# CS 744: GOOGLE FILE SYSTEM

Shivaram Venkataraman

Fall 2021

# ANNOUNCEMENTS

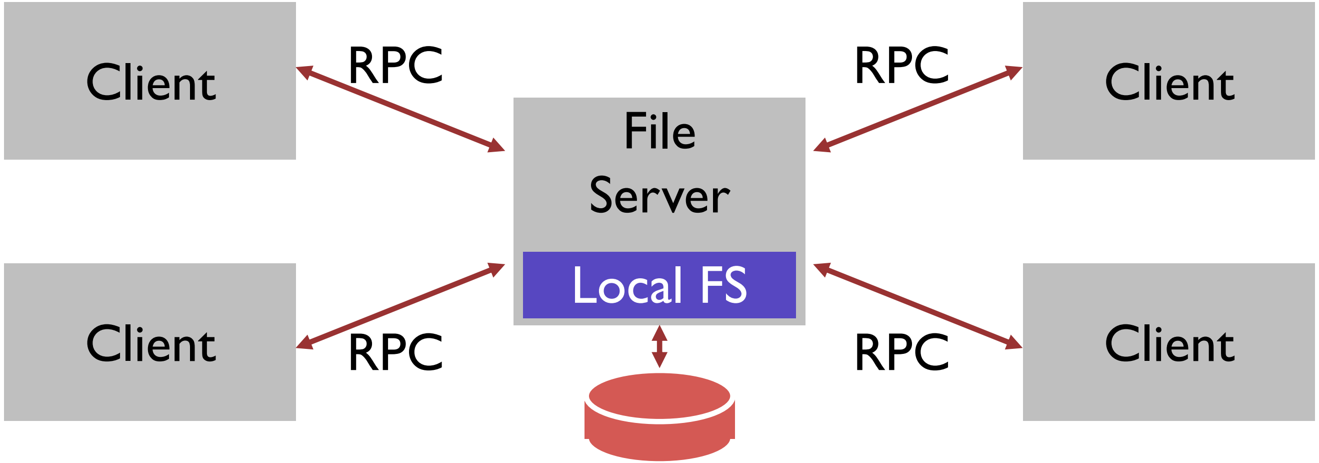
- Assignment 1 out later today
- Group submission form
- Anybody on the waitlist?

