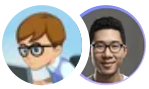
EP 24 - How to Design WhatsApp or Facebook Messenger? Messaging App System Design in 5 Minutes

How to design a Chat application like WhatsApp, Facebook Messenger, Telegram? Here is a high level design you can master in 5 minutes



JAVINPAUL AND NEO KIM

OCT 27, 2023 • PAID

Hello guys, Javin here! I write the Javarevisited Newsletter. It offers simplified weekly discussions on Java and coding interview questions. Consider subscribing to my newsletter. And prepare for your next interview for FREE.

Today, we will discuss a popular System design problem, how to design a messaging app like WhatsApp, telegram, or Facebook Messenger, or any chat application like Yahoo Messenger which I used to love.

In 5 to 6 minutes, you will learn what kind of database you can use, how you can store messages, what API you can provide, and how to design the whole system so that you can handle it when it comes to interviews.

This is a guest post by NK, if you also want to write a guest post for this publication, feel free to contact me or comment on this article.

Over to NK now:

Hey, NK here! I write the System Design Newsletter. It offers simplified weekly case studies on system design. Consider subscribing to my newsletter. And get the powerful template to approach system design for FREE.

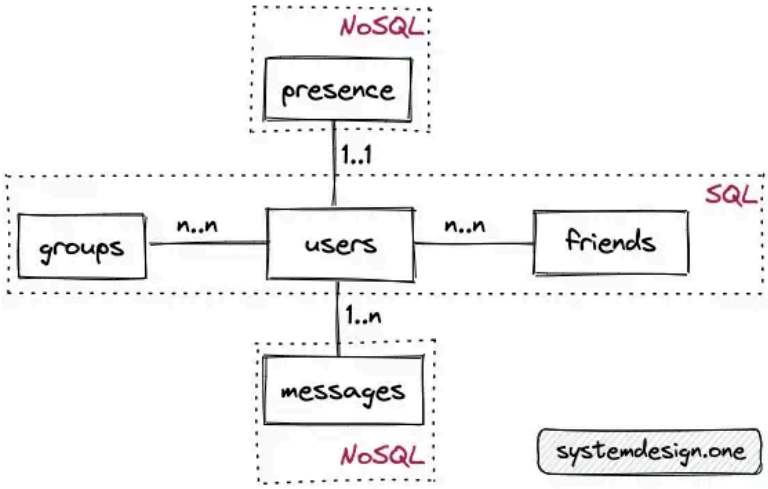
Note: Affiliate links in this post are not from me.

Requirements

The requirements are simple:

- A user can send a chat message to another user
- Users are spread across the globe

Data Storage



The main database tables are shown in the diagram. The friends and groups tables are used as join tables between users.

A NoSQL database such as Cassandra is a potential solution to store chat messages.

But I'll use MySQL to store account metadata and chat messages because:

- It offers strong tooling support
- The relational data model is a solid discipline

AWS S3 Object storage can be used to store media files shared in the chat.

Memcached or Redis can be used to improve data access efficiency by storing group chat messages and the presence status of people.

The Set data type in Redis can be used to track the online presence status of users.

API

The messaging app API is categorized into web API and real-time API.

The web API provides HTTP endpoints for users to log in and perform actions such as changing account details.

