⇒ Design a Messaging Appⁿ

(Ex: FB Messenger).

1) MVP
2) Scale Estimation
3) Design TradeOffs.
4) Design Deep dive / API's / Data flow.

MVP.

1) Send / receive messages. → Can contain media content as well.
2) Message History.
3) Conversation History. ⇒ list of Chats.
4) Chats should be realtime.
5) Group Conversations.
6) Edit / Delete a msg
7) Online / Offline indicator.
8) Message seen or delivered.
   ↳ Timestamp.

Estimation of Scale.

# of users = 3B

DAU = 1B

Avg no. of msgs sent by a user everyday = 20

Total # of msgs = 20B/Day.

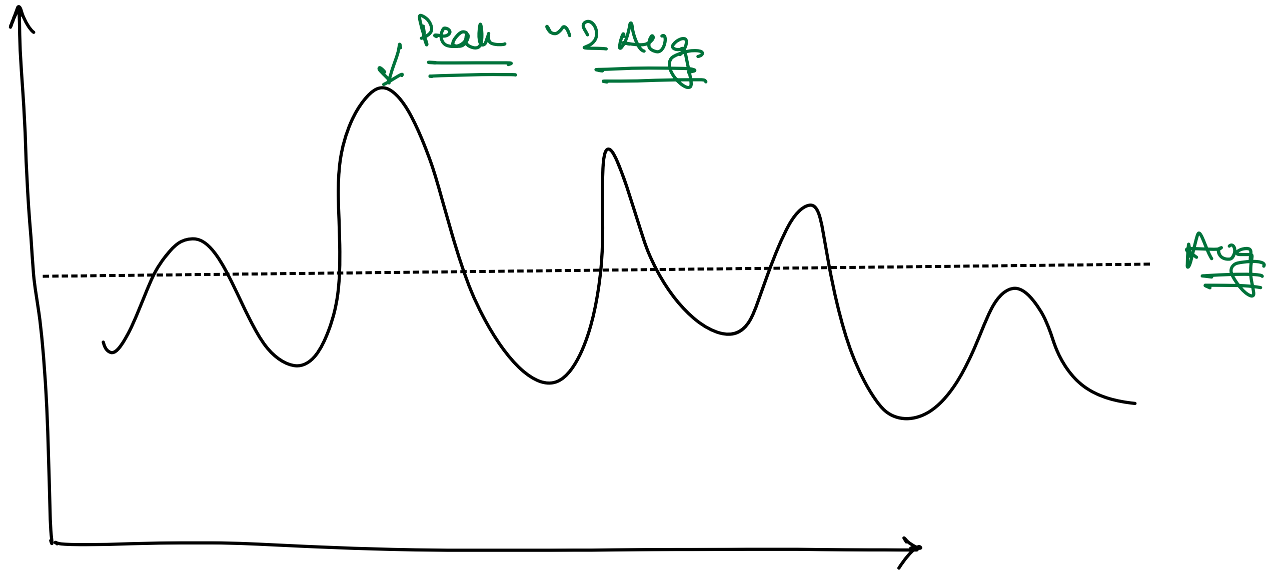
Write Queries.

Reads ≈ Writes.

$$\text{Write QPS} = \frac{20 \times \cancel{10^9}\, 10^4}{\cancel{86400}\ \cancel{10^5}}$$

$= 200K$

⇒ Read + Write heavy.



Storage estimation

Total # of msgs = 20B | Day.

msg
- msg-id → 8B
- sender-id → 8B
- receiver-id → 8B
- timestamp → 8B
- Content (Text) → 100B.
- media – image, video, files. } file Storage : S3 | Blob Storage.
  - S3 url : 30B
- metaData
  - 20B.

≈ 200 Bytes

1 Day = $20 \times 10^9 \times 200$ B.

= 4×10^{12} Bytes.

= 4TB.

10 Yrs : 4TB × $\cancel{365}$ × 10
400

16×10^3 TB

16 PB.

⇒ Sharding would be required.

