

Distributed File System: NFS, AFS, GPFS, and GFS

Instructor

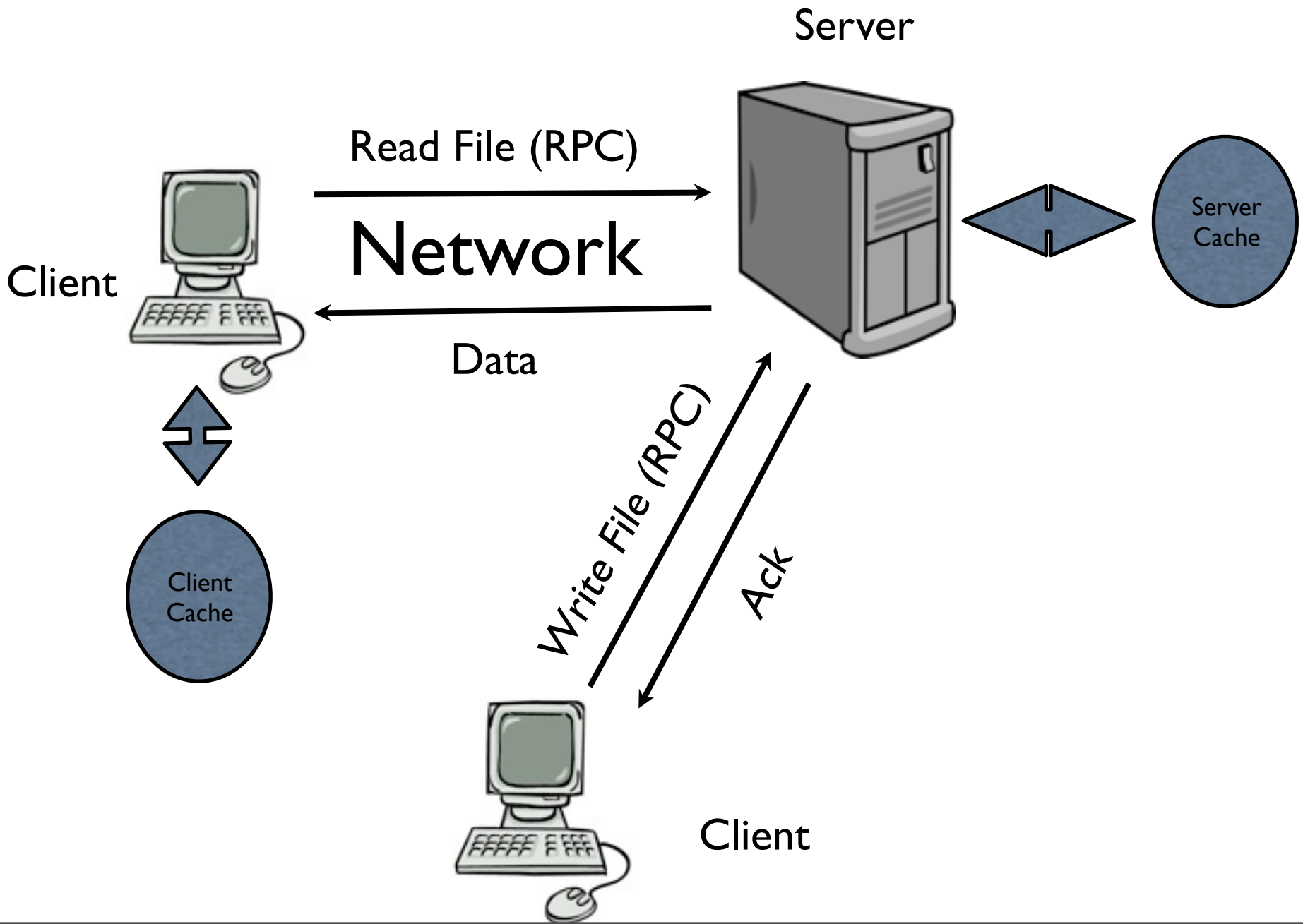
Sandhya Dwarkadas

Presenters

Tiantong Yu and Phyo Thiha

November 08, 2011

A Simple Model of a Distributed File System



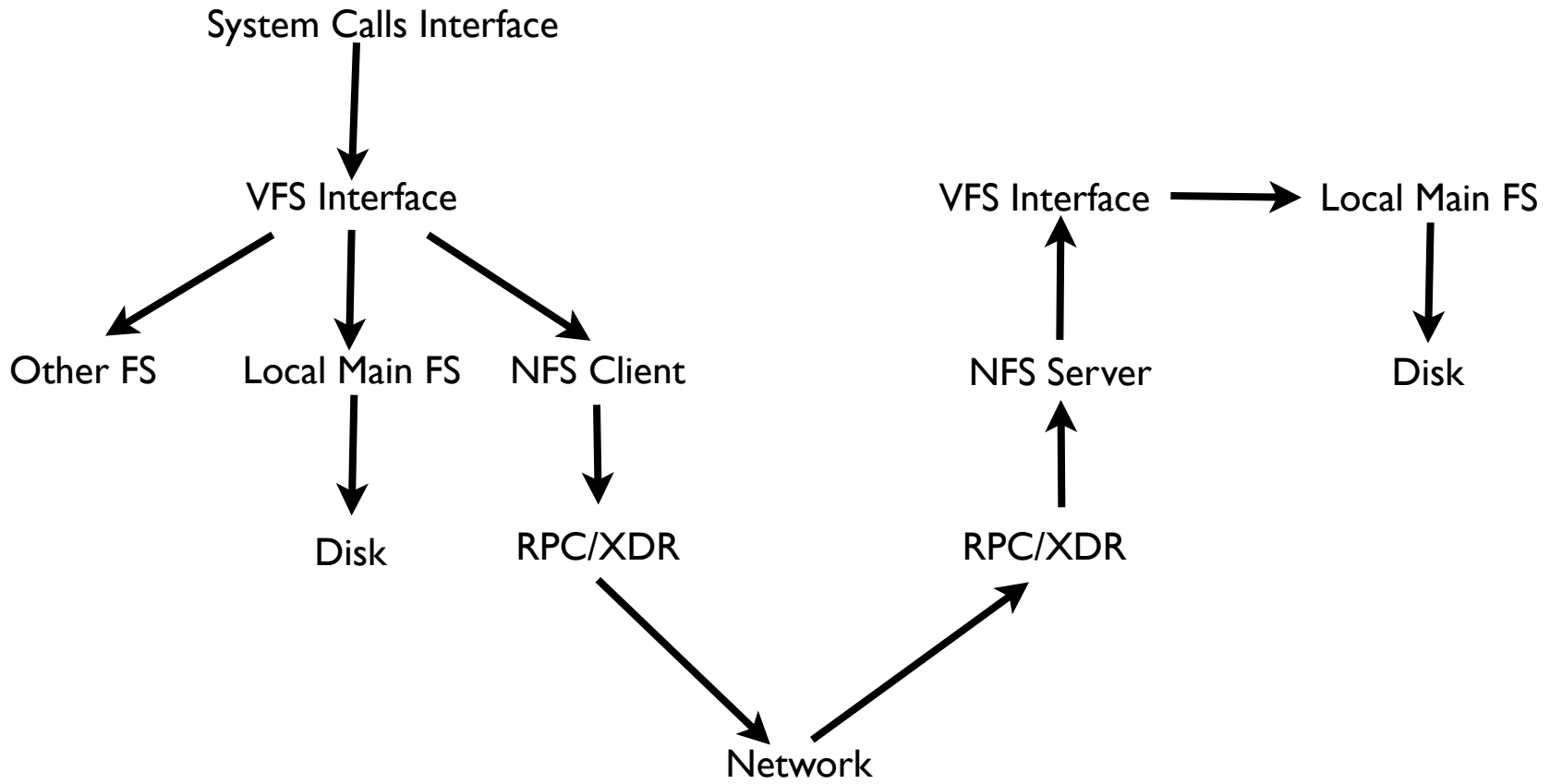
Issues To Consider For A Distributed File System

- Name Resolution
- Security
- **Consistency**
 - Are client copy and server copy of data consistent?
 - Are metadata and actual file data consistent?
- **Synchronization**
 - Simultaneous reads and writes
- **Reliability**
 - What happens when crashes or disk failures occur
- **Scalability** (Related to performance)
 - throughput, network load, delay

Network File System (NFS)

- Three layers (see picture)
- RPC for file operations on server
 - read, write
 - search/maintain directories
 - access metadata
- Write-through caching - safer but slower
- Stateless - Quick recovery from server failure
- Polling, no synchronization (better for newer versions)
- Bad scalability
- But simple and portable

NFS



Andrew File System (AFS)

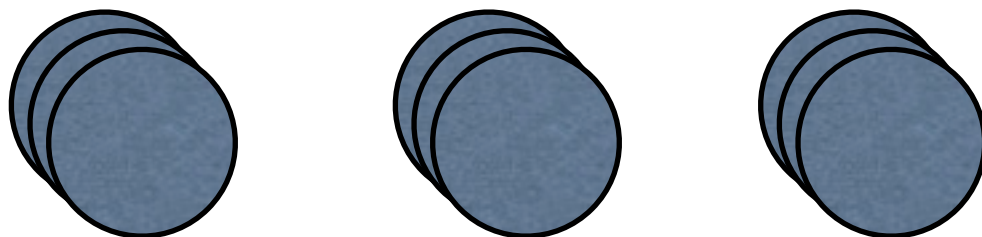
- Linux-like interface
- Unified namespace (fid)
- Centrally manages clients' states
- Use callbacks to maintain consistency
- Good Scalability
- Enhanced Security (access control list, encrypted password)

Special Case: GPFS

- Designed for large computing clusters (support up to 4096 1TB disks), truly parallel
- Imitates the behavior of a general POSIX file system
- Integrate data striping in the file system
- Divide data into large trunks (256k) and distributed in round-robin.
- Use logging system to recover from faults
- Use distributed locking to ensure synchronization (vs. central management)
- There is still a token manager needed (possible bottleneck)

GPFS Structure

File System Nodes



Disks

Name Resolution

- Hostname:local-path-name
- Mounting
- Globally unique file name
 - Need some name translation service
 - AFS fids
 - Common in peer-to-peer systems

Consistency

- Goal: every client sees the same data
- Sources of inconsistency
 - client crashes before a write completes
 - server crashes
 - loses data in memory (metadata and actual file inconsistency)
 - loses client states (what might happen?)
 - caching
 - disk failures

