Agreement Protocol Unit-3

In D.S. processes often compete as well as cooperate to achieve a common goal. It is often required that sites reach mutual agreement.

for Ex. In ~~Distributed~~ distributed data base system data managers at sites must agree on whether to commit or to abort a transaction

Reaching an agreement requires that sites know knowledge about the value of other site.

when the system is free from failure an agreement can easily be reached among the processors.

when faulty processes are present in the system; they can send conflicting value to other processes. preventing them from reaching an agreement.

In the presence of faults, processors must exchange their value with other processors and relay the values received from other processors. several time to isolate the effect of faulty processors

Entire process of reaching an agreement is called an agreement protocol.

In Agreement problem non faulty processor in a D.S. should be able to reach a common agreement even if certain components in the system are faulty

System Model

(1) There are n processors in the system and at most m of the processors can be faulty.

(2) The processors can directly communicate with other processors by msg passing. System is logically fully connected

(3) A receiver processor always knows the identity of the sender processors of the message

(4) The communication medium is reliable.

Agreement is to be reached b/w only 2 values 0 and 1.

Classification of Agreement Problem

(1) Byzantine Agreement Problem

(2) Consensus Problem

(3) Interactive Consistency Problem

Byzantine Agreement Problem

A single value, which is to be agreed on is initialized by an arbitrary processor and all non faulty processors have to agree on that value.

In this problem an arbitrarily chosen processor called the source processor, broadcast its initial value to all other processors

A soln to the Byzantine agreement problem should meet the following 2 objectives.

Agreement = All Non faulty processor agree on the same value

Validity = If the source procen is non faulty then the Common agreed upon value by all non faulty processors. should be initial value of the source

The Consensus problem:

* Every processor broadcast its initial value to all other processors. Initial value of the processors may be different

A protocol for reaching consensus should meet the following condition.

Agreement: All non faulty processor agree on the same single value

Validity: If the initial value of every non faulty processor is V, then agreed upon Common value by all non faulty processor must be V

The interactive Consistency Problem

Every processor broadcast its initial value to all other processors The initial value of the processor may be different

A protocol for the interactive consistency problem Should meet the following condition.

Agreement: All non faulty processor agree on the same vector $(V_1, V_2 \ldots V_n)$

Validity; If the i^{th} processor is nonfaulty and its initial value is V_i then the ith value to agreed on by all non faulty processor must be V_i

Solution for Byzantine Agreement Problem

⇒ First defined and solved by Lamport

⇒ Source broadcast its initial value to all other processors.

⇒ Processors send their value to other processors and also relay received values to other

⇒ During execution faulty processors may confuse by conflicting values

⇒ If faulty processors dominate in no. they can prevent non faulty processors from reaching an agreement

⇒ No of faulty processors should not exceed certain limit.

⇒ Peas showed that in a fully connected network it is impossible to reach an agreement if no faulty processor 'm' exceeds $(n-1)/3$.

n = no of processors

Lamport Shostak Peas Algorithm:

This algo also known as oral message algorithm OM(m) where m is the no of faulty processor.
n = no of processors and $n \geq 3m+1$

Algo is recursively defined as follow.

Alg. OM(0)

. Source processor send its value to every processor

Each processor ~~send~~ uses the value it receives from source (If no value is received default value 0 is used

Alg OM(m) m>0

1) The source processor sends its value to every processor.

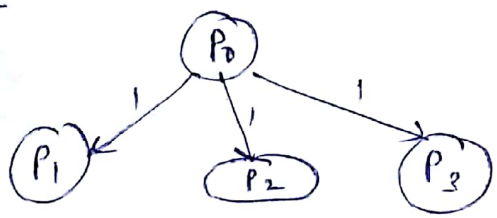
2) for each i, let v_i be the value processor i receives from source.

3) Processor I act as the new source and initiates algo OM(m-1) when it sends the value v_i to each of the (n-2) other processors.

