# Recitation 14: GFS

MIT - 6.033
Spring 2021
Henry Corrigan-Gibbs

# Plan

* Recitation Qs
* Setting for GFS
* How GFS works

## Logistics

* Midterm Th 3/31 - see site
* Technical feedback on DPR by Sat
* Volunteers for recitation Qs next week?

# Note on terminology:

GFS paper uses "master" for server that coordinates chunkservers

↳ We will use "main" instead

(Katrina prefers "coordinator")

↳ Most open-source projects & companies have deprecated use of "master" and we will follow that convention.

# Recitation Qs

1. What assumptions does GFS rely on?

- few modifications, mostly appends
- large streaming reads, small random reads
- b/w more important than latency

2. How does it exploit those assumptions?

- record append
- large chunk size

3. Why does GFS make those assumptions?

- Google workload
  - → control both ends
    - → Why systems people love the data center!

# The Setting


- Google crawls the web and needs to store the data somewhere
- Later, they need to process the data into a search index
- Today, there's also youtube, maps, analytics, ....
  - ↳ To be clear, this version of GFS probably not used anywhere today.

One of the innovations at Google was to use cheap but less reliable machines
- ↳ instead of heavier expensive ones

Why?
- ↳ price/perf ratio better for workstations
- ↳ if you have enough machines you'll have to handle failures anyways.

Fun fact: Ghemawat was MIT PhD, advised by Liskov & Kaashoek.

To understand GFS, let's try to design our own storage system.

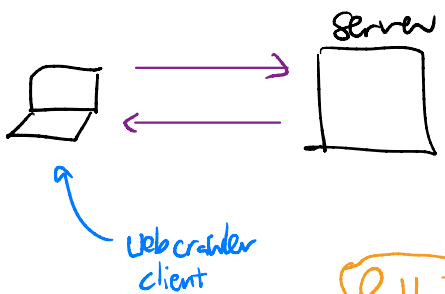
## Attempt 1. Single Server

Problem? Take vote

↳ Ask for rep

+ Simple!!

- Fault tolerance
- Capacity
- Throughput

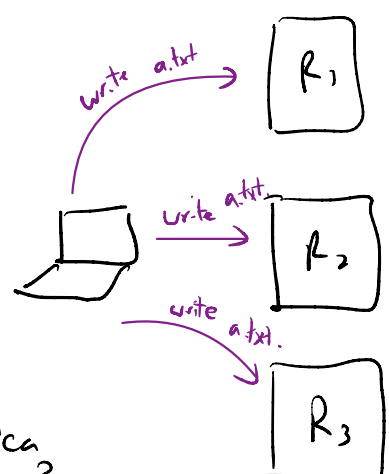



Problem? Take vote

## Attempt 2. Replicas of database

+ Better tolerance to server faults!

- Potential for inconsistency.
  - ↳ Client crashes midway
  - ↳ Two clients write at same time
  - ↳ Could read inconsistent data (!)

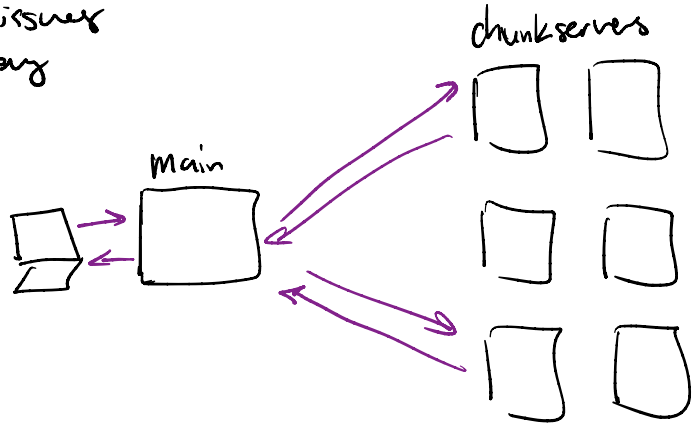



- Also: Problems with scale!
  What happens when FS replica is too big to fit on server?

## Attempt 3

Client sends R/W reqs through a single coordinator
↳ keeps track of which file is where

+ No consistency issues
+ Can scale to very large # of disks




Problem? Take note

- Main fails?
- Throughput problems
  ↳ Limited by throughput of main server.

⇒ GFS is essentially Attempt 3 with some improvements to handle throughput.