Consistency - Con't

- Solutions

- write-through caching (eg. NFS)
 - modified data must be committed to server first
 - pros and cons ?
- stateless protocols (eg. NFS)
 - `readAt(inumber, position)` instead of `read(filename)`
- polling (eg. NFS), call-backs(eg. AFS)

Synchronization

- Not much different from general synchronization model
- There is a tradeoff between synchronization and performance
- NFS does nothing about synchronization
 - race conditions possible
- AFS uses write-on-close
 - no partial writes
 - do not get newer versions until reopen
- GPFS uses distributed locking protocol

Reliability

- Server data integrity (e.g. replicas, RAID system)
- Recover from client crash
 - not big deal, just need to ensure data consistency (refer to consistency)
- Recover from server crash
 - wait-and-continue (NFS) (pros and cons?)
 - recover from information kept by clients (AFS)
 - recovers from log of operations (GPFS)

Security

- Normal file system securities (e.g. access control, identity authentication)
- Network considerations (encrypted message)

Scalability

- Depends on the implementation decision on other issues
- At the same time, important factor that affects the implementation decisions on other issues
- Special efforts to improve scalability
 - GPFS's striping (e.g enable control over load balancing)
- NFS does not scale well (why?)
- AFS scales better (why?)
- GFS and GPFS has excellent scalability (teaser)

Big Picture of GFS

▶ Workload

- > one billion searches a day¹
- Gmail, Google Maps, YouTube, ...
- Network Transactions

▶ Goals

- ▶ Scalable, Reliable, Available
- ▶ Cheap?

▶ 1. Retrieved on November 06, 2011 from: <http://www.forbes.com/sites/quentinhardy/2011/06/11/google-scale-changes-everything/>

Big Picture of GFS

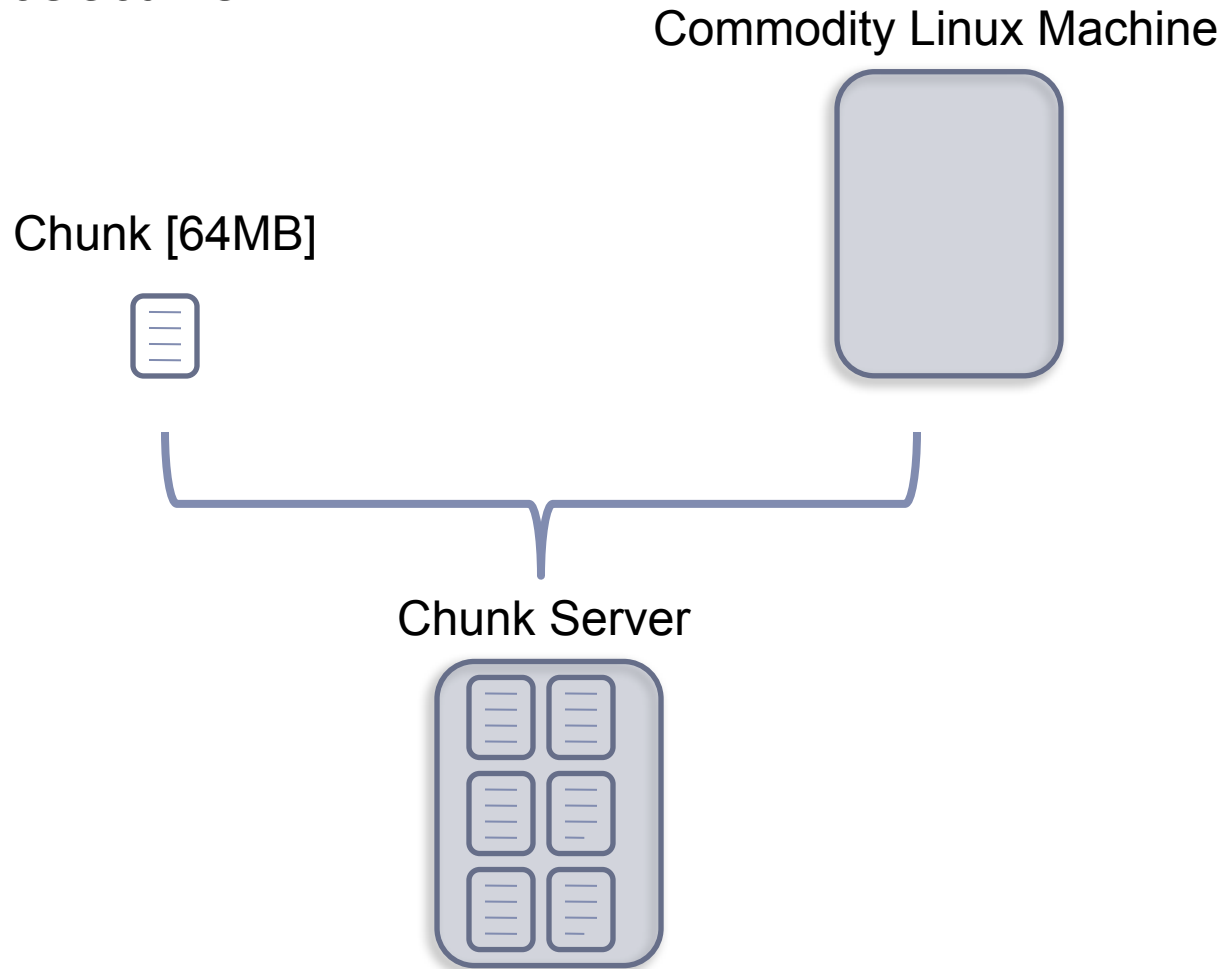
▶ Assumptions

- Failures are norm
- Files are HUGE (multi GB)
- Sequential reads
- Append to write
- Hundreds of producers write concurrently
- **Bandwidth** more important than **Latency**



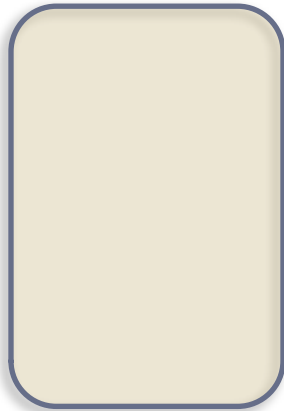
Big Picture of GFS

► Architecture

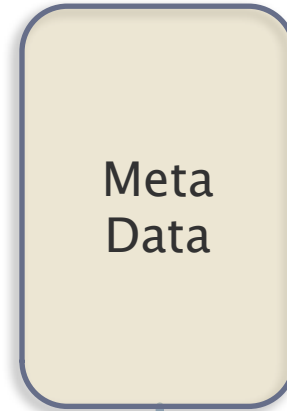


Big Picture of GFS

Commodity Linux Machine



Master Server

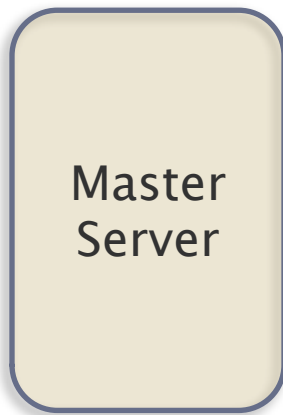


- name space
- access control
- mappings of files to chunk
- location of chunks
- chunk version numbers



