Google File System

Om Prakash Mahato

Assistant Director

Nepal Telecommunications Authority (NTA)

Architecture of Google File System

- ***Google File System***

- Google file system is a scalable distributed file system developed by Google to provide efficient and reliable access to data using large clusters of commodity hardware.

- It is designed to meet the rapidly growing demand of Google's data processing need.

- It provides performance, scalability, reliability and availability of data across distributed system for handling and processing big data.

Why built GFS?

1. Node failures happen frequently
2. Files are huge
3. Most files are modified by appending at the end
4. High sustained bandwidth is important than low latency

Characteristics of GFS

1. Files are organized hierarchically in directories and identified by path name.
2. It supports all the general operations on files like read, write, open, delete and so on.
3. It provides atomic append operation known as record append.
4. The concurrent write to the same region are not serializable (traditional).

---

5. It performs two operations : snapshot and record append.

Common goals of GFS

1. Performance
2. Reliability
3. Automation
4. Fault Tolerance
5. Scalability
6. Availability

Limitations of GFS

1. It lacks support for POSIX features.
2. It has high storage overheads.
3. It needs specialization for record append operation.

GFS Architecture

The basic architecture of Google File System is shown in given figure:

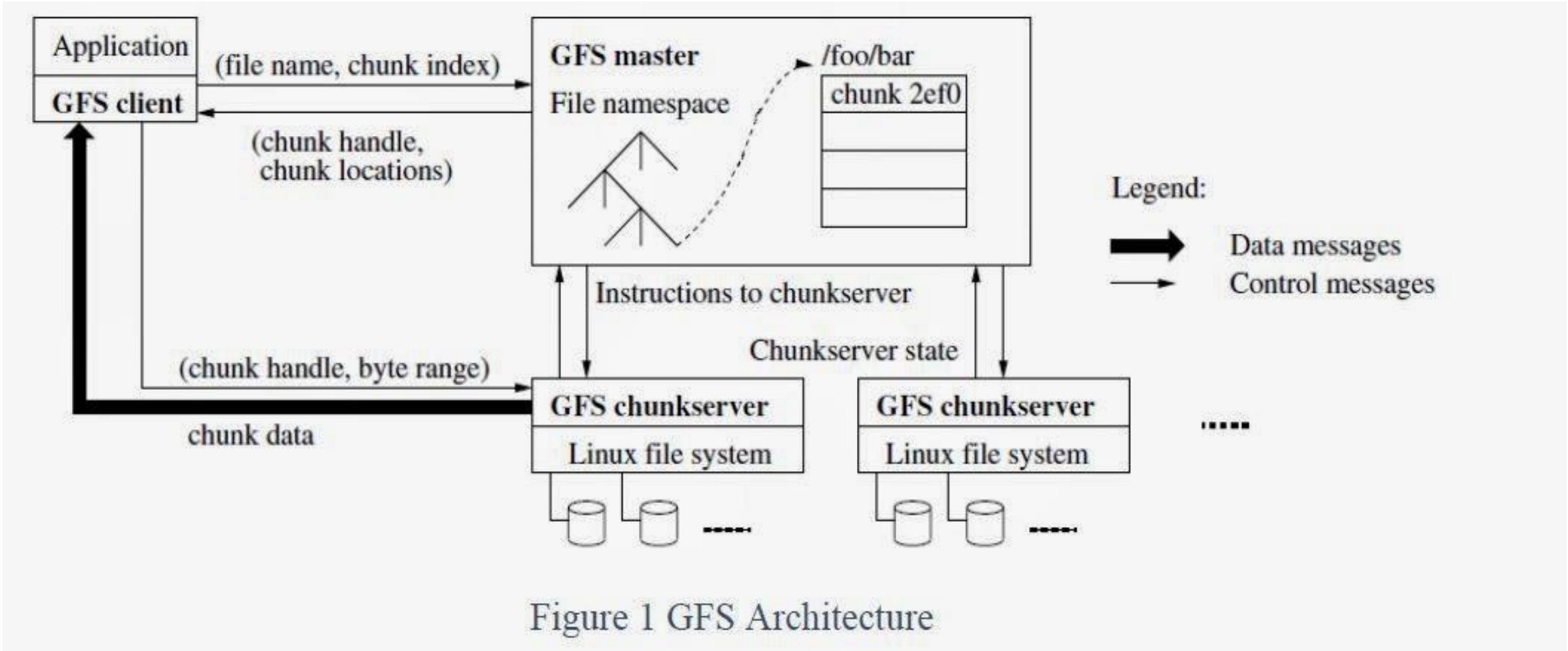

Figure 1 GFS Architecture

1. Master Node:

- It is responsible for the activities of the system such as managing chunk leases, load balancing and so on.
- It maintains all the file system metadata.
- It contains operation log that stores namespaces and file to chunk mappings.
- It periodically communicates with chunk server to determine chunk locations and assesses state of the overall system.
- Each node on the namespace tree has its own read-write lock to manage concurrency.

