GOOGLE FILE SYSTEM

Overview

- Introduction of GFS
- Data Storage & Replication
- Operations & GFS Architecture
- Security, GFS Vs Other file systems
- Realworld Usecases & Future Development

INTRODUCTION TO GOOGLE FILE SYSTEM

-BY Thakshin Kumar.T
22Z266

GOOGLE FILE SYSTEM

The Google File System (GFS) is a scalable distributed file system designed by Google to handle large-scale data processing workloads across thousands of commodity servers. Traditional file systems were not well-suited for the rapidly growing storage needs and massive datasets generated by Google’s search and indexing operations.

FEATURES OF GOOGLE FILE SYSTEM

1. **Scalability** – GFS is designed to store petabytes of data and scale across thousands of machines.
2. **Fault Tolerance** – It handles failures efficiently by replicating data across multiple nodes.
3. **High Throughput** – Optimized for large sequential reads/writes rather than low-latency small file operations.
4. **Chunk-Based Storage** – Files are divided into fixed-size chunks (64MB), each replicated across multiple servers.
5. **Master-Slave Architecture** – A single GFS Master manages metadata, while Chunkservers store actual file data.
6. **Automatic Load Balancing** – The system redistributes data and workload to avoid bottlenecks.

GFS Architecture

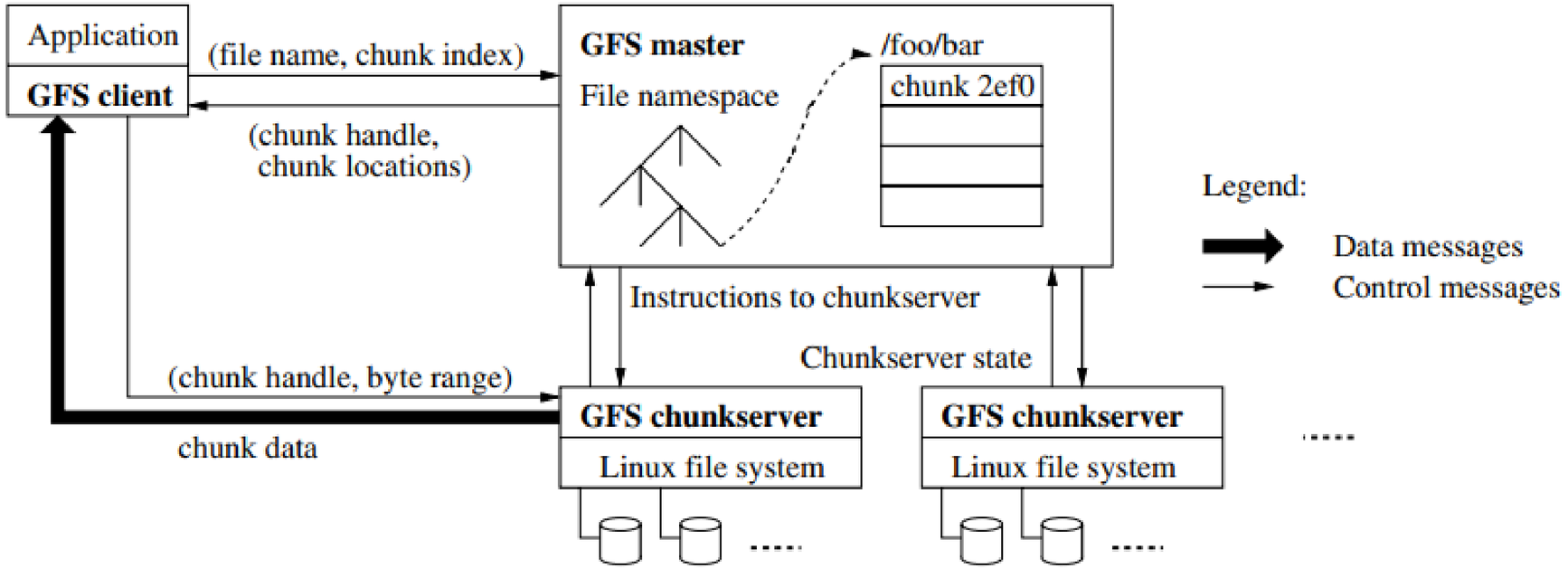


Figure 1: GFS Architecture

EXPLANATION OF ARCHITECTURE

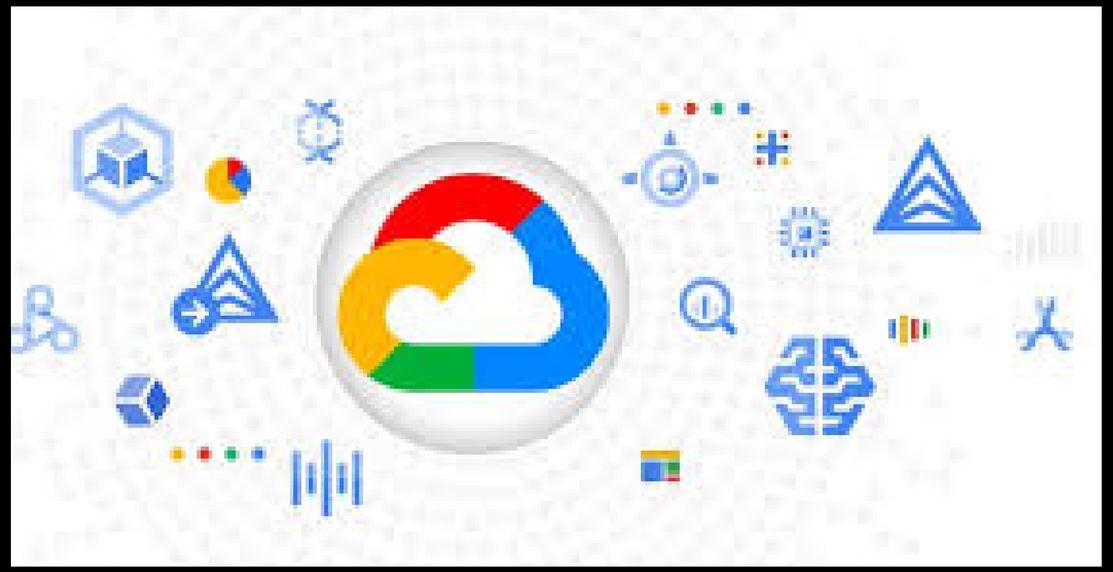
- **Components:** GFS consists of a Master (manages metadata), Chunkservers (store data chunks), and Clients (request and process data).
- **File Storage:** Files are split into fixed-size chunks (e.g., 64MB) and stored across multiple chunkservers with unique chunk handles.
- **Data Flow:** Clients get chunk locations from the Master and directly fetch data from Chunkservers, reducing load on the Master.
- **Fault Tolerance:** Chunks are replicated (typically 3 copies) to ensure data availability in case of server failures.
- **Efficiency:** Optimized for large sequential reads/writes, automatic load balancing, and garbage collection for better performance.