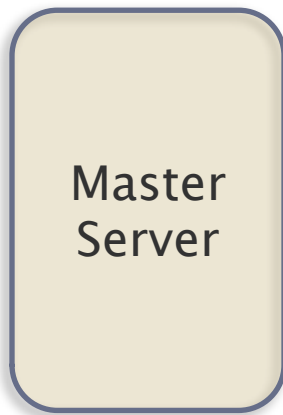
Big Picture of GFS

- ▶ Master – Chunk Servers Relationship



Big Picture of GFS

- ▶ Master – Chunk Servers Relationship

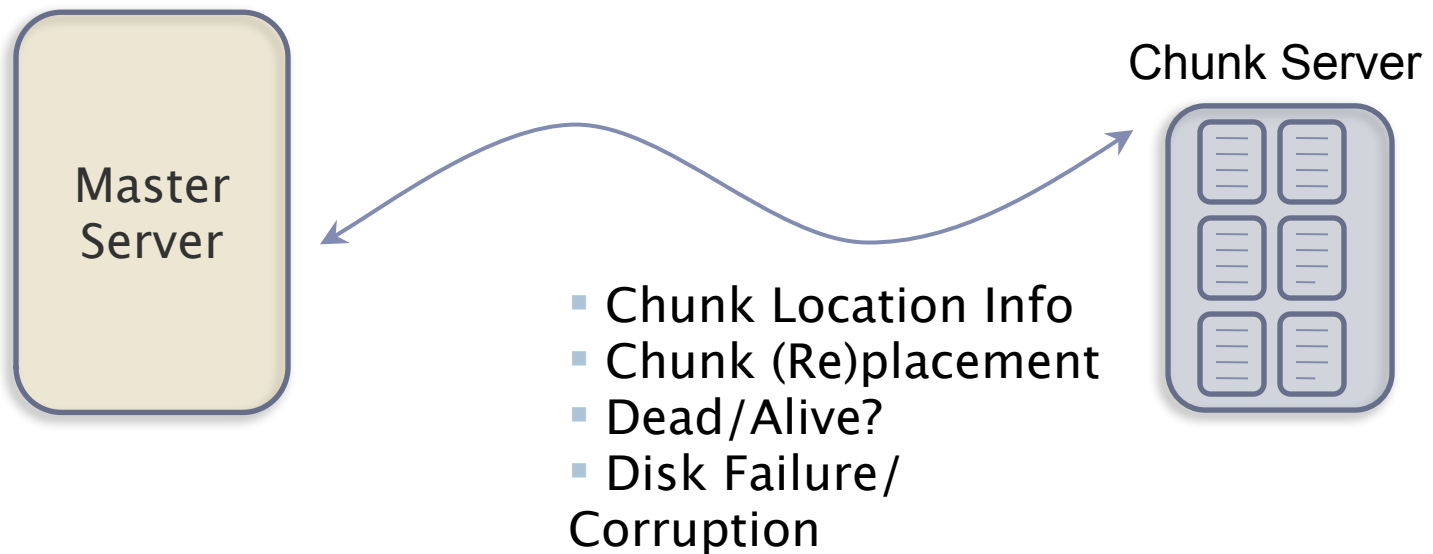


Chunk Server



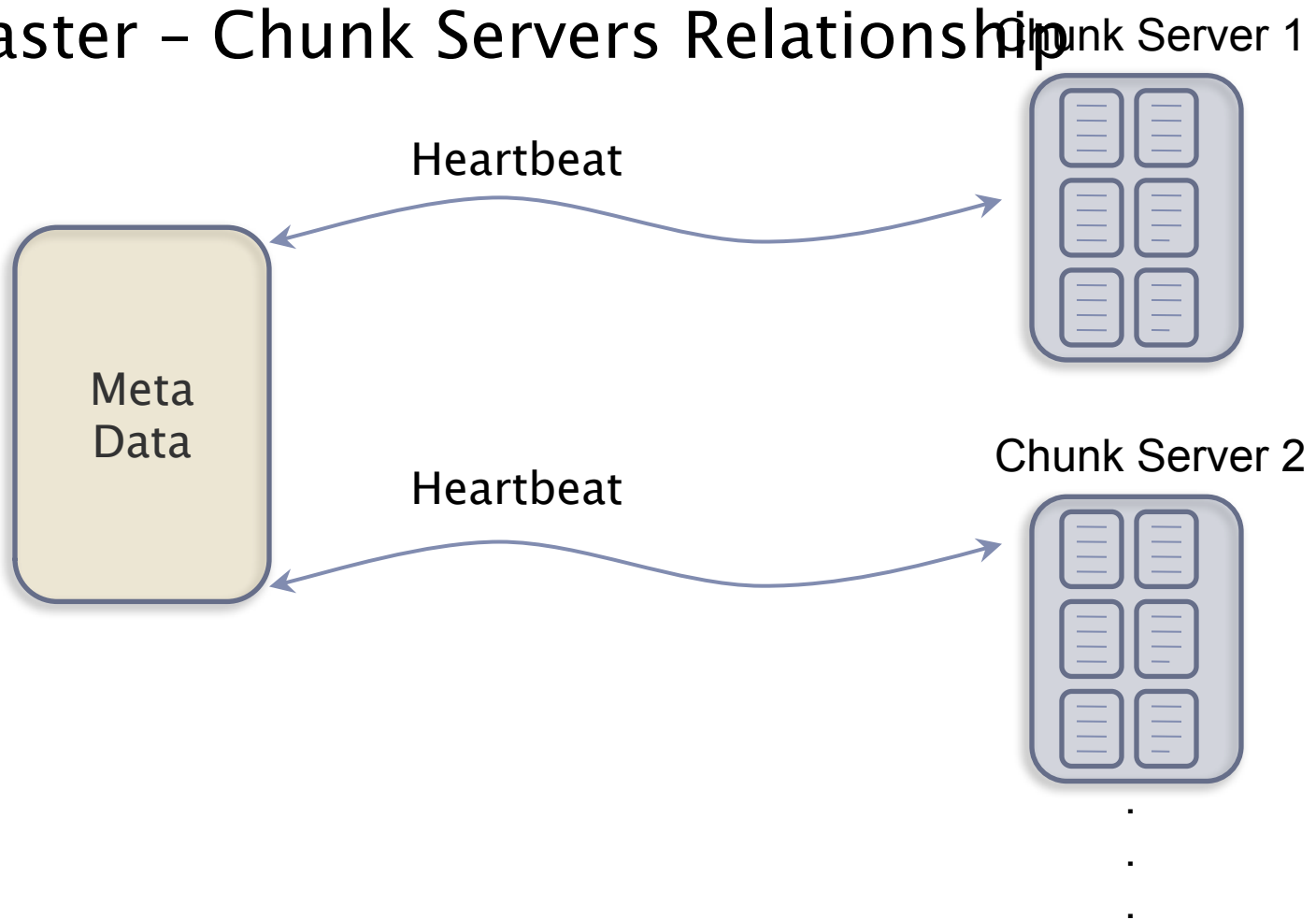
Big Picture of GFS

▶ Master – Chunk Servers Relationship



Big Picture of GFS

▶ Master - Chunk Servers Relationship



Big Picture of GFS

▶ Clients



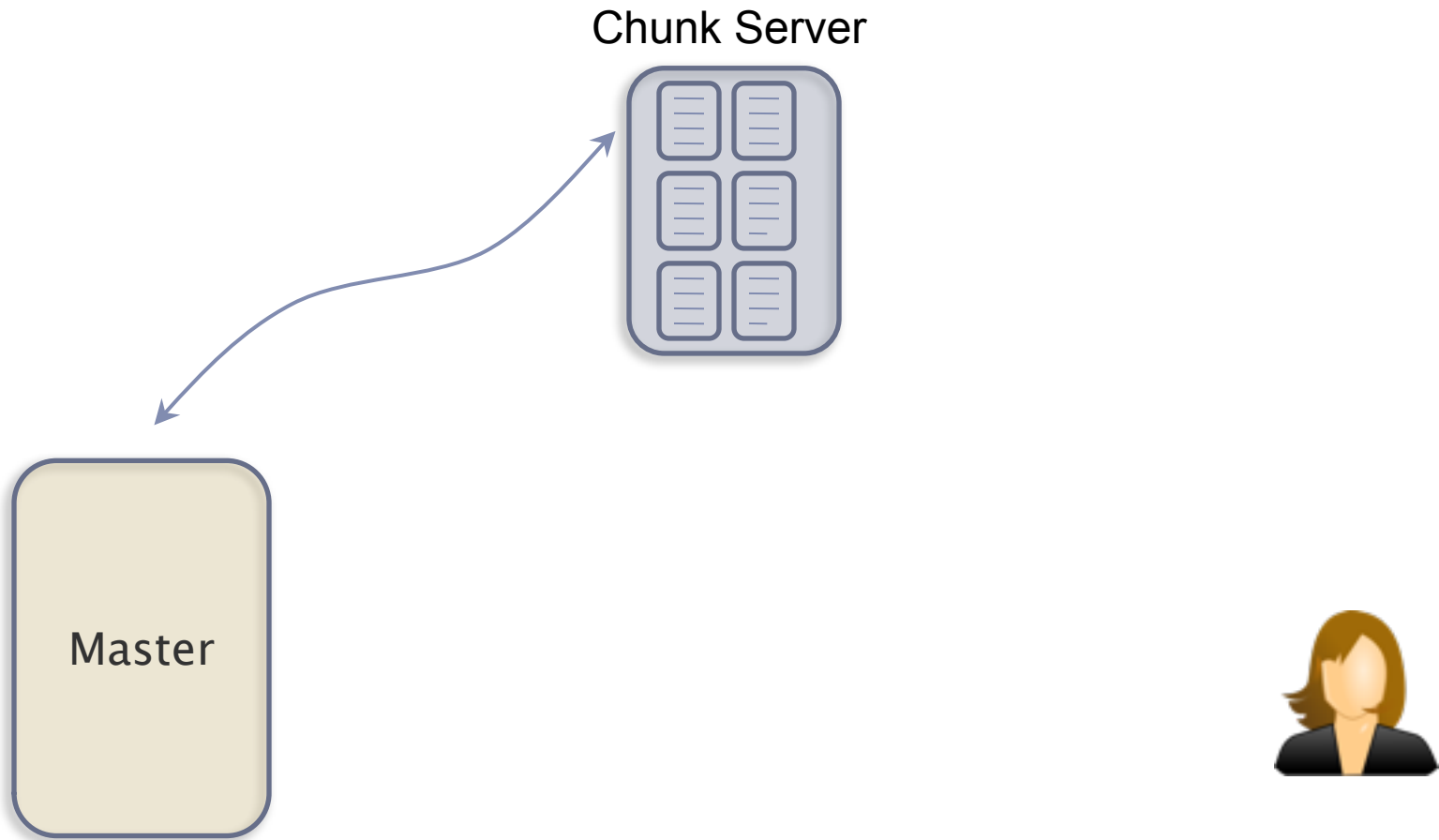
Google
Programmer



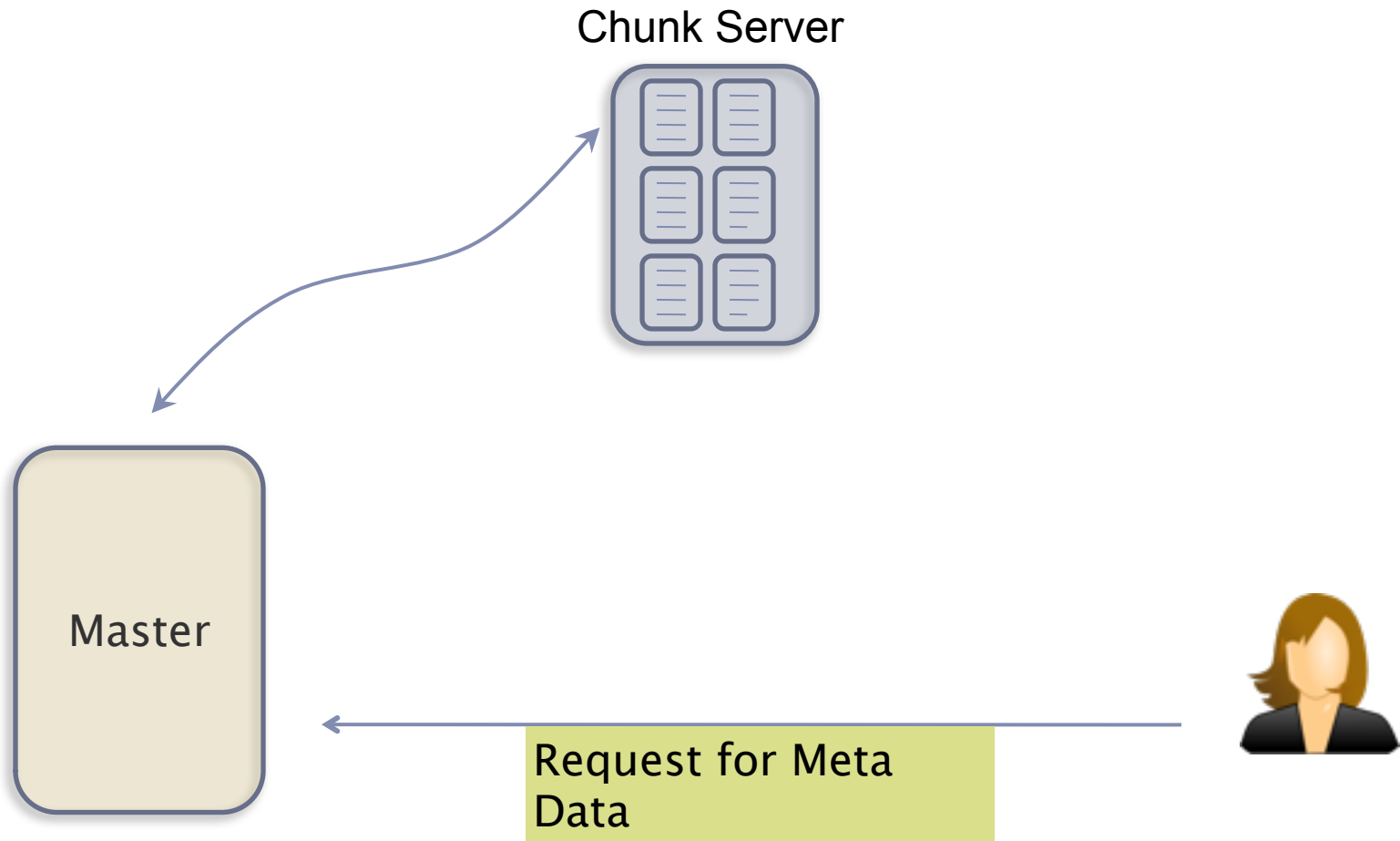