4) for each i and j let v_j be the value processor i received from processor j in step 3. Processor I use the value majority($v_1, v_2 \cdots v_{n-1}$)

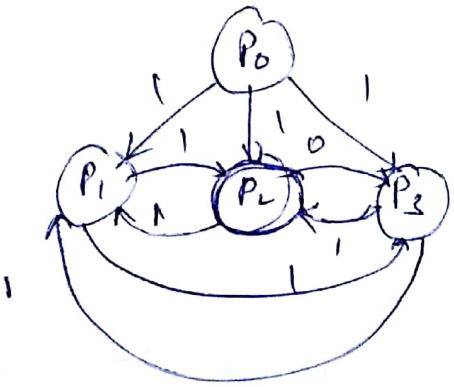
The function majority($v_1, v_2 \cdots v_{n-1}$) computes the majority value if exists otherwise it uses default value 0

Example



⇒ P_2 = faulty process
P_0 = non faulty.

Process P_0 executes the algo OM(1)



Process P_1 P_2 P_3 execute the algo OM(0)

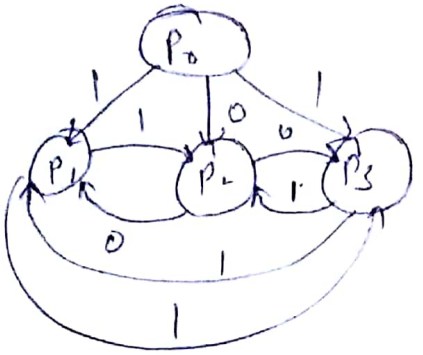
P_1 received = (1 1 1)

P_2 = (1 1 0)

P_3 = (1 1 0)

Majority value = 1

both cond^n of Byzantine Agreement are satisfied



P_0 = faulty

Process $P_0 P_1 P_2 P_3$ execute OM(1) & OM(0)

P_1 received = {1, 0, 1}

P_2 = {0, 1, 1}

P_3 = {1, 0, 1}

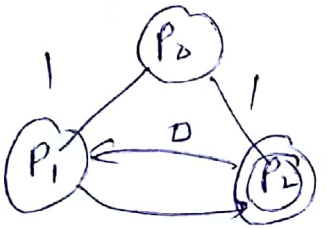
Majority = 1

Nonfaulty process or agree on same value.

Byzantine Agreement can not be reached among three processors if 1 process is faulty

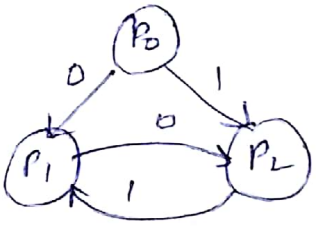
P_0 is some processor

P_0 is Non faulty



P_2 = faulty

P_0 is faulty



Application of Agreement Algo.

1) Fault tolerant Clock Synchronization
   D.S. require physical clocks to synchronized
   Physical clock have drift problem
   Agreement protocol may help to reach a common clock value

2) Atomic Commit in DDBS
   DDBS sites must agree whether to commit & abort the transaction
   Agreement protocol may help to reach a consensus

In the problem of atomic Commit. sites of a DDBS must agree whether to Commit or abort a transaction

Phase First = sites execute their part of a distributed transaction and broadcast their decisions to all other sites

Second Phase = Each site based on what it received from other sites in the 1st phase decides whether to commit or abort its part of the distributed transaction

⇒ Every site receives an identical response from all other sites. they will reach the same decision. If some sites behave maliciously they can send a conflicting response to other sites. causing them to make conflicting decisions

In this situation we can use algorithm for the Byzantine ~~Agreement~~ to insure that all non faulty processor reach a common decision about a distributed transaction.

It work as follow:

In the 1st phase after a site has made a decision it start the Byzantine Agreement

In the 2nd Phase = Processors determine a common decision based on the agreed vector of Phase

Distributed file Systems.

Introduction: A distributed file system is a resource management component of a distributed operating system

It implement a common file system that can be shared by all the autonomous computers in the system

Two important goals of D.F.S.

