distributed objects. A distributed object is an object that can be accessed remotely from anywhere on network.
they are use - to share info.
- to synchronise machine
- increase performance
- work together by sharing data & invoking method.

local object - local object are those whose methods are invoked by local process, a process running on same computer.

local obj	vs	Distributed method
→ invoked by local process that run on same comput.		invoked by remote process a process running on different computer.
→ Remote referenc are simple		Remote ref. are complex than simple pointer to memory.
→ faster latency.		slower latency.
→ No parallel execution		Execute in parallel.
→ Single point failure.		more point failure
→ less vulnerable to attack.		vulnerable to attack

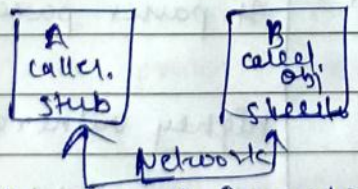
Communication — RMI

→ widely used approach using stub & skeleton.

Part of RMI

→ Stub - It is an object act as gateway for client side. all outgoing request go throug it.

→ Skeleton - It an object act as gateway for sever side. All incoming request go through it.



(RRL - Remote Reference layer - which manages reference made by client to remote object)

RMI - Remote method Invocation -

→ when caller want to perform remote call it request stub, a stub passes arguement over N/w to skeleton. the skeleton passes received data to called object wait for response and return result to client stub.

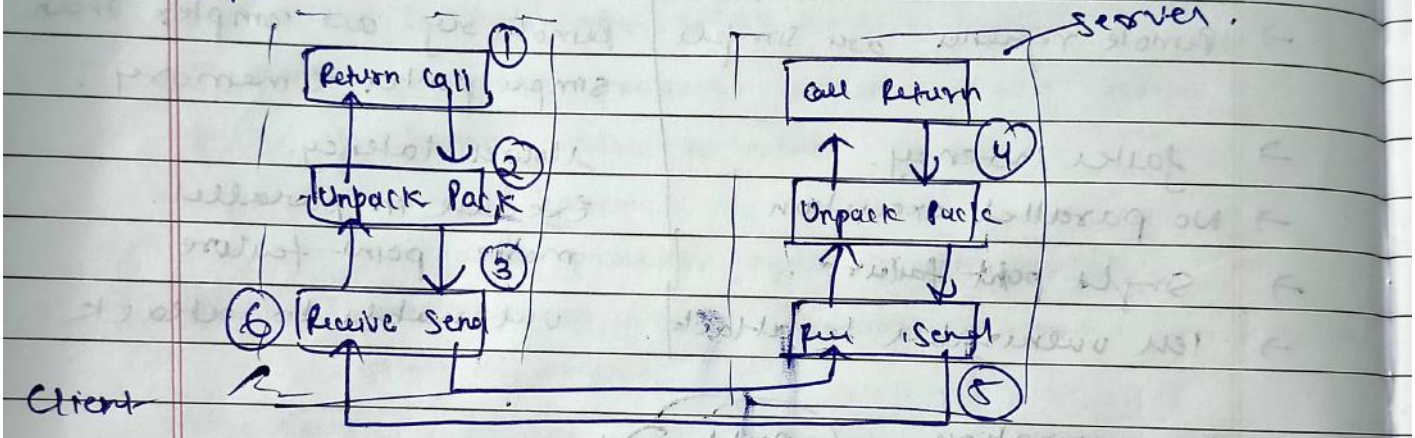
→ No direct communication.

architecture of RMI → client → stub → RRL → Transport layer ⟷ sever. Skeleton → RRL (virtual connection) → Transport layer (N/w connection)

RPC - Remote Procedure calls - It is Interprocess communication technique used for client server application.
It is used when computer program calls a procedure or subroutine to execute in different address space which is coded as normal procedure call without knowing about remote interaction.
→ also manage low level transport protocol, like TCP/IP, UDP

Architecture - 1) Client 2) Client Stub 3) RPC Runtime 4) Server Stub 5) Server.

Steps - 1, 2, 3, 4, 5, 6



Characteristics of RPC - : called procedure is in different computer.
- Process do not share address space
- Parameter are passed by value.
- It do not offer access to calling procedure env.

