STSTEM DESIGN

Scalability -

Vertical - Buy bigger machine and can process req faster

-Single point of failure

-Inter process Comm

-Hardware limit

-data consistent

Horizontal - Buy more machines - req can distributed on any machine

-Load Balancing - Single machine req no LB.

-Resilliant - if one server crashes other is avail

-Network calls (RPCs)

-Data consistency is real issue

Consistent hashing:

-hashing - mapping key to hashes

req id

Message queue

-asynchronous call in queue

Chat

• One to one chat

• Online Sent/Read

• DB

• Security - End to end encryption?

• Length of messages stored?

-> USER A

-> Load Balancer (Stateless) - LB can operate at L3, L7 and L5 level

n1 or n2 or n3

-> DB (Cassandra

Redis Cache (distributed cache)

• Holds data in memory, master slave, can push data to db

• Holding 1 bl use data in memory.

• Redis saves USER, server name, time

-> HTTP architecture is req and response architecture.

Server can’t initiate req. Doesn’t work when res

1. Websocket - bidirectional connection - server and client can initiate req, Peer to peer protocol -

sticky session - req from user to one server always goes there

2. BOSH - works over http, server holds to req till certain time and new connection is send back

3. Long Poll HTTp - client send http Req send to server for new req

User table - user , converged, encrytionKey

Conversation Table -

Blob storage in encrypted

Search - in text message

AUTOCOMPLETE FEATURE

Feature - Language, geography, scalable, durable

Google or Bing -

REQ AND DATAFLOW

REQ FLOW ->

Get a prefix, look into your distributed trie

Api receives a prefix.

Trie -

Every node will have a top letters for letter

DC

• Getting list of strings and aggregate it in a trie

USER UI -> req -> Load Balancer ->Nodes -> Tries <- CDN

Looks into distributed cache

DESIGN DISTRIBUTED DATABASE

METADAT, REPLlication, edge case

Durable

Availability perf

Security - encryption

Consistency Model - Strongly consistent

NO ACID

YOUTUBE

• Million line of code

• Inflated views

• How much time they watch

• No of active users

TINDER:

Image storage - BLOB(Binary Large object) or Binary

DB feature-

1. Mutability

2. Transaction properties - ACID

3. Indexes are to improve search

4. Access Control - secure file system

Make a CDN - CDN allows fast access

DB - profile id, file URL, image URL

Distributed

steps-

1. Update profile

2. Authentication - response has a token,

SESSIONS:

Shard - data based on go to certain node

Master slave

---------------------