→ N/w transparency: User donot have to be aware of the location of files to access them. This property of a D-F.S. is known as N/w transparency

→ High Availability: Another major goal of D.F.S is to provide high availability. System failure or regularly scheduled activities such as back ups or maintenance should not result in the unavailability of files

Architecture

* In D.F.S. files can be stored at any machine or the computation can be performed at any machine
* When a machine needs to access a file stored on remote machine, the remote machine performs the necessary file access operations and return data if a read operation is performed

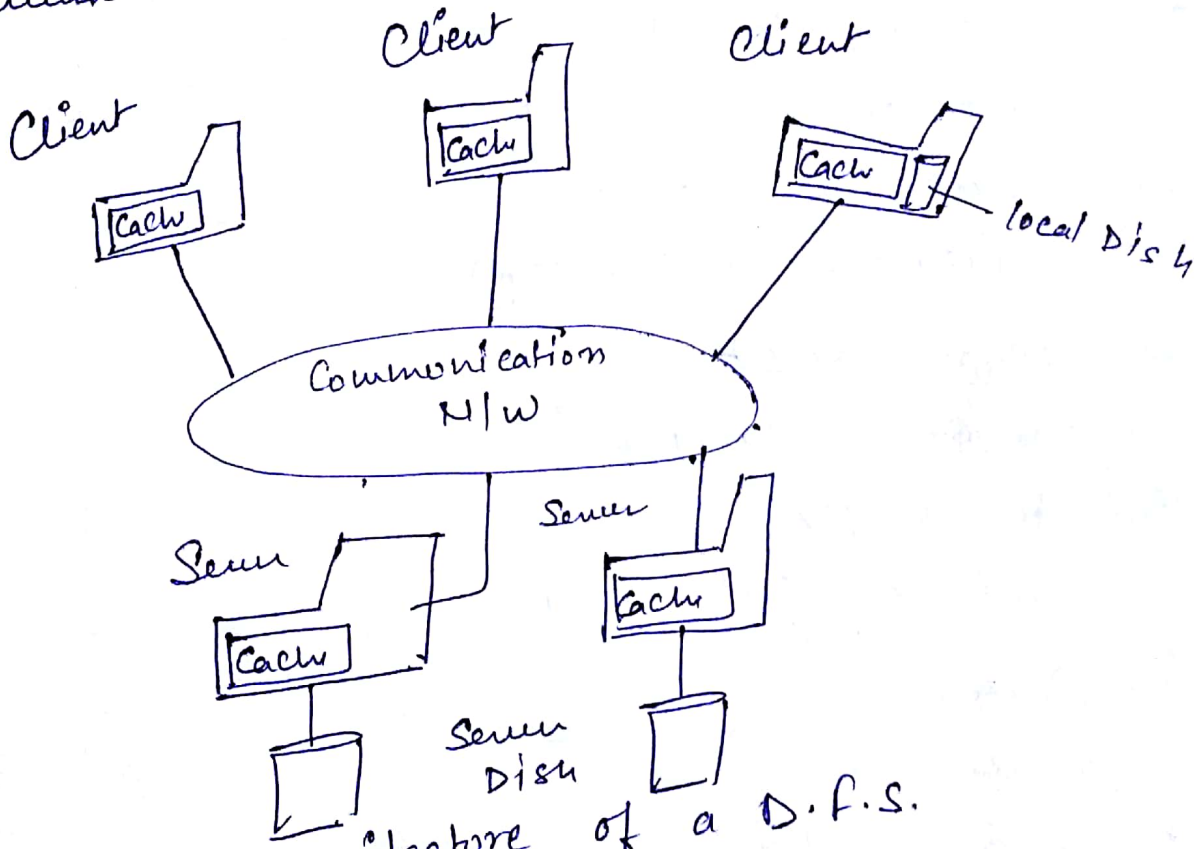
The two most important services present in a D.F.S. are:

Name Server: is a process that maps names specified by client to stored objects such as file and directories.

Cache Manager : is a process that implements file caching.

In file caching, a copy of data stored at a remote file server is brought to the client's machine.

Cache manager can be present at both client and file servers



Architecture of a D.f.s.

Mechanisms for Building D.F.S.

