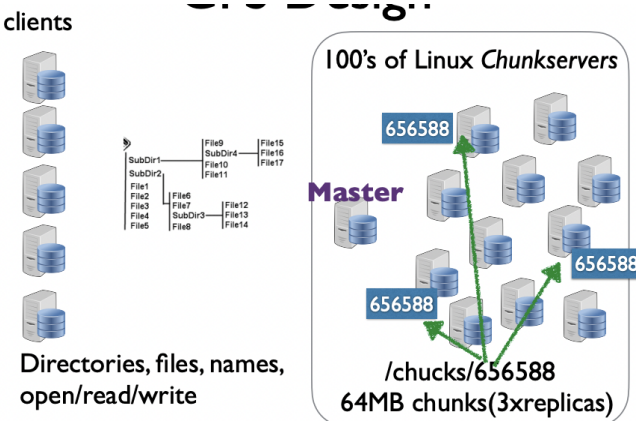
CAS CS 350
Lec 8

Google File System (GFS)

1. Review
    a. GFS goal → manipulate and store large amounts of data, multiple users trying to access the data of the same kind (ensure and handle and balance between concurrency and consistency), handle failures, target specific type of workload, how to provide an abstraction that looks like a simple file system running on the laptop (GFS system actually has large amounts of machine but on the computer, it acts as if a simple system where the file is simply stored on the users' computer)
    b. Strong consistency system → users will always observe the latest update of data, Weak consistency system → users may access the stale data
        i. Which one is better? → depends on the circumstances
    c. Inconv
2. Inconvenient Truth
    a. Strong consistency is great for design of distributed operations
    b. Strong consistency is terrible for the performance of distributed operations
    c. Weak consistency is terrible for the design for the design of distributed operations
    d. Weak consistency is great for the performance of distributed operations
    e. In reality, many different consistency models exist employing various degrees of correctness guarantees
3. GFS not ideal by design
    a. Paper illustrates a design struggle between:
        i. Data correctness
        ii. Fault-tolerance
        iii. Aggregate performance/scalability
        iv. Simplicity of design, usability for developers
4. GFS goal
    a. Shared filesystem for Google developers
    b. Hundreds or thousands of physical machines - i.e., cheap, disposable
    c. To enable storing massive data sets, sequential read and append-heavy workloads
5. The files
    a. Multi-terabyte data sets
    b. Most of the files are large
    c. Authors suggest 1Million files * 100 MB = 100TB
    d. Majority of writes are append only
6. The challenge
    a. Failure of machines

b. High-performance workloads
    i. Many concurrent readers and writers
    ii. Huge sequential reads and writes
    iii. Streaming data / use network efficiently

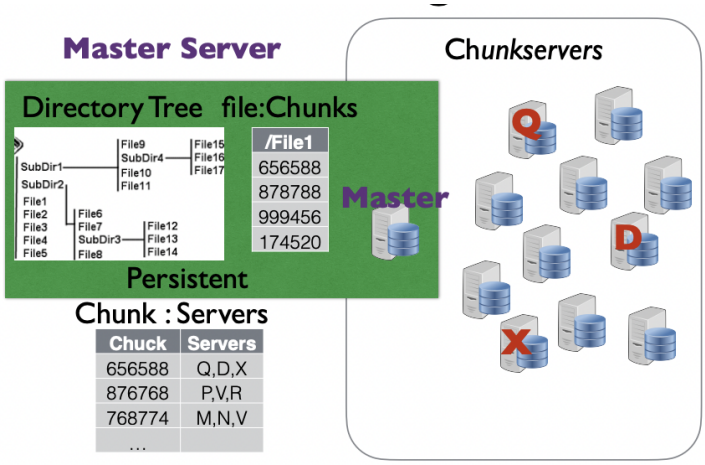
7. GFS design
    a. 