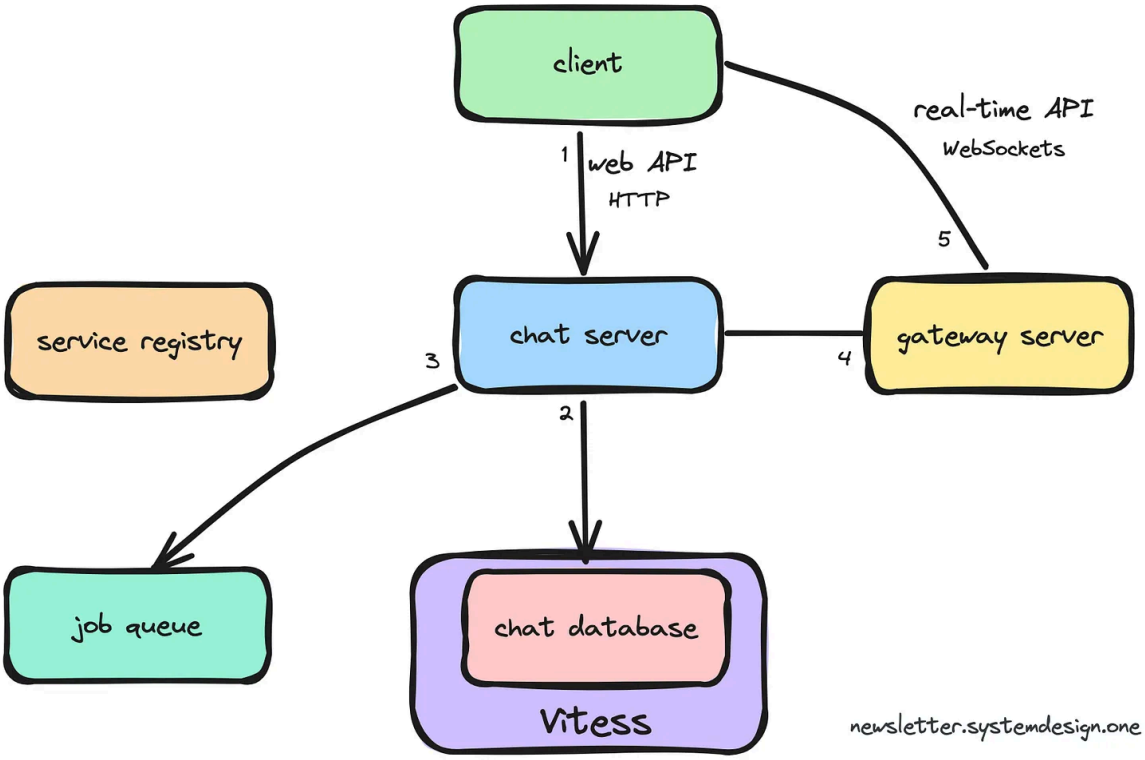
And the real-time API should provide WebSockets for bi-directional communication. It can relay chat messages and online presence status.

The API should be paginated for scalability. For example, the user only needs to get a limited number of messages.

A popular technique for API pagination is offset pagination. The idea is to send the number of results per page and the page number in the request parameters.

High-Level Design

A client-server architecture can be used to build the messaging app.



A monolith written in Java can be used to build the chat server.

The gateway server should be a stateless in-memory service. The messages get pushed to the client over WebSockets.

Consistent hashing can be used to route the user connection to the relevant gateway server.

Vitess can be used to partition MySQL because it automates the burden of scaling out.

The replication factor of the chat data store must be set to 3 for durability.

A service registry should be installed for services to discover each other and communicate.

A job queue using Redis Streams or Kafka can be used to defer non-critical actions like creating a message search index.

An edge proxy can be used for SSL termination.

Protobuf or Thrift can be used for high-performance data serialization.

A random unique token can be added to chat messages to prevent them from being displayed twice.

