

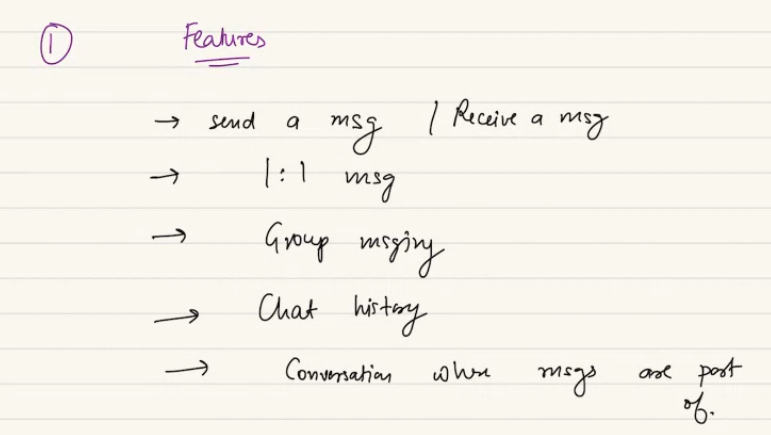
To build such systems, we will do 4 step process to design..

1. Build MVP – set of features should be there in 1st ver of product
2. Estimate of scale.: read heavy or write ehay or both.. we need sharding or single machine will enough.
3. Design trade off.. design choice.
4. Actual system design deep dive. Look at API, design the flow. Data storage.

**1: MVP: what are important feature in the messenger system:**

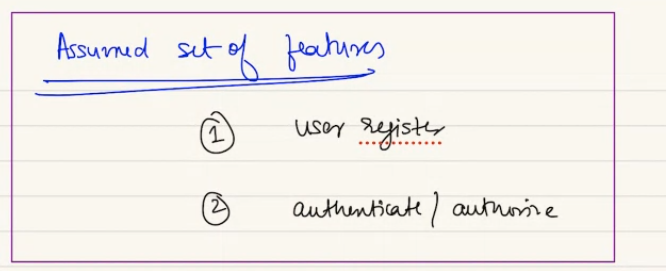
Receive message, send message. 1-1 messaging.. also group messaging.. chat history—should see historical messages also.. these messages are clubbed together in as per conversation. A user, his name. open the conversation..

Each of the messages esentatially need to be a part of conversation, where messages are part of.

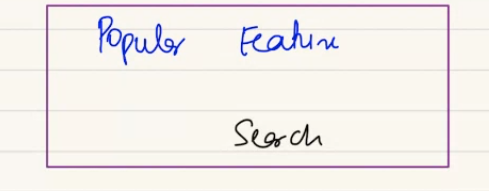


Assume set of features:

1. User registration.
2. When you join you should be able to authenticate and authorize yourself..common in every system we build

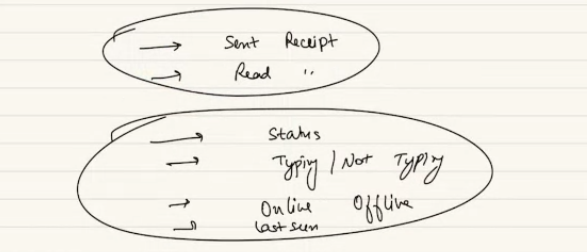


A popular feature we see in lot of service, search feature… to search old messages. In uber search prev rides. Data you accumulate and let user search… we will have a special class on search..



Sent receipt: if messge is sent. Whatsapp tick.

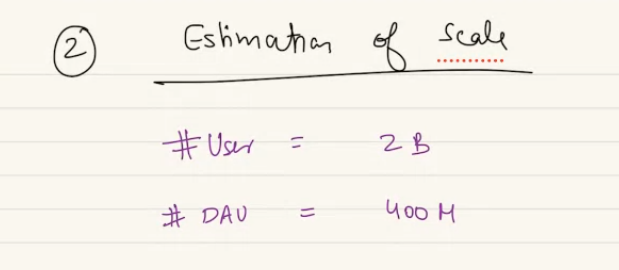
Also read repeipt: double tick. Status: if user is online or away. User is typing or not



Another common feature: sending notification to user. As a service send notification, even app is not on. A push notification.. not part of curriculam.

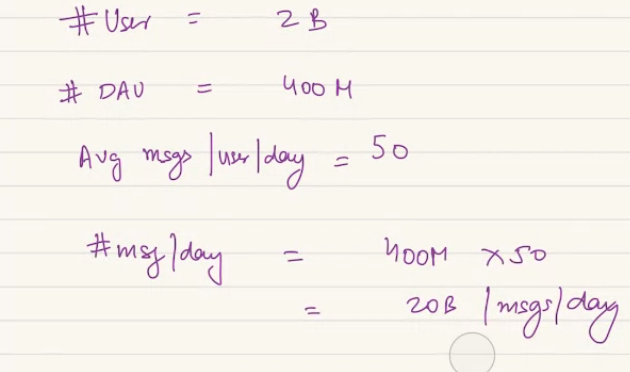
**Part -2 : SCALE**

Facebook has 2B. ¼ of them will be online daily.. 300-500 active user per day. 20%



Active users can send what no of avg message a day..

