The Google File System

Authors : Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung

Presentation by: Vijay Kumar Chalasani

Introduction

- GFS is a scalable distributed file system for large data intensive applications
- 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**

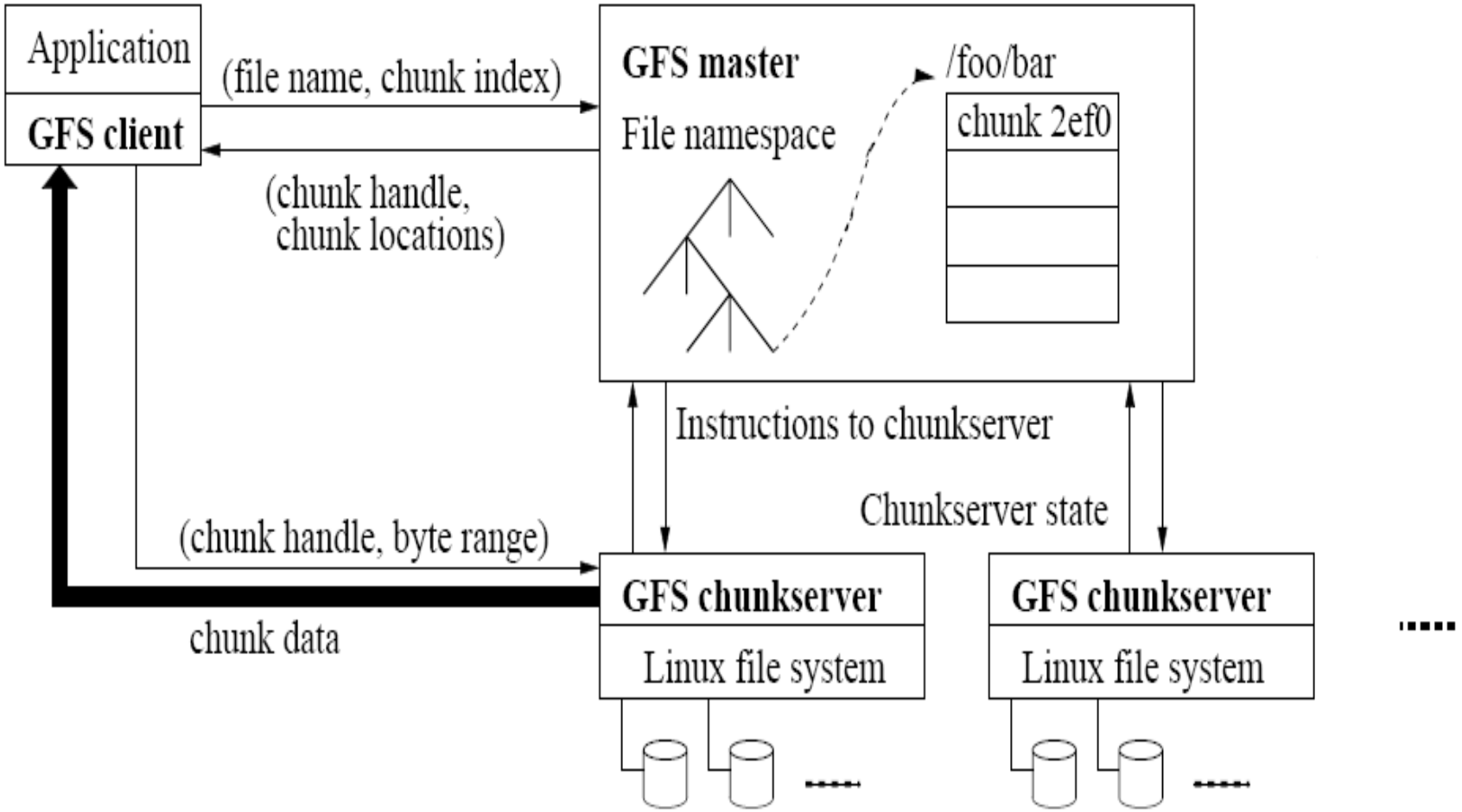
Assumptions

- **GFS has high component failure rates**
  - System is built from many inexpensive commodity components
- **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
- **Workloads : two kinds of reads, and writes**
  - Large streaming reads (1MB or more) and small random reads (a few KBs)
  - Small random reads
  - Sequential appends to files by hundreds of data producers
- **High sustained throughput is more important than latency**
  - Response time for individual read and write is not critical

