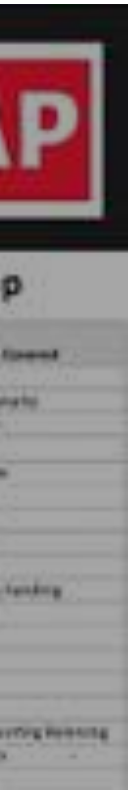
System Design1

Tuesday, 13 September 2022 1:03 AM

System design -> https://systemdesignprep.com/home
https://towardsdatascience.com/system-design-101-b8f15162ef7c
GeeksForGeeks -> https://www.geeksforgeeks.org/system-design-tutorial/?ref=ghm
Codekarle -> https://www.codekarle.com
LLD & HLD roadmap -> Ultimate LLD and HLD Roadmap | System Design RoadMap | LLD & HLD Topics to be covered for Interview

Algorithms to know
https://blog.bytebytego.com/p/algorithms-you-should-know-before

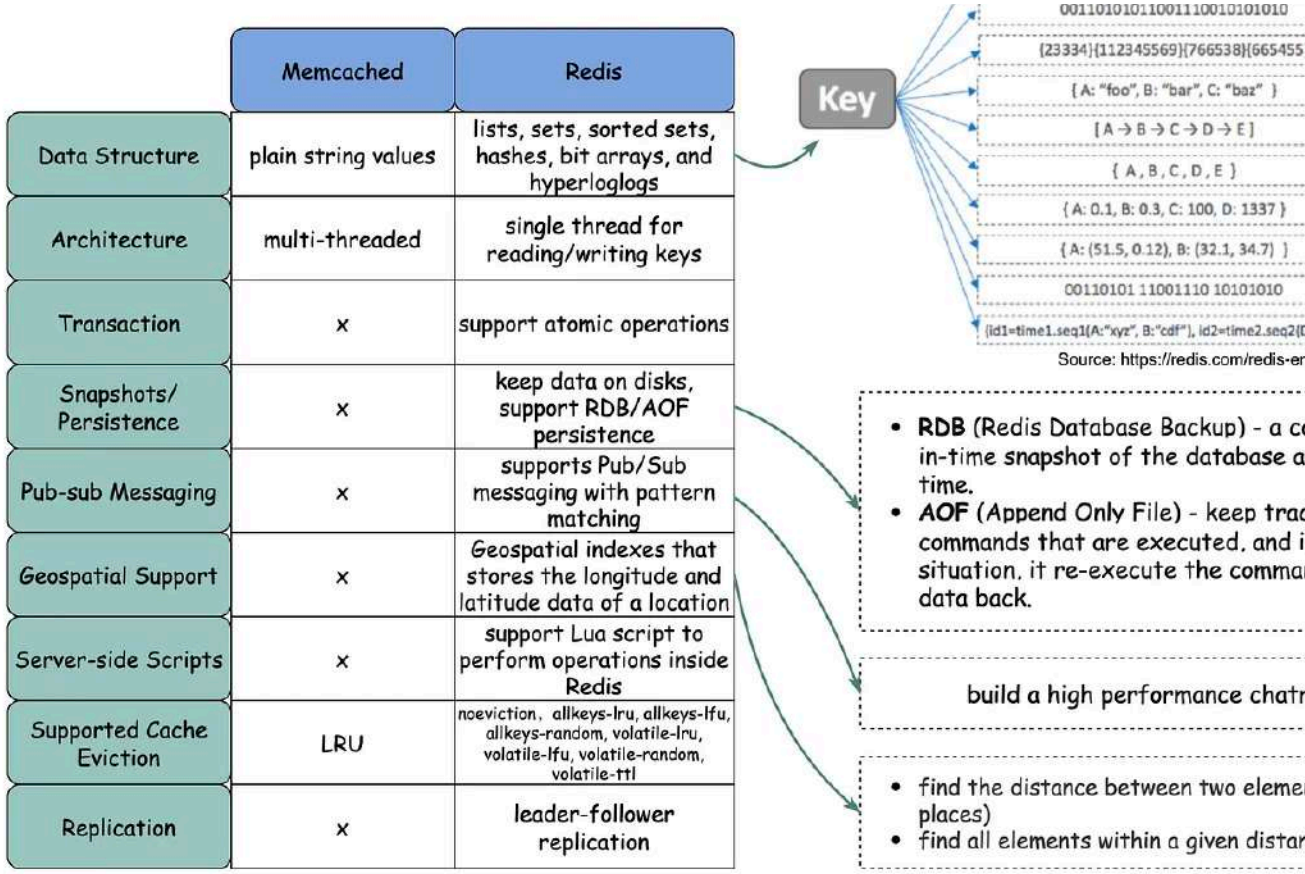
Redis vs memcache
https://blog.bytebytego.com/p/redis-vs-memcached



