Q1. Define Distributed System. What are the advantages of Distributed System?

- **Definition**: A distributed system is a collection of independent computer that appears to its user as a coherent (logical) system.
- A distributed system consists of concurrent (parallel) processes accessing distributed resources.
- A distributed system consists of autonomous computers linked by a computer n/w and equipped with a distributed system s/w.
- Resources are shared through message passing in a network environment that may be unreliable and contain un-trusted components.

**Characteristics of Distributed System**

- Difference between various computers and the ways in which they communicate are mostly hidden from users.
- User and application can interact with a distributed system in a consistent (reliable) way, regardless where and when interaction takes place.
- It should be relatively easy to expand.
- DS will normally be available continuously, if some parts may be temporarily out of order.

**Advantages of Distributed System**

- Inherently distributed applications
- Information sharing among geographically distributed users
- Resource sharing: hardware and software
- Better price performance ratio
- Shorter response time and higher throughput
- Higher reliability and availability against component failure
- Extensibility and incremental growth
- Better flexibility in meeting user’s needs

Q2. What is the difference between Shared Memory Architecture & Distributed Memory Architecture? Explain it with diagrams.

**Shared memory architecture:**

```
PE1   PE2   PE3   PE4
      Internet Network
      Memory shared by all processing elements
```

- Shared memory systems form a major category of multiprocessors. In this category, all processors share a global memory (See above fig.). Communication between tasks running on deferent processors is performed through writing to and reading from the global memory.
- All interprocessor coordination and synchronization is also accomplished via the global memory.
  Two main problems need to be addressed when designing a shared memory system:
• Performance degradation due to contention. Performance degradation might happen when multiple processors are trying to access the shared memory simultaneously.
• A typical design might use caches to solve the contention problem.
• Coherence problems. Having multiple copies of data, spread throughout the caches, might lead to a coherence problem.
• The copies in the caches are coherent if they are all equal to the same value. However, if one of the processors writes over the value of one of the copies, then the copy becomes inconsistent because it no longer equals the value of the other copies.
• Scalability remains the main drawback of a shared memory system.

Distributed memory architecture:

- In this architecture each processor has its own local memory and all the processing is done locally.
- All systems are interconnected through a LAN.
- No sharing among the address spaces.
- Easy to implement, built through commodity hardware with the interconnection being established through a standard Ethernet or ATM switches.
- If a processor wants to access a remote memory location, the information will be sent as a message transfer through the communication network.

Q3. Write short note on Comparison of different Distributed Computing Models.

There are three computing models as below
• Workstation Model
• Workstation-server Model
• Processor-pool Model

The comparison between workstation model and workstation-server model are following
• It is economically more viable to use a few high-end costly servers and more diskless workstations. Diskless workstations are easier to maintain than diskful ones.
• In a workstation-server model, the request–response protocol indicates that the client does not get burdened and the process migration becomes unnecessary.
• The user also has the flexibility of changing his workstation.
• Workstation is connected in a suitable network configuration using the star topology.
• Workstation-server model consists of multiple workstations coupled with powerful servers with extra hardware to store the file systems and other software.
• Workstation-server model is suitable for sharing the resources between different systems in a modular fashion.
• Consider a brief comparison of the other two processor-pool model and workstation-server model are following
• The processor-pool model uses computing resources more effectively, all the resources of the system being available to the currently working users.
• Workstation-server model offer services only to individual clients.
• Processor-pool model is that some of its processors can work as servers, if the load has increased or if more users are logged in and demanding new services.
• A hybrid model of a distributed system may be built by combining the features of the workstation-server model and the processor-pool model.


Workstation-server Model:
• The Workstation-server model consists of multiple workstations coupled with powerful servers with extra hardware to store the file systems and other software like databases.

In workstation-server model workstations can be distributed among users.
• User logs onto a workstation called home workstation.
• Normal computation activities required by the user’s process are performed at the user’s home workstation.
• Request for services are sent to server and return the result of request processing to the user’s workstation.
• The file system being common to all the workstation, a user can actually log into any one of them and carry on his work.
• The important files can be stored in the main server which is better maintained.
• User’s process need to be migrated to the server machine.
• The V-system is a distributed computing system based on the workstation-server model.
Processor-pool Model:
- The processor-pool model consists of multiple processors and a group of workstations.
- The model is based on the observation that most of the time a user does not need any computing power.
- In this model the process is pooled together to be shared by the users as needed.
- The Processor-pool of process consists of large microcomputers and minicomputers attach to the n/w.
- Each processor has its own memory to load and run.
- The processors in the pool have no terminals attached directly to them, and the user access the system from terminals that are attached to the n/w via special device.
- The processors in the pool have no terminals attached directly to them, and the user access the system from terminals that are attached to the n/w via special device.
- An appropriate number of processors are temporarily assigned to his or her job by the run server.

Q5. What are the various issues related to Design Distributed System?

The issues in designing a distributed system include:
- Transparency
- Reliability
- Scalability
- Flexibility
- Performance
- Security
Transparency:

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource is replicated</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
</tbody>
</table>

Reliability:

- Reliability in terms of data means that data should be available without any errors.
- Distributed system where multiple processors are available and the system become reliable.

Scalability:

- The techniques used to handle scalability issues in distributed systems are hide communication latencies, hide distribution, and hide replication.

Characteristics of decentralized algorithms:

- No machine has complete information about the system state.
- Machines make decisions based only on local information.
- Failure of one machine does not ruin the algorithm.
- There is no implicit assumption that a global clock exists.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single on-line telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Doing routing based on complete information</td>
</tr>
</tbody>
</table>

The three techniques commonly used for scaling are discussed next.

- Hide communication latencies
- Hide distribution
- Hide replication
Flexibility:
- To achieve flexibility is to take a decision whether to use monolithic kernel or microkernel on each machine.
- Kernel is the central control, which provides the basic system facilities.
- The major functions of the kernel are: memory management, process management and resource management.
- To achieve flexibility, we can use two types of approach: **Monolithic kernel approach** & **Microkernel approach**
- Monolithic kernels use the ‘kernel does it all’ approach with all functionalities provided by the kernel irrespective of whether all machines use it or not.
• Micro kernels use the minimalist, modular approach with accessibility to other services as needed.

Performance:
• Response time, throughput, system utilization, bandwidth requirements

Security:
Security consists of three main aspects, namely,
• Confidentiality: Protection against unauthorized access.
• Integrity: Protection of data against corruption.
• Availability: Protection against failure and always being accessible.

Q6. Explain the different types of Transparencies used in Distributed System.

