2.4 whatsapp system design

**Requirement :**

**How** do we design a chat application something like whatsapp or facebook messenger.

Lets look at some functional and non functional requirement that this platform need to support

**FR**

A white paper with writing on it

Description automatically generated with low confidence

Will support one to one chat wherein a person can send message to another person or user

Whenever ay person send message in group that message will received by everyone in group

The message could contain text,images and videos

Read receipt : single tick when message is sent and double click when message is delivered

To the user and bule tick when messages is seen by the user.

We will also show last seen time of particular user

**NFR**

A whiteboard with writing on it

Description automatically generated with low confidence

**Low latency ( less transfer time)**

Lower latency refers **to a minimal delay in the processing of computer data over a network connection**

**It** should have very low latency because for chat application it should looks very real

**While we**sending a message the other person should immediately to see it.

It should have very high availability. The system should not go time what happen.

There should not be any lag (delay)like I m typing something to u and then u respond to it

**Lag is the result of high ping or high latency** – which essentially means your internet connection is at a lower speed or quality than it should be. The main giveaway for lag is that it causes delay in your gameplay, with your commands not being acted on by the game immediately.

Coming to some scale number these are approximate number of whatsapp

2 billion users

65 billions messages

**Overall architecture of whole system**

**What is cassandra**

Cassandra is an open-source NoSQL distributed database that **manages large amounts of data across commodity servers**. It is a decentralized, scalable storage system designed to handle vast volumes of data across multiple commodity servers, providing high availability without a single point of failure

**When should I use Cassandra DB?**

Cassandra is suitable for **applications that can't afford to lose data, even when an entire data center goes down**. There are no single points of failure. There are no network bottlenecks. Every node in the cluster is identical.

Diagram

Description automatically generated

Diagram

Description automatically generated

Will first start with chatting solution then we will look at some small pieces that surrounds it

Let start with user flow

**U1,u2 :**--- user devices could be mobile phone, web browser

Let say user1 want to send message to user 2

**What is WebSockets used for?**

The WebSocket API is an advanced technology that makes it possible to **open a two-way interactive communication session between the user's browser and a server**. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply

**So let the flow goes like this**

**User1**want to send message **user2**with message id m1. Now what happens user 1 talking to

Something called a web socket handler 1. there will be a lot of websocket handler.

**A websocket handler is a server on our backend which is keeping open connection with all the active users.** These are live user who are having an active internet connection king of things.All of these users are connected to some of the machines. There will be  a lot of machines.And all of these would be distributed across the globe. So that people in certain geography talk to the server which are nearer to them so to have lesser latency.now let say user 1 is connected to web socket handler 1 saying send message m1 to user u2.

**Now this web socket handler 1 does 2 or 3 things when happens**

1:- it talk to something called **web socket manager.** web socket manager is a repository of the fact that which device is connected to what all user. Because these are probably thousand of devices we need to keep that information at some central place  which is web socket manager service. It sit on top of **redis**which store two info mainly. First of all it stores that user 1 is connected to web socket handler 1 and web socket 1 is connected to user 1 , user 2 ,user 10 etc. So a list of user id that a particular machine is connected to that info get stored in redis.

Now let say connection break between a user and a machine and now user is connected to diff machine so that info is updated through web socket manager into this redis.

2:- second thing web socker handler 1 does is talk to something called **message service.**

Message service is basically the repository of all the messages in the system. It will have api to get message by message id and get the message by user id and various filter.it sit on top of **cassandra d/b. Why cassandra** because again we have billion of users will send million of messages . Cassandra is fairly good data store to store that volume

There are t**wo kind of message service(platform**) that we can build.

One is something like facebook which stores all of your message so in that everything stored in cassandra permanently. And there is something like whatsapp which stored the message only till the time it is not delivered to the user and once it get a delivery acknowledgement  then it delete the message. So if we want to build a whatsapp kind of system we could delete the message from here also. Deletes are bit inefficient in casandara so if we build that kind of system we might choose  some other data store as well. But for now cassandra work fairly fine,

Coming back user1 send message to web socket handler 1 which will talk to web socket manager and message service. Message service will save message into cassandra and resturn message id . Let say it is m1. Web socket manager will tell web socket handler 1 saying user u2 is connected to web socket handler 2. **now web socket handler 1 will talk to web socket handler 2 saying that I have got a message for u2  go and send the message**

Now there are 2 or 3 possibility

Web socket handler 2 not connected to the user then it will not do anything and flow ends here.

Now web socket 2 send message to user u2 but user may be not  using the app actively but the message got delivered so this device will send a message to websocket handler 2 saying that I have received the message or  other option u2 open the chat and read the message  so that device would send a message and saying I have received and seen the message m1 to web socket handler 2. now web socket handler 2 again queries websocket manager  saying that this message either delivered or read. I need to inform u1 so tell me which machine is talking to user u1. It will get this particular machine ip saying this guy is now handling user u1 and it will send the message to web socket handler 1 saying this particular message id m1 has now been read or received and go and inform the user and this will be informed to the user.

Now how are these connection happening (user to websocket) these are web socket connection which is basically a bidirectional connection so there is no client and server here any party can send a message so that how even server is able to send a message to the client. These are built on top of tcp. Now in this whole process we queried a websocket manager a lot of times.

Now at the same time if the user send multiple message again we will try to do that.

**Let try to cache this information somewhere**. So each machine(web socket handler) will cache two kind of things. One it will cache is  : the list of users that are connected to itself. It would have data in memory it would never have to call this manager. The other thing is it would temporarily for a very short duration of time may be 20 or 30 second cache the information about conversation that it has had recently.

