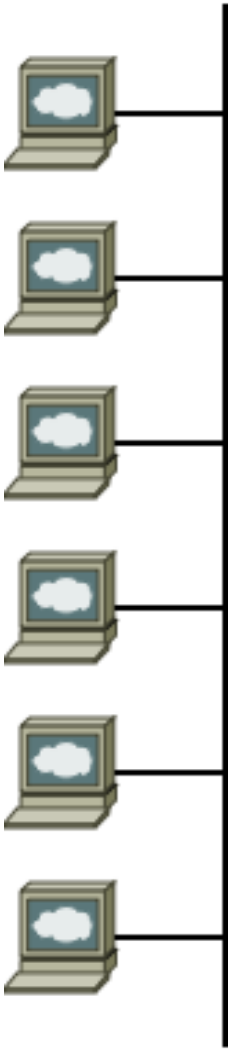
GOOGLE FILE SYSTEM (GFS)

Hans Vatne Hansen

UNIVERSITY OF OSLO

Distributed File Systems



- Andrew File System (AFS)
- Network File System (NFS)
- Coda
- Microsoft Distributed File System (DFS)
- Apple Filing Protocol (AFP)

Distributed File Systems

Andrew File System

- Performance
- Scalability
- Availability
- Security

Coda

CODA = AFS +

- Disconnected operation
  - For mobile computing
  - For partial network failures
- Network bandwidth adaptation
- Client side caching
- Server replication

Motivation for GFS

- Nothing is small in Google land
  - Peta-bytes of data
  - Millions of users
  - Lots of services and servers

  → **Scalability**
- Failures are normal
  - Network connections
  - Hard disks
  - Power supplies

  → **Fault tolerance**
- Monitoring and maintenance is hard

  → **Autonomic computing**

- Clusters all over the world
- Thousands of queries served per second
  - One query reads hundreds of MB of data
  - One query consumes billions of CPU cycles
- A distributed, fault-tolerant file system is needed!

Google Data Centers

- Scaling out on commodity hardware is cheaper than scaling up on high-end servers
- Google servers:
  - > 15 000 servers (2003)
  - ~ 200 000 (2005)
  - ~ 1 M servers (2010)
- Data centers are composed of standard shipping containers with 1160 servers in each