# Key GFS properties

* Single main server

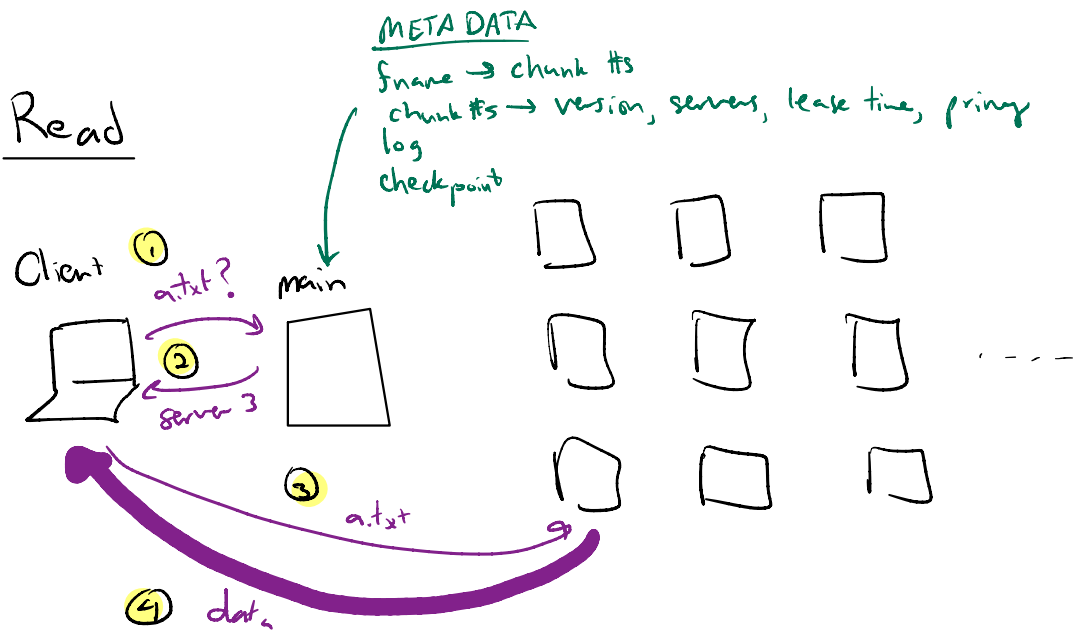
* Data reads/writes bypass main server
  ↳ Can introduce some inconsistency..
  let app worry about it.

* Weak consistency guarantee
  ↳ Two replicas might not store exactly the same copy of a file.

LESSON: Simplicity wins! (e.g. 80 page papers)

# GFS

## Read




- Why does this provide fault tolerance?
- Why is performance good enough?
  - ↳ What would happen if chunk size was very small?
    - metadata large
    - load on main
    - client talks to many chunks

Write is similar, except that primary must take a "lease" on file to be written (like a lock)

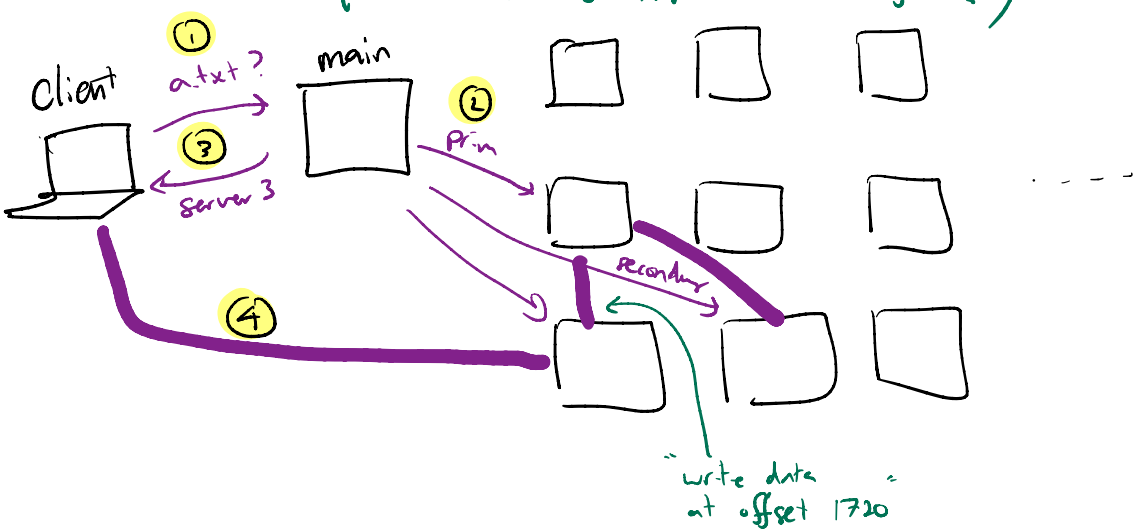
# Record Append

**Guarantee:** Append at least once (maybe more!)

↳ Order one burrito... fulfilled "at least once" ?!?!

↳ Different clients see different file contents.

(When would this be useful?)




Why is this useful?

- Many appends can happen concurrently.
- What happens if write fails?
- What happens if two clients write concurrently?

# Problems?

- Hot spots (e.g. run binary on 1000s of clients at once)

- Main server can fail
  * Log - write important things to disk
  * Shadow main server

- Many files → throughput ↓
  * Main is a bottleneck

- Concurrent writes
  * Not great, but not terrible

- What if three servers fail?

- All main replicas fail?