ADVANTAGES OF GOOGLE FILE SYSTEM

- **Scalability:**
  GFS can efficiently handle very large files and scale across thousands of machines, making it suitable for massive data processing.
- **Fault Tolerance:**
  Data is automatically replicated across multiple chunkservers, so even if some servers fail, data can still be retrieved from other replicas.

DISADVANTAGES OF GOOGLE FILE SYSTEM

- **Not Suitable for Small Files:**
  GFS is designed for large files (gigabytes to terabytes). Handling a large number of small files is inefficient due to the overhead of metadata management and chunking.
- **High Latency for Random Access:**
  GFS performs well with sequential access, but random reads/writes are slower due to its distributed nature and chunk lookup process.



Data Storage,Replication & Fault Tolerance

-Prateekshaa T

22z246

DATA STORAGE

1

File Chunks

2

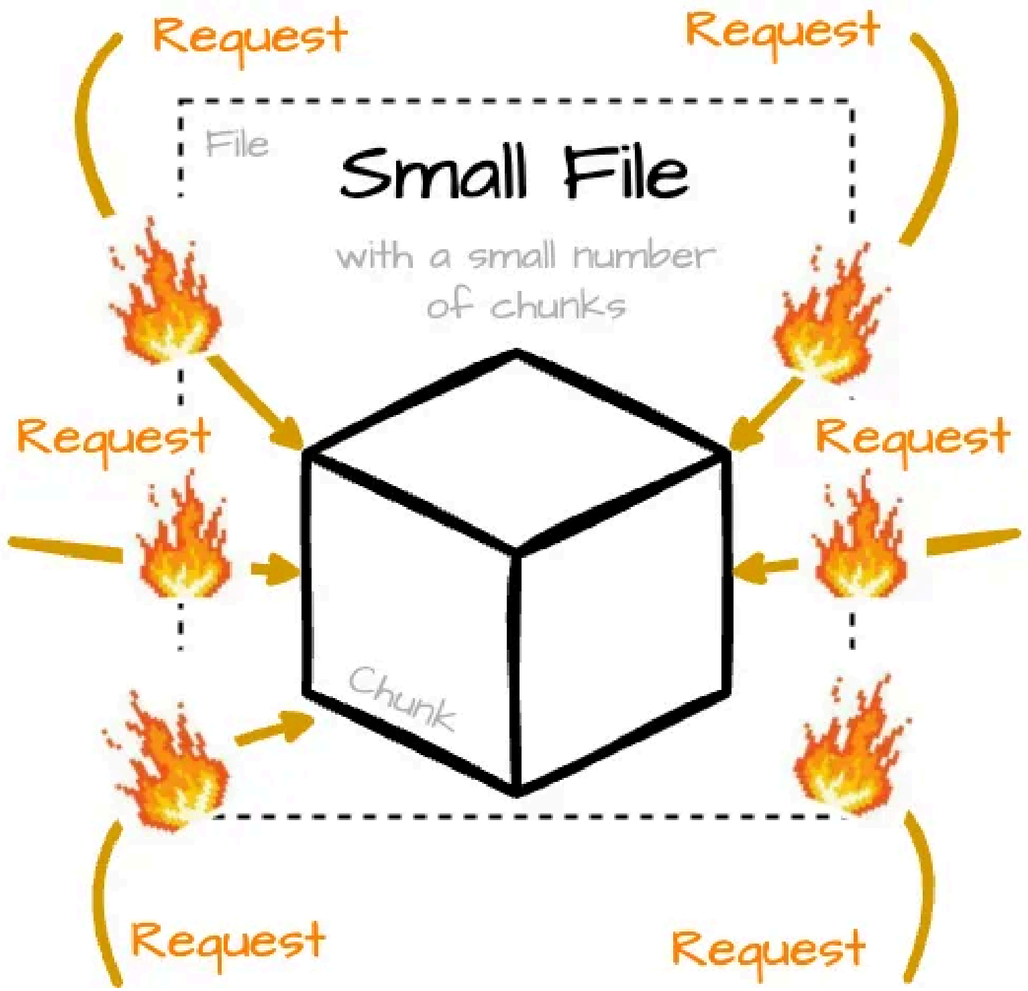
Chunk Placement Strategy

3

Metadata & Namespace Management

“Chunk Size”

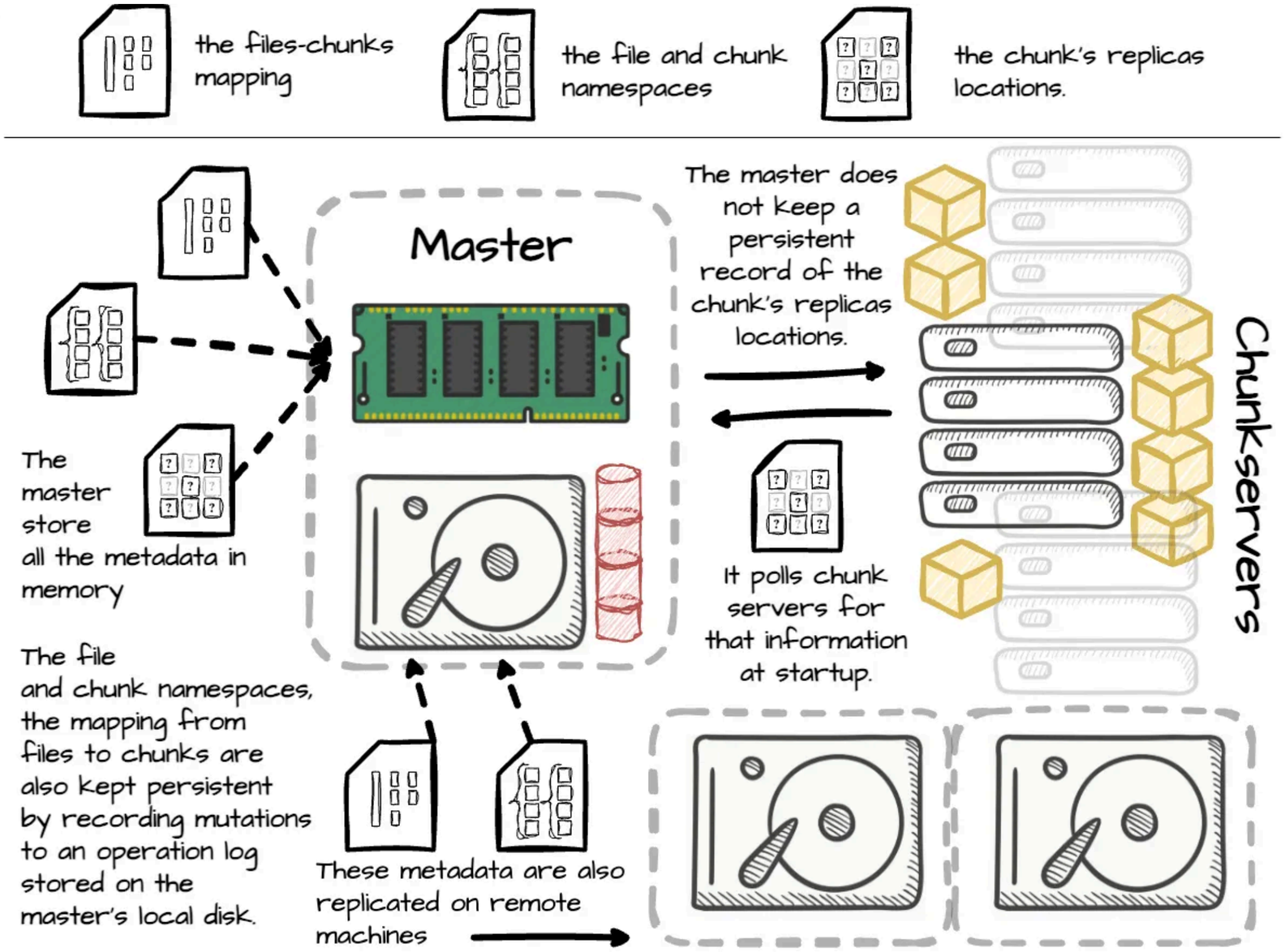
- Google decided on a chunk size of 64 MB, which was more significant than most file system block sizes at the time.
- GFS stores each chunk replica as a plain Linux file on a chunkserver