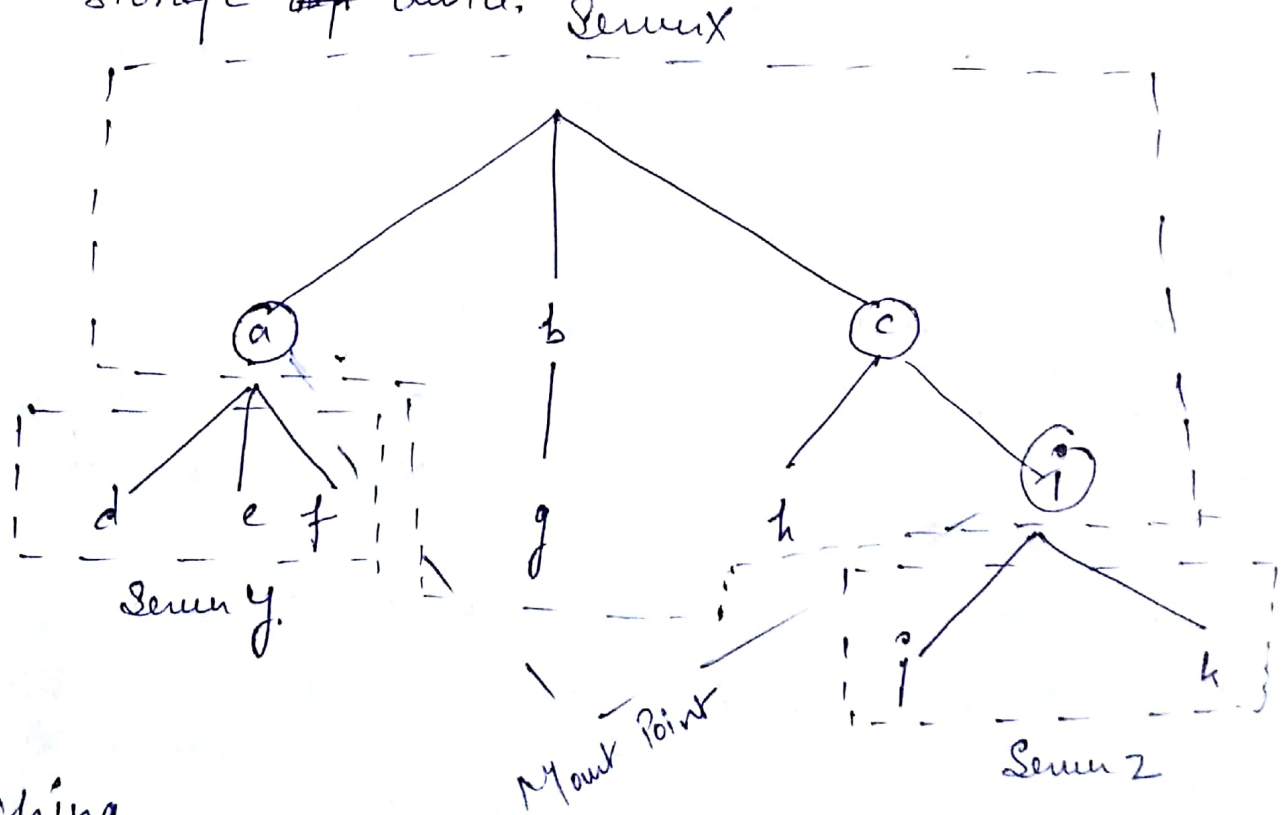
Mounting:

⇒ Mounting: Binding together different filename space to form a single hierarchically structured name space

⇒ A name space can be bounded to or mounted at external node or a leaf node of a name space tree.

~~The mount table~~

* The kernal maintain a structure called the mount table which map. mount point to appropriate storage ~~dev~~ device.



Caching = A common technique to exploit locality and reduce delay.

→ In file caching, a copy of data stored at remote file server is brought to the client when referenced by the client

Hints = Caching without cache coherence Cache coherence operations too expensive in D. Systems.
Hints work well for applications that can recover from individual data.