Adv:	Disadv
→ Communication with server in HLL	It pass parameter by value
→ Rewriting, redeveloping code is minimum.	highly vulnerable to failure.
→ Process & thread oriented model support	No flexibility for hardware architecture.
→ Usage of applicatn in distributed env.	

classmate
Date
Page

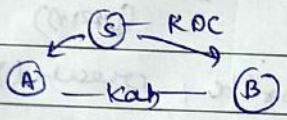
Security - making system secure - following method are used -

- Encryption
- decryption
- cryptography - Public key Private key. RSA algorithm.

Symmetric key cryptography, asymmetric key cryptography

→ Needham-Schroeder is a security protocol designed to ~~insert~~ ensure secure communication over insecure network. based on key distribution center to securly distribute keys to communicating parties.

→ It establish secure communication session b/w client & server



(Provides Kab to A & B)

→ Kerberos - N/w authentication protocol uses symmetric key cryptography to provide secure communication.

→ It allow client and server to authenticate themselves and to encrypt data to ensure confidentiality & integrity.

→ It contains - KDC, client and server.

* distributed file system -

- file system with data stored on server. data is accessed and processed as if was stored on local machine.

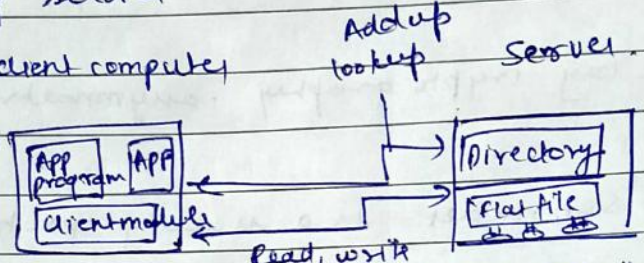
- distributed file system increase, reliability, availability and scalability compared to single file system.

→ File service architecture - in this distributed file system is organised as collection of file server that provide access to file and directory.

- Each file server maintains it local file system and client access to it by connecting to appropriate server.

File service architecture has three component -

1) client module
2) flat file service
3) directory service



model

1) Flat file service - It is used to perform operation on content of file. Unique file identifier are associated with each file. (UFID)

Operation - Read, write, create(), delete(), getAttribute(), SetAttribute.

2) Directory servig - directory service server the purpose of relating file text name with their UFID. The directory servig provide operation for creating directory and adding new file to directorie.

operation - Lookup() - (Returns relevant UFID)
Addname() - Add (Name, file) to directory
Unname() - If name is present it is unnamed
getname() - return all text name.

3) Client module - It is present on each computer. It store information about network location of flatfile and directory, recently used block are hold in cache.

→ DFS issues - Naming, Transparency, Caching, Replication

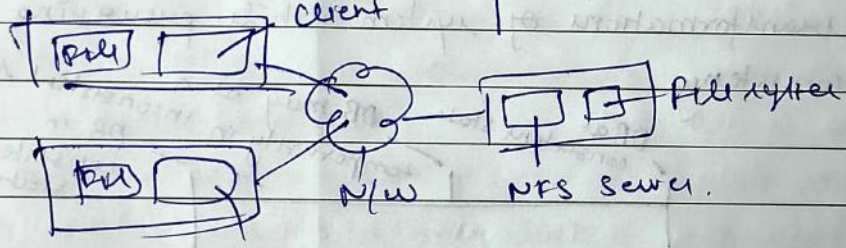
Advantage - Access time reduced, Safe if node fails, client need not to have large memory.

disadv - slower, difficult to keep local and remote copy consistent.

NFS - network file system

→ It allows a user on client computer to access file over a computer network much like local storage.
→ It is a client-server application.
→ It use RPC
→ It use stateless protocols.

Stateful Protocol	Stateless protocols
→ require server to maintain state and session info.	It do not require.
→ dependency b/w client & server	no tight dependency
→ complex in design. and make server very complex	simplify server design.
→ handle transaction slowly	fastly.



NFS

→ Andrew file system is designed to provide scalable and highly available file storage across a network of servers.
→ use client-server architecture
- It has advance features of replication across multiple server, use caching, security etc.

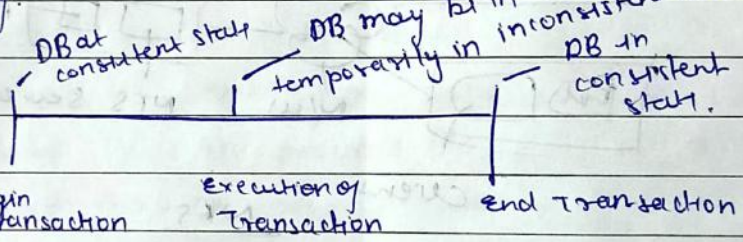
→ SSL & millicent - It is protocol for establishing secure links b/w networked computers.
- In order to provide privacy SSL encrypts data that is transmitted across web.
- SSL initiates authentication b/w communicating devices using handshake.
- SSL provides digital sign to provide data integrity.

