DISTRIBUTED COMPUTING

TY1

Q1 Objective of middleware in DS

ANS
- Hide heterogeneity
- Location independence
- Common functionality needed by many applications
- Software portability and mobile code
- Help integrate legacy features
- Aid application interoperability
- Aid scalability

Q2 Issues in designing a distributed system.

ANS
1. Transparency
2. Flexibility
3. Reliability
4. Performance
5. Scalability
6. Security.

TRANSPARENCY : Types of transparencies

- Access : Hide the differences in data representation and how a resource is accessed.
- Location : Hide where a resource is physically located
- Migration : Hide movement of a resource to another location
- Relocation : " while in use
- Replication : Hide the fact that multiple copies exist.
- Concurrency : Hide the fact that resource might be shared
- Failure : Hide the failure and recovery of a resource
- Persistence : Hide whether a resource is in memory or on disk.

FLEXIBILITY:

- Monolithic kernel approach uses the minimalist modular approach with accessibility to other services as needed
- Microkernel approach uses the 'kernel does it all' approach with all functionalities provided by the kernel irrespective whether all machines use it or not.

RELIABILITY:

- Available in case of hardware failure
- Data recovery in case of data failure
- Maintain consistency in case of replicated data

PERFORMANCE:

Metrics are: → Response time
→ Throughput
→ System utilization
→ Amount of network capacity used.

SECURITY:

- Confidentiality means protection against unauthorized access
- Integrity implies protection of data against corruption.
- Availability means protection against failure always accessible.

FLEXIBILITY:

- Monolithic kernel approach uses the minimalist, modular approach with accessibility to other services as needed
- Microkernel approach uses the 'kernel does it all' approach with all functionalities provided by the kernel irrespective whether all machines use it or not.

RELIABILITY:

- Available in case of hardware failure
- Data recovery in case of data failure
- Maintain consistency in case of replicated data

PERFORMANCE:

Metrics are: → Response time
→ Throughput
→ System utilization
→ Amount of network capacity used.

SECURITY:

- Confidentiality means protection against unauthorized access
- Integrity implies protection of data against corruption.
- Availability means protection against failure always accessible.

Q3 Types of Middleware

ANS
1. Message oriented Middleware
2. Remote Procedure calls
3. Object Request Broker
4. Transaction Processing Monitors.

MESSAGE ORIENTED MIDDLEWARE:

- MOM uses messages as the method of integration
- In MOM, applications are decoupled i.e. senders and receivers are unaware of each other
- Messages are sent asynchronously
- Messages are queued.
- Intermediate message servers does filtering, transforming, logging, etc.

ADVANTAGES:
- Asynchronous
- Flexible
- Portability
- Interoperability
- Reduces complexity

REMOTE PROCEDURE CALL

- Inter-process communication
- One client can request a service from a program located in another computer in a network without having to understand network details.
- synchronous. Therefore, a waiting period between request a response is there.

3. OBJECT REQUEST BROKER

- Supports the development of distributed object-oriented application
- CORBA (Common Object Request Broker Architecture) is an e.g.
- ORB allows communication between objects in distributed computing
- Because they are built over standards, two compliant objects can exchange info or call each other's method without learning its actual location in the network.

TRANSACTION PROCESSING MONITORS

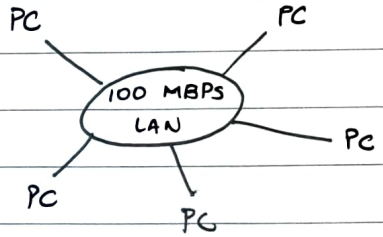
- For long the dominant form of middleware
- Main goal is to support execution of distributed transactions
- Is a set of information which process the data transaction in database system.

