

CS 744: GOOGLE FILE SYSTEM

Shivaram Venkataraman

Spring 2025

ANNOUNCEMENTS

- Assignment I out today
- Group submission form

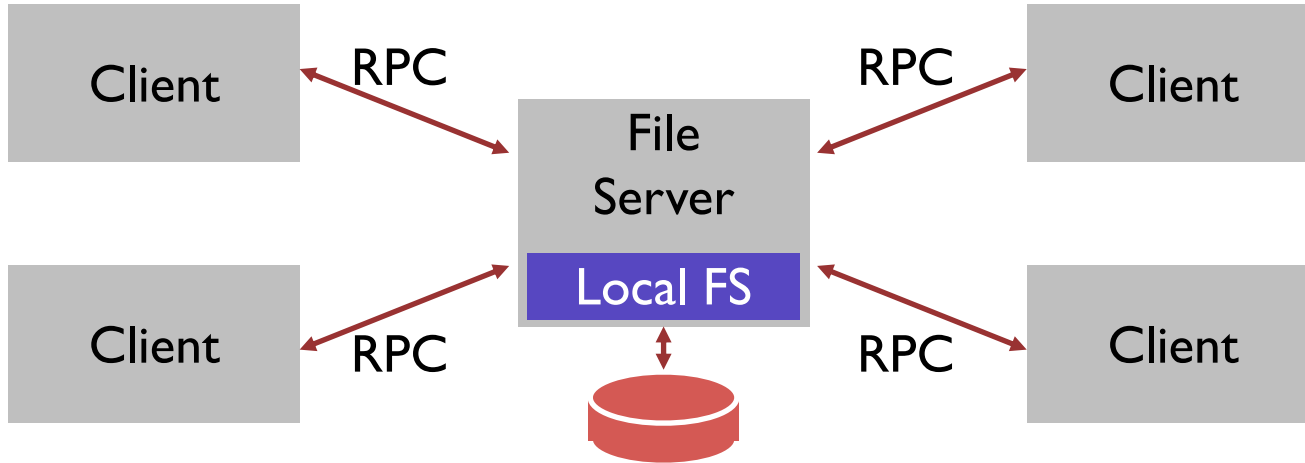
- Office hours
 - Shivaram Venkataraman: Tue 3pm-4pm at CS 7367
 - Tareq Mahmood: Tue 4pm-5pm, Thu 4pm-5pm at CS 3250

