GFS with Dynamic Replication

Team 37:
Archit Pethani (2022101098)
Neel Amrutia (2022101117)

Objective:
Distributed systems require a storage solution that ensures **resilience**, **scalability**, and **high throughput** for data-intensive applications.

Key Challenge:
How to efficiently manage data replication in distributed systems to optimize **performance**, **fault tolerance**, and **resource utilization**?

Solution:
Develop a Google File System (GFS)-inspired distributed file system with a **dynamic replication feature** that adjusts the number of replicas based on real-time metrics like access patterns and server load.

Project Scope

Our goal was to build a GFS-inspired file system with dynamic replication capabilities.

This system will adapt the number of data chunk replicas based on usage patterns.

We aim to increase replicas for high-demand chunks, ensuring data availability.

Seldom-accessed data will see reduced replication, optimizing resource allocation.

This dual approach enhances overall data management efficiency.

Core Architecture

- Master Server: Responsible for managing metadata, chunk distribution, and overseeing initial replication.
- Chunk Servers: Store file data chunks, manage read/write operations, and execute replication adjustments as instructed by the master.
- Client Interface: Provides basic file operations (e.g., create, read, write) and collects access metrics to facilitate replication management.

Dynamic Replication Design

DEFINING REPLICATION POLICIES

MONITORING LOAD FOR CHUNKS AND CHUNK SERVERS

DYNAMIC REPLICATION CONTROL

Dynamic Replication Implementation

- Access counts for each chunk are tracked over fixed intervals (here, 15 seconds)
- **Increase Replicas**: When access exceeds an upper threshold, new replicas are created on underloaded chunkservers.
- **Decrease Replicas**: When access falls below a lower threshold, excess replicas are deleted to save resources.
- The master server assigns new replicas to chunkservers with the lowest load, ensuring even distribution.
- Similarly when the number of replicas have to be decreased then chunk is removed from the chunkservers with higher load.

Current Limitations

- **Delayed Adjustments**: Replica adjustments are based on periodic checks, which may introduce lag in response to rapid changes in load.
- **Minimum Replication Constraint**: A hard limit on the minimum number of replicas (e.g., 2) ensures fault tolerance but may hinder optimal resource usage.
- **Dependency on Load Metrics**: Inaccurate load metrics could lead to suboptimal replication decisions

Key Outcomes

- Efficient use of system resources.

- Improved data availability and fault tolerance.

- Enhanced scalability and adaptability to workload variations.

Conclusion

Dynamic replication addresses the challenges of managing distributed storage in data-intensive systems.

The project provides a scalable, resilient, and high-performance solution inspired by the success of GFS.