GFS Design Overview

- **Single Master**
  - Centralized management
- **Files stored as chunks**
  - With a fixed size of 64MB each.
- **Reliability through replication**
  - Each chunk is replicated across 3 or more chunk servers
- **Data caching**
  - Due to large size of data sets
- **Interface**
  - Suitable to Google apps
  - Create, delete, open, close, read, write, snapshot, record append

GFS Architecture



Master

- **Mater maintains all system metadata**
  - Name space, access control info, file to chunk mappings, chunk locations, etc.
- **Periodically communicates with chink servers**
  - Through HeartBeat messages
- **Advantages:**
  - Simplifies the design
- **Disadvantages:**
  - Single point of failure
- **solution**
  - Replication of Master state on multiple machines
  - Operational log and check points are replicated on multiple machines

Chunks

- **Fixed size of 64MB**
- **Advantages**
  - Size of meta data is reduced
  - Involvement of Master is reduced
  - Network overhead is reduced
  - Lazy space allocation avoids internal fragmentation
- **Disadvantages**
  - Hot spots
    - **Solutions: increase the replication factor and stagger application start times; allow clients to read data from other clients**

Metadata

- **Three major types of metadata**
  - The file and chunk namespaces
  - The mapping from files to chunks
  - Locations of each chunk's replicas
- **All the metadata is kept in the Master's memory**
- **Master "operation log"**
  - Consists of namespaces and file to chunk mappings
  - Replicated on remote machines
- **64MB chunk has 64 bytes of metadata**
- **Chunk locations**
  - Chunk servers keep track of their chunks and relay data to Master through HeartBeat messages

Operation log

- **Contains a historical record of critical metadata changes**
- **Replicated to multiple remote machines**
- **Changes are made visible to clients only after flushing the corresponding log record to disk both locally and remotely**
- **Checkpoints**
  - Master creates the checkpoints
  - Checkpoints are created on separate threads

Consistency Model

- Atomicity and correctness of file namespace are ensured by namespace locking
- After successful data mutation(writes or record appends), changes are applied to a chunk in the same order on all replicas.
- In case of chunk server failure at the time of mutation (stale replica), it is garbage collected at the soonest opportunity.
- Regular handshakes between Master and chunk servers helps in identifying failed chunk servers and detects data corruption by checksumming.

System Interactions: Leases & Mutation Order

- Master grants chunk lease to one of the replicas(primary).
- All replicas follow a serial order picked by the primary.
- Leases timeout at 60 seconds.(also possible to extend the timeout)
- Leases are revocable.

