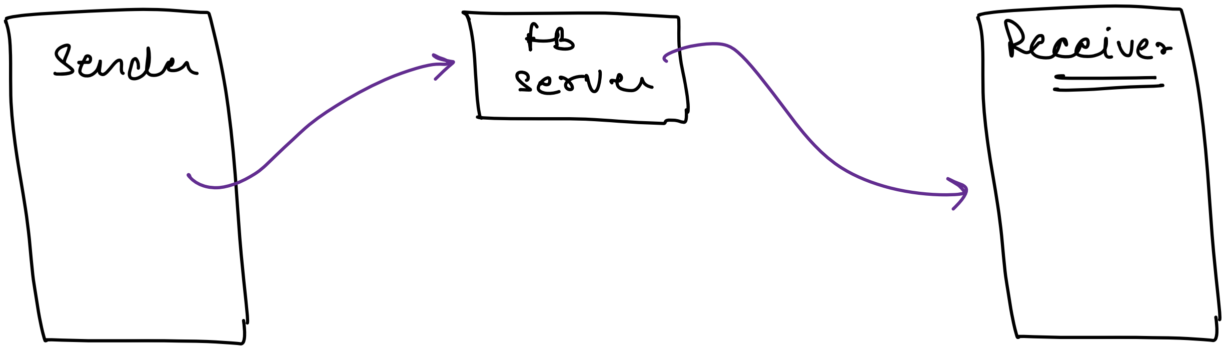
Designing Messaging Application.

- FB Messenger
- WhatsApp
- LinkedIn
- Instagram.
- Game
-
-
-



System Design case study.

1) MVP (Minimum Viable Product).

2) Back of the Envelope calculation (Scale Estimation).

   ↳ if read QPS > (50-100) write QPS.

   ⇒ Read Heavy.

3) Trade offs.

   ↳ Consistensy vs Availability.

→ Latency.

4) API Design.

5) HLD Deep Dive.

MVP of FB Messenger.

1) Send | Receive a msg. → text
→ Image
→ Emoji
→ Video
→ file.

2) Messages should be delivered in almost realtime. ⇒ low latency.

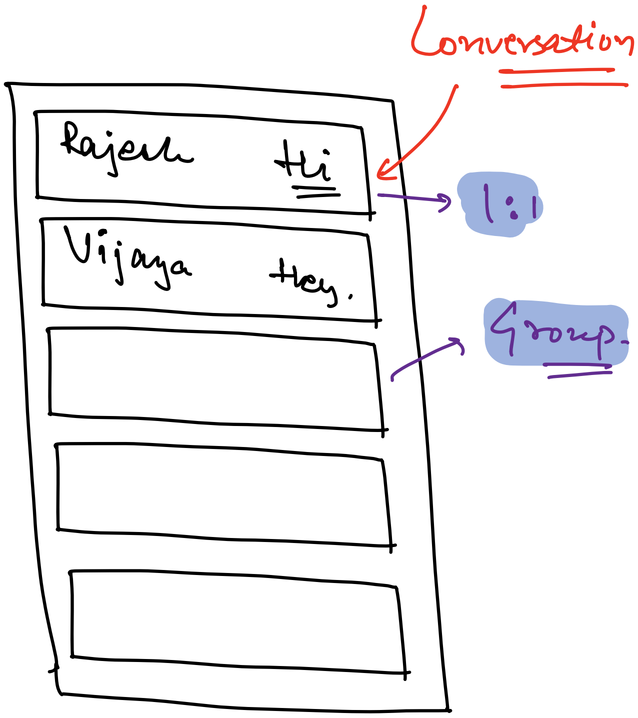
3) Message History.

4) Multiple conversations.

5) 1:1 Conversation vs Group Conversation.

6) Delete message.

7) Read Receipt.



8) Online | offline

Scale Estimation.

# of users on FB = 3B.

DAU = 1B.

Avg messages per user per day = 20

Total messages per day = 20× 1B

= 20B.