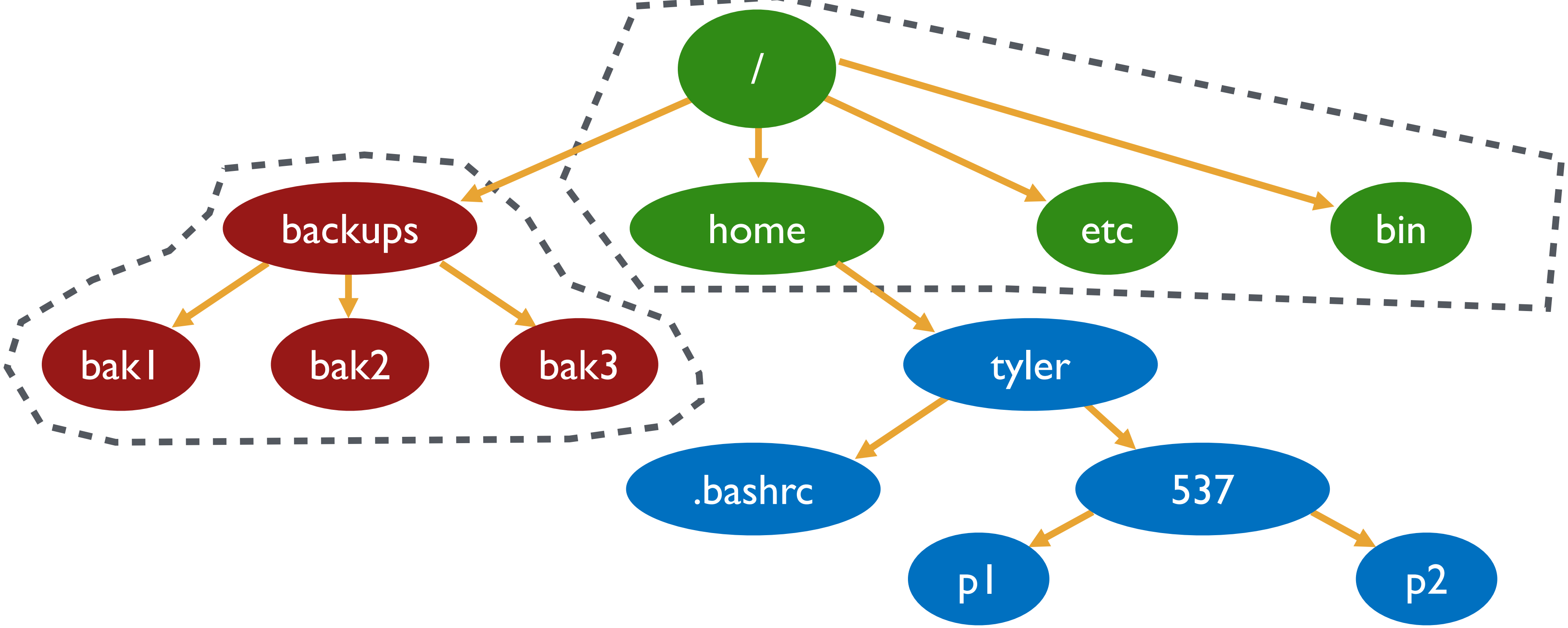
# OUTLINE

1. Brief history
2. GFS
3. Discussion
4. What happened next?

# HISTORY OF DISTRIBUTED FILE SYSTEMS

# SUN NFS


Client
RPC
RPC
Client
File Server
Local FS
Client
RPC
RPC
Client

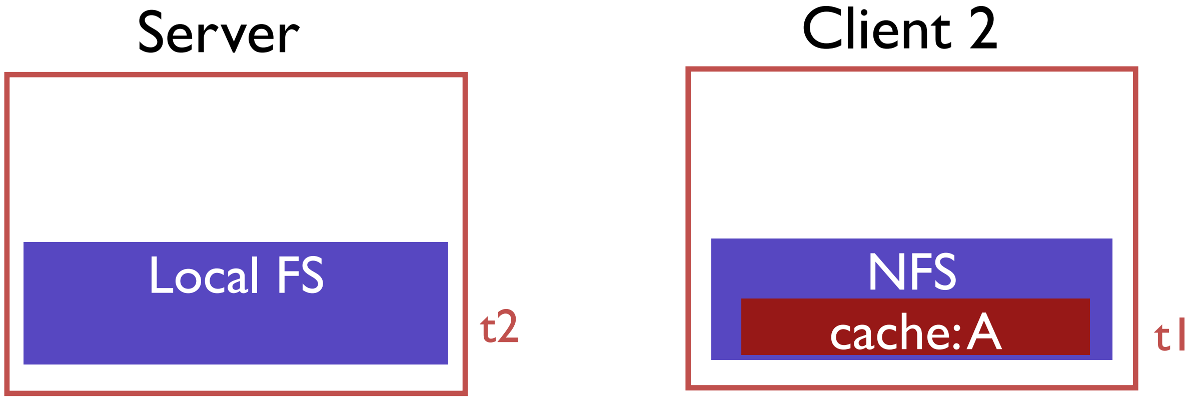


/dev/sda1 **on** /

/dev/sdb1 **on** /backups

NFS **on** /home

# CACHING



Client cache records time when data block was fetched (t1)

Before using data block, client does a STAT request to server

- get's last modified timestamp for this file (t2) (not block…)
- compare to cache timestamp
- refetch data block if changed since timestamp (t2 > t1)

