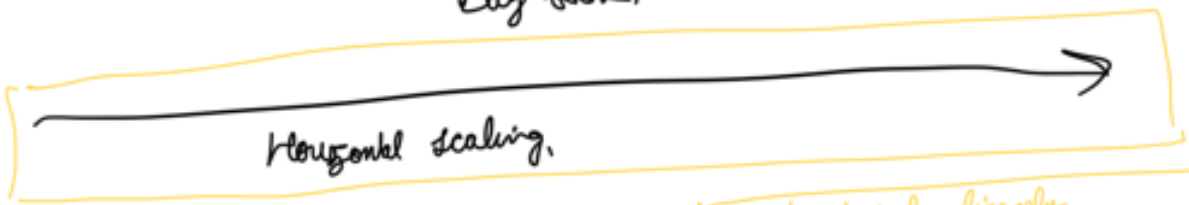
System Design Lecture Nikhil Lohia

Nikhil Lohia →

Beginner → Intermediate → Expert.

What is Scaling →. A way to expand and accomodate more customers.



Creating a Basic Application →

* Cost do not scale linearly

4 TB RAM x$
4 TB RAM x$
2x$

< 8 TB RAM → 3x$

cost of (2) 4 TB RAM < cost of (1) 8 TB RAM
thats why we use horizontal scaling

Course structure →

→ Basic of system Design
→ going overall terminology used.
→ Start with Basic design.
→ Understand each component and use cases
→ explore some complex design.
→ Based on how it goes, look at some real design too.

---

(2) Client - Server Model. →

→ App hosted on server.

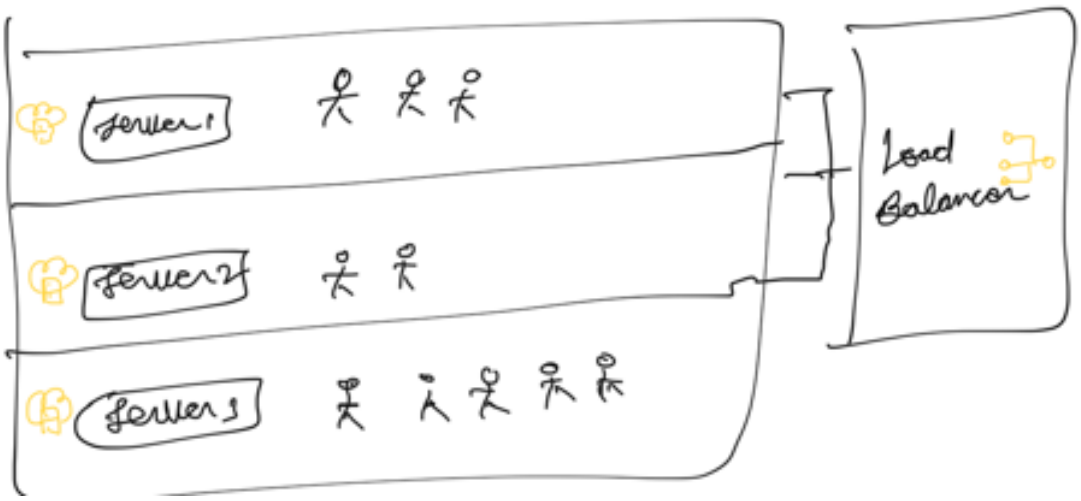
Advantages →

- central resource management
- scalability.

Disadvantages →

- single point of failure.
- Network Congestion.
- security Risk.

③ Load Balancer → Balancing a lots of request to a particular server.



Types of Balancing →

(I) Least connection
(II) Round Robin Scheduling
(III) Weighted Round Robin.
(IV) IP Hashing.

Advantages →

→ Traffic is distributed and processed parallel
→ Your design becomes easy to scale up.
→ High Availability.

Disadvantages →

→ A load balancer can get very complex and not easy to setup
→ The load balancer itself becomes a single point of failure.
→ If a load balancer is compromised, system can go down.

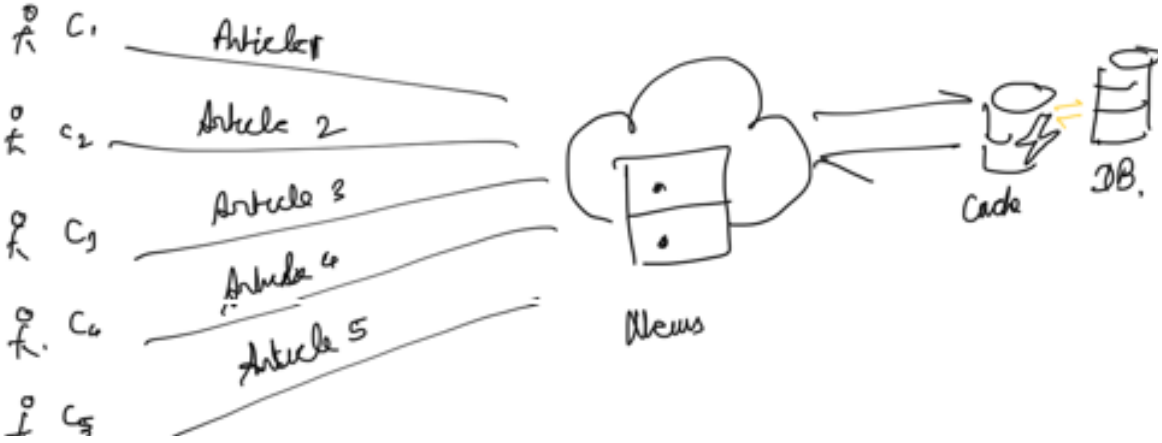
if load balancer goes down

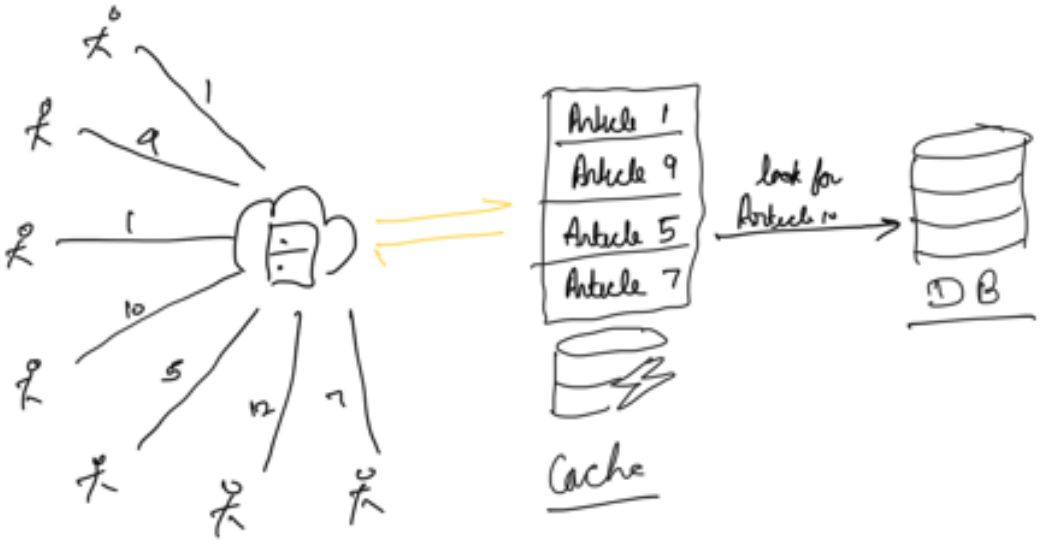
(4) Caching →

→ Increase efficiency of server
→ Saves time of client.
→ Increase speed of transaction

Symbol of Caching →

→ Kind of a memory / small Database
→ Very fast.
→ Usually small in capacity.





→ Higher the value of (Cache hit / Cache miss), better is the system.

Cache Eviction Method →

- Random Eviction
- FIFO → (First in First out.) → 1 2 3 → 2 3 4 → 3 4 5
- LFU (Least Frequently used)
- LRU (Least Recently used)

Things to be careful about →

- consistency of resource → suppose article 16 को बारबार fetch किया जा रहा है और किसी ने article 16 change कर दिया तो consistency affect होगा
- Coherence: Cache in different regions could get out-of sync.
- Security: Sessions and credentials should be cached with caution. sometime हम लोग login credential भी cache कर लेते हैं।

---

(5) DataBases →

→ Efficient Data Retrieval

↳ Quick queries for any information that you need. You don't need to scan entire storage.

```
Select name, marks from students. class 6
```

→ Data integrity or Consistency : All operations make sure that records remain accurate.

→ Handle Complex Queries → Find out performance, usage metrics, correlation, advanced insights, data mining.

challenges →

Uber → new rides | ongoing rides | payment

Scalability → Imagine a scenario that a lot of customers start coming to the store. lot of sales and a lot of new orders (Imagine you have USA, what happens at rush hour)

Consistency | region 1 | region 2 | region 3 | city 1 | city 2

Availability | back up 1 | back up 2

SQL V/s No SQL →

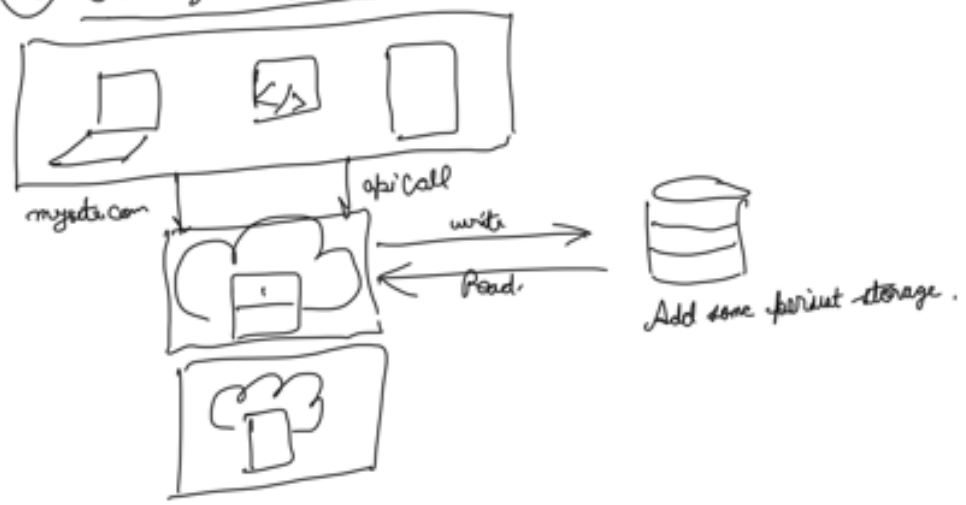
date	transaction	credit	debit	balance

SQL → PostgreSQL, MariaDB, Microsoft SQL Server

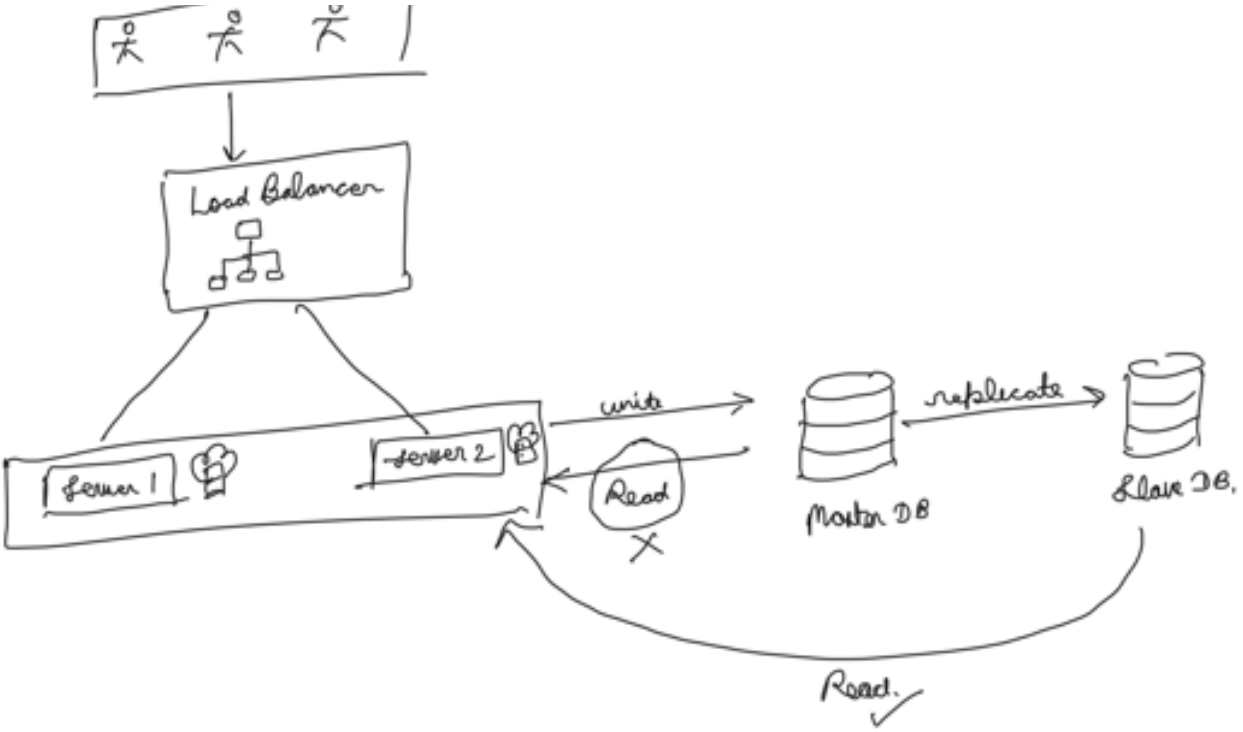
NOSQL → MongoDB, Cassandra, redis

---

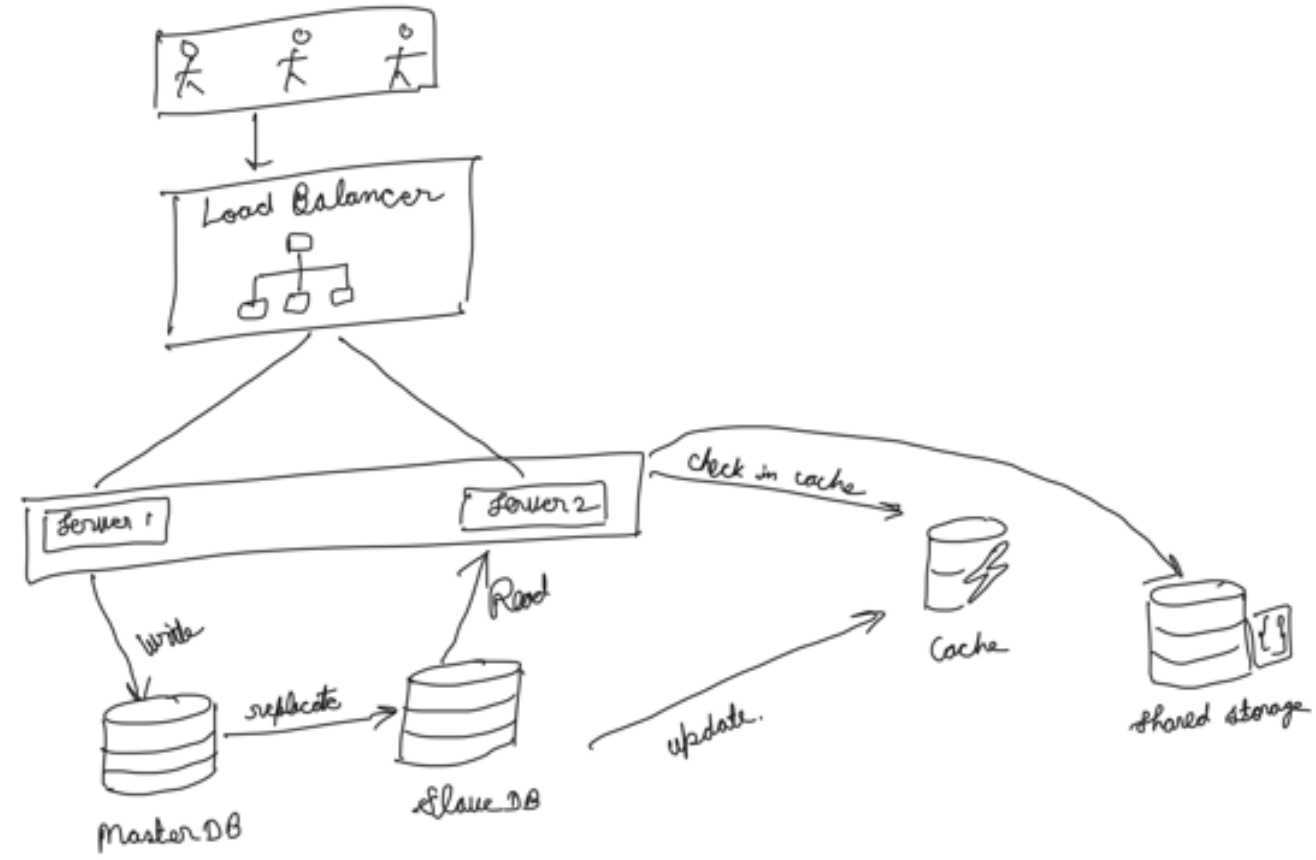
⑤ Creating Diagrams →

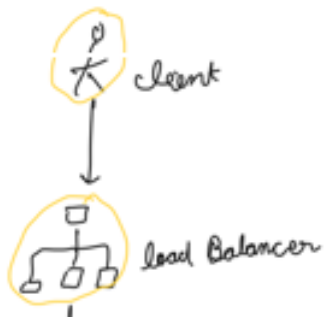


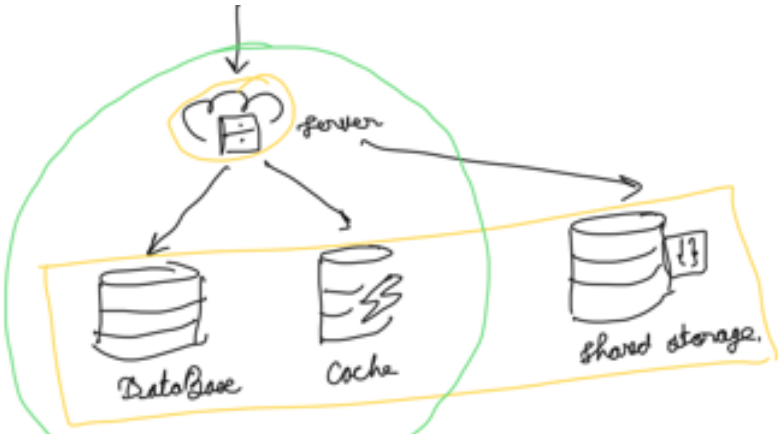
Load Balancer →


Load Balancer
Server 1
Server 2
write
Read
replicate
Master DB
Slave DB.
Read.