“Metadata”

Types of Metadata

- File and chunk namespaces
- Files-chunks mapping
- Chunk’s replica location



Metadata (contd.)

Namespaces

- This metadata defines the hierarchical structure of the file system, similar to directories and files in a traditional file system.
- It includes the names of files and directories, their ownership, and access permissions.

Mappings

- This metadata maps files to the chunks that make up those files.
- It essentially tells the Master which chunks belong to which files and in what order.

Replica locations

- This metadata indicates the locations of the replicas for each chunk.
- It tells the Master which chunk servers store copies of a particular chunk.

Replication Strategy in GFS

Replication Factor

Chunk Replica Placement Strategy

Load Balancing for Replicas

FAULT TOLERANCE

☑ Handling ChunkServer Failures

The Master continuously monitors ChunkServers through heartbeat messages.

If a ChunkServer fails:

- The Master detects the failure (due to missing heartbeats).
- It re-replicates the lost chunks on other healthy ChunkServers.
- Clients automatically switch to other replicas when a server fails.

☑ Handling Master Node Failures

The Master Node is a single point of failure, but GFS has mechanisms to recover:

- Periodic checkpoints & operation logs are used to restore the Master state.
- A shadow Master (backup) can take over if the main Master crashes.

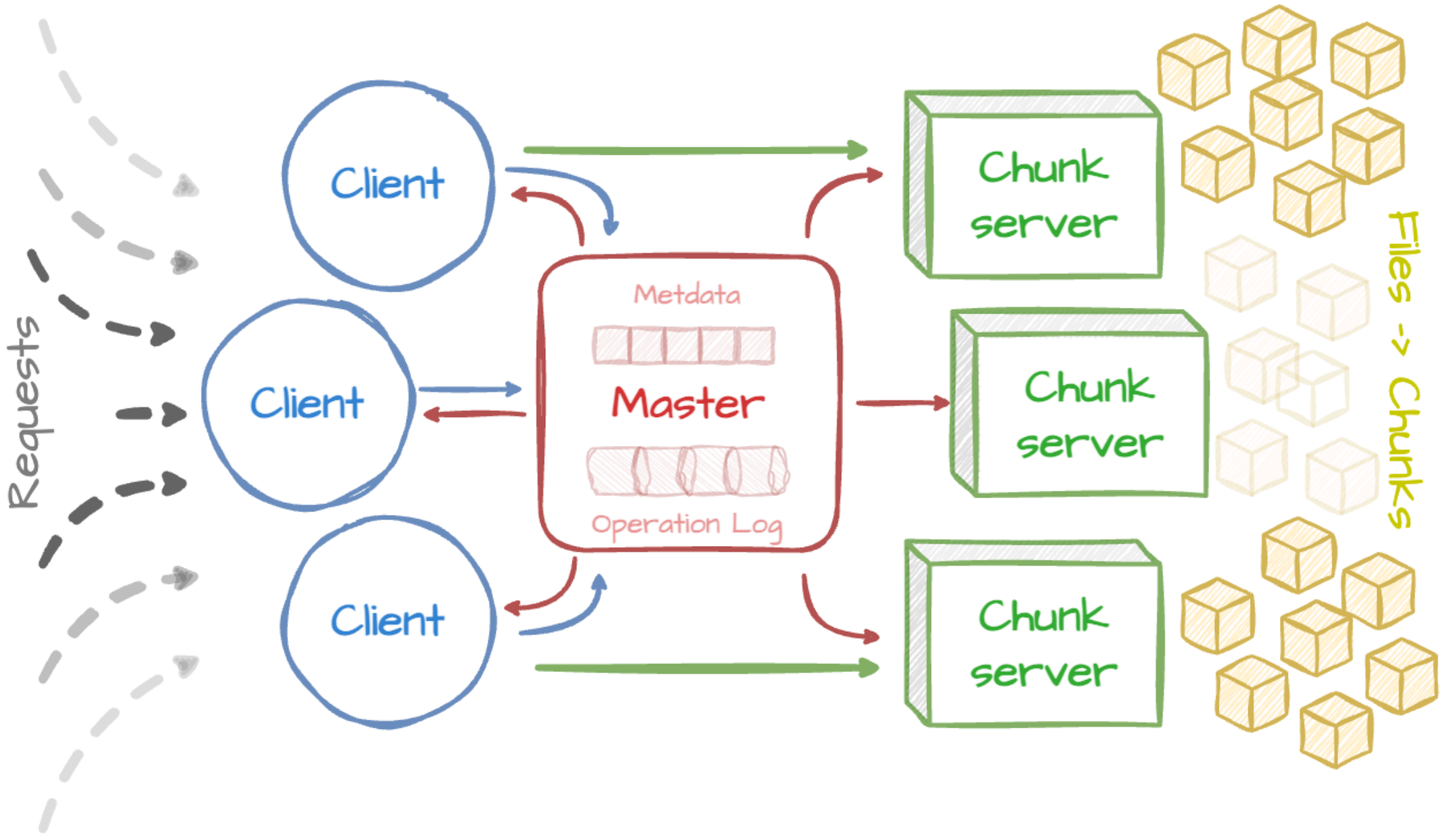
GFS Operations

Hemanthkumar V (22z225)