    b. Hierarchy of folders
    c. Files are divided into 64MB chunks (chunkservers → linux machine with local hard disk) → single file users view are 64 MB chunks (each chunk has 3 replicas in different machines)
    d. Why 64 MB chunks?
        i. Coordinator can keep small amount of data
        ii. Can do way more requests in the same time without fetching new chunks again
        iii. Availability
        iv. Load balancing
        v. Locality
    e. Why 3x and not 2x and 4x?
        i. More replicas you have, the more fault tolerant you are
        ii. However, the more replicas you have, you need extra disks (the cost of it is expensive) to store them and the cost of updating the duplicates is expensive as well
        iii. Using 3 replicas significantly reduces chances of losing data
        iv. Using more than 3 replicas is too costly

8. Quiz 1
    a. Assume there are 23 machines
        i. Each machine fails once a year
        ii. Machines fail independently
    b. What is the probability of having at least two machines down the same day - birthday paradox
        i. More than 50%

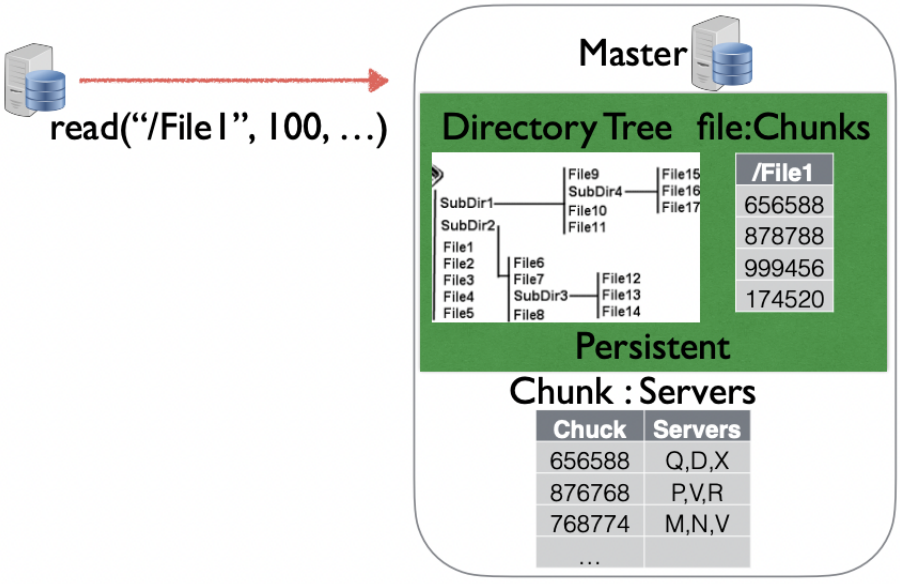
ii. We lose data with more than 50%
iii. In addition, in real world, the failure of machines is not independent of each other → makes the situation worse

9. GFS design

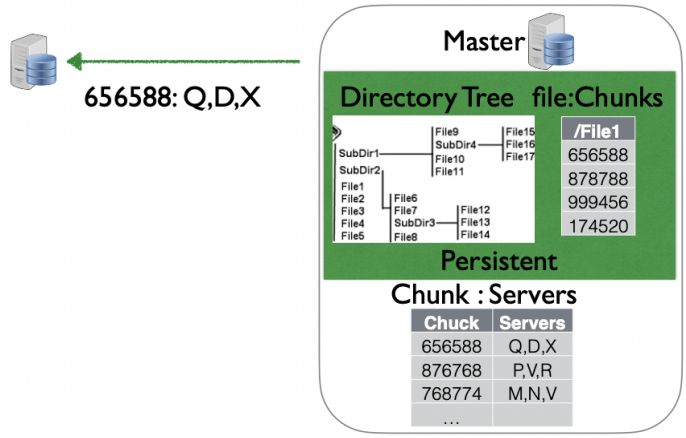


a.
b. Master/coordinator knows everything (doesn't store actual data, store metadata regarding the files)
c. Chunkservers → stores data