ByteByteGo

Strings

	Memcached	Redis
Data Structure	plain string values	lists, sets, sorted sets, hashes, bit arrays, and hyperloglogs
Architecture	multi-threaded	single thread for reading/writing keys
Transaction	x	support atomic operations
Snapshots/ Persistence	x	keep data on disks, support RDB/AOF persistence
Pub-sub Messaging	x	supports Pub/Sub messaging with pattern matching
Geospatial Support	x	Geospatial indexes that stores the longitude and latitude data of a location
Server-side Scripts	x	support Lua script to perform operations inside Redis
Supported Cache Eviction	LRU	noeviction, allkeys-lru, allkeys-lfu, allkeys-random, volatile-lru, volatile-lfu, volatile-random, volatile-ttl
Replication	x	leader-follower replication



URL Shorterner

[URL shortener system design | tinyurl system design | bitly system design](https://www.geeksforgeeks.org/system-design-url-shortening-service/)
https://www.geeksforgeeks.org/system-design-url-shortening-service/

DB: cassandra

Highly avaialble

Both Read & Write Heavy

nterprise/data-structures/

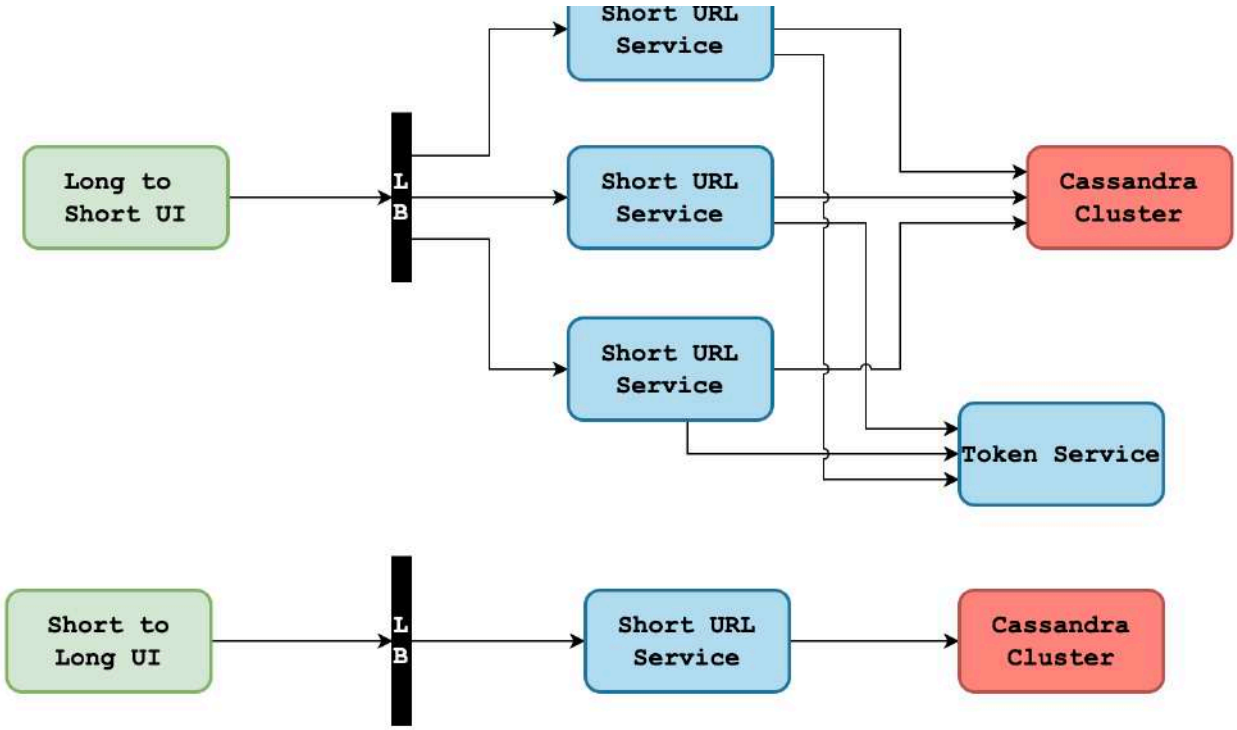
ompact, point-
t a specific

ck of all the
n a disastrous
nds to get the

room

nts (people or

nce of a point



Total short URL length: 7
Total size to store:
Long url : 2kb(2048 characters)
Short url: 17 bytes
Created_at
Expire_at

Types of encoding:

- **Base62 encoding**
- **Base10 encoding**
- **MD5 encoding**

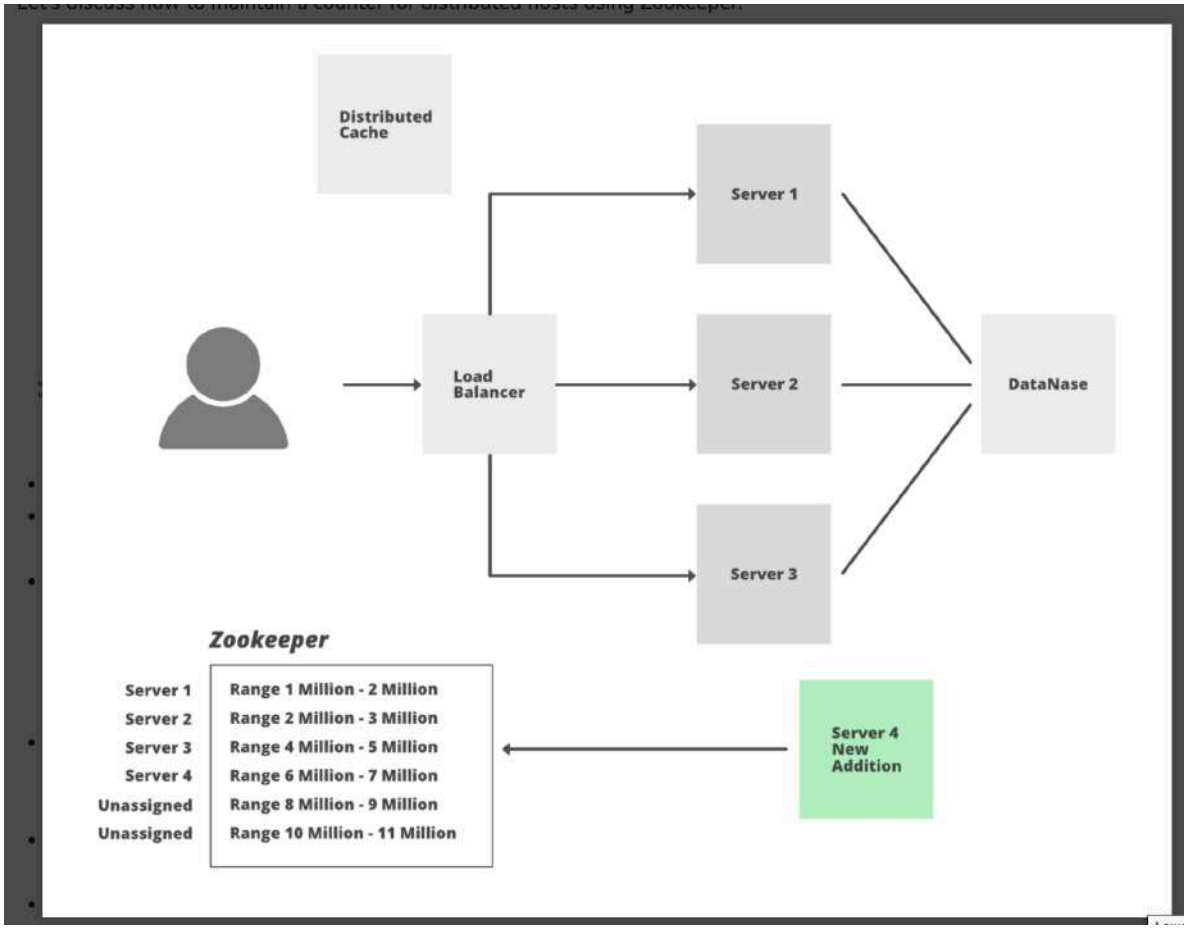
```
def to_base_62(deci):
    s = '012345689abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
    hash_str = ''
    while deci > 0:
    hash_str= s[deci % 62] + hash_str
    deci /= 62
    return hash_str

print to_base_62(999)
```

Random number to be passed for each request.. So couter based approach works fine.