GFS Architecture

Components

- GFS Clients
- GFS Master
- GFS Chunkservers



Operations in GFS

create

read

update

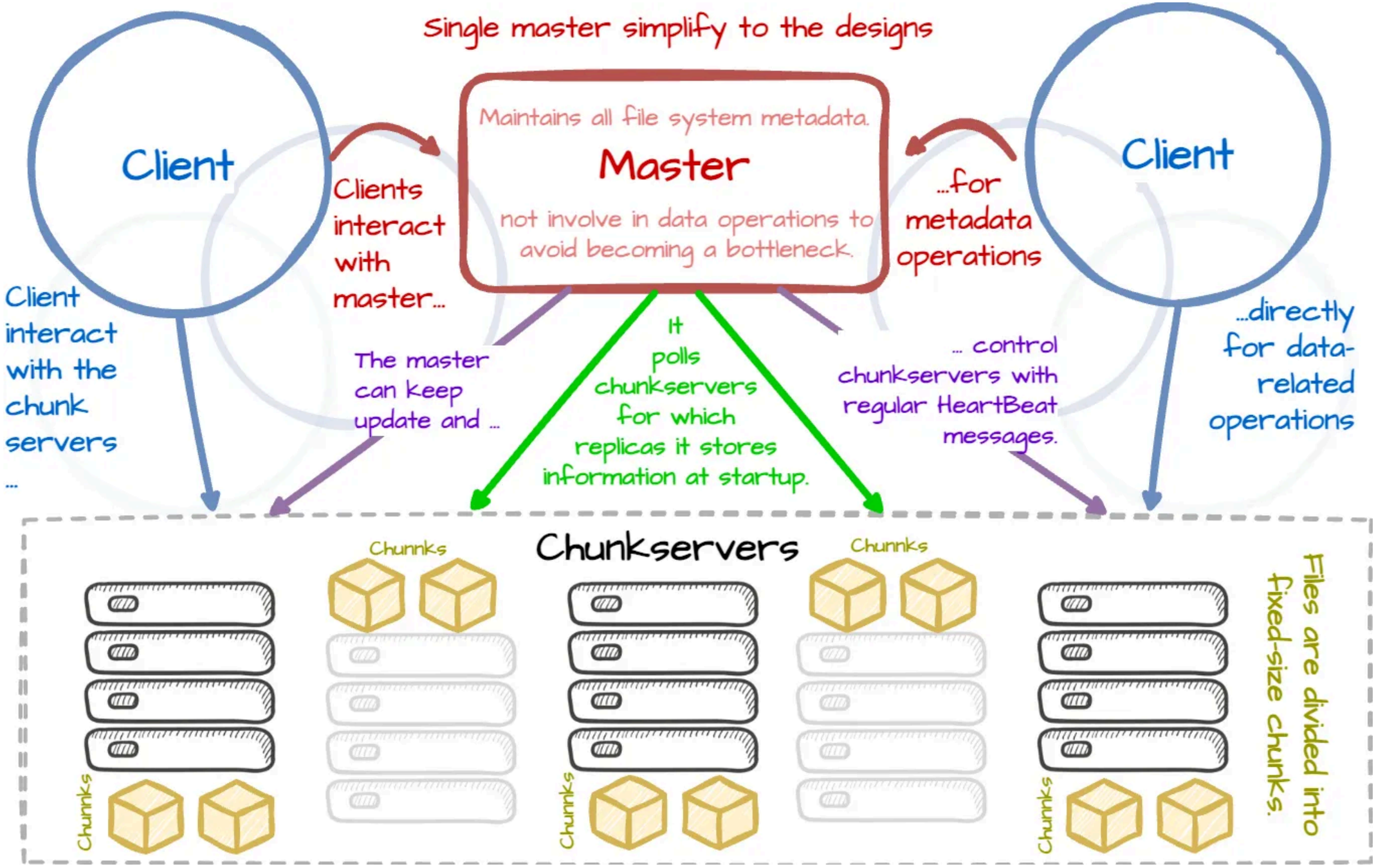
delete

I) Snapshots

- Snapshot creates a copy of a file or a directory tree cheaply

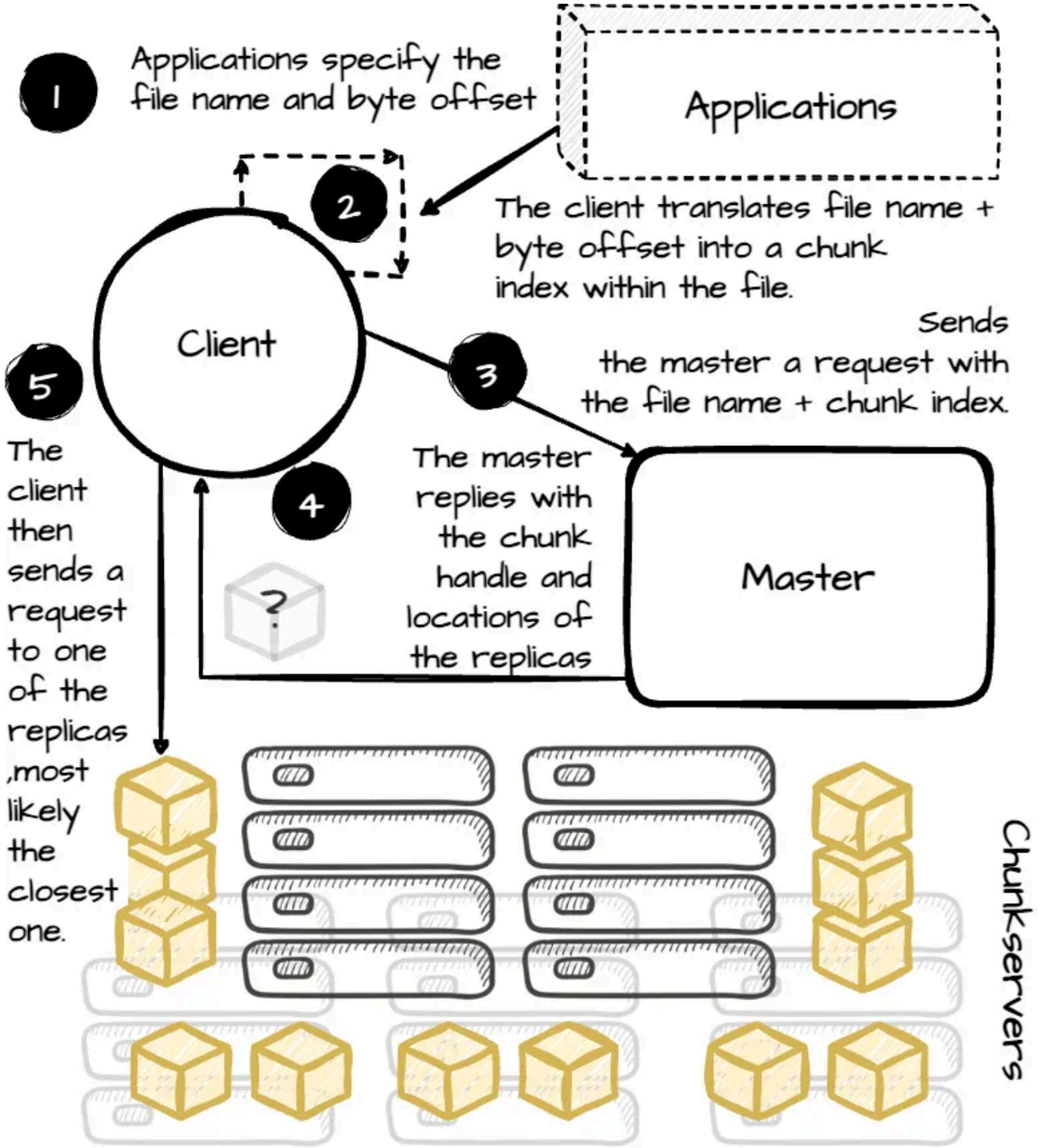
II) Record Append

- Record append allows multiple clients to append data to the same file concurrently while guaranteeing the atomicity of each client's append.


Single master simplify to the designs
Client
Clients interact with master...
Maintains all file system metadata.
Master
not involve in data operations to avoid becoming a bottleneck.
...for metadata operations
Client
Client interact with the chunk servers
...
The master can keep update and ...
It polls chunkservers for which replicas it stores information at startup.