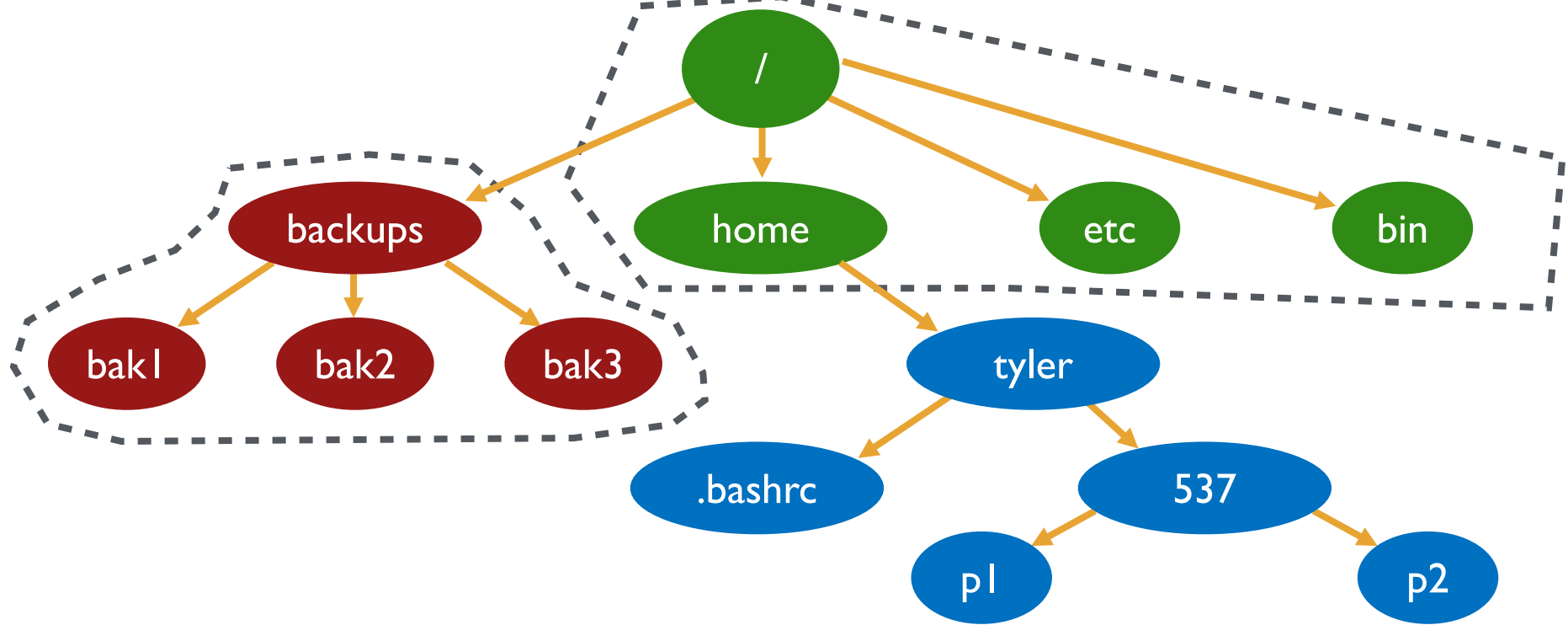
OUTLINE

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

HISTORY OF DISTRIBUTED FILE SYSTEMS

SUN NFS



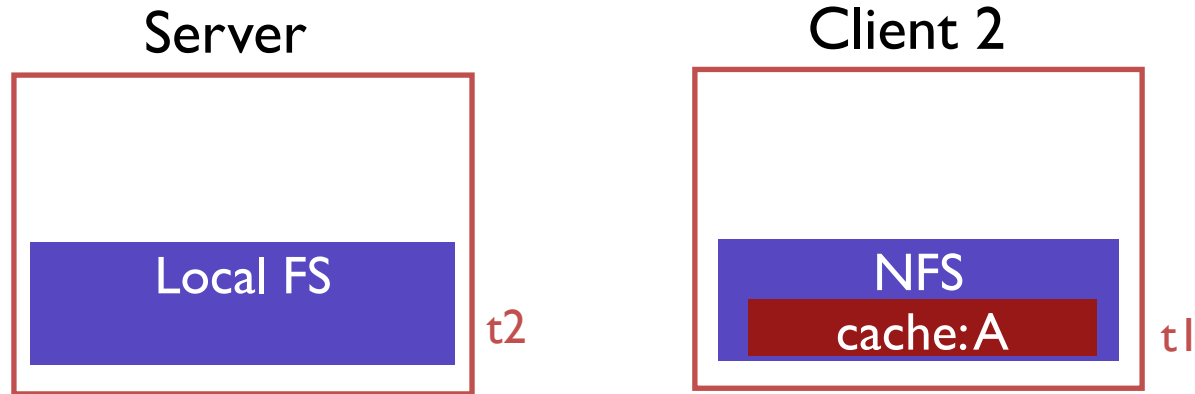


/dev/sda1 **on** /

/dev/sdb1 **on** /backups

NFS **on** /home

CACHING



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

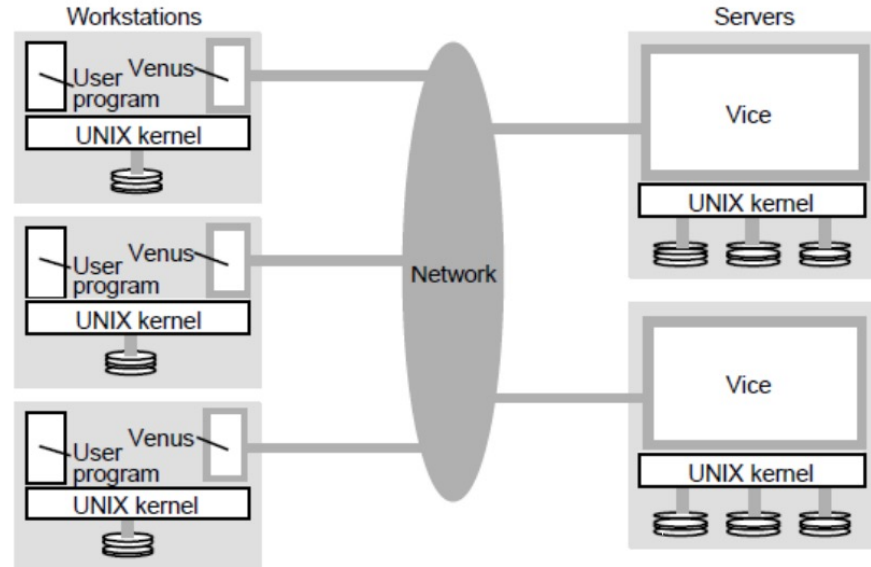
Before using data block, client does a STAT request to server