System Interactions

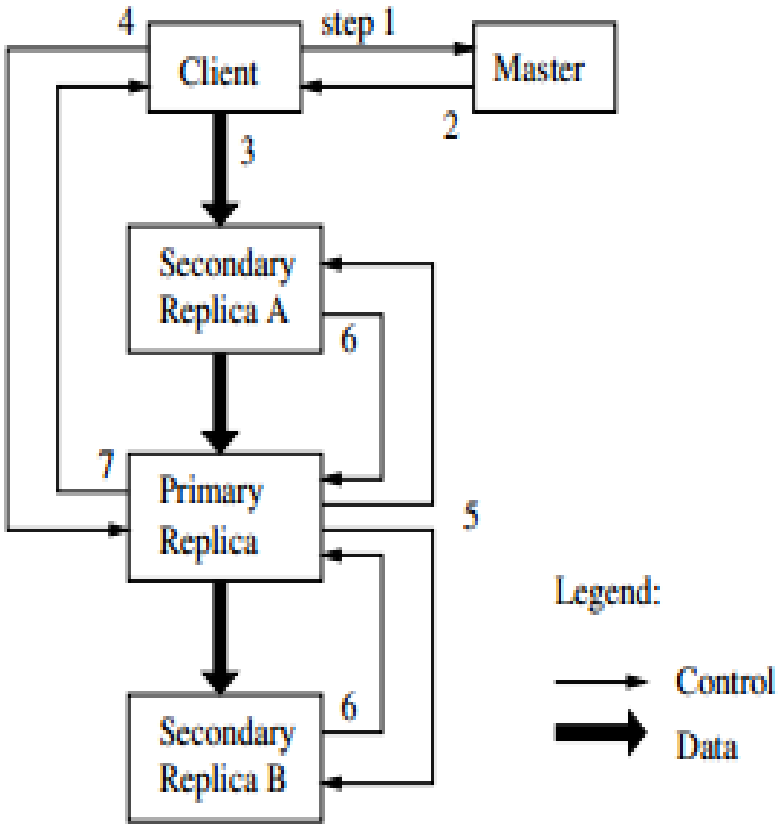


Figure 2: Write Control and Data Flow

- 1.Client asks master which chunk server holds current lease of chunk and locations of other replicas.
- 2.Master replies with identity of primary and locations of secondary replicas.
- 3.Client pushes data to all replicas
- 4.Once all replicas have acknowledged receiving the data, client sends write request to 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 write request to all secondary replicas. They apply mutations in the same serial number order.
- 6.Secondary recplicas reply to primary indicating they have completed operation
- 7.Primary replies to the client with success or error message

System Interactions

- **Data Flow**
  - Data is pipelined over TCP connections
  - A chain of chunk servers form a pipeline
  - Each machine forwards data to the closest machine
- **Atomic Record Append**
  - "record append"
- **Snapshot**
  - Makes a copy of file or a directory tree almost instantaneously

Master Operation – Namespace Management & Locking

- Locks are used over namespaces to ensure proper serialization
- Read/write locks
- GFS simply uses directory like file names : /foo/bar
- GFS logically represents its namespace as a lookup table mapping full pathnames to metadata
- If a Master operation involves /d1/d2/../dn/leaf, read locks are acquired on d1,/d1/d2,..d1/d2/../leaf and either a read or write lock on the full pathname /d1/d2/…..dn/leaf

Master Operation

- Replica Placement
  - **Maximize data reliability and availability**
  - **Maximize network bandwidth utilization**
- Re-replication
  - **The Master Re-replicates a chunk as soon as the number of available replicas falls below a user specified goal**
- Rebalancing
  - **The Master Rebalances the replicas periodically (examines replicas distribution and moves replicas for better disk space and load balancing)**

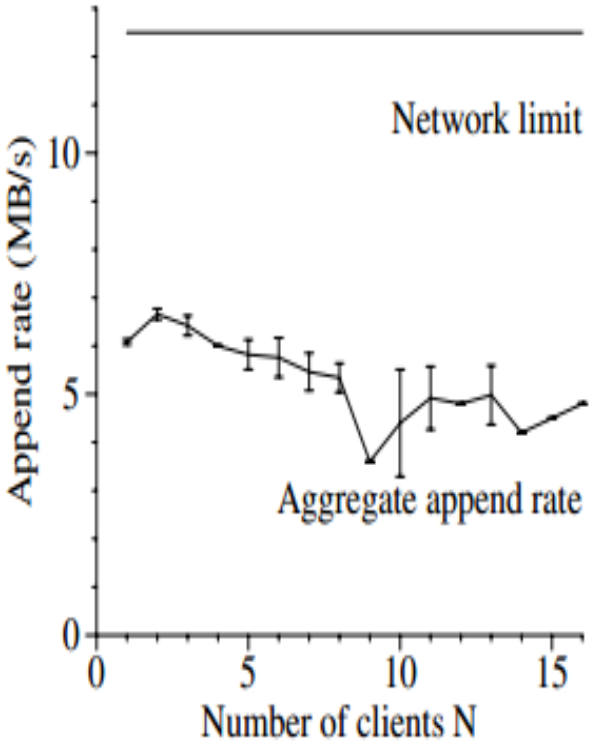
Master Operation

- Garbage collection
  - **Lazy deletion of files**
  - **Master deletes a hidden file during its regular scan if the file have existed for 3 days**
  - **HeartBeat messages are used to inform the chunk servers about the deleted files chunks**
- Stale Replica Detection
  - **The Master maintains a chunk version number**
  - **The Master removes stale replicas in its regular garbage collection**