Write.

$$\text{Write QPs} = \frac{20B}{\cancel{86400}} \rightarrow 100{,}000$$

86400 → 24×60×60

$$= \frac{20 \times 10^9}{10^5} = 20 \times 10^4$$

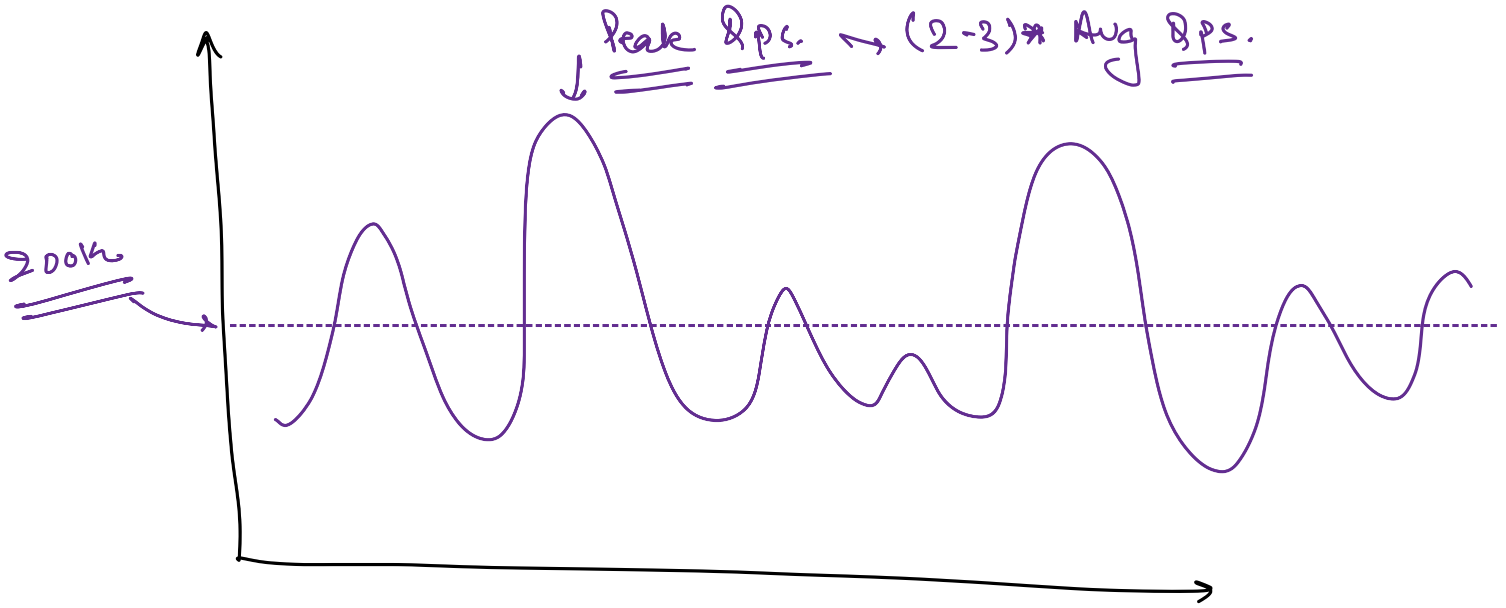
= 200K qps.

Read Qps ≈ Write Qps.

