GFS Consistency Model and Snapshotting

This lesson will explain how GFS handles the consistency of its operations and data. Additionally, we will look into how GFS implements a snapshotting operation.

We'll cover the following

GFS consistency model#

To keep things simple and efficient, GFS has a relaxed consistency model.

Metadata operations (e.g., file creation) are atomic. They are handled exclusively by the master. Namespace locking guarantees atomicity and correctness, whereas the master's operation log defines a global total order of these operations.

In data mutations, there is an important distinction between `write` and `append` operations. Write operations specify an offset at which mutations should occur, whereas `appends` are always applied at the end of the file. This means that for the `write` operation, the offset in the chunk is predetermined, whereas for `append`, the system decides. Concurrent writes to the same location are not serializable and may result in corrupted regions of the file. With `append` operations, GFS guarantees the append will happen at-least-once and atomically (that is, as a contiguous

sequence of bytes). The system does not guarantee that all copies of the chunk will be identical (some may have duplicate data).

Snapshotting#

A snapshot is a copy of some subtree of the global namespace as it exists at a given point in time. GFS clients use snapshotting to efficiently branch two versions of the same data. Snapshots in GFS are initially **zero-copy**. This means that data copies are made only when clients make a request to modify the chunks. This scheme is known as **copy-on-write**.

When the master receives a snapshot request, it first revokes any outstanding leases on the chunks in the files to snapshot. It waits for leases to be revoked or expired and logs the snapshot operation to the operation log. The snapshot is then made by duplicating the metadata for the source directory tree. Newly created snapshot files still point to the original chunks.

When a client makes a request to write to one of these chunks, the master detects that it is a copy-on-write chunk by examining its reference count (which will be more than one). At this point, the master asks each ChunkServer holding the replica to make a copy of the chunk and store it locally. These local copies are made to avoid copying the chunk over the network. Once the copy is complete, the master issues a lease for the new copy, and the write proceeds.

← Back

Anatomy of an Append Operation

Next →

Fault Tolerance, High Availability, and ...

Mark as Completed

Report an Issue