Recitation 14 — GFS

Overview

- Goal of GFS: Shared file system spread across hundreds or thousands of physical machines
- Adds complexity: replication, many users writing to same files, huge files.
- Relies on some assumptions: mainly append-only workloads, read-only after write
    - Means complicated consistency models and atomicity guarantees aren't necessary

Components

- Controller ("master" in paper): has an organizational/control role
- Chunk servers: where the data actually lives
- Chunks: large pieces of data

How it works

Figures 1 and 2 of the GFS paper are some of the best figures you will see in any paper in 6.033. Take inspiration for your design project system diagrams!

- Reads:
    - Client sends file name + offset within file to controller
    - Controller replies with set of servers that have that chunk
    - Client asks nearest chunk server
- Writes:
    - Client asks controller where to store the file; controller responds
    - Client pushes data to the chunk server that's closest, which forwards data along to the others
    - Primary chunkserver applies serial numbers
- Notice that the controller handles control traffic/metadata type things. Not moving real file data.

Discussion

- Huge sequential reads and writes, appends work well in GFS
- 3x replication lets it be quite fault tolerant (think about this more as you read MapReduce)
- The controller is a single point of failure, though it can also be replicated, and is quite lightweight
    - Also remember: the probability of a specific machine failing is relatively low. GFS is addressing the problem that the probability of *some* machine failing — and thus us losing data — is higher.