⇒ Read Heavy (or) Write Heavy

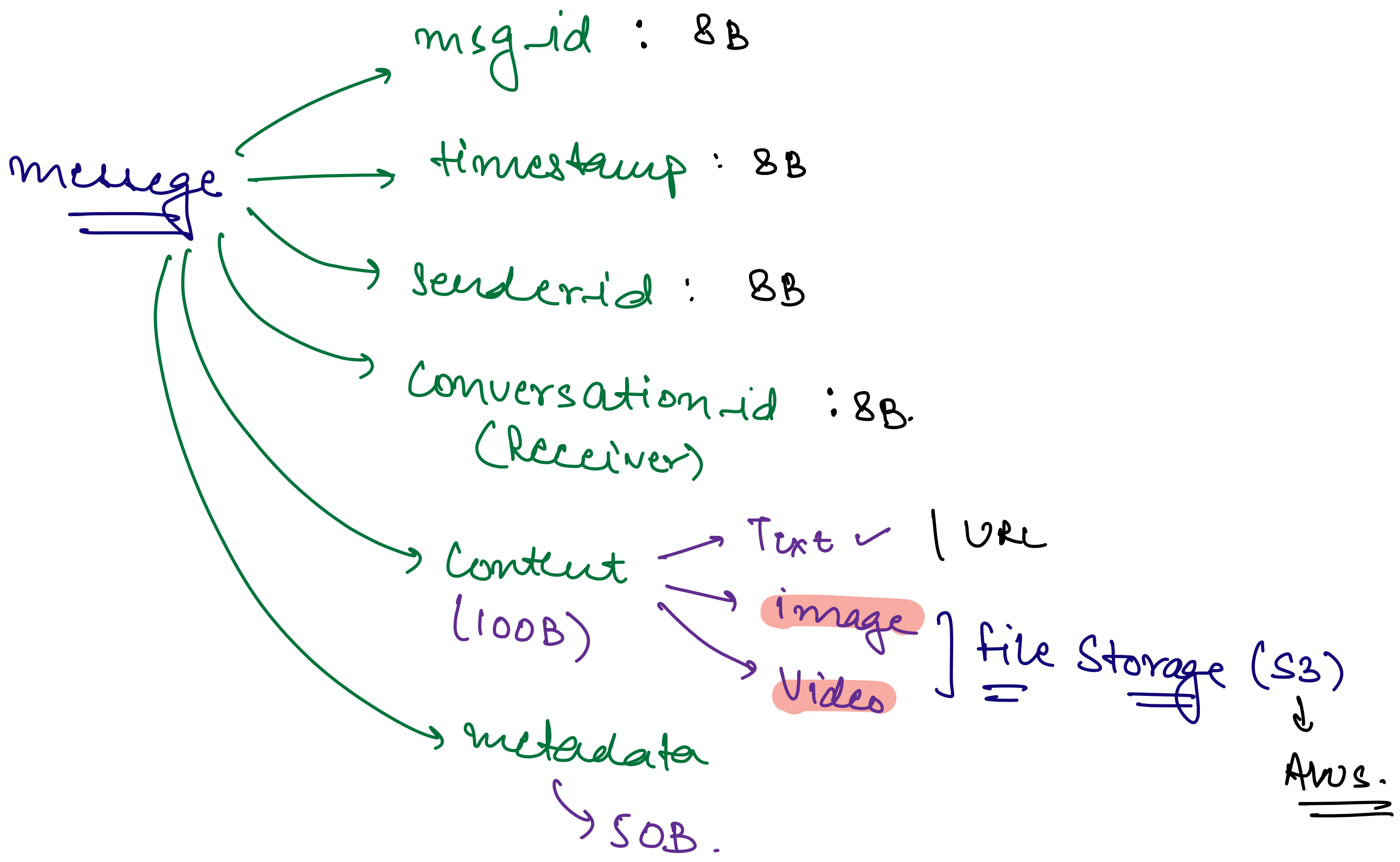
⇓

Both.


Peak Qps. → (2-3)* Avg Qps.

Storage Calculation

↳ 20B messages | Day.



message

msg_id : 8B

timestamp : 8B

senderid : 8B

Conversation_id : 8B.
(Receiver)

Content
(100B)

Text ✓ | URL

image

Video

file Storage (S3)
↓
AWS.

metadata
↳ 50B.

≈ 200B per message

20B × 200 Bytes per Day.

$4 \times 10^3 \times 10^9$ Bytes.

↓

GB.

4000 GB

≈ 4TB.

10 Yrs : $4TB \times 3\cancel{6}5 \times 10$.

400

16×10^3 TB.

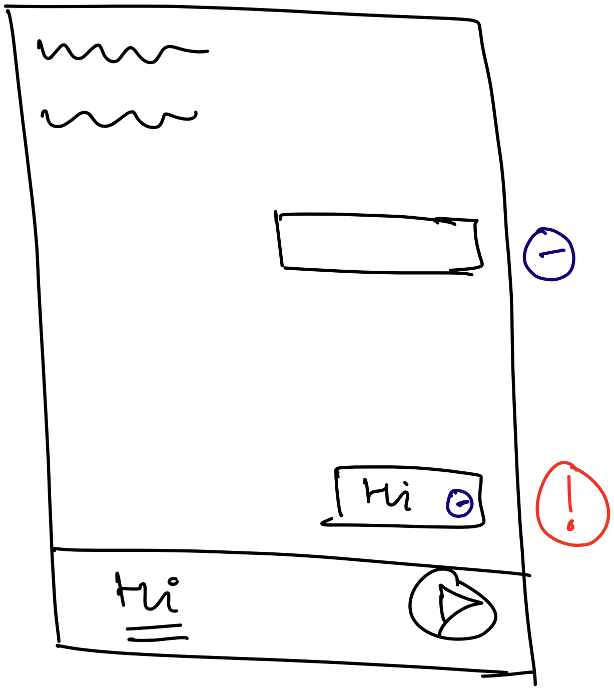
$\boxed{16 PB}$

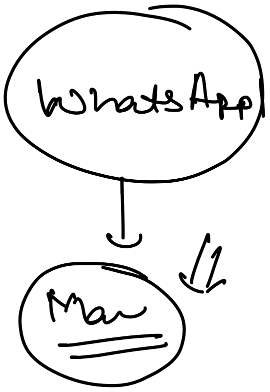
⇒ SHARDING. ✓

Trade Off.