Types of Transparencies are includes
• Access transparency
• Location transparency
• Migration transparency
• Replication transparency
• Concurrency transparency
• Persistence transparency

There are many types of transparency:
Access transparency - Regardless of how resource access and representation has to be performed on each individual computing entity, the users of a distributed system should always access resources in a single, uniform way.
Location transparency - Users of a distributed system should not have to be aware of where a resource is physically located.
Migration transparency - Users should not be aware of whether a resource or computing entity possesses the ability to move to a different physical or logical location.
Relocation transparency - Should a resource move while in use, this should not be noticeable to the end user.
Replication transparency - If a resource is replicated among several locations, it should appear to the user as a single resource.
Concurrency transparency - While multiple users may compete for and share a single resource, this should not be apparent to any of them.
Failure transparency - Always try to hide any failure and recovery of computing entities and resources.
Persistence transparency - Whether a resource lies in volatile or permanent memory should make no difference to the user.
Security transparency - Negotiation of cryptographically secure access of resources must require a minimum of user intervention, or users will circumvent the security in preference of productivity

The degree to which these properties can or should be achieved may vary widely. Not every system can or should hide everything from its users. For instance, due to the existence of a fixed and finite speed of light there will always be more latency on accessing resources distant from the user. If
one expects real-time interaction with the distributed system, this may be very noticeable.

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
<tr>
<td>Persistence</td>
<td>Hide whether a (software) resource is in memory or on disk</td>
</tr>
</tbody>
</table>

Q7. **Write the different Addressing Techniques of Client-Server Model.**

The client-server addressing techniques include machine addressing, process addressing, and the name server technique.

**Machine addressing:**
- In this addressing method, the client sends the address as a part of the message.
- This method works well if only one process running on the server machine.
- If multiple processes running on the server, the process ID should be sent as a part of the message.

![Diagram](a) machine addressing

![Diagram](b) process addressing

**Process addressing:**
- In process addressing machine ID and process ID is used for addressing.
- Client kernel uses the machine ID to locate the current machine.
• Server kernel uses the process ID to locate the process on that machine.
• This method is not transparent because the user knows the location of the server.

Name server technique:
• Use extra machine to map ASCII level names to machine address. The ASCII strings are embedded in the program. This special server is called a name server.
• Hardwire the machine number into the client server.
• Processes pick random address and the machine address is located by broadcast method.
• Use a two-level naming scheme with mapping carried out by the name server.

Q8. **Draw the Diagram of Three-Tiered Client-Server Architecture and briefly explain it.**
• The client server architecture depends on how the client server application is distributed across the user interface, the application, and the database tier between the client and the server.
• The client machine contains only the terminal dependent part of the user interface, applications get remote control over presentation of data.
• The user interface is completely on the server side, and the front end does no processing except for presenting the application interface.
• A part of the application can be moved to the front end.
• The client is a PC connected across the network to a distributed file system and database. This arrangement is called the three tiered architecture.
• The processing is done on a separate server or is distributed across several machines.

![Diagram of Three-Tiered Client-Server Architecture](image_url)
Q9. Write a short note on:

(a) World Wide Web 1.0

(b) World Wide Web 2.0

World Wide Web 1.0:
- WWW is an excellent example of resource sharing using network computers.
- The World Wide Web is the universe of network accessible information—embodiment of human knowledge.
- Standards or protocols for addressing systems and exchanging data known as the transmission control protocol/internet protocol (TCP/IP).
- Internets were able to exchange electronic mail using file transfer protocol (FTP) to exchange data.
- Internet is widely used for applications and operations like web mail, chat, photos, social networks, planning trips, forums, and blogs, etc.
- Though continuous evolutions publish and access resources and services across the internet.
- The web is based on three main standard technological components:
  - The Hypertext Markup Language (HTML) is a language for specifying the content and layout of pages to display by web browser.
  - Uniform Resource locators (URL) that identify documents and other resources stored as part of the web.
  - Hypertext Transfer Protocol (HTTP) a client-server architecture with standard rules for interaction by which browser and other clients fetch documents and other resources from web server.

World Wide Web 2.0:
- The web server based model aims to facilitate communication, secure information sharing, interoperability, and collaboration on the World Wide Web. For example, web 2.0
- Web 2.0 encapsulates the idea of proliferation of interconnectivity and interactivity of web delivered content.
- For example, Google’s server infrastructure is divided into various types, each assigned a different purpose.
- In this execution consists of sent by users and then format the result into an HTML page.
- eBay, Dodgeball, AdSense, etc., products use Web 2.0.

Q10. How does a Distributed Operating System differ from a Network Operating System?
- Network operating system is a software architecture used to build a distributed system from a network of workstations connected by high-speed networks.
- A distributed operating system enables a distributed system to behave like a virtual uniprocessor, even though the system operates on a collection of machines. This is possible using various characteristics like:
  - Interprocess communication
  - Uniform process management mechanism
• Uniform and visible file system
• Identical kernel implementation
• Local control machines
• Scheduling issues
• A modern operating system functions on and within a specific computer although it may collect, use or deliver data outside of its self.
• A distributed system utilites computing power from two or more computer systems. A task may be performed within this computer or on another within the group, or may be split such that portions of a smaller task can be performed on different computers.
• Remote computing uses messages over a network to request tasks to be undertaken. These messages pass the necessary parameters for the task and on completion messages return the results. It’s as if the computers email each other with a request and an answer.
• The distinction "modern operating system" versus "distributed operating system" makes little sense, though. Distributed operating systems are in many ways more complex and sophisticated. With high-end computing demands, distributed operating systems continue to be a very active field of research and development; in many ways making these systems more modern than the mainstream operating systems. Those, instead of "modern operating" system, might better be called "desktop operating system," "single machine operating system," or "mainstream operating system."
• Finally, it should be noted that almost any current operating system is a mix between a distributed and a single-machine operating system, including some of the features required for removing.
Q11. Write a short note on:

(a) Wireless LAN

(b) Internetworks

(c) WAP

(a) Wireless LAN:
- WLAN technology allows computer network devices to have network connectivity without the use of wires i.e. using radio waves.
- This network uses Bluetooth, infrared link or low power radio network operating at a bandwidth of 1-2 mbps over a distance of 10m.
- Users are connected to the LAN through a wireless client adapter.
- In figure 2-4 top level, wireless networks can be classified as wireless LAN, wireless MAN, and wireless WAN.
- These wireless LANs classified as PAN and business LANs.
- Wireless MANs are wireless local loops.
- The wireless WANs are classified as cellular networks, satellite systems and paging networks.

