**Google File Systems**

**Introduction:**

Google’s data processing requirements has led the company to work on a new design process, a new design of file system to meet their demands. They met with the demands like component failures, files with large size, scalable systems and a better Application Interface to work with the file systems. The result is Google file systems.

GFS runs on commodity hardware, that offers horizontal scaling, a novel File system API more convenient to work with, considering fault tolerance systems as a matter of high importance with continual monitoring of the processes. The file storage is done by splitting file into chunks of 64 MB and storing in the file systems.

**Design Overview:**

**Interface:** GFS uses a familiar file system interface and has snapshot, record append features in it.

**Snapshot** -creates a copy of file/directory at low cost. **Record append**- allows multiple clients to append data of each individual client simultaneously maintaining the atomicity as well.

**Architecture:**

**Single Master:** Uses only one master. Client makes a request with the master to know the file location. Master replies with the chunk name and locations. Master does not interact with chunk fileserver for client requests (Data requests).

**Chunk Size:** A 64 MB is the maximum file size of chunks in chunk file server. Huge files are split into small chunks for better accessibility and storage purposes. When multiple clients access a particular chunk, hotspots are created on the chunk. A better improvement is to ask clients contact other clients who are currently using for the chunk.

**Metadata:** Master stores meta data information namely file, chunk namespace, location of each chunk’s replica, mapping from files to chunks.

**In Memory Data Structures:** master stores data in memory and so re replication in the presence of chunk server failures, chunk migration to balance load, disk space usage across chunk servers could be monitored periodically.

**Chunk Locations:** Master uses heart beat messages to verify the status of chunk server and manage the chunk locations efficiently.

**Operation Log:** Provides details about the status, checkpoint, location of chunks at a certain point of execution. This is really helpful in times of failover wherein, we can locate the checkpoint and find the chunk involved with the issue.

**Consistency:** Guarantee by GFS: Master uses namespace locking to achieve atomicity and correctness of information written to the file, during file mutations.

**System Interactions:**

**Leases and Mutation Order:** File modification is called as mutation and it is achieved uniformly across all the replicas using leases. Lease is a 60 second time out state that reduces operational overhead of master as well as provides time for the replicas to update the mutation.

**Data Flow:** Data flows from client to primary and then to other secondaries. This helps utilize full network bandwidth and avoid network bottleneck. A linear distribution helps achieve this unlike a tree.

Atomic Record Appends and snapshot are some of the other system interactions involved in GFS.

**Master Operation:**

Using Namespace management and locking: GFS uses namespace as a lookup table mapping full pathnames to metadata. This helps achieve atomicity and locking.

**Replica Placement:** Chunk servers are distributed in the server racks and so communication between chunks may have latency due to network. In order to avoid this, a replica of the chunk is placed closer to the server. Write traffic is a tradeoff for this scenario, though we achieve high availability and reliability.

Creating, re-replicating, rebalancing: Chunk replicas serve the following purposes efficiently,

1. To place replicas on chunk servers with below average utilizations
2. Limit the number of recent creations on each chunk server.
3. Latency due to network traffic

**Fault Tolerance and diagnosis:**

**High Availability:** As chunk replication is carried out, failover of chunk servers, does not impact the process that run across systems at any point of time.

**Master replication**: Master is replicated for reliability. Its operation logs are replicated to several machines. Shadow masters could be helpful in providing read only access, when a master in GFS fails over.

**Data Integrity:** Checksum operation is carried out to verify the integrity of data in GFS. Thus data corruption and disk failures are identified from data issues.

**Diagnostic Tools:** RPC logs include exact details about the request and responses sent over wire. These logs are written sequentially and asynchronously.

**Related Works:** GFS closely resembles NASD (Network attached Storage Drives).

**Conclusion:** GFS meets the needs of Google as a storage platform, providing a high reliability over fault tolerance, scalability and data storage.