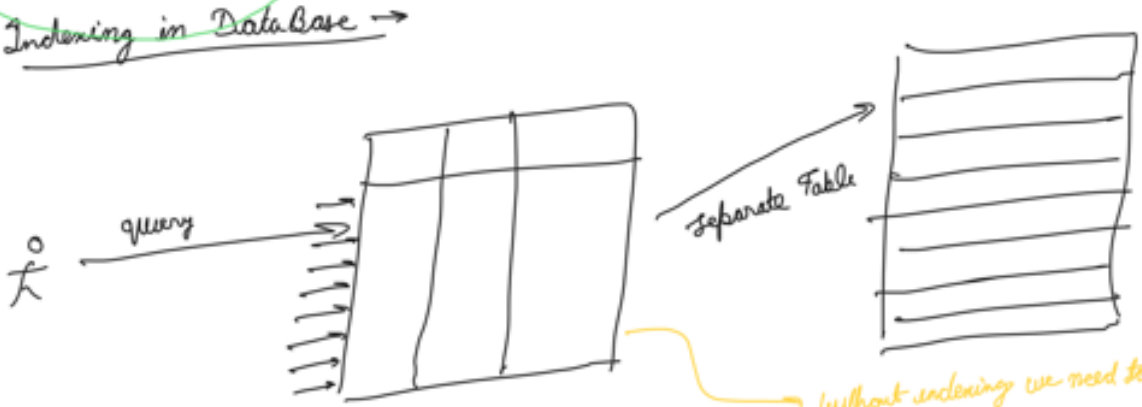
Caching →


Load Balancer
Server 1
Server 2
Check in cache
Cache
Shared storage
Write
Read
replicate
update.
Master DB
Slave DB


client
load Balancer



(7) Indexing in DataBase →



'B-Trees' is used for Database Indexing.

Types of Indexing Methods ⇒

P.K.

Date	Transaction	Credit	Debit	Balance.

→ Primary → Main Value for uniqueness and fast access.

→ Secondary Indexing → use another column with the primary key

→ Composite Indexing → Indexing Based on multiple columns to identify results.

e.g. → Amazon → multiple filter

→ Hashing Indexing → Using a function to store the data in a calculated bucket.

Challenges →

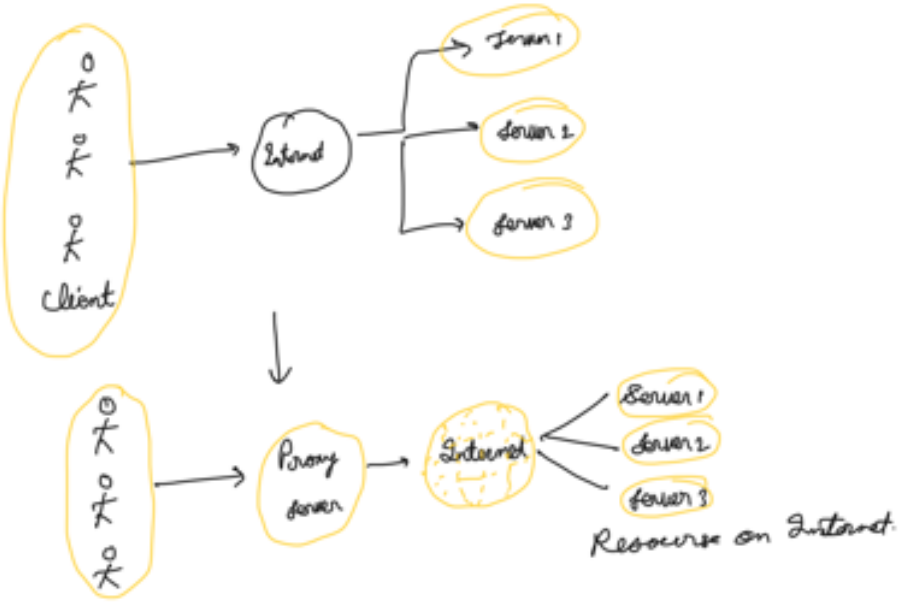
(i) The no of write operation will increase. (Update index table too)

(ii) Adding index is not free, we need more space → more cost incurred.

(iii) cleanup unused indexes periodically. (Audit regularly).

---

(8) Forward and Reverse Proxy →



Advantages →

(i) Logging → Allowing you to log each request made by the user.

(ii) Traffic Control → You can govern what traffic to pass and which to Block.

(iii) Encryption → Make sure that all the traffic going through the proxy server is encrypted.

(iv) Client Anonymity → Server dont know who the client are?

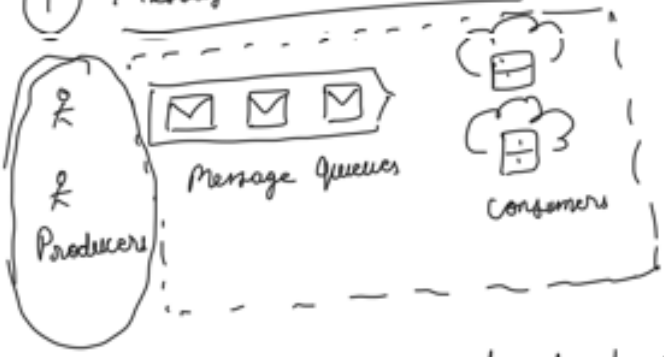
(i) Server Anonymity → clients do not know what servers are they connecting to.

(ii) DDos Attack Prevention → Without information about the server, we can't have a DDoS attack.

(iii) Optimization Caching → Can act as a load Balancer (Based on use case).

---

9 Message Queue and CDN →



→ Decoupling allows you to separate components and make them free sooner

→ The system becomes more scalable as more clients can send in requests asynchronously

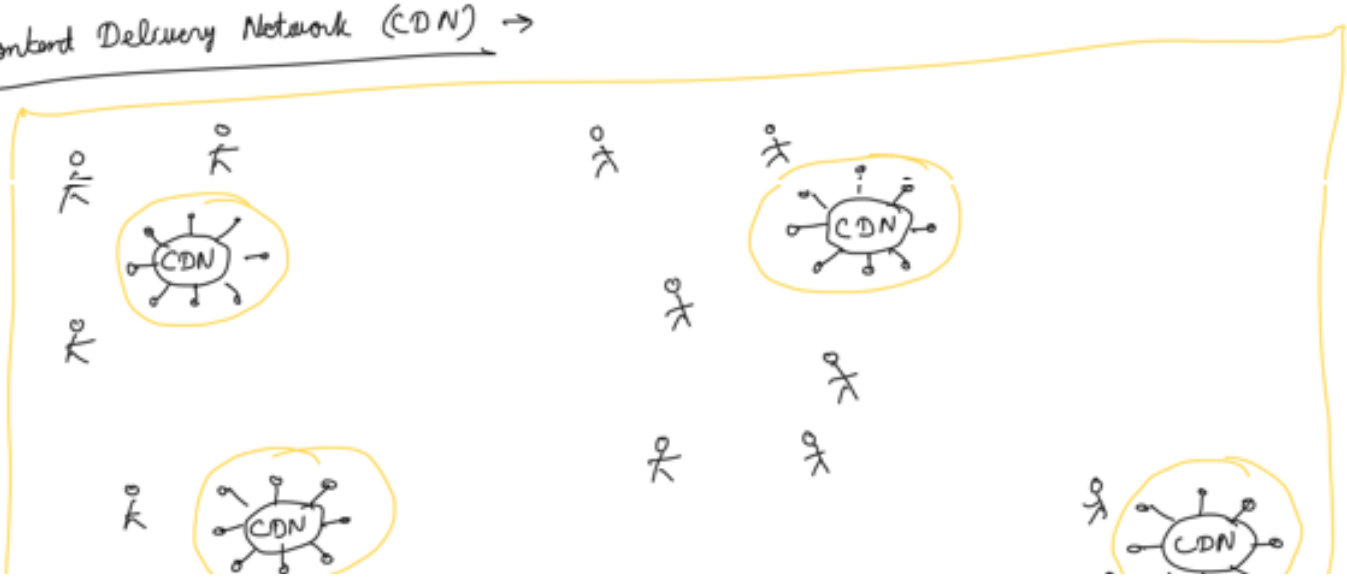
→ you can process the message again, if the first attempts fails. This makes the system fault tolerant.

→ Making sure that the ordering of the messages remain same in the order they were received

→ Handling duplicate messages to avoid redundancy and processing cycles

→ Adding something in the middle introduces latency. The benefits should overweigh the lag.

Content Delivery Network (CDN) →



Pros →

(i) Faster Load Times
(ii) Reduce Latency
(iii) Caching static Content

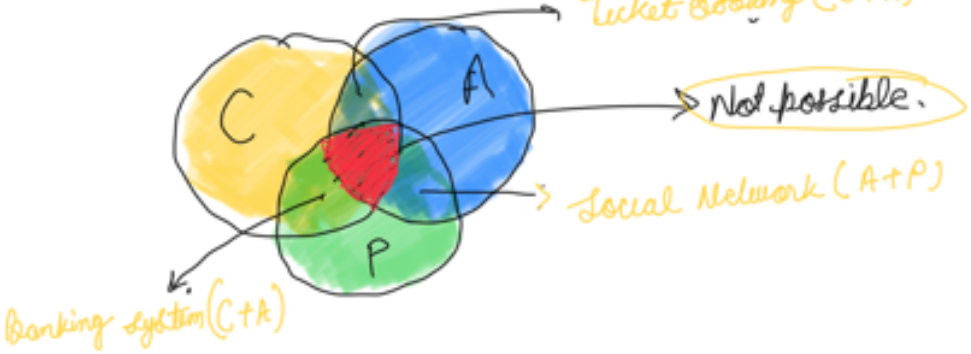
Cons → # Challenge →

(i) your data should always be fresh.
(ii) All the CDN infrastructure is costly.

---

(10) CAP Theorum

C → Consistency → System should deliver the same result no matter how and from where we query.
A → Availability → The system should always return some response or Result.
P → Partition Tolerance → The system should be functional even if one partition is disconnected.



The Paradox →

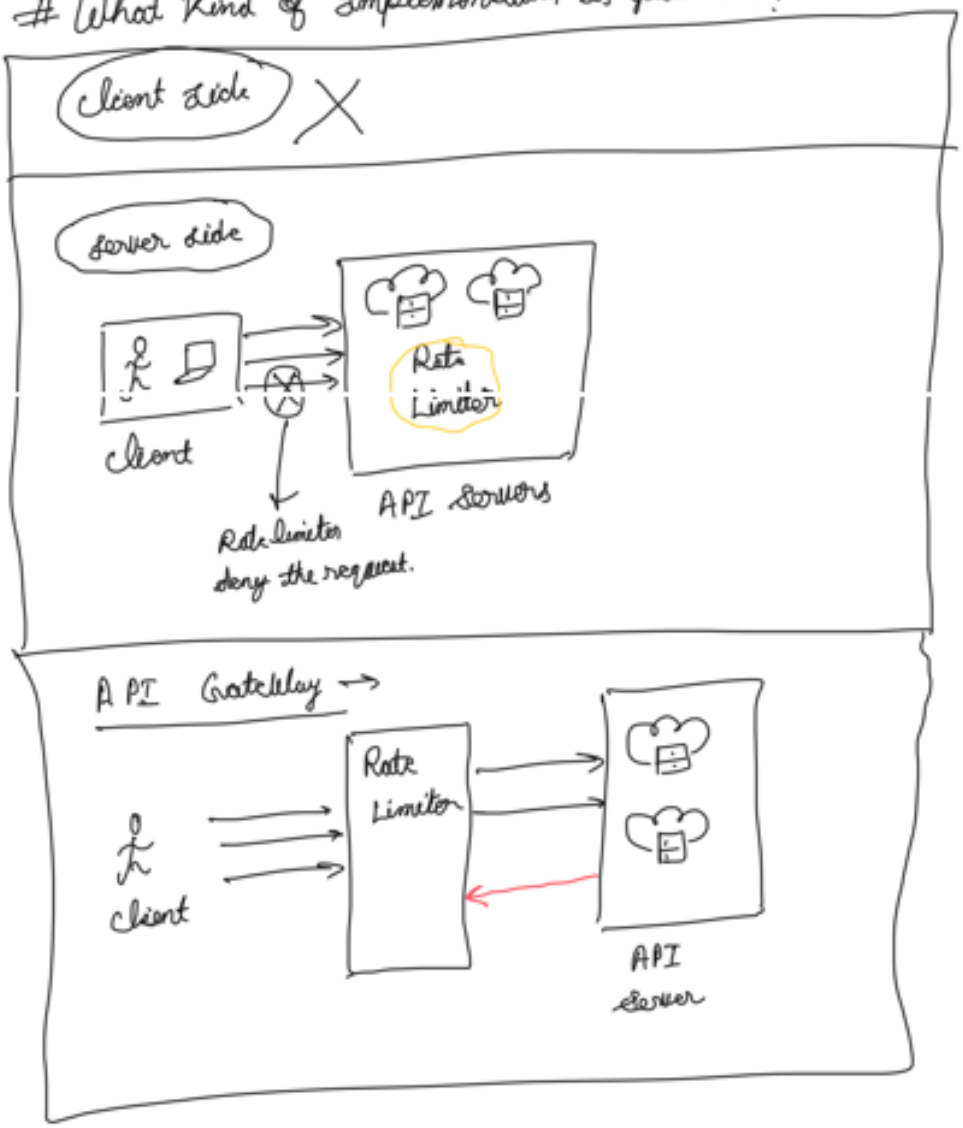
---

(11) Design Rate Limiter →

Q → Is this rate limiter client side / server side ?
A → It will be server side implementation.
Q → How should I throttle the request ?
A → There can be multiple conditions / rules to throttle a request.
Q → Do you want to inform users that requests got denied ?
A → yes, we should inform the clients.

Q→ Do you want a design a distributed system ?
A→ A Distributed Style.

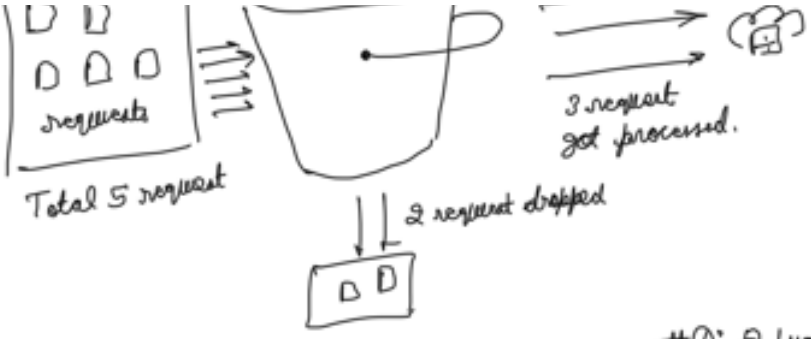
What Kind of Implementation is possible ?



Different Algorithm ⇒

(i) Token Bucket Algorithm
(ii) Fixed Window Algorithm
(iii) Sliding Window Log

(i) Token Bucket Algorithm →



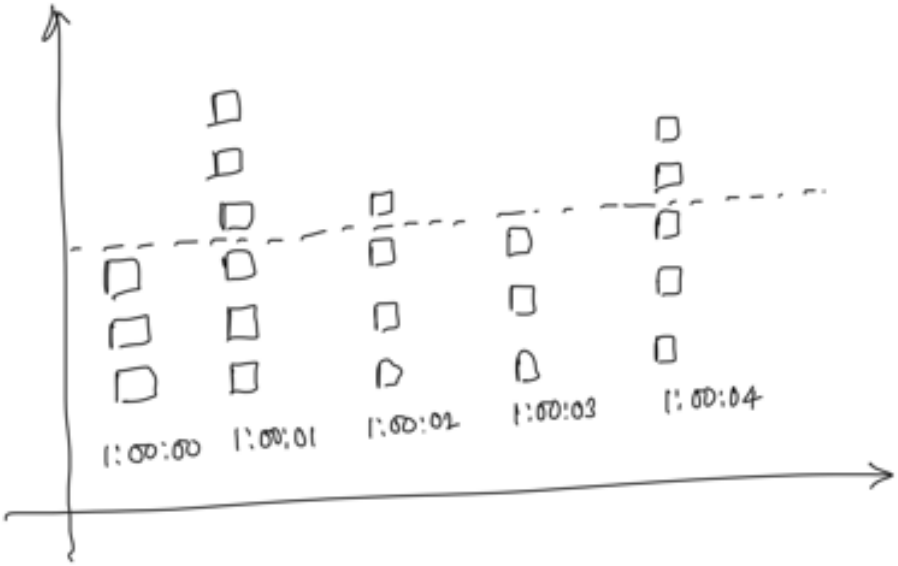
#Advantages →

(i) Very Easy to implement
(ii) ~~Memory~~ Efficient.
(iii) Allows Burst of traffic

#DisAdvantages →

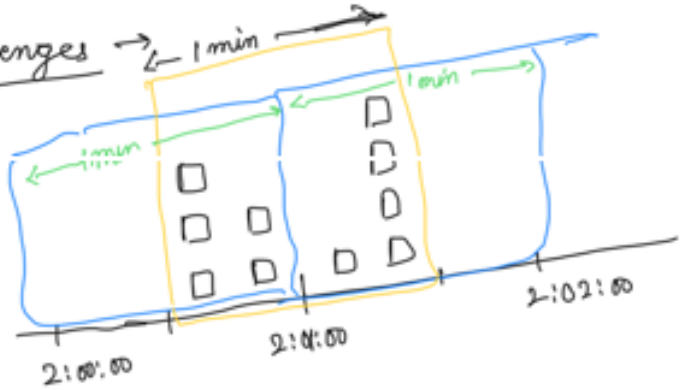
(i) Tuning the refill rate and bucket size is tricky