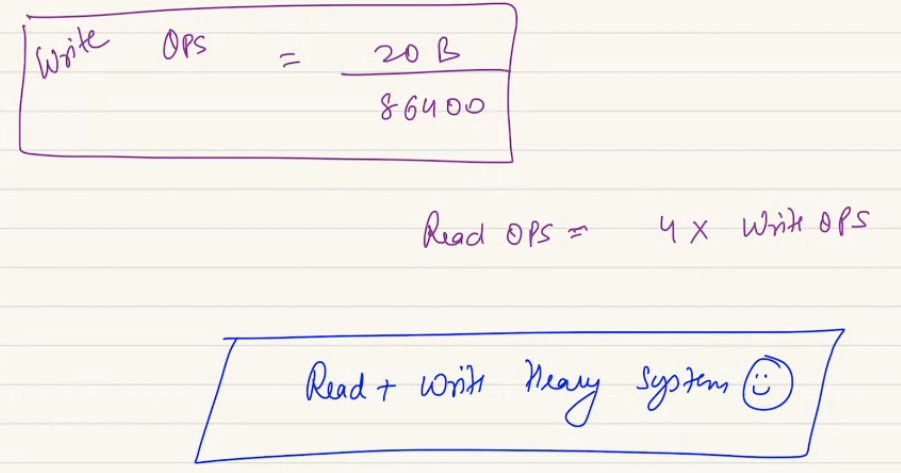
Avg no of message send per user per day: 50



QPS = queries per second = 20B/ no of sec per day

Read QPS will be 4 times write QPS.. everytime I send a message that’s write.. but a message can be seen by 4 times a day. Again and again..

In group 1 message can be read by 10-50-60 people..

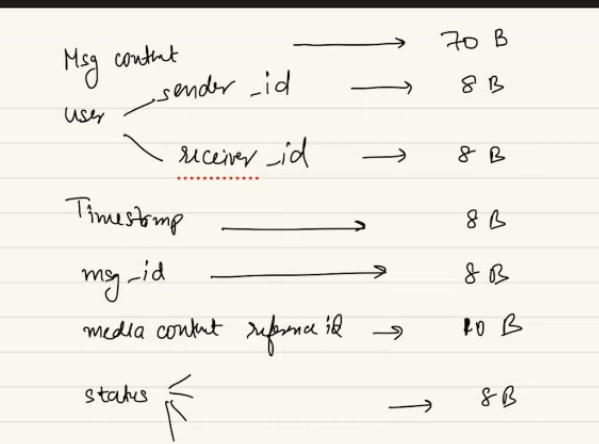


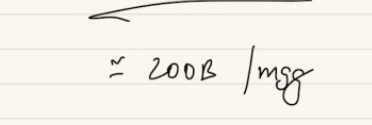
Read is 4 time than write so its read+ write heavy system as it just 4 times..

Write QPS is multipe by 4 or 20 or 50 even than its both.. not until its few hundred.

AMOUNT of Storage:

To store a message we store the message content, who sent it, sender user id. User id of receiver. Time stamp. Store the message id.





We store image/ video in blog storage.. we can store ref in main DB.

Also status of message. Can be multiple flags.. read, not read, delivered, not delivered..

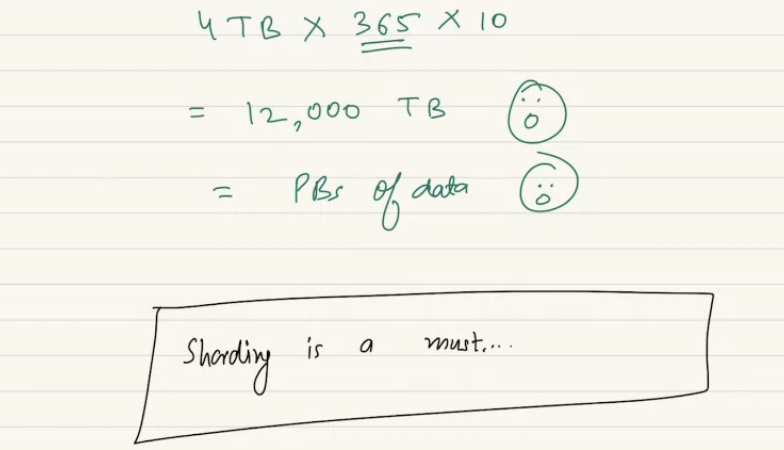
Avg character per message = 50 char.

Media data is stored in objected store, file store… in normal database there will be a table where we store metadata of media content. This message has a media content, the ref id should be added in main DB. Some space in main db for ref of media. We keep id of that blob storage location in my db. Kind of metadata.

For message we don’t use CDN. For media ontent we can use CDN.

4TB messgaes per day..

Main storage we keep data for 10 years then we move to archive.

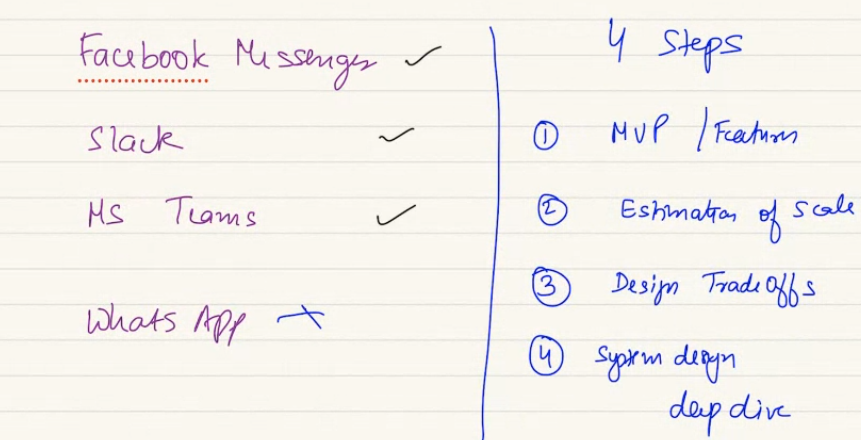


12 Peta byte

We cant store these 12 PTB in single machine. So sharding is must.