- 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

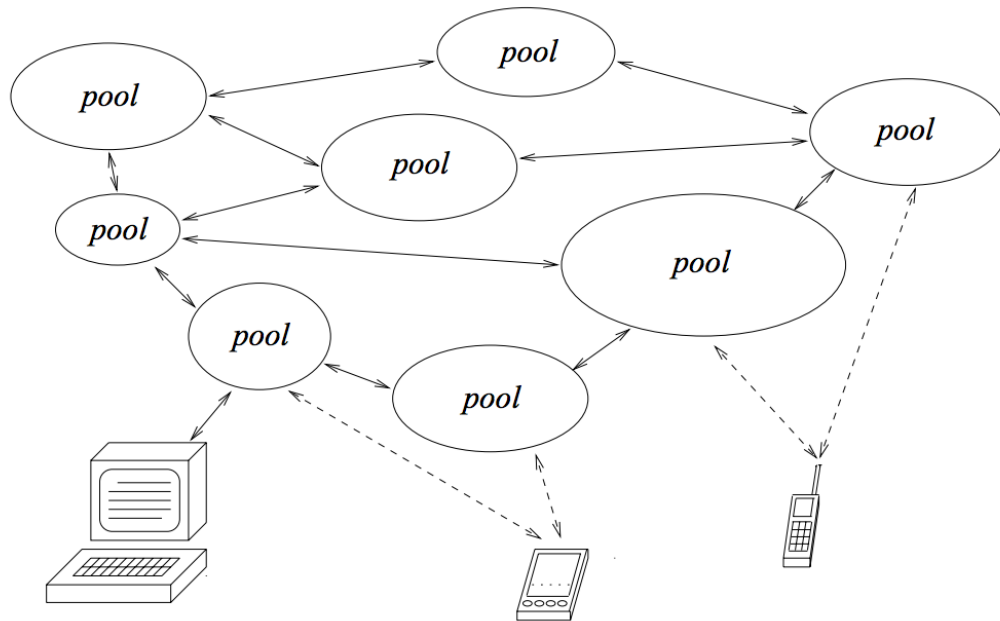


OCEANSTORE/PAST

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!