2. Chunk and Chunk Server

- The files are divided into fixed sized chunks.
- Each chunk has immutable and globally unique 64-bit chunk handle.
- Chunk server is responsible to store chunks on local disk as a linux files.
- By default, each chunk is replicated 3 times across multiple chunk servers.
- The size of chunk is 64 MB.
- Due to such large chunk, it results in space wastage because of internal fragmentation.
- The advantages of large chunk size are as follows:

a) It reduces client's need to interact with master. It means read or write in a single chunk requires only one request to master.

b) It reduces network overhead by keeping a persistent TCP connection to the chunk server for multiple operations performed by client.

c) It reduces the size of metadata stored in the master. It enables storage of metadata in memory.

3. Client Node

- Client node is linked with the application that implements GFS API.
- It communicates with the master and the chunk server to read or write data.
- Client communicates with master to get the metadata.
- For read and write, client directly communicates with the chunk server.

Operation Log and Meta Data

- Operation log is the persistent records of metadata.
- It defines the logical timeline about serialized order of concurrent operations.
- The state is recovered by master by replaying the operation log.
- The metadata stored in GFS master are as follows:

1. Namespace (directory hierarchy)
2. Access control information per file
3. Mapping from file to chunk
4. Current location of chunks (Chunk servers)

Reasons for Single Master in GFS

- Single master design simplifies the GFS design.

- It enables the master to make sophisticated chunk placement and replication decisions.

- Since, the master interacts with client for sharing metadata only, there is no necessity to have multiple master.

- All the heavy processes of read and write is handled by the chunk servers. So, lot of chunk servers are necessary for performance of the system.

Manage overloading of the Master

- The read and write operation is completely separated from master. Instead, these operations are performed by the chunk servers.
- For reliability, master state is replicated on multiple machines, using the operation logs and checkpoints.
- If master fails, GFS starts a new master process at any of these replica.
- Such replica of master state is known as shadow master.
- Shadow master is also able to perform read only access to the file system even when the primary master is down, but it lags the primary master by few fractions of second.

Consistency Model of GFS

- A file region is said to be consistent if all the clients will see the same data each time they load data from any replica of that file region.
- A file region is said to be defined after a file data mutation if it is consistent and the client is able to see what the mutation writes in it.
- The general consistency model is shown in given figure:

	Write	Record Append
serial success	defined	defined interspersed with inconsistent
concurrent success	consistent but undefined	
failure	inconsistent	

Interactions in GFS

1. Leases and Mutation Order

- A mutation is an operation that changes the metadata of the chunks.
- It is resulted due to the write or append operations to the file.
- Leases are used to maintain consistent mutation order across replicas.
- The master grants chunk lease to one of the replica.
- The replica is known as primary replica.
- Primary replica is responsible to design the order for all the mutations to the chunk.

2. Record Append

- It is the atomic append operation provided by GFS.
- The client specifies the data only. The GFS automatically appends it to the file at least once at any offset chosen by the GFS itself and returns the offset to the client.
- The client then pushes the data to all the replicas of the last chunk of that file and then sends request to the primary replica.
- The primary replica checks whether appending process will exceed the chunk size. If it exceeds, then it pads the chunk to the maximum size, tells all other replicas to do so and replies client indication re operation on next chunk. Otherwise, the data is appended on the chunk. and replies the client with success.
- For operation to be success, the data should be written at the same offset on all the replicas of same chunk.

3. Snapshot:

- The snapshot operation is responsible to makes a copy of a file or a directory tree.
- The client uses snapshot to create branch copies of huge data sets or to checkpoint the current state.
- When the master receives snapshot request, it revokes any leases on the chunks. After the leases have been revoked or expired, the master logs the operation to disk.
- The master then apply log record to in-memory state by duplicating the metadata for the source file or directory tree.
- The created snapshot points at the same chunk as the main files.

