System Design - Google Docs

YouTube @MsDeepSingh

Functional Requirements -

1. * Multiple people edit document concurrently - max 10D
2. Share the document c (with) people
3. Edit - create / update / delete / undo / redo
4. Offline doc editing
5. Comments / suggest edits / doc versioning

NFR

① Low latency → users should be able to view other people changes in real time.

② High availability

③ Durability

④ Consistency + Conflict resolution

⑤ multi Device Support

⑥ Users authentication + security

Estimations

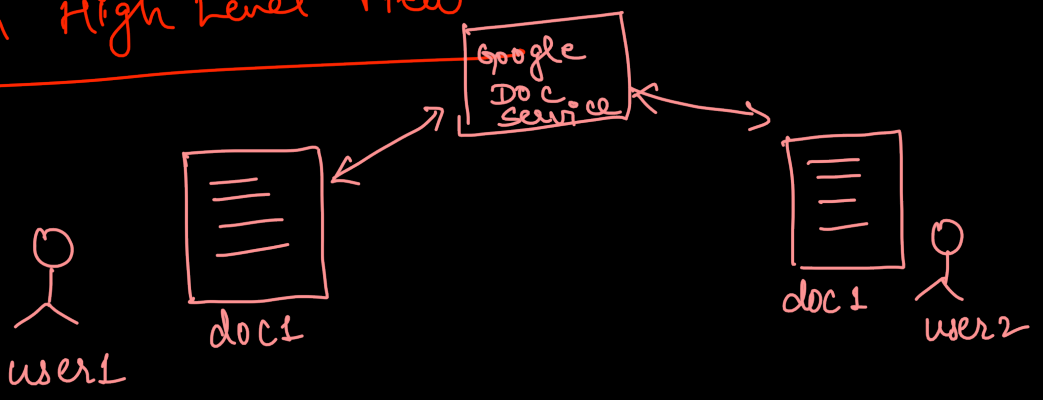
① ~25 million people use google docs (source: Internet)

29% use it daily ~8 million DAU

② Average 10 users are part of same document (assumption)