Bulk Data Transfer = Transferring data in bulk reduce the protocol processing overhead at both server and clients..

Encryption: used to enforcing security in D.S.

Design Issues

(1) Naming and Name Resolution:
- A Name in file system is associated with an object
Name resolution refers to the process of mapping a name to an object
Name space - a collection of names.
There different methods to name file.

(1) Concatenate the host name to the name of files that are stored on that host.
- filename is unique
- No N/w transparency
- Moving a file from 1 machine to another may require modifications to the application

Advantage - Name resolution is simple. - Don't need a name server

(2) Mount Remote directories onto local Directory
- Once mounted, accessing the files become location transparent
Name resolution is simple

(3) Have a single Global directory: all files in the system belongs to a single Name space
- Not scalable, how to maintain a system with unique file names.

NameServers: In a centralized system, name resolution can be accomplished by maintaining a table that maps names to objects.
In distributed systems, name servers are responsible for name resolution

A Name Server is a process that map names to stored object

* The ~~easiest~~ easiest approach to name resolution in D.S. is for all client to send their queries to a single name Server which maps name to objects.
→ Drawback: If the name Server crashes the entire system is drastically affected.
→ (b) Degrade the performance of the System

* Second Approach → Having several name Servers wherein each server is responsible for mapping object stored in Different Domain

Caches on Disk or Main Memory:

Advantage for caching on main memory -
- Can be used in diskless workstation
- accessing a cache in memory is much faster
- Same technique for server cache and client cache

Disadvantage - Compete with virtual memory
- Virtual memory system more complex
Data cannot be in both VM & in cache
large file cannot be cached completely.

Advantage of caching on Disk:
large file can be cached. Virtual memory is simply

Writing Policy: It decides when a modified cache block at a client should be transfered to the server

The simplest Policy - Write Through - In write through all writes requested by the applications at clients are also carried out at the server immediately

The main advantage of write through = reliability
Disadvantage = In the event of client crash, little info. is lost,, Does not take advantage of cache

Delayed Writing Policy - Delays the writing at the server

Modifications done to a write are reflected at the server after some delay.
Access ~~locality~~ Locality: Some data can be deleted right away - 20-30%. data are erased within 30 second.
May loss data when Client Crashes.

⇒ Another writing policy delays the updating of the files at the server until the file is closed at the client.
The traffic at the server depends on the avg. period that files are open.
If the avg period for which files are open is short, then this policy does not greatly benefit from delaying the updates.
If the avg period for which files are open is long this policy is also susceptible to losing data in the event of a client crash.

Cache Consistency

There are 2 approaches to guarantee that the data returned to the client is valid.

(1) Server-initiated: Server informs cache managers whenever the data in the client cache become stale. Cache Manager at client can then retrieves the new data or invalidate the blocks containing the old data in their cache

2) Client-Initiated - the client make sure the cache is clean before returning the data to the application.

* Both the schemes are expensive
* Not allow caching when concurrent write sharing occur.
* Sequential-write sharing may also cause problem a client opens a files that has recently modified and closed by another client

⇒ Out date · Blocks · in the cache: time stamp · the cache copy. Contacting the server can find the inconsistency.

⇒ Update data in another client's cache not in the server: flushing writes when other clients open

Availability

(*) Replication is the main Mechanism

(*) How to keep replicas consistent

(*) How to detect inconsistencies among replicas

(*) Consistency problem may decrease the availability

(*) Replica Management: voting mechanism to read and write to replica

Scalability

* How to meet the demand of a growing system?

* Biggest hurdle: Consistency issue.
  → caching: Reduce N/W load & Server load.
  → Server initiated cache invalidation to maintain cache consistency
  complexity increases as the system size increases.