In a single machine how big data can store? 4TB..

If we make it 5 years it will be half storage..



**Whatsapp doesn’t store on own. They work as postman. Client side keeps and whatsapp delete the message from their side. To save petabytes of data.**

They decided they they will not store. They save billion of dollars every single year.

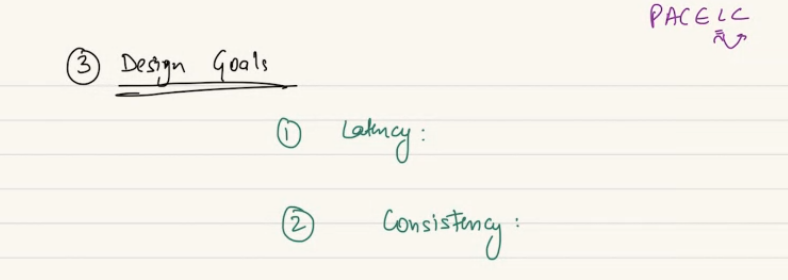
Facebook messanger, they invested billion of dollar in storage.

FB build the messanger costs billion. Not a big addition to arsenal.

**DESIGN GOAL:**

paceLC theorm: superlow latency 🡪 super high consistency.

Message delivery system needs consistency.



Not immediately consistant. But eventuall consistant - Means some amount of time it says Deepak got messages but after it says depeak didn’t got message. Not good experience

I want message to send in consistant manner. Not diff version of reality. So system like messenger should highly consistent. For this availability should be compromised. We coul not deliver the message. That’s fine.

I cant give guarentee message is delivered immediately. So latency will be there. I want reasonabily low latency. Not super low latency.

Has to be highly consistent: distribute system can be fault tolerant, when highly consistant, compromised in availability, “please retry, ” to read, please reload messages..

When 1-1 message: expectation should be super high consistence.

When in group consistency can be compromised… I can relux the consistency behaviour in group.

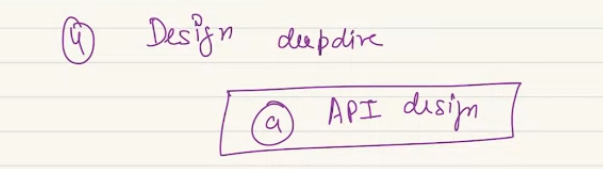
In 1-1 ideally I want consistency.. in Slack when I send message.. I want consistant. But when phoen is off and message is not delivered. And I get notification message not delivered. Event could not happen.

But when I am told message sent. Means should be sent..

Gaming chatting feature that chat no need to be super consistency.. depends on use case…

1-1 me I would want consistency over availability..

**Forth: Design deep Dive:**

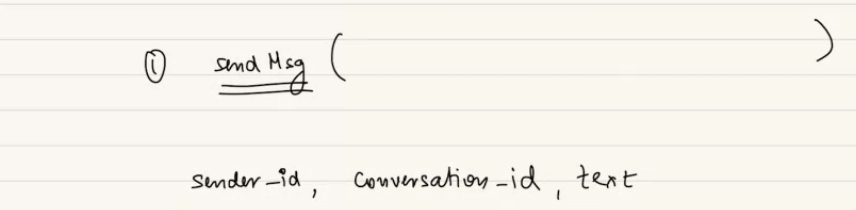


What all API should be there?

1. Should be able to send a message: sender user id,

receiver id – 1-1 me proper id, one to many me conversion id/ group id.. conv id is given to a conversation in group.. here I add conv id rather than user id.. a conversation can be 1-1 or group me type of conv wil be grup. And id will eb list of ids. API will have sender and conversation id.. rather than having a receipent id.

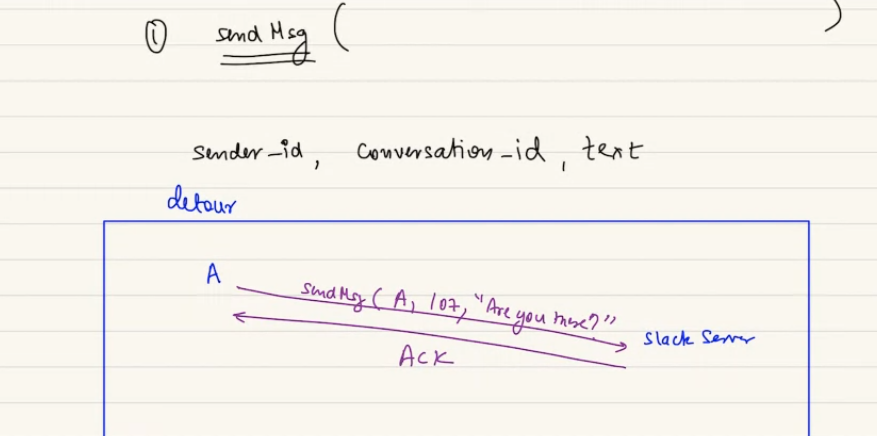
Message text, media content in send message



In conv table 101 con id. It can be 1-1 then 2 participant id. If group: collection of id..