Gmail



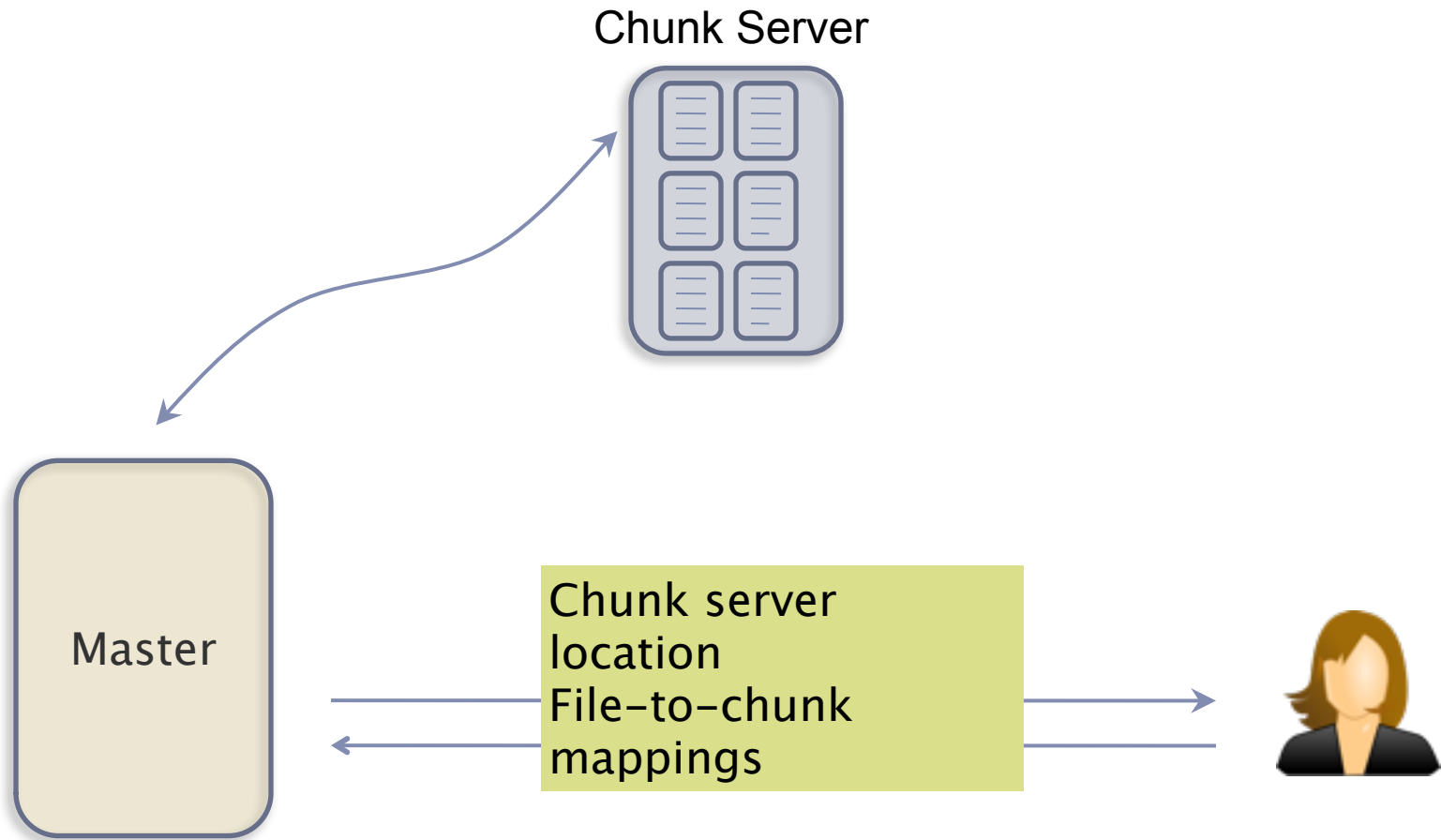
Big Picture of GFS



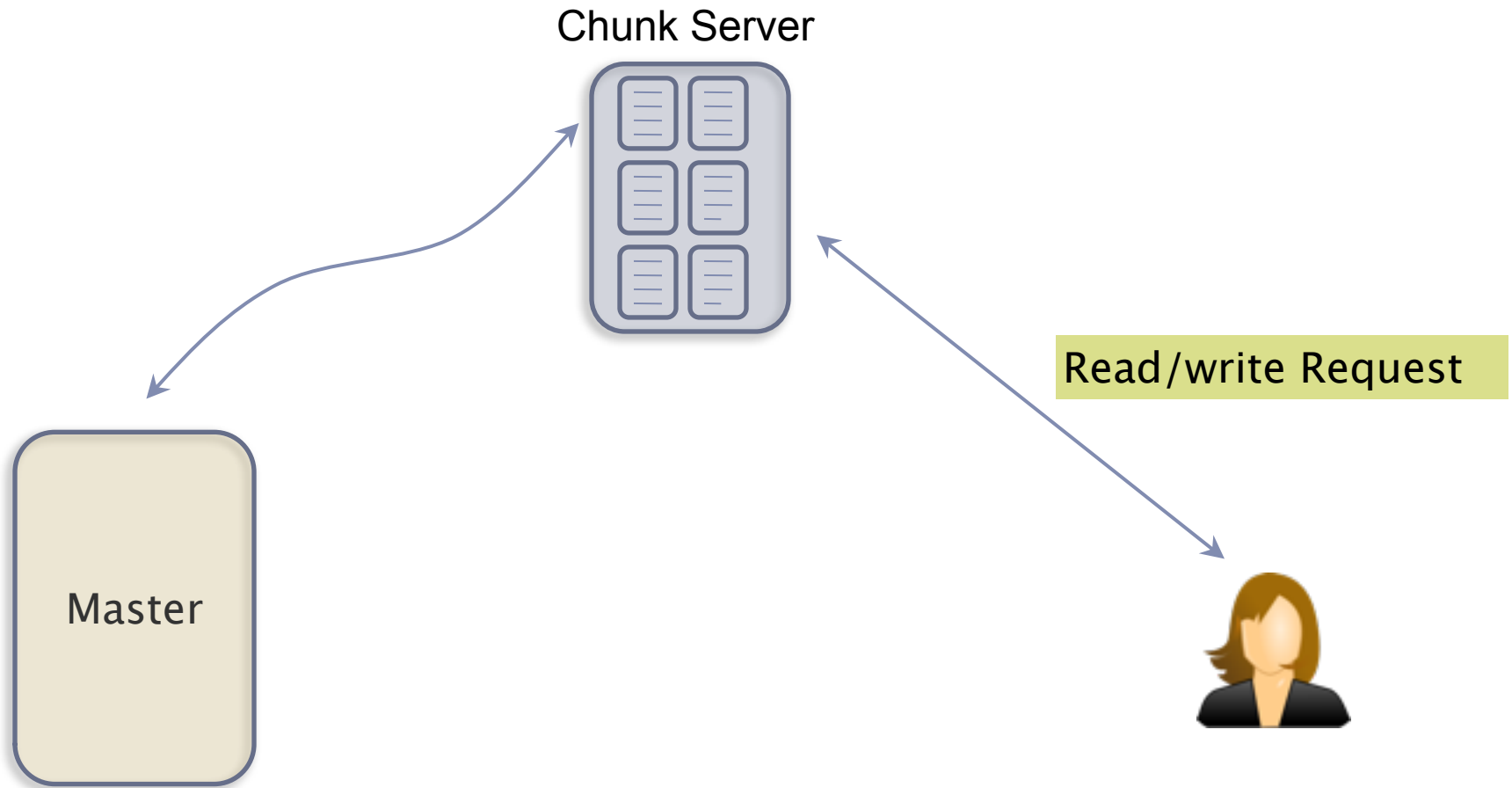
Big Picture of GFS



Big Picture of GFS



Big Picture of GFS



Big Picture of GFS

- ▶ Scalability – Very High
 - Commodity hardware
 - Minimal Master–Client communication
 - Simple

- ▶ ~One million servers²

▶ 2. Retrieved on November 06, 2011 from: <http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/>

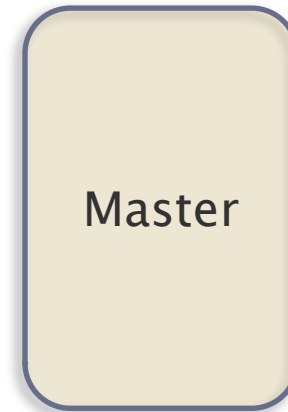
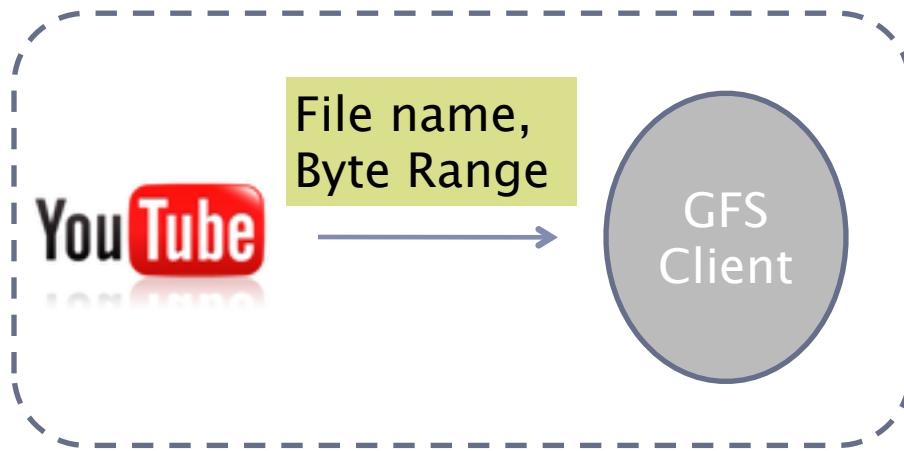
Big Picture of GFS