... control chunkservers with regular HeartBeat messages.
...directly for data-related operations
Chunkservers
Chunnks
Chunnks
Chunnks
Chunnks
Chunnks
Files are divided into fixed-size chunks.

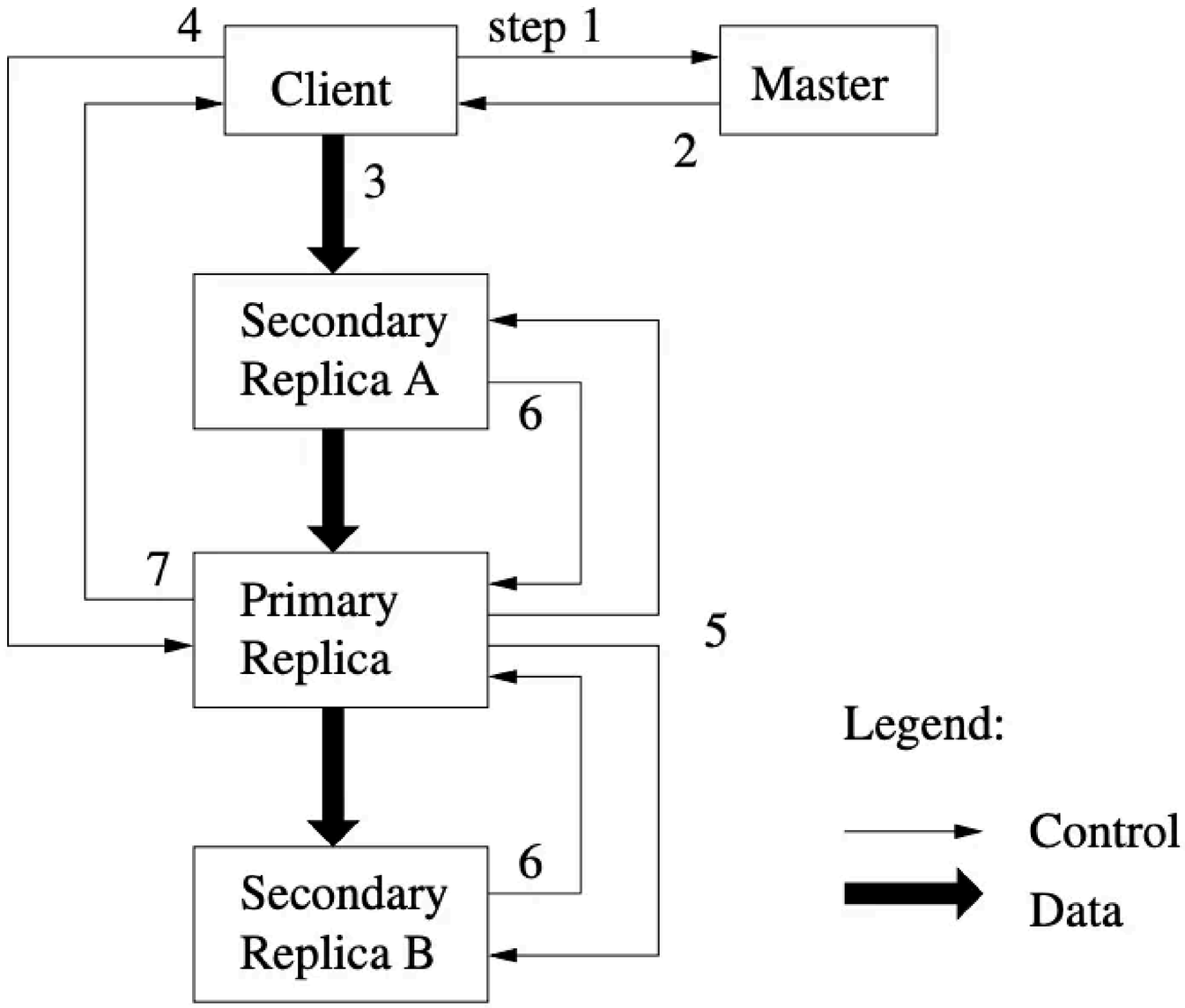
Read

- **Application Request**
- **Client Translation**
- **Master Request**
- **Master Response**
- **Client - Chunkserver Communication**



Write

- **Mutations and Consistency**
- **Lease Mechanism**
- **Write Operation Steps**
- **Data Flow vs. Control Flow**
- **Large Writes**



Mutations

A mutation is any operation that alters the data or metadata of a chunk. This includes

- **Writes:** Changing the data within a specific range of a chunk.
- **Appends:** Adding data to the end of a chunk.
- **Metadata changes:** Modifications to chunk versions, or other internal data.

