(/learn)

Summary: GFS

Here is a quick summary of Google File System for you!

We'll cover the following

Summary#

- GFS is a scalable distributed file storage system for large data-intensive applications.
- GFS uses commodity hardware to reduce infrastructure costs.
- GFS was designed with the understanding that system/hardware failures can and do occur.
- Reading workload consists of large streaming reads and small random reads. Writing workloads consists of many large, sequential writes that append data to files.
- GFS provides APIs for usual file operations like `create`, `delete`, `open`, `close`, `read`, and `write`. Additionally, GFS supports `snapshot` and record `append` operations. Snapshot creates a copy of the file or directory tree. Record append allows multiple clients to append data to the same file concurrently while guaranteeing atomicity.
- A GFS cluster consists of a **single master** and **multiple ChunkServers** and is accessed by multiple clients.

- **Chunk**: Files are broken into fixed-size chunks where each chunk is 64 megabytes in size. Each chunk is identified by an immutable and globally unique **64-bit chunk handle** assigned by the master at the time of chunk creation.
- ChunkServers store chunks on the local disk as Linux files.
- For reliability, each chunk is replicated on multiple ChunkServers.
- Master server is the coordinator of a GFS cluster and is responsible for keeping track of all the filesystem metadata. This includes namespace, authorization, mapping of files to chunks, and the current location of chunks.
- Master keeps all metadata in memory for faster operations. For fault tolerance and to handle a master crash, all metadata changes are written to the disk onto an **operation log**. This operation log is also replicated onto remote machines.
- The master does not keep a persistent record of which ChunkServers have a replica of a given chunk. Instead, the master asks each ChunkServer about what chunks it holds at master startup or whenever a ChunkServer joins the cluster.
- **Checkpointing**: The master's state is periodically serialized to disk and then replicated so that on recovery, a master may load the checkpoint into memory, replay any subsequent operations from the operation log, and be available again very quickly.
- **HeartBeat**: The master communicates with each ChunkServer through Heartbeat messages to pass instructions to it and collects its state.
- **Client**: GFS client code which is linked into each application, implements filesystem APIs, and communicates with the cluster. Clients interact with the master for metadata, but all data transfers happen directly between the client and ChunkServers.
- **Data Integrity**: Each ChunkServer uses checksumming to detect the corruption of stored data.
- **Garbage Collection**: After a file is deleted, GFS does not immediately reclaim the available physical storage. It does so only lazily during

regular garbage collection at both the file and chunk levels.

- **Consistency**: Master guarantees data consistency by ensuring the order of mutations on all replicas and using **chunk version numbers**. If a replica has an incorrect version, it is garbage collected.
- GFS guarantees **at-least-once writes** for writers. This means that records could be written more than once as well (although rarely). It is the responsibility of the readers to deal with these duplicate chunks. This is achieved by having checksums and serial numbers in the chunks, which help readers to filter and discard duplicate data.
- **Cache**: Neither the client nor the ChunkServer caches file data. Client caches offer little benefit because most applications stream through huge files or have working sets too large to be cached. However, clients do cache metadata.

System design patterns#

Here is a summary of system design patterns used in GFS.

- **Write-Ahead Log**: For fault-tolerance and in the event of a master crash, all metadata changes are written to the disk onto an operation log which is a write-ahead log.

- **HeartBeat**: The GFS master periodically communicates with each ChunkServer in HeartBeat messages to give it instructions and collect its state.

- **Checksum:** Each ChunkServer uses checksumming to detect the corruption of stored data.

References and further reading#

- GFS paper (https://research.google/pubs/pub51/)
- Bigtable (https://research.google/pubs/pub27898/)
- GFS: Evolution on Fast-forward (https://queue.acm.org/detail.cfm?id=1594206)

← Back
Criticism on GFS

Next →
Quiz: GFS

✓ Mark as Completed

Report an Issue