~~Transaction and concurrency~~

	RPC	RMI
1)	RPC is library and OS dependent	whereas it is Java platform
2)	Suppor procedural programming.	RMI support object Oriented programming.
3)	less efficient	more efficient
4)	parameter passed are ordinary or normal data	objects are passed as parameter.
5)	It is older version of RMI	Successor of RPC
6)	No security	provide client level security.
7	high development cost	low development cost.

Transaction and concurrency control

→ A transaction is collection of action that make consistent transformations of system while preserving system consistency.



→ Transaction is sequence of operations must be performed atomically and consistently across multiple node.

→ characterization → Read set (RS) Set of data item read by transaction
write set (WS)
Base set (RS ∪ WS)

→ Principles of transaction - Atomicity
Consistency
Isolation
Durability

→ characterization of transaction based on -

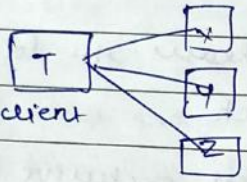
1) Application area → - Non distribute, distributed
- compensating
- Heterogeneous.

2) Timing - online
batch.

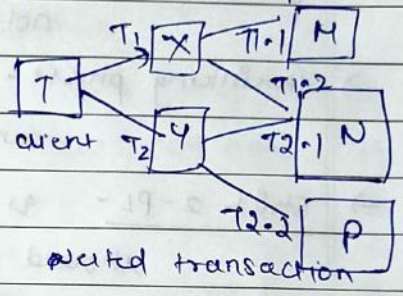
(3) - Flat transaction
- Nested
- workflow

→ Flat transaction - A flat transaction has single point (Begin) and a single end (point). They are simple and used for short activities.

→ Nested transaction - A transaction that includes other transaction within its initiating point and end point.



flat transaction, T that perform operation on object in X, Y, Z server.



Nested transaction
T has two nested transaction T1 & T2

- Nested transaction has high performance than single transaction.

→ Locks - lock are used to manage access to shared resources and prevent conflict between concurrent operation or transaction.

There are two types of lock -

1) Shared lock - It is read only lock.
2) Exclusive lock - In exclusive lock data item can be both read as well as written by transaction.

- Concurrency control algorithm

concurrency control technique ensure that multiple transaction are executed simultaneously while maintaing ACID property of transaction and serializability.

→ locking based concurrency control Protocol

- It uses concept of locking data items.
- A lock is variable associated with data item that determine whether read / write operation can be performed on data.

→ One phase locking - Each transaction locks an item before use and release lock as soon as it has finished using it. Do not enforce serializability.

→ Two phase locking - all locking operation proceed the first lock-release or unlock operation.
Contain two phase - growing phase.
- shrinking phase.

→ growing phase - transaction acquire all lock and do not release any lock

→ shrinking phase - the transaction release the lock and cannot request any lock.

→ Strict 2-PL - It same as PL and all exclusive lock should hold untill commit / Abort

→ Rigorous 2PL - Same as PL + all shared, exclusive lock should hold untill commit Abort.

→ conservative 2-PL - Also known as static 2PL. By predeclaring its read, write set, this protocol compel transaction to lock all item before it begin execution.

→ If any data are not available for locking, then no data item are locked.
Read and write data item, need to know before transaction begin. This is not generally possible.

→ Optimistic concurrency control
In system with low conflict rate, the task of validating transaction for serialization may have low performance.
In this case test for serializability is postponed.
In this approach transaction life cycle is divide into 3 phase -

1) Execution phase - transaction fetches data item to memory and perform operation on them.
2) Validation phase - A transaction perform check to ensure committing changes to the database pass serializability.
3) Commit phase - A transaction write back modified data item to memory.

Timestamp Ordering protocol -

→ The main idea for this protocol is to order the transaction based on their timestamp.

→ The order of transaction is nothing but ascending order of transaction creation.

→ Priority of older transaction is higher.

→ Ex there are two transaction T1 & T2 suppose T1 has entered at 007 time and T2 has 009 time, T1 has higher priority.

→ Timestamp ordering protocol work as -

1) Check the following condition whenever Ti ~~the~~ issues Read(X) operation.