- ▶ Availability & Reliability – Very High
 - Replicate Data (Default 3 copies)
 - Data on Different Racks
 - Shadow Masters
 - Fast Recovery

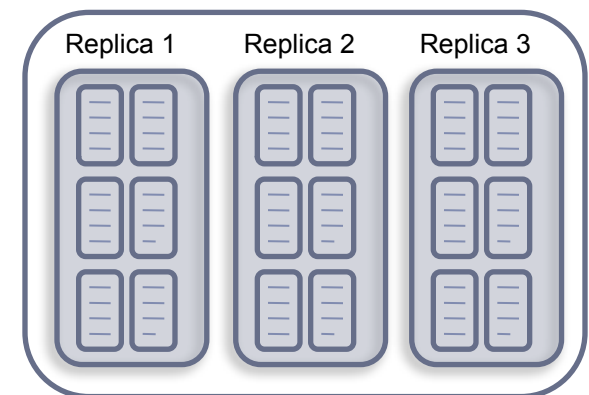
▶ 2. Retrieved on November 06, 2011 from: <http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/>

Details of GFS

- ▶ Read – Step 1
 - Issues Read Request

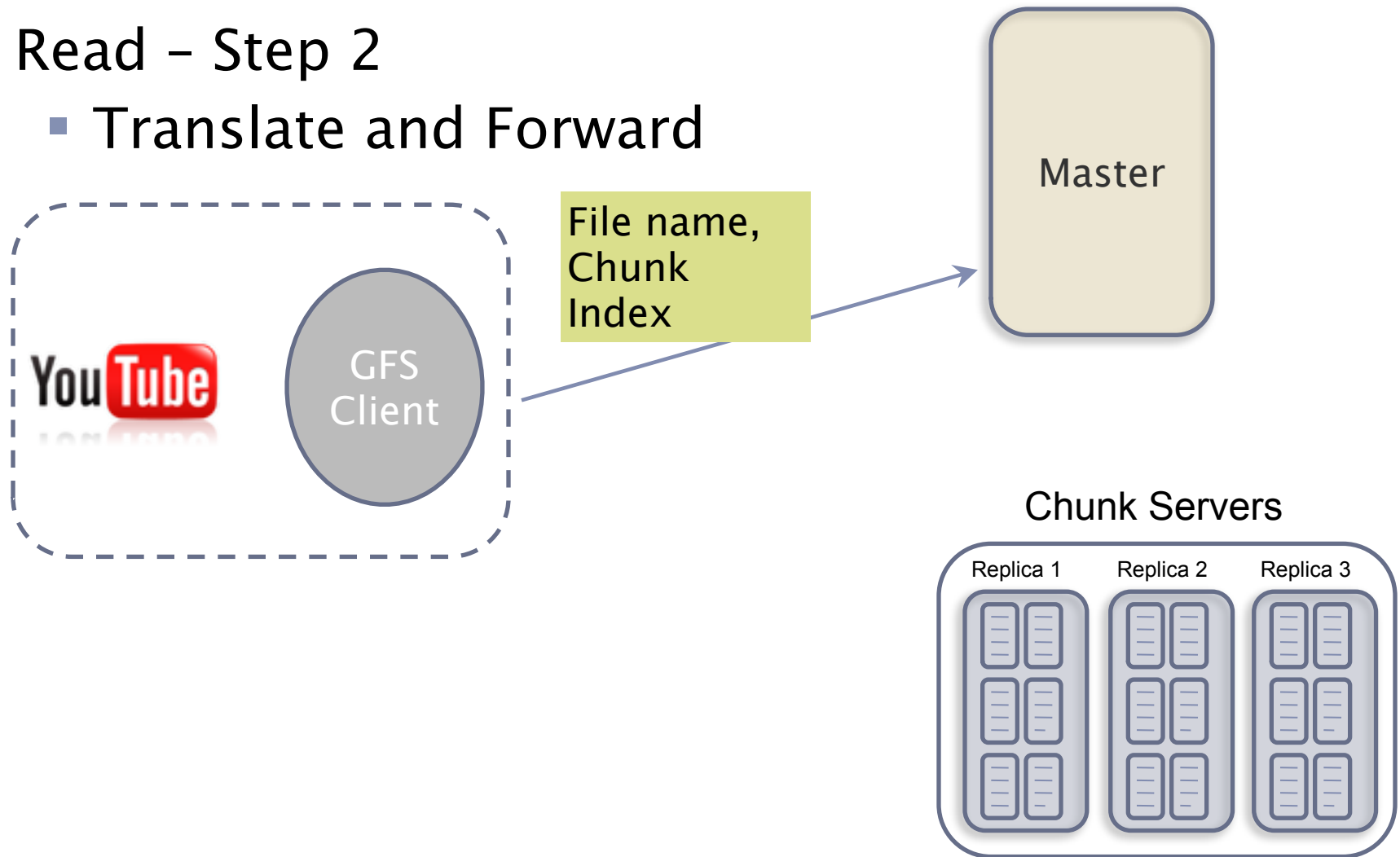


Chunk Servers



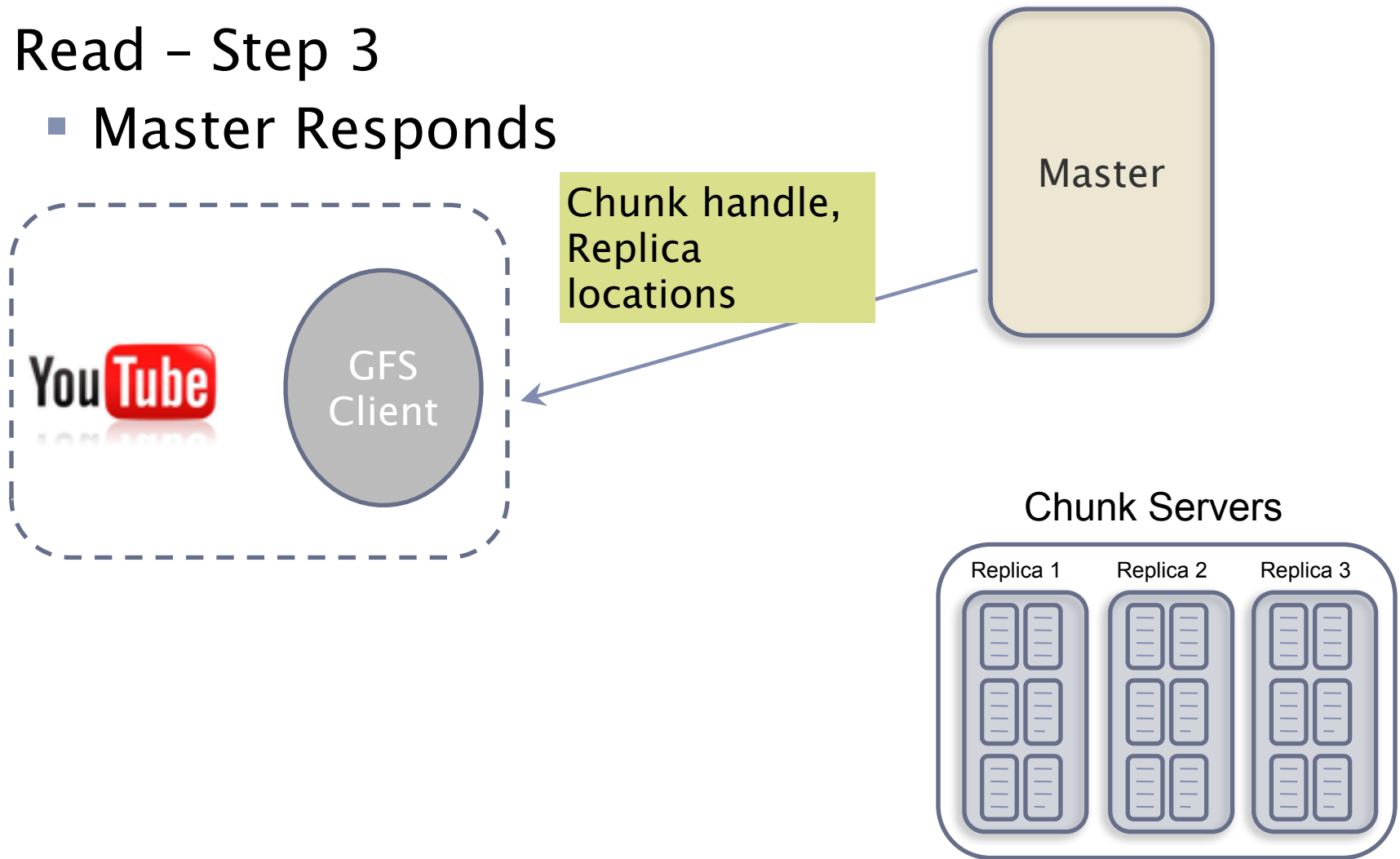
Details of GFS

- ▶ Read – Step 2
 - Translate and Forward