(ii) Fixed Window Counter Algorithm ⇒

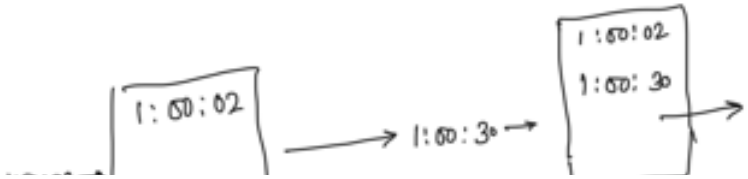


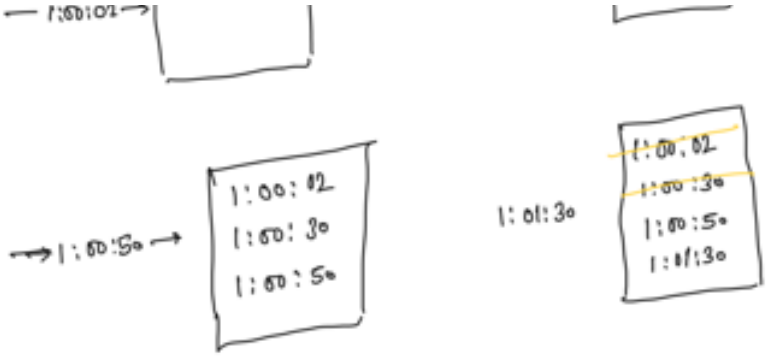
K=3

Challenges →



(iii) Sliding Window Log Algorithm →

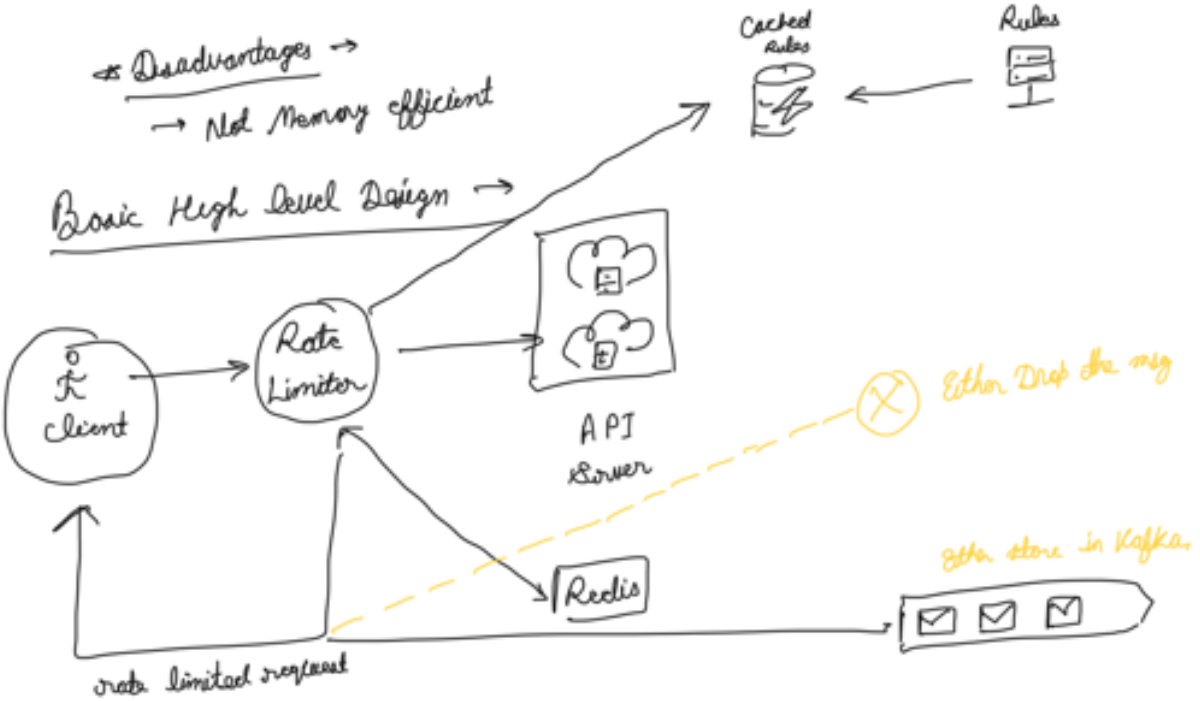




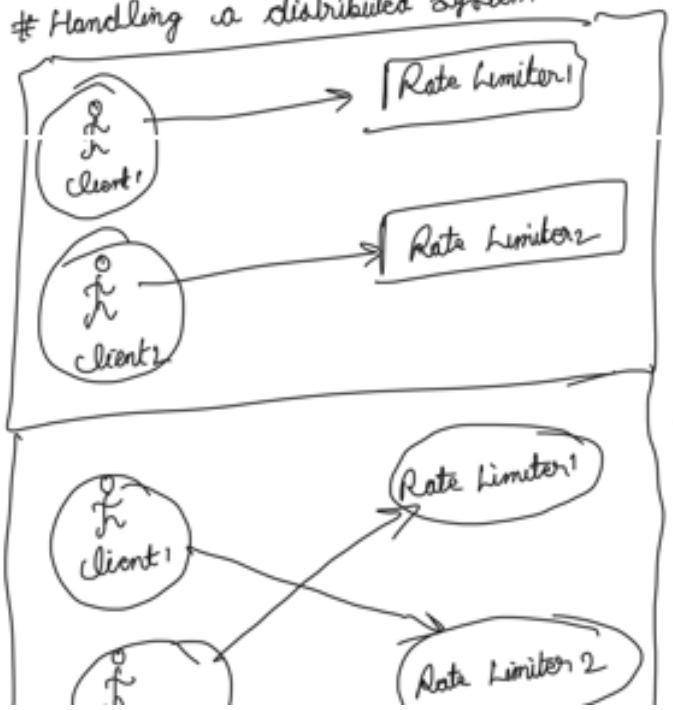
* Advantages →
→ very accurate and requests are handled correctly in a rolling window.

* Disadvantages →
→ Not Memory efficient

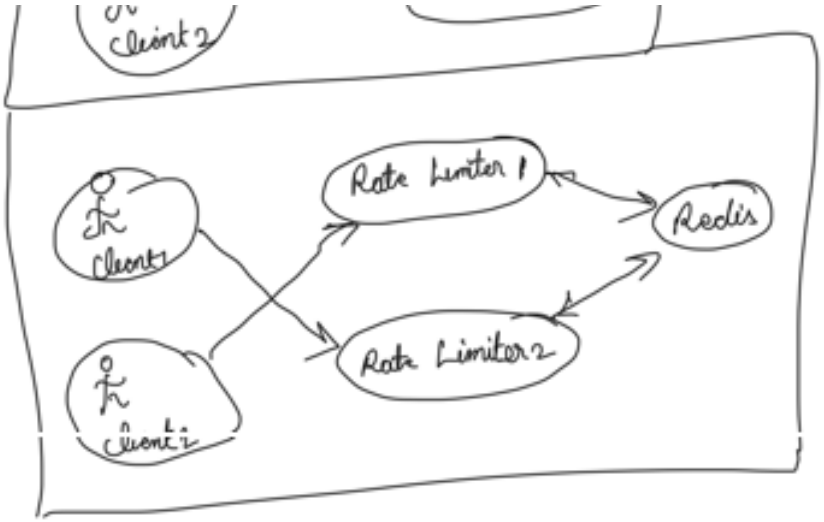
Basic High level Design →



Handling a distributed System →



→ Rate Limiter1 does not know about Client2 that how many req. Client2 has made and Vice Versa



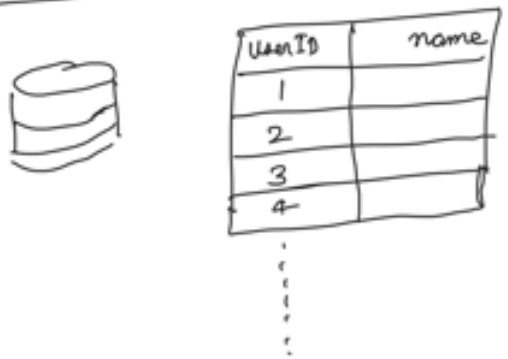
---

(12) Design a Unique ID Generator →

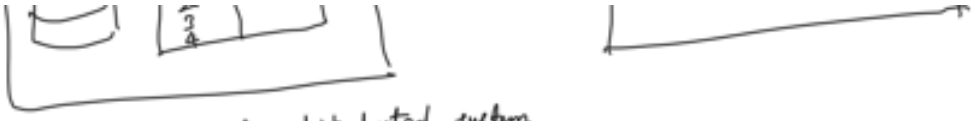
Why do we need Unique IDs ?

→ image / social media post
→ Video for youtube
→ Songs on spotify
→ order-ID for e-commerce
→ Comments on social network
→ Cash transaction for Banking
→ IoT device identification
→ message and even driven architecture

A Very simple sol^n →



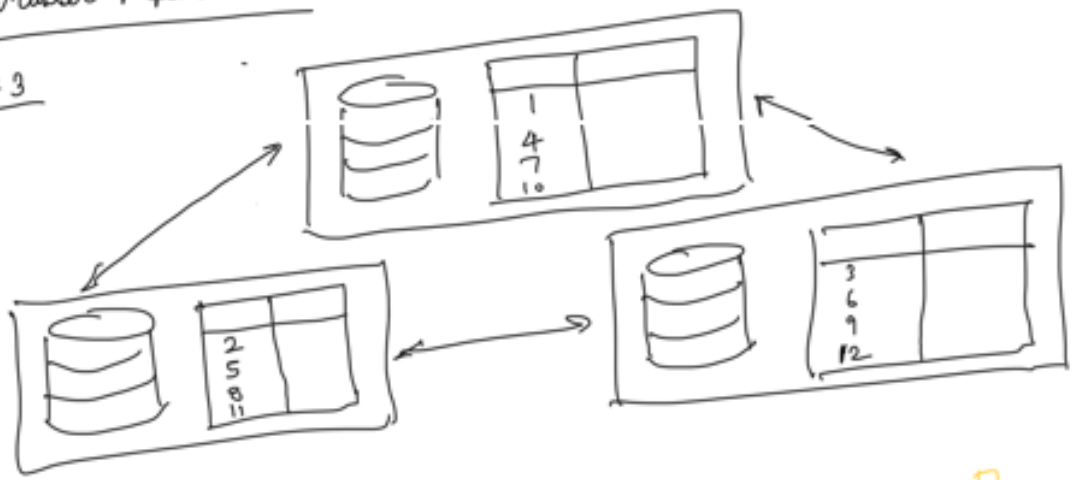
Good for single server.



→ Not good for distributed system
→ A single ID can get assigned to multiple resources.

CN MultiMaster Replication →

K = 3



→ Easy to Implement
→ Not Easy to scale.

Using UUIDs → a0ge26v8-----

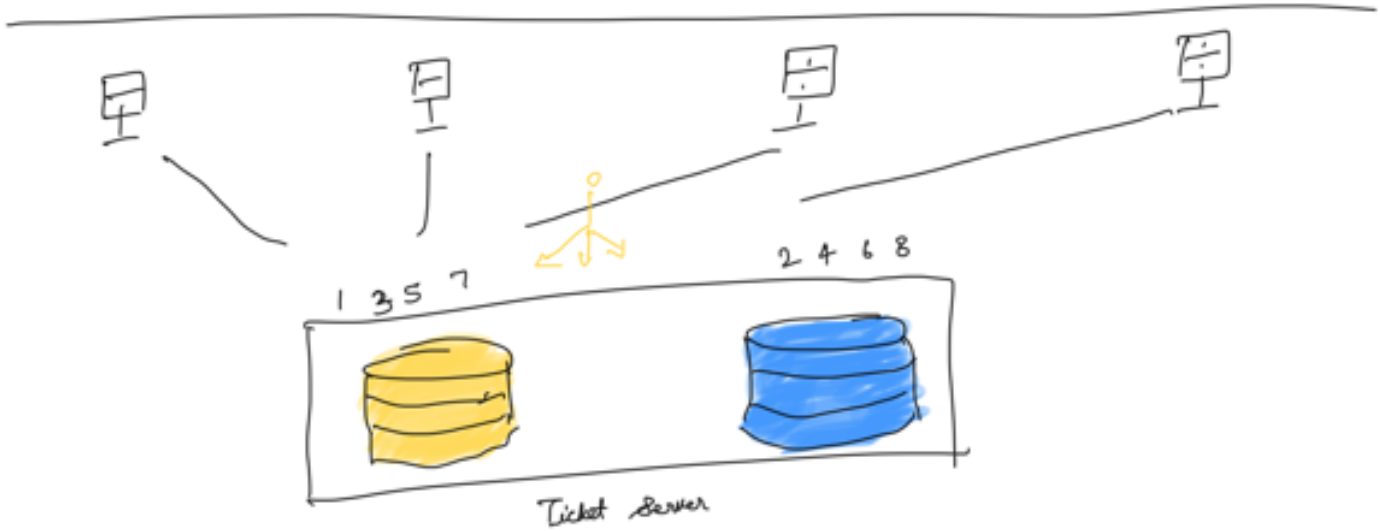
→ Simple way to generate UUIDs
→ No Co-ordination need b/w servers
→ Easy to scale and add new nodes

} Advantages

→ IDs generated are 128-bit (too long)
→ UUIDs are alphanumeric
→ Not Sortable

} Disadvantages

Ticket server Approach →



Advantages →

→ Easy to Implement.

→ We can get uniqueIDs in any format.

DisAdvantages →

→ Not good for a huge system.

→ Single point of failure.

Twitter Snow Flake Approach →

1 bit	41 bits	5 bits	5 bit	12 bit
0	timestamp	datacenter ID	machine ID	sequence number

Advantages

(i) Unique IDs are generated without any collision.

(ii) Fast, sortable ids (Also has information about timestamp).

Disadvantages →

(i) Time synchronization across server is important.

(ii) The implementation can be tricky

(iii) Limited by time range (69 years)

---

(13) Design a URL shortner → (long URL ⟶ short long URL)

Requirements for a URL shortner →

Q → What is the volume of URLs that we want to handle?

A → 100 millions / day.

Q → How short the URLs should be?

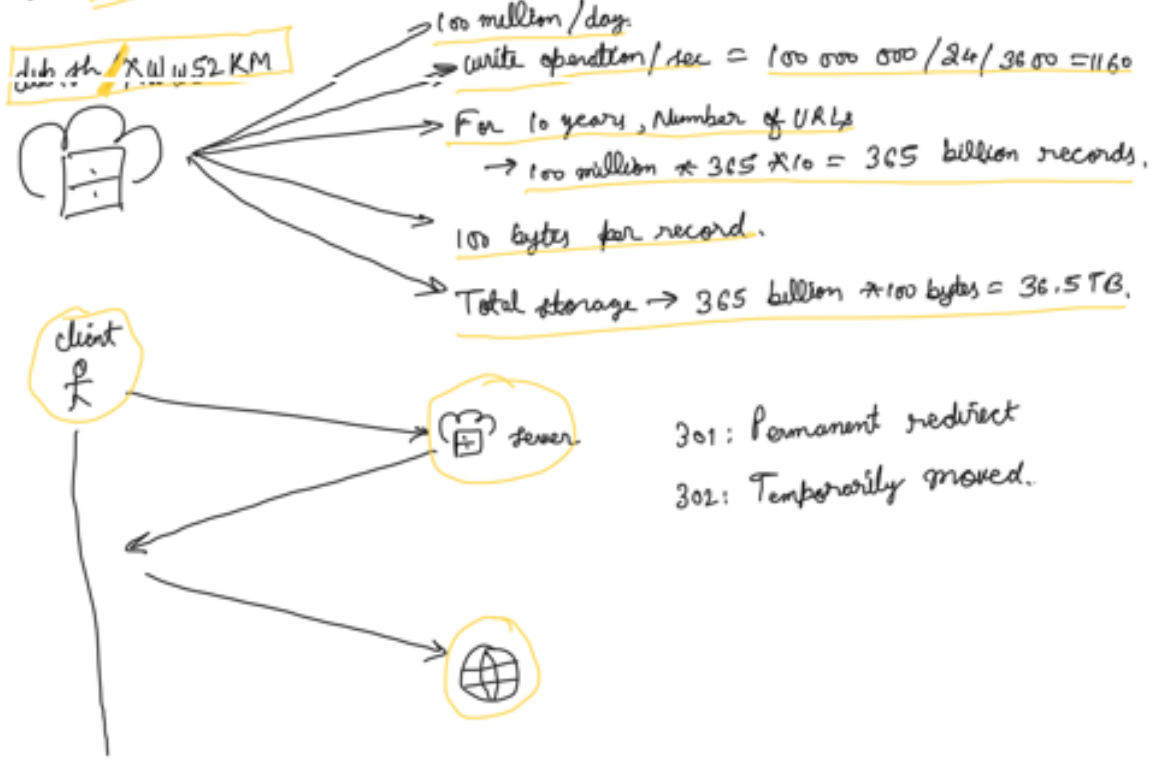
A → As small as possible.

Q → What kind of character do you need?

A → Alphanumeric without symbols.

Q → Can URLs be updated / deleted?

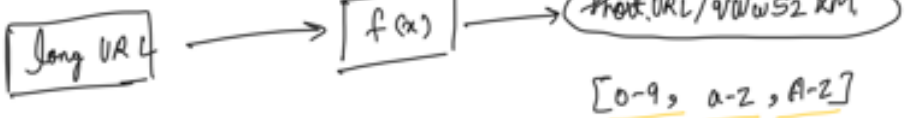
A → No.



301: Permanent redirect

302: Temporarily moved.