~~If W-TS(X) > TS(Ti) operation is rejected~~.

RTS (~~Red~~ Read time stamp) Last transaction which performed Read.

WTS (write time stamp) last transaction which performed write.

TS (Time stamp)

Rule) -1) Transaction Ti issues a Read (A) operation

a) If $WTS(A) > TS(Ti)$ Rollback Ti

b) otherwise execute R(A)

Set $RTS(A) = \max\{RTS(A), TS(Ti)\}$

2) Transaction Ti issues write(A) operation.

a) If $RTS(A) > TS(Ti)$ Rollback Ti

b) $WTS(A) > TS(Ti)$ "

c) otherwise execute write(A)

Set $WTS(A) = TS(Ti)$

→ Atomic commit Protocol - Atomic commit Protocol guarantee the atomicity property of a transaction in which transaction are completed or not.

Types -

1) distributed one-phase commit - It involves one co-ordinator who communicate with servers and perform each task regularly to inform them to perform or cancel operation.

- distributed two phase -

Phase-1 voting - A "prepare message" is sent to participant by co-ordinator
- co-ordinator wait for response until each reply
- If transaction is ready worker sends "ready" called.
- If not ready abort is called.

Phase-2 - completion of voting
- If all worker send "ready" then only commit happens, otherwise abort
- Now wait for acknowledgement unit received from each worker.

→ concurrency control in distributed system - Locking protocol timestamp

1) Distributed Two phase locking - Basic principle is same as two phase locking. however in OS there are site designated a lock manager. A lock manager control lock acquisition.
- To order co-ordination one site is given authority to see all transaction & detect lock conflict.

different locking approaches -
1) centralised two PL - In this one site is designated as lock manager.
2) Distributed 2PL - In this there are number of lock manager where each lock manager controls lock of data item stored at its local site.

- distributed timestamp concurrency - In centralised system timestamp of any transaction is determined by physical clock reading. But in distributed system any site's local clock reading cannot be used as global timestamp. So timestamp comprise of combination of site ID and site's clock reading.
- distributed optimistic concurrency control Algorithm

It extends basic optimistic protocol as -

Rule 1 - a transaction must be validated locally, if invalid abort it. Local validation maintain serializability at site, if it passed then tested at global level.

Rule-2 If local validation is ~~true~~ correct it is globally validated. It ensure that if two conflicting transaction run together at more than one site they should commit in same relative order at all site they run together.

→ Distributed deadlocks -

distributed deadlock can occur when distributed transaction or concurrency control are utilized in distributed system.

→ It may be identified by edge chasing or by creating global wait for graph.

→ In distributed system deadlock cannot be prevented or avoided because large system it can be detected.

Approaches to detect

1) Centralised approach - Only one resource is responsible for detecting deadlock in centralised method and it is simple and easy.
2) Hierarchical approach - It is integration of both centralised and distributed approaches. In this a single node handles a set of selected node that are incharge of deadlock detection
3) distributed approach - In this various node work to detect ~~dd~~ deadlock. & There is no single point of failure as workload is divided among all nodes, also increase speed of detection.

Transaction recovery -

→ Transaction in distributed system sometime fail due to network error, inaccurate data. Transaction failure are impossible to detect. when mistake occur one must be able to identify and correct them. Transaction Recovery is used for this -

→ methods of transaction Recovery

1) Two phase commit - Contain two stages, first step is "prepare" phase in which co-ordinator deliver a prepare message.
second step decision-making in which co-ordinator send a "commit" message if all node can complete transaction. or abort if at least one node cannot.

* Linear 2PC - Subordinate in this can communicate with each other.
→ As a result "prepare" message is propagated in sequential manner
→ As a result transaction take longer to complete.
→ Finally last node send out global commit.

* Distributed 2PC - All node interact with one another. It do not require 2nd phase.
- In order to know that each node must hold a list of all participating nodes.
- when co-ordinator deliver a prepare message to all nodes.
- when participant receive prepare it transmits his or her vote to all other participant.

→ group communication - Broadcast communication
Multicast communication
Unicast communication.

→ Fault tolerance - dynamic method that used to keep interconnected system together, maintain reliability and availability in distributed system.

- Replication - Replicating data and service across multiple nodes.
- Redundancy - System continue to function if one or more instance fail.
- Checkpointing - ~~prev~~ ~~prev~~ periodically saving state of system.
- Failure detection and recovery - finding failure and recovering them.