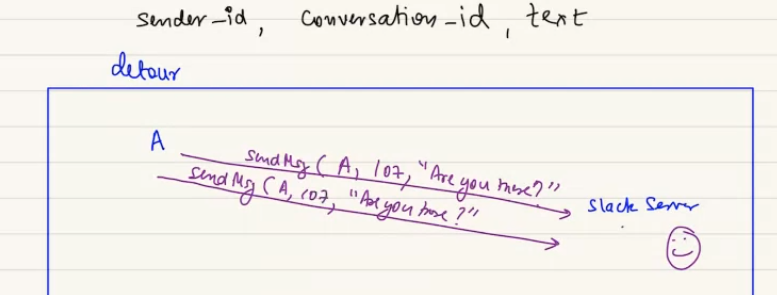
Detour:

If activity doen multiple times, should be registered once.. in mobile a client uses slack. I send a message to anand. Client will send message req to server of slack. Slack got my request, registered it. Network bad and client didn’t get the receipt acknowledgement of HTTP call. If client didn’t get that, client will retry to send. Again API call to send. No way for server to know if it’s a new message or retry to send a message multiple times.



Network kharap, ack is lost.. client think message wasnot sent to slack.. client will retry.. again make same call. Send message to conv id 107. And wantto ask r u der?

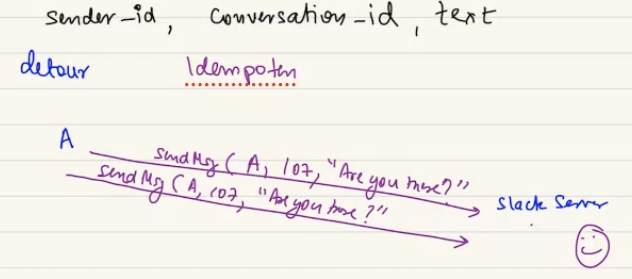
No way server to decide client is retrying or sending a message again and again



Buging or genuine retry,..

Idem potency: so slack knows a new message or same again and again..

In GPAY I need to pay 100 to vivek, if client retry it needs to ensure there is idem potency. There need to be way to be idem potent. We can do by client generated unique request id. Send that as a part of req. so server can understand..

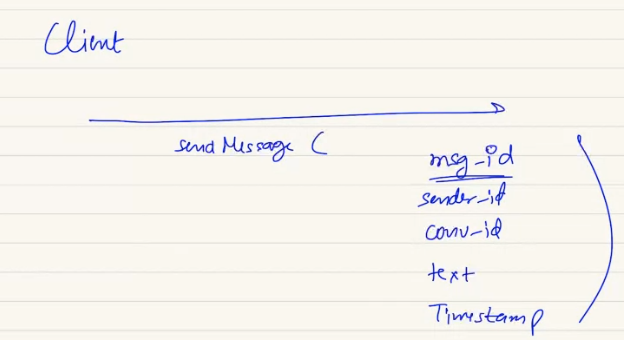


Diff way we can communicate with server, some reqcan be idem potent. Some not.

Every req should have unique id so recipient know its new or same.

Your client wheile going to send message call to back end. Its going to have sender id, conver id, text, timestamp. Also onemore entry is message\_id..

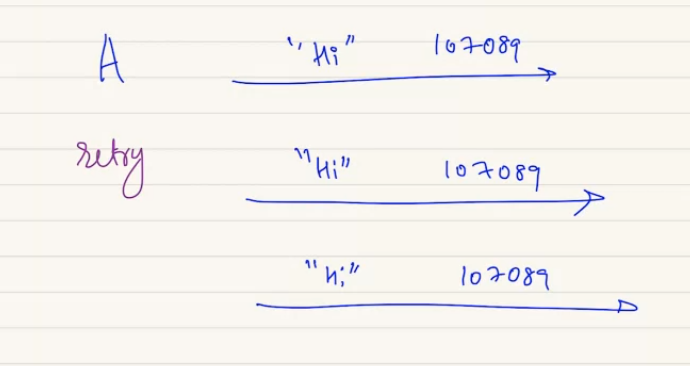
When send a message to backend client should generate a new unique id.



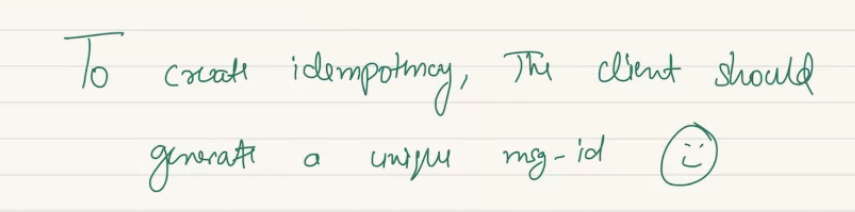
Akshay sends hi to someone generated message id 107089. If he didn’t get ack he will retry..

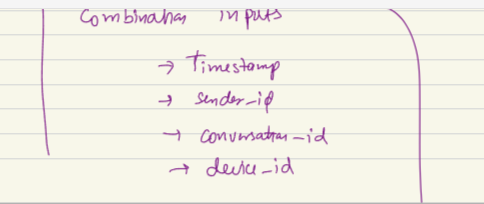
He will retry and the message id will remain same. If he sends 10 times, server will now same message is sending again and again. So server can decide and dnt send the same message 10 times to receiver..

When content is same, receipent is same hi sending again and again means different id. A unique message id.