Replication: GFS applies each mutation to all replicas of a chunk to maintain data redundancy and fault tolerance.

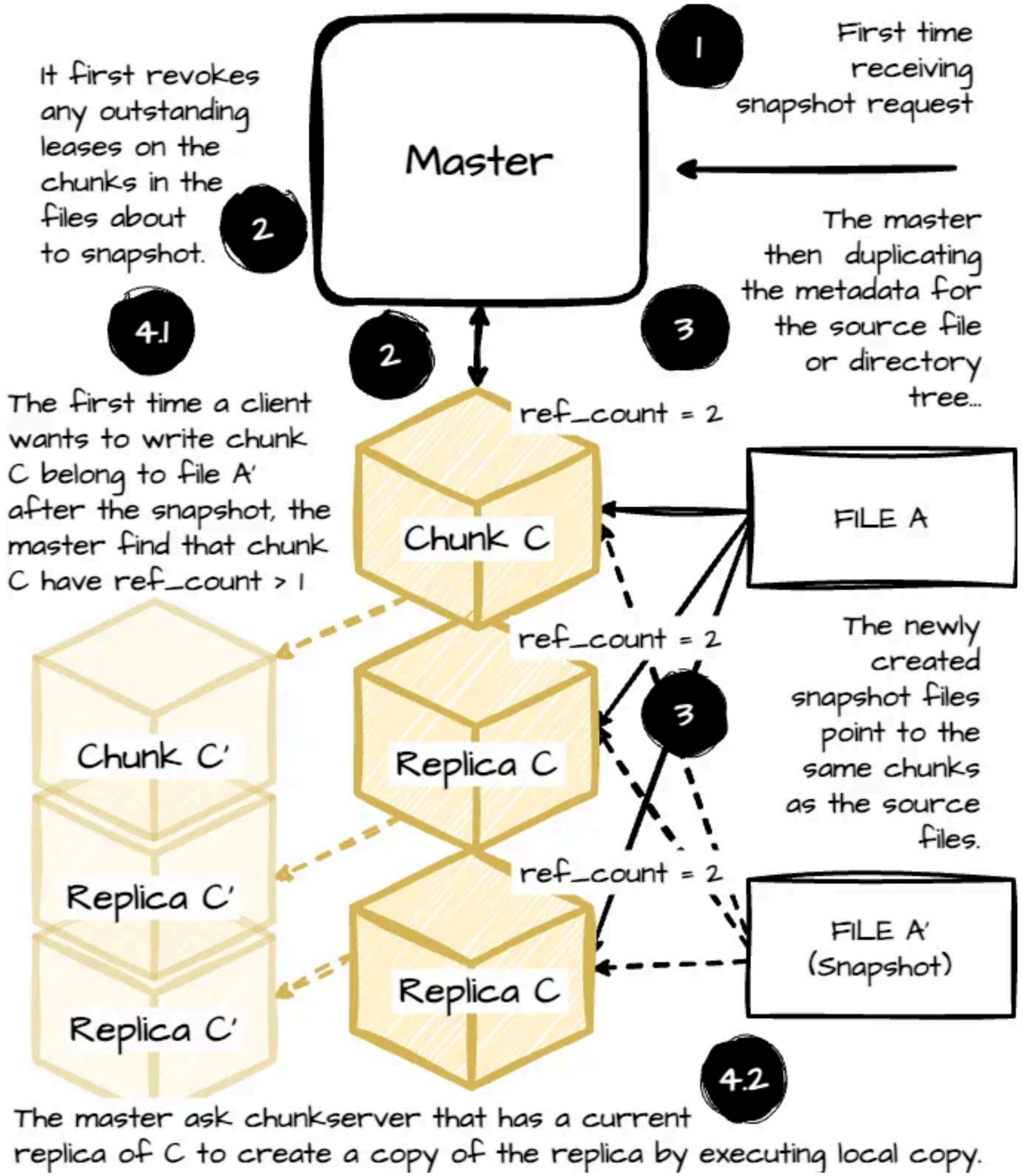
Consistency Model

	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

File Region State After Mutation

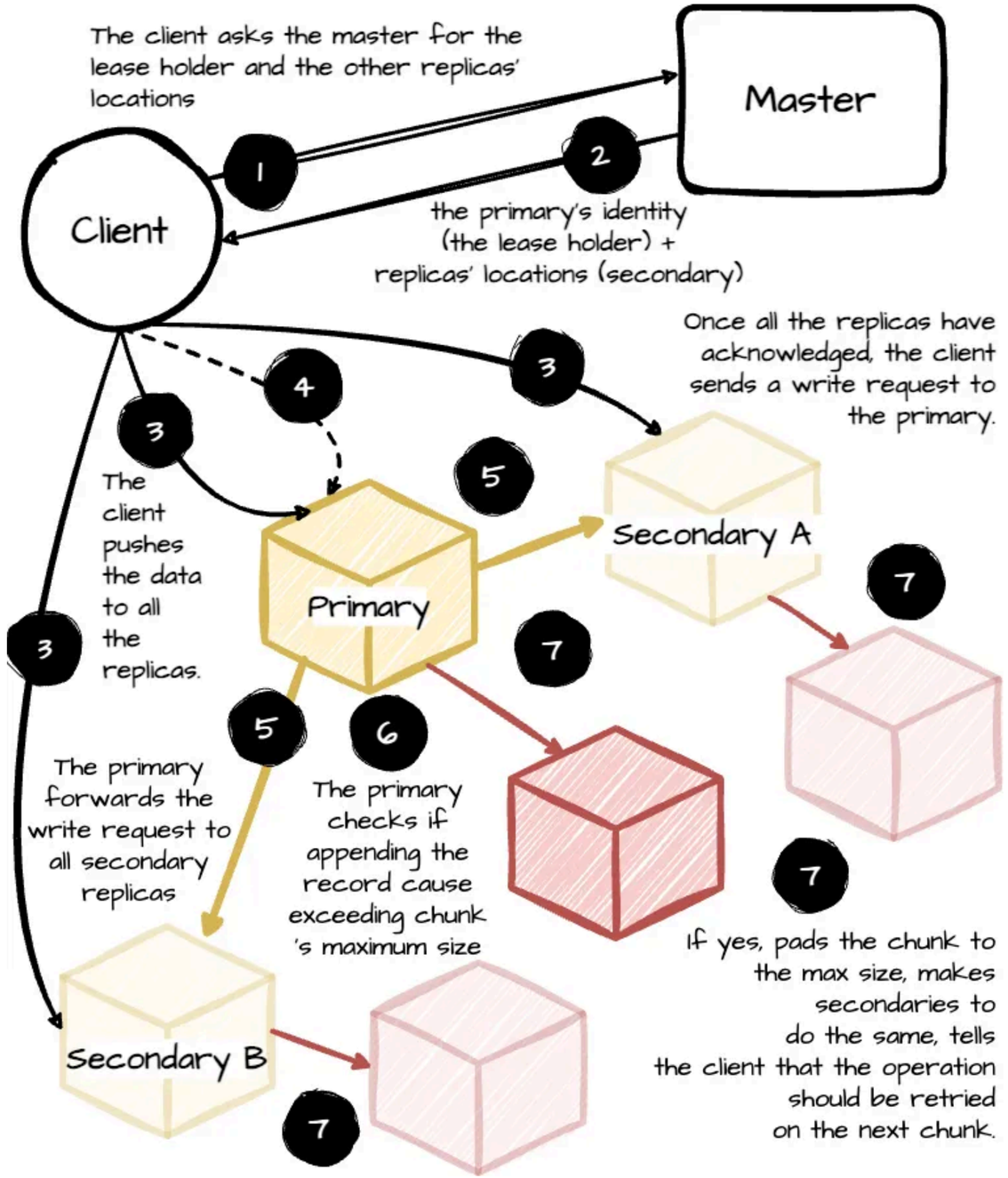
Snapshot

- Snapshot Request
- Lease Revocation
- Metadata Duplication
- Copy-on-Write
  - First write after Snapshot
  - Chunk Copying
  - Subsequent Operation



Record Append

- Request Lease and Replica Locations
- Master Response
- Data Push
- Write Request to Primary
- Primary Forwarding
- Chunk size check
- Chunk Padding and Retry



Master Operations in GFS

- Namespace Management and Locking
- Replica Placement
- Creation, Re-replication, Rebalancing
- Garbage Collection
- Stale Replica Detection

MASTER OPERATIONS IN GOOGLE FILE SYSTEM

-Jeevashakthi V (22z228)

The Master plays a central role in managing files, storage, and system performance. It keeps track of where data is stored, ensures data remains available, and optimizes how the system handles file operations.

To keep things running smoothly, the Master performs several key operations, including:

- **Namespace Management & Locking** to organize file access
- **Replica Placement** to keep data safe and accessible
- **Re-replication & Rebalancing** to maintain system efficiency
- **Garbage Collection** to manage deleted files
- **Stale Replica Detection** to ensure users always access up-to-date data
- **High Availability** strategies to minimize downtime
- **Data Integrity** Checks to prevent corrupted data
- **Diagnostic Tools** for monitoring system health

Namespace Management & Locking

What is a Namespace in GFS?

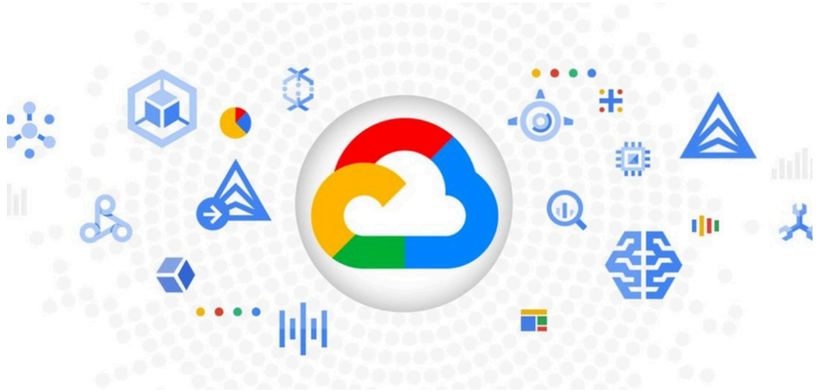
- Unlike traditional file systems, GFS does not store files in directory structures.