(b) Internetworks:
- In figure 2-5 internetworks consists of several interconnected networks providing data communication facilities.
• The heterogeneous nature of protocols, technologies like Ethernet, FDDI, and token ring, and the methods for interconnection are hidden from the users.
• The major components of an internetwork are hardware interconnection by routers, gateways and software layer which support addressing and data transmission.

![Internetwork with FDDI, Ethernet, and Token ring](image)

**Figure 2-5** Internetwork with FDDI, Ethernet, and Token ring

(c) Wireless application protocol (WAP):
• WAPs are used to implement mobile computing platforms, as shown in figure 2-6.
• In top level mobile connects to the server though a physical wireless network which includes antennas, transceivers, base stations, cellular networks, and satellites.
• Web server architecture consists of server, which has three layers the application layer, middleware services, and the lowermost layer local platform services and transport services.
• Example of mobile computing application includes wireless messaging (SMS), mobile ecommerce and its variants.

![Mobile computing platform](image)

**Figure 2-6** Mobile computing platform
Q12. Discuss the architecture of OSI reference model.

- The seven layer of the OSI reference model are shown in figure 2-7
- Every layer performs independent network function dealing with some specific aspect of communication.
- In figure 2-8 are demonstrated the communication between data link layer
- For example communicates with the network layer and the physical layer of system A and the data link layer of system B.

<table>
<thead>
<tr>
<th>Table 2-1 Functions of the OSI Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Layer</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
<tr>
<td>Physical Layer</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Data Link Layer</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Network Layer</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Q13. Compare TCP and UDP transport protocols.

- TCP provides connection oriented reliable byte stream service.
- UDP is a connectionless unreliable datagram service.
- TCP handles the establishment and termination of connection, sequence of data, end to end reliability with error control and end to end flow control.
- UDP is similar to IP with two additional features of process addressing and checksum to verify datagram contents.
- Both protocol use 16 bit port number header attached to data.
- When packet reaches the destination host, the transport layer protocol dispatches it to the specified port.
- TCP provides a reliable service to the application programmer.
- TCP is complex and is complex to use, but is powerful. It works well with various applications such as short interactive messages or large bulk transfer.
- TCP is even more complex than we have seen as it also implements congestion control, a topic that we will study in a follow-up lecture.
- UDP service is message oriented
- If a UDP message is larger than MTU, then fragmentation occurs at the IP layer
- Most applications use TCP rather than UDP, as this avoids re-inventing error recovery in every application

Q14. Describe the ATM protocol reference model.

- This model consists of three layers: physical layer, ATM layer, AAL-ATM adaption layer.
ATM layer functions

<table>
<thead>
<tr>
<th>Layer</th>
<th>Feature</th>
</tr>
</thead>
</table>
| Physical Layer | - Manages transmission and reception of electrical or optical signals between two devices.  
|                | - Maps the cells into time-division multiplexed frames, which can be sent over the physical medium.  
|                | - Physical Medium Dependent layer defines the data transmission speed for transmission across a physical link. (PMD)  
|                | - Transmission Convergence layer defines a protocol for encoding and decoding cell data for passing over the physical medium. (TC)  
|                | - Uses Synchronous Digital Hierarchy (SDH) or FDDI standards.             |
| ATM Layer      | - Handles multiplexing and switching of cells and maintains Quality of Service (QoS).  
|                | - Manages cell flow, sequencing, cell payload type marking, differentiation and cell reception and header validation.  
|                | - The ATM cell and frame formats are explained later.                     |
| AAL Layer      | - Adapts the user traffic into the cell-format.                          |
|                | - On the sender side, data of various types with different characteristics is split into 48 byte-cells and conversely on the receiving end, the cells are reassembled into the original form by the AAL layer.  
|                | - Service classes A, B, C, and D are defined based on:  
|                |   - the timing between the source and destination  
|                |   - the bit rate of the source  
|                |   - the connection mode                                                  |

- Standard A:
  - The User-Network Interface (UNI) standard contains the Generic Flow Control field (GFC) and Virtual Path Identifier (VPI) one-octet field.

- Standard B:
  - The Network-Network Interface (NNI) has one and a half octets long VPI but does not contain the GFC field.

- In figure 2-18 ATM frame consists of a frame header and a group of cells, Which can be in use or idle.
The two ATM layer standards are shown in figure 2-19.

- Versatile message transfer protocol (VMTP):
  - VMTP supports request/response behavior.
  - Provides transparency and group communication, selective retransmission mechanism, rate based control flow.
  - Supports execution of non idempotent operations, and conditional delivery of real time communication.
  - This connectionless protocol provides end to end datagram delivery.
  - In figure 2-21 packets divided into 16kb segments, which are split into 512 kb blocks.
  - Each block consists of data and a 32 bit mask, delivery mask field, indicating data segment in the packet.

Q15. Briefly explain FLIP & VMTP.

**Versatile message transfer protocol (VMTP):**

- Provides request/response behavior.
- Provides transparency and group communication, selective retransmission mechanism, rate based control flow.
- Supports execution of non idempotent operations, and conditional delivery of real time communication.
- This connectionless protocol provides end to end datagram delivery.
- In figure 2-21 packets divided into 16kb segments, which are split into 512 kb blocks.
- Each block consists of data and a 32 bit mask, delivery mask field, indicating data segment in the packet.
The VMTP request response behavior is depicted in figure 2-22. This technique provides feedback when transmission rate is high.

Rate based flow control mechanism: VMTP users burst protocol and sends blocks of packet to the receivers with spaced inter packet gaps.

This technique helps in matching data transmission and reception speeds.

For example money transfers from a bank account. Server does not respond to such request by executing them more than once.

Conditional delivery for real time communication: message should be delivered only if the server can immediately process it.

Fast local internet protocol (FLIP):

- Connectionless protocol
- The main features of FLIP are transparency, security, group management, network management, and efficient client server based communication.
- Transparency to support process migration: entities called the NSAP (network service access point) are identified with 64 bit location independent IDs, one per process.
- FLIP routes the packets the source NSAP to the destination NSAP based on the data stored in the routing table.
- FLIP automatically handles dynamic network configuration changes.
- FLIP provides two methods for secure delivery. First method, sensitive messages are marked with security bit.
- Second method, sensitive message routed over the untrusted network are marked unsafe bit.
Q16. What is Message Passing? How does the Message Passing Approach differ from Share Memory Approach?

Definition: A type of communication between processes. Message passing is a form of communication used in parallel programming and object-oriented programming. Communications are completed by the sending of messages (functions, signals and data packets) to recipients.