To create idem potency the client should generate a unique message id.

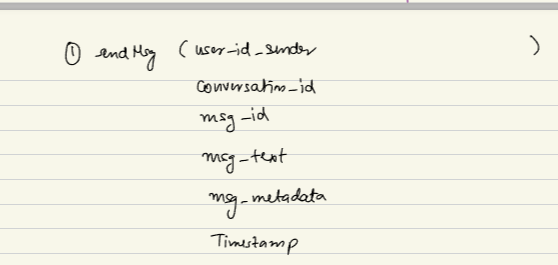




If I use timestamp: it cannt be a good metric.

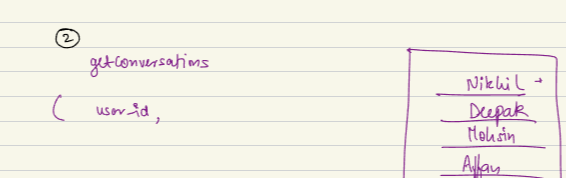
Send same message 2 times to register urgency. Timestamp is bad. Right way is to create a hash, combination of input.

**Read UUId**



If I open whatsapp, 1st screen shows list of conversation.. user or group..

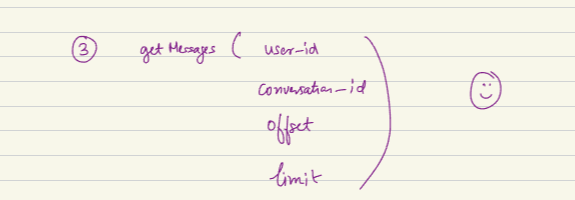
When I click on one of them I see messages… on slack I see messages directly but left side me I see conversation. Slack don’t have 2 page idea..



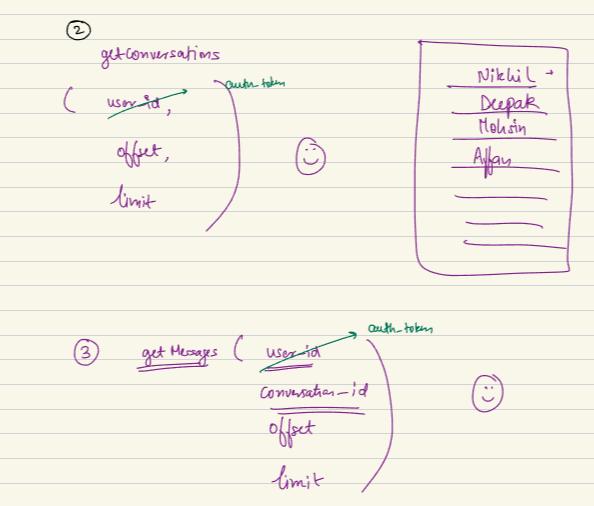
I should have pagination. Done based on offset and limit..

PAI for get conversation will have suer id, offset, limit.

Another Api will be getmessage (suer id, conv id, I cant load all conv at same time, (I will have to do pagination so offset and limit.)

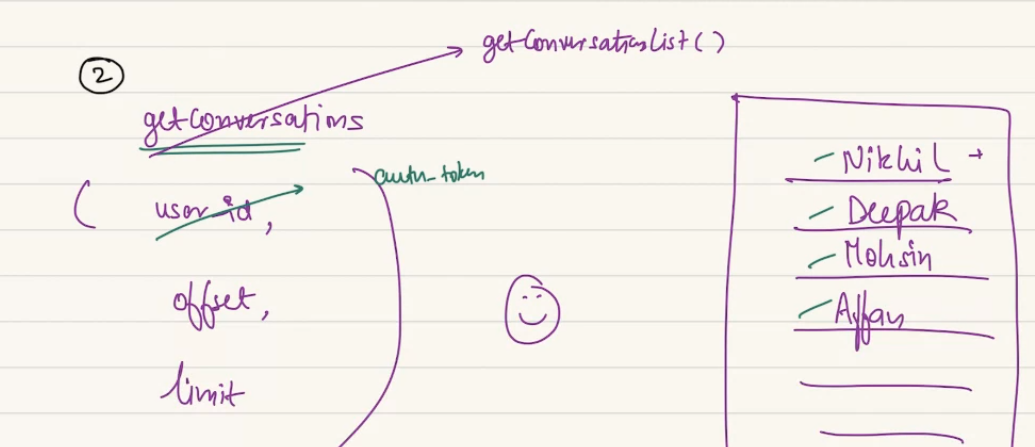


We pass user id: who is the person trying to see their messages. Or we can say it authentication token.. you pass auth token everytime u read. We can say user\_id = auth\_token. You tell server I am authenticated for my request.