Google Data Centers

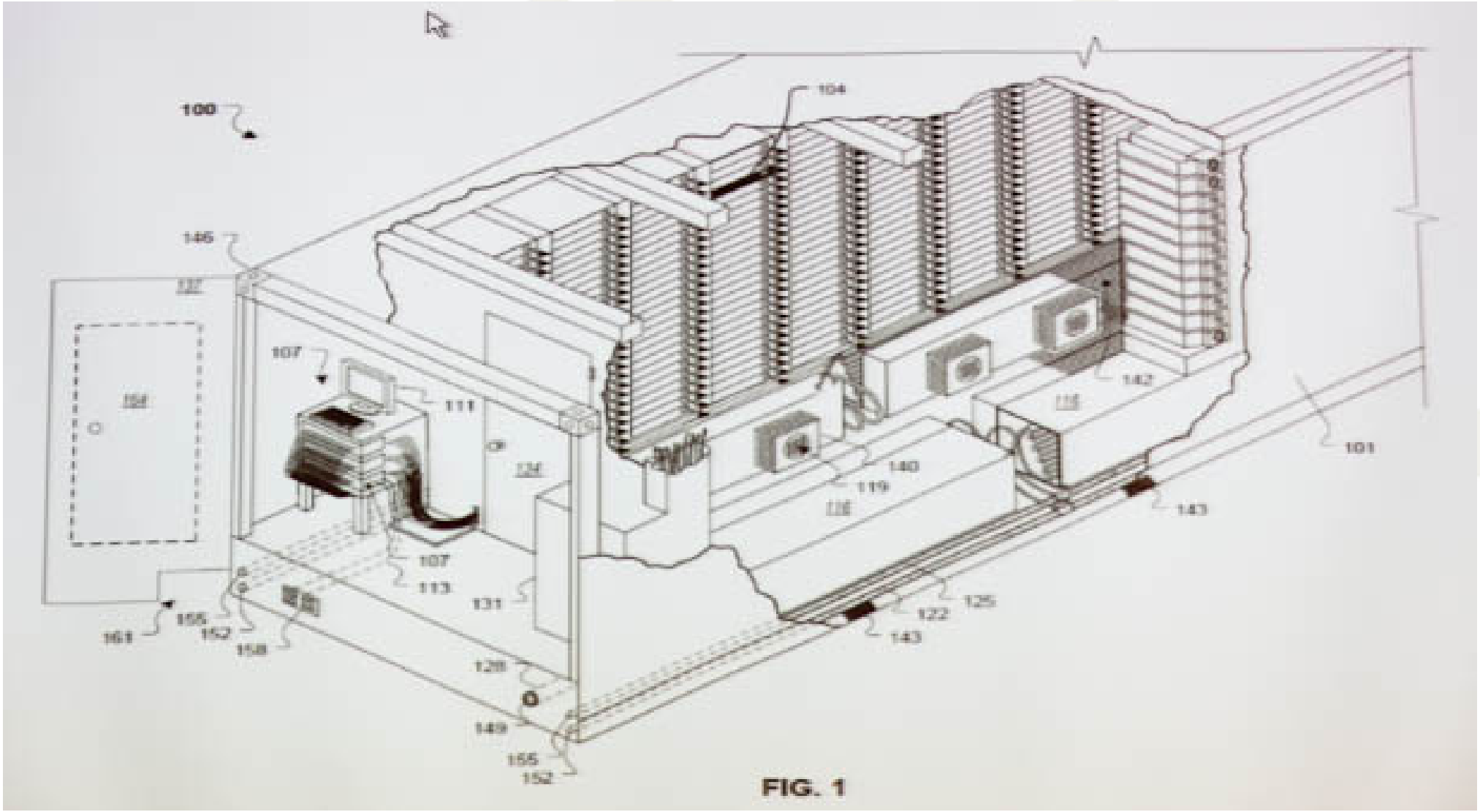
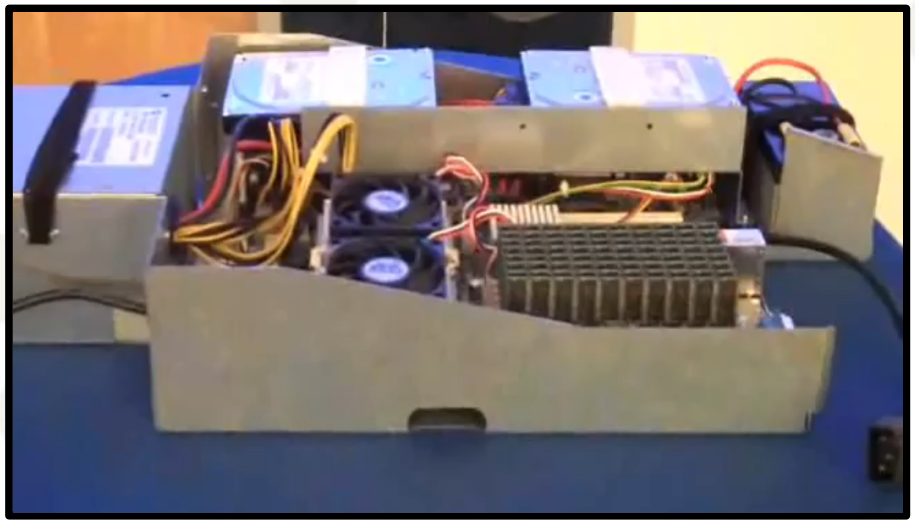