Semantics

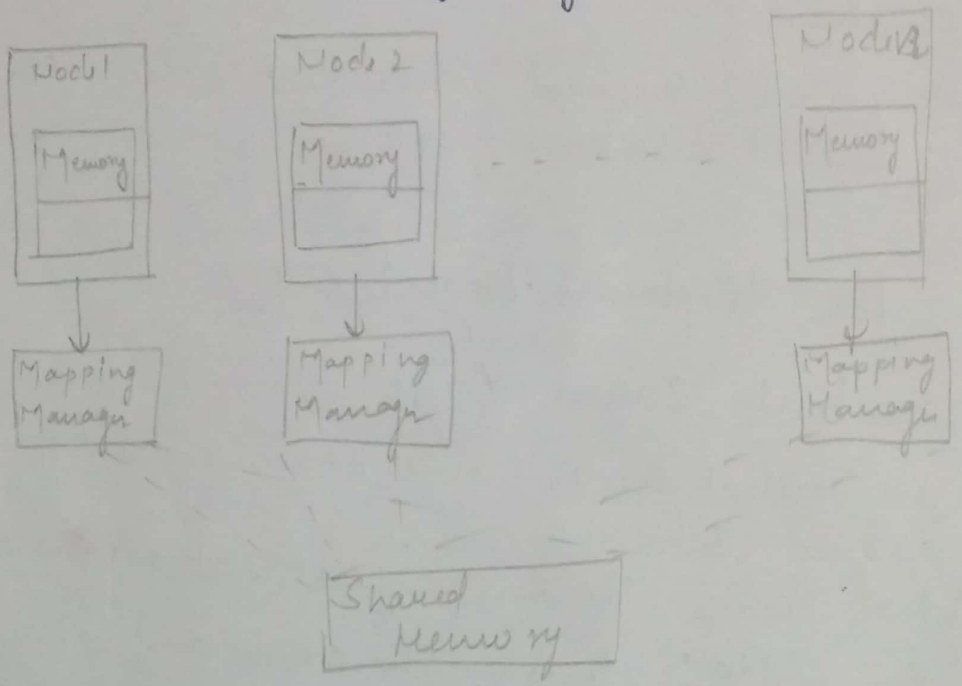
what a user want ? Strict Consistency

Users can usually tolerate a certain degree of errors in file handling — No need to enforce strict consistency

The Read Replication Algo:

Distributed Shared Memory.

DSM is a resource management component of a distributed operating system that implements the shared memory model in distributed system which have no physically shared memory.



Advantages of DSM.

- Programming is a lot easier
  - No need to deal with communication details.
  - Easy to handle complex data structure
- DSM systems are much ~~easier~~ cheaper than tightly coupled multiprocessor system
- DSM takes advantages of the memory reference locality. - Data are moved in the unit of pages
- DSM can form a large physical memory

•) Programs for shared Memory multiprocessors can easily be ported to DSM.

Challenges in DSM:

⇒ How to keep track of the location of remote data?

⇒ How to overcome the communication delay & high overhead?

⇒ How to allow controlled concurrent access to shared data?

Algorithms for Implementing DSM.

(1) The Central-Server Algo

(•) A central server maintains all the shared data.

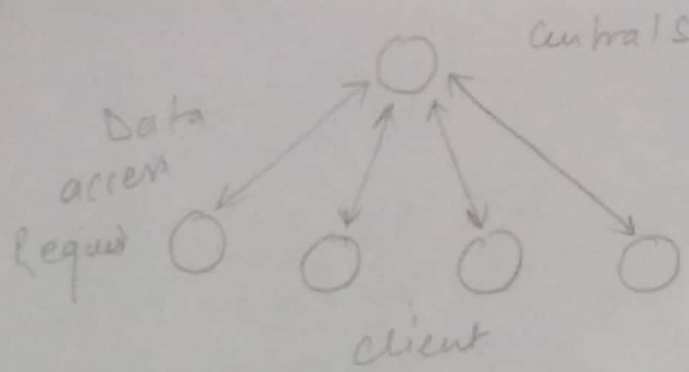
(•) For Read: The server just return the data

(•) For write: Update the data and send acknowledge for distributed applications.

•) A simple working solution to provide shared memory for distributed applicatio.

•) Low efficiency - Bottleneck at the server, long memory access latency

•) Data can be distributed - need a directory to store the location of a Page



The migration Algorithm:

⇒ Data is shipped to the location of the data access request. - subsequent accesses are local.