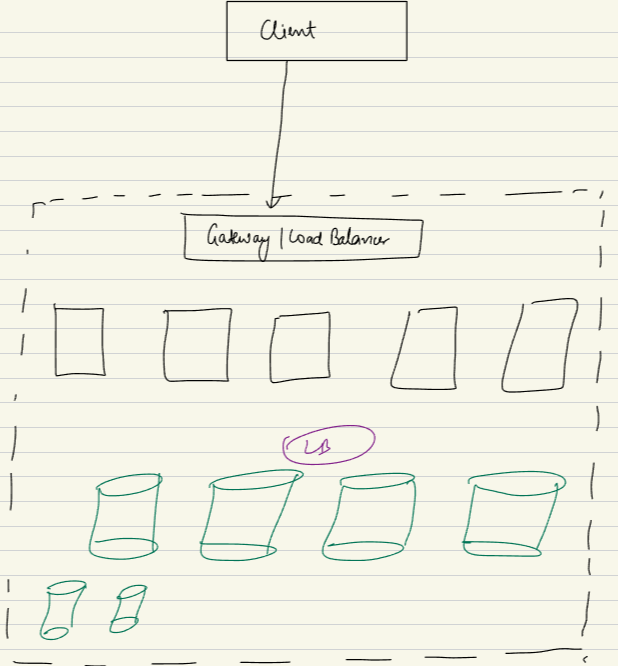
**why we do not need conversation id for getConversation?**

In get conv tells you all active threads you have… once u click on any threads, you know which conv id.. we can say it get conv list



do we use kafka as you mentioned topic haha

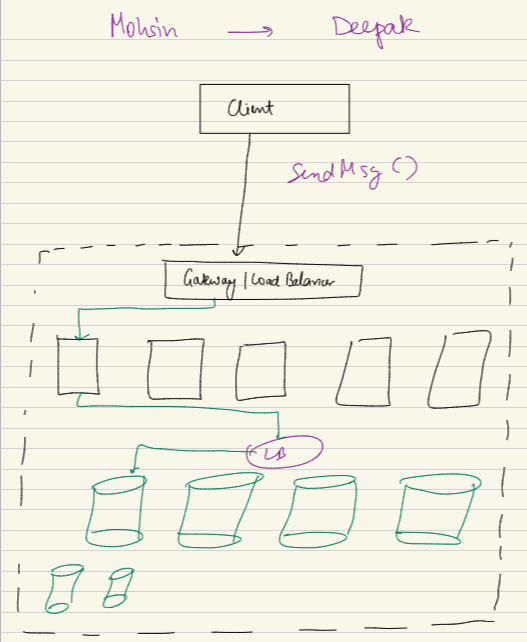
federation of database: read federation idea in his old class… single or multiple type of DB.



When I call a sendMessage Call.. this req will go to one of the APP server… we will decide stateless or full… will decide…

This message needs to be registered in somewhere.. measn needs to be saved in backend. Will go to some DB (one or many).. then write the message…

what will be the sharding key here? Will do later..



There needs to be a way for receipent to know that they have a message. There are 2 ways. If Deepak is actively to my system.. if not active, app is closed, app not sending constant traffic. Then I will send a mobile notification that you have a new message..

If my slack app open, animesh sends a message, Slack will send a new message, I will know.

If slack is off. As a client I am not connected right now. Then app wont be able to show the message, there will be a notification ios, android.. notification at top so I iknow and click on it..

Or there might be a email notification. Slack, or MS teams sends..

subscribe and publish: seems kafka best use case

**first question: what kind of sharding key will decide:**

what should be sharding key. What all we need to shard.. messages- they are part of conversation

I can probably based on conversation id. It means in my system every conversation will have unique con id. A 1-1 con can have id 2070.. [Mohsin--deepak] another can be 3010 [akash-ritik]

4079 – [Akshay - ravi]. We can have group conversation if

9090 [m1, m2, m3, …………… m10] = a group conversation..

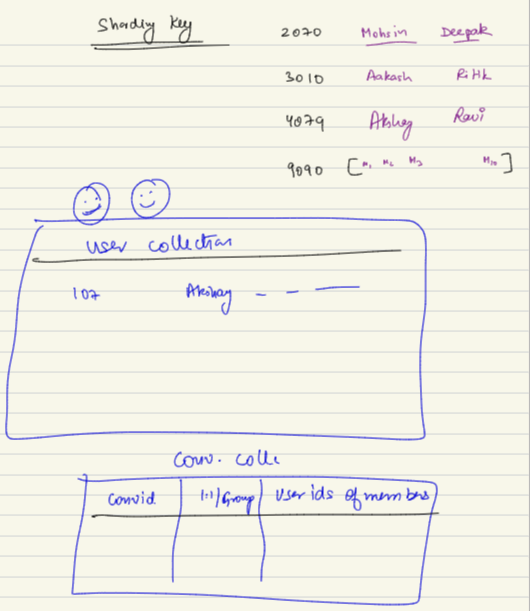
Each of the name wil become ID. There needs to be a collection users. Where I wil have user id with name and details

Also a conversation collection: conv\_id, 1-1 or grup, member id{2 or more},..

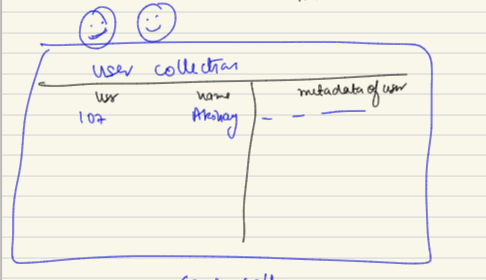
Then every conv will have a message list as well.

A message will be part of a conversation.. mes wil have metadata, timestamp.. these are collection I will have. Can be SQL, noSQl or can be collection of diff DBs.

Seen, delivered etc will be metadata.

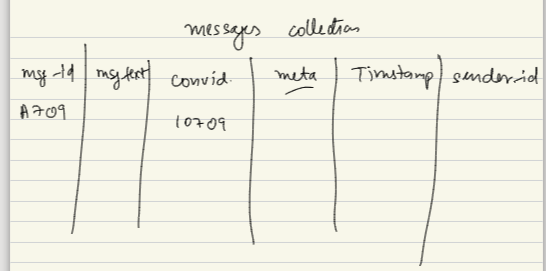


User status: busy, away will eb stored in user collection.. Link of photo of user will be – user collection as attribute of user..