- The two basic methods for information sharing are:
  - Message passing approach
  - Shared memory approach
- In the message passing technique, the sender and the receiver processes communicate with each other using send and receive primitives.
- Two processes exchange a message by sender’s address space to the receiver’s address space. As shown in figure 3-2.
- Sender kernel message passing between process A and process B. In the shared memory approach, sending and receiving process share a common memory area.
Q17. How the Message Format is required in IPC? Explain the components of Message Format with diagram.

- A message consists of data and control information, formatted by the sender so that it is meaningful to the server. The sender process adds control information to the data block with headers and trailers before transmission. So message format is required for IPC.

![Message Format Diagram](image)

**Figure 3-4 Components of an IPC message**

- The header information of a message block consists of the following element, as shown in figure 3-4.
- Address: it uniquely identifies the sender and receiver processes in the network by specifying the source and destination address.
- Sequence number: it is a message identifier which takes care of duplicate and lost message in case of system failure.
- Structural information: it has a two parts. Type part specifies the message carries data or only the pointers to data. Length part specifies length of the variable sized data in bytes.
In a message passing system, the sender is responsible for formatting the contents of the message. Receiver is fully aware about the message format used by the server in the communication process.

Q18. Compare Blocking and Non-Blocking Primitives of IPC.

A blocking primitive’s invocation blocks the execution of the invoker. The receiver blocks itself until a message arrives. Invocation of a non blocking primitive does not block the execution of the invoker. Non blocking operation return straight away and allow the sub program to continue execution. In fig 3-9 the blocking and non blocking primitives for client server communication. The client sends a message, it makes a trap to the kernel and blocks itself. Message passing primitives like blocking and non blocking operation are used to achieve synchronization during IPC.
Q19. Write short note on:

(a) 4-Message reliable protocol

(b) 2-Message reliable protocol

(c) Explicit Addressing

(d) Implicit Addressing

(a) 4-message reliable protocol:

- As shown in figure 3-20, the communication between two processes the client sends a request message to the server.
- Server kernel sends acknowledgment to the client. Client does not response in specific timeout period, retransmit the message.
- Server kernel completes the execution of the request, it sends the reply message to the client.
- Client sends acknowledgment message to the server during a timeout period, the server retransmit the reply message.

![Figure 3-20](image)

**Figure 3-20** 4-message reliable IPC protocol

(b) 2-message reliable protocol:

- This protocol prevents unnecessary retransmission of requests.
- The client sends a request to the server and remains blocked until the server sends a reply.
- The server processes the request and returns the reply message to the client. If the reply is not received in the timeout period, the client retransmits the request, as shown in figure 3-22.

![Figure 3-22](image)

**Figure 3-22** 2-message IPC protocol

(c) Explicit Addressing:

In the explicit scheme, the primitive directly names the process with which it communicates, the communication primitives used are send (process_ID, message) and receive (process_ID, message).
(d) Implicit Addressing:
In the implicit addressing scheme, the primitive specifies the process with which it has to communicate but not the process ID. Client server communication uses this addressing scheme. The primitive used are send_any (service ID, message) and receive_any (service ID, message).

Q20. How does the Single-Message Buffering Scheme is differ from Multiple-Message Buffering Scheme?

- For synchronous communication, a single message buffer strategy server the purpose. In case of receiver is not ready to receive the message, it is stored in the buffer.
- Message is readily available to the receiver when it is ready.
- This message passing technique involves two copy operations, as shown in figure 3-14.
- The message buffer can be located either in the receiver’s address space or the kernel process’s address space.
- Multiple message buffers with a capacity of a few messages on the receiver side.
- Multiple message buffers avoid the buffer overflow issue. In this technique sender may not aware that the receiver buffer has overflowed and will keep on sending messages.

Q21. What is difference between Broadcast Communication and Multicast Communication?

Broadcast Communication:
- It is special type of multicast communication, as shown in figure 3-28. One processor sends the same message to several destinations with a single operation.
- Clock synchronization is a same type of example in distributed system.
- In broadcast communication depends on how the group interacts with other group outside the system.
- Closed groups: parallel processing application use closed group communication.
- Open groups: outside process can send a message to a group as a whole.
- Peer group: all process are equal and there is no single point of failure.
- Hierarchical group: in this group coordinator decides which worker should carry out the tasks.
Multicast Communication:

- One to much group communication called multicasting communication. And is widely used in distributed system.
- Multicasting is shown in figure 3-27. For example a server manager needs a volunteer for load balancing.
- In multicast communication send a request to all servers processes and selects the one which responds first.
Q22. Explain the role of middleware in remote communication.

- Middleware is software which provides a programming model one level above the basic building block of processes and also message passing.
- Middleware provides location transparency, independence from communication protocol, operating system and hardware.
- Location transparency: The clients need not know which server process is running its RPC whether it is local or remote or where it is located.
- Communication protocol: processes supporting middleware abstraction are independent of the lower level transport protocol.
- Computer hardware RPC uses techniques like external data representation to hide heterogeneous nature or hardware architecture.
- Operating system: the higher layer abstraction provided by the middleware is independent of operating system.
- Middleware is designed using programming language like Java, CORBA, or IDL.

Q23. Explain the steps for complete RPC execution process.

- The client calls a local procedure, the client stub. For the client, it appears as if the client stub is the actual server procedure, which it has called. A stub packages the argument for the remote procedure by putting into some standard format and builds one or more network message. Marshaling is the process of packaging the client’s argument into a network message.
- The client stub sends the network messages to the remote system via system call to the local kernel.
- A connection oriented or a connectionless protocol transfers the network messages to the remote system.
- The server stub procedure waits on the remote system for the client’s request. It unmarshals the arguments that it received from the client stub.
- The server stub executes a local procedure call to invoke the actual server function passing the arguments that it received from the client stub.
- When the server procedure is completed, it sends the return values to the server stub.
- The server stub converts the return values, marshals them into one or more network messages, and sends them back to the client stub.
- The messages are transmitted across the network back to the client stub.
• The client stub reads the network messages from the local kernel.
• After converting the return values, the client stub returns to the client function.

Q24. Draw RPC call/request message format.

<table>
<thead>
<tr>
<th>Message identifier</th>
<th>Message type</th>
<th>Client identifier</th>
<th>Remote procedure identifier</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A typical RPC call message format

Q25. Describe Call-by-Value parameter passing semantics.
• The semantic copies all the parameters into a message before transmitting them across the network. It works well for compact data types like integer, character and arrays.
• The process of packing the parameters into a message is termed as marshalling.
• A good RPC process system should generate its own marshalling code for every RPC.
• Packing parameters into a message is called parameter marshaling. As a very simple example, consider a remote procedure, add \((i, j)\), that takes two integer parameters \(i\) and \(j\) and returns their arithmetic sum as a result.
• The call to add is shown in the left-hand portion (in the client process) in Fig. 2-3. The client stub takes its two parameters and puts them in a message as indicated. It also puts the name or number of the procedure to be called in the message because the server might support several different calls, and it has to be told which one is required.