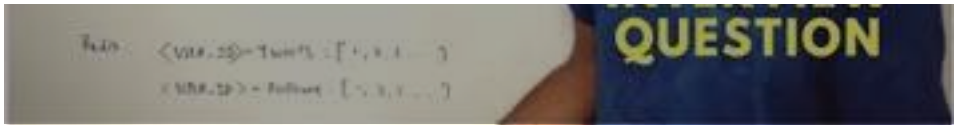
URL shortening logic:

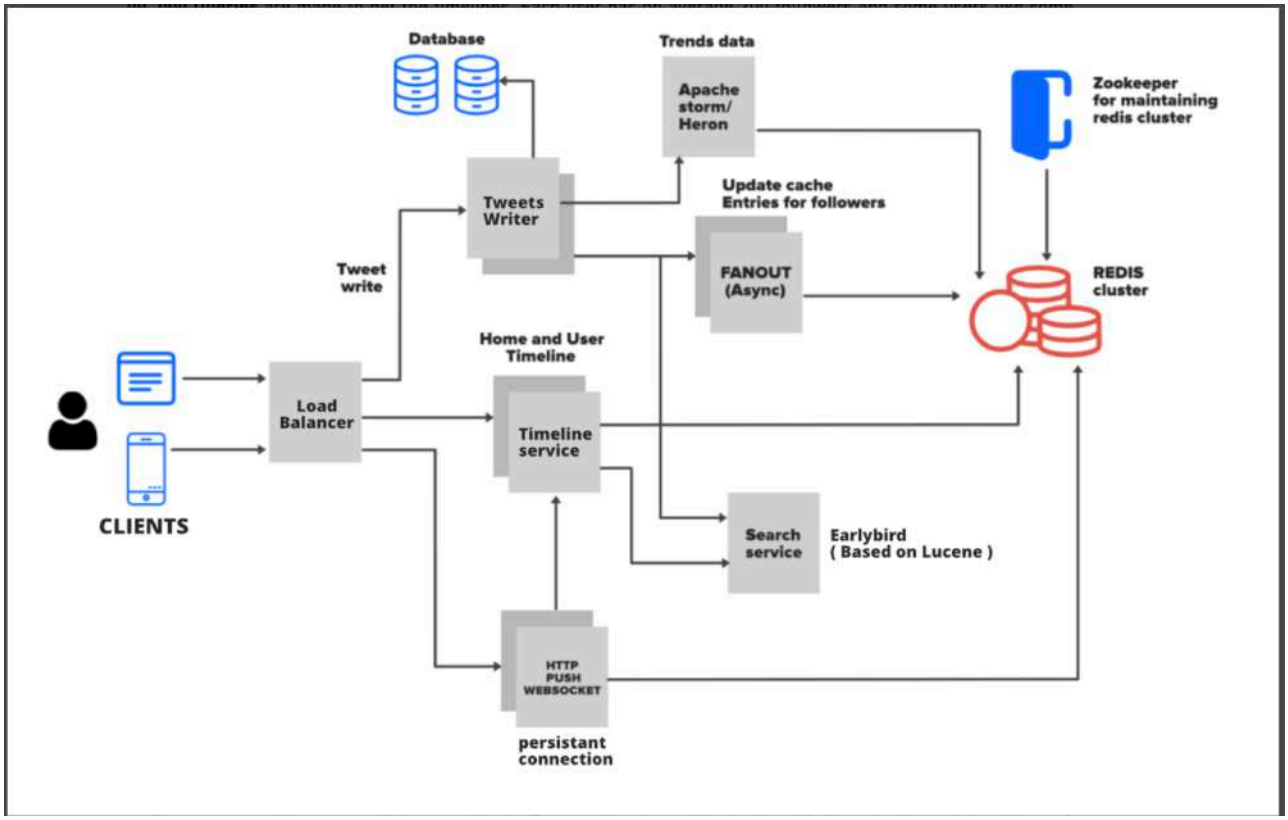
- **Single server**
- **MD5 approach**
- **Counter based distributed logic**