Trade Offs.

⇒ High Consistency vs High Availability.

⇒ Low latency

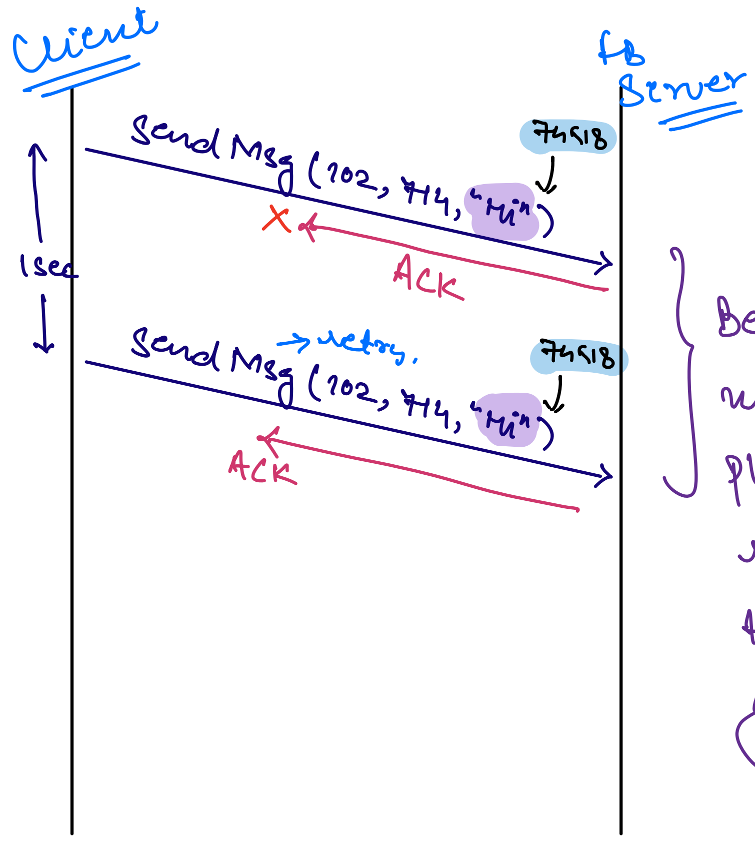
API's.

→ send_message(sender_id, receiver_id, content, msg_id)

→ getConversations(user_id, offset, limit)

→ getMsgs(user_id, conv_id, offset, limit)

⇒ How to make our Messaging App Idempotent.



Because the ACK got missed at the first place so, client will retry the msg and the same msg will get delivered twice.

⇒ Online Payments.

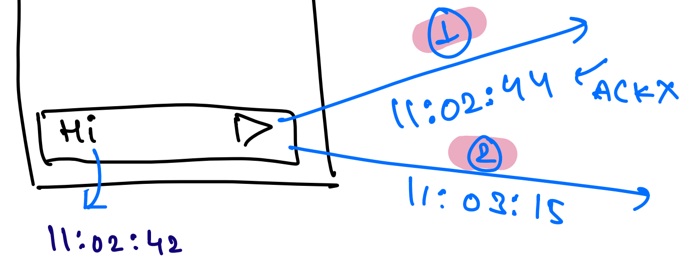
send_message(sender_id, receiver_id, content, msg_id)

Idempotency key.

msg_id : user_id + device_info + timestamp of last typed character

+ . . . .

Idempotency key.

