Reading Assignment
Google File System

- Omanshu Mahawar (181CO237)

Introduction

GFS is a scalable distributed file system for large data intensive applications
It shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability.
The design of GFS is driven by four key observations-
Component failures, huge files, mutation of files, and benefits of co-designing the applications and file system API

Assumptions

The following assumptions are made while designing file systems that offer both challenges and opportunities.

1. GFS has high component failure rates.System is built from many inexpensive commodity components

2. The system stores a modest number of huge files. A few million files, each typically 100MB or larger (Multi-GB files are common). No need to optimize for small files

3. The workloads primarily consist of two kinds of reads: large streaming reads and small random reads. In large streaming reads, individual operations typically read hundreds of KBs, more commonly 1 MB or more. Successive operations from the same client often read through a contiguous region of a file. A small random read typically reads a few KBs at some arbitrary offset. Performance-conscious applications often batch and sort their small reads to advance steadily through the file rather than go backand forth.

4. High sustained throughput is more important than latency. Response time for individual read and write is not critical

GFS Design Overview

GFS cluster comprises a single master and multiple chunk servers accessed by multiple clients. Since files to be stored in GFS are huge, processing and transferring such huge files can consume a lot of bandwidth. To efficiently utilize bandwidth files are divided into large 64 MB size chunks which are identified by a unique 64-bit chunk handle assigned by master. The design supports the usual posix operations open, close, read, write. In addition provides two more operations:

1. Record append: atomic append operation.
2. Snapshot: copy of a file or directory instantaneously.

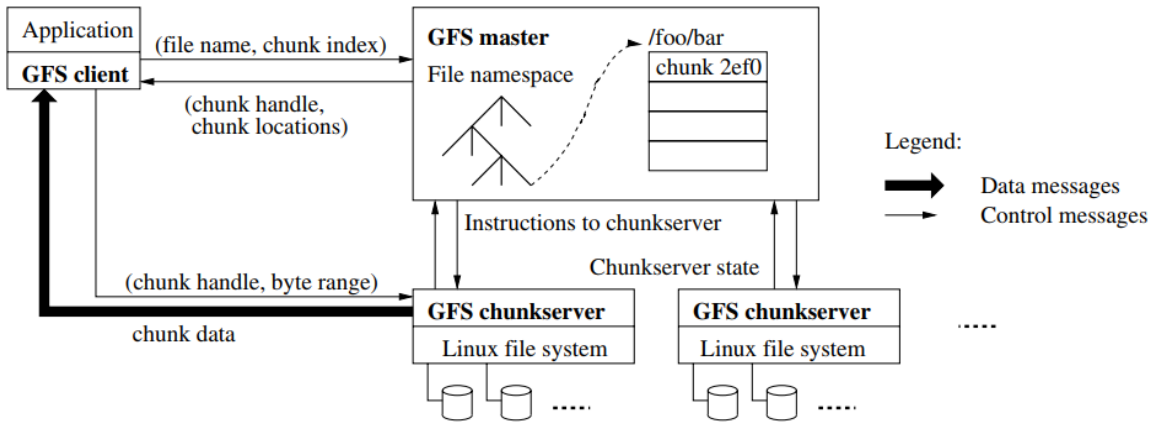

Figure 1: GFS Architecture

Single Master

Simplifies design and allows a simple centralized management. Master stores metadata and co-ordinates access. All metadata is stored in master's memory which makes operations fast. It maintains 64 bytes/chunk. Hence master memory is not a serious bottleneck. In order to minimize master involvement lease mechanism is used. Lease is used to maintain a consistent mutation (append or write) order across replicas.

Chunks

It's of a fixed size of 64MB. Advantages of this are:-

1. Size of Metadata is reduced
2. Involvement of Master is reduced
3. Network Overhead is reduced.
4. Lazy space allocation avoids internal fragmentation

Metadata

There are 3 major types of metadata.

1. The file and chunk namespace
2. The mapping from files to chunks
3. Location of each chunk's replicas

All metadata is kept in Master's memory

Operation Log

It contains a historical record of critical metadata changes. Its replicated to multiple remote machines.All Changes are made visible to clients only after flushing the corresponding log record to disk both locally and remotely

- **CheckPoints**
  Created by Master on separate threads

Relaxed consistency model

1) File namespace mutations are always atomic.
2) File region is consistent if all clients read the same values from replicas.
3) File region is defined if clients see mutation writes in entirety.
4) Operation log ensures metadata stored by master is always consistent and defined.

System Interactions

- Dataflow

  Data is pipeline over TCP connections. Here a chain of chunk servers form a pipeline. Each machine forwards data to the closest machine.
- Snapshot

  Makes a copy of file or a directory tree instaneosly

Master Operation

1. **NameSpace Management & Locking**
   Locks are used over namespaces to ensure proper serialization. GFS simply uses directory file names like: /foo/bar. GFS logically represents its namespace as a lookup table mapping full path names to metadata.
2. **Replica Placement**
   It maximizes data reliability and availability and maximizes network bandwidth utilization
3. **Re-replication**
   The Master Re-replicates a chunk as soon as the number of available replicas falls below a user specified goal
4. **Rebalancing**
   The Mater Rebalances the replicas periodically (maximines replicas distribution and moves replicas for better space and load balancing)
5. **Garbage Collection**
   Lazy deletion of files. Master deleted a hidden file during its regular scan if the file had existed for 3 days. Heartbeat messages are used to inform the chunk servers about the deleted files chunks
6. **Stale Replica Detection**
   The Master maintains a chunk version number. The master removes stale replicas in its regular garbage collection

Fault Tolerance

Fast Recovery, Chunk Replication, Master Replication and Data Integrity.

Advantages:

1) Very high availability and fault tolerance through replication: a) Chunk and master replication and b) Chunk and master recovery.
2) Simple and efficient centralized design with a single master. Delivers good performance for what it was designed for i.e. large sequential reads.
3) Concurrent writes to the same file region are not serializable. Thus replicas might have duplicates but there is no interleaving of records. To ensure data integrity each chunkserver verifies integrity of its own copy using checksums.
4) Read operations span at least a few 64KB blocks therefore the check summing costs reduces.
5) Batch operations like writing to operation log, garbage collection help increase the bandwidth.
6) Atomic append operations ensures no synchronization is needed at client end.
7) No caching eliminates cache coherence issues.
8) Decoupling the flow of data from control allows us to use the network efficiently.
9) Orphaned chunks are automatically collected using garbage collection.
10) GFS master constantly monitors each chunkserver through heartbeat messages.