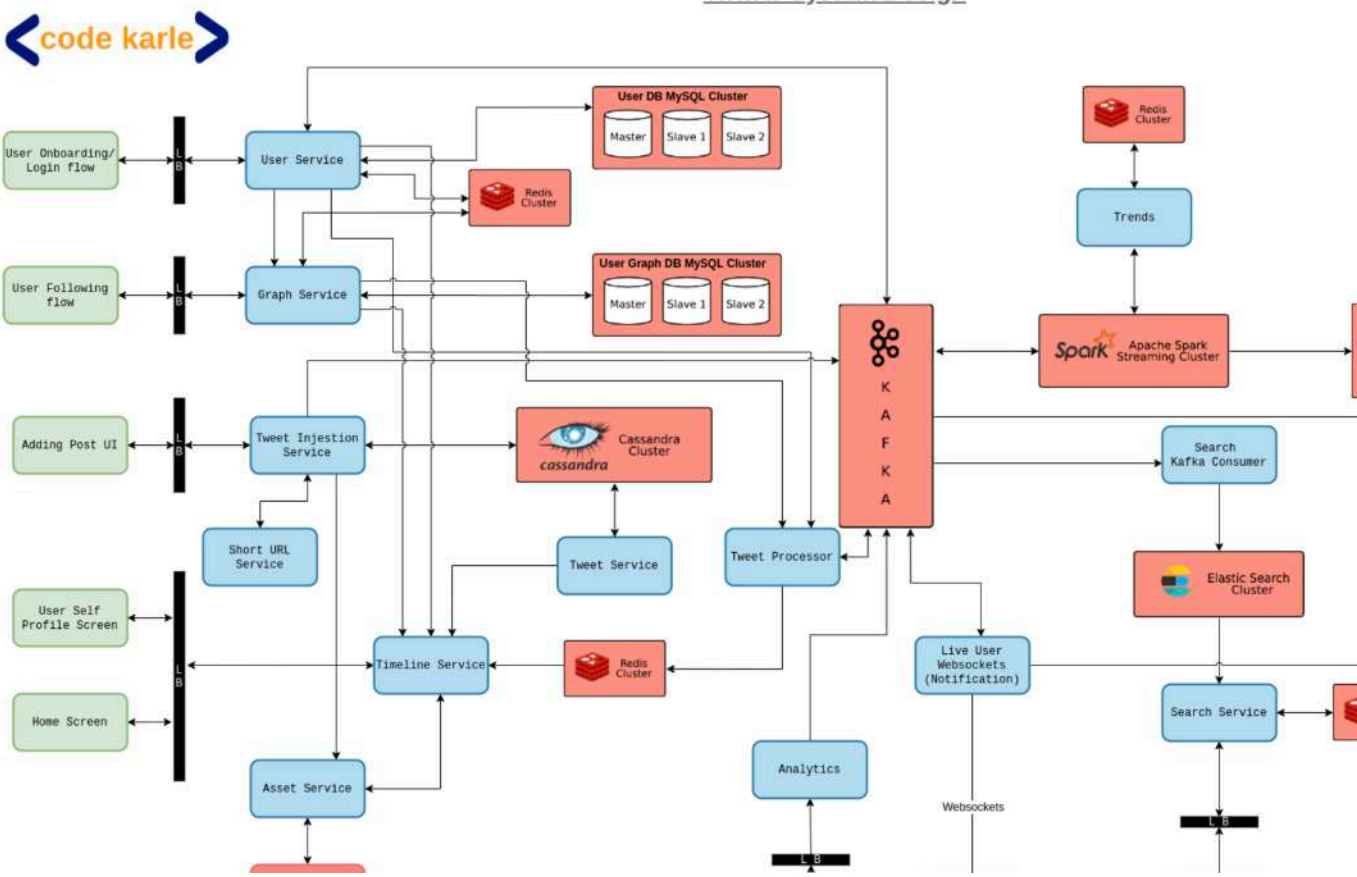
Twitter Design

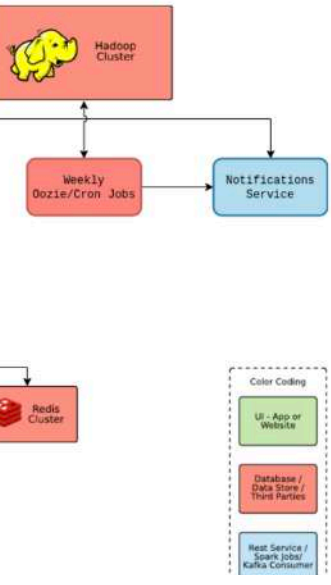
[Twitter system design | twitter Software architecture | twitter interview questions](https://www.geeksforgeeks