URL shortening (High level view)



[0-9, a-z, A-Z]

62 × 62 × 62 × 62 × 62 × 62 × 62 (7 space

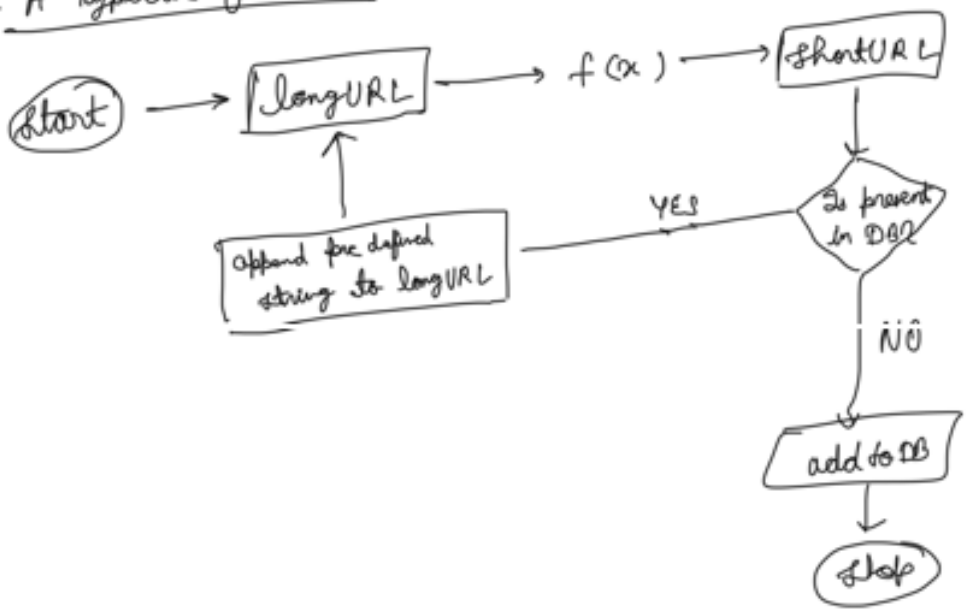
⇒ 3.5 trillion combination

Hash + collision Resolution →

https://www.youtube.com/playlist?list=PLFdAYMIVJQH0WJgRxfv_RH-ng95B2h3ON

MD5	5c654054
SHA-1	5a62509a84df9ce03fe123069fd8684c
CRC32	0ecae7916c06853901d9ccbefbfcaf4de57ed856

A Typical flow →



ID Based Approach →

[0-9 a-z A-Z]

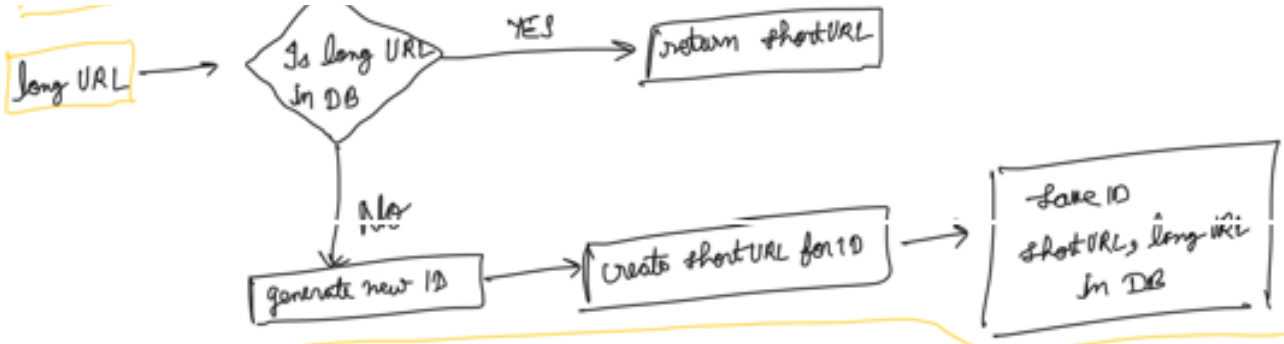
0-0	a-10	A-36
1-1	b-11	B-37
⋮	⋮	⋮
9-9	z-35	Z-61

62 | 11157 — 59 → X

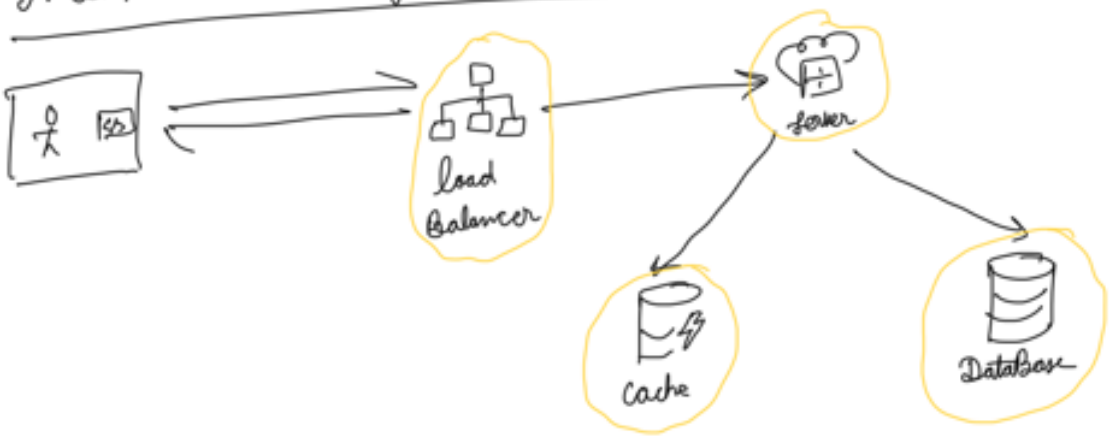
62 | 179 — 55 → T

2 — 2 → 2

url.source/2TX



A complete overview of the Design →



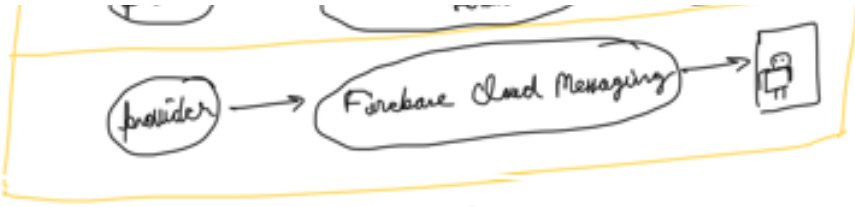
---

(14) Notification System →

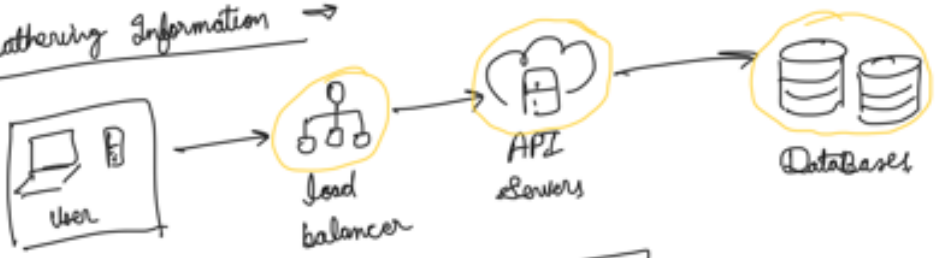
Push	Email	SMS

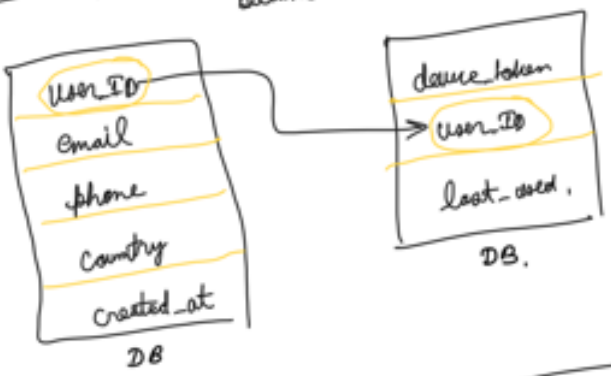
Q→ what kind of notification do you want to send?
A→ All 3 type (Push Email SMS)
Q→ Are these notification realtime?
A→ we want them in near realtime. A small delay is fine in busy hours.
Q→ What kind of devices do we want to send the notification to?
A→ All device (Laptop / Mobile / IOS / Android).
Q→ How are the notification triggered?
A→ Can be automatic / scheduled / manual.

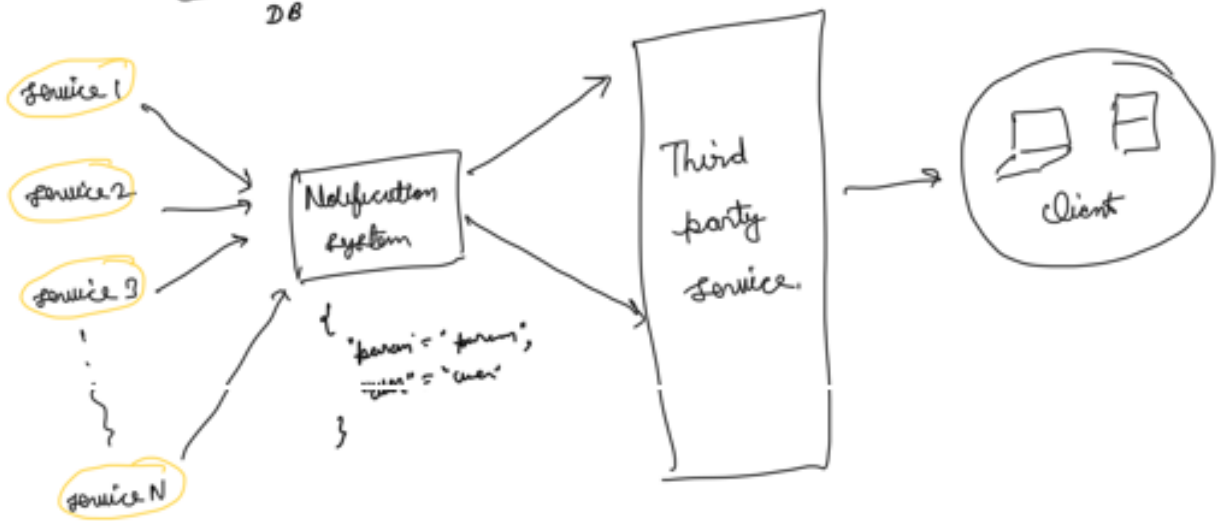
High level design →



Gathering Information →

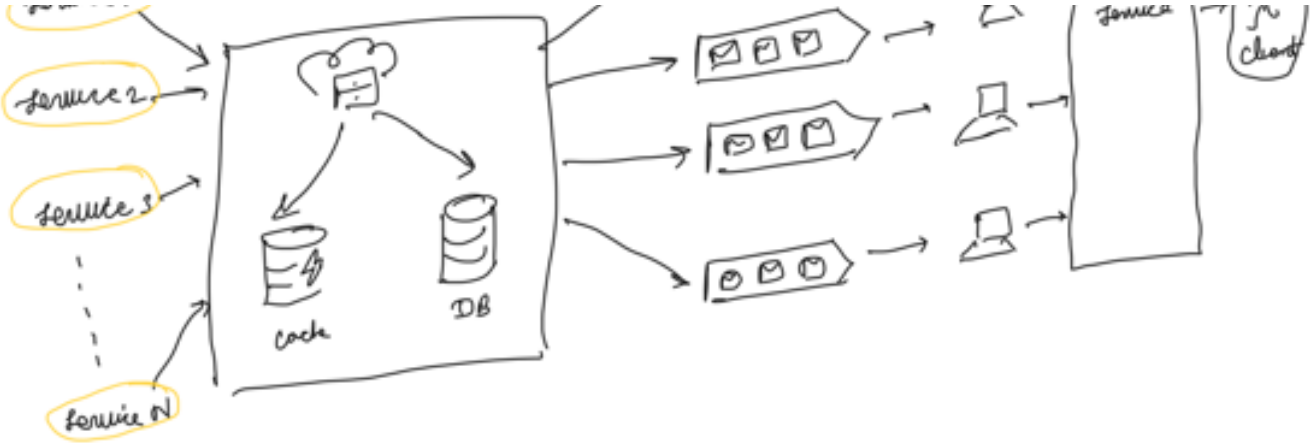




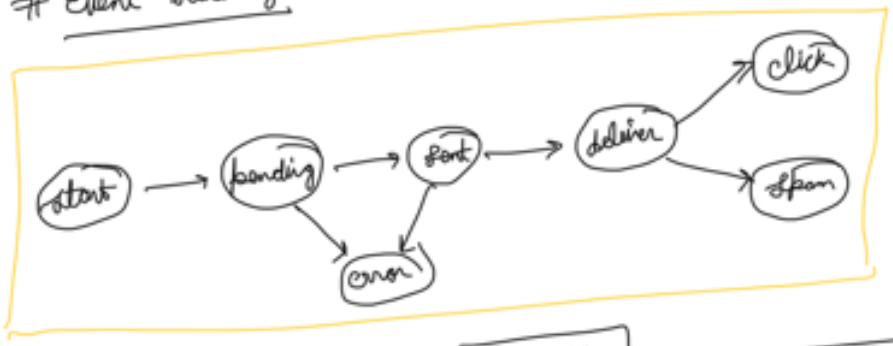


Notification Retry →

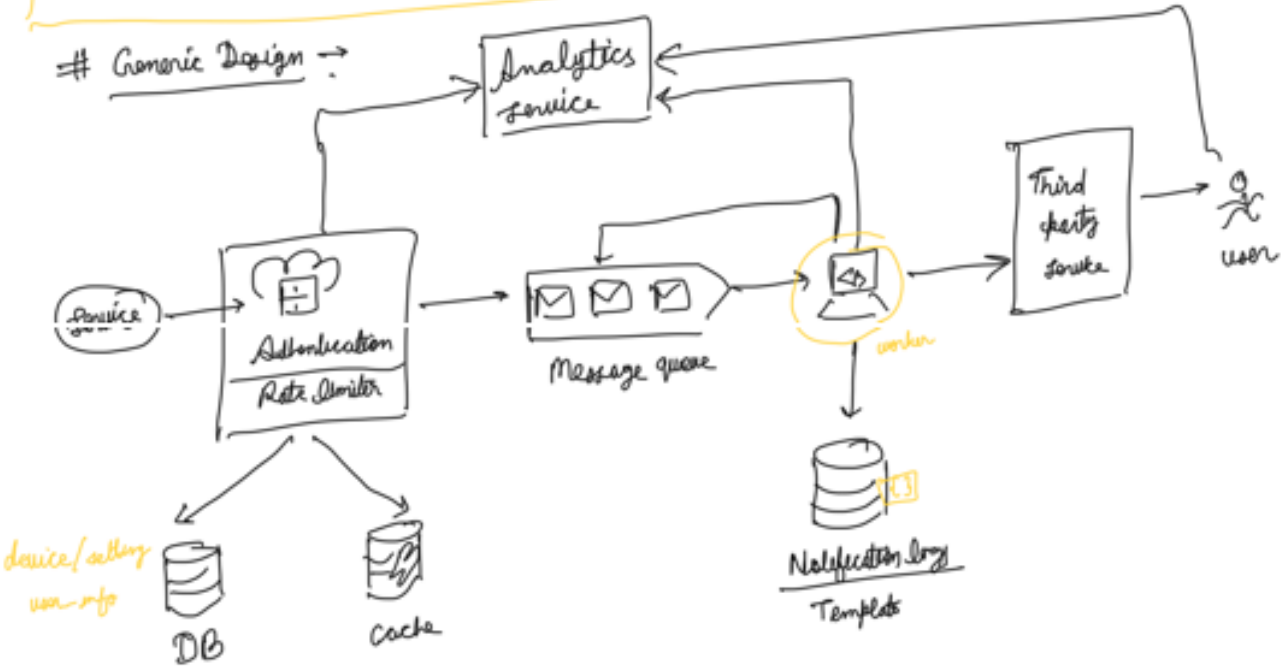
Improved Design →



Event Tracking →



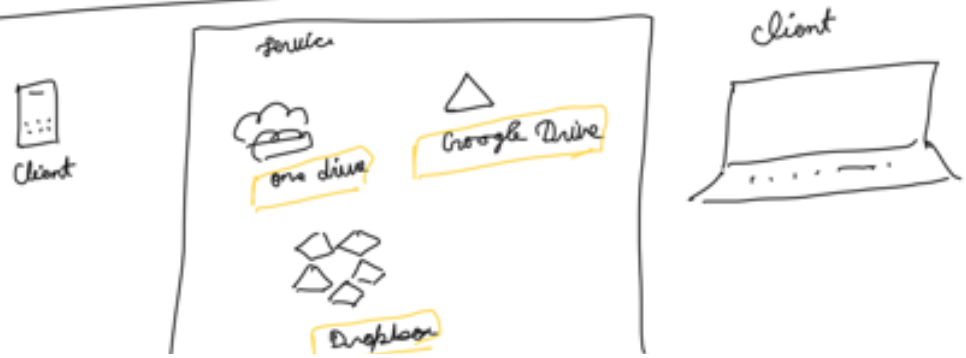
Generic Design →



---

(15) Design Drop Box →

A service like Dropbox →



Advantages →

(i) Availability → Data is available everywhere with internet connection.
(ii) Reliability and Durability → Data is secured and remains on the cloud.
(iii) Scalability → Your needs can expand and reduce as you desire.

Requirements and Considerations →

(i) User can upload and download from all of the configured clients.
(ii) User can share files with others.
(iii) Offline editing of files.
(iv) Synchronise files on different clients.

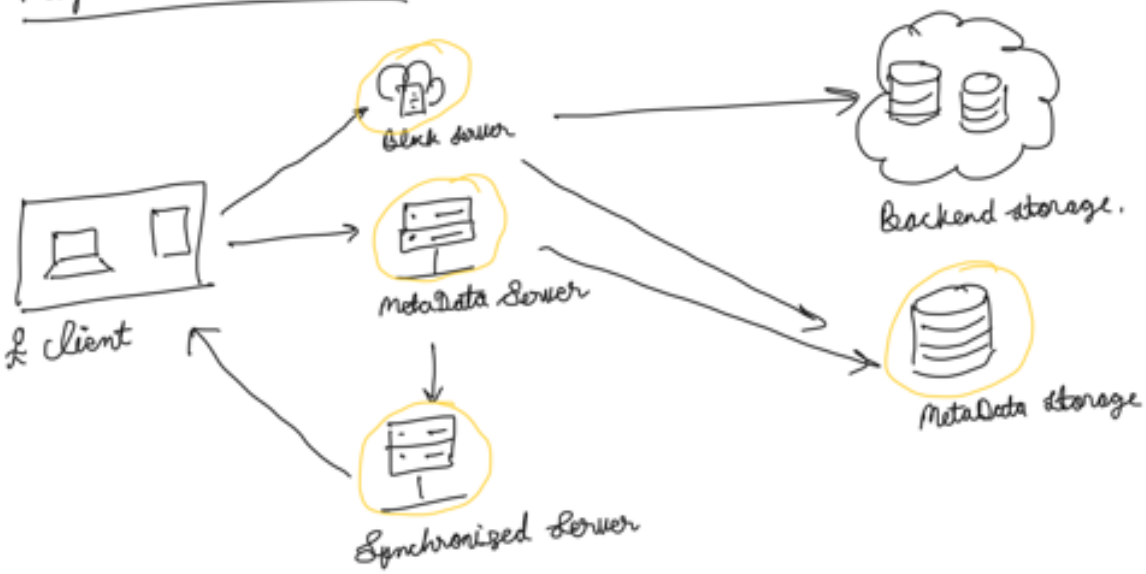
→ Versioning → Restore previous versions.
→ Premium subscription with more features
→ Very heavy on bandwidth. High upload/ downloads.