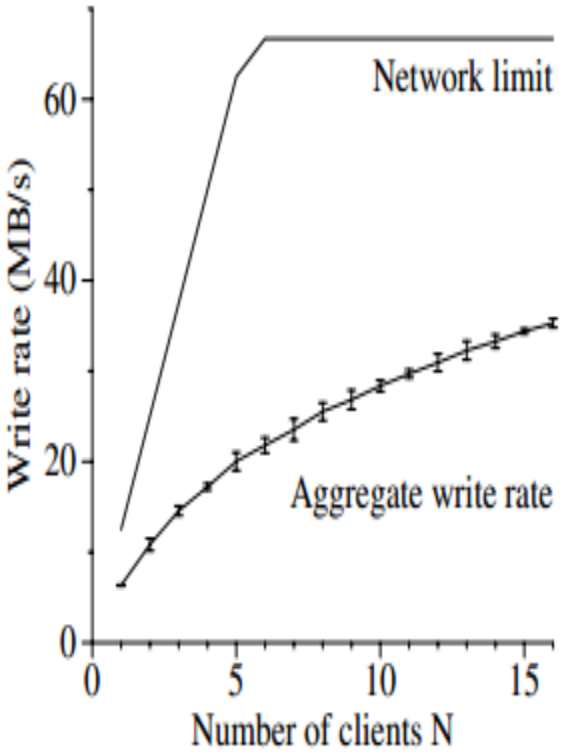
Fault Tolerance

- Fast Recovery
- Chunk Replication
- Master Replication
- Data Integrity

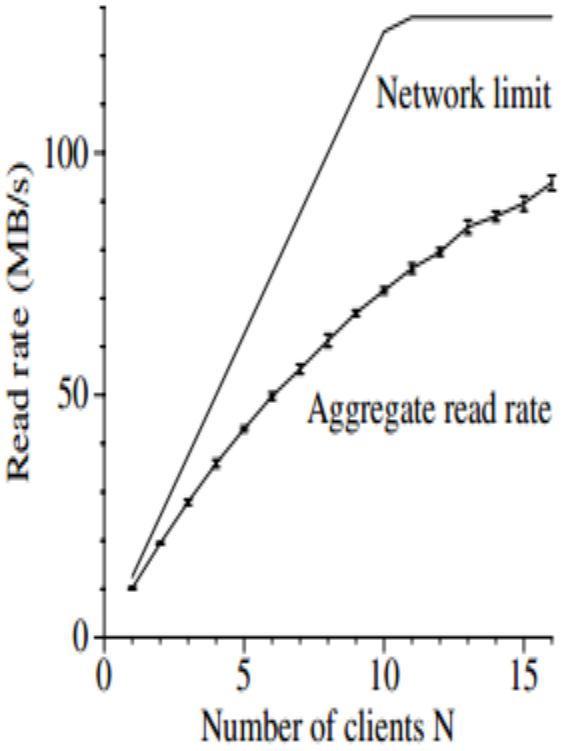
Aggregate Throughputs



(c) Record appends



(b) Writes



(a) Reads

Characteristics & Performance

Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

References

- http://courses.cs.vt.edu/cs5204/fall12-kafura/Papers/FileSystems/GoogleFileSystem.pdf
- http://www.youtube.com/watch?v=5Eib_H_zCEY
- http://media1.vbs.vt.edu/content/classes/z3409_cs5204/cs5204_27GFS.html