **Healthcare:** Online patient records, Health Informatics.
- **Education:** E-learning.
- **Transport and logistics:** GPS, Google Maps.
- **Environment Management:** Sensor technologies.

Definition

- Autonomous processors communicating over a communication network
- Some characteristics
  - No common physical clock
  - No shared memory
  - Geographical seperation
  - Autonomy and heterogeneity

Characteristics

- **No common physical clock** This is an important assumption because it introduces the element of "distribution" in the system and gives rise to the inherent asynchrony amongst the processors.
- **No shared memory** This is a key feature that requires message-passing for communication. This feature implies the absence of the common physical clock.

  It may be noted that a distributed system may still provide the abstraction of a common address space via the distributed shared memory abstraction. Several aspects of shared memory multiprocessor systems have also been studied in the distributed computing literature.
- **Geographical separation** The geographically wider apart that the processors are, the more representative is the system of a distributed system. However, it is not necessary for the processors to be on a wide-area network (WAN). Recently, the network/cluster of workstations (NOW/COW) configuration connecting processors on a LAN is also being increasingly regarded as a small distributed system. This NOW configuration is becoming popular because of the low-cost high-speed off-the-shelf processors now available. The Google search engine is based on the NOW architecture.
- **Autonomy and heterogeneity** The processors are "loosely coupled" in that they have different speeds and each can be running a different operating system. They are usually not part of a dedicated system, but cooperate with one another by offering services or solving a problem jointly.

LIKE

COMMENT

SHARE

SUBSCRIBE

Relation to Computer System Components

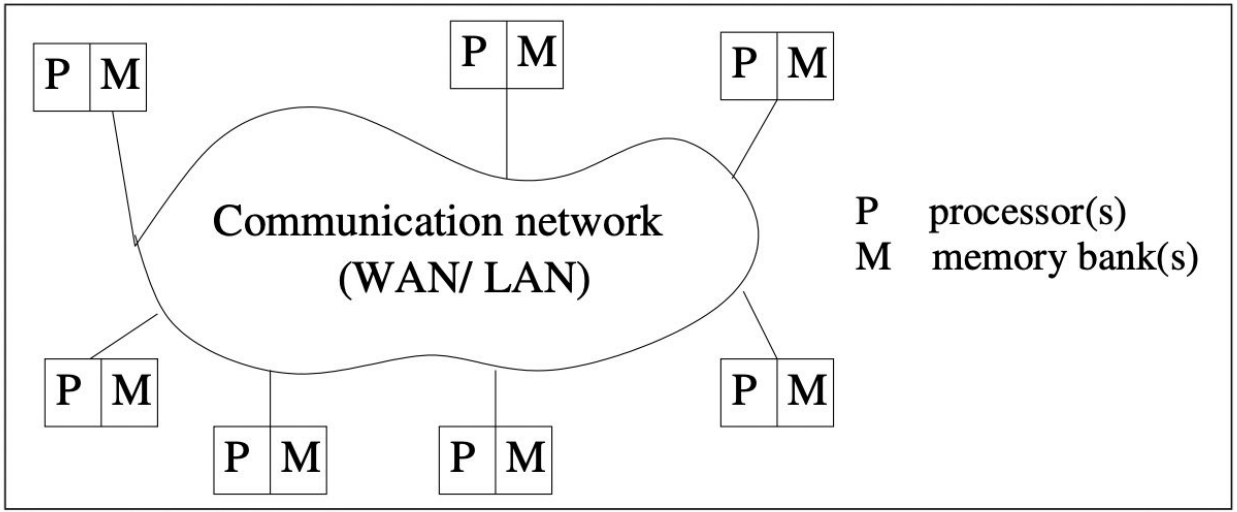


Figure 1.1: A distributed system connects processors by a communication network.