concurrent editing

System High Level View

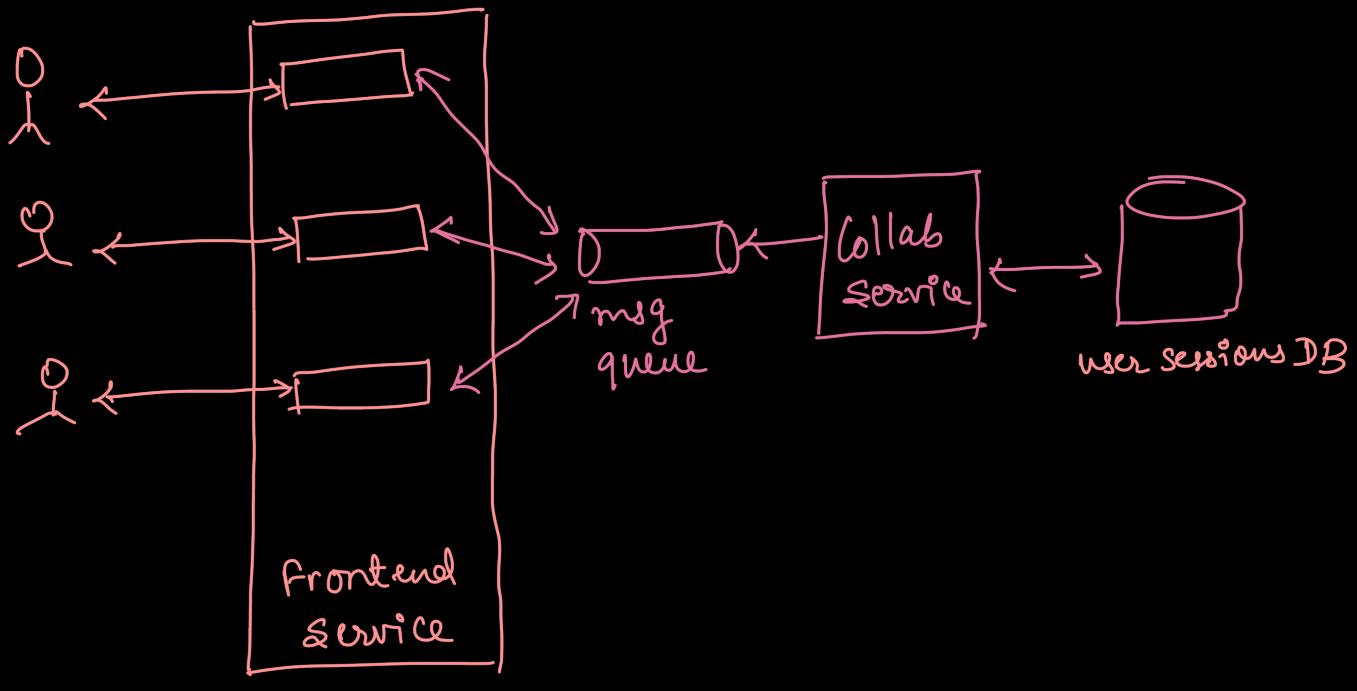


A

How different users are in sync always?

↳ This is similar to a chat/messenger application where two or more users are sharing some msgs.

In Google doc as well, two or more users are sharing the local updates with others. So, similar architecture will be helpful here.



B Concurrent users doc update Handling

① Locking is one way

→ Pessimistic Locking such as java ReentrantLock()

↳ will only allow one user at a time, doesn't solve our use case.

