Distributed System Fault tolerance

Poonam Abraham Lakra

Assistant Professor

Department of MCA

What is Fault Tolerance?

- Fault tolerance is an important issue in the design of a DFS.
- Various types of fault could harm the integrity of the data stored by distributed file system
- Examples -
- a) In event of crash: processors loses the content of the Memory. Such failure s could result in logically complete but physically incomplete file operations. This results in file inconsistency.
- b) In case of request processing between client and server, loss of information state.

What is fault tolerance?

- Similarly due to environmental phenomena such as transient fault caused by electro magnetic fluctuations
- Due to decay of disk storage devices, portion of data got irretrievable.
- The primary file properties that influences the ability of DFS to tolerate the various faults are as follows:

1. Availability – file Replication is primary mechanisms for improving the availability of file. Availability means fraction of time for which the file is available for use. This feature depends on the location of the file and the clients. If some problem exist for some network area, file may be available for the rest of the client‘a node.

Fault tolerance

2. Robustness – It means the power to survive crashes of the storage devices and decays of the storage medium on which it is stored. Redundancy mechanism enable robustness such as stable storage device. This feature is independent of location of file and client.
3. Recoverability – It means the ability to rolled back to an earlier, consistent state. The state when an operations fails or is aborted by the client. Atomic transaction is used to implement recoverable files.

Storage

- In context to crash resistance capabilities, storage may be classified into 3 types:
- Volatile Storage – this kind of storage Cannot withstand any kind of failure , whether it is power failure or system failure. This means that the data stored in a volatile storage is lost in the event of any kind of failure. Example is RAM.
- Nonvolatile Storage – this kind of storage can withstand CPU Failure but cannot withstand transient I/O failure or decay of the storage media. Example is Disk.
- Stable Storage – this kind can withstand both the transient failure and decay of the storage media. The basic idea of stable storage is the use of duplicate storage devices. This way it keep the device stable.

Storage

- The main idea to keep disk 2 advanced exact copy of disk 1, that is each block on disk 2 is an exact copy of corresponding block on disk 1.
- To ensure that both the disk is not damage at the same time, some restrictions on how the two disks are accessed.
- Two basic operations – read & write is Synchronized. A read operation first attempts to Read from disk 1.If it fails, the read is done from disk 2.
- A write operation writes to both disks but write to disk 2 does not start until disk 1 has been successfully completed.

Effects of Service Paradigm on Fault Tolerance

- There are 2 main paradigms by which server may be implemented.

1. **Stateful File Servers** – In this paradigm, a stateful file server maintain s the client's state information. For subsequent requests made by the client to the stateful server for accessing the file, some state information stores the service performed for the client as a Result of first request execution.
   - To allow the file server to decide how long to retain the state information till a session is established. To understand how Stateful file server works :-
   - **Open(filename mode) :** this operation is used to open a file identified by Filename in the specified mode. When the server executes the operation, it creates an entry in the file-table. The file-table maintain the file state information of all thevopen file. This consist of the identifier of the file, the open mode and the read- write pointer which is set to 0.

- Read(*fid*, *n*, *buffer*) – this operation is used to get *n* bytes of data from the file identified by *fid* into specified buffer. When server executes this operation, it returns to the client *n* bytes starting from read-write pointer and then increment by *n*.
- Write(*fid, n, buffer)* - this operation is used on execution of the operation, the server take *n* bytes of data from the specified *buffer,* writes it on the file identified by *fid* specified by currently addressed by read-write pointer and then increment by *n*.

- **Seek(***fid, position)* – this operation cause the server to change the Value if the read-write pointer of the file identified by *fid* to the new value specified as position.
- **Close(***fid)* – this statement cause the server to delete from its file-table the file state information of the file identified by *fid*.

Stateless File Servers

- A stateless file Server does not maintain any client state information.
- Therefore, every request from a client must be accompanied with all the necessary parameters to successfully carry out the desired operation.
- That is, each request identifies the file and the position in the file for read/write access.
- It has 2 operations :
- Read(*filename*, *position, n, buffer)* – this operation specify that the server returnsbto the client *n* bytes of data of the file identified by *filename*. The returned data is placed in the specified *buffer*. The *Position* specify the from where the Reading begins.

Stateless File Server

2. **Write(*filename, position, n, buffer)*** – when the server executes this operation, it takes *n* bytes of data from the specified *buffer*. And writes into the file identified by *filename*. *The position* parameter specified the position within the file from where to start writing. The server returns the actual number of bytes written.

Stateful vs Stateless File Server

- There are certain points which shows significant difference between stateful and stateless file server. They are as follows:-

1. Parameter State –

- Stateful – this remember the client data (state) from one request to next.
- Stateless – this doesn't keep the client data information.

2. Programming

- Stateful – it is harder to code.

- Stateless – it is straight forward to code.
- 3. Efficient
- Stateful – here client donot have to provide full file information every time they perform an operation.
- 4. Crash recovery
- Stateful – difficult due tobloss of information.
- Stateless – can easily recover from failure as there no client state.

- 5. Information transfer
- Stateful – client send less data with each request.
- Stateless – client has to specify complete information with each request.
- 6. Extra Service
- Stateful – it offers file locking mechanism.
- Stateless – no locking mechanism.

Thank you.