Members: -
Gagan Agarwal (2021201009)
Lalit Gupta (2021201018)
Sarthak Rawat (2021202006)
Shreyash Agrawal (2021201074)

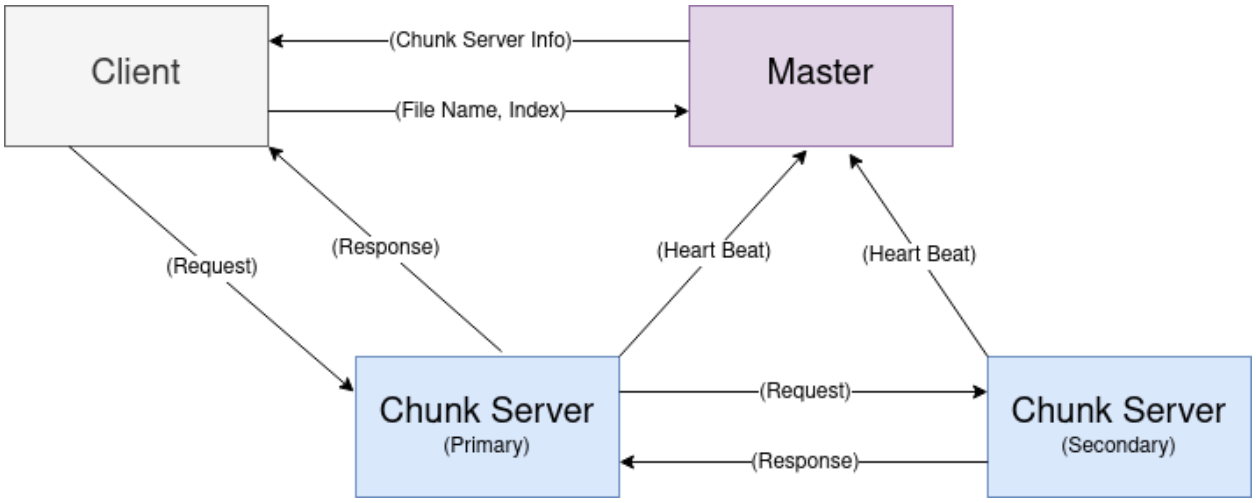
Distributed Systems
Google File System

Introduction

Google File System (GFS). It is a proprietary distributed file system developed by Google for storing and managing large amounts of data across a cluster of commodity hardware.

GFS is designed to provide high reliability and fault tolerance by replicating data across multiple servers. It uses a master-slave architecture, where a single master node manages the namespace and coordinates operations across multiple slave nodes that store the actual data.

GFS is optimized for handling large files, typically on the order of hundreds of megabytes or gigabytes, rather than small files. It uses a distributed architecture to provide high throughput and low latency for accessing data. GFS has been used extensively within Google for a variety of applications, including storing web crawl data, user-generated content, and log data.



Architecture

Chunk Server

A chunk server is a key component in a distributed file system like GFS. Its primary responsibility is to store and serve data chunks in the system. Each chunk server can store multiple chunks, and each chunk is replicated across multiple chunk servers for redundancy and fault tolerance. When a client needs to read or write a chunk, it sends a request to the appropriate chunk server, which serves the data or forwards the request to another server if necessary. The chunk server also performs checksum validation to ensure data integrity and can perform caching to improve read performance. In addition, the chunk server maintains a heartbeat with the master server to provide up-to-date information about its status and the chunks it is responsible for. In the event of a chunk server failure, the master server can re-replicate the lost chunks to other servers to maintain the desired level of redundancy.

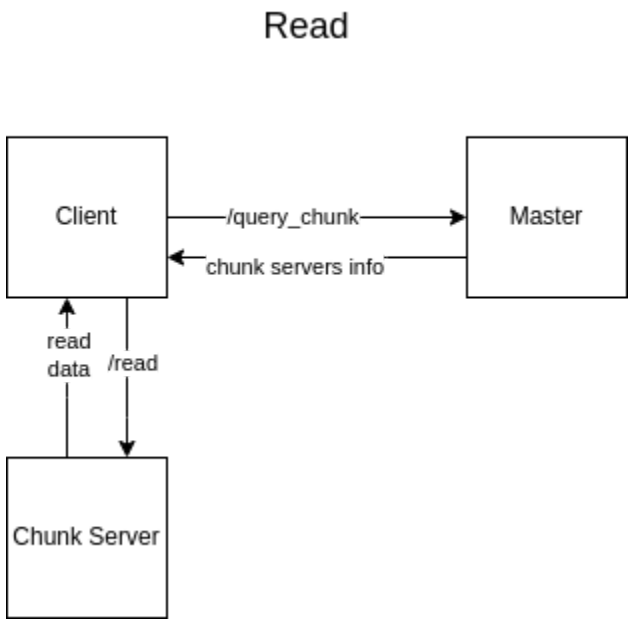
Functions

Heartbeat

- Chunk servers periodically send heartbeat messages to the master server to report their status and the chunks they are responsible for.
- The heartbeat messages help the master server detect failures or network partitions in the system.
- If a chunk server fails to send a heartbeat message, the master server considers it to be offline and initiates replication of its chunks to other servers to maintain the desired level of redundancy.