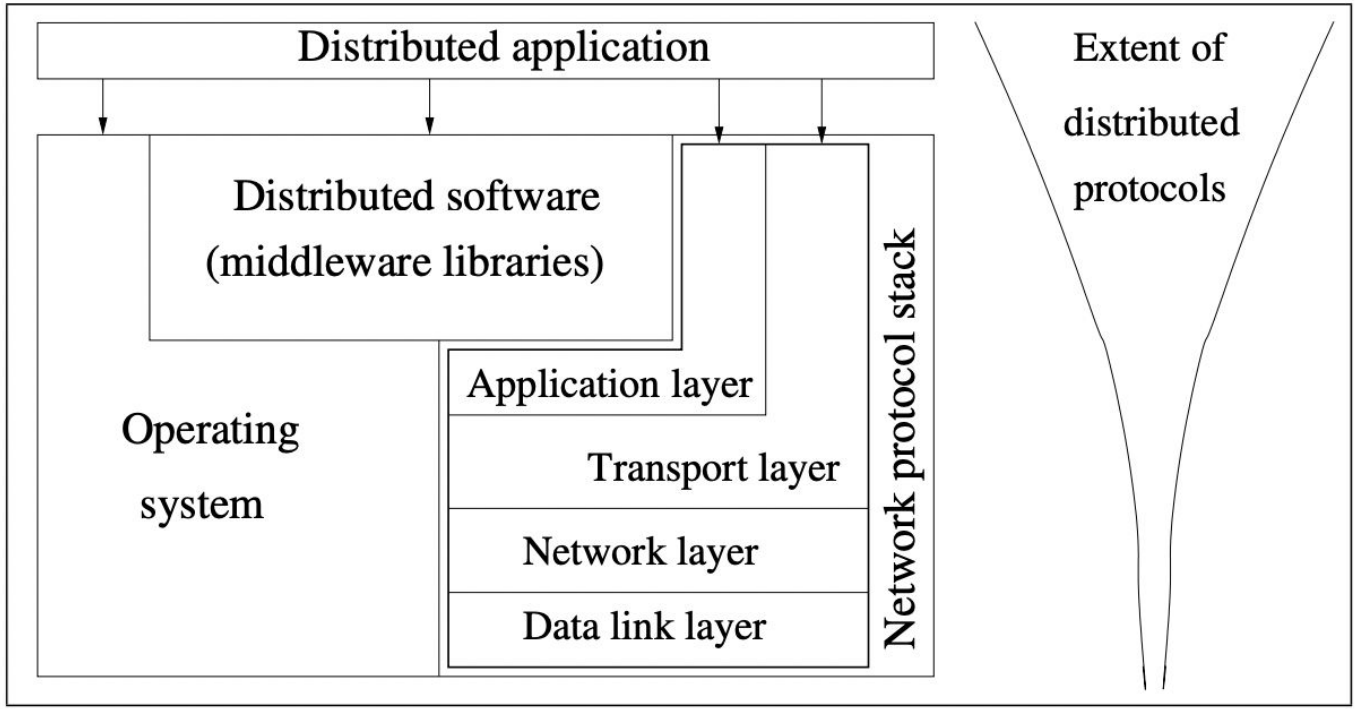


Figure 1.2: Interaction of the software components at each process.

Key Points

- The distributed software is also termed as middleware.
- A distributed execution is the execution of processes across the distributed system to collaboratively achieve a common goal. An execution is also sometimes termed a computation or a run.
- The middleware is the distributed software that drives the distributed system, while providing transparency of heterogeneity at the platform level.
- Middleware layer does not contain the traditional application layer functions of the network protocol stack, such as http, mail, ftp, and telnet.
- Various primitives and calls to functions defined in various libraries of the middleware layer are embedded in the user program code.
- There exist several libraries to choose from to invoke primitives for the more common functions – such as reliable and ordered multicasting – of the middleware layer.
- There are several standards such as Object Management Group's (OMG) common object request broker architecture (CORBA) [36], and the remote procedure call (RPC) mechanism.

LIKE

COMMENT

SHARE

SUBSCRIBE

Motivation/ Benefit of Distributed Computing

- Inherently distributed computation
- Resource sharing
- Access to remote resources
- Increased performance/cost ratio
- Reliability
  - availability, integrity, fault-tolerance
- Scalability
- Modularity and incremental expandability

Key Points

1. **Inherently distributed computations**
    a. In many applications such as money transfer in banking, or reaching consensus among parties that are geographically distant, the computation is inherently distributed.
2. **Resource sharing**
    a. Resources such as peripherals, complete data sets in databases, special libraries, as well as data (variable/files) cannot be fully replicated at all the sites because it is often neither practical nor cost-effective.
    b. Further, they cannot be placed at a single site because access to that site might prove to be a bottleneck.
    c. Therefore, such resources are typically distributed across the system. For example, distributed databases such as DB2 partition the data sets across several servers, in addition to replicating them at a few sites for rapid access as well as reliability.
3. **Access to geographically remote data and resources**
    a. In many scenarios, the data cannot be replicated at every site participating in the distributed execution because it may be too large or too sensitive to be replicated. For example, payroll data within a multinational corporation is both too large and too sensitive to be replicated at every branch office/site. It is therefore stored at a central server which can be queried by branch offices. Similarly, special resources such as supercomputers exist only in certain locations, and to access such supercomputers, users need to log in remotely.
4. **Enhanced reliability**
    a. A distributed system has the inherent potential to provide increased reliability because of the possibility of replicating resources and executions, as well as the reality that geographically distributed resources are not likely to crash/malfunction at the same time under normal circumstances.
    b. Reliability entails several aspects:
        i. • availability, i.e., the resource should be accessible at all times; •
        ii. integrity, i.e., the value/state of the resource should be correct, in the face of concurrent access from multiple processors, as per the semantics expected by the application;
        iii. • fault-tolerance, i.e., the ability to recover from system failures, where such failures may be defined to occur in one of many failure models.