Conversation id between Akshay-avishek will be set in conversation collection..

Mesasage is stored in message collection.

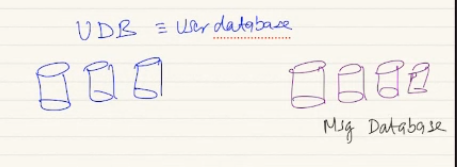


Each of these message should be devided in diff shard. We cant have entire message, user, conversation in same shard.

We have to determine which id will be sharding key….

User collection should be decoupled from all of them. User will be federated..

In my DB layer I will have different UDB = user data base… tehn I will have different DB for conversation and messages combined. I will call it as Message database.

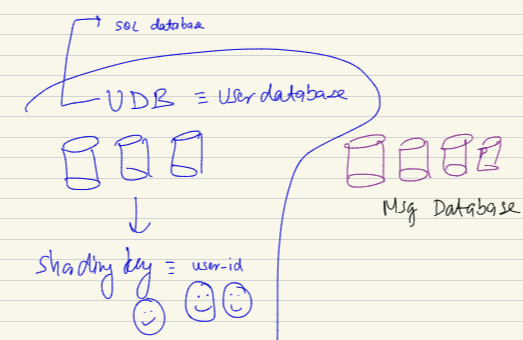


User collection sitting in UDB.

Message collection and conversation collection siting in MSG DB.

For user DB a good sharding key will be user\_id..

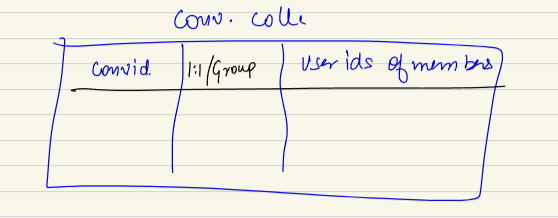
**Since, this has a fixed schema we use SQL or NoSQL database**? For UDB we can Sue SQL..



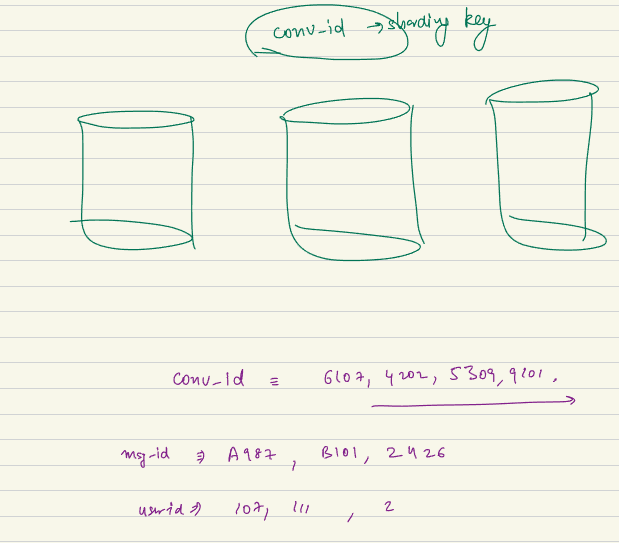
We federated as user is different from conversation or message.

For mesg db, we can explore sharding key…

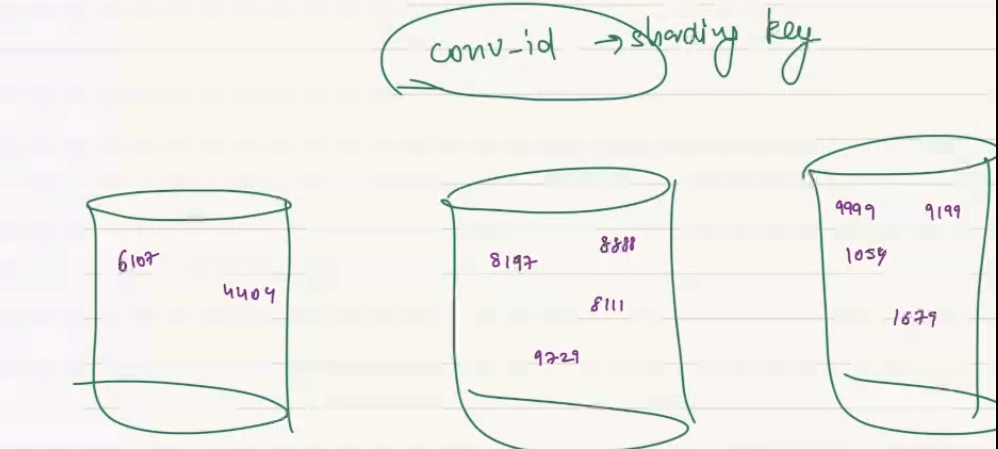
1. Conv id:



In msg db if we do sharding based on conv id… if any conv grows very large it may pose a problem.. no problem..not a concern. A conver cant be too big to stay in a machine.