10. Basic Reads



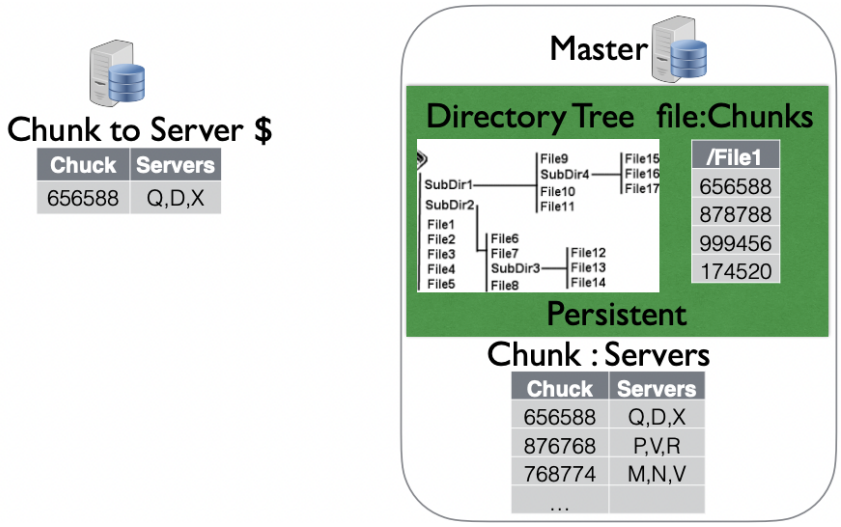
a.
b. Client application communicates with the master

c. 

d. Master checks the data and sends back the servers that contain the data

e. If one of the data fails, the user simply requests the machine (not the master) to see the duplicates of the data

f. This is because recontacting the master takes a lot of time (master is just one)

g. 

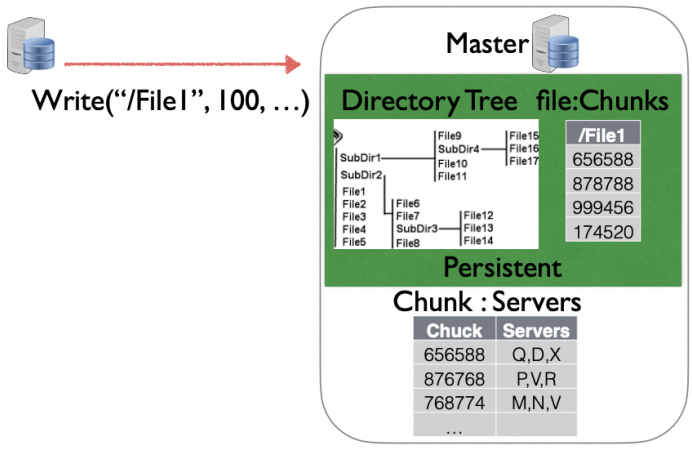
h. 

i. How does client know the closest server?

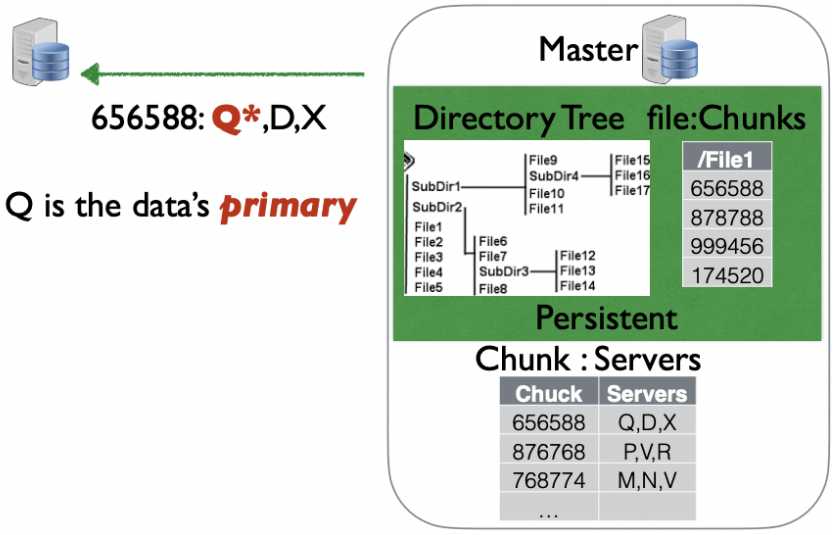
   i. Smallest number of hops to reach to the destination

11. Basic Writes

a.



b.