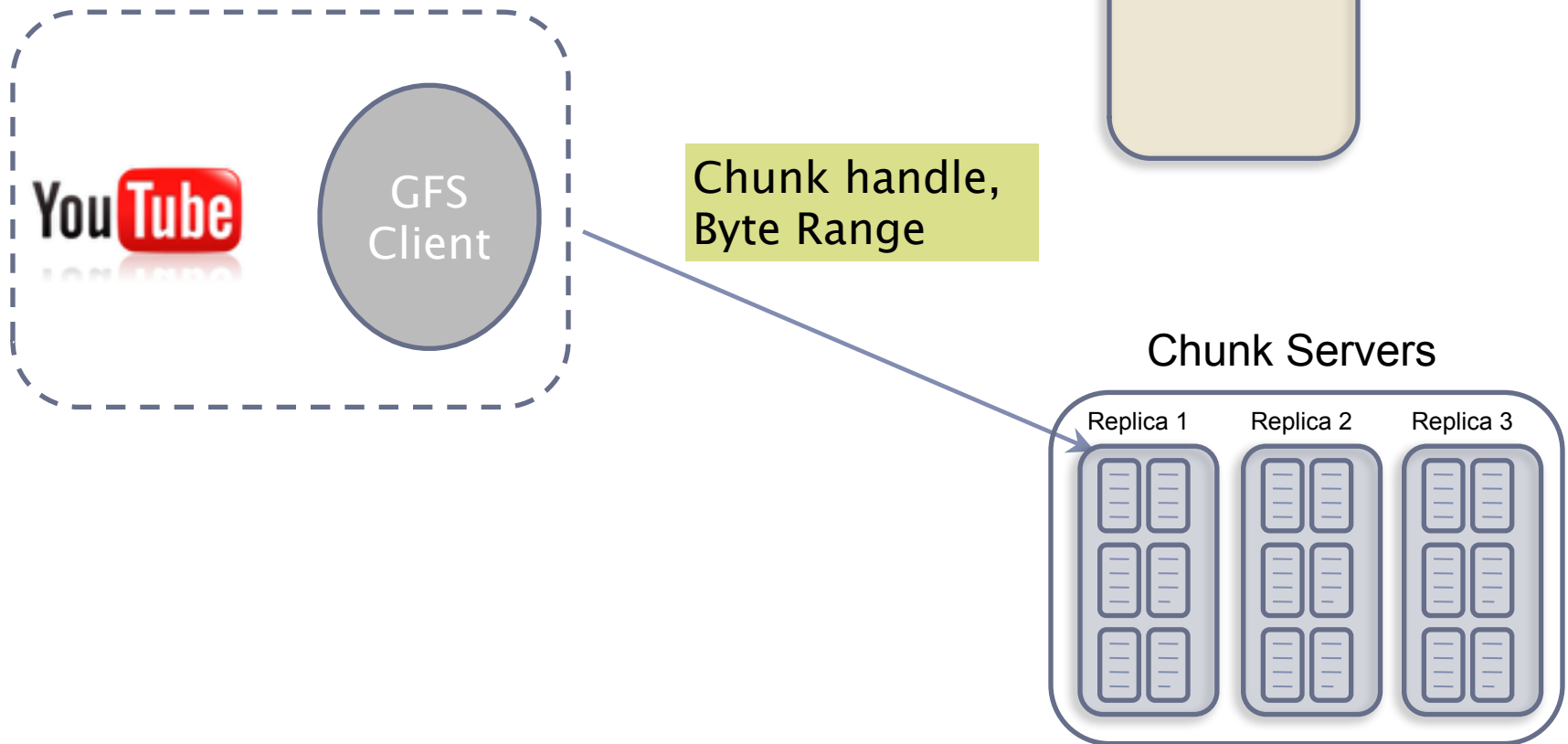
Details of GFS

- ▶ Read – Step 3
 - Master Responds



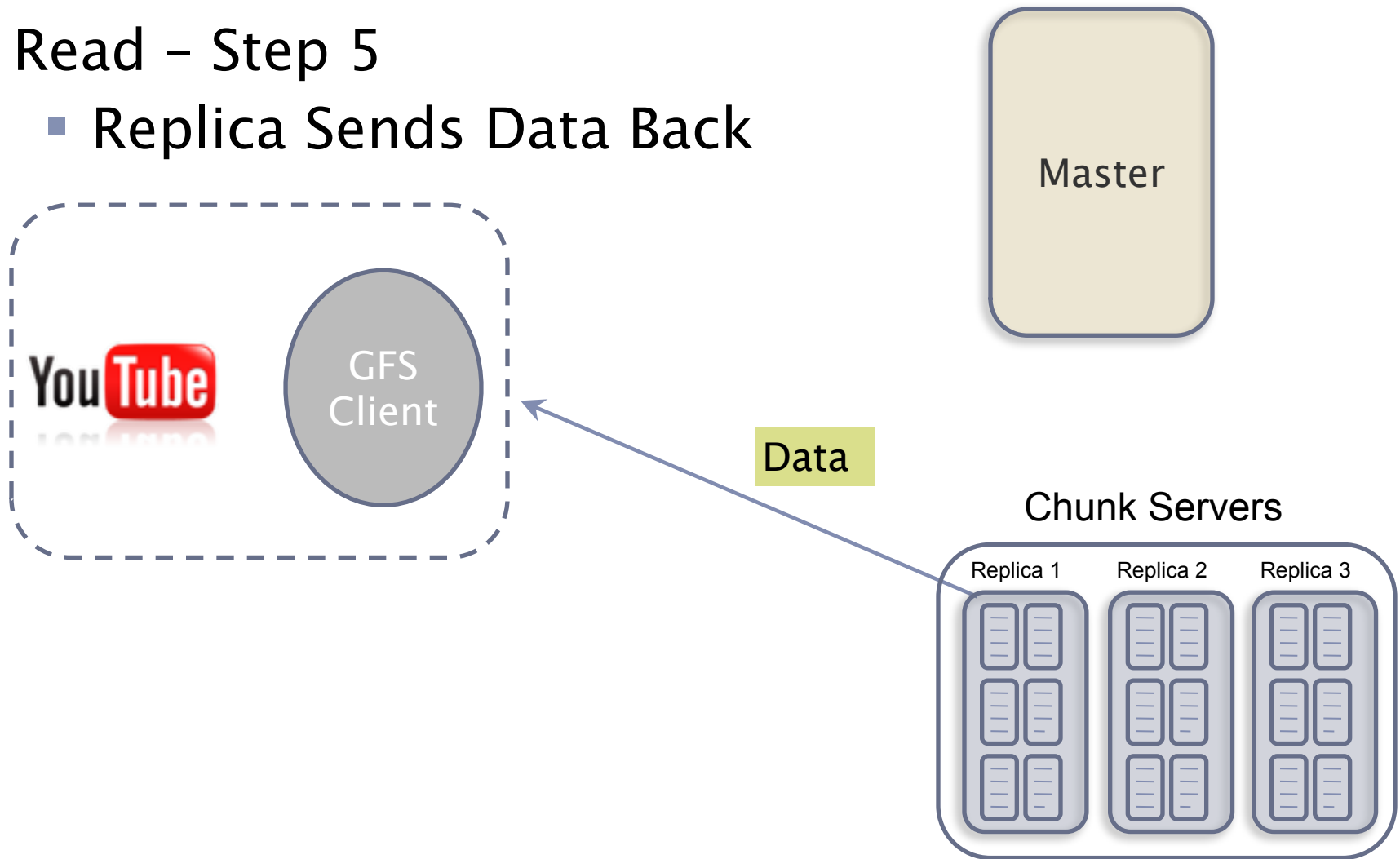
Details of GFS

- ▶ Read – Step 4
 - Picks Replica, Sends Request



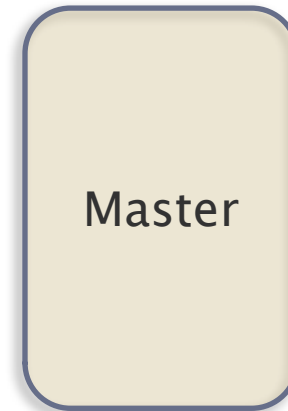
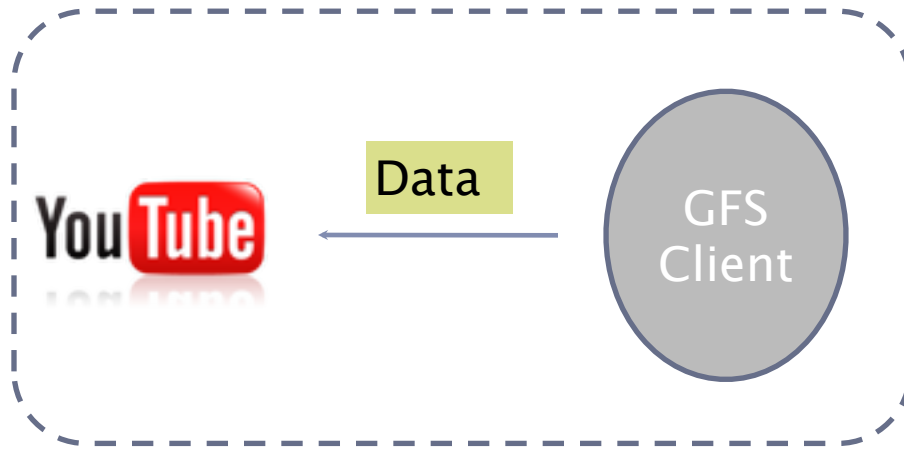
Details of GFS

- ▶ Read – Step 5
 - Replica Sends Data Back

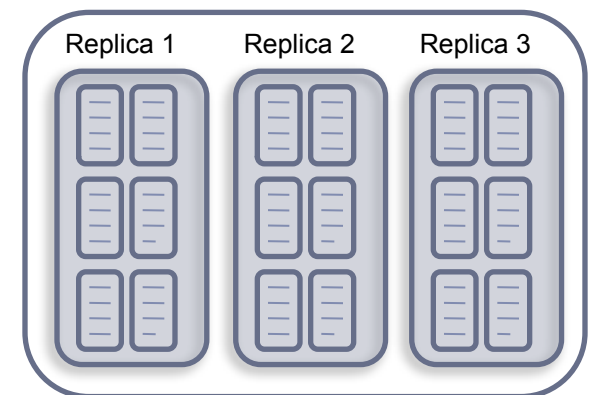


Details of GFS

- ▶ Read – Step 6
 - Client Forwards Data

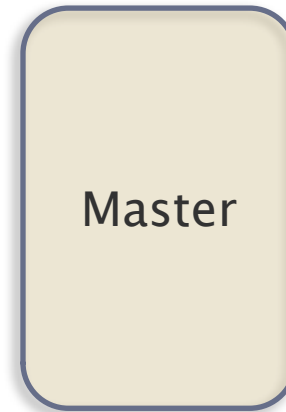
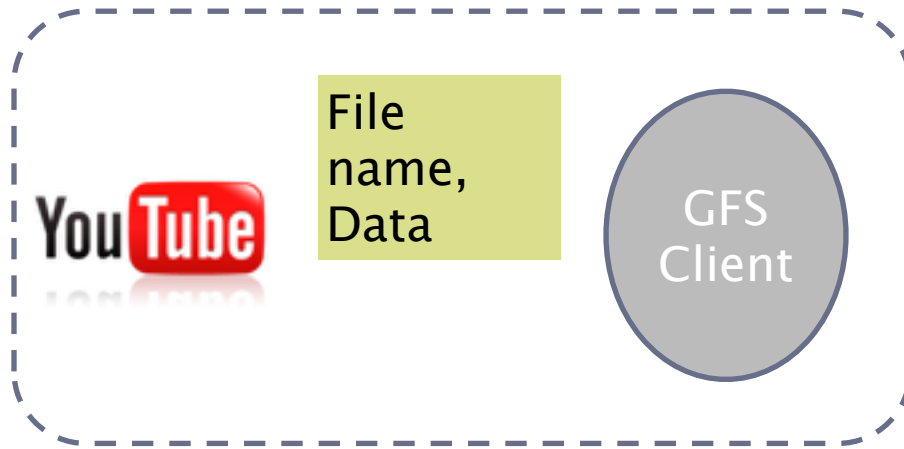


