System Design : Google Drive

YouTube @MsDeepSingh

Functional requirements

1. Upload Files – file size upto 10GB + any type of file
2. Download Files
3. Sync files across devices
4. Share files with other users.

NFR

1. Availability
2. Scalability
3. Consistency – all users should see same file.
4. Durability + Reliability
5. Large files download/upload should not fail on network intruption and should be resumable.

Estimations → based on assumptions

Number of users - Total 1 Billion

Daily Active users - 10 M

Storage allowed/user - 15 GB

lets take storage for 2B users – considering user growth

2B * 15 GB = 28610 PetaBytes ~ 3K PB

→ assuming each user upload 2 files/day of 100 KB

Storage = 10M * 2 * 100 KB

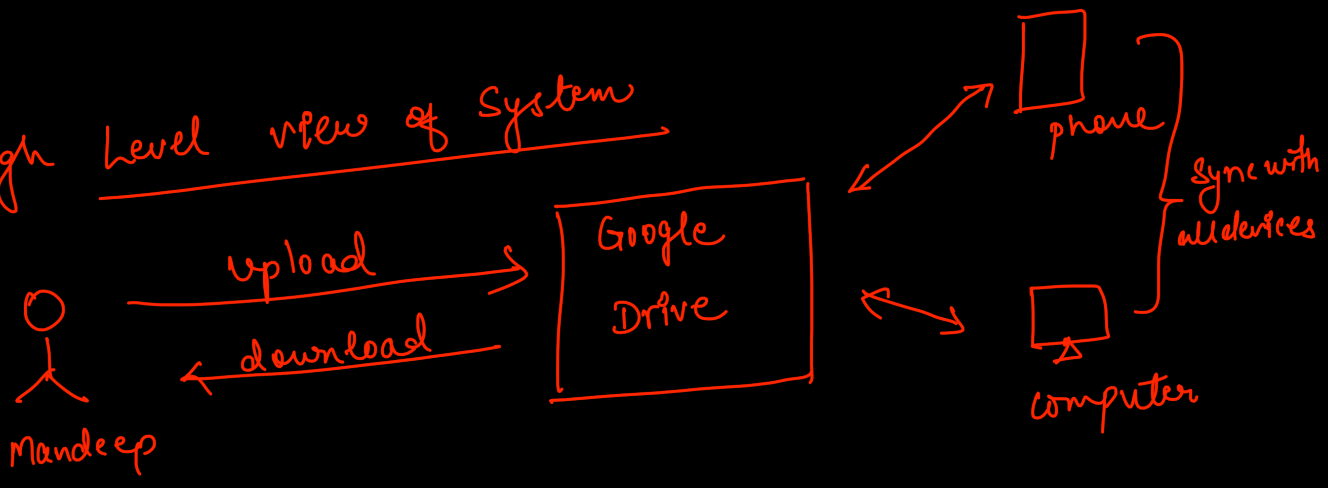
Upload TPS = (10m * 2) / day

$= \frac{10m * 2}{24 * 60 * 60}$ = 232 requests/sec.

Active connections will be **not more** for sync feature.

→ in proportion to daily active users.

High Level view of System



How does upload work?

File size <= 10GB

requirements →

1. It should work on devices with different network types.
2. minimum possible network bandwidth usage.
3. Upload should not fail due to internet connectivity.