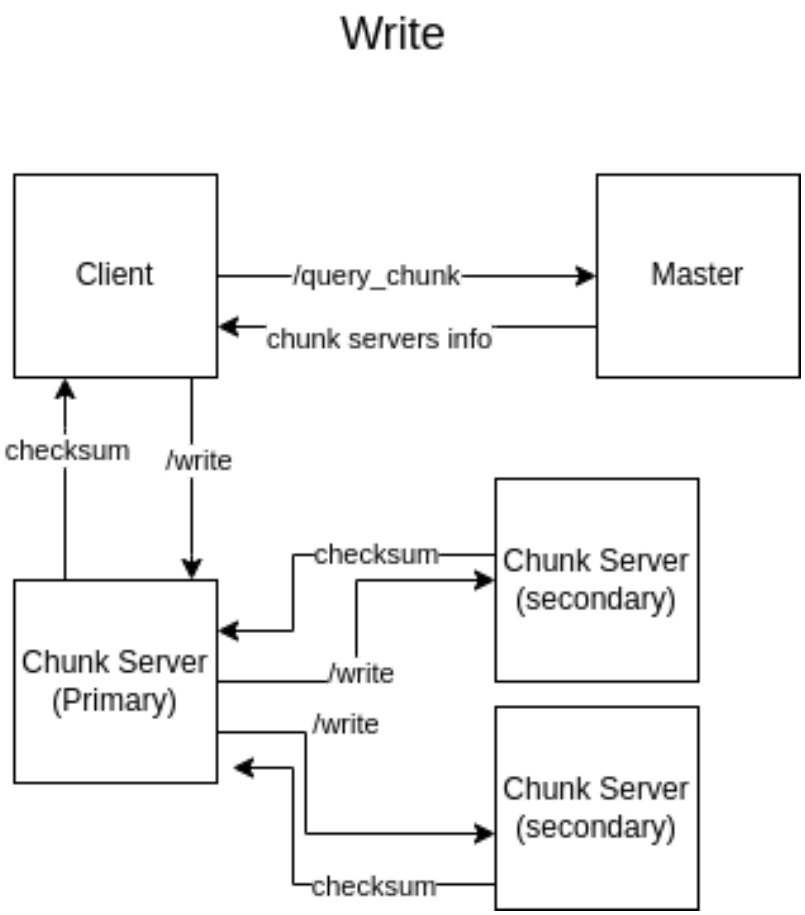
Read

- When a chunk server receives a read request from a client, it checks its cache for the requested chunk. If the chunk is not in cache, the server reads it from disk and sends it to the client.
- If the chunk is in cache, the server returns it immediately. If the cache is full, the server evicts the least recently used (LRU) chunk to make room for the requested chunk.



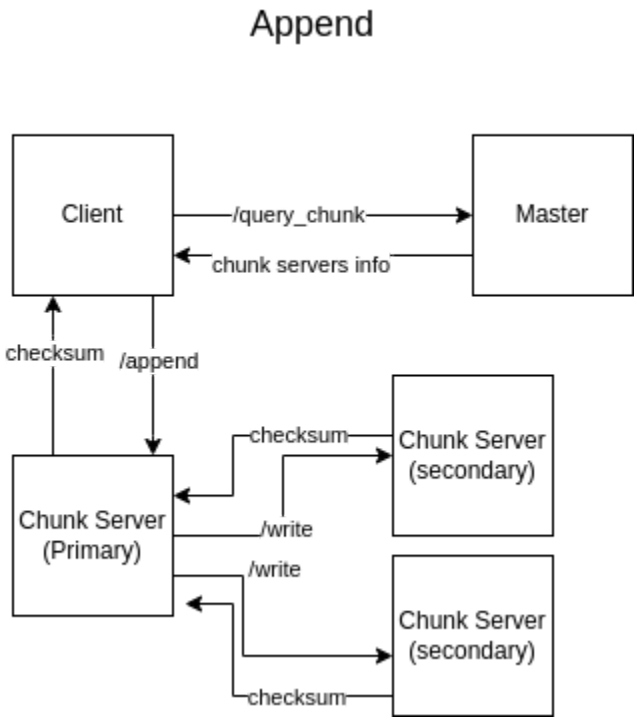
Write

- When a chunk server receives a write request from a client, it first generates a unique checksum for the data. The checksum is used to verify the integrity of the data during subsequent read operations.
- The server then writes the data to disk and updates its in-memory metadata.
- If the chunk is replicated on other chunk servers, the primary chunk server sends the updated chunk to the replicas to ensure consistency across all replicas.
- Finally, the primary chunk server sends an acknowledgement to the client in the form of data checksum to indicate that the write operation has successfully completed.