Chunk Servers

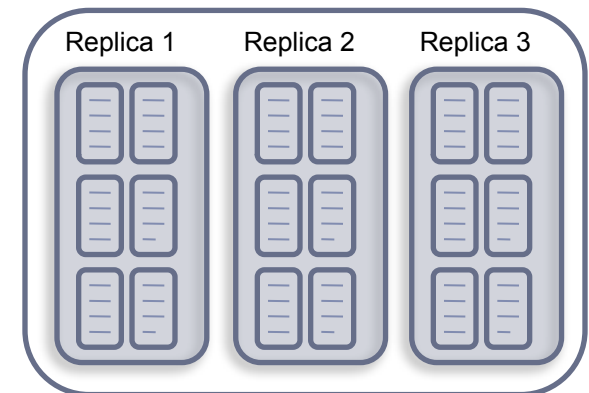


Details of GFS

- ▶ Write - Step 1
 - Application Sends Data

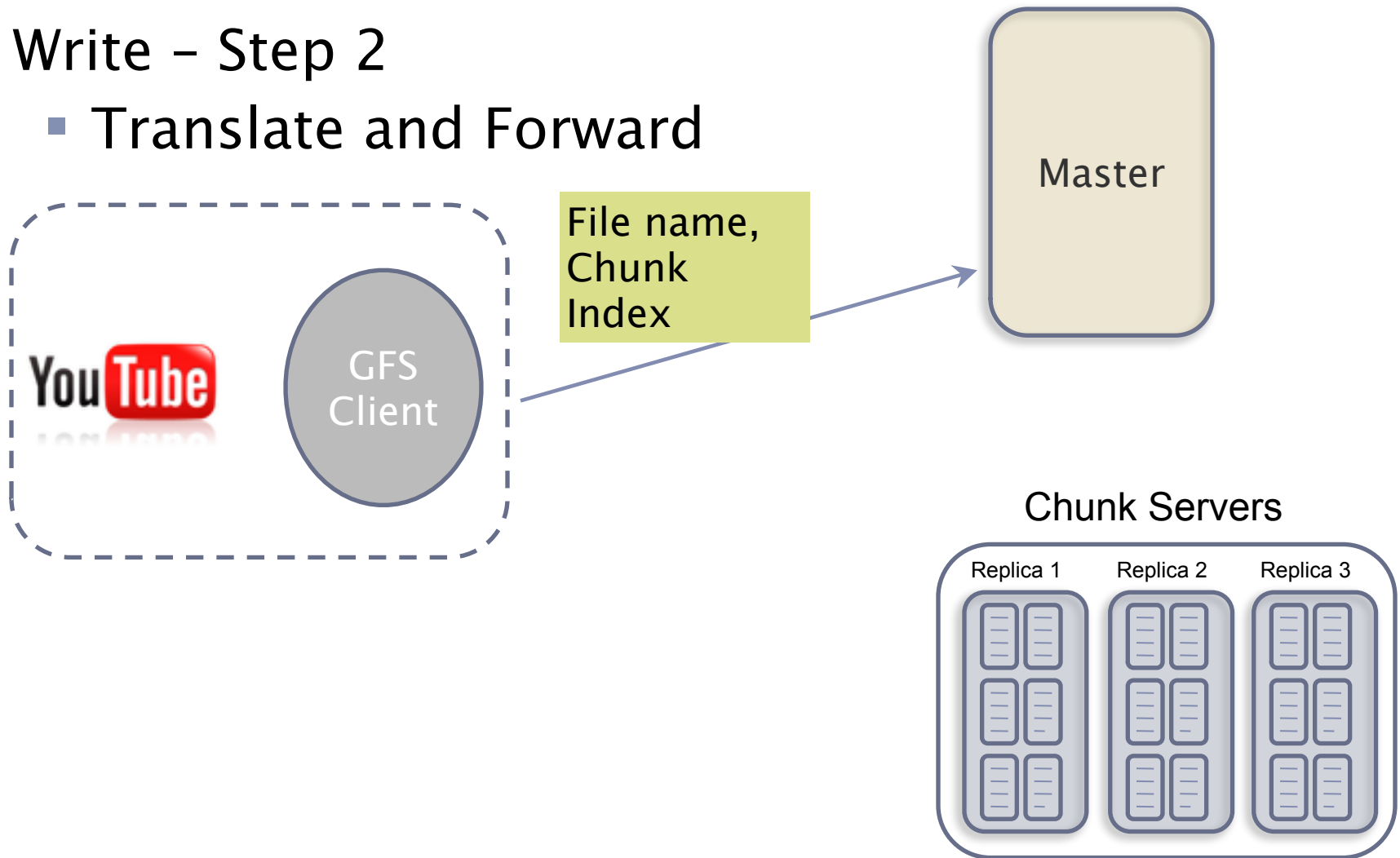


Chunk Servers



Details of GFS

- ▶ Write – Step 2
 - Translate and Forward

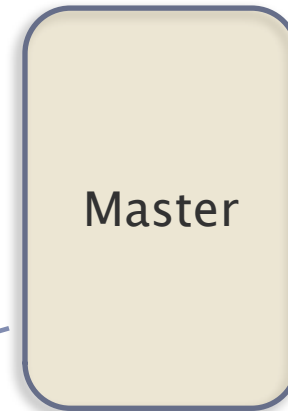


Details of GFS

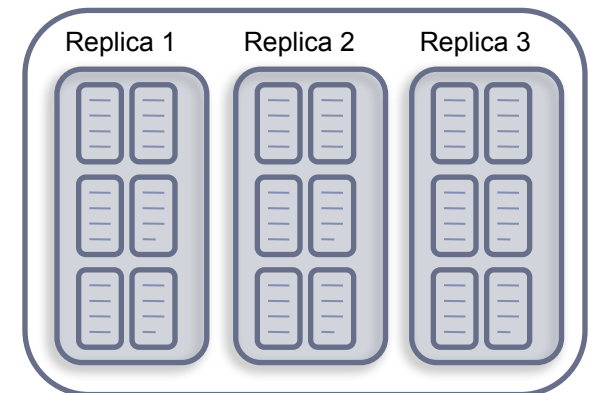
- ▶ Write – Step 3

- Master Responds

Chunk handle,
Primary + Secondary
Replica locations

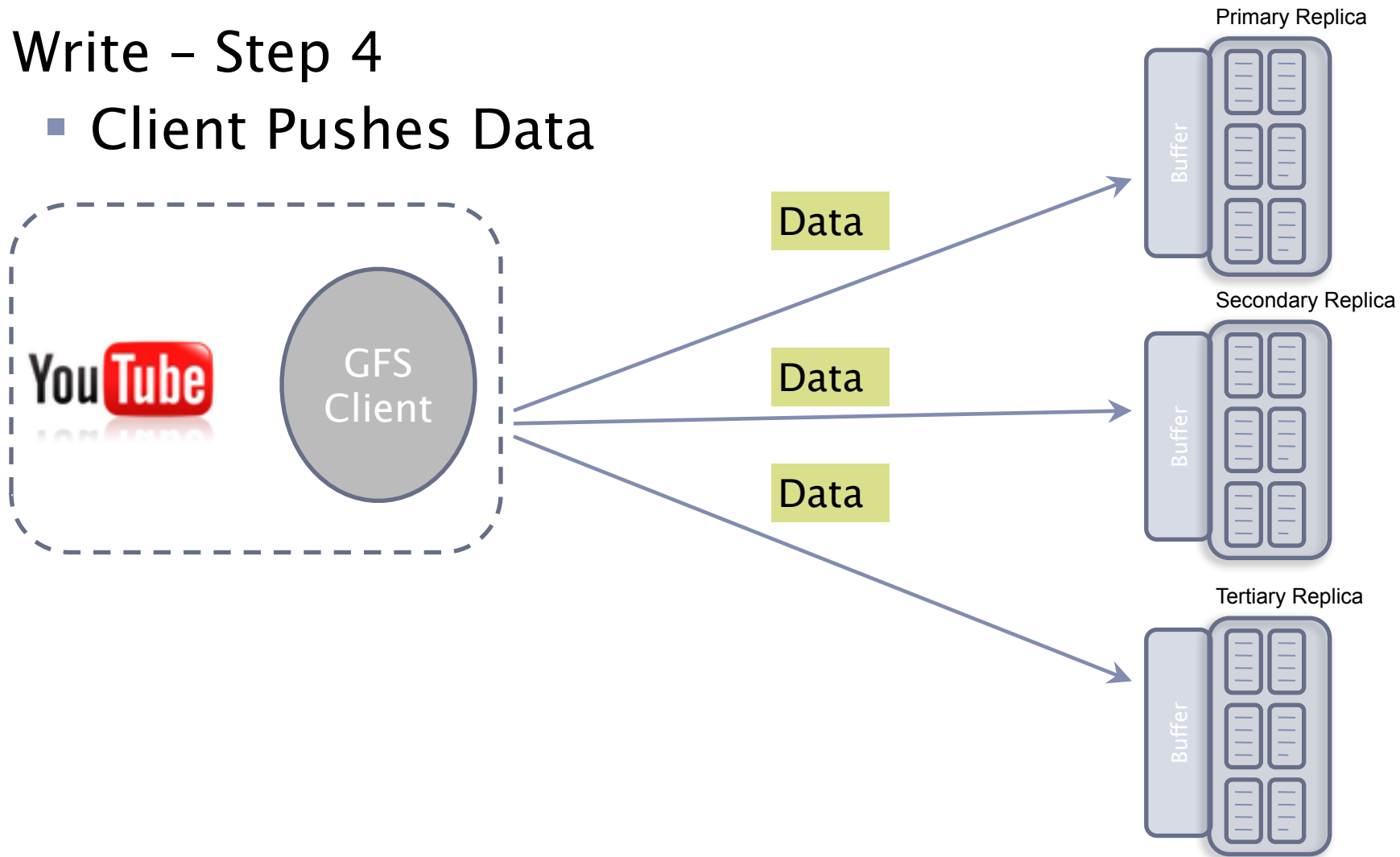


Chunk Servers



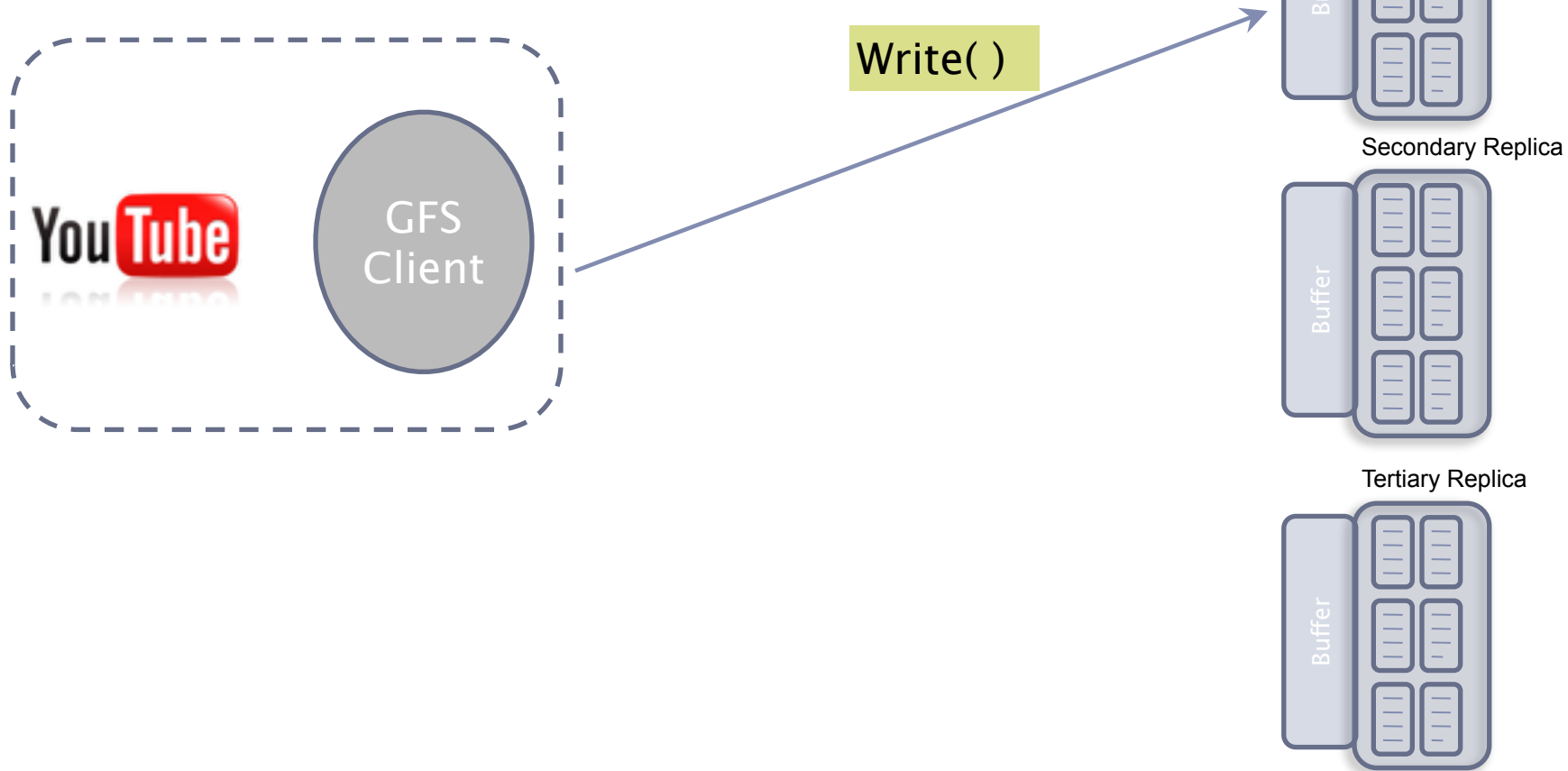
Details of GFS

- ▶ Write - Step 4
 - Client Pushes Data



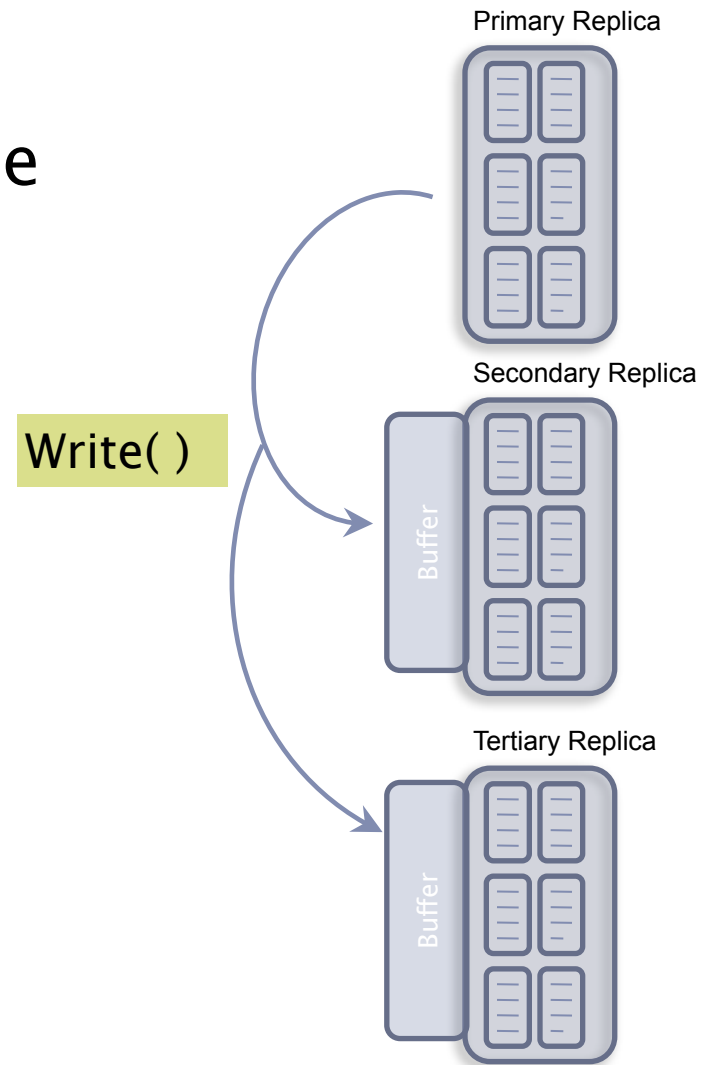
Details of GFS

- ▶ Write - Step 5
 - Client Tells Primary to Write



Details of GFS

- ▶ Write - Step 6
 - Primary Tells Others to Write



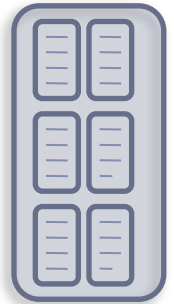
Details of GFS

- ▶ Write - Step 7
 - Primary Notifies Client

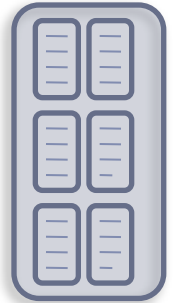


FailOrSuccess()

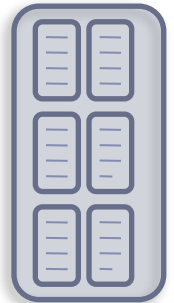
Primary Replica



Secondary Replica



Tertiary Replica



Details of GFS

- ▶ Append more common
- ▶ Difference between **Append** and **Write**
 - At Step 6
 - Primary checks space remaining
 - Enough space → same as regular write
 - Not enough space → pad the space,
 - Tells client to retry on next chunk
- ▶ Atomic, GFS chooses offset



Details of GFS

- ▶ Atomic mutations (Write or Append)
- ▶ Master's operation log keeps global total order
 - Concurrency
- ▶ Locking
 - Namespace = mapping full pathnames to metadata