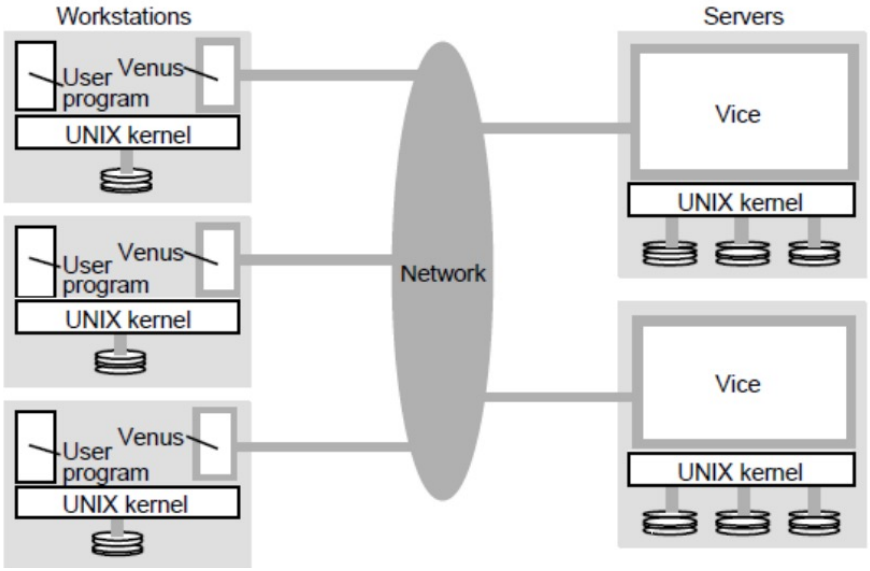
# ANDREW FILE SYSTEM

- Design for scale

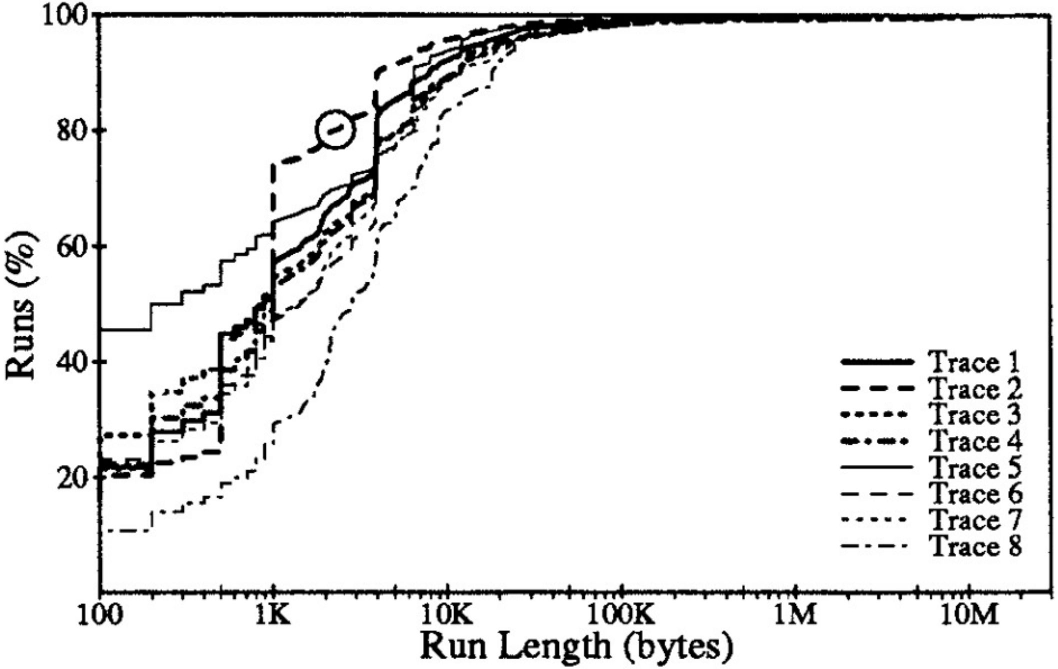
- Whole-file caching

- Callbacks from server

Architecture



# WORKLOAD PATTERNS (1991)



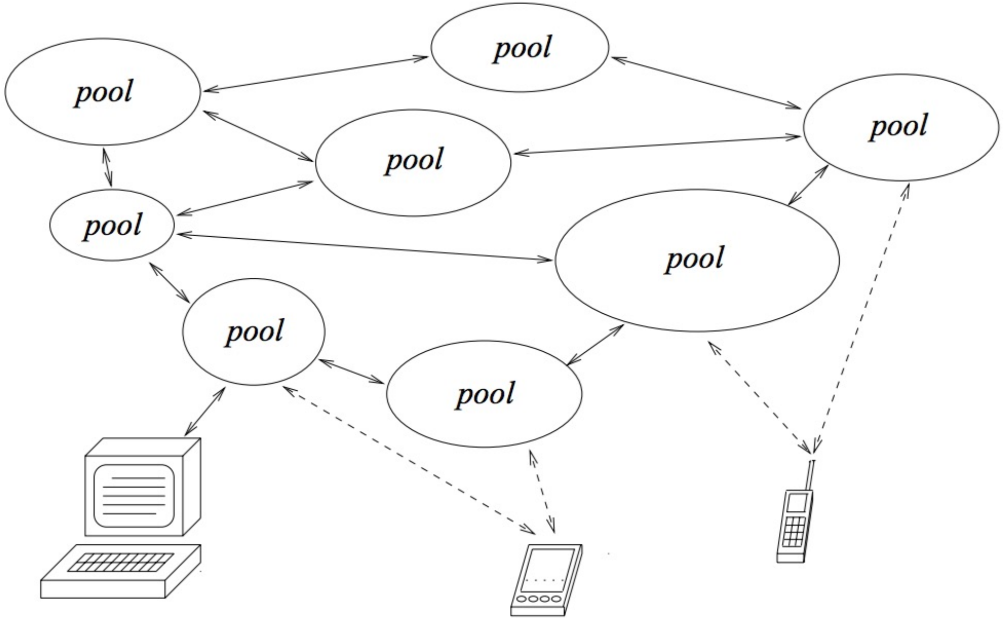
Mary G. Baker, John H. Hartman, Michael D. Kupfer, Ken W. Shirriff, and John K. Ousterhout

# OCEANSTORE/PAST

Wide area storage systems

Fully decentralized

Built on distributed hash tables (DHT)



# GFS: WHY ?

Components with failures

Files are huge !

# GFS: WHY ?

Applications are different

# GFS: WORKLOAD ASSUMPTIONS

“Modest” number of large files

Two kinds of reads: Large Streaming and small random

Writes: Many large, sequential writes. Few random

High bandwidth more important than low latency

# GFS: DESIGN

- Single Master for metadata

- Chunkservers for storing data

- No POSIX API !
- No Caches!

