Metadata

Let's explore how GFS manages the filesystem metadata.

We'll cover the following

The master stores three types of metadata:

1. The file and chunk namespaces (i.e., directory hierarchy).
2. The mapping from files to chunks.
3. The locations of each chunk’s replicas.

There are three aspects of how master manages the metadata:

1. Master keeps all this metadata in memory.
2. The first two types (i.e., namespaces and file-to-chunk mapping) are also persisted on the master’s local disk.
3. The third (i.e., chunk replicas’ locations) is not persisted.

Let’s discuss these aspects one by one.

Storing metadata in memory#

Since metadata is stored in memory, the master operates very quickly. Additionally, it is easy and efficient for the master to periodically scan through its entire state in the background. This periodic scanning is used to implement three functions:

1. Chunk garbage collection
2. Re-replication in the case of ChunkServer failures
3. Chunk migration to balance load and disk-space usage across ChunkServers

As discussed above, one potential concern for this memory-only approach is that the number of chunks, and hence the capacity of the whole system, is limited by how much memory the master has. This is not a serious problem in practice. The master maintains less than 64 bytes of metadata for each 64 MB chunk. Most chunks are full because most files contain many chunks, only the last of which may be partially filled. Similarly, the file namespace data typically requires less than 64 bytes per file because the master stores file names compactly using **prefix compression**.

If the need for supporting an even larger file system arises, the cost of adding extra memory to the master is a small price to pay for the simplicity, reliability, performance, and flexibility gained by storing the metadata in memory.

Chunk location#

The master does not keep a persistent record of which ChunkServers have a replica of a given chunk; instead, the master asks each chunk server about its chunks at master startup, and whenever a ChunkServer joins the cluster. The master can keep itself up-to-date after that because it controls all chunk placements and monitors ChunkServer status with regular HeartBeat messages.

By having the ChunkServer as the ultimate source of truth of each chunk's location, GFS eliminates the problem of keeping the master and ChunkServers in sync. It is not beneficial to maintain a consistent view of chunk locations on the master, because errors on a ChunkServer may cause chunks to vanish spontaneously (e.g., a disk may go bad and be disabled, or ChunkServer is renamed or failed, etc.) In a cluster with hundreds of servers, these events happen all too often.

Operation log#

The master maintains an operation log that contains the namespace and file-to-chunk mappings and stores it on the local disk. Specifically, this log stores a historical record of all the metadata changes. Operation log is very important to GFS. It contains the persistent record of metadata and serves as a logical timeline that defines the order of concurrent operations.

For fault tolerance and reliability, this operation log is replicated on multiple remote machines, and changes to the metadata are not made visible to clients until they have been persisted on all replicas. The master batches several log records together before flushing, thereby reducing the impact of flushing and replicating on overall system throughput.

Upon restart, the master can restore its file-system state by replaying the operation log. This log must be kept small to minimize the startup time, and that is achieved by periodically checkpointing it.

Checkpointing#

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. To further speed up the recovery and improve availability,

GFS stores the checkpoint in a compact B-tree like format that can be directly mapped into memory and used for namespace lookup without extra parsing.

The checkpoint process can take time, therefore, to avoid delaying incoming mutations, the master switches to a new log file and creates the new checkpoint in a separate thread. The new checkpoint includes all mutations before the switch.

← Back

Single Master and Large Chunk Size

Next →

Master Operations

Mark as Completed

Report an Issue