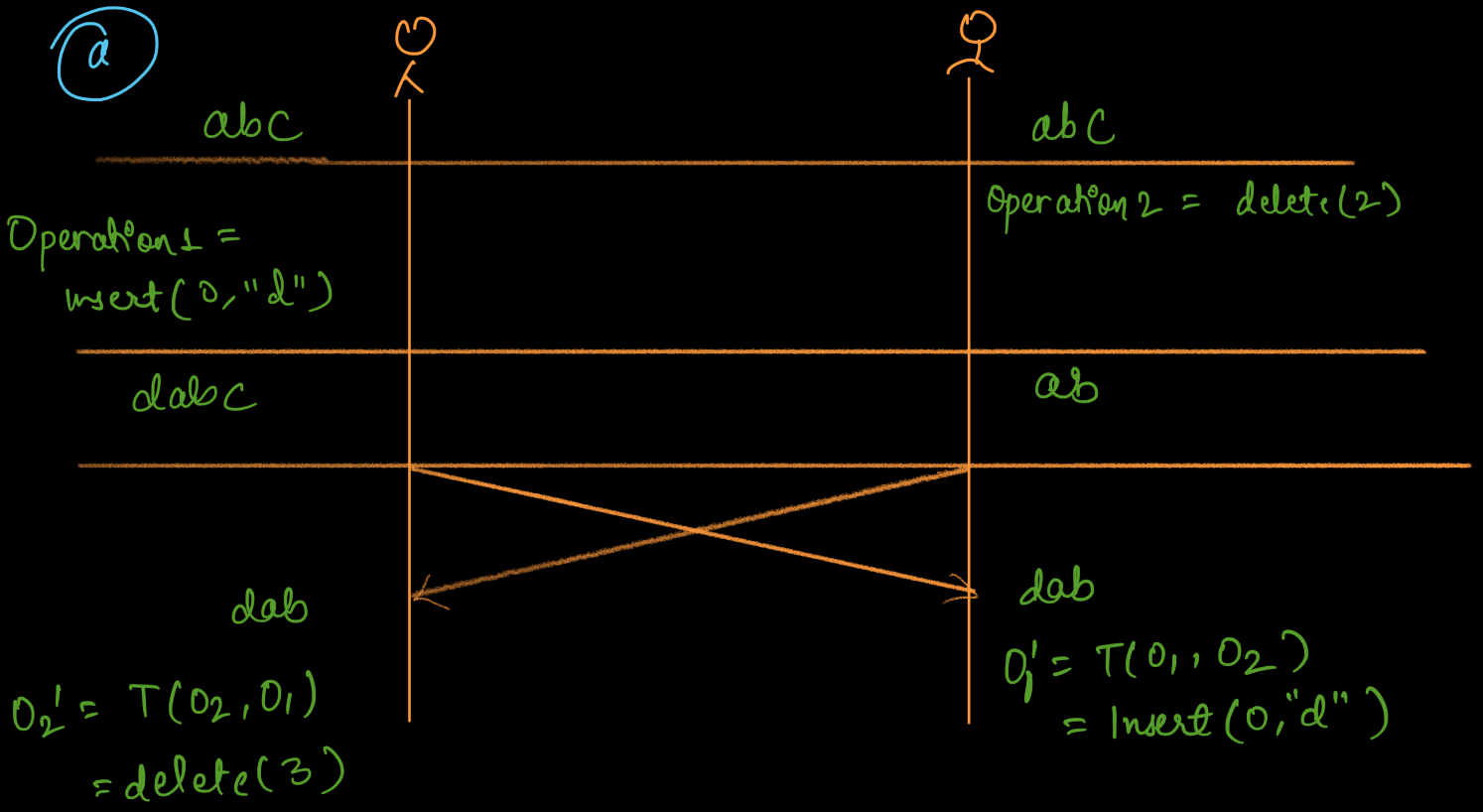
② Optimistic Locking mechanisms

ⓐ Operational Transformation

ⓑ Differential Synchronization → similar to git diff & merge.

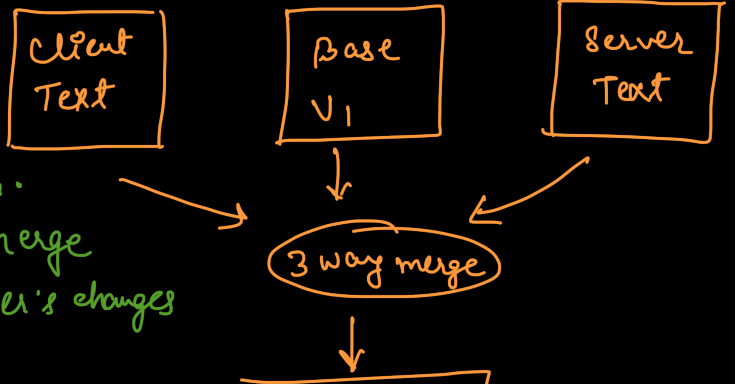
ⓒ Three-way merges ⓓ CRDT

ⓐ



⇒ Google search "Google wave Operational Transformation whitepaper" and read on wiki for more information.

ⓑ 3-way merge



① Client send text to Server.

② Server perform 3way merge to merge client & other user's changes

③ server send new copy to client.