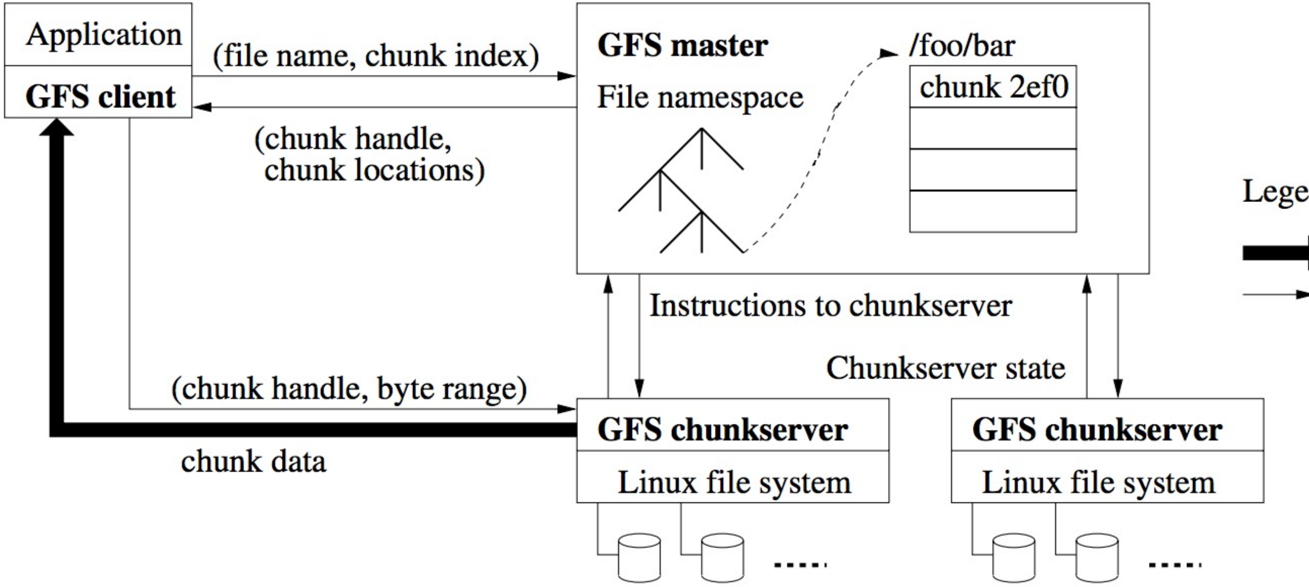


**Figure 1: GFS Architecture**

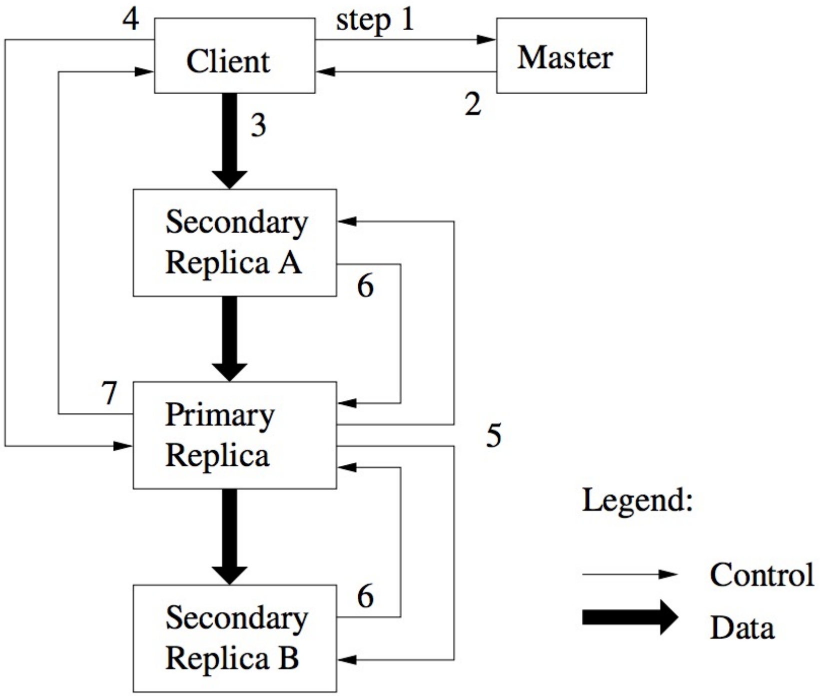
# CHUNK SIZE TRADE-OFFS

Client → Master

Client → Chunkserver

Metadata

# GFS: REPLICATION



- 3-way replication to handle faults
- Primary replica for each chunk
- Chain replication (consistency)

- Decouple data, control flow
- Dataflow: Pipelining, network-aware

# RECORD APPENDS

| | |
|---|---|
| Write | Client specifies the offset |
| Record Append | GFS chooses offset |

Consistency

- At-least once
- Atomic

