Agreement Protocols

Gaurav Parashar

October 13, 2020

Table of Contents I

Table of Contents II

Table of Contents III

Introduction I

- In Distributed Systems, where sites often cooperate to achieve a common. goal, it is often required that sites reach Mutual agreement.

Introduction II

- When a system prone to failure this method does not work. This is because faulty processors can send conflicting value to other processors preventing them from reaching an agreement.

Introduction I

- In the presence of faults, processors must exchange their values with other processors and rely the values received from other processors several times to isolate the effects of faulty processors

The System Model I

Agreement problems have been studied under the following system model

- There are n processors in the system & almost m if the processors can be faulty

Synchronous Vs Asynchronous Computations I

- In synchronous computation, processes in the system run in lock step manner, where in each step, a process receives messages (sent to it in previous step), performs a computation, & sends messages to other processes. Steps of synchronous computation is also referred as a round

Model of Processor failures I

- In agreement problem we consider a very general model of processor failures

Classification of Faults I

Based on components that fault

- Program/process

Based on behaviour of faulty component

Authentication Vs Non Authentication Messages I

- To each an agreement, processors have to exchange their values & relay the received values to other processors several times.

Classification of Agreement Problems I

There are three well known agreement problems in DS:-

- The Byzantine Agreement Problem

Classification of Agreement Problems I

The Byzantine Agreement Problem I

- In the Byzantine agreement problem, an arbitrarily chosen processor, called the source processor, broadcasts its value to all other processors

The Consensus Problem I

- Every processor broadcasts its initial value to all the other processors. The initial value of the processors may be different. A protocol for reaching consensus should meet the following conditions:-

The Interactive Consistency Problem I

- All the non faulty processor agree on the same vector $\{S_1, S_2, \ldots, S_n\}$

Solution to Byzantine Agreement Problem I

- It is obvious that all the processors must exchange the values through messages to reach a consensus. Processors send their values to other processors & relay received value to other processors.

Solution to Byzantine Agreement Problem I

In general, a solution to an agreement problem must pass three tests: termination, agreement, and validity. As applied to the Byzantine General's problem, these three tests are:

- A solution has to guarantee that all correct processes eventually reach a decision regarding the value of the order they have been given.

An impossibility Result I

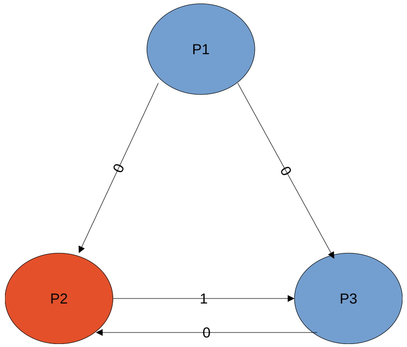

Figure 1: Processor P1 is non faulty

An impossibility Result I

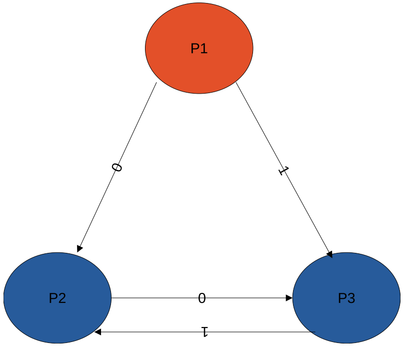

Figure 2: Processor P0 is faulty

The Lamport, Pease and Shostak Algorithm I

In general, a solution to an agreement problem must pass three tests: termination, agreement, and validity. As applied to the Byzantine General's problem, these three tests are:

- A solution has to guarantee that all correct processes eventually reach a decision regarding the value of the order they have been given.

The Lamport, Pease and Shostak Algorithm I

Lamport's algorithm is a recursive definition, with a base case for m=0, and a recursive step for $m > 0$: Algorithm OM(0)

- The general sends his value to every lieutenant.