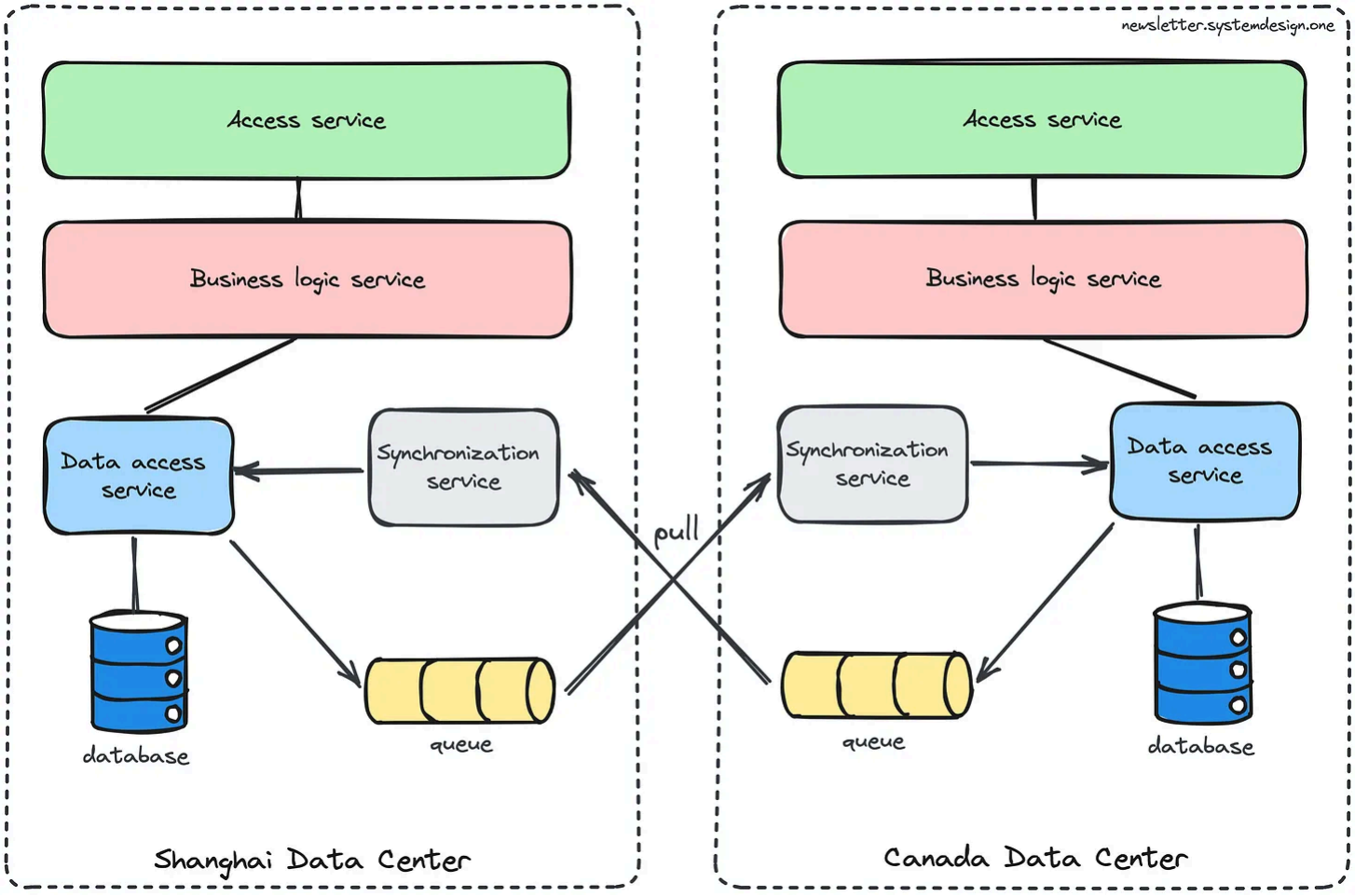
A CDN can be used to store the media assets that are shared in chat messages.

Design Deep Dive

A snapshot of the data on the server can be sent to the client for data synchronization between the client and the server.

The client can connect to the nearest data center using GeoDNS.

The data can be replicated asynchronously between data centers. This will reduce complexity and prevent data consistency issues.



Are you looking for a weekly newsletter on system design? Consider subscribing to my newsletter. And get simplified system design case studies delivered straight to your inbox. I cover the fundamentals and deep dives in my newsletter.

- *System Design Interview Vol 2*

- *Grokking the System Design Interview*
- *Designing Data-Intensive Applications*
- *ByteByteGo* ***by Alex Xu***
- *https://www.infoq.cn/article/the-road-of-the-growth-weixin-background*
- *https://systemdesign.one/slack-architecture/*
-

8 Likes · 2 Restacks

← Previous

Next →

A guest post by

Neo Kim

I write a weekly newsletter to help you learn system design.

Subscribe to Neo

Comments

Write a comment...