Write Control and Data Flow in GFS

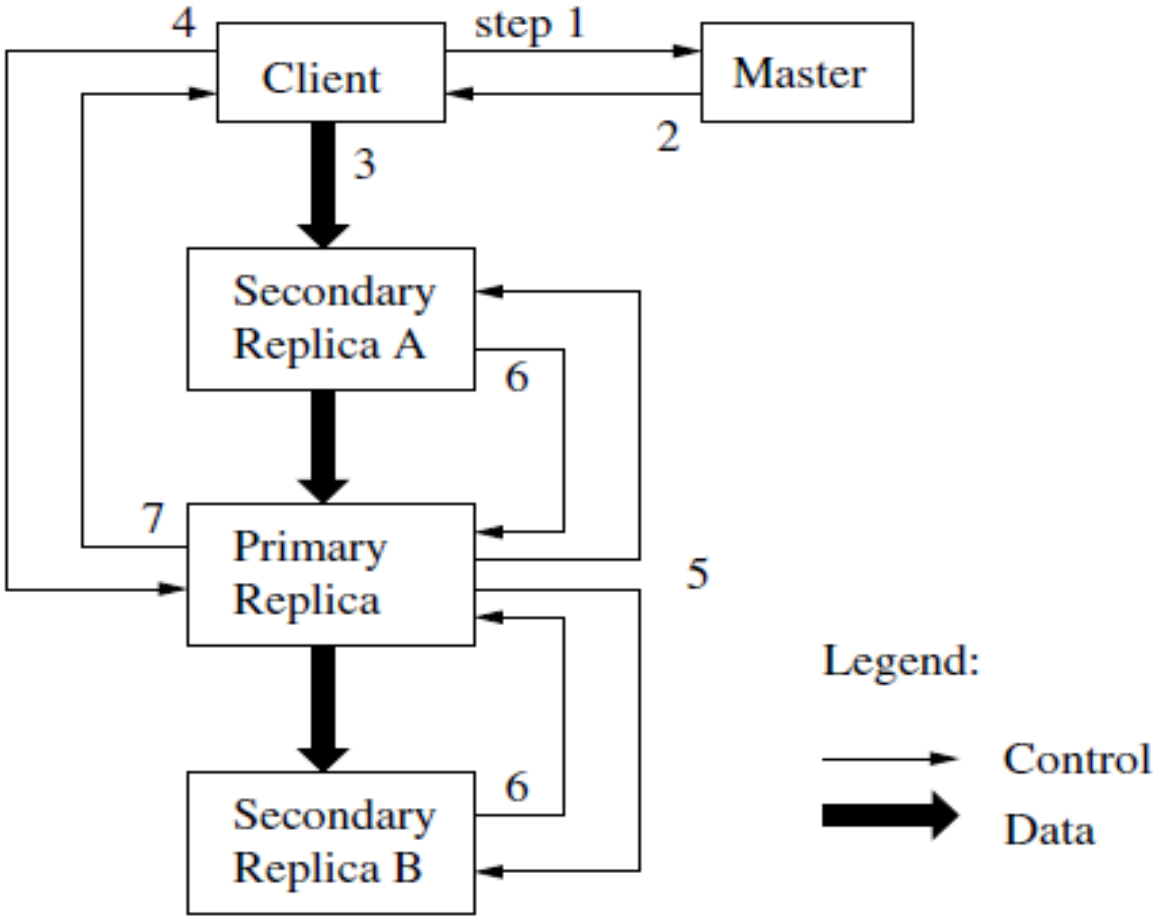

Figure 2: Write Control and Data Flow

1. Client asks the master which chunk server holds the current lease for the chunk and the locations of the other replicas. If no one has a lease, the master grants one to a replica it chooses.
2. Master replies with the identity of the primary and the locations of the other (secondary) replicas. The client caches this data for future mutations. It needs to contact the master again only when the primary becomes unreachable or replies that it no longer holds a lease.

Write Control and Data Flow in GFS

3. The client pushes the data to all the replicas. A client can do so in any order. Each chunk server will store the data in an internal LRU buffer cache until the data is used or aged out.

4. Once all the replicas have acknowledged receiving the data, the client sends a write request to the primary. The request identifies the data pushed earlier to all of the replicas. The primary assigns consecutive serial numbers to all the mutations it receives, possibly from multiple clients, which provides the necessary serialization. It applies the mutation to its own local state in serial number order.

5. The primary forwards the write request to all secondary replicas. Each secondary replica applies mutations in the same serial number order assigned by the primary.

6. The secondaries all reply to the primary indicating that they have completed the operation.

7. The primary replies to the client. Any errors encountered at any of the replicas are reported to the client.

Fault Tolerance in GFS

Garbage Collection

- Whenever the client deletes the file, the file is not deleted but renamed to hidden file with delete timestamp.
- During regular scan of file namespace, hidden files are removed if existed for more than 3 days.
- Until that time, it is possible to undelete the file.
- When it is removed, the memory metadata stored by master is erased.
- Master and chunk server exchanges information about the files with the Heartbeat message.

Garbage Collection

- Advantages:
  - Simple and reliable in distributed system
  - Uniform and dependable way to clean up replicas
  - Performed in batches only when the master is free
  - Provides protection against accidental deletion
- Disadvantages:
  - The storage can not be used immediately
- ***Stale Replica***

- A replica is said to be stale if a chunk server fails and misses mutations to the chunk while it is down.
- The master is responsible to maintain a chunk version number to distinguish between up-to-date replica and stale replica.