![Figure Example of call by value semantic](image)

Prepared by: Nishant R. Viramgama 180702 – Distributed System
When the message arrives at the server, the stub examines the message to see which procedure is needed and then makes the appropriate call. If the server also supports other remote procedures, the server stub might have a switch statement in it to select the procedure to be called, depending on the first field of the message.

The actual call from the stub to the server looks much like the original client call, except that the parameters are variables initialized from the incoming message.

As long as the client and server machines are identical and all the parameters and results are scalar types, such as integers, characters, and Booleans, this model works fine. However, in a large distributed system, it is common that multiple machine types are present.

Each machine often has its own representation for numbers, characters, and other data items. For example, IBM mainframes use the EBCDIC character code, whereas IBM personal computers use ASCII. As a consequence, it is not possible to pass a character parameter from an IBM PC client to an IBM mainframe server using the simple scheme of Fig. 2-3: the server will interpret the character incorrectly.

**Q26. Write short note on: RR protocol & RRA protocol.**

**RR protocol:**
- This protocol is used for RPC whose arguments and results fit into a single packet buffer and where the call duration and the time between the call is short.
- This protocol avoids explicit transmission of messages.
- Figure 4-15 shows the message exchange between the client and the server.
- RR protocol uses timeouts and retries for handling failures, but it requires that the cache maintain the replies.
- If servers interact with a large number of clients, the messages that are not successfully delivered to the client are lost.
**RRA protocol:**
- RRA protocol requires that the client should acknowledge the receipt of the reply messages.
- RRA protocol involves three messages per call, as seen in figure 4-16.
- Assign unique IDs to request message.
- Reply message are matched with the corresponding acknowledgement message.
- Client acknowledges the reply message only if it receives the replies to all earlier requests.
- It depends on number of messages involved in the communication between the client and the server for RPC execution.

**Q27. Define Client-Server binding. Briefly explain all the issues in client-server binding.**
- Client Stub must know the location of a server before RPC can take place between them.
- Process by which client gets associated with server is known as BINDING.
- Servers export their operations to register their willingness to provide services and
- Clients import operations, asking the RPC Runtime to locate a server and establish any state that may be needed at each end.

- The server registers itself with the binding agent.
- The client requests the binding agent for the server’s location.
- The binding agent returns the server’s location information to the client.
- The client calls the server.

**Issues for client-server binding process:**
- How does a client specify a server to which it wants to get bound?
- How does the binding process locate the specified server?
- When is it proper to bind a client to a server?
- Is it possible for a client to change a binding during execution?
- Can a client be simultaneously bound to multiple servers that provide the same service?

**Server naming:**
- The naming issue is the specification by a client of a server with which it wants to communicate.
- Interface name of a server is its unique identifier.
- It is specified by type & instance.
• Type specifies the version number of different interfaces
• Instance specifies a server providing the services.
• Interface name semantics are based on arrangement between the exporter and importer.
• Interface names are created by the users, not by the RPC package.
• The RPC package only dictates the means by which an importer uses the interface name to locate an exporter.

Server locating:
• Broadcasting
  • A message to locate a node is broadcast to all the nodes from the client node.
  • Node having desired server returns a response message. (could be replicated)
  • Suitable for small networks.
  • Binding Agent
  • A name server used to bind a client to a server by providing the client with the location information.
  • Maintains a binding table containing mapping of server interface name to its location.
  • The table contains identification information and a handle to locate it.
  • The location of the binding agent (having a fixed address) is known to all nodes by using a broadcast message.
• Primitives used by Binding Agent
  • Register
  • Deregister
  • Lookup

Q28. Define fixed binding & dynamic binding.
• Fixed binding, the client knows the network address of the server. the client binds directly with server and carries out RPC execution.
• Dynamic binding, which can be carried out in any of the following three ways: compile time, link time or call time

Binding at compile time:
• Servers’ network address can be compiled into client code.

Binding at link time:
• Client makes an import request to the binding agent for the service before making a call.
• Servers handle cached by the client to avoid contacting binding agent for subsequent calls.
• This method is suitable for those situations in which a client calls a server several times once it is bound to it.

Binding at call time:
• A client is bound to a server at the time when it calls the server for the first time during its execution.
• The commonly used approach for binding at call time is the indirect call method.
• The client or server of a connection may wish to change the binding at some instance of time due to some change in the system state.
When a binding is altered by the concerned server, it is important to ensure that any state data held by the server is no longer needed or can be duplicated in the replacement server.

Q29. Write the techniques for RPC optimization.

- There are various optimizations which are possible and can be adopted to achieve better performance of distributed applications using RPC.
- The following techniques for optimization:

Concurrent access to multiple servers
- RPC has one benefit for synchronization property, and many distributed systems benefit from concurrent access to multiple servers.
- Use of thread
- Early reply technique
- Call buffering

Serving multiple requests simultaneously
- RPC systems need to handle the delays caused by servers waiting for a resource.
- This is use a multi threaded server approach with a dynamic thread creation facility for server implementation.

Reduce per-call workload of servers
- Improve the overall server performance and reduce the server workload, it is beneficial to keep the request short and the amount of work to be done by the server low for each request.
- Stateless server help to achieve this objective with the client tracking the request made to the server.

Using reply cache for idempotent RPCs
- The reply cache is helpful in such a situation because the server processes the request only once.
- A reply cache can also be associated with idempotent RPC to improve the server’s performance when it is heavily loaded.

Proper selection of timeout values
- Timeout based retransmissions are necessary to deal with failure in distributed applications.
- RPC systems, servers may take varying amounts of time to service individual requests.
- The time required for execution depends on various factors like server load, network traffic and routing policy.

Proper design of RPC protocol specification
- The protocol specification of an RPC system should be designed so as to minimize the amount of data sent across the network and the frequency of transmission of requests.
Q30. Define RMI. Explain design issues in RMI.

The server manages the objects and clients invoke the method called the RMI. The RMI technique sends the requests as a message to the server which executes the method of the object and returns the result message to the client. Method invocation between objects in the same process is called local invocation. While those between different processes are called remote invocation.

