WhatsApp, WeChat, messenger

Features? What do we need?

Basic functions: 1 to 1 with just text, login system, status, contact list

additional function: group chat? Sent/delivered/read confirmation? push notification? history message, multi-device login

send multi-

user A ------> servers ------> user B

message parallel

web servers

multi-

push servers

parallel

Asynchronous mode is necessary, no need to synchronize all the time.

Horizontal scaling, multiple servers are same set, easy to expand in the future.

Inside the server is a queue, producer-consumer model? Do we need to store all the messages in the server, or another database? Or we delete the message once the delivered is done? WhatsApp does not store messages.

Bunch of queues in one server.

Some other ideas involving, using cache, history and audit server, database sharding and replica, message backup (connecting to dropbox, google drive), some business consideration (ads)

load balancing for traffic between users and servers

Sent/delivered/read confirmation:

sent: TCP/IP, from user A to server, then ack back from server to user A.

delivered: server sent to B, then server sent delivered to A

read: set up another queue for read notification in server. When user B is online, send the message to server, then sent to A that user B has read the message.

Push notification: WhatsApp use google cloud messaging notification. Time critical is not as much as 1-to-1 text.

From database point: 2 tables. Message table and thread table because message table cannot handle group chat.

Thread table: thread id, participants, created timestamp, nickname, is mutated or not, last modification time, owner id

message table: message id, thread id, user id, content, creation timestamp

Message table: NoSQL (no need to change), why NoSQL?

Thread table (SQL): order by owner id and secondary index (last modification time, or owner id)

DB sharding: message table based on thread id; thread table based on participants id

Receiving message from server to user is by socket, by TCP connection. receiver also heartbeats with server to tell server it is online. After 1 minutes no heartbeat, push server will set this user offline.

Socket is for message delivering.

Web server is now for storing the message. Web server send the message to push server. Push server is in charge of sending to user B.

Web server gives sent/deliver/received function to push server? If in this way, push notification is not very useful.

Pull model is not efficient as it has to set periodically signal to the web server and most of the response is empty, wasting a lot of resources.

Another way to keep communication is by long polling service of http.

To support group chat, set up channel service, connect between web server and push servers

User A -----> Web server -----> channel server -----> user B

Channel keeps tracks of who are active in which thread. Less unnecessary sent from web server to push server.

Scale estimation: 500 million users active at the same time, one user can handle 50k concurrent, so this needs 10k servers.

Load balancer might be needed to redirect one user to a server.

Server and clients’ device should store some message for a short while in case of offline case happens.

Hbase for database storing the messages, not SQL or NoSQL.