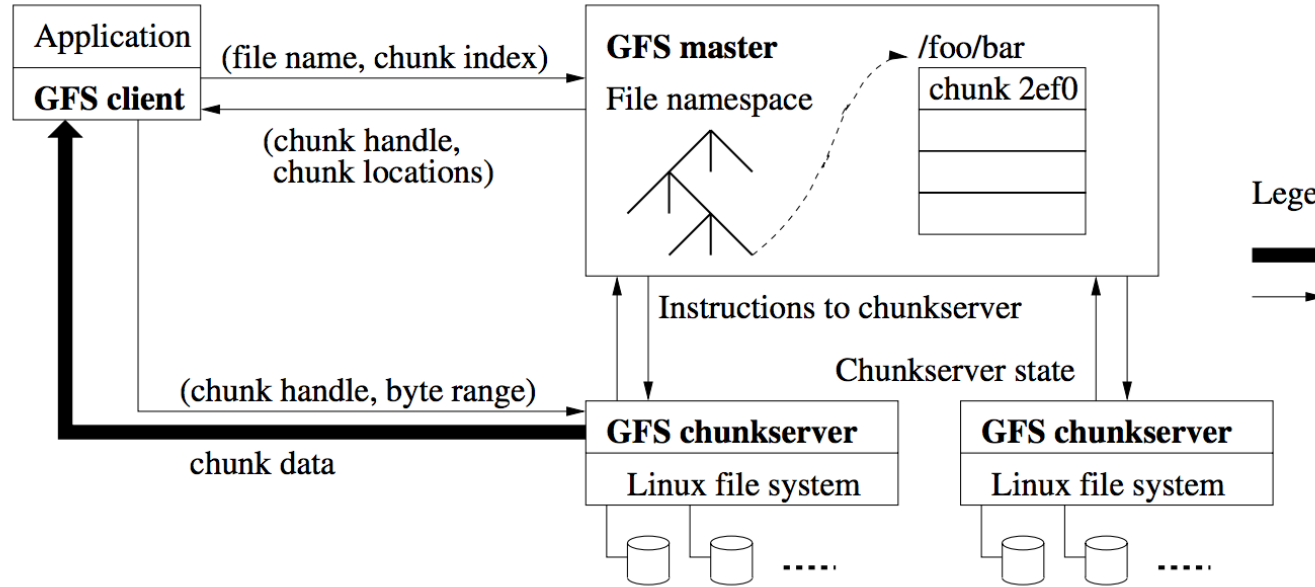


Figure 1: GFS Architecture

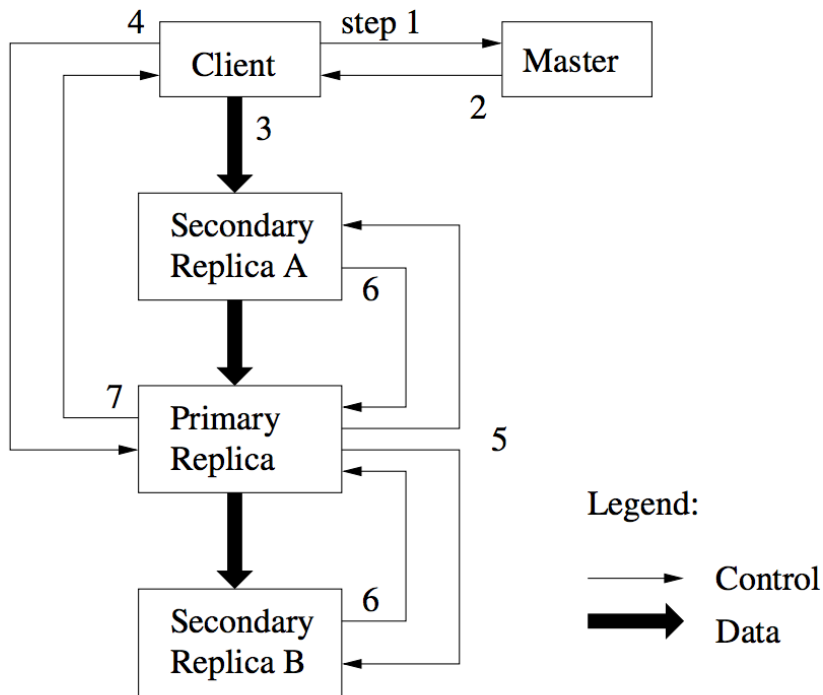
CHUNK SIZE TRADE-OFFS

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

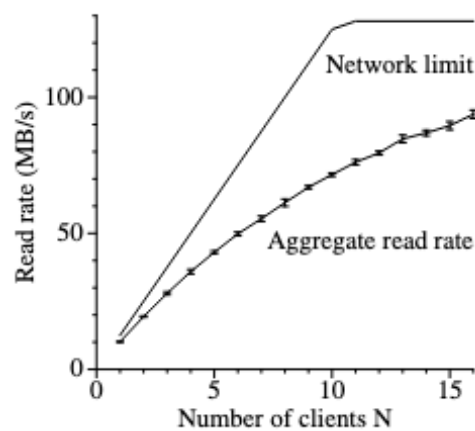
- Chunk replication with 3 replicas
- Master
 - Replication of log, checkpoint
 - Shadow master
- Data integrity using checksum blocks

DISCUSSION

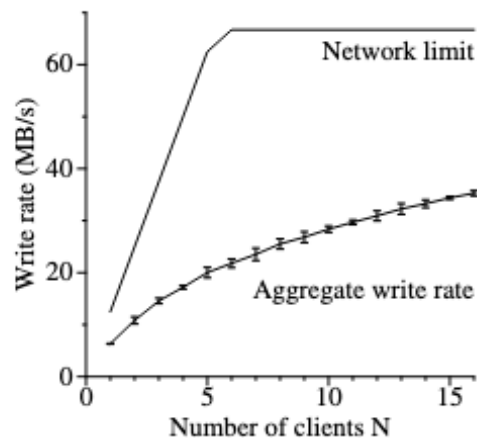


<https://forms.gle/Egik6VxfhHhBXsXM7>

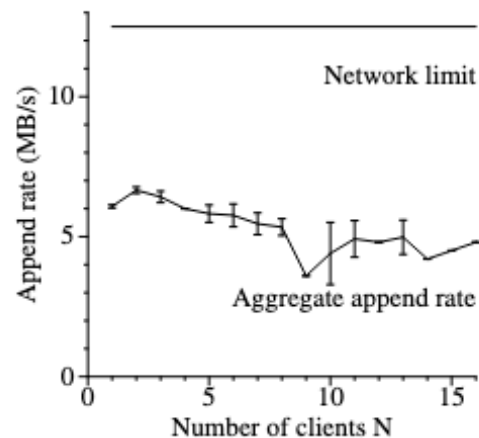
Takeaways



(a) Reads



(b) Writes



(c) Record appends

The evaluation (Table 2) shows clusters with up to 180 TB of data. What part of the design would need to change if we instead had 180 PB of data?