High Consistency >>> Higher Availability.

⇒ Super low latency.





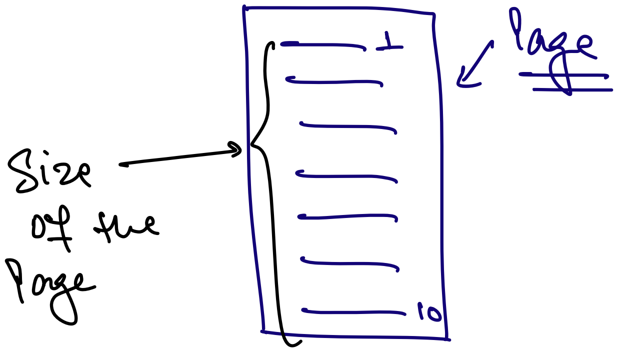
API's.

1) sendMessage (sender_id , conv_id , content , msg_id)

2) getConversations (user_id , limit, offset)

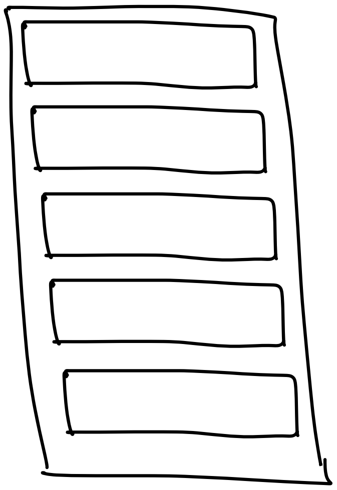
Pagination

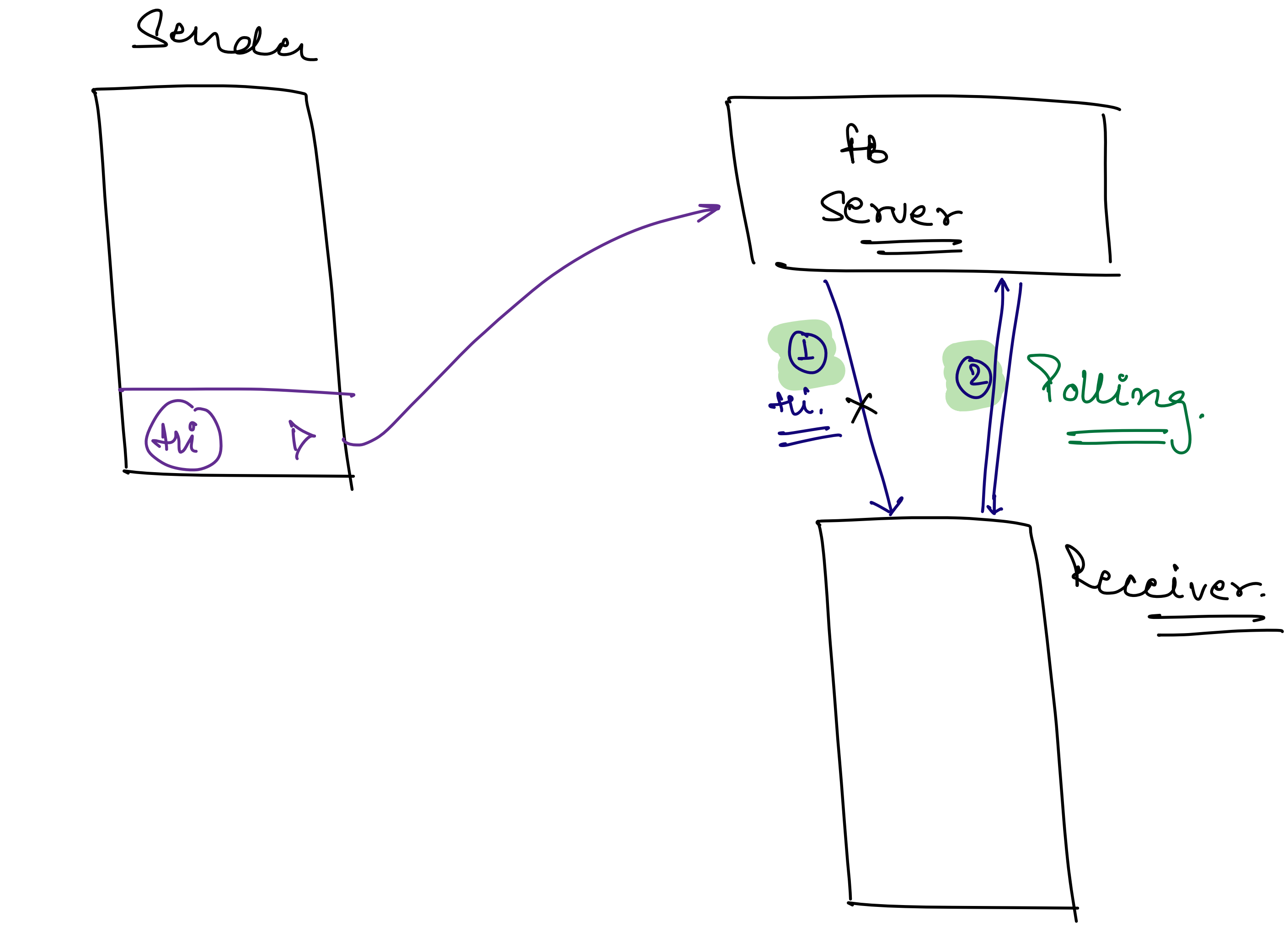
(Lazy loading)