Idea of the Scale of System →

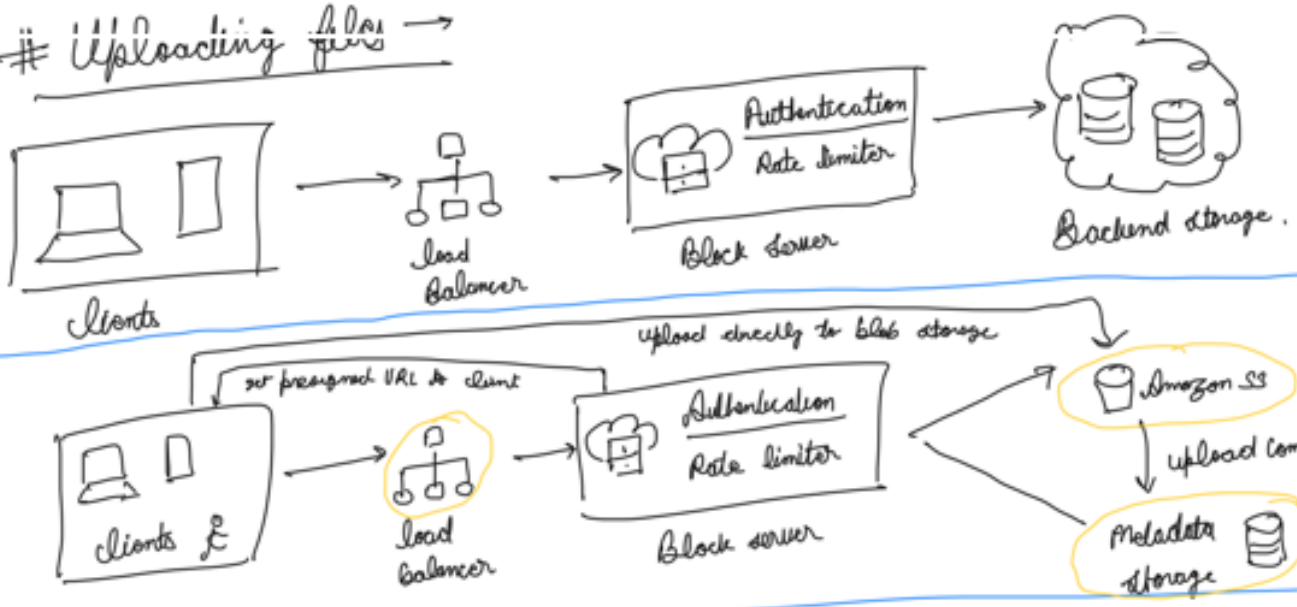
500 million total users
(100 million active) (20%)

→ Each users has average of 3 devices
→ Each user has average of 200 files.
Total files ⇒ 200 * 500 million = 100 B
→ Average file size = 100 KB
Total storage = 100 B * 100 Kb = 10 PB.

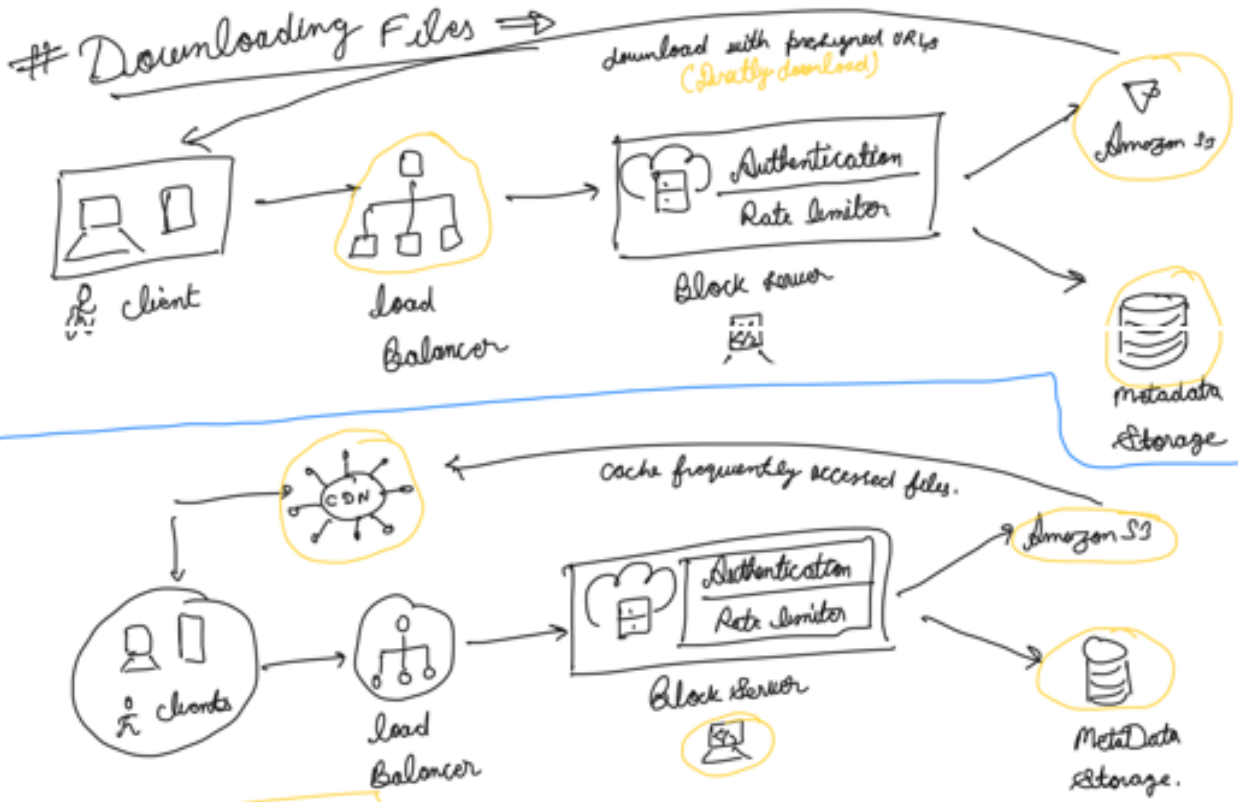
High Level Design ⇒



Uploading files →



Downloading Files ⇒



Presigned URLs → Gives you temporary access to private resources

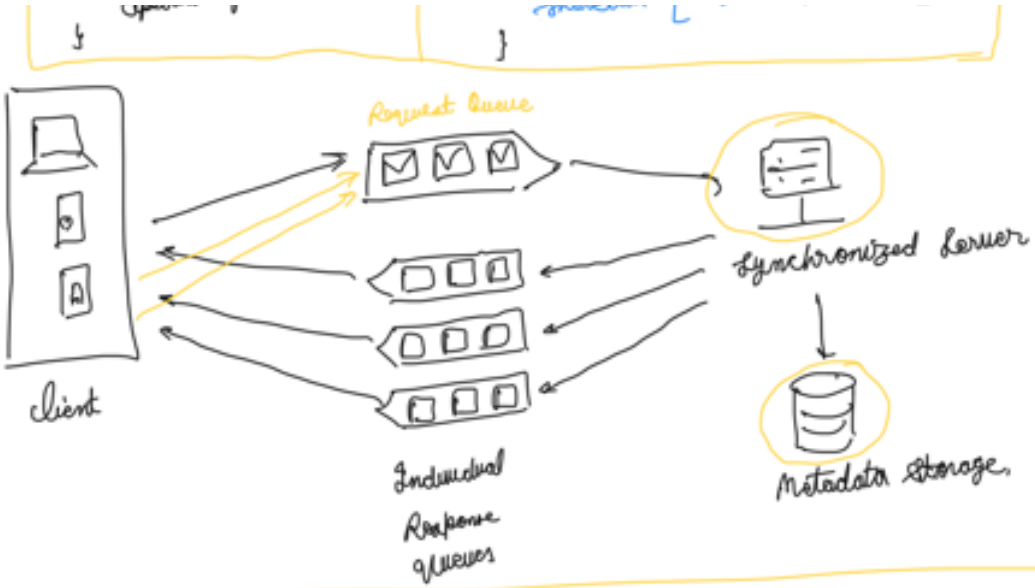
Sharing files →


```
{
  id : 123
  name : file.txt
  size : 1000
  mimetype : text/plain.
  uploadedBy: user1
}
```

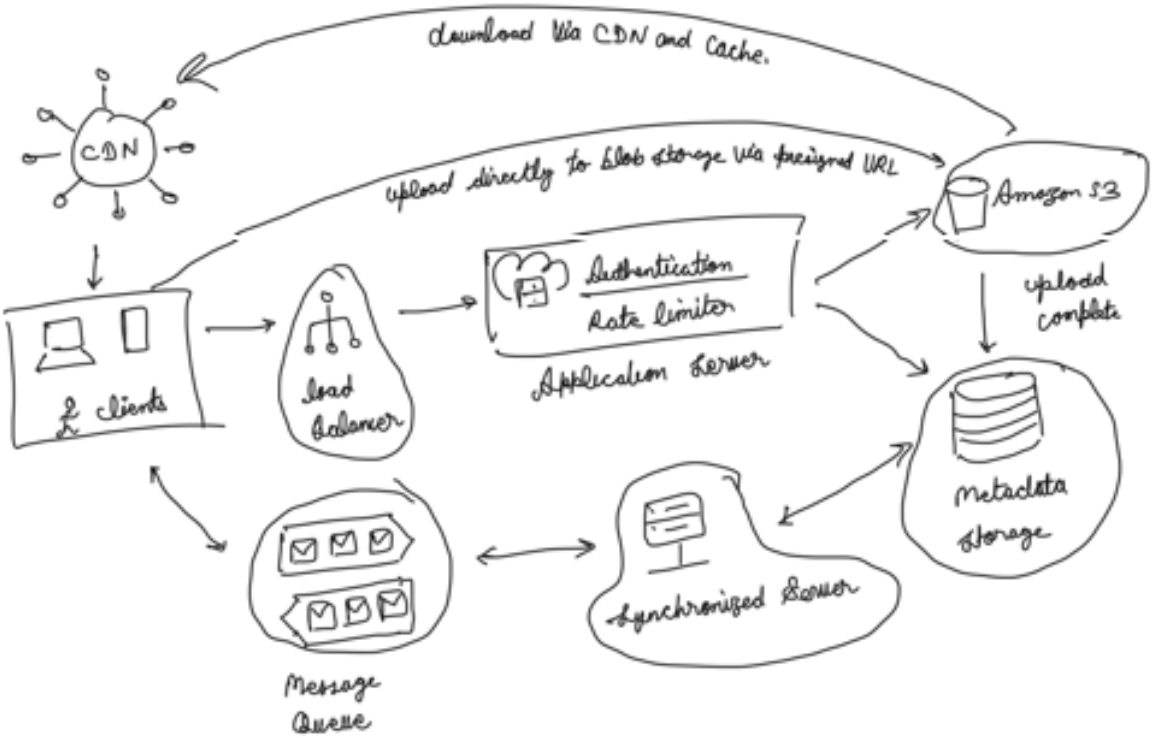
⇒

```
{
  id : 123
  name : file.txt
  size : 1000
  mimetype : text/plain
  UploadedBy : user1
  sharedlist : [ "user2" , "user3" ]
}
```

user1 : [file ID1 , file ID2]



Full Picture →



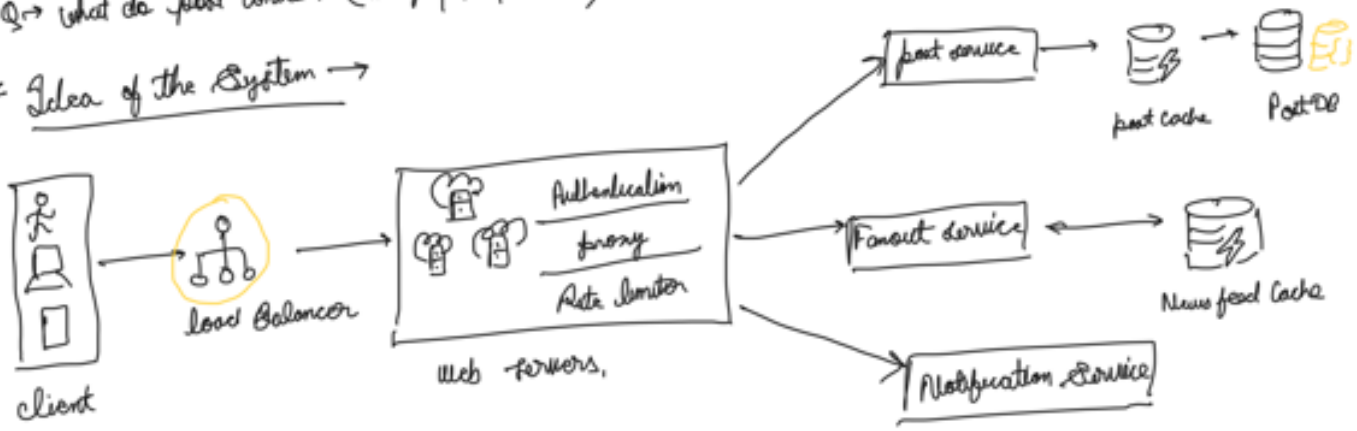
(16) Design a NewsFeed →

→ Should be available for mobile and as a web application too.
→ Most important feature: should be able to publish updates and push to friends.
→ Update should be in near realtime.
→ Have the update in a reverse chronological order (Newest first).

Q → what traffic volume is expected ?
Q → what is the maximum no. of friends?

Q→ what do post contain (image | text | Video)

Idea of the System →



Fanout Service →

Fanout on write (PUSH Model)

News Feed Cache is pre-populated during write.

→ generated in real time, can be pushed immediately

→ fetching news feed is fast, as it is pre-computed

→ if a user has many friends fetching and generating the newsfeed is very slow.

→ for inactive users, creates a lot of resources.

Fanout on Read (Pull Model)

News Feed Cache is populated during read time

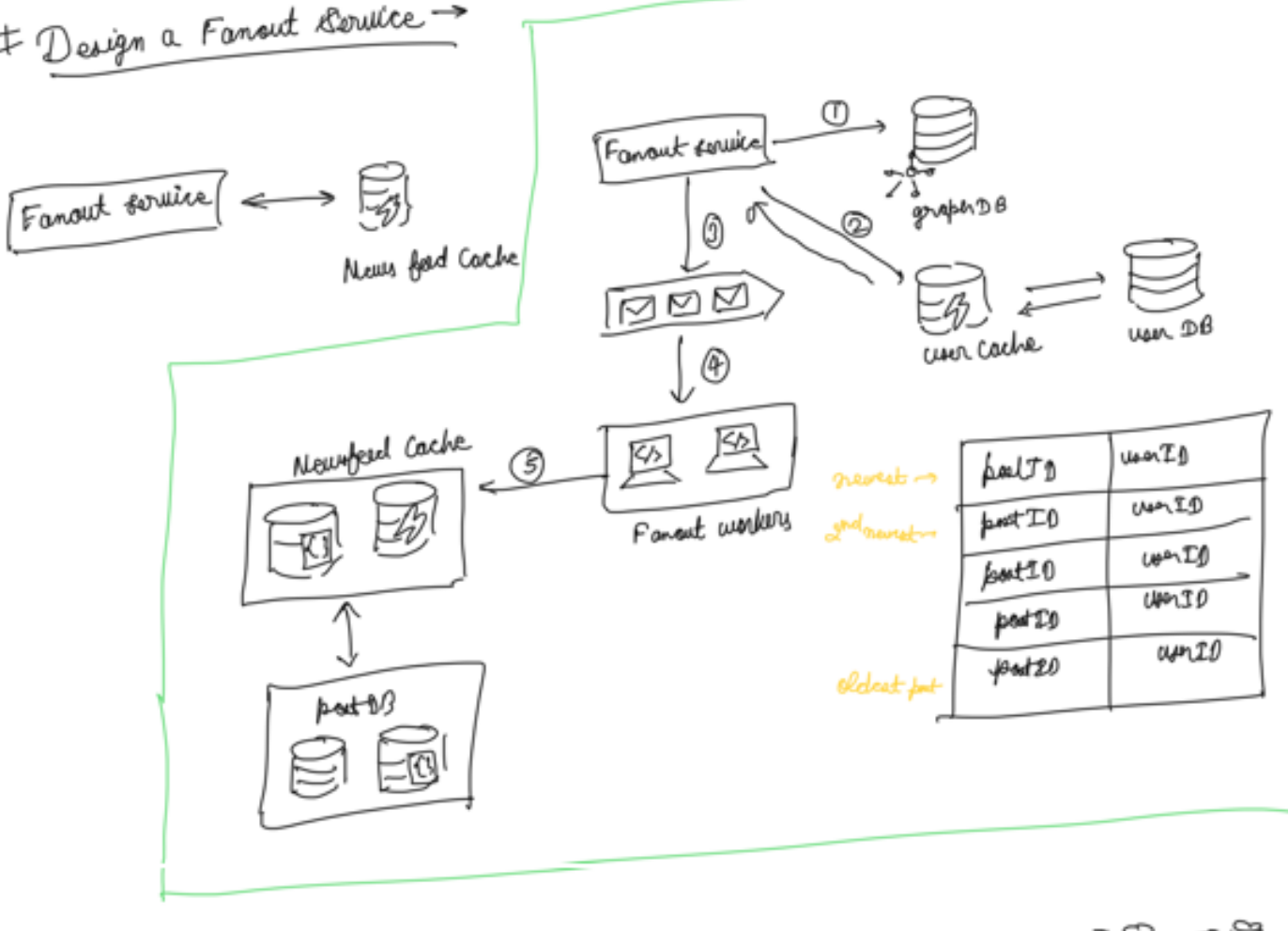
→ for inactive users, fanout on read works better,

→ data is not pushed to everyone, saves bandwidth.