# MASTER OPERATIONS

- No "directory" inode! Simplifies locking
- Replica placement considerations

- Implementing deletes

# FAULT TOLERANCE

- Chunk replication with 3 replicas
- Master
    - Replication of log, checkpoint
    - Shadow master
- Data integrity using checksum blocks

# DISCUSSION

https://forms.gle/YpDcxPncdqnZ7JXG6

# GFS SOCIAL NETWORK

You are building a new social networking application. The operations you will need to perform are

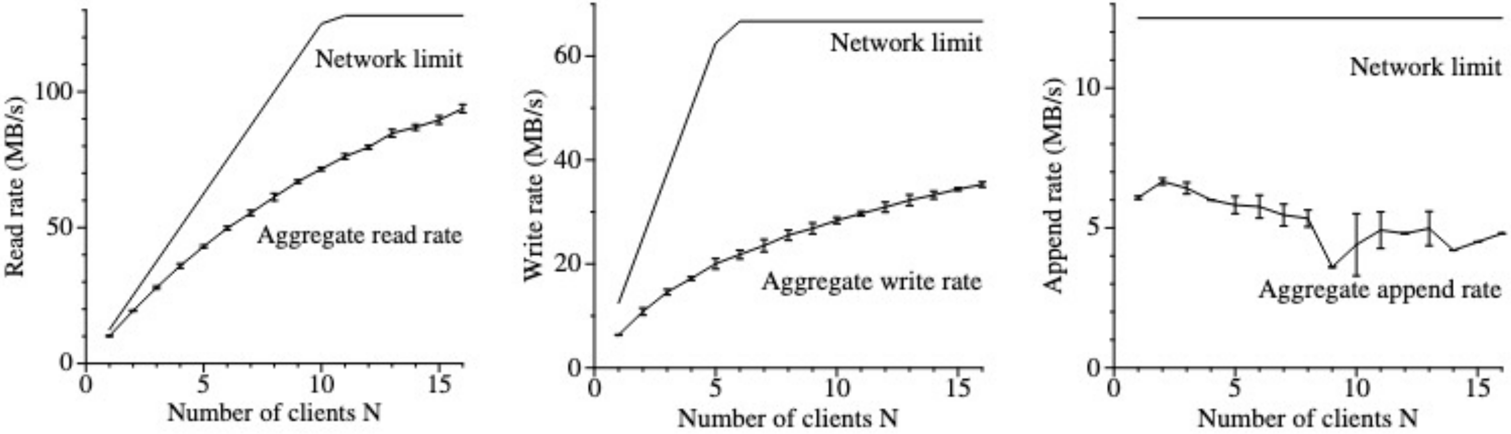
(a) add a new friend id for a given user