Base
V2

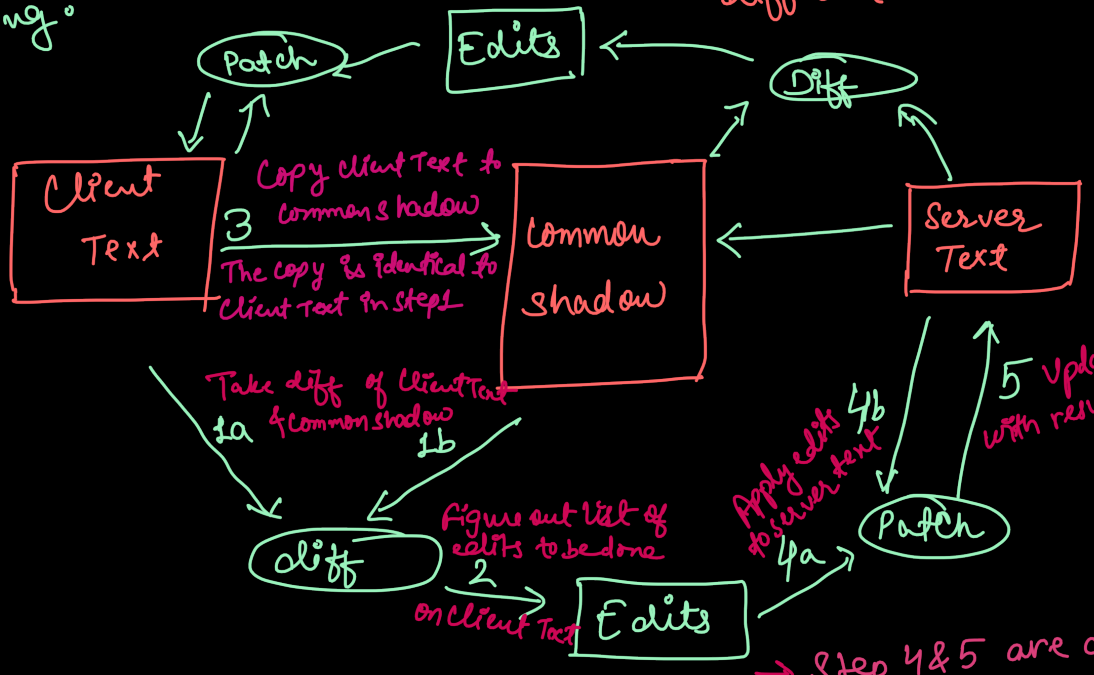
This is extension of pessimistic locking and similar to Two phase locking.

Example → for 5 seconds, all users are allowed to write. Then everyone wait for 1 second for server to merge all users changes.

Read paper "Differential Synchronization" by "fraser@google.com"

ⓒ Differential Synchronization → based on taking diff of doc copies from different clients.

→ All three are same in beginning.



→ Step 4 & 5 are atomic.