 - E.g., Snapshot-ing '/home/user' to '/save/user'
 - Avoid creating '/home/user/foo'



Take-away Points of GFS

- ▶ Cheap, scalable, highly reliable and available
- ▶ Single master bottleneck
 - Google has developed a distributed master system³
 - Scales to hundreds of masters
- ▶ Used by Google!
- ▶ No full disclosure of the technology

▶ 3. Retrieved on November 06, 2011 from: <http://storagemojo.com/2009/08/17/google->

NFS vs. AFS vs. GPFS vs. GFS

	State	Name Resolution	Caching	Consistency
NFS	UDP → TCP/IP	Local directory	YES	Write-through caching
AFS	UDP, TCP	Global Name	YES	Strong Consistency
GPFS	SAN	Local directory	YES	Strong Consistency
GFS	TCP/IP	Full path name	None	Relaxed consistency model



NFS vs. AFS vs. GPFS vs. GFS – Cont.

	Concurrency	Reliability	Scalability
NFS	None?	Normal	Weak
AFS	Write-on-close	Normal	Normal
GPFS	Distributed Lock	Logging	Very High
GFS	Single master, Lease mechanism, Serialization	Replication	Very High



References

- ▶ Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. SIGOPS Oper. Syst. Rev. 37, 5 (October 2003), 29–43.
- ▶ John H Howard. 1988. An overview of the Andrew File System.
- ▶ Michael Leon Kazar. Synchronization and Caching issues with Andrew File System.
- ▶ Frank Schmuck and Roger Haskin. 2002. GPFS: A Shared-Disk File System for Large Computing Clusters
- ▶ The Google File System presentation slides. Presented at SOSP '03. URL: <http://os.inf.tu-dresden.de/Studium/DOS/SS2011/04-GFS-2.pdf>

THANK YOU
and
QUESTIONS?