FIG. 1

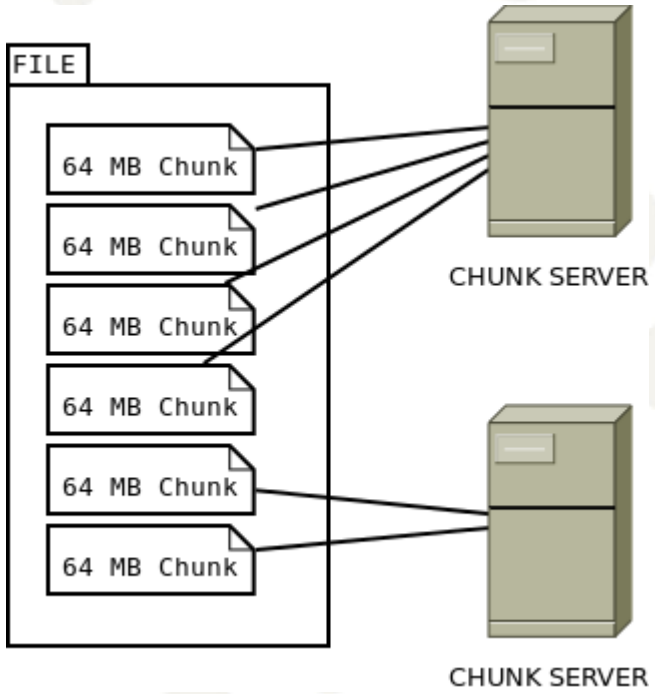
Google Data Centers



Chunks and Chunk Servers

Chunk

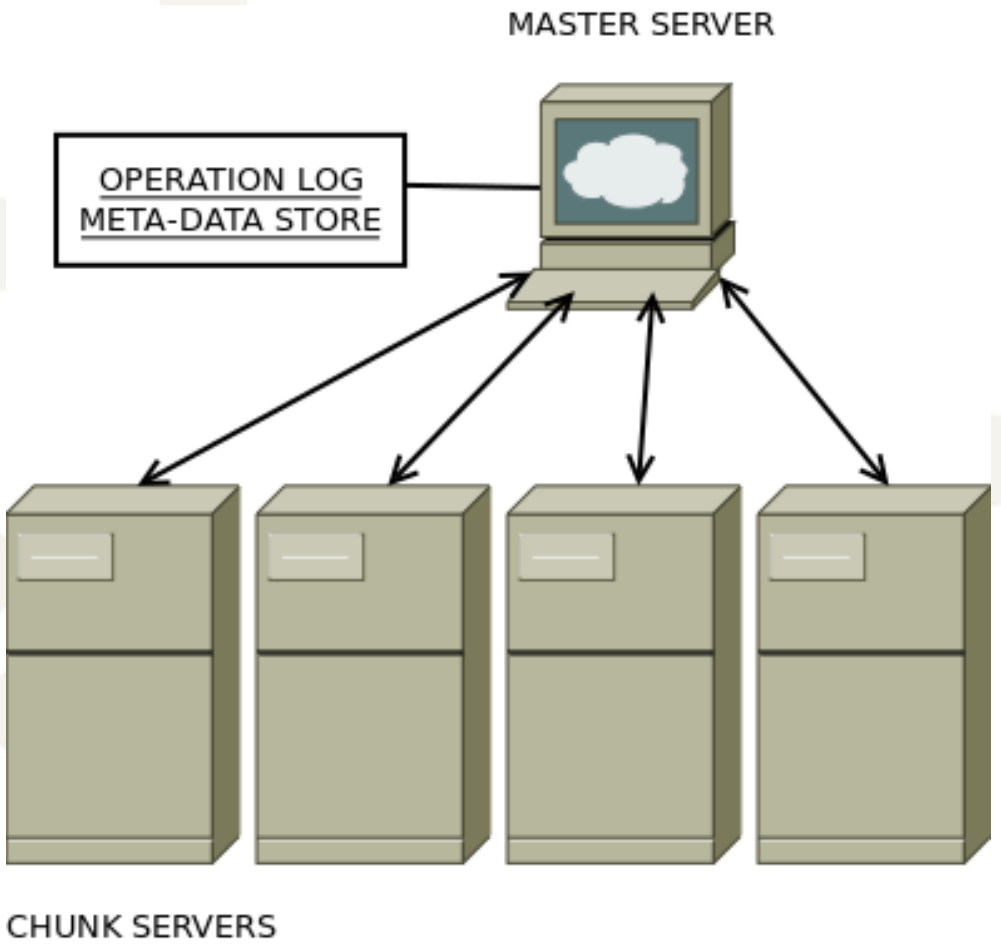
- Similar to block in file systems
- Size is always 64 MB
- Less fragmentation
- Eases management
- Sent directly to clients



