Anatomy of a Write Operation

Let’s learn how GFS handles a write operation.

We'll cover the following

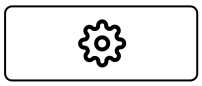
What is a chunk lease?#

To safeguard against concurrent writes at two different replicas of a chunk, GFS makes use of chunk lease. When a mutation (i.e., a write, append or delete operation) is requested for a chunk, the master finds the ChunkServers which hold that chunk and grants a chunk lease (for 60 seconds) to one of them. The server with the lease is called the primary and is responsible for providing a serial order for all the currently pending concurrent mutations to that chunk. There is only one lease per chunk at any time, so that if two write requests go to the master, both see the same lease denoting the same primary.

Thus, a global ordering is provided by the ordering of the chunk leases combined with the order determined by that primary. The primary can request lease extensions if needed. When the master grants the lease, it increments the chunk version number and informs all replicas containing that chunk of the new version number.

Data writing#

The actual writing of data is split into two phases:

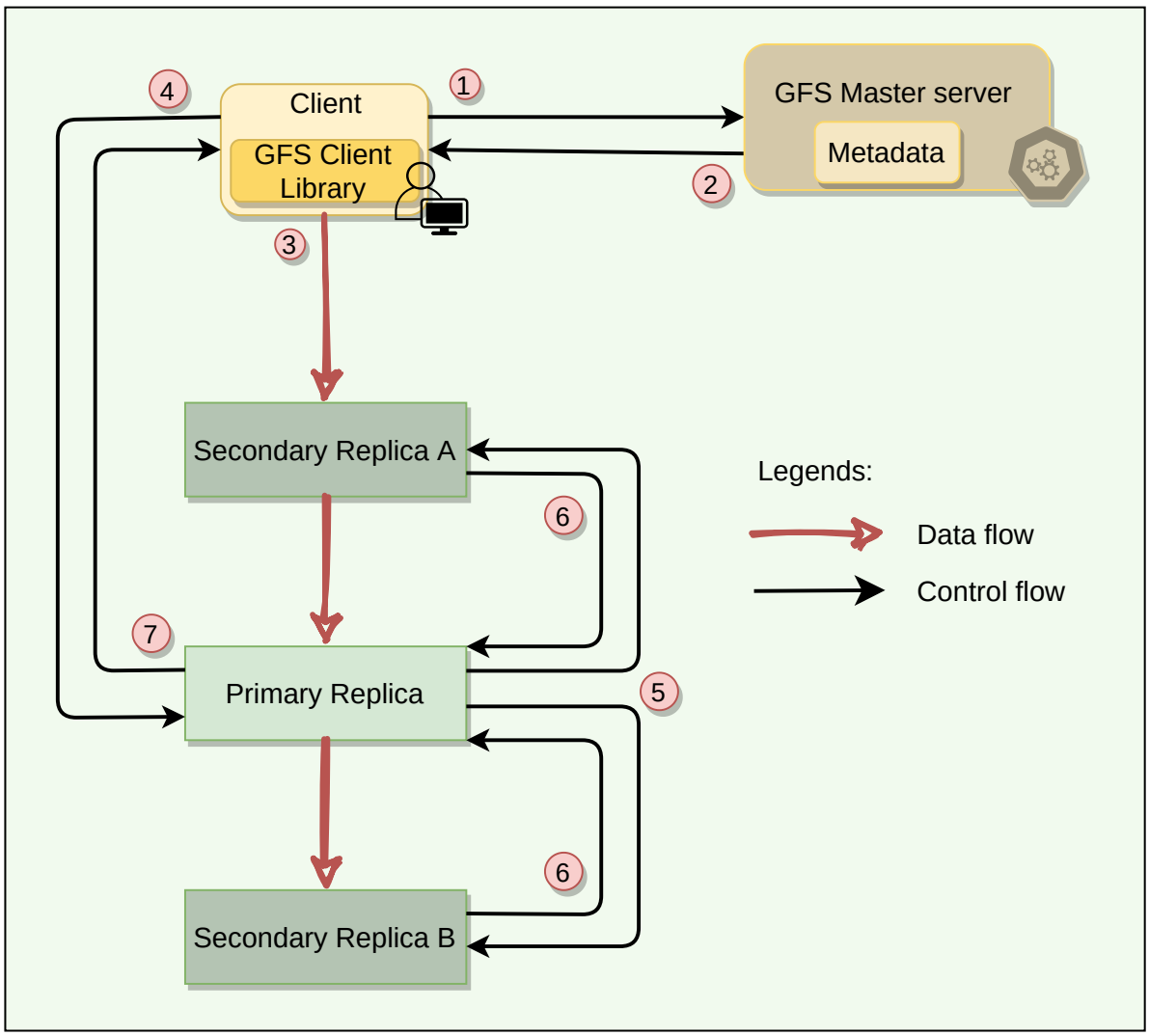


- **Sending**: First, the client is given a list of replicas that identifies the primary ChunkServer and secondaries. The client sends the data to the closest replica. Then replicas send the data in chain to all other replicas to maximize bandwidth and throughput. Eventually, all the replicas get the data, which is not yet written to a file but sits in a cache.
- **Writing**: When the client gets an acknowledgment from all replicas that the data has been received, it then sends a write request to the primary, identifying the data that was sent in the previous phase. The primary is responsible for the serialization of writes. It assigns consecutive serial numbers to all write requests that it has received, applies the writes to the file in serial-number order, and forwards the write requests in that order to the secondaries. Once the primary gets acknowledgments from all the secondaries, the primary responds back to the client, and the write operation is complete. Any errors at any stage in this process are met with retries and eventual failure. On failure, an error is returned to the client.

Following is the stepwise breakdown of the data transfer:

1. Client asks master which chunk server holds the current lease of chunk and locations of other replicas.
2. Master replies with the identity of primary and locations of the secondary replicas.
3. Client pushes data to the closest replica. Then replicas send the data in chain to all other replicas.
4. Once all replicas have acknowledged receiving the data, the client sends the write request to the primary. The primary assigns consecutive serial numbers to all the mutations it receives, providing serialization. It applies mutations in serial number order.
5. Primary forwards the write request to all secondary replicas. They apply mutations in the same serial number order.

6. Secondary replicas reply to primary indicating they have completed operation.
7. Primary replies to the client with success or error message



The anatomy of a write operation

The key point to note is that the data flow is different from the control flow. The data flows from the client to a ChunkServer and then from that ChunkServer to another ChunkServer, until all ChunkServers that store replicas for that chunk have received the data. The control (the write request) flow goes from the client to the primary ChunkServer for that chunk. The primary then forwards the request to all the secondaries. This ensures that the primary controls the order of writes even if it receives multiple concurrent write requests. All replicas will have data written in

the same sequence. Chunk version numbers are used to detect if any replica has stale data which has not been updated because that ChunkServer was down during some update.

← Back

Anatomy of a Read Operation

Next →

Anatomy of an Append Operation

Mark as Completed

Report an Issue