Algorithm OM(m), $m > 0$

Applications of Agreement Algorithms I

Algorithms for agreement problems find application in problems where processors should reach an agreement regarding their values in the presence of malicious processors. Two such applications are there:-

- Fault Tolerant Clock Synchronzation

Applications of Agreement Algorithms I

- Fault Tolerant Clock Synchronzation

Fault Tolerant Clock Synchronzation

Applications of Agreement Algorithms II

- A non faulty process's clock runs at approximately the correct rate
- A non faulty process can read the clock of another non faulty process with almost a small error ϵ

Applications of Agreement Algorithms I

Atomic Commit in DDBS

Applications of Agreement Algorithms I

- At any time the values of the clocks of all non faulty processes must be approximately equal

Distributed Resource Management I

Introduction A distributed file system is a resource management component of a distributed operating system. It implements a common file system that can be shared by all the autonomous computers in the system. Two important goals of distributed file systems are:-

- Network Transparency

Network Transparency I

- The primary goal of a DFS is to provide the same functional capabilities to access files distributed over a network as the file system of a time sharing system does to access files residing at one location.

High Availability I

- Users should have the same easy access to files, irrespective of their physical location

Architecture of DFS I

- Ideally, in a DFS, files can be stored at any machine and computation can be performed at any machine

Services in DFS I

Name Server: It is a process that maps names specified by the client to stored object files & directories. The mappings occurs when a process references a file or object for the first time

Cache Manager I

Cache Manager II

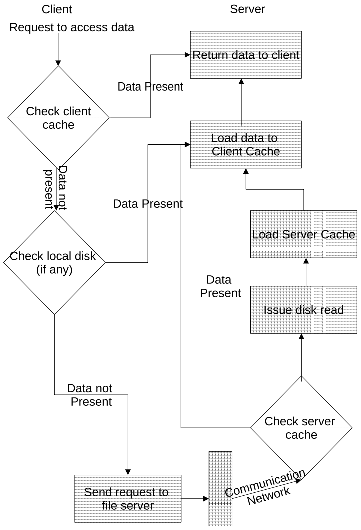

Figure 3: Typical data access actions in DFS

Mechanism for building DFS I

The basic mechanism for building DFS involves following steps:-

- Mounting
- Caching
- Hints
- Bulk data transfer
- Encryption

Mechanism for building DFS I

Mechanism for building DFS II

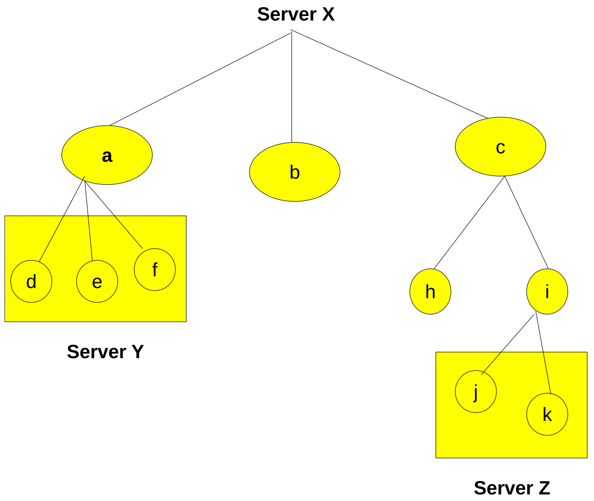

Figure 4: Namespace Hierarchy

Design issues in Design and Implementation of DFS I

- Naming & Name Resolution
- Caches on Disk or Main Memory
- Cache Consistency
- Availability
- Scalability
- Semantics

Naming and Name Resolution I

- A name is associated with an object, such as file or directory
- Name Resolution: mapping a name to an object
- Name Space: A collection of name. There are three methods to name files

  Concatenate the host name to the name of the file on the host:
  - It guarantees that the file is unique systemwide
  - It conflicts with the goal of network transparency
  - One more problem is there, when we transfer a file from one host to another then we need to change the filename and applications those are accessing the file will have issues. Therefore, this naming scheme is not location independent
  - the advantage of this scheme is that just by looking at the filename system can identify the host and host system is not be required to be consulted

Naming and Name Resolution II

Mount remote directories onto local directories: once mounted, accessing becomes location transparent and name resolution is also simple

Have a single Global Directory: where all the files belong to the single namespace. The major disadvantage of this scheme is that there is only one computing facility or few cooperating computing facilities and not stable to maintain unique filenames

Contexts: To overcome the difficulties associated with systemwide unique names the notion of context has been used ti partition a namespace.