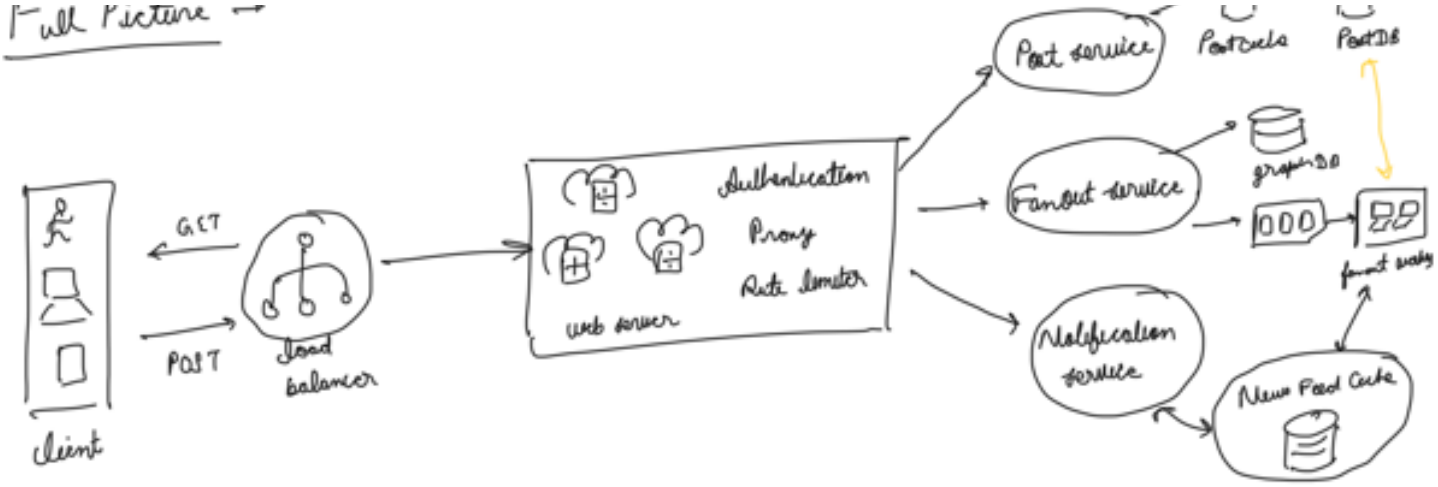
→ the process is slow

→ User can have a bad experience.

Design a Fanout Service →

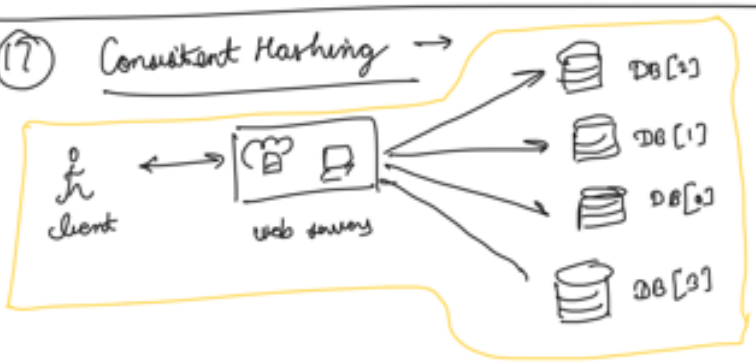


Full Picture →



---

(17) Consistent Hashing →



(eventID) % (no of servers)

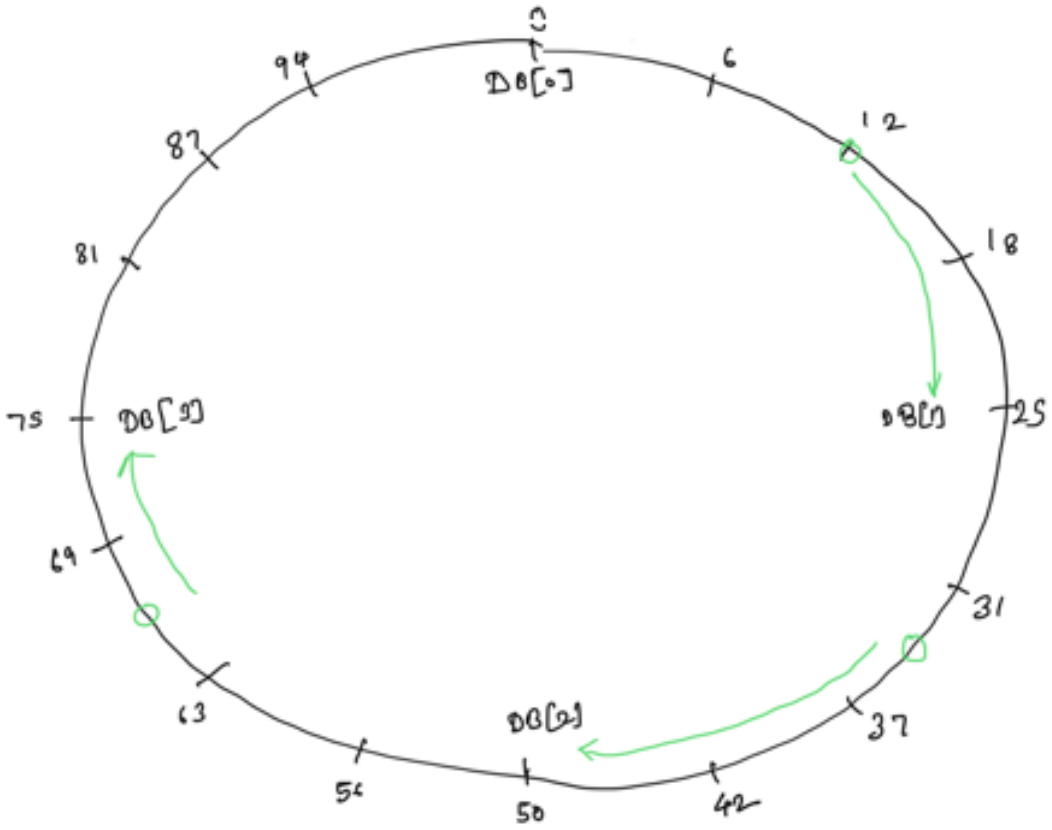
1234 % 3 = 1

6666 % 3 = 0 → 6666 % 4 = 2

5612 % 3 = 2.

↓

we need to rehash



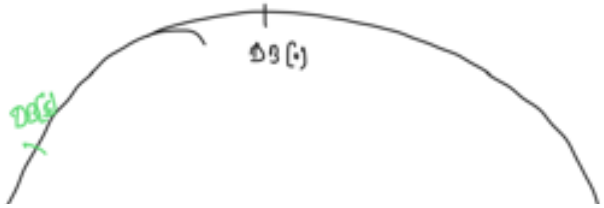
1234 % 100 = 34

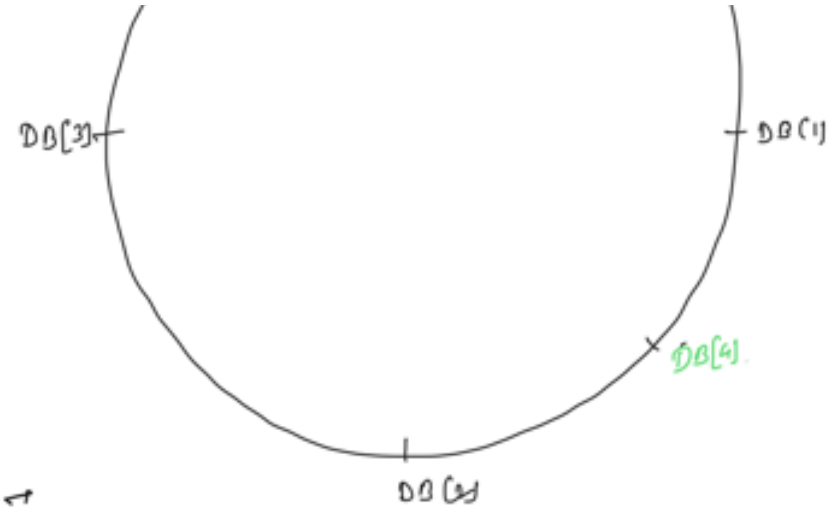
6666 % 100 = 66

5612 % 100 = 12

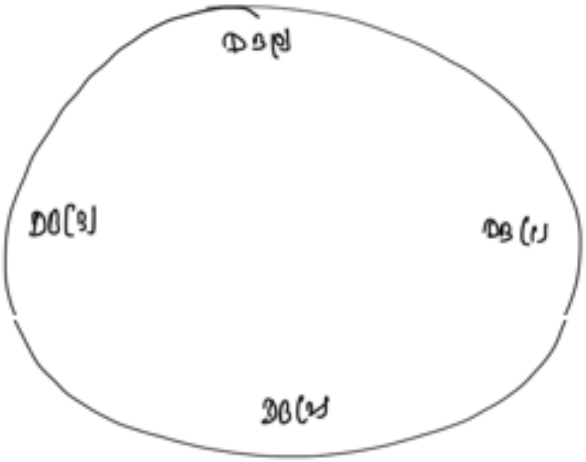
Lets say we need to add new DB[4]

Adding DB →

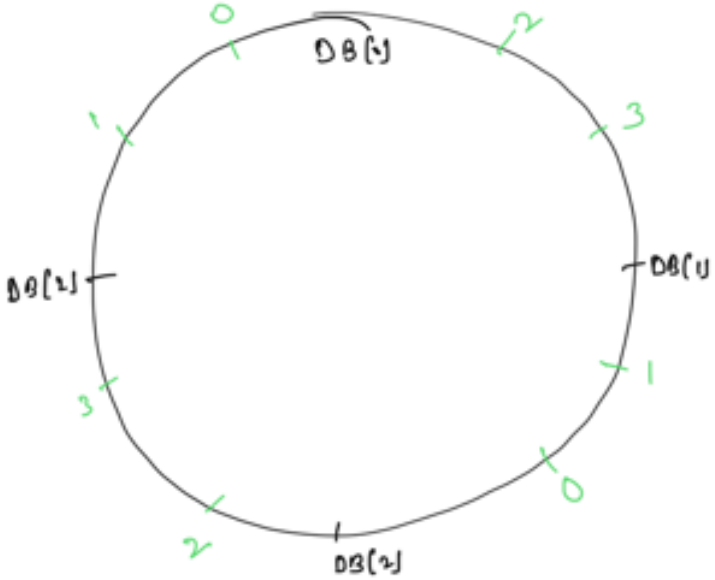




Removing DB →

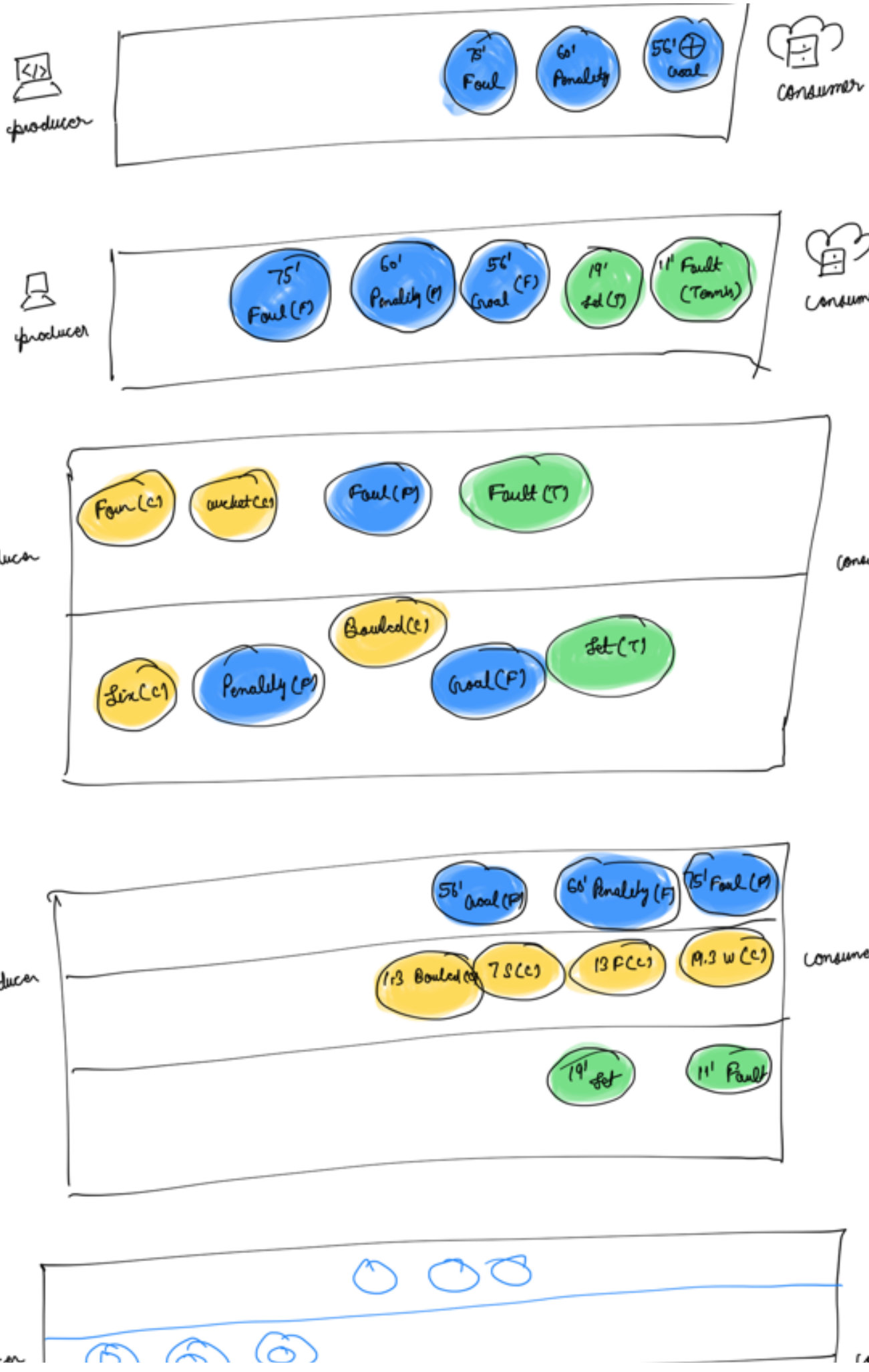


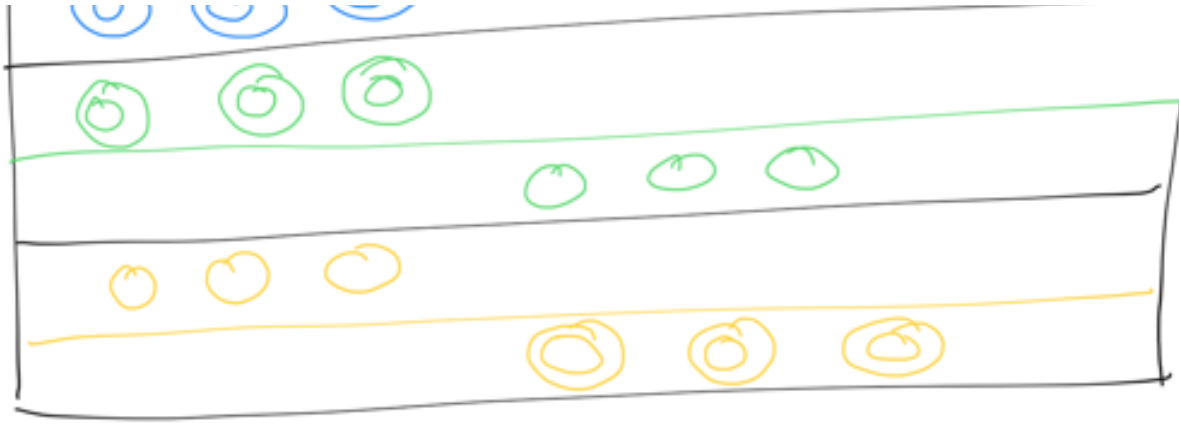
Virtual DB Nodes →



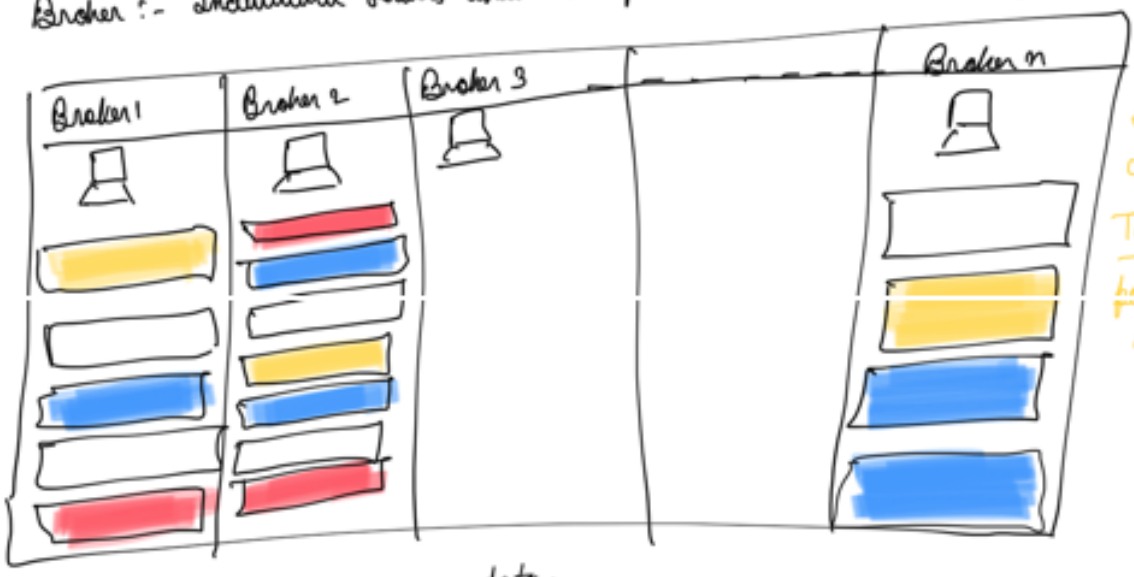
(18) KAFKA →

→ A motivational Example


producer
75' Foul
60' Penality
56' Goal
consumer
producer
75' Foul (F)
60' Penality (P)
56' Goal (F)
19' Fed (T)
11' Fault (Tennis)
consumer
Producer
Four (C)
cricket (C)
Foul (F)
Fault (T)
consumer
Bowled (C)
Six (C)
Penality (P)
Goal (F)
Set (T)
Producer
56' Goal (F)
60' Penality (F)
75' Foul (F)
consumer
7 S (C)
13 FC (C)
19' Set
11' Fault



Broker :- Individual servers that make up the cluster and store information.