Two fundamental concepts: Remote Object Reference and Remote Interface

- Each process contains objects, some of which can receive remote invocations are called remote objects (B, F), others only local invocations.
- Objects need to know the remote object reference of an object in another process in order to invoke its methods, called remote method invocations. Every remote object has a remote interface that specifies which of its methods can be invoked remotely.

Remote Object References
- accessing the remote object
- identifier throughout a distributed system
- can be passed as arguments

Remote Interfaces
- specifying which methods can be invoked remotely
- name, arguments, return type
- Interface Definition Language (IDL) used for defining remote interface

Design issues in RMI
Two design issues that arise in extension of local method invocation for RMI
The choice of invocation semantics

- Although local invocations are executed exactly once, this cannot always be the case for RMI due to transmission error
- Either request or reply message may be lost
- Either server or client may be crashed

The level of transparency

- Make remote invocation as much like local invocation as possible

RMI invocation semantics

Maybe semantics

- In this semantic, client may not know whether the remote method is executed once or not at all.
- Useful in application where failed invocation are acceptable.

At least once semantics:

- The server has executed the method at least once, or an exception informing that no result was received.
- Retransmission of request messages is used to achieve this semantic.

At most once semantics:

- Client receives that method executes exactly once, or an exception informing that no result was received so far.
- Achieve by using fault tolerance measure.

Level of transparency:

- This involves hiding, marshalling, message passing, locating and contacting the remote object for the client.
- Differences between local and remote invocations
- latency: a remote invocation is usually several order of magnitude greater than that of a local one
- availability: remote invocation is more likely to fail
- Errors/exceptions: failure of the network? Server? hard to tell
- syntax might need to be different to handle different local vs remote

Q31. Describe types of objects.

- Objects are classified in two main classes based on when they are bound and how long they exist.
- Based on the time of binding, objects are classified as runtime and compile time.

Runtime objects:

- Distributed applications are built easily using compile time objects.
- These types of objects are dependent on the programming language. To avoid this dependency, they are bound at runtime.
- So application reconstructed from objects written in multiple languages.
Compile time objects:
- The object adapter acts as a wrapper around the implementation detail to give in the appearance of an object.
- An interface is defined to implement the object which can be later registered. The objects are now available for remote invocations.

Persistent objects:
- These objects exist even if they are not contained in the server process address space.
- It implies that the client manages the persistent object and can store its state on any secondary storage and exit.

Transient objects:
- They exist only for the time when the server manages the object. When the server ceases to exist, the object gets vanished.

Q32. Explain case study: Java RMI.
- Java hides the difference between local and remote method invocation from the user. After marshalling the type of object, it is passed as a parameter to the RMI in java.

Server class:
- This class contains the implementation of serve side code, objects which run on the server. It also consists of a description of the object state and the implementation of methods which operate on that state. The skeleton on the server side stub is generated from the interface specification of the object.
Client class:

- This class contains the implementation of client side code and proxy. It is generated from the object interface specification. The proxy basically converts each method call into a message that is sent to the server side implementation of the remote object.
- Communication is set up to the server and is cut down when call is complete. The proxy state contains the server network address, end to the server and local ID of the object.
- Proxies are serialized in java, marshaled and sent as a series of bytes to another process where it is unmarshalled, and can invoke methods in remote objects.

- Marshalling code is replaced with the implementation handle for remote object reference. In java, RMI references to objects are a few hundred bytes.
- Java RMI allows object specification solution, which is flexible. Objects whose state rarely changes are made truly distributed.
- At each invocation, the client checks the state at binding to ensure consistency. Each process executes the same JVM.
Q33. What do you mean by physical clock and what are the problems with unsynchronized clocks?

Physical clock:
- Every computer needs a timer mechanism to keep track of current time and various accounting purpose.
- Such as calculating the time spent by a process in CPU utilization, disk I/O etc.
- In DS an application may have process that concurrently run on a multiple nodes of the system.
- For correct result applications require that the clock of the nodes is synchronized with each other.
- For example: Distributed on-line reservation system.
- Each CPU has its own clock is required that all the clocks in the system display the same time.

Problems with unsynchronized clocks:
- Consider a distributed online reservation system in which the last available seat may get booked from multiple nodes if their local clocks are not synchronized.
- Synchronized clocks enable measuring the time duration of distributed activities which starts on node and terminate on another node. For example: there is a need to calculate the time taken to transmit a message from one node to another at any time.
- As shown in figure 5-1 event occurred after another event may nevertheless be assigned an earlier time.
- In figure newly modified ‘output.c’ will not be re-compiled by the ‘make’ program because of a slightly slower clock on the editor’s machine.

![Figure 5-1 Problem with unsynchronized clocks](image)

- For correctness, all distributed application needs the clocks on the nodes to be synchronized with each other.

Q34. Explain centralized clock synchronized algorithm.

- In centralized clock algorithm, once node has a real time receiver called the time server node.
- Based on role time server node, centralized clock synchronization algorithm can be classified as passive time server and active time server algorithm.
Passive time server centralized algorithm:

- In this algorithm each node periodically sends a message called time = T to the time server.
- When the time server node receives the message, it responds with time=T message. This is depicted in figure 5-7.

Active time server centralized algorithm:

- In the active server method, the time server periodically broadcast its clock time ‘time=T’.
- All other nodes receive the broadcast message and use the clock time in the message for correcting their own clocks.
- This method is not fault tolerant.
- If a broadcast message reaches a node a little late at a node due to, say, a communication link failure, the clock of the client node will readjusted to an incorrect value.

Another active time server algorithm is the Berkeley algorithm.

- This method avoids readings from unreliable clocks whose values, if chosen, would have modified the actual value.
• Calculated average time to which all the clocks should be readjusted.
• Time server only sends the amount by which each individual computer’s clocks require adjustment.
• Value can either positive or negative. Approximate time required for message propagation from each node to its own node.

Q35. **What is ostrich algorithm?**
• This approach is reasonable if deadlocks occur very rarely and the cost of prevention is high.
• UNIX AND Windows take this approach. This approach is a trade off between convenience and correctness.
• The ostrich algorithm is a good and as a popular in distributed systems as it is in single processor systems.
• Ostrich algorithm used for programming, office automation, and process control, where no system wide deadlock mechanism is present.
Q36. Write the features of a Global Scheduling Algorithm.

- Designing a good global scheduling algorithm helps in improving the performance of a distributed system.

No a priori knowledge about processes to be executed

- Algorithm should operate with no a priori knowledge of the processes to be executed, since it places extra burden on the user to specify this information before the execution.

Ability to make dynamic scheduling decisions

- Algorithm should be able to make scheduling decision dynamically based on the current system status.