(b) generate a histogram of number of friends per user.

How will you do this using GFS as your storage system ?

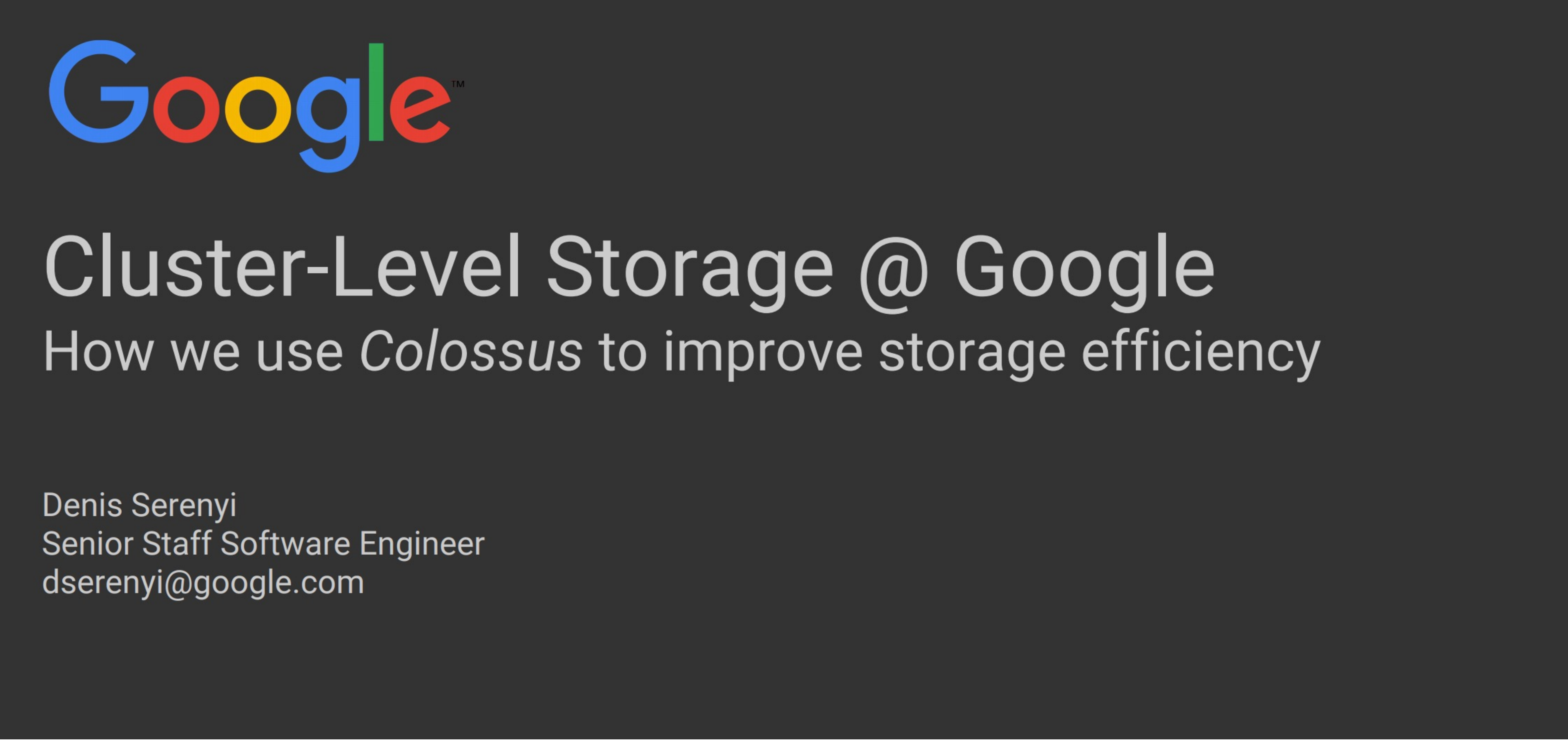
# GFS EVAL

List your takeaways from “Table 3: Performance metrics”



(a) Reads (b) Writes (c) Record appends

# WHAT HAPPENED NEXT



Google™

# Cluster-Level Storage @ Google

## How we use *Colossus* to improve storage efficiency

Denis Serenyi
Senior Staff Software Engineer
dserenyi@google.com

Keynote at PDSW-DISCS 2017: 2nd Joint International Workshop On Parallel Data Storage & Data Intensive Scalable Computing Systems

# GFS EVOLUTION

Motivation:

- GFS Master
  - One machine not large enough for large FS
  - Single bottleneck for metadata operations (data path offloaded)