Q4 Distributed computing models

ANS
- workstation model
- workstation – server model
- Processor - pool model

WORKSTATION MODEL :

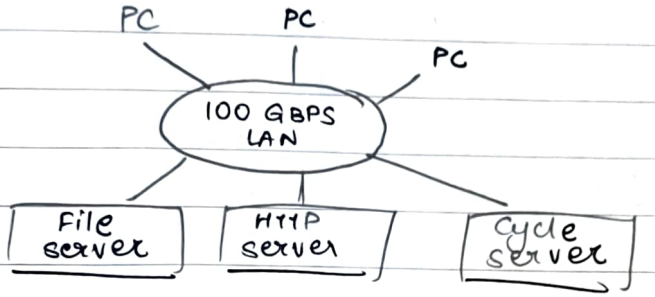
- Network of PC's with own hard disk & local file system



- what happens if user logs on a PC that running a remote job
- How to find idle PC

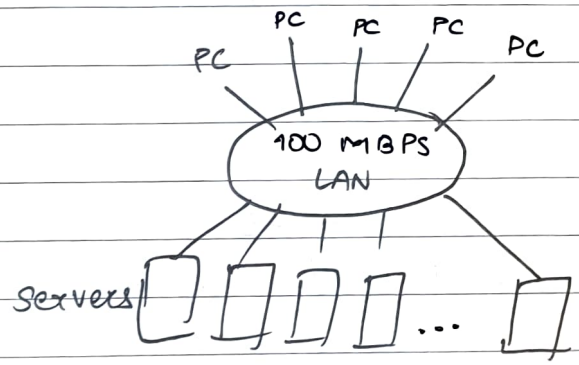
WORKSTATION - SERVER MODEL

- Multiple workstations (diskless) coupled with powerful servers with extra hardware to store the file systems and other software like databases



PROCESSOR - POOL MODEL

- Consists of multiple processors : a pool of processors and a group of workstations.



Q5 RPC call semantics.

ANS RPC call semantics defines how often a remote procedure would be executed under faulty conditions

1) Possibly or Maybe call semantics
2) Last-one call-semantics
3) Last of many call semantics
4) At-least-once call semantics
5) Exactly-once call semantics

1] POSSIBLY OR MAY-BE CALL SEMANTICS

→ One of the weakest semantics
→ Caller waits for a pre-determined timeout ~~money~~ period and continues with its execution
→ No guarantee of receipt of call message.

2] LAST-ONE CALL SEMANTICS

→ Retransmits the call message, based on a predetermined timeout, until the caller receives the response
→ Difficult to achieve with orphan calls
→ Orphan calls are calls whose caller has expired due to a node crash. No parent is waiting for such calls, resulting in unwanted computation. These calls waste CPU cycles, lock files, and may tie-up resources. If client reboots, the computation repeats.

last-one call sematic techniques:-

1) Extermination : client log is checked on reboot and orphan process is killed. $$$

2) Reincarnation : Time is divided into epochs. If replies contain outdated epoch, it is an orphan.

3) Gentle Reincarnation : Tries to find owner. if not found, orphan is killed.

4) Expiration : Each RPC is given a quantum of time. can be extended. When server reboots, orphans are gone.

3] LAST-OF-MANY CALL SEMANTICS

- Each RPC has a unique call-ID
- Neglects orphan calls by checking call ID
- client checks call ID and accepts reponse if ID matches recent call

4] AT-LEAST-ONCE CALL SEMANTICS

- call is executed once or more but we don't know which results the caller gets
- message timout based retransmission.
- caller accepts first response message.

5] EXACTLY-ONCE CALL SEMANTIC

- Irrespective of calls, function is executed only once
- Strongest, most desireable, cheapest
- Idempotent operations are those that give same results always.

Q7 Explain parameter passing semantics

1. call-by-value semantics
2. call-by-reference semantics
3. call-by-copy/restore semantics

1. CALL-BY-VALUE SEMANTIC

- This semantic copies all the parameters into a message before transmitting them across the network
- This process is called marshalling.

steps in Marshalling:-
- Identify the message data-arguments
- Encode the data on the sender's machine
- Receiver decodes the message

- call-by-value semantic is required because client and server dont share memory.
- unsuitable for large value transmissions

2. CALL-BY REFERENCE SEMANTIC:

- Pointers to the parameters are passed from client 2 server.
- used in systems having a shared memory
- Caller as well as called process can modify the values in the shared array

3. CALL BY COPY / RESTORE SEMANTICS:

- This mechanism uses a temp storage which is accessed by both caller and called process
- Caller copies variables into a stack for executing the RPC and later copies back the values after execution is done
- called process can modify values in stack

Q8 RPC communication protocols.

ANS
1. Request Protocol (R-Protocol)
2. Request / Reply Protocol (RR-Protocol)
3. Request / Reply / Acknowledgement Protocol (RRA-Protocol)

REQUEST PROTOCOL (R-Protocol)

- Used where server procedure has no response
- Client is not blocked waiting for result
- Server need not send a reply. Asynchronous.