- It identifies a namespace in which to resolve a given name.
- contexts can partition the name space along with the following:
  - geographical boundary
  - organisational boundary

Naming and Name Resolution III

    - specific to hosts
    - a file type
- a filename can be thought of as composed of a context and a name local to that context
- resolving a name involves interpreting the name with respect to the given context

Name Server

- In a centralised system, name resolution can be accomplished by maintaining a table that maps names to objects
- In DS, name servers are responsible for name resolution
- a name server is a process that Maos names specified by clients to stored objects such as files and directories
- the easiest approach to name resolution in DS is for all clients. To send their queries to a single name server

Naming and Name Resolution IV

- This approach has some serious drawbacks:
  - if name server crashes- the entire system drastically gets affected
  - name server can become bottleneck due to load

Caches on Disk or Main Memory I

The advantages of having the cache in the main memory are as follows:

- can be used in diskless workstations
- accessing a cache in main memory is much faster than accessing a cache on local disk
- The server cache is in main memory at the server and hence a single design for a caching mechanism is applicable to both clients and servers
- The main disadvantage of having client-cache in main memory is that it competes with the Virtual memory(VM) system for physical memory space.
- VM system is more complex- data cannot be in both VM and cache
- large files can't be cached completely- need to be chopped

Empty Slide I

Writing Policy I

- A writing policy decides when a modified cache at a client should be transferred to the server
- The simplest policy is write through, in this all writes requested by the applications at clients are also carried out at the servers immediately
- the main advantage of the write through is reliability. In the event of client crash littles information is lost
- *An alternative writing policy*, delayed writing policy, delays the writing at the server. In this case modifications due to a write are reflected at the server after some delay.
- This approach can potentially take advantage of the cache by performing many writes on a block present locally in cache.
- Another advantage is that some of the data could be deleted in a short time, in which case data need not be written at the server at all but *this leads to reliability problem*.

Empty Slide I

Cache Consistency I

- The scheme that can guarantee consistency of the data cached at clients.
- there are two approaches to guarantee that the data returned to the client is valid
- In *server-initiated approach*, servers inform cache managers whenever the data in the client caches becomes stale. Cache managers at clients can then retrieve the new data.
- In *client-initiated approach*, it is the responsibility of the cache managers at the clients to validate data with the server before returning it to the clients.

Both these approaches are expensive as they require elaborate cooperation between servers & cache managers. In both the approach

Cost is huge

Empty Slide I

Availability I

- It is one of the important design issues, failure of which affects the availability of files
- It is solved using *replication*
- The most serious problems in replication are consistency of files and find inconsistencies among replicas & recover these inconsistencies

Empty Slide I

Scalability I

- How to meet the demand of the growing system?
- Consistency issue
- The solution is, caching reduce network load and server load
- exploit the *made for file use*(Most shared files are readonly)

Semantics I

- What a user wants? Strict Consistency
- Users can usually tolerate a certain degree of errors in file handling

Introduction I

- Distributed computing has been based on the Message Passing model, in which processes interact & share data with each other by exchanging data in form of messages.
- DSM is a resource management component of Distributed Operating Systems that implements the shared memory model in DS, which have no physical shared memory.
- The shared memory model provides a virtual address space that is shared among all nodes in a DS. Refer Figure 5 for more details.

Introduction II

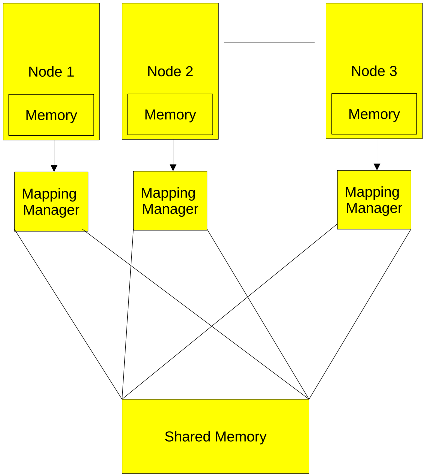

Figure 5: Distributed Shared Memory

Architecture I

- with DSM, programs access data in the shared address space just as they access data in traditional VM.
- Data moves between secondary memory, main memory as as in between memories of different nodes.
- Each node can own data stored in shared address space and ownership may change when data moves from one node to another.
- when a process access data in the shared address space, a mapping manager maps the shared memory address to the physical memory
- The mapping manager is a layer of software implemented either in the OS kernel or as a runtime library routine.
- To reduce delays due to communication latency, DSM may move data at the shared memory address from a remote node to the node that is accessing data.