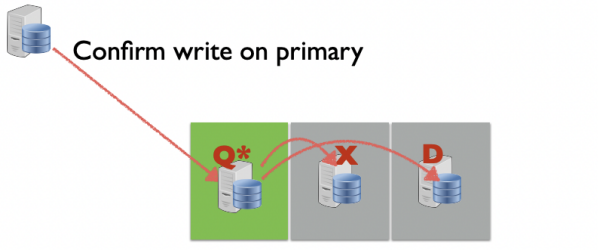
c. Similar step as read

d. The master does not know whether client is changing the data

e. Client directly contacts the replicas using its ID (the closest one) and writes on it

f. The closest replica sends data to its closest replica

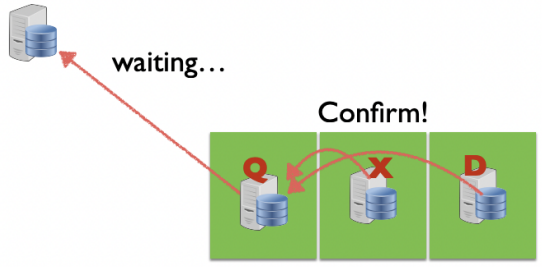
g. Continue until all replicas contain the update (the user only writes one time)



Primary job is to accept the request and chooses order of application (with concurrent requests) and applies change

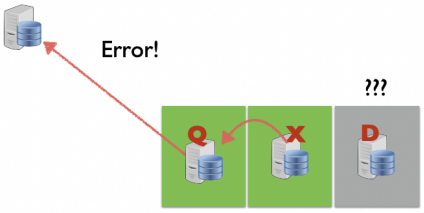
h.

i. Primary replica that decides which client writes first



Primary confirms when the write is completed across all three replicas.

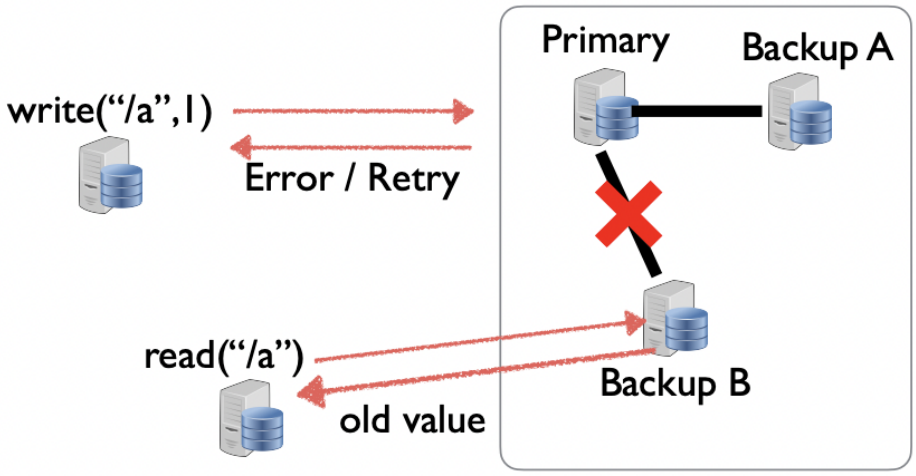
j.



Else, error is reported to client. Client retries

k.

12. Example



a.

b. Benefit of client reading stale data?
   i. One possible solution is to block connection to backup B until the error is fixed → if this happens, the reader can see updated data, not stale data
   ii. However, if this happens, there will be no connection to one of the backups in the system and you lose concurrency here (there will be limited number of requests in the total system)