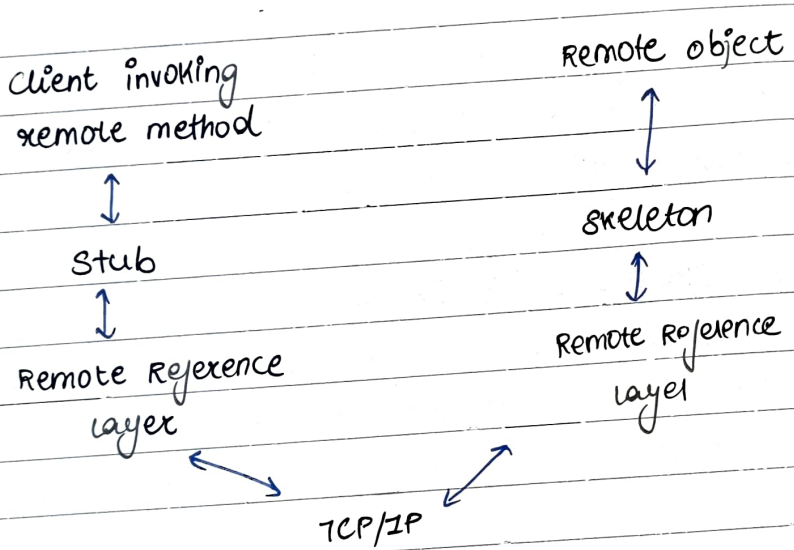
REQUEST / REPLY PROTOCOL (RR)

- call duration and time between calls is short.
- server's reply acts as acknowledgement
- subsequent call from client is considered an ack of the server's reply.
- The protocol uses timeout and retries for handling failures but requires a cache.
- If no response ⇒ retransmit lost request message
- This method is not reliable, because the messages that are not successfully delivered to the client are lost

REQUEST/REPLY ACKNOWLEDGEMENT (RRA)

- This protocol requires that the client should acknowledgement the receipt of the reply msgs.
- Unique ID's are assigned to each request.
- Reply messages are matched with the corresponding acknowledgement message.

Q9 Explain RMI components & execution.



RMI FLOW DIAGRAM

- When the client binds to a distributed object, the object interface called PROXY is loaded into the clients address space.
- Proxy is similar to client-server stub of RPC
- It marshals the RMI request into a message before sending it to a server and unmarshals the reply of the method invocation when it reaches back to the client

RMI COMPONENTS :

1. Communication module
2. Remote reference module
3. RMI software
4. Server and the client programs
5. The binder.

COMMUNICATION MODULE

- The client and the server processes form a part of the communication module which uses RR protocol.
- The format of the request-and-reply message is very similar to the RPC message.

REMOTE REFERENCE MODULE

- Responsible for translating between local & remote object references and creating remote object references.
- It uses a remote object table which maps local to remote object references and is used to locate the invoked object.
- This module is called by components of RMI software for marshalling & unmarshalling remote object references.

RMI SOFTWARE

- Middleware layer and consists of the following:-

A) Proxy

- Makes RMI transparent to the client and forwards the msg to the remote object
- It marshals the arguments & unmarshals the results
- There is 1 proxy for each remote object for which it holds remote reference.

B) Dispatcher.

- server consists of 1 dispatcher and skeleton for each class which represents remote object.
- This unit receives the request msg from the communication module, selects the appropriate method from the skeleton, and passes the request msg

C) Skeleton

- It implements the method in the remote interface.
- unmarshalls the msg and invoke the corresponding method in the remote object.
- After execution, it marshals the result in the reply msg to send to proxy's method.

SERVER AND CLIENT PROGRAM :

- server program contains classes for dispatcher, skeleton and remote objects that it supports.
- client program contains classes of processes for the entire remote object, which it will invoke.
- client looks up the remote object reference using the Binder.

BINDER

- It is a service in distributed systems that maintains a table of textual names to be remote object referenced.
- servers use it to look up the remote object reference.

Q10 Desirable features of Global Scheduling Algorithm.

ANS

1) General Purpose
2) Efficiency
3) Fairness
4) Dynamic
5) Transparency
6) Scalability
7) Fault Tolerance
8) Quick decision making capability.
9) Balanced system performance & scheduling overhead
10) Stability.