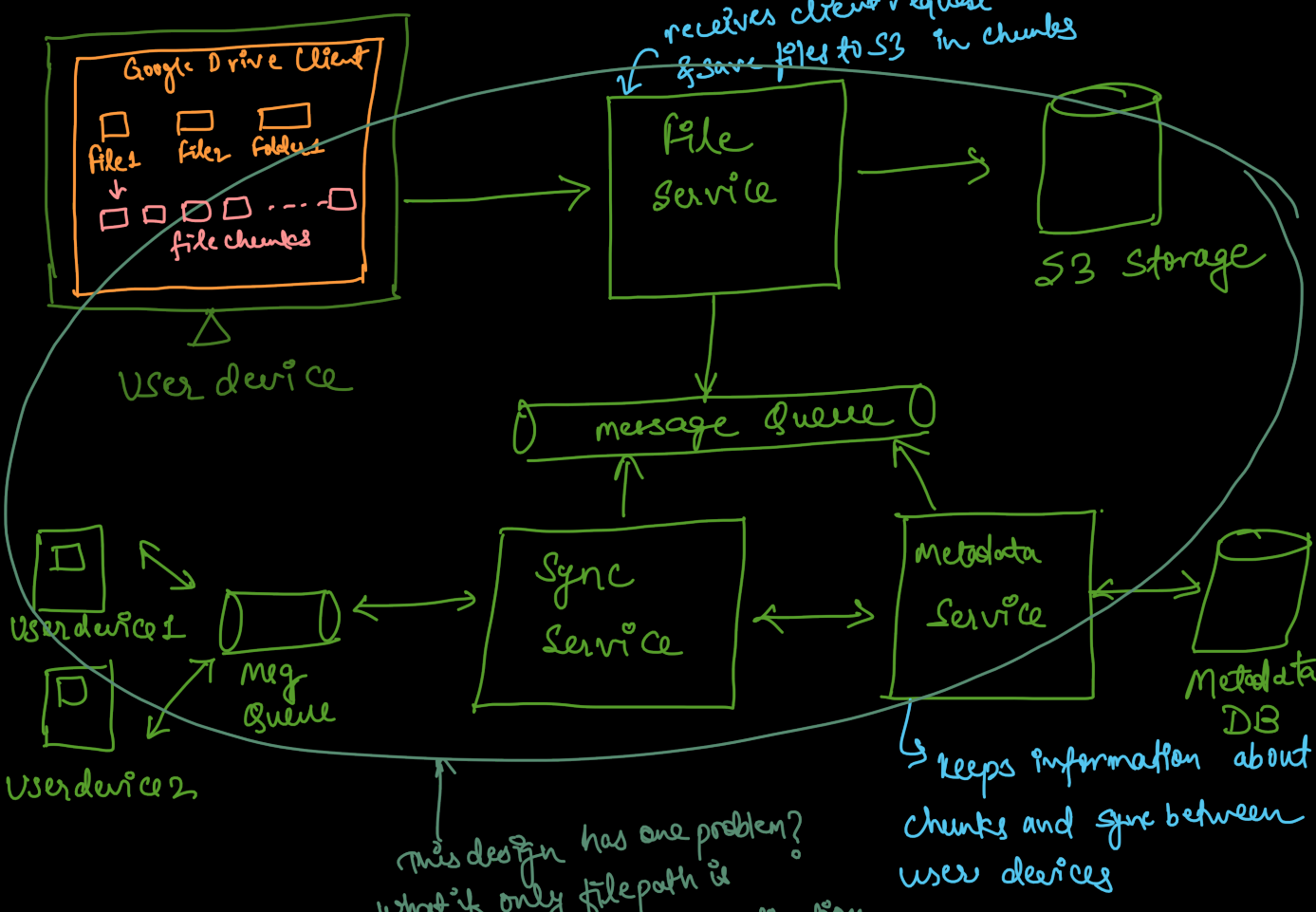
such issues are in general solved via breaking large files into smaller chunks.

Helps in

↳ parallel chunk upload

↳ failure only require uploading remaining chunks

↳ Helps in versioning → easily figure out the diff.