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 $\sim 1/10000$ the size of data

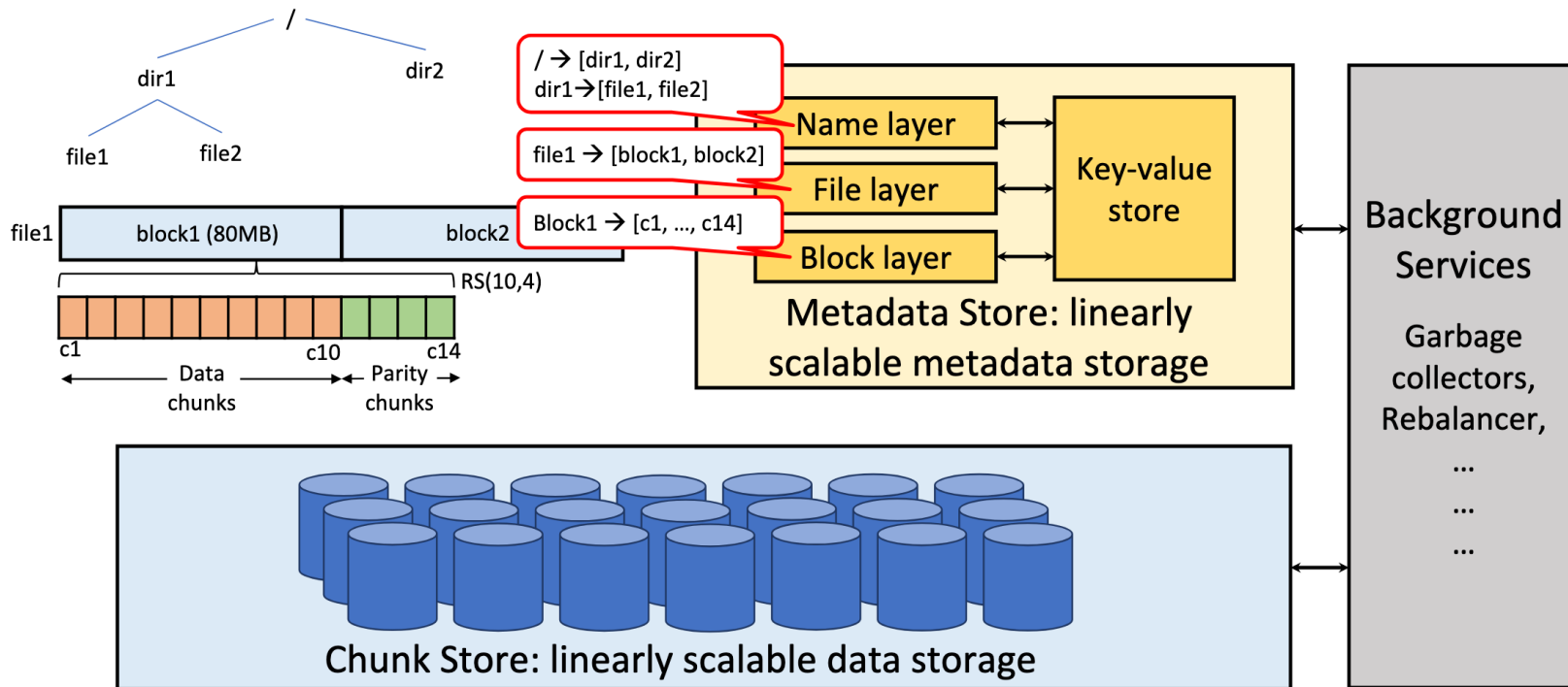
100 PB data \rightarrow 10 TB metadata

10TB metadata \rightarrow 1GB metametadata

1GB metametadata \rightarrow 100KB meta...

META: TECTONIC

Scalability: Support Exabyte Scale Clusters



NEXT STEPS

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