Master Servers

Master Server

- Coordinates cluster
- Updates operation log
- Stores meta-data



Master Server – Chunk Server Communication

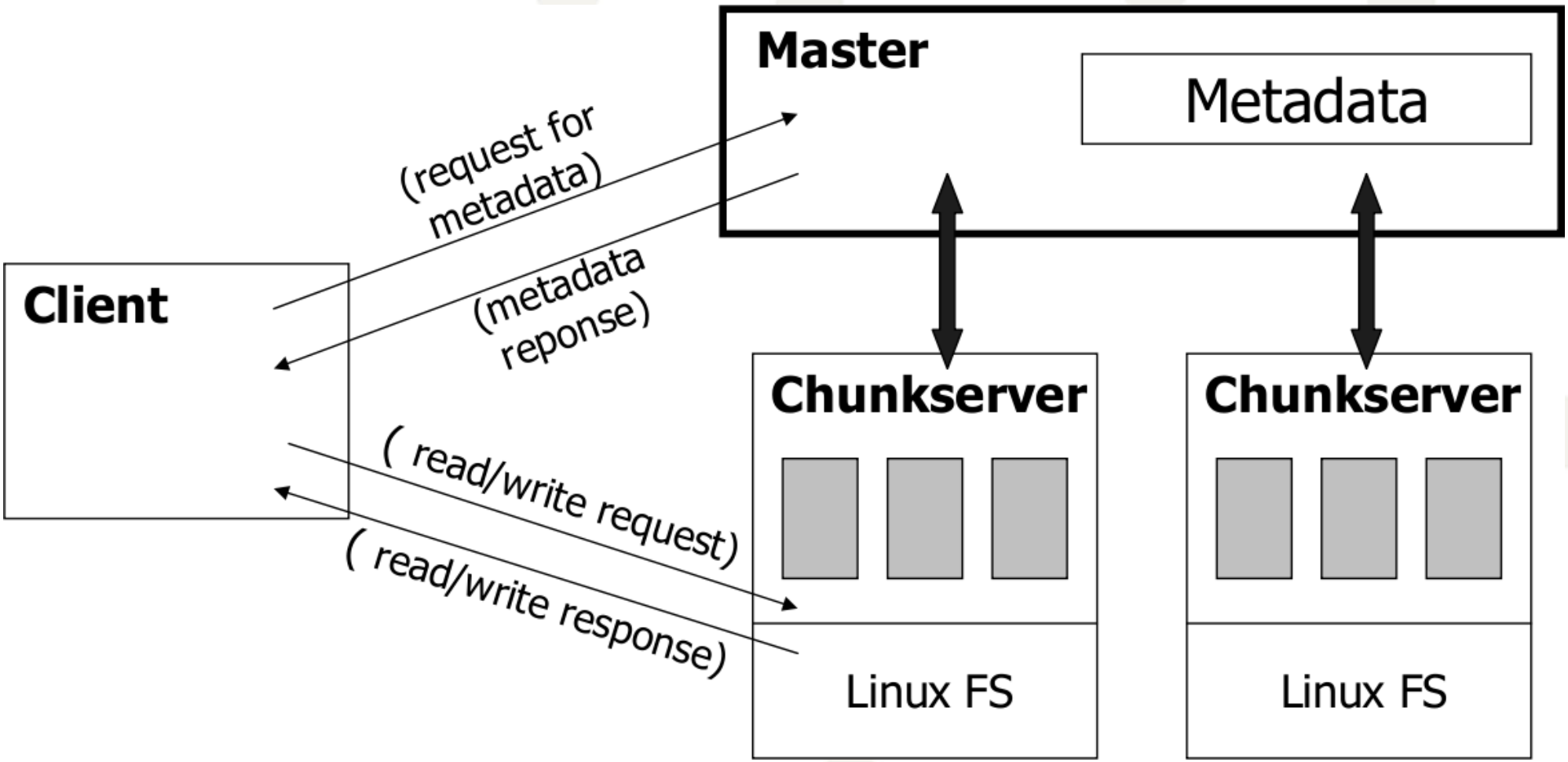
State updates

- Is a chunk server down?
- Are there disk failures on a chunk server?
- Are any replicas corrupted?
- Which chunk replicas does a chunk server store?