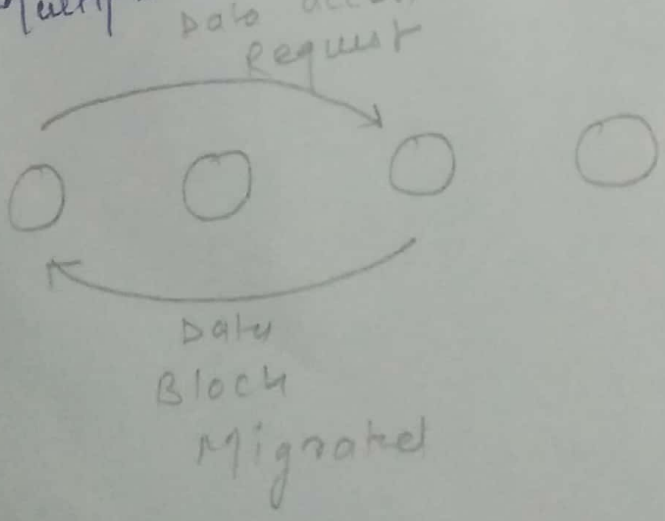
⇒ The migration algo allow only one node to access a shared data at time.

⇒ for both read/write: get the remote page to the local machine, then perform the operation

⇒ keeping track of memory location: Location service

⇒ Problems: thrashing - Pages move b/w nodes frequently. false sharing

⇒ Multiple read can be costly



The Read Replication Algo:

⇒ Extends the migration algo by replicating data blocks and allowing multiple nodes to have read access or one node to have read+write access

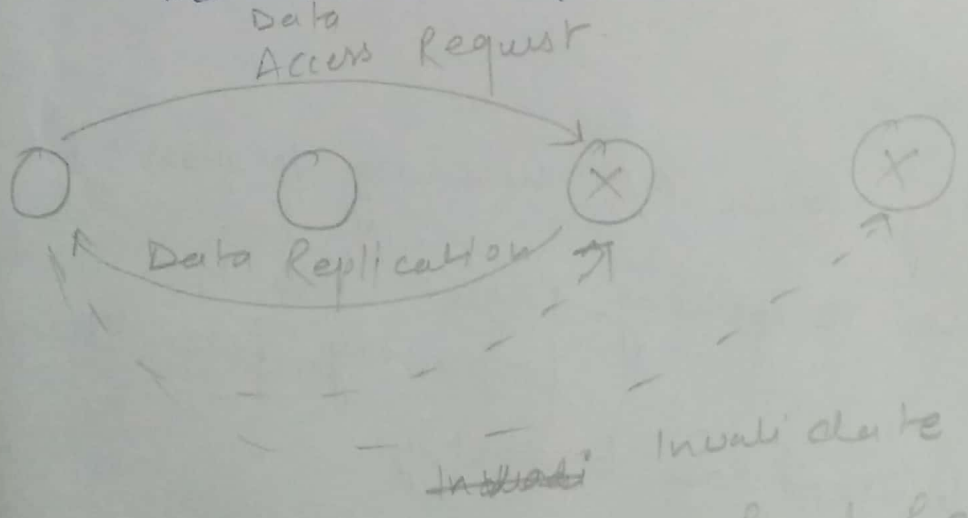
⇒ On Read, a page a replicated. (Mark the page as multiple reader

⇒ On write, all copies except One must be updated or invalidated

⇒ Multiple read one write

⇒ Allowing multiple readers to a page

⇒ All the location must be kept track of:
location server / home machine



write operation in the Read Replication Algo

The Full Replication Algo

⇒ Allow multiple nodes to have both read and write access to shared data blocks (Multiple reader multiple writer protocol)

⇒ Many node can write shared data concurrently the access to shared data must be controlled to maintain its consistency.

⇒ All nodes wishing to modify shared data will send the modifications to a sequencer

⇒ The sequencer will assign a sequence no and multicast the modification with the sequence no to all the nodes that have a copy of the shared data item
Each node processes the modification request in the sequence no. order