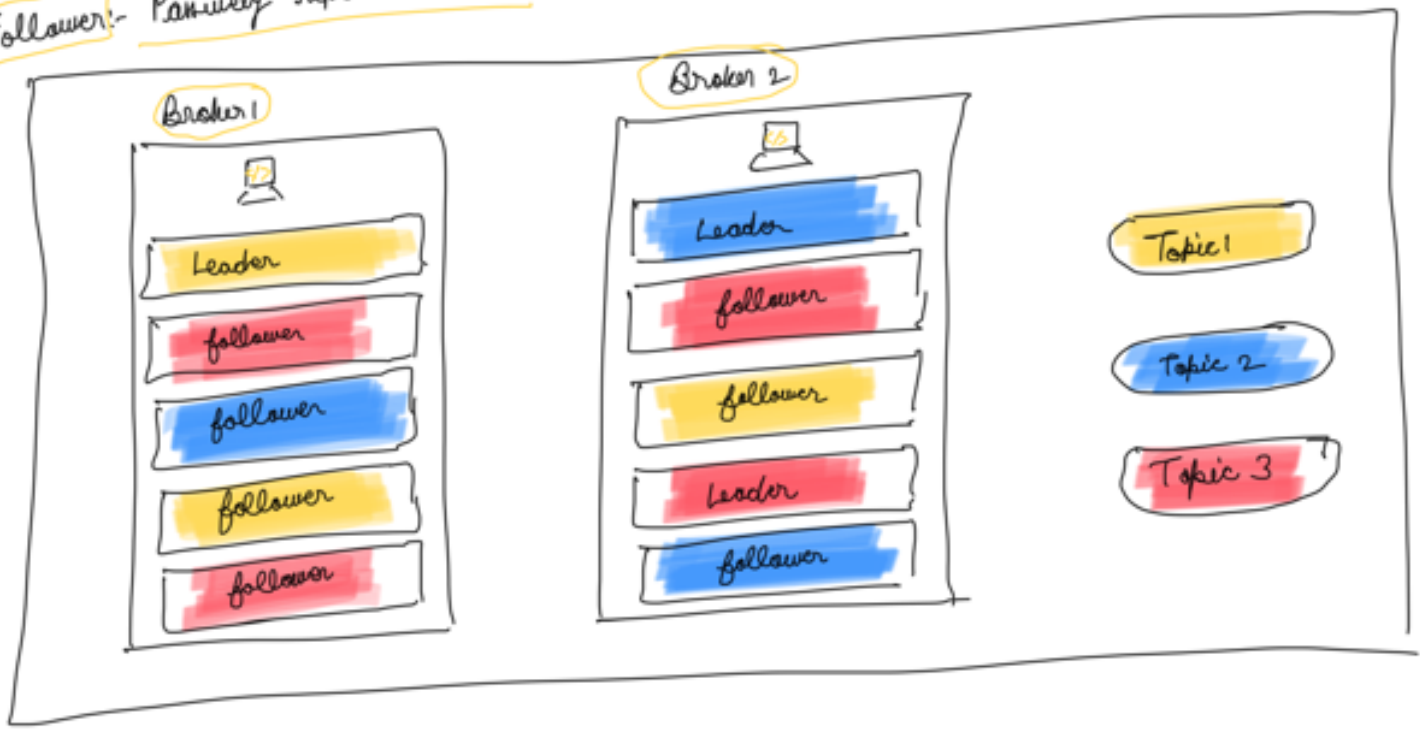


Partitions → Ordered immutable sequence of message that are continually appended to

Topics → Logical grouping of partitions. You publish & subscribe to topic.

Leader: Handles all incoming data.

Follower:- Passively replicate leader

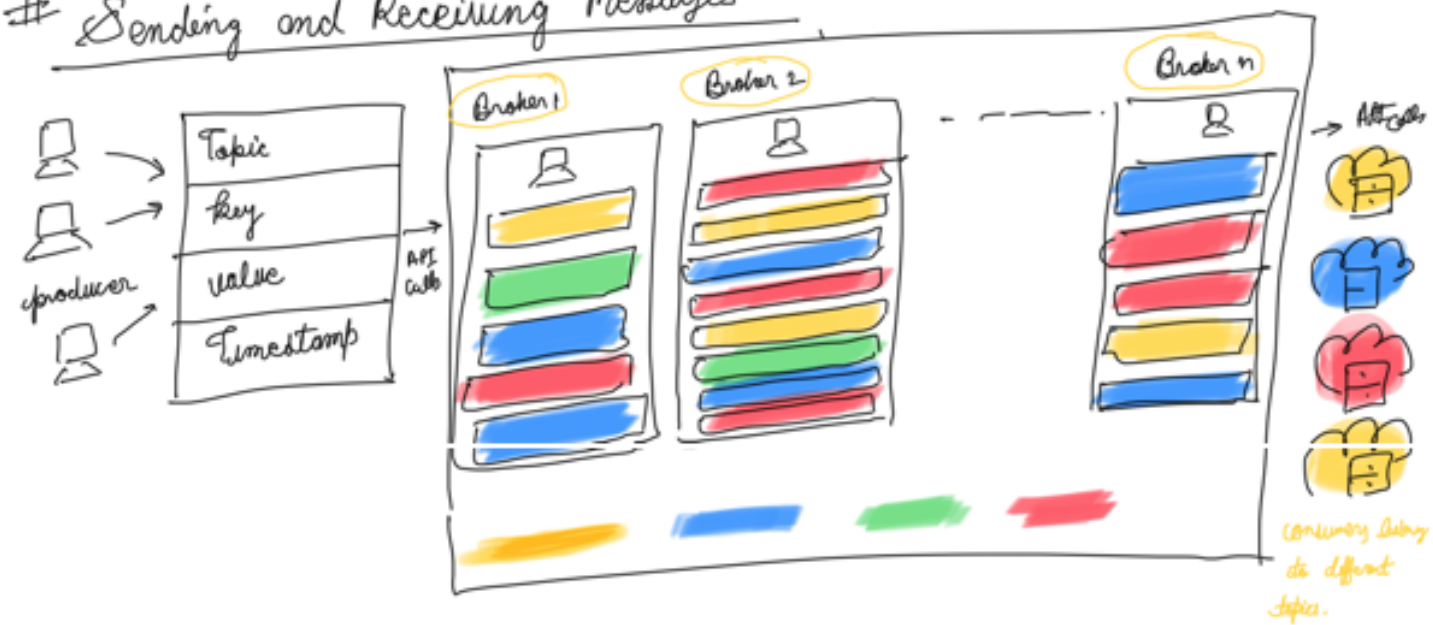


Topic 1 → 1 Leader
2 follower

Topic 2 → 1 Leader
2 follower

Topic 3 → 1 Leader
3 follower

Sending and Receiving Messages ⇒

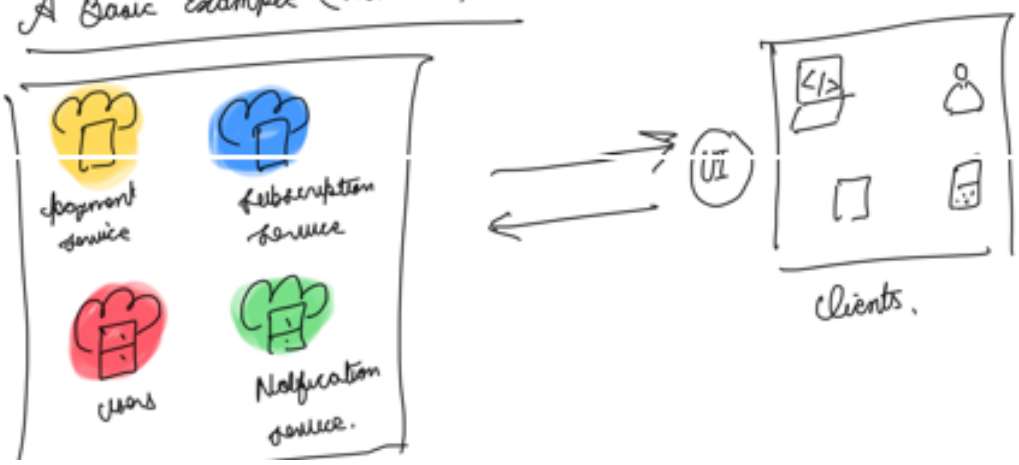


---

(19) API Gateway →

A Basic Example ⇒

A Basic Example (Zoom In) →

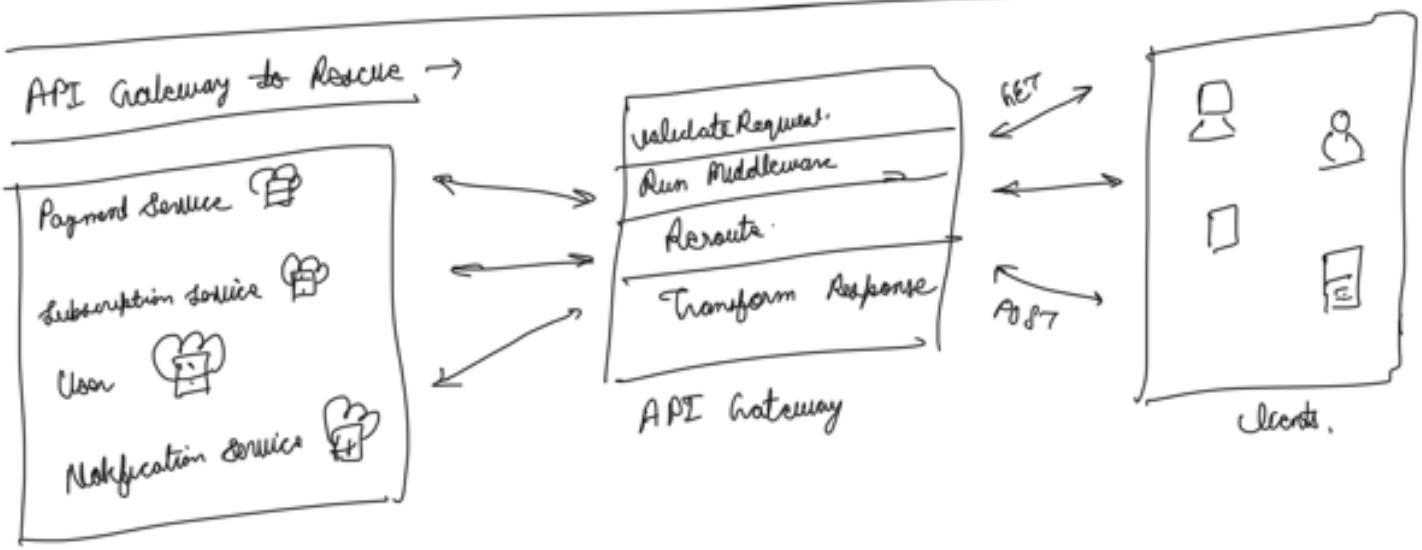


Microservice

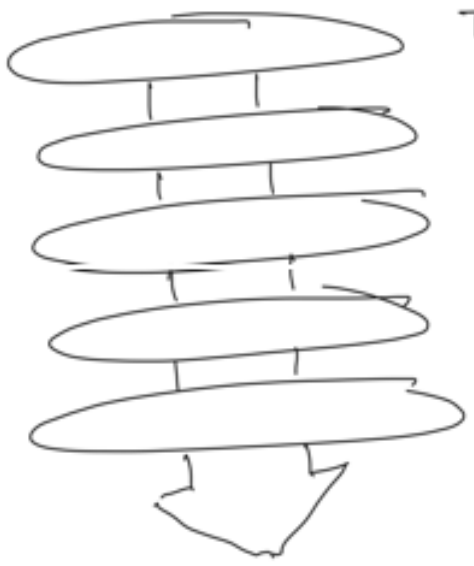
A chaos of Microservice →

→ chaotic direct connection
→ No unified Monitoring.
→ Scaling nightmares
→ Security Vulnerabilities.
→ Inefficient data aggregation
→ Redundant logic in service

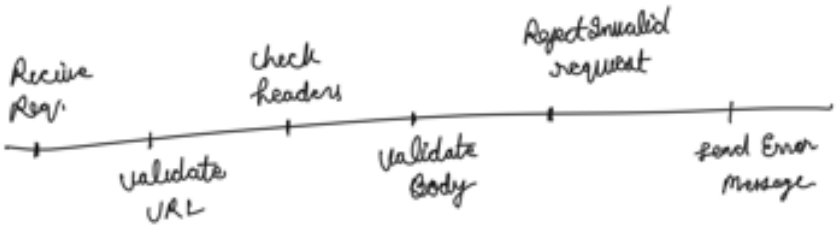
API Gateway to Rescue →



Filtering Visualized →



Request Validation process →



Middle Ware →

Monitoring
Complaince
Security
Performance.

→ Authenticate req. using JWT Tokens
→ Limit request rates to prevent abuse.
→ Terminate SSL connections.
→ Log and monitor traffic
→ Compress responses
→ Handles CORS headers
→ Whitelist / Blacklist IPs
→ validate request size
→ Handle response timeouts
→ Version APIs
→ Throttle traffic
→ Integrate with service discovery

API Gateway

Pros	Cons
→ centralized security	→ Operational Complexity
→ Reduced Coupling	→ cost.
→ Improved performance	→ Latency
→ Scalability	
→ Fault Tolerance	

→ Fault tolerance

Which API Gateway should we choose ?

Managed Service → offers seamless integration and ease of use but is more expensive. (Amazon API Gateway) (Azure API)

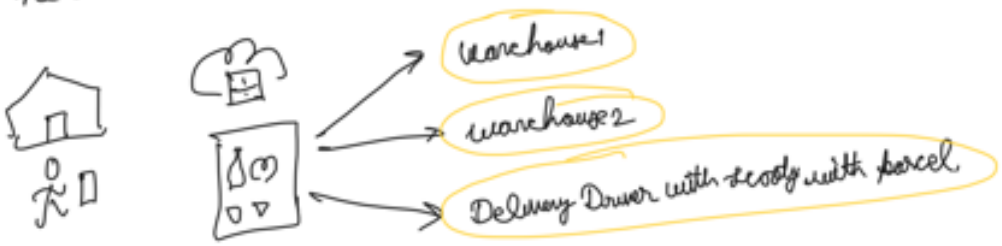
Open source soln → Provides more control and flexibility but requires more setup.

TYK & KONG

---

20 Design a Quick-Commerce Service →

The 30 minute magic.



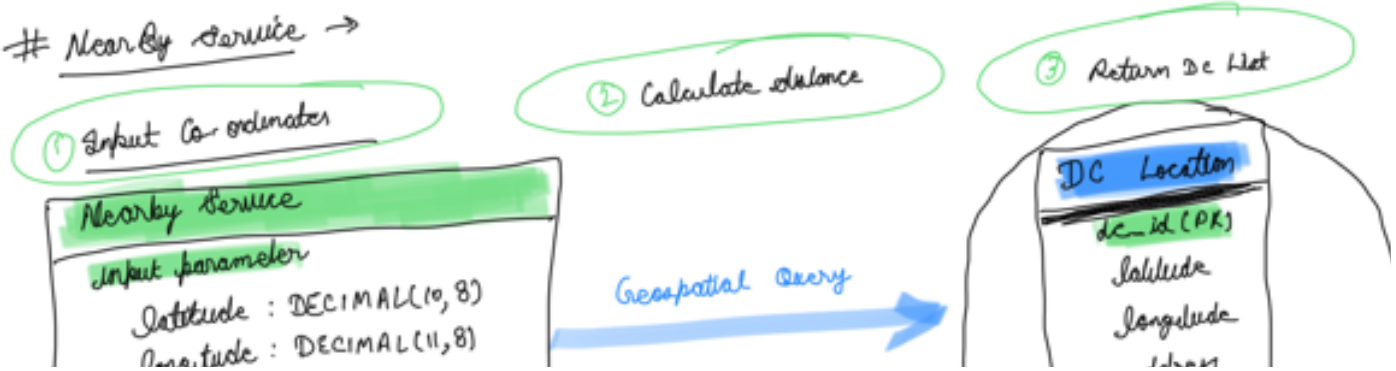
System Requirements →

→ Availability connections → Connect to any data centre within one hour
→ Ordering Consistency → Ordering must be strongly consistent across all system
→ Order Volume → System must be handle one million orders per day
→ Order Items → Ability to order items through the system
→ Availability Speed → Fast availability with 100 ms response time
→ System Size → Ten Thousand data centres, one hundred thousand catalog items
→ Query Availability → Query availability by location within one hour.

Inventory Management Components →

(i) Inventory → Physical Item instance at DC
(ii) Item → Type of Item e.g → Cheetos.
(iii) Distribution Center → Physical location where items stored
(iv) Order → Collection of inventory ordered by user.