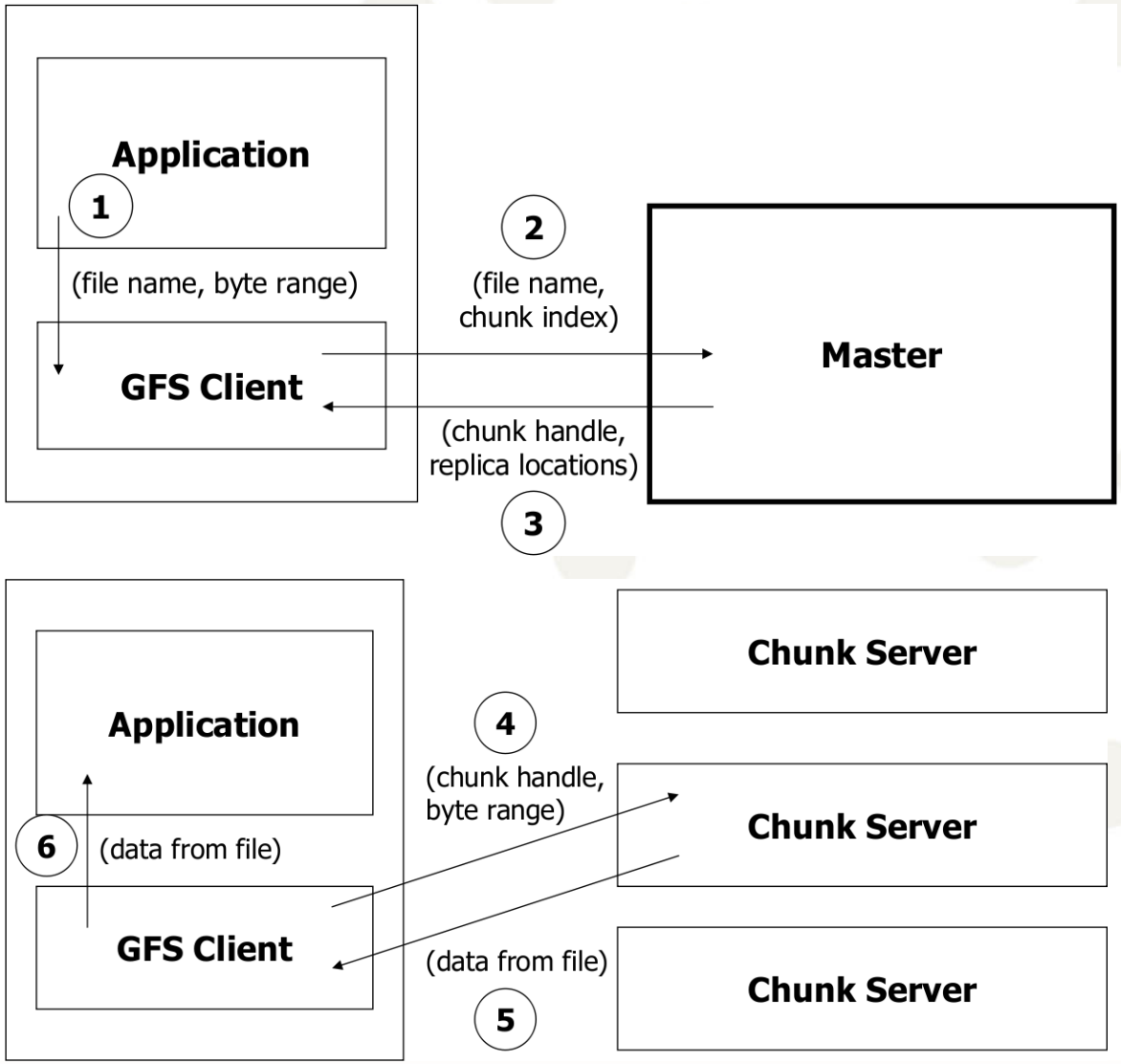
Instructions

- Create new chunk
- Delete existing chunks

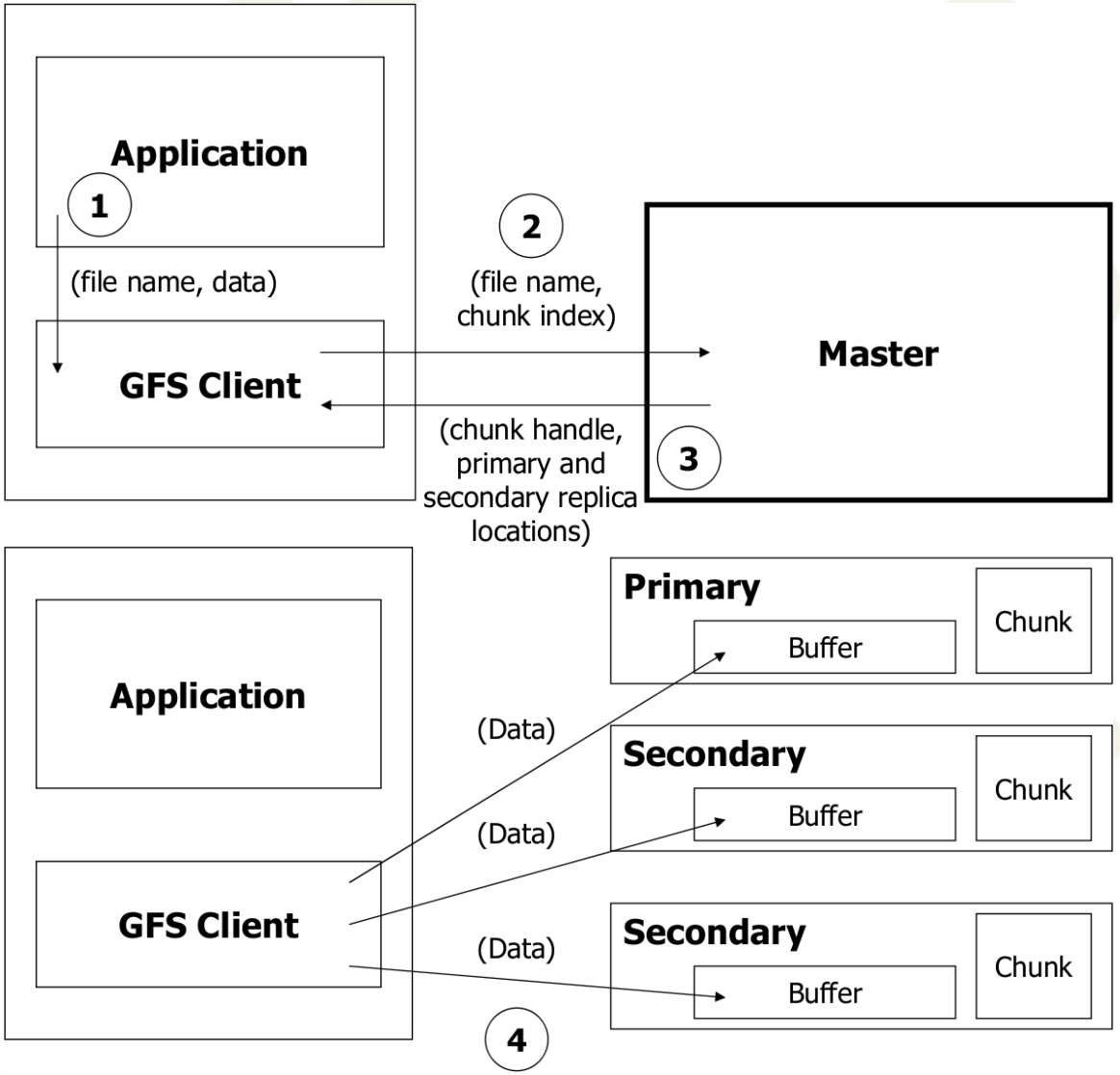
GFS Architecture



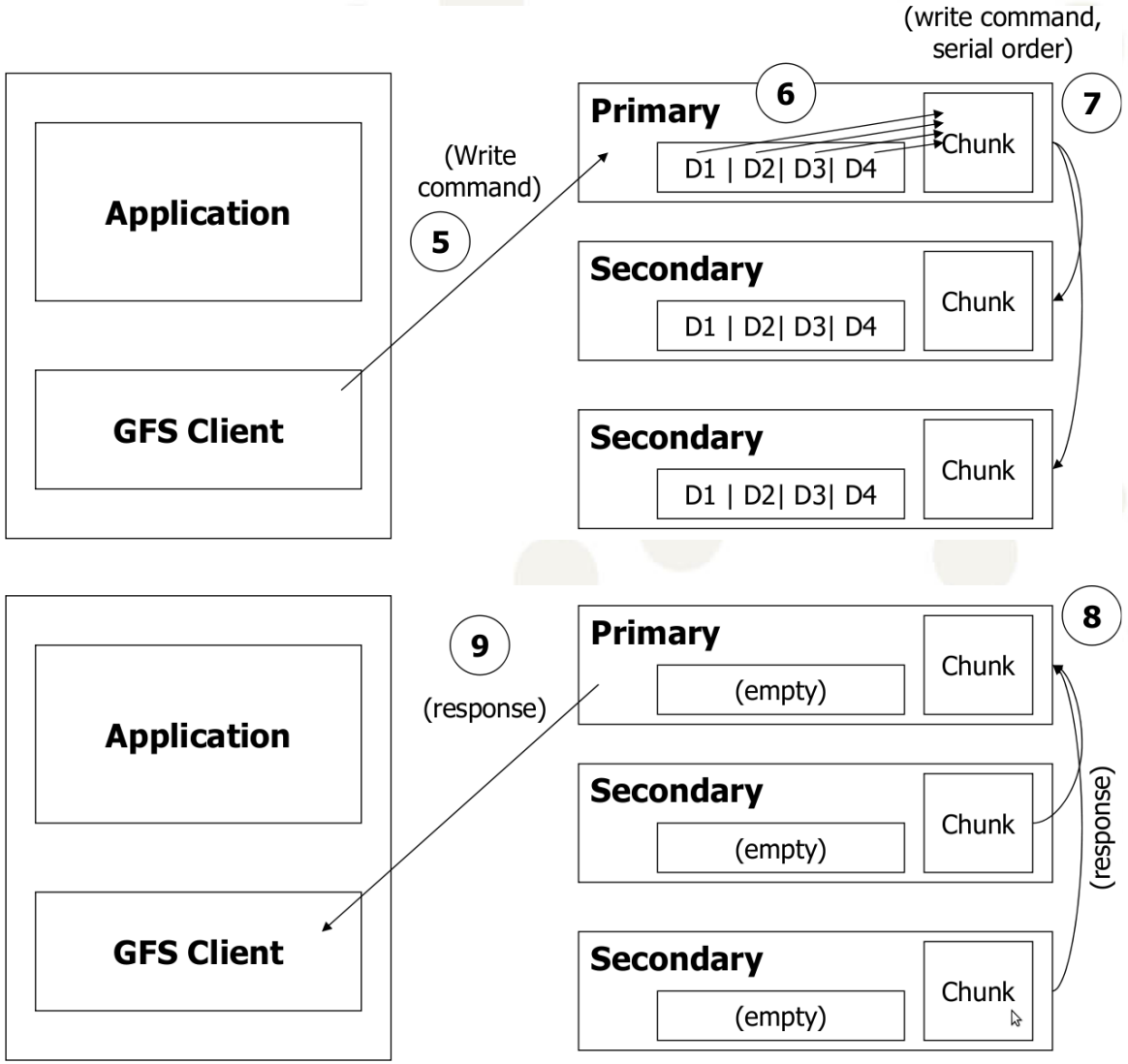
Read operation



Write operation (1/2)



Write operation (2/2)



Record Append

- Record append allows multiple clients to append data to the same file concurrently while guaranteeing atomicity

Algorithm

- Application originates record append request.
- GFS client translates request and sends it to master.
- Master responds with chunk handle and (primary + secondary) replica locations.
- Client pushes write data to all locations.
- Primary check if record fits in specified chunk.
- If record does not fit, then the primary:
  - Pads the chunk.
  - Tells secondaries to do the same.
  - Informs the client.
  - Client retries append with the next chunk.
- If records fits, then the primary:
  - Appends the record.
  - Tells secondaries to do the same.
  - Receives responses from secondaries.
  - Send final response to the client.

Fault Tolerance

- Master and chunk server recovers extremely fast
- Chunks, operation log and master state is replicated
- Replication is done across multiple machines and data centers in case of severe failures

Conclusions

- GFS has
  - Performance
  - Scalability
  - Fault-tolerance
- GFS is
  - Easy to maintain
  - Cheapest solution for Google
- Clients and applications can
  - Read in parallel
  - Write in parallel
  - Append in parallel

References

- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, **The Google File System**, ACM Symposium on Operating Systems Principles, 2003
- Naushad UzZaman, **Survey on Google File System**, CSC 456 (Operating Systems), 2007
- Jonathan Strickland, **How the Google File System Works**, HowStuffWorks.com, 2010
- Wikipedia Contributors, **Google File System**, Wikipedia - The Free Encyclopedia, 2010