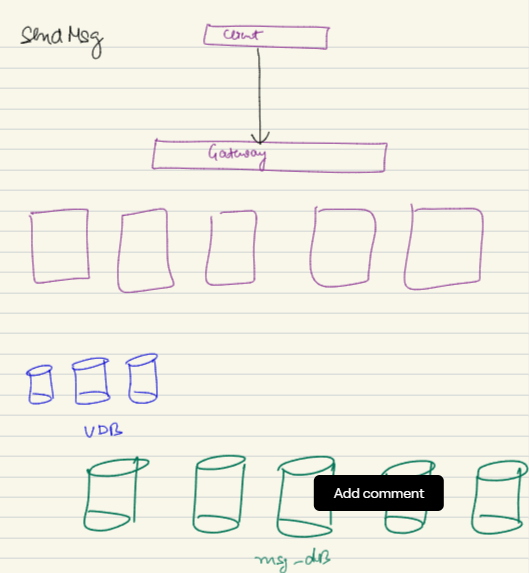
Conv are sharding key measn some conv can sit one machine some in diff machine:



All message will be sit in that message.. all message based on con sit on one machine.

Conv for message db and user id for UDB.

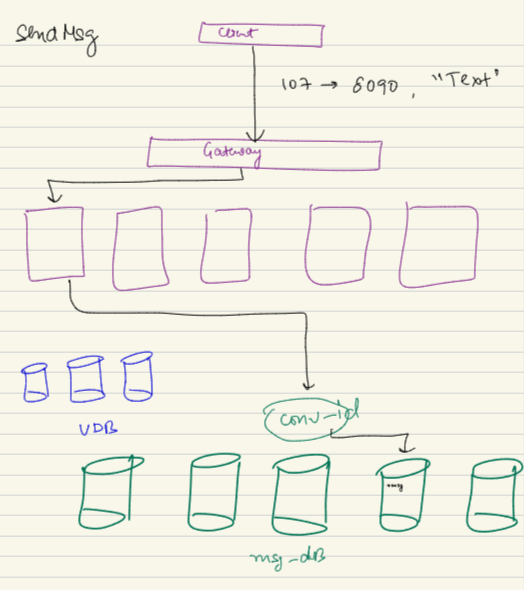
A global cache will have popular ones [few million] will be called again and again. We will ahev that mapping in cache. So we can save time and do go db again and again. Caching teast wise.



When I send a message will go to one machine.

Based on conversation I will go to one machine if conv\_id is my sharding key… I write the message there . inmessage collection I will write a message.

Again I will go to conv collection and rearrange the list of conversation.



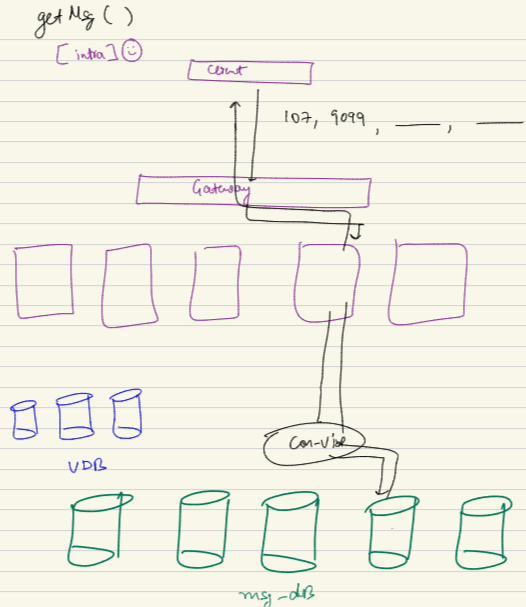
Its intra shard: go to one shard.

If 107 user wants to send message toa group 9099…

Irrespecitive of user or grup it will go to 1 shard.. where con is stored.

Second kind of query is getmessage, also for a particular conversation. U want to get a message as a person for a conversation. Getting message froma particular conversation..

107 user, 9099 is is converataion. Base don conv id I will go to one machine, read all message base don offset and limit. Getmessage is a intra shard..



As a user I need to know which are the convertaion I am recently a part of..

I ahev to go to every single shard.. so cant be handled in present .

As it will be a all shard query..

To figure out all the converasaion a user is part of. If conv and message are split based on con-id. I will have to go to every single shard. I will have to merge and present to user..

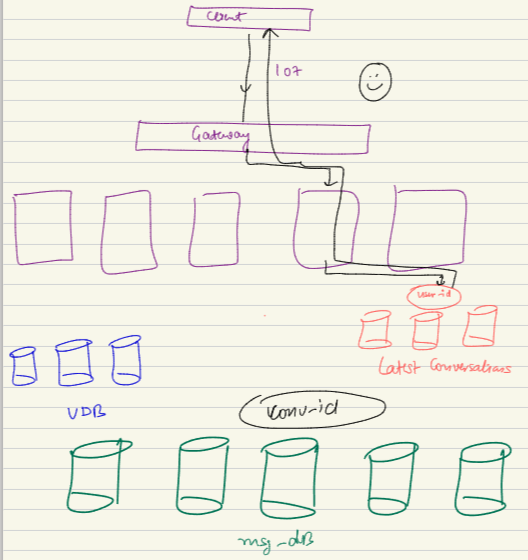
We can think ina caching in FB news feed…

Another is have a duplication of storage. Duplicate of storage have a secondary storage.. in there we ahd recent post db, eher we can have recent conv DB sharding key is…….

To support rather than having to dfo to a db, we can think of a federated db/ diff DB… can call it latest conversations..

Sharding key of latest conv ..

When a user comes



I have to polpulate in send message also..

can we use UDB for latest\_conv? You can… do that.

Or we can have a latest conv DB..

A diff db for latest conv..

Wide column family store: read about it.

In send message , a user tgo to dwy…

ll