Nearby Service →



radius_km : INTEGER

Processing:
→ Haversine distance formula
→ Earth curvature compensation

DC List Response

address
capacity
region

Availability Service →

Availability Service

Input parameter
dc_ids : [Array<UUID>]
item_filter : String (optional)
category : String (optional)

Processing
→ JOIN Items and inventories table
→ FILTER By DC IDs from nearby service

Inventory Query [DC_ID]

Available Items [ITEM : QTY]

Distribution Center Database

Items	Inventory
id	id
name	item_id
description	quantity
category	reserved qty
price	available qty

join on (id)

Order Service

Order Request
customer ID (UUID)
Items : [{ item, quantity }]
dc_id : UUID

Atomic Transaction
check inventory
create order + order items
update inventory status

Atomic transaction

Order confirmation

PostgreSQL Database
ACID Transaction + Row-level Locking

Orders	Order_Items
order ID	OrderItem ID
customerID	order ID
dc_id	item ID
status	quantity
total amount	unit price

Inventory
Inventory_id
item_id
dc_id
quantity
reserved qty
status

Tying Together →

Clients, Application

HTTP/REST

API Gateway
Authentication
Rate limiting
Load Balancing
Request Routing
port : 80 80

check stock

find DC

place order

Availability service
(Real time inventory check)

Location Service
(find nearest DCs)

order service
process order and payments

query inventory

Location lookup

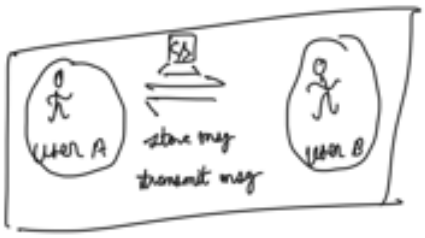
create transaction

Items	Inventory
Orders	OrderItems

Distribution Center

(21) Chat System ⇒

The Basic Need →

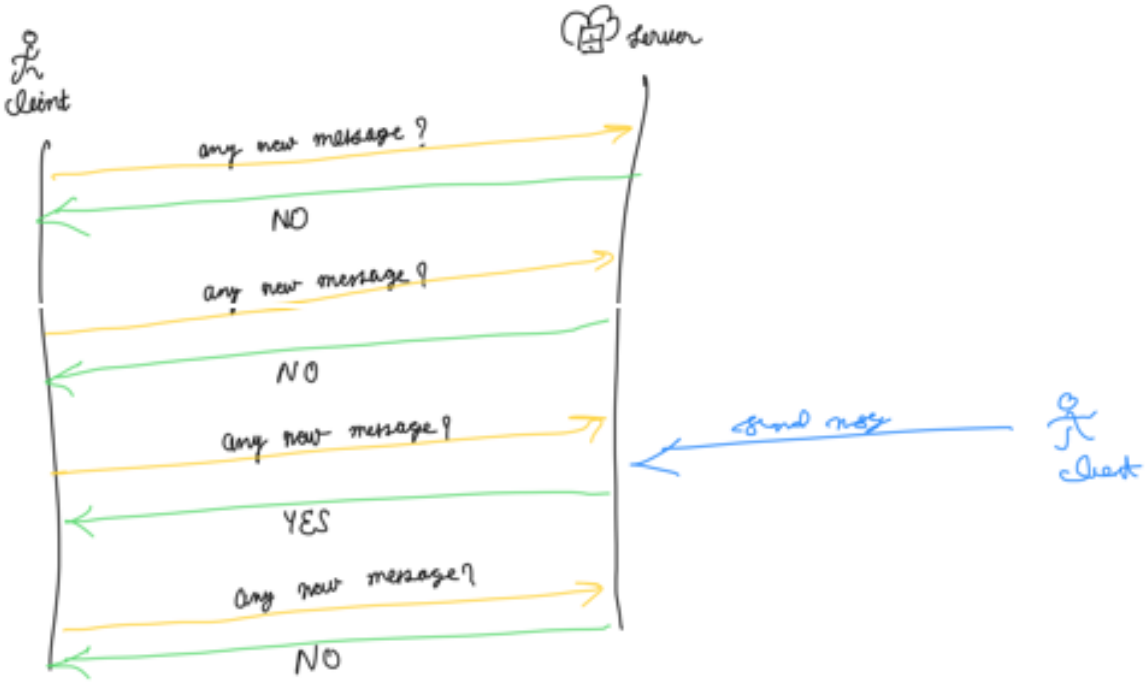


Design Requirement

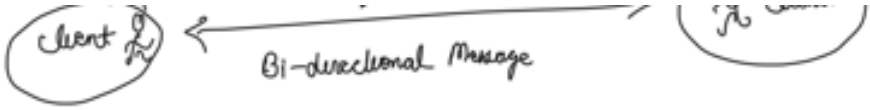
① One on One Chat → focuses on personal communication with low latency.
② Group Chat → Support interaction among up to 100 members
③ Mobile App → Accessible on smartphone and tablets
④ Web App → Accessible through web browsers
⑤ Large Scale → Designed to support 50 million daily active users.

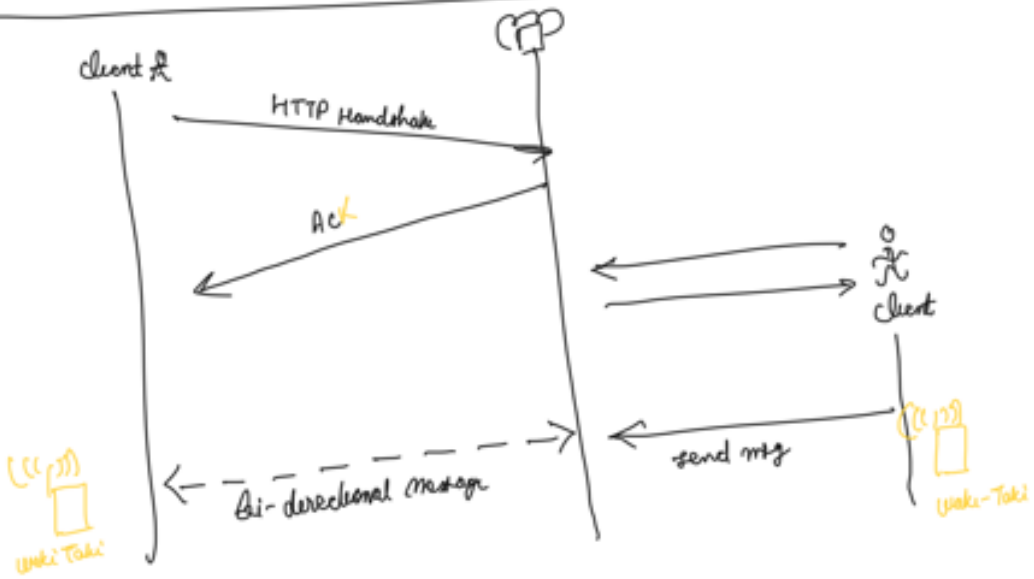
Communication Technique →

Web Socket → A persistent connection for real time communication
Polling → Clients periodically check for new message from the server.
Long polling → Clients wait for a server response before checking again.



✱ Web Socket →





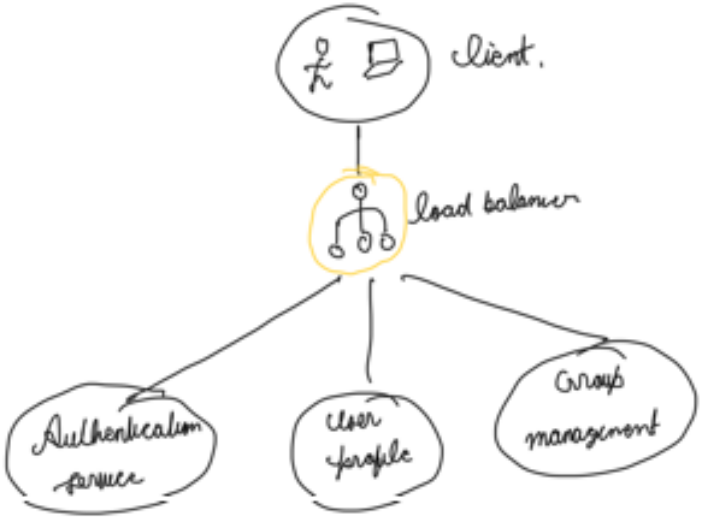
Stateless System →

Each request from the user is independent. Server does not remember any previous request.

→ All information must be sent with every request.

→ No session information

→ Easier to scale and load balance



Stateful System →

Server remembers the connection OR user session across multiple interaction

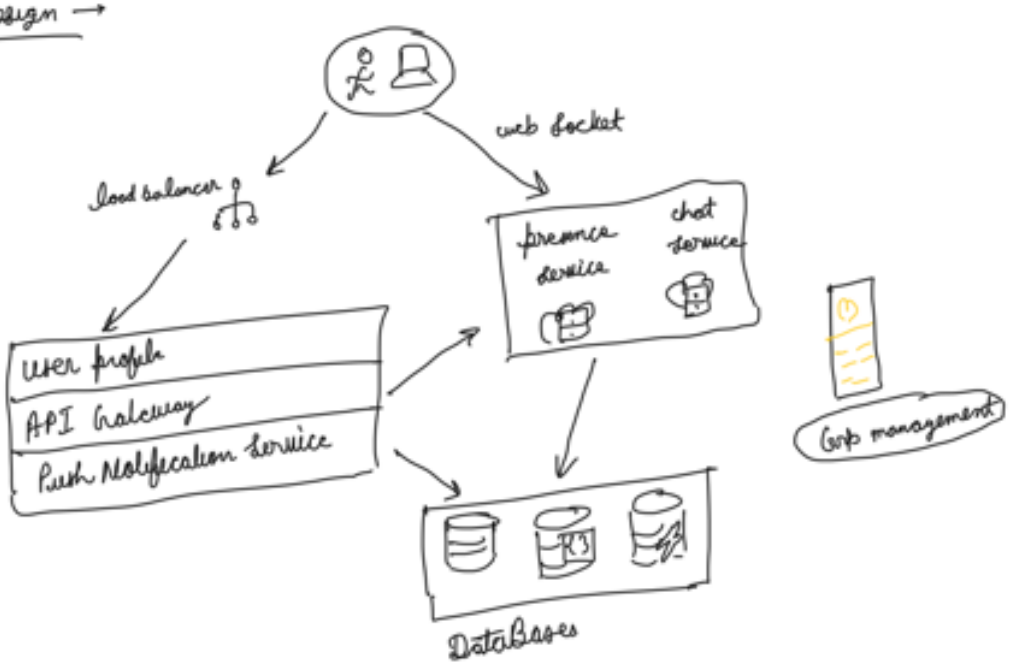
→ Maintain ongoing context

→ Usually needed for real-time communication

→ Harder to scale.

☆ High level Design →

High level Design →



22 Chat System (Part 2) →

★ Sending a Message →

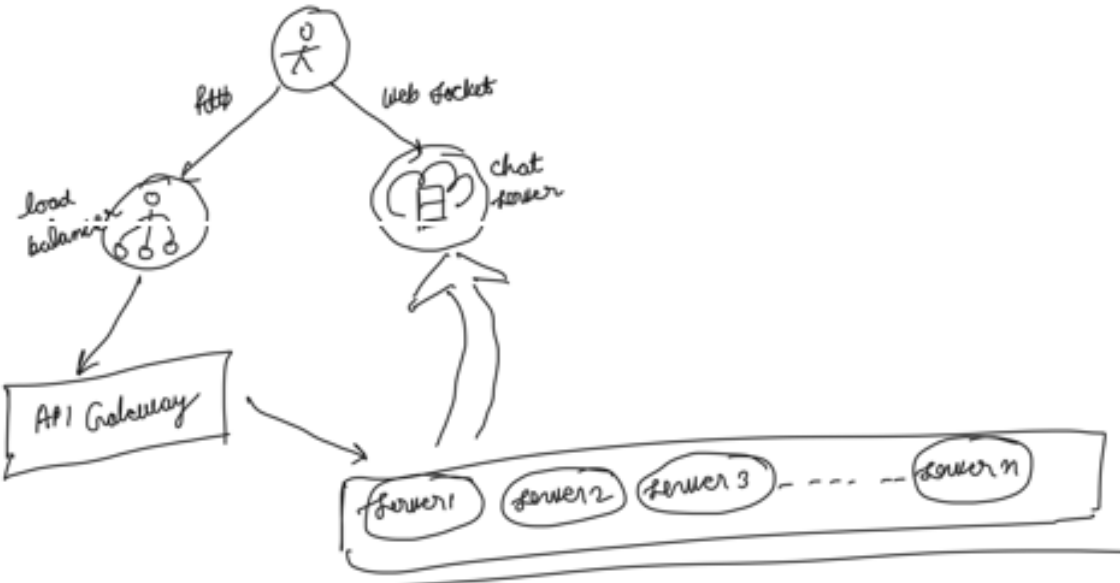
Message	
MessageID	BigInt
MessageFrom	BigINT
MessageTo	BigINT
Content	text
Created At	timestamp

→ Auto Increment → Suitable for SQL databases but not for NOSQL

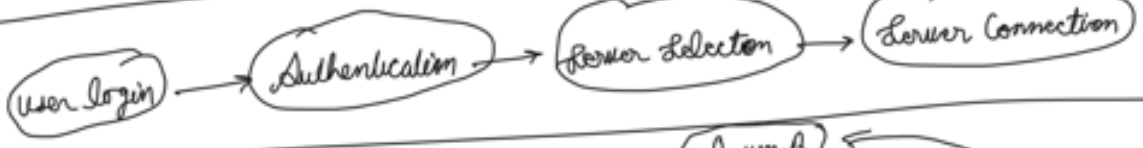
→ Global Sequence Generator → Ensures uniqueness and sortability across system.

→ Local Sequence Generator → Simpler implementation but limited access.

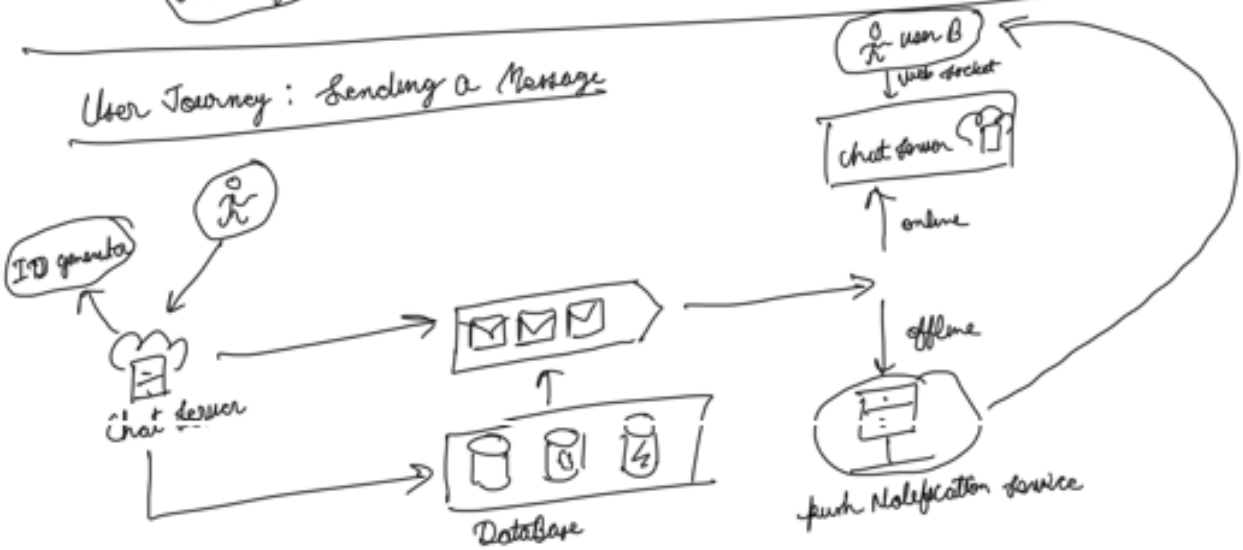
User Journey →



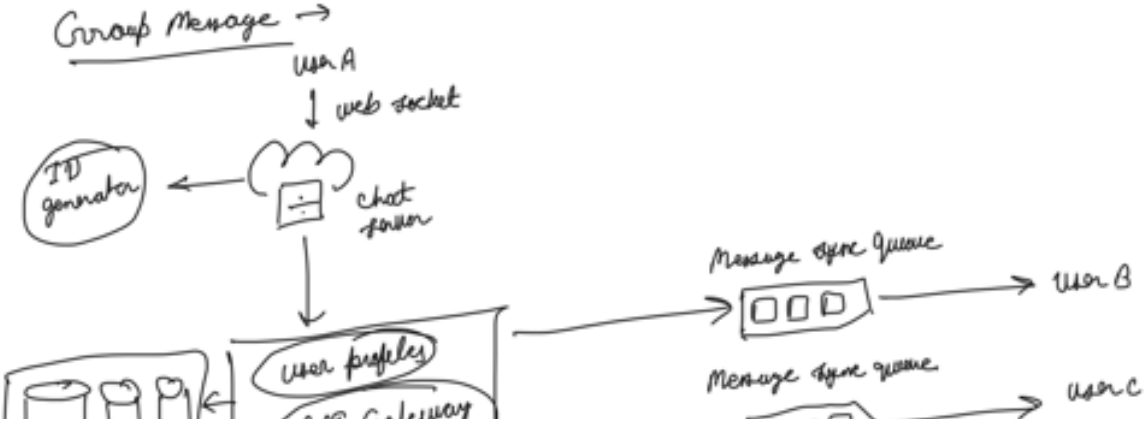
Service Discovery →



User Journey : Sending a Message

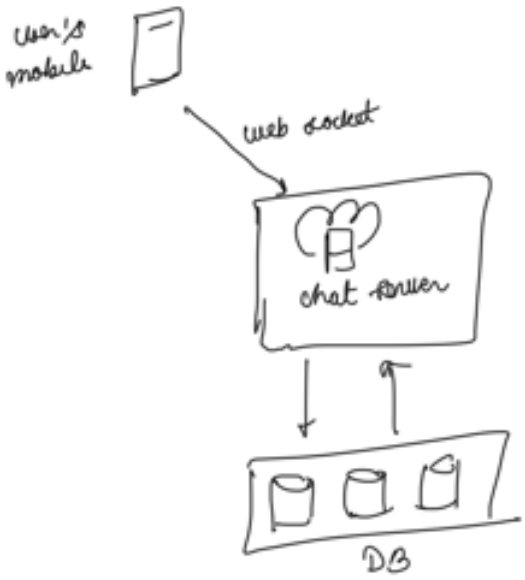


Group Message →



Synchronizing Device →

max_message_id = 765



max_message_id = 690

1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4 5 6 7 8 9 10 11 12