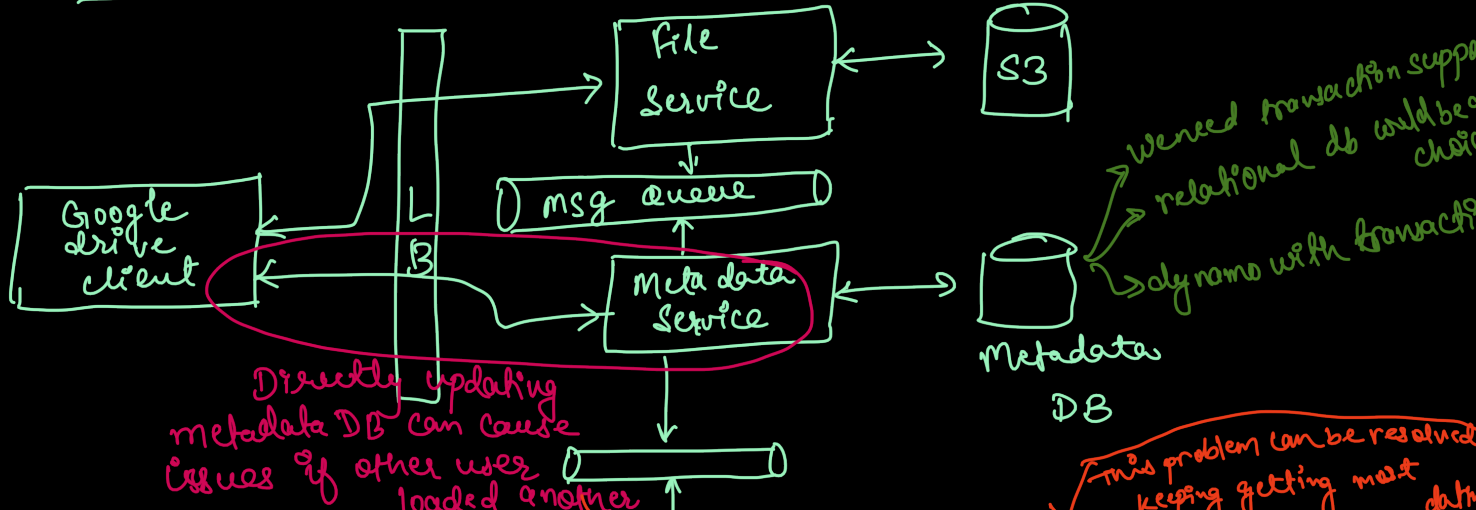
Event Driven Architecture

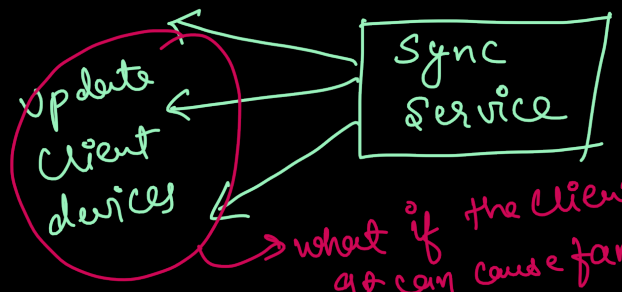
↳ Since file upload is not latency sensitive operation, asynchronous mechanism can be preferred.

① client to Uploader Service

② Sync Savice to Clients

Architecture Improvements





→ what if the client list is huge? It can cause fanout problem

recent metadata & up

client.

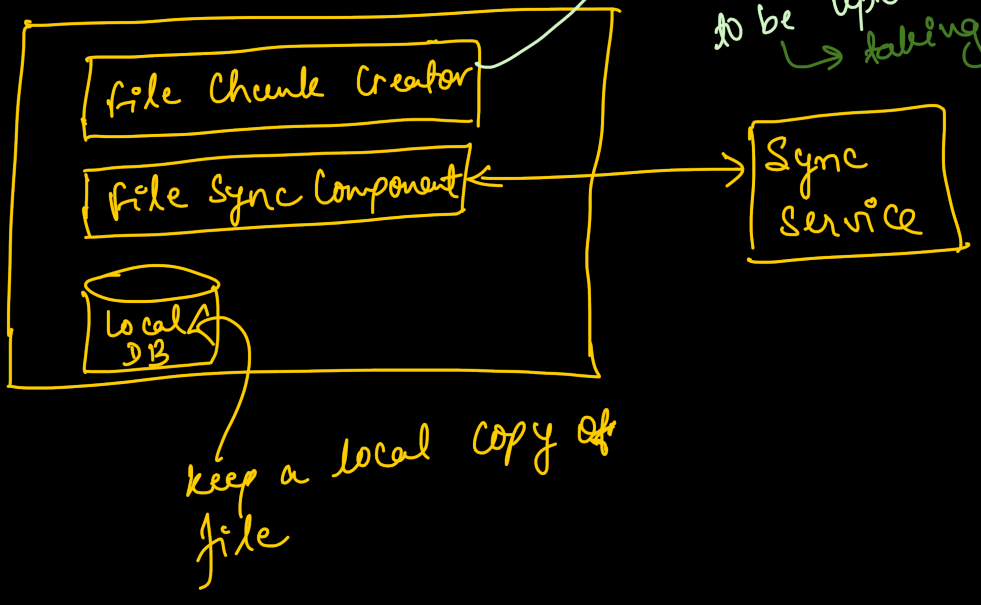
for concurrent writes, it usually committed but is resolving any merge conflicts.