- Instead, it maintains a lookup table that maps file paths to metadata.
- This allows efficient file management without requiring separate directory listings.

Locking Mechanism for Parallel Operations

- Since some operations take longer to execute, GFS allows multiple operations to run simultaneously.
- To avoid conflicts, it uses read-write locksat different levels:
- Read locks prevent deletion or modification of a directory while accessing it.
- Write locks prevent conflicting operations on the same file or directory.

Example: If a directory /home/user is being snapshotted, no new files can be created in it until the snapshot is complete.

Replica Placement



Why is Replica Placement Important?

- Ensures reliability in case of hardware failure.
- Optimizes network performance by distributing workload.

How Are Replicas Placed?

GFS places replicas across different racksto:

- Prevent data loss if an entire rack fails.
- Distribute read traffic efficiently.
- Reduce network congestion for large-scale data operations.
- Trade-off: Write operations require data to be sent across multiple racks, slightly reducing speed.

Chunk Creation, Re-replication, and Rebalancing

Chunk Creation

New chunks are created based on:

- Available disk space on chunkservers.
- Avoiding overloading specific servers.
- Distributing replicas across racks for fault tolerance.

Re-replication

Happens when a chunk loses replicas due to:

- Chunkserver failures.
- Corrupted data.
- Disk space issues.

Prioritization of Re-replication:

- Most critical chunks (actively in use) are recovered first.
- Chunks with fewer remaining replicas get higher priority.

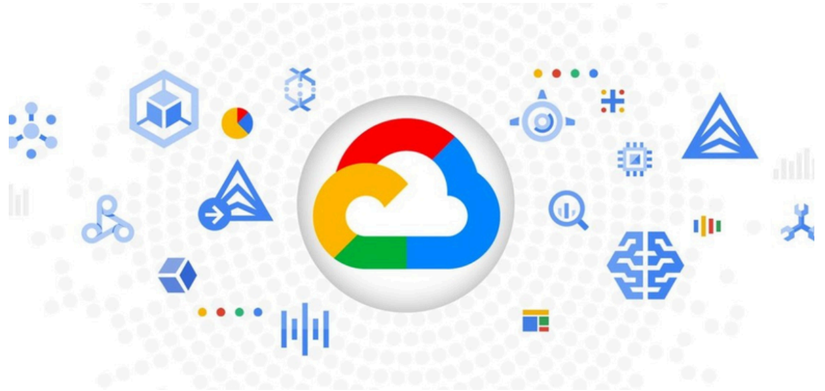
Rebalancing

Why rebalancing is needed:

- Some servers might get overloaded while others remain underutilized.