  - Fault tolerant, but not HA

- Lack of predictable performance
  - No guarantees of latency
  - (GFS problems: one slow chunkserver -> slow writes)

# GFS EVOLUTION

GFS master replaced by Colossus

Metadata stored in BigTable

Recursive structure ? If Metadata is ~1/10000 the size of data

100 PB data → 10 TB metadata

10TB metadata → 1GB metametadata

1GB metametadata → 100KB meta...

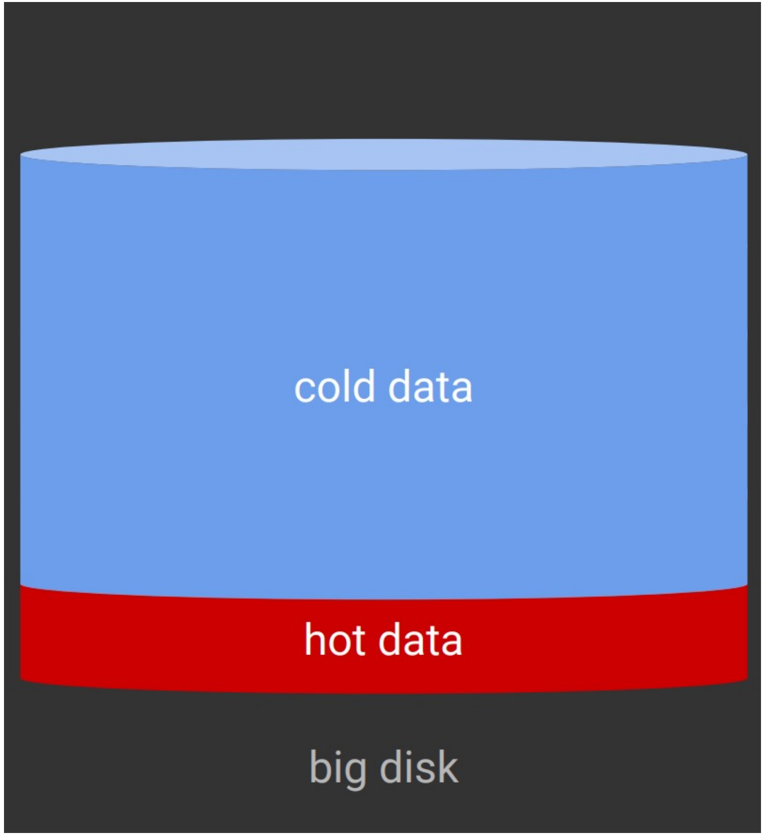
# GFS EVOLUTION

Need for Efficient Storage

Rebalance old, cold data

Distributes newly written data evenly across disk

Manage both SSD and hard disks



# HETEROGENEOUS STORAGE

amazon web services™ | S3

F4: Facebook

Blob stores

DynamoDB

redis

Key Value Stores

# NEXT STEPS

- Assignment 1 out tonight!
- Next up: MapReduce, Spark