.org/design-twitter-a-system-design-interview-question/)
https://www.geeksforgeeks.org/design-twitter-a-system-design-interview-question/

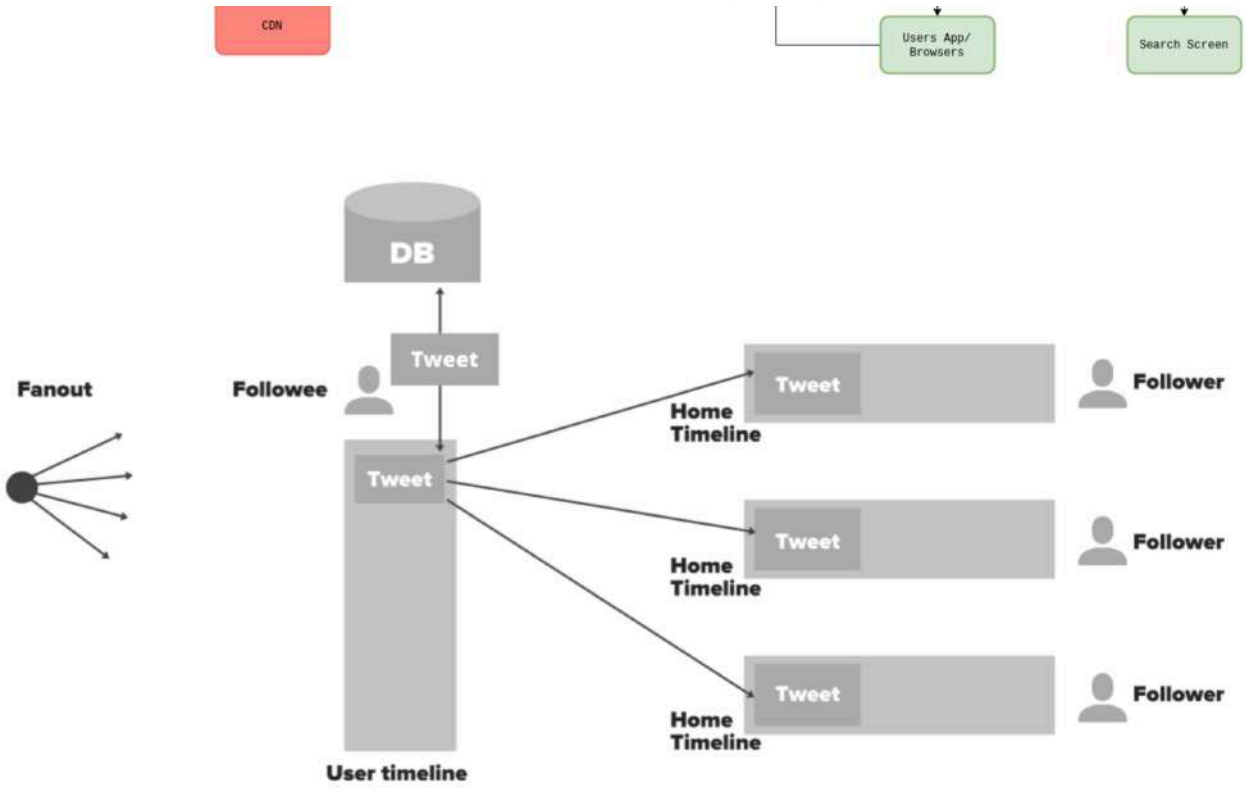




Twitter System Design