Suppose u1 send message to u3 and u3 is offline and when u3 come online  websocket handler 3 query to message service  and ask are there any message for u3 that are not in  received and read status. This message service will response back and u3 will see all the messages. Now these machine will talk to each other.

A**race condition** occurs when two or more threads can access shared data and they try to change it at the same time

**There can be race condition Example** :

When u1 was sending message for u3  assume at the same exact time u3 came online. We said these are parallel calls wherein a person is figuring out the machine which is being used for user u3 and storing the message. Let say the very first thing happened was web socket manager return that there is no machine talking to user u3 , the next thing was happened user u3 comes online and query message service saying that get me all the messages . The third thing that happens it stored into web socket manager that  this machine 3 is handling user u3 .

The four thing that this message get stored into the message service now web socket handler 3 thinks it fetch all the message for user u3 and displayed it on the device. The websocket handler 1 think that user u3 is offline I have stored the messages and done my work . But fundamentally whatever the message m2(u1) that has not been received by u3. Because before that message was stores this websocket handler 3 retrieved all the messages.  so these type of race condition occur when will do parallel call. So there are multiple ways to handle it.

One solution : websocket handler keep polling server is there any message that I have missed

But that will be happen in bulk call like u1 u2 u100 any message are there which is in sent status.

User send message when device was offline so it will store in local disk like android application so when net come back it will pull all those messages and same flow ill occur

**Now we have next thing as group messages. How does that work**

**Group m**essage almost work in very similar way. Let say another scenario u1  wanted to send a message to group g1 a message called m3  (u1:g1:m3). In case of group behaviour is slightly diff. For group message web socket just send a message to message service saying user u1 want to send a message to group g1 store it and do whatever u need to do so this message service will insert it into Cassandra saying  the message is stored. Along with that it will also do for all the group message it will put all of those message into  **kafka**topic saying  there is a group message I have got from user u1 sending to group g1 a message m3 go send it out.

From kafka there will be something called **group message handler**. This is basically a kafka consumer  which is listening on to the same topic and it is responsible for sending out all the group message. First of all it will do some validation like user u1 is even in the group to send message to g1 assuming all not  part of scope. It basically requires **group service to** get all the users that are into group g1 . It get a list of user then it will remove the user u1 from that list.and now it has list of user that are suppose to get this message. Not it follow the exact same flow that was used for one to one message but all of these user. So it basically calls web socket manager  saying give me all the machines that these user are connected with. It will get a response saying machine 1 is connected to user u3,u4 and u5  machine 2 is connected to some other set of user and all of that. Then it will send the message to individual machine to this user send this message

**Now next thing when people are uploading asset like images or videos**

 The very first thing that happen when user trying to upload  any content is that there is some kind of compression that happens now in real whatsapp there is encryption also that happens

Right at device and then the encrypted message are sent out and when the message are received even the  encrypted message are received and they are decrypted but we will keep that out of scope.we will just think that some compression there.

Let say u3 is sending image to u2  so what will happen is u3 is sending to u2 some image .

**This is basically two step process**

**First** step is upload and image to a server and get an image id  and then send that image id via regular route to user u2  and then u2 can fetch the image  so how the image upload works  again these are  websocket are light weight server they wouldn't have any logic  so very first thing compression of the image over here or video or anything  then it will talk to something called**asset service so** this u3 is talking to asset service through this load balancer  and again all the authentication and authorization will occur at this layer and now asset service will get the whole image or video whatever. Asset service will use amazon  s3 as datastore and it will store all the images there . Now based on which image is getting what kind hit on s3 it might decide to move some of the images on CDN and it could also decide to replicate on multiple CDN or some CDN from where it is getting traffic . But assuming it uploaded the image on s3 and it send a response to u2 through same websocket handler as we talk earlier.

**Now there could be one optimization  that can be happen**

Let say some sport event a lot of people sharing the same image for some reason . Do we need to upload the same image multiple times if five people uploading the same image probably not .

So before it even upload  to asset service the first thing that the device will do is take a hash of those service and send the hash content to asset service basically it a way of asking do u have already this images or not . If u have it then just send to user u2 . If u don’t have it then ask me and upload it

**Now let looks at remaining few components and  how they help in the whole system**

Diagram

Description automatically generated

This is the user app which talk to all these component . T could be browser as well .

The very first thing something called **user service. This** is  basically a service which own the user profile info like user id , name , wts image it would be reside in cluster mysql d/b which runs the user d/b. And all of those info would also be cached in redis  cluster . Against a key we will have user profile info. There is something called **group service  which** basically maintain all the group info. It basically has info which group has what all users . Again this info would be stored in mysql cluster  which would be geographically distributed and it has multiple slaves through which read queries will be run . It will also have a redis sitting on top of it . Each time user service and group service calling redis and it did not get data then it will call one of the slave and put data into redis.

Let any user is interested in sport so that analytic we can do using analytics service

**Spark streaming consumer**running on top of kafka that  consumer  which can do lot of inference it will classify based on message that is one kind of analytics will also put data into hadoop then we can run a lot of analytics query based on hadoop  which are the prople who talk about sport a lot .

These event are also having one more usage so each time a person does an activity on the app those event are listen by something called**last seen service. Now** this last seen service is basically the service use to store the last seen time of user . This service sit on top of cassandra d/b.

**What is Spark Streaming?**

Spark Streaming is an extension of the core Spark API that allows data engineers and data scientists to process real-time data from various sources including (but not limited to) Kafka, Flume, and Amazon Kinesis. This processed data can be pushed out to file systems, databases, and live dashboards.

**Monitoring**  : cpu utilization, disk utilization all those metrics we can show using graphana

NFR: it should be real time