The Master periodically moves replicas to balance storage and processing load.

Garbage Collection



Why is Garbage Collection Needed?

GFS does not immediately delete files.

Instead, deleted files are marked as hidden for a set period (default: 3 days).

How Does It Work?

When a file is deleted, its metadata is updated instead of removing it immediately.

After 3 days, the Master removes the file's metadata permanently.

The Master then notifies chunkservers to delete orphaned chunks.

Benefits of Delayed Deletion:

Allows recovery if the file was deleted by mistake.

Prevents unnecessary load on the system due to immediate deletions.

Stale Replica Detection

What is a Stale Replica?

A stale replica is a chunk that missed updates while its chunkserver was down.

If an outdated chunk is used, data inconsistency can occur.

How Does GFS Detect Stale Replicas?

1. GFS assigns a version number to each chunk.
2. When a chunkserver restarts, it reports its chunk versions to the Master.
3. The Master compares versions and discards outdated replicas.
4. Ensuring Clients Get Updated Data
5. The Master sends version numbers to clients.
6. Clients check the version before accessing data to ensure they are using the latest

Real-World Use Cases, Evolution of GFS & Future Developments

L Monish Rajan
22Z240

Evolution of GFS

GFS → Colossus (Google’s New File System)

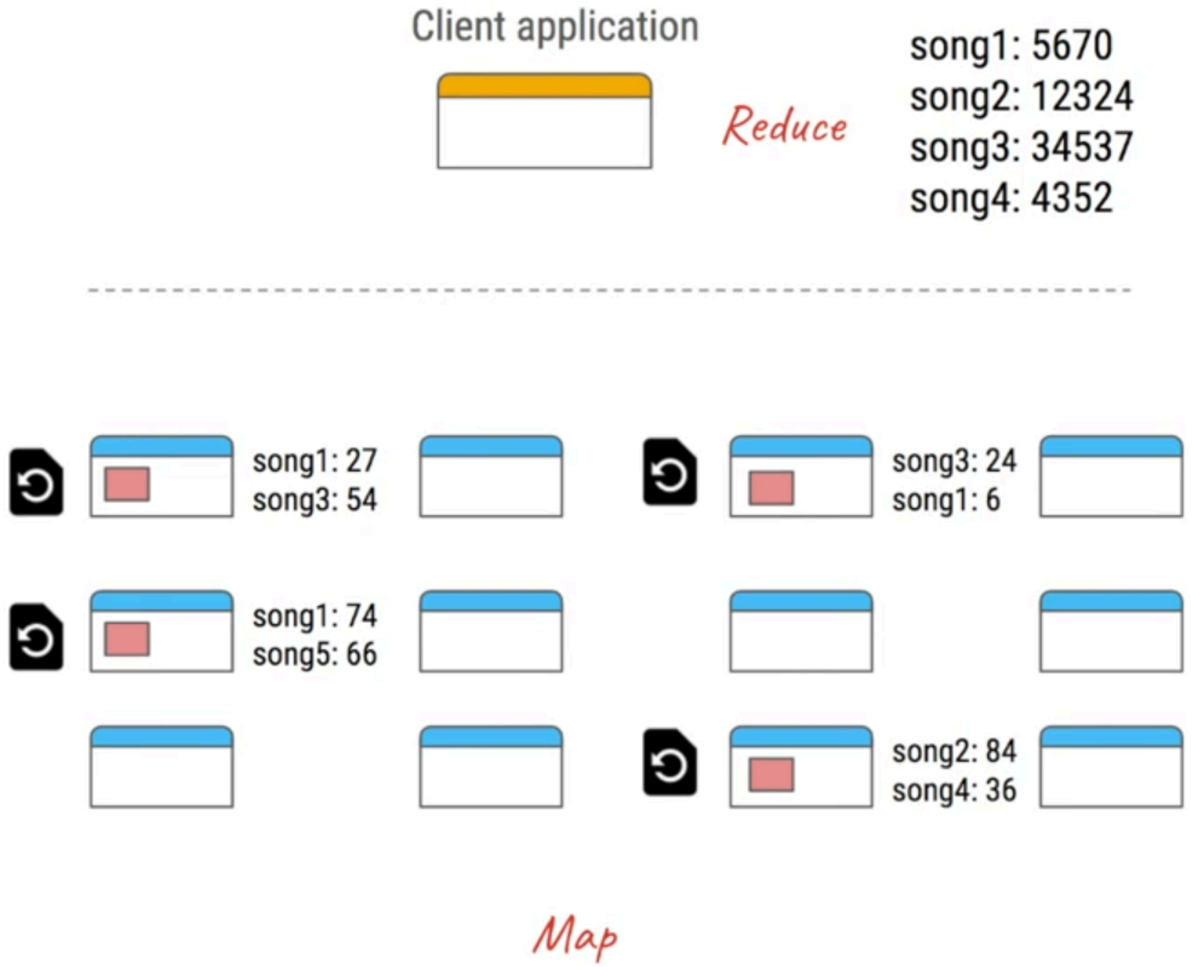
- GFS v1 (2003-2010): Original system, limited scalability.
- GFS v2 (2010-2012): Improved metadata management, reliability.
- **Colossus (2012-Present):**
  - Replaced GFS.
  - Uses distributed master servers (no single point of failure).
  - Higher performance for real-time applications.
  - Powers Google Photos, Gmail, Cloud Storage.

How is a large amount of data processed efficiently?

Map + Reduce

Map = Processing part of data

Reduce = Aggregation



Real-World Use Cases of GFS

Google Services using GFS:

- Google Search Indexing → GFS stores web crawled data.
- YouTube & Google Drive → Manages huge media files.
- Google Earth & Maps → Handles geographical datasets.
- Big Data Processing (MapReduce & Hadoop) → Inspired by GFS.

Other Companies Inspired by GFS:

- Hadoop Distributed File System (HDFS) (Open-source version of GFS).
- Facebook, Amazon, Netflix use similar distributed file storage techniques.

Case Study Example:

- Google's Caffeine Indexing System (2010) → Faster web search by processing smaller incremental updates instead of batch processing.

Future Developments & Trends in Distributed File Systems

Key Trends & Innovations:

AI-Powered Storage Optimization

- Intelligent caching, automated replication strategies.

Hybrid Cloud & Multi-Cloud Storage

- Integrating distributed file systems with AWS, Azure, Google Cloud.

Edge Computing & Storage

- Data storage closer to users for faster access.

Quantum Computing Impact

- Potentially revolutionizing storage and retrieval speeds.

Blockchain-Based Storage Security

- Ensuring integrity and distributed trust in file storage.

Thank You