5. **Increased performance/cost ratio**
    a. By resource sharing and accessing geographically remote data and resources, the performance/cost ratio is increased.
    b. Although higher throughput has not necessarily been the main objective behind using a distributed system, nevertheless, any task can be partitioned across the various computers in the distributed system.
    c. Such a configuration provides a better performance/cost ratio than using special parallel machines.
6. **Scalability**
    a. As the processors are usually connected by a wide-area network, adding more processors does not pose a direct bottleneck for the communication network.
7. **Modularity and incremental expandability**
    a. Heterogeneous processors may be easily added into the system without affecting the performance, as long as those processors are running the same middleware algorithms.
    b. Similarly, existing processors may be easily replaced by other processors.

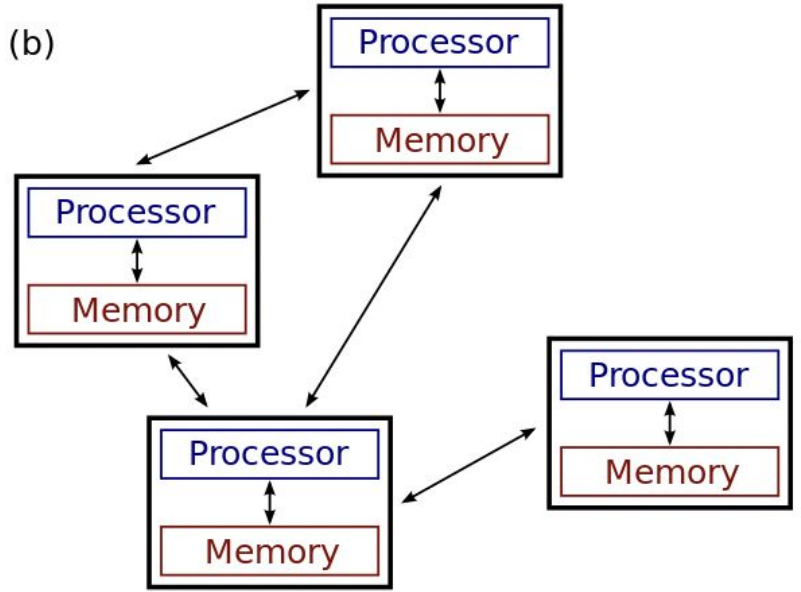
LIKE

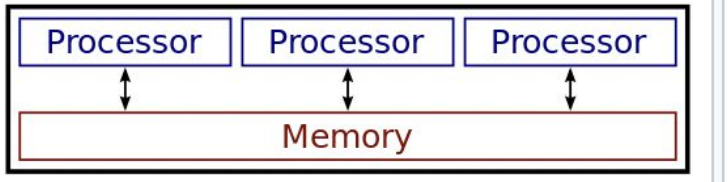
COMMENT

SHARE

SUBSCRIBE

Distributed Vs Parallel Computing





Difference #1: Number of Computers Required

Parallel computing typically requires one computer with multiple processors. Distributed computing, on the other hand, involves several autonomous (and often geographically separate and/or distant) computer systems working on divided tasks.

Difference #2: Scalability

Parallel computing systems are less scalable than distributed computing systems because the memory of a single computer can only handle so many processors at once. A distributed computing system can always scale with additional computers.

Difference #3: Memory

In parallel computing, all processors share the same memory and the processors communicate with each other with the help of this shared memory. Distributed computing systems, on the other hand, have their own memory and processors.

Difference #4: Synchronization

In parallel computing, all processors share a single master clock for synchronization, while distributed computing systems use synchronization algorithms.

Difference #5: Usage

Parallel computing is used to increase computer performance and for scientific computing, while distributed computing is used to share resources and improve scalability.

