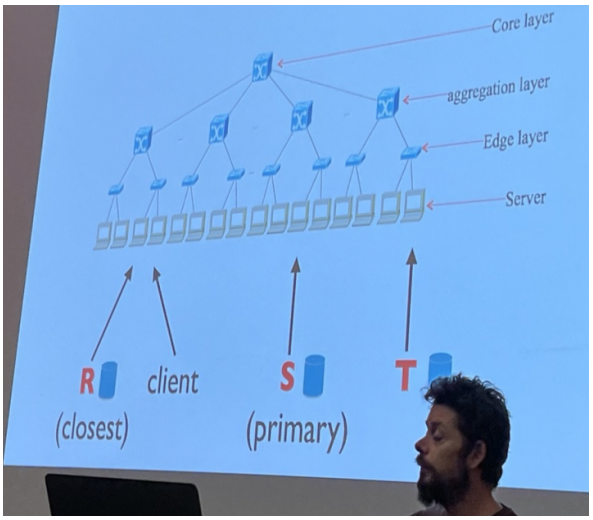
CAS CS 350
Lec 9

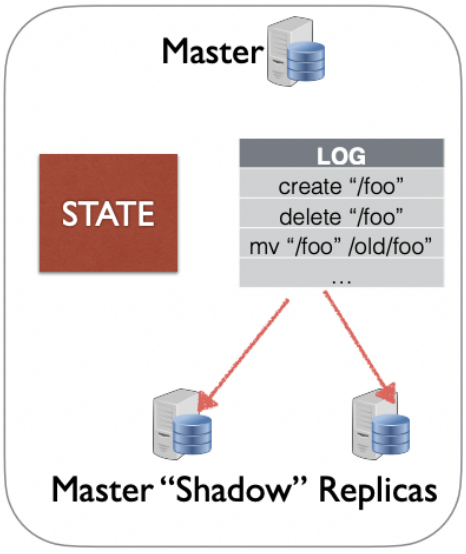
Google File System

1. Review
    a. Write Steps
        i. When a client needs to write on a chunk, it contacts the primary to get the nodes that have the chunk (3 replicas of the same chunk)
        ii. Client sends data to the closest replica → replica contacts the closest replica and continued
        iii. When there is a problem and data does not arrive to one of the replicas (say Backup B), backup B keeps stale data
        iv. If another user tries to access backup B, he/she sees stale data
2. Example Network Topology
    a. 

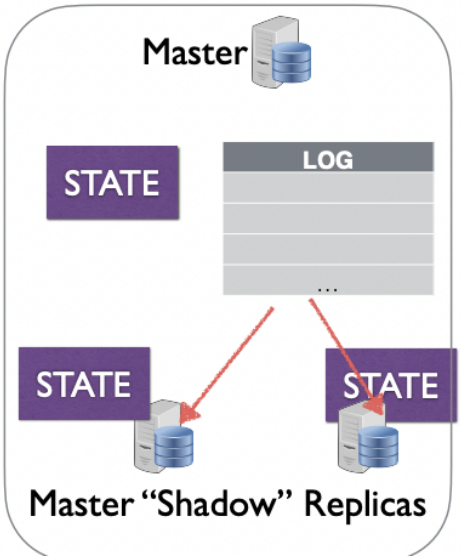
    b. In this case, the client is closest to replica R, so it will send data to replica R
    c. Why not primary S?
        i. If client decide to send to primary S, primary S needs to send data to replica R and T
        ii. Latency is the issue
        iii. It also has redundant transverse (we have to go to core layer once if going from client → R → core → S compared to client → core → S → core → R) → consume more bandwidth if going from client to S to R
3. Concurrent Writes/Appends
    a. Concurrent writes to same region of file
        i. Write on a region result in a mix of those writes
        ii. Few apps do this - but same semantics on Linux
    b. Special support for “record appends”

i. Guarantees atomic at-least-once appends
ii. Primary coordinates – tell replicas exact offset
1. Fail to client and retry - duplicates can arise

4. GFS Fault Tolerance
   a. Two failure domains
      i. Master
      ii. Chunk server
5. Fault tolerance: Master

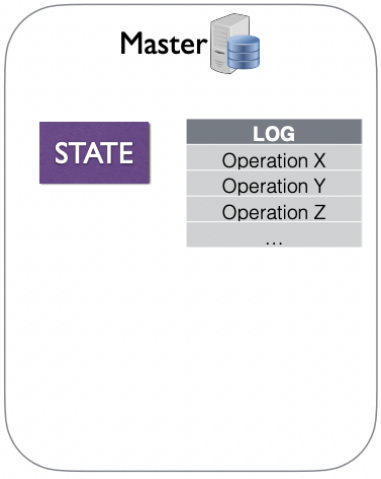


   a.
   b. Save history into an ordered log
   c. Log is replicated across several backups (if the disk/node the coordinator runs dies, we can use the replicas of the log)
   d. Clients operations that modify state return "after" recording changes in "logs"
   e. Coordinator contains state that contains every important information
   f. Coordinator keeps track of changes that users do in a "log" and then apply the change (reason: if something goes wrong or bug appears, then you can undo and go to previous version)
   g. Replication logs play a central role in most "strongly consistent" protocols



h.

i. Reconstruct the disk by following the statements within in the log (if the disk/node dies)
j. You cannot submit "write" requests in subtle replicas (because it may give access to stale data)
k. Log files can become huge (when too much data is included → use checkpoint to save space)
l. Limit the size of the log with checkpoints
m. Make a checkpoint of the master state
n. Remove all operations from log from before checkpoint
o. Checkpoint is replicated to backups (sends the checkpoints to the replicas as well)



p.

q. When there exists a checkpoint, recovery is easier
r. Replay events from log, starting from last checkpoint
s. How chunk location information is retrieved after a failure?
    i. Chunk location information is recreated by asking chunk servers

6. Summary

a. Master is single point of failure
    i. Recovery is “fast”, because master state is small
    ii. Service maybe unavailable for “short time”
b. Shadow master (log replicas):
    i. State lags behind master
    ii. Replay from the log that is replicated
    iii. Can serve read-only – may return stale data