→ Client Text, Common shadow & Server Text are same in beginning

ⓒ Doc Versioning

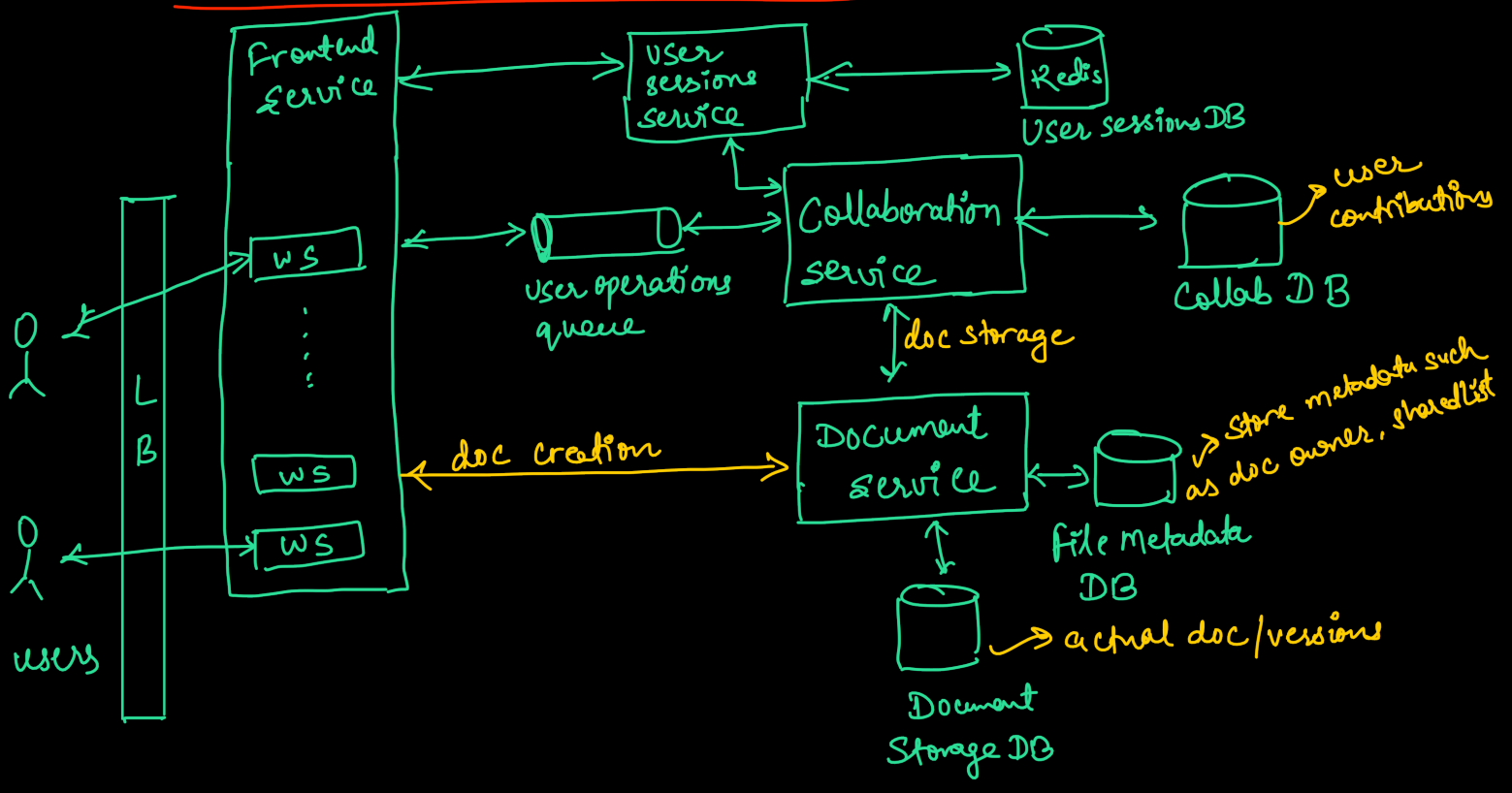
① keep all copies of document → will require lot of storage to maintain duplicate copies.

② Only keep what was changed → Divide the file into chunks and check their hashes, one such approch is Merkle Tree

③ Time Series DB

Discussed in 'Dynamo Architecture' Series in content of replica synchronization.

⑤ End to End Architecture



② User Operations Queue → Captures keystrokes by users and these changes are applied to all docs that are being edited concurrently.

① Frontend Service → dumb service which allows users to create a persistent connection.

③ User sessions Service → Sessions DB data is accessed by multiple components. This service provides APIs to perform CRUD.

④ Collab Service → Consume continuous keystrokes and apply

operational transformation and send response so as client application can update the doc.

Consideration point → The client should send new request once ack is received from server to ensure updates are applied smoothly.

→ Collab Service can also help in
  ↳ user chats.
  ↳ show who is online.

→ Services communicate with each other basis events published.

→ Distributed Transactions to ensure consistent DB stage among multiple services.

Data Storage Choices

① where is actual document stored?
  ⓐ S3 → S3 is more helpful in storing entire document and keep on replacing as document gets updated
  ↑ costly operation
  ↳ providing search capability on couple of doc will be complex task as we're not building any document content index.

  ⓑ Document DB → To enable rich text like bold, bullets, etc the file in storage can be saved as markdown | odf | plain Html/RTf format.
    ↳ DynamoDB
    ↳ Cassandra
    ↳ HBase
    ↳ Amazon Document DB
    ↳ Google Propriety Database

What are the requirements?
① Store entire document
② Easy to access and apply transformations.
  ↳ Update data at character index = 6

We'll be discussing database choices in details in follow up video.

Stay Tuned

Subscribe for more - @MsDeepSingh

Happy Learning ☺