Flexible

- Algorithm should be flexible enough to migrate the process multiple times in case there is a change in the system load.

Stable

- Algorithm must be stable such that processors do useful work, reduce thrashing overhead and minimize the time spent in unnecessary migration of the process.

Scalable

- Algorithm should be scalable and able to handle workload inquiry from any number of machines in the network.

Unaffected by system failures

- Algorithm should not be disabled by system failure like node or link crash, and it should have decentralized decision making capability.

Q37. Describe Centralized Heuristic Algorithm.
• The centralized heuristic algorithm does not require advance information and is also called as the top down algorithm.
• In figure 6-4, a coordinator maintains the usage table with one entry for every user and this is initially zero. When events happen, messages are sent to the coordinator and the table is updated.
• For example: the entries could be that a processor is requested, a processor becomes free, or a clock tick has occurred.
• If the machines become overloaded, the coordinator to allocate a processor to it. The request is granted if the processor is available and no one else wants processor to it; request is denied temporarily, and note is made if no processor are free.
• It always favours a light user than a heavy one.
• The heuristic used for processor allocation is that when the CPU becomes free, pending request whose owners have the lowest score win.

Q38. Briefly explain Load-Balancing approaches.
• Load-Balancing approach are based on the fact that an even load distribution helps in better resource utilization.
• This algorithm classified as shown in figure below

![Figure 6-6 Taxonomy of load balancing algorithms](image)

Static vs. Dynamic:
• It use only information about the average behavior of the system, ignore the current state of the system.
• It is simple because it does not need to maintain and process system state information.
• It is limited by the fact that they do not respond to the current system state.
• It responds to the current system state that changes dynamically.
• Dynamic algorithms are better able to avoid those states with unnecessarily poor performance.
• Due to this reason it is better than the static algorithms
Static: Deterministic vs. heuristic:
- It uses the information about the properties of the nodes and the characteristics of the process to be scheduled to allocate processes nodes.
- It uses the information regarding static attributes of the system such as number of nodes, the processing capability of each node, the n/w topology, etc, to formulate process placement rules.
- For Example: Suppose a system has two processors \( p_1 \) and \( p_2 \) and four terminals \( t_1, t_2, t_3 \) and \( t_4 \) then placement rule can be, assign all process originating at terminals \( t_1 \) and \( t_2 \) to processor \( p_1 \) and so on.

Dynamic: Centralized vs. Distributed:
- In this algorithm single node is responsible for scheduling.
- In this approach, the system state information is collected at a single node at which all scheduling decision are made. This node is called the centralized server node.
- It decides about the placement of a new process using the state information stored in it.
- This approach can efficiently make process assignment decision, Because the centralized server knows both the load at each node and the number of processes needing service.
- The other nodes periodically send status update message to the central server node.
- To keep the system state information up to date at the centralized server node.
- Theimer and Lantz proposed an approach of reinstatiation to improve the reliability of the centralized scheduling mechanism.
- In this approach rather than maintaining \( k +1 \) server replicas, a single server is maintained and there are \( k \) entities monitoring the server to detect its failure.
- When failure is detected, a new instance of the server is brought up, which reconstructs its state information by sending a multicast message for state update.
- The time during which the scheduling service is unavailable will be the sum of the:
  - The time to detect failure of the previous server.
  - The time to load the server program on the new server.
  - The time to resolve the possibility of multiple concurrent instantiations among the \( k \) entities and the time to reconstruct the global state information on the new server.
  - The process assignment decision is physically distributed among the various node of the system.
- Allows the scheduler to react quickly to dynamic changes in the system state.
- This algorithm is composed of \( k \) physically distributed entities \( e_1, e_2, \ldots, e_k \).
- Each entity is considered a local controller.
- Each local controller runs asynchronously and concurrently with the others and each is responsible for making scheduling decision.

Cooperative Vs Non cooperative:
- No cooperative:
  - In this algorithm, individual entities act as autonomous entities and make scheduling decisions independently.
- Cooperative:
In this algorithm, the distributed entities cooperate with each other to make scheduling decisions.

- It is more complex and its stability is better than no cooperative algorithm.

**Q39. What are the issues in Load-Sharing approach?**

- To design a good load sharing algorithm, it is important to make proper decision regarding estimation policy and location policy etc.
- Load estimation policies
- Process transfer policies
- Location policies
- Sender initiated location policies
- Receiver initiated location policies
- State information exchange policies

**Q40. What do you mean by process migration?**

Process migration is the relocation of a process from its current location to another node.

- It may be migrated either before it starts executing on its source node or during the course of its execution.
- The former is known as non-preemptive process migration and,
- The latter is known as preemptive process migration.
- Preemptive process is costlier than non-preemptive.
- Process migration involves the following major steps:
- Selection of process that should be migrated.
- Selection of the destination node to which the selected process should be migrated.
- Actual transfer of the selected process to the destination node.
- Once we identify the process to be migrated and where it is migrated, the next step is to carry out process migration.

**The major steps involved in process migration are:**

- Freezing process on the source node
- Starting process on the destination node
• Transporting process address space on destination node
• Forward the messages addressed to migrated processes

**Q41. What is the difference between Process and Thread?**

**Table 6-1 Comparison of processes versus threads**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Process</th>
<th>Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control block</td>
<td>Process Control Block (PCB): program counter, stack, and register states; open files, child processes, semaphores, and timers</td>
<td>Thread Control Block (TCB): program counter, stack, and register states</td>
</tr>
<tr>
<td>Address space</td>
<td>Separate for different processes, provides protection among processes</td>
<td>Share process address space, no protection between threads belonging to the same process</td>
</tr>
<tr>
<td>Creation overhead</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Context switching time</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Objective of creation</td>
<td>Resource utilization, to be competitive</td>
<td>Use pipeline concept, to be cooperative</td>
</tr>
</tbody>
</table>

**Q42. Explain Thread synchronization.**

- If two threads want to double the same global variable, it is best done one after another. One thread should exclusively access it, double it, and pass the control to the other thread.
- A critical region is implemented using a mutex variable which is a binary semaphore having two states: locked and unlocked.
- In figure 6-26 lock operation attempt to lock the mutex. It succeeds if unlocked and the mutex becomes locked in a single atomic action.
- If more than thread is waiting in the mutex, one is released, while the others continue to wait.
• The trylock operation attempt to lock a mutex. Based on the status of the mutex, either the mutex unlocks, or trylock returns a status code which indicates success.

• The thread package provides another synchronization feature called the condition variable which is associated with a mutex at the time it is created.
Q43. Explain Basic Distributed Shared Memory Architecture

A DSM system provides a logical of shared memory which is built using a set of interconnected nodes having physically distributed memories.