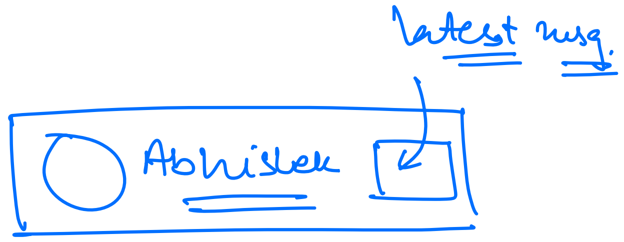


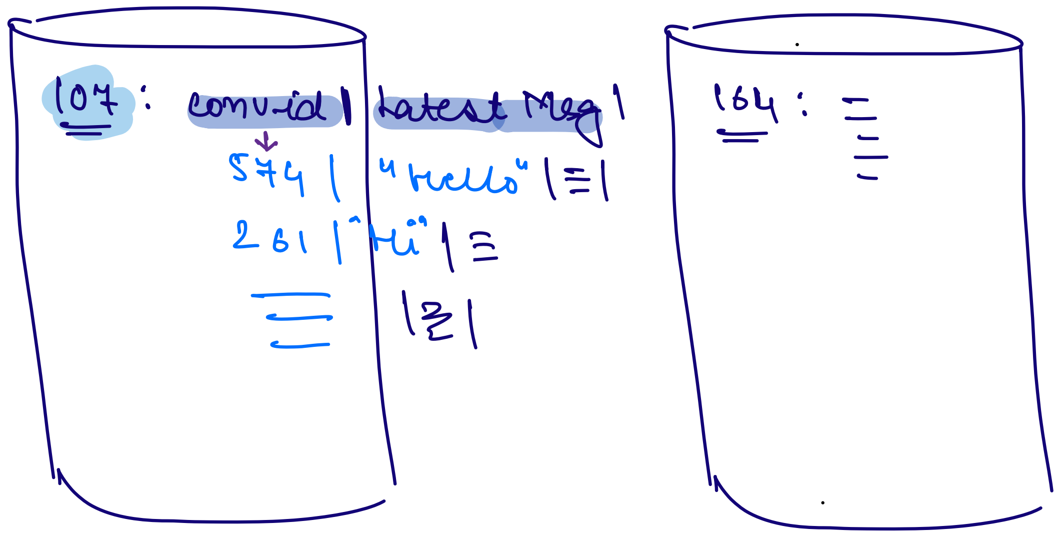
→ SHARDING KEY → userid

1) getConversations (userid)

userId.

intra shard query.

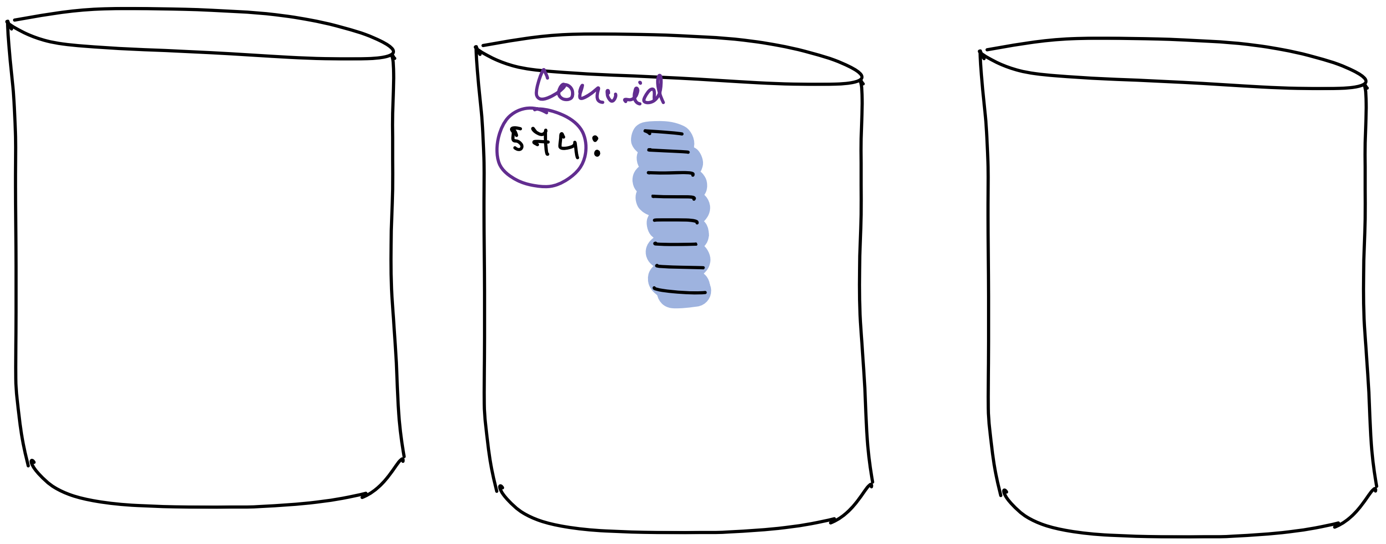




2) getMessages (conv_id)

