Recitation 14: GFS
Plan

* Recitation Qs
* Setting for GFS
* How GFS works

Logistics

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

GFS paper uses "master" for
server that coordinates chunk servers

⇒ 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
   - Blu more important than latency

2. How does it exploit these assumptions?
   - Record append
   - Large chunk size

3. Why does GFS make these 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: Ghemawal was MIT PhD, advised by Liskov & Kastenhofer.
To understand GFS, let's try to design our own storage system.

**Attempt 1. Single Server**

- Simple?
- Fault tolerance
- Capacity
- Throughput

**Problem? Take vote**

- Ask for rep

**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?
Client sends R/W regs through a single coordinator which keeps track of which file is where.

- No consistency issues
- Can scale to very large # of clients

Problem? Take note
- Main fails?
- Throughput problem
  - 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)
Write is similar, except that primary must take a "lease" on file to be written (like a lock)
Append at least once (maybe more!)
- Order one burst...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. our binary on 100s 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?