As shown in figure 7-1 DSM is a collection of workstation connected by a LAN sharing of virtual address space.

Processes running on different nodes can exchange messages through the implementation of a simple passing system.

The DSM abstraction exists only virtually and presents a consolidated memory space to all the nodes.

DSM systems are claimed to be scalable and achieve very high computing power.

Q44. What are the types & advantages of DSM?

DSM systems are classified depending on the level of DSM implementation, the type of DSM algorithm and the organization of DSM management. DSM systems are classified as:

- Hardware-level DSM
- Software-level DSM
- Hybrid level DSM
- Advantages of DSM are following
  - Simple abstraction
  - Improved portability of distributed application programs
  - Better performance in some applications
  - Large memory space at no extra cost
  - Better than message-passing systems
Q45. What is Thrashing?

- Access to different variables on the same page is called thrashing.
- False sharing
- Ping-pong effect means data access made by more than one node on same data block within a short interval of time

Techniques to reduce thrashing:

- Application controlled lock
  - Other nodes can be prevented from accessing the block
- Pin the block to a node for specific time
- Customize algorithm to shared data usage pattern
  - reduced the access to shared data with help of programmer

Q46. What are the replacement strategies in DSM?

- The commonly used replacement algorithms are divided into the following groups: usage-based versus non-usage-based algorithms, and fixed versus variable-space algorithms (see Figure 7-34).
Usage-based versus non-usage-based algorithms:
- The usage-based algorithm maintains a history of the page usage, and makes the replacement decision based on this information. An example of usage-based algorithm is the Least Recently Used algorithm (LRU).
- Non-usage-based algorithms like FIFO and Random algorithm do not record the use of cache lines, and make the replacement decision randomly.

Fixed versus variable space algorithms:
- The replacement strategy is designed as follows.
- Both unused and nil blocks are no longer useful, and are the first to be replaced. The read-only blocks are the next to be replaced, since copies of these blocks are available on other nodes, and they can be procured when needed from the owner.
- The read-owned and writable blocks for which replicas exist on other node are the ones which can be replaced in the list after passing ownership to the replica.
- The read-owned and writable blocks for which only this node has a copy have the lowest replacement priority. This is because the block's ownership and the block itself have to be transferred to some other node.

Q47. Explain the case studies: Munin & Linda.

Munin:
- Munin treats the collection of all memories in the distributed system as a single address space with coherence enforced by the software.
- Munin views memory on each machine as a collection of disjointed segments.
- The virtual address space of each processor is partitioned into shared and private areas. The private area is local to each processor and contains non-shared data; the runtime structures are used to manage memory coherence; and the system memory map is used to record the segments of global shared memory that are currently mapped into the local portion of shared memory.
- The system map may also contain hints about other processors' shared memory areas, which may not always be reliable, due to infrequent updates.
- Munin servers on each machine interact with the application program and the underlying distributed OS to ensure that the segments are correctly mapped into the local memory when they are accessed.
- Munin performs fault handling in a manner analogous to page fault handling in a virtual memory system. When a thread accesses an object for which there is no local copy, a memory fault occurs. This causes Munin to suspend the faulting thread and invoke the associated server to handle the fault.
- The server checks to see what type of shared object the thread faulted on and invokes the appropriate fault handler. The suspended thread is then resumed after handling the fault.
- Munin treats each shared data object as one of the following nine types of objects, private, write once, write many, result, synchronization, migratory, producer consumer, read mostly and general read write.
Linda:

- Language-independent tuple-based DSM system.
- It provides processes on multiple machines with structured shared memory
- Global data is written to and read from the DSM in passive tuple.
- Four Operation on Tuple:
  - Out: puts a tuple into tuple space
  - In: retrieves a tuple from the tuple space
  - Read: Similar to out but does not remove tuples from tuple space
  - Eval: Parameters to be evaluated to be in parallel and resulting tuple is deposited in the tuple space.

If reliable broadcasting is available, it is possible to replicate all subspaces in full on all machines.
- Out operation: New tuple is broadcast and entered into subspace on each machine.
- In operation: local subspace is searched, and tuple is removed from the tuple space.
- Drawback – Does not Scale well due to the every tuple is stored on each machine.

Out operation: Outs locally and store the tuples only on machine it was generated.
In operation: the machine must broadcast a template.
Q48. What are the desirable features of a Good Naming System?

- A good naming system should have desirable features like location transparency, location independency, and scalability.
- The features related to the actual naming of objects include using a uniform naming convention, assigning meaningful names, allowing multiple user-defined names for the same object, and group naming.
- Location Transparency
- Scalability
- Uniform naming convention
- Meaningful names
- Allow multiple user-defined names for the same object
- Group Naming
- Performance
- Fault Tolerance
- Location independency
- Replication transparency
- Locating nearest replica
- Locating all replicas

Q49. What are the basic approaches to generate System-Oriented Names?

- There are two approaches to generate system oriented names

**Figure 9-14** Centralized approach to system-oriented naming

**Figure 9-15** Hierarchical concatenation strategy

**Centralized approach:**
- This approach used for generating unstructured names and is shown in figure 9-14. This system incorporates a centralized global unique identifier generator that generates a standard global unique identifier for each object in the system.
- This approach is simple and easy but it exhibits poor efficiency and reliability.
- In this approach each node may either bind or map the global identifier to the locally created object.
• It is totally dependent on the reliability of the centralized generator.

**Distributed approach:**
• The distributed approach to system naming overcomes the drawback of poor efficiency and reliability of the centralized approach.
• The hierarchical concatenation strategy is used to create global unique identifier.
• Using timestamp here, each node is treated as a domain of namespace and uses the clock timestamp as a unique identifier within the node. The global identifier is in the form of a pair (Node-ID, timestamp).
• Using servers as a domain here, each server is treated as a domain and hence, the server generates object identifiers. So, the global identifier takes the form of a pair.

**Q50. Explain the concept Name Caches.**
• A name cache stores the name resolution information about recently accessed objects names.
• Name caches can be categorized depending on the type of information stored in each entry.
• Name resolution operations are not cheap. It was found that in operating systems that provide a hierarchical namespace, the system overhead involved in name resolution operations is large.
• Also in distributed systems, the network traffic is naming-related. Hence, substantial network traffic can be reduced if the client can cache the result of name resolution operations.
• Literature indicates that a distributed name cache is advantageous due to the following characteristics of name service related activities.
  • High degree of locality of name lookup
  • Slow update of name information database
  • On use consistency of cached information is possible