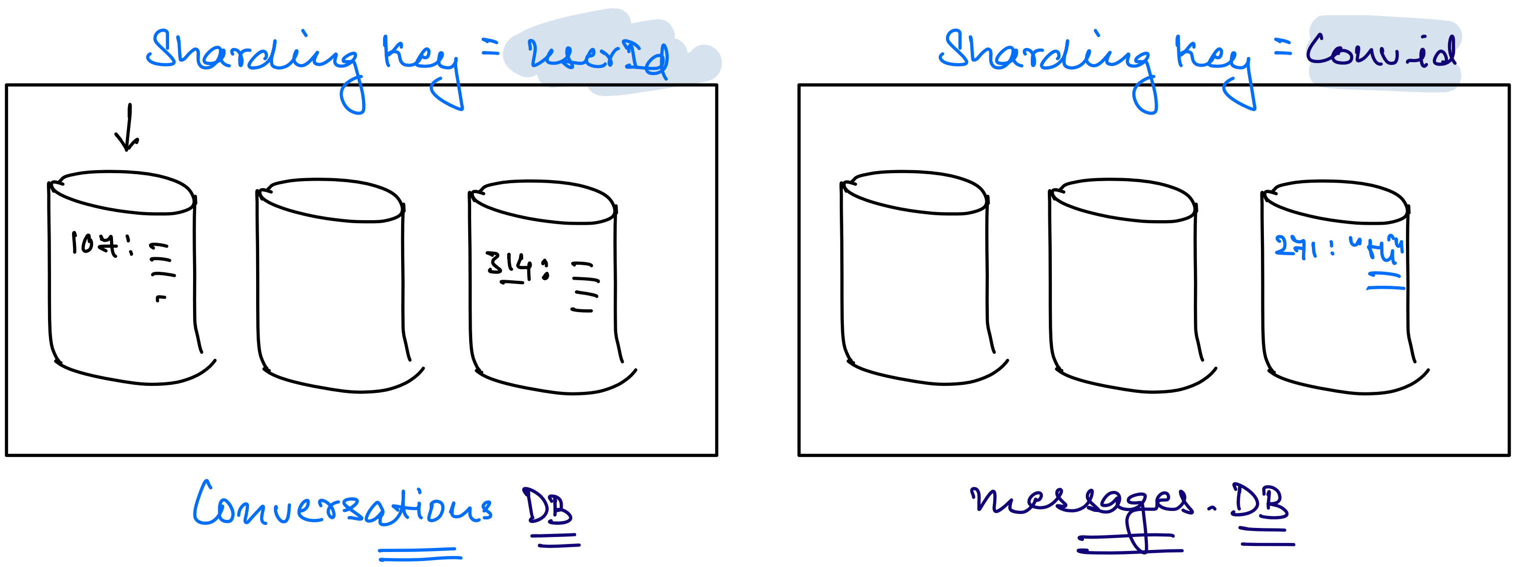
$\hookrightarrow$ Sharding key : conv_id. = (Chat_id)

$\hookrightarrow$ intra shard query.



Conversation (Chat) < 1:1 / Group.

⇒ 2 Databases



Write

↳ sendMsg(s-id, r-id, conv-id, . . . - -)

I) Msgs DB

↳ Sharding key ⇒ Conv-id } 1 Shard.

II) Conversations DB.

↳ Sharding key ⇒ userId. } 2 Shards

Read

↳ getMsg(conv-id)

↳ msgs DB : 1 Shard query.