Conflict Handling for files

↳ last write wins

↳ ask user for which version to keep or combination of both files.

Google Client Service



→ create file into chunks say 4 MB

→ for file update, which specific chunks to be uploaded again.

↳ taking hash or merkle tree usage

watch dynamo video

keep a local copy of file

Think Further

① As files are divided into chunks → there are huge chances that chunks of different users are same → Can we optimise our storage space??? via file service,

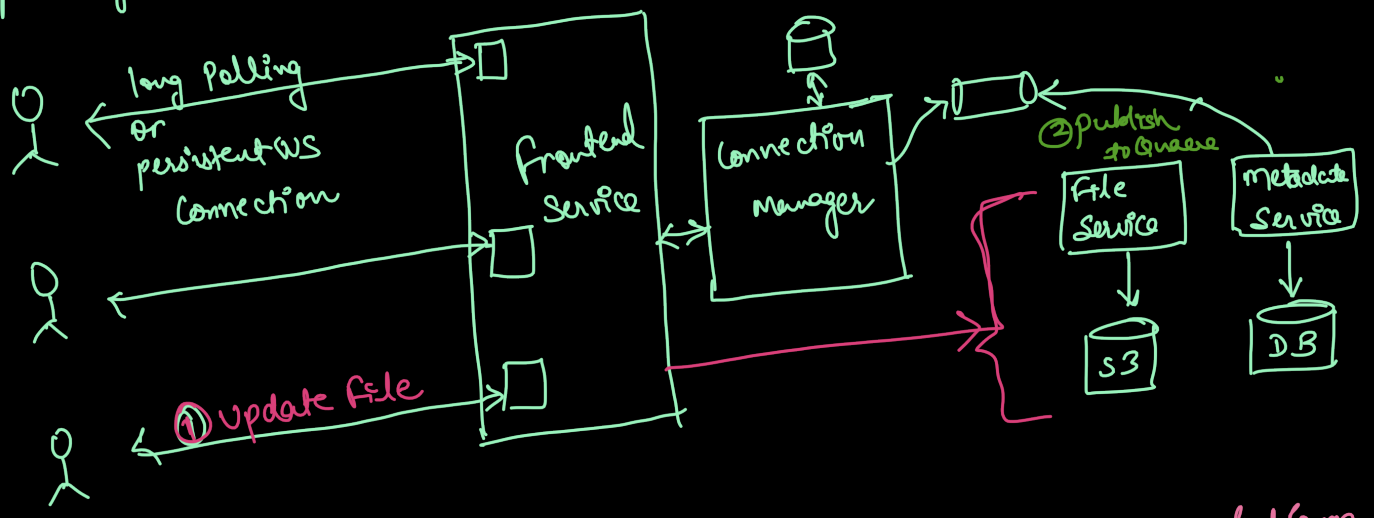
② Instead of uploading files to S3 Can we store file to CDN? → and then move to S3.

↳ This will add extra complexity.

We don't need all data on CDN.

starting point could be storage to S3 and then cache more accessed files to CDN.

③ Updating all clients for any update in file?



The above architecture is similar to messaging platform, it is useful for regularly file update service such as 100 users working simultaneously on Google Doc.

Subscribe for more such content.

@msDeepSingh @msdeep14

Happy Learning ☺