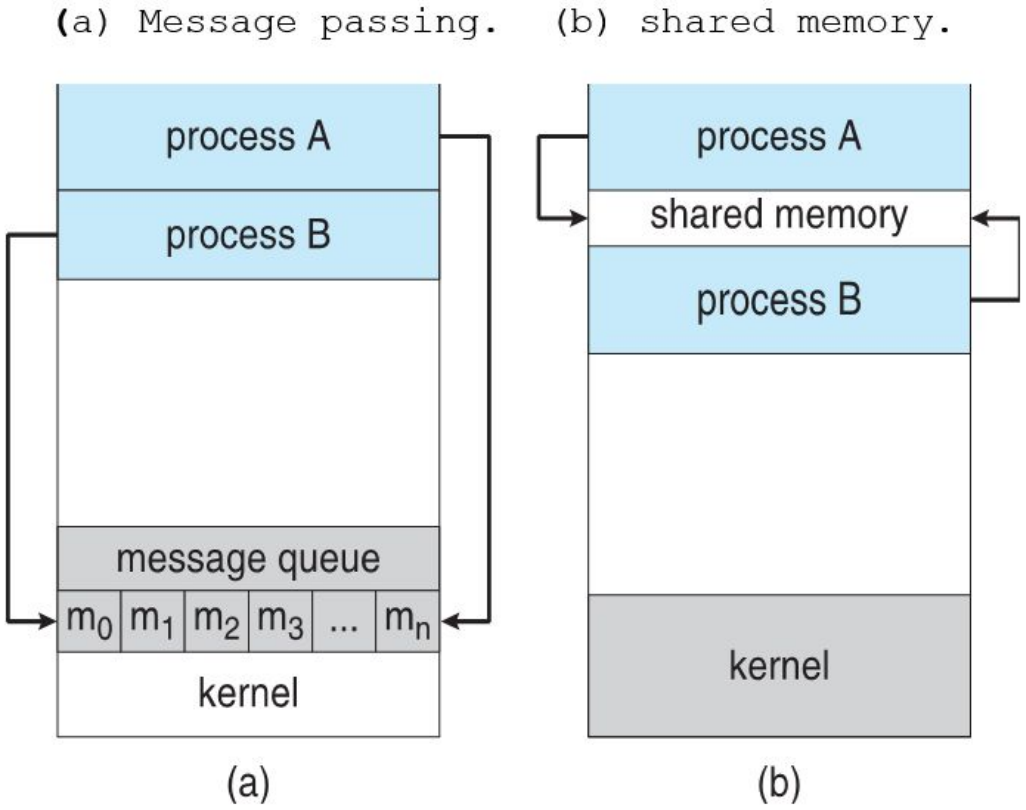
LIKE

COMMENT

SHARE

SUBSCRIBE

Message Passing vs Shared Memory



Key Points

- Shared memory systems are those in which there is a (common) shared address space throughout the system.
- Communication among processors takes place via shared data variables, and control variables for synchronization among the processors.
- Semaphores and monitors that were originally designed for shared memory uniprocessors and multiprocessors are examples of how synchronization can be achieved in shared memory systems.
- All multicomputer (NUMA as well as message-passing) systems that do not have a shared address space provided by the underlying architecture and hardware necessarily communicate by message passing.
- For a distributed system, this abstraction is called distributed shared memory. Implementing this abstraction has a certain cost but it simplifies the task of the application programmer.

Emulating MP in SM

1. The shared address space can be partitioned into disjoint parts, one part being assigned to each processor.
2. “Send” and “receive” operations can be implemented by writing to and reading from the destination/sender processor’s address space, respectively. Specifically, a separate location can be reserved as the mailbox for each ordered pair of processes.
3. A P_i–P_j message-passing can be emulated by a write by P_i to the mailbox and then a read by P_j from the mailbox.
4. The write and read operations need to be controlled using synchronization primitives to inform the receiver/sender after the data has been sent/received.

Emulating SM in MP

1. This involves the use of “send” and “receive” operations for “write” and “read” operations.
2. Each shared location can be modeled as a separate process;
   a. “write” to a shared location is emulated by sending an update message to the corresponding owner process;
   b. a “read” to a shared location is emulated by sending a query message to the owner process.
3. the latencies involved in read and write operations may be high even when using shared memory emulation
4. An application can of course use a combination of shared memory and message-passing.
5. In a MIMD message-passing multicomputer system, each “processor” may be a tightly coupled multiprocessor system with shared memory. Within the multiprocessor system, the processors communicate via shared memory. Between two computers, the communication is by message passing

LIKE

COMMENT

SHARE

SUBSCRIBE