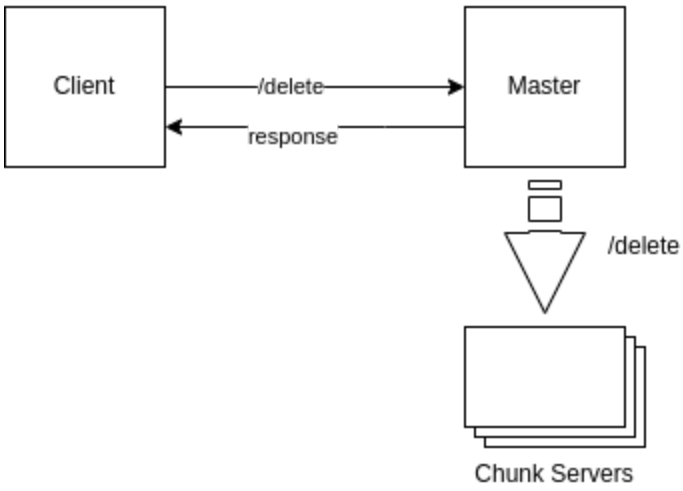
Append

- The append function is used to add data to the end of an existing chunk, which can be useful in scenarios where the chunk is being written to by multiple clients concurrently.
- The append function may also trigger the creation of new replicas of the chunk if the existing replicas do not meet the desired level of redundancy, similar to the replication process for new chunks.



Delete

When a chunk is no longer needed, it must be deleted to free up space on the chunk server's disk.



Maintaining Replication

- Each chunk in the system is replicated across multiple chunk servers to ensure redundancy and fault tolerance.
- When a new chunk is created, the master server selects one of the chunk servers to act as the primary chunk server for that chunk.
- The primary chunk server is responsible for coordinating the replication of the chunk to the other replica chunk servers.
- The primary chunk server sends the new chunk data to the replica servers and updates their metadata to reflect the new replica.

Client

GFS Client provides an interface to handle all the read, write, append requests. Client requests the metadata of a chunk from the master, and then sends the chunk-server a request for read, write or append. Multiple clients can append their data to the same file.

Metadata:

Client requests for metadata of a chunk from the master. Metadata contains chunk handle, chunk-servers' information, primary chunk-server, expiration time. Client caches this information and only makes a new request after expiration time.

Read:

Client requests the closest chunk-server for the chunk-handle of the chunk it wants.

We can read a particular chunk, whole or partly. We can also read the whole file sequentially.

Write:

Client sends a write request to the primary chunk-server of that chunk. Primary chunk-server then makes sure that the request is sent to all the secondary chunk-servers. Write requests updates a given byte range of a particular chunk.

Append:

Client finds the metadata of the last chunk. Then sends an append request to its primary chunk-server. In case of overflow, the client requests the master to create a new chunk, then retries on the new chunk. We can append data of size at max (1/4)th of the total chunk capacity. When appending large files, client breaks the file into small chunks and then appends them sequentially.

Master

The master executes all namespace operations. In addition, it manages chunk replicas throughout the system: it makes placement decisions, creates new chunks and hence replicas, and coordinates various system-wide activities to keep chunks fully replicated, to balance load across all the chunk servers, and to reclaim unused storage. Overall it is responsible for managing and distributing all the operations across all the entities.

Functions

Heartbeat

- Master periodically receives heartbeat from the Chunk Servers to keep a check if the Chunk Server is Alive or not.
- In case of Chunk Server failure the Master initiates the replication of all the chunks of that Chunk Server to maintain the desired level of redundancy.

Metadata

- Master Stores and Maintains all the metadata corresponding to the file and chunks.
- On the Client's requests the Master sends the metadata corresponding to the chunks of that file including chunk handle, chunk-servers' information, primary chunk-server, expiration time.
- Master generates the unique Chunk Handle for every chunk it creates for the file.

Load Balancing

- Master periodically receives Information from the chunk Servers regarding the load server is bearing and keeps the track of it for all the servers.
- Master redistributes data chunks to balance the load across the chunk servers and ensure optimal performance.

Delete

- When a chunk is no longer needed, the Master receives the message from the client and will inform the corresponding chunk server to free up the space.
- The Deleted chunk can also be recovered if the client has mistakenly requested to delete the chunk and has requested to recover it within 90 sec after the delete request.
- Once the Deletion Operation has been executed successfully the garbage collector gets invoked to free up the space of the deleted data.

Other

- Master assigns the primary server to which the client will be directly interacting while performing the read and write operation.
- Master keeps the track of the lease time for the primary server and extends it on the request of the chunk server otherwise on the expiry of the lease time Master revokes the primary status of the Chunk Server.