Advantages of DSM I

- Data sharing is implicit, hiding data movement
- Passing data structures containing pointers is easier
- Moving entire object to user, takes advantage of locality of reference
- Less expensive to build then tightly coupled multiprocessor system
- very large total physical memory for all nodes
- no serial access to common bus for shared physical memory like in multi -processor system
- programs written for shred memory multiprocessors can be tun on DSM system with minimum changes

Algorithms for implementing DSM I

- The *Central-Server Algorithm*
- The *Migration Algorithm*
- The *Read Replication Algorithm*
- The *Full Replication Algorithm*

The central Issue in the implementation of DSM are:-

- How to keep track of the location of remote data?
- How to minimise communication overhead when accessing remote data?
- How to access concurrently remote data at several nodes

The Central-Server Algorithm I

- The central server shown in Figure 6 maintains all the shared data
  - Read Request: returns data items
  - Write Request: update data & returns acknowledgement
- a timeout is used to reset a request of ACK fails
- associated seq numbers can be used to detect duplicate write requests
- if an application's request to access shared data fails repeatedly, a failure condition is sent to the application.
- while the central algorithm is easy to implement, the Client Server become a bottleneck.
- to overcome this problem the shared data can be distributed over several servers.

The Central-Server Algorithm II

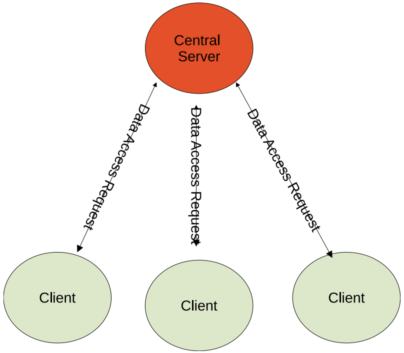

Figure 6: The Central-Server Algorithm

The Migration Algorithm I

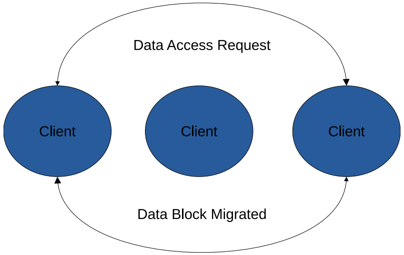

Figure 7: The Migration Algorithm

- Migrate entire data object containing data item to requesting location. Refer Figure 7
- Allow only one node to access a shared data at a time
- Takes advantage of locality of reference

The Migration Algorithm II

- DSM can be integrated with VM provided by the OS at individual nodes
- make a DSM page multiple of VM page size
- a locally held shared memory page can be mapped into the VM page address space
- if page not local, fault handler migrates page & removes it from the address space at remote node
- To locate a remote data object
    - use a location server
    - maintain hints at each node
    - broadcasts query

Issues:-

- only one node can access a data object at a time
- thrashing can occur to minimise it at minimum time data object resides at a node

The Read-Replication Algorithm I

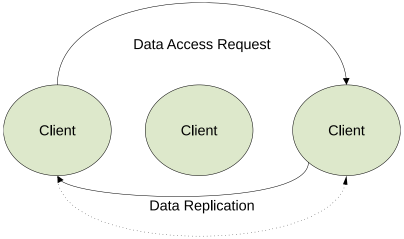

Figure 8: The Read-Replication Algorithm

The Full-Replication Algorithm I

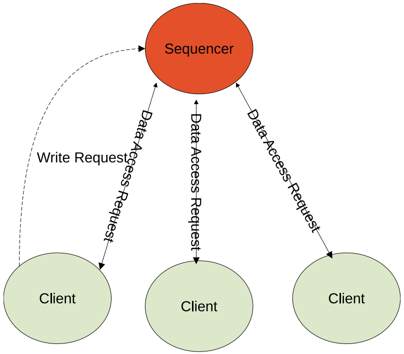

Figure 9: The Full-Replication Algorithm