Requirememts

1:1 chats Online/Offline msg Read receipts

Chat History and Group chat

Real time chat(Low Latency) Consistency

Capacity Estimation

Assuming 500 Mn DAU and each user send 40 msgs daily = 500mn*40 = 20*10^9=20B

Assume each msg - 100 bytes => 20B*100=2TB

Store for 5 yrs = 2TB*5*365=3.65PB

Extra Info -> userId, msgId, metadata(time stamp, chatId) = 200Byte

Bandwidth Estimation & Services

As we have 2TB/day = (2*10^12)/24*60*60 = 25MB/sec

MessageHandlingService

Pull - User will periodically fetch msgs.

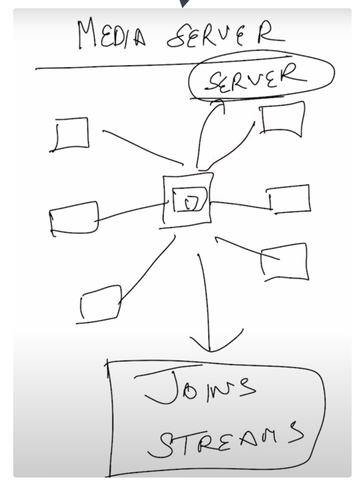
Disadvantage - Too many req to server even when no new msgs are there.
If user is offline, server has to track how many new msgs are there

Push - Server will notify user when new msgs are there,less resources wasted.

Disadvantage - Track msgs when user offline. We need to maintain open connection.

Web RTC for Voice Call

Upto 100 connections you can use Peer to peer but for more than that it will be difficulat and it's ebtter to use Client Server - Media RTC connection.
Web RTC is used to share data between multiple peer browsers. So Media RTC joins multiple streams and sends to one server. So iss case mai instead of N^2 connection ke vo sirf N connections rakh skta hai. Agar N^2 hua to zyada disk utilisation and CPU usage hoga, jo zyada resource utilise krega.



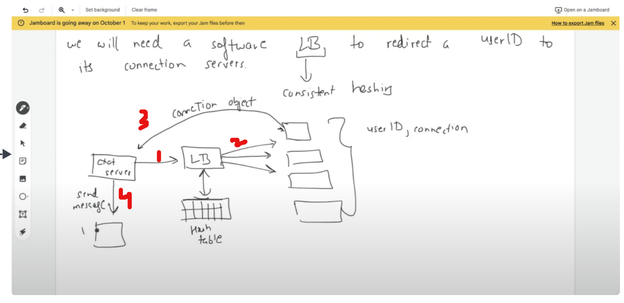
Maintain Open Connection

Maintain a map of connection(User-> connection)

Total no of sockets on avg 65536 but usable are 50k/system

Total connection = (500 * 10^6)/50000 = 10k servers

So we use Load Balancer for this, and we can use consistent hashing.



Storage

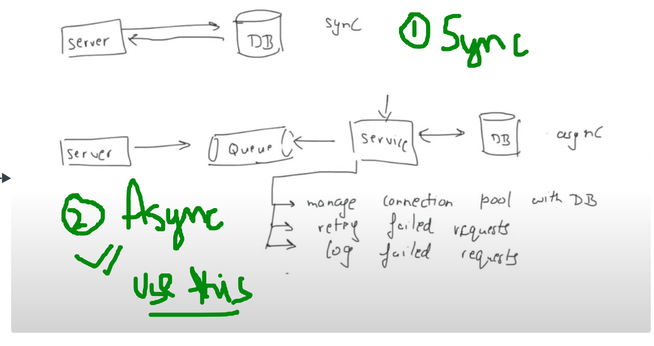
We can save by 2 ways - Sync and Async

Sync - create a thread to store msg in DB

ASync -send event to other service which will in turn save in DB

Database with RDBMS if want to use, then use with CQRS(different server for reads and writes) in async.
Now FB uses MyRocks which is a MYSQL DB

NoSQL with Hbase(Wide column DB) - multiple rows can be inserted agains 1 key. For Hbase backend is Hadoop(Map-reduce)



Data partitioning

MessageId OR UserId

Total Msg storage = 3.6PB, modern dB can store = 4TB
so total = 3.6PB/4TB = 900 shards

MessageId sharding will cause joining of 900 shards so it will be super slow so we do on userId so all msgs are there at 1 place.

DB

Chat - senderId, timestamp, recieverId, msgId, contentId

For Group chat replace recieverId with groupChatId and use one more table to see who all are there in the group