3) getMessages(user_id, conv_id)

Paginated.





Receiver needs to fetch messages from the backend.

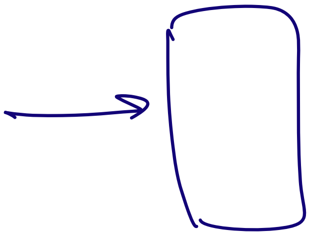
1) Polling

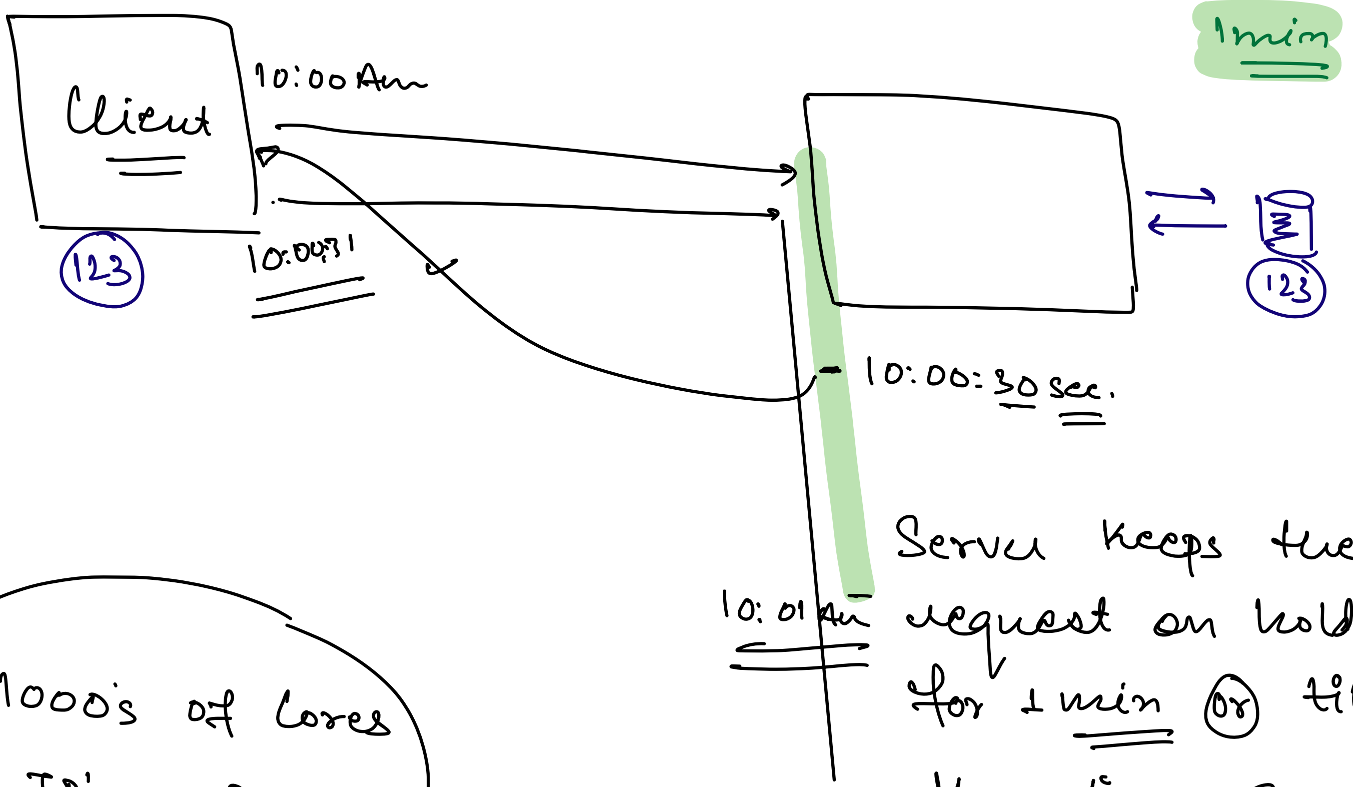
⇒ Client keep on making a call to backend every 1 sec to check if there's any message for them.

⇒ lot of calls to the server.

⇒ N/w Congestion.

2) Long Polling



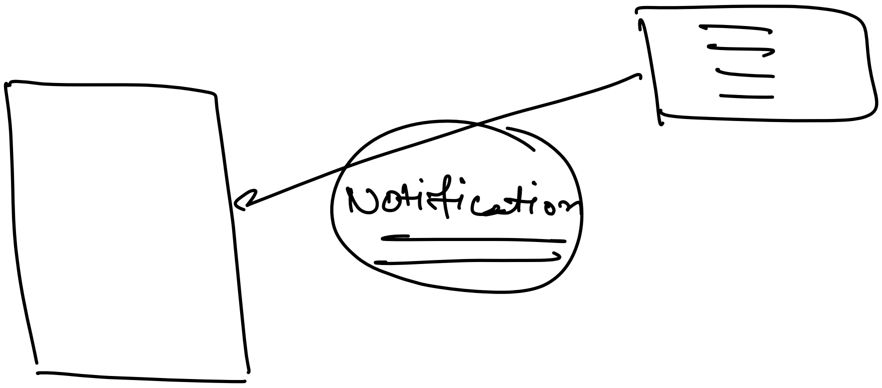


Server keeps the request on hold for 1 min (or) till the time server receives a msg for this client.

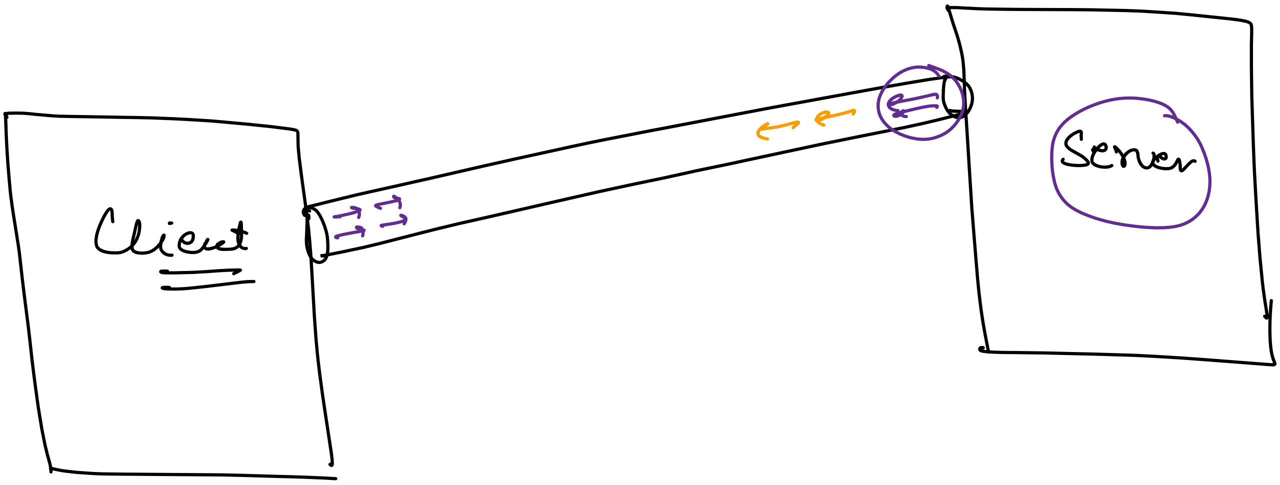
⇒ 1000's of Cores

⇒ TB's of RAM

⇒ for active conversations, this is as good as Normal polling.



3) Web Socket.



⇒ Lot of Resources

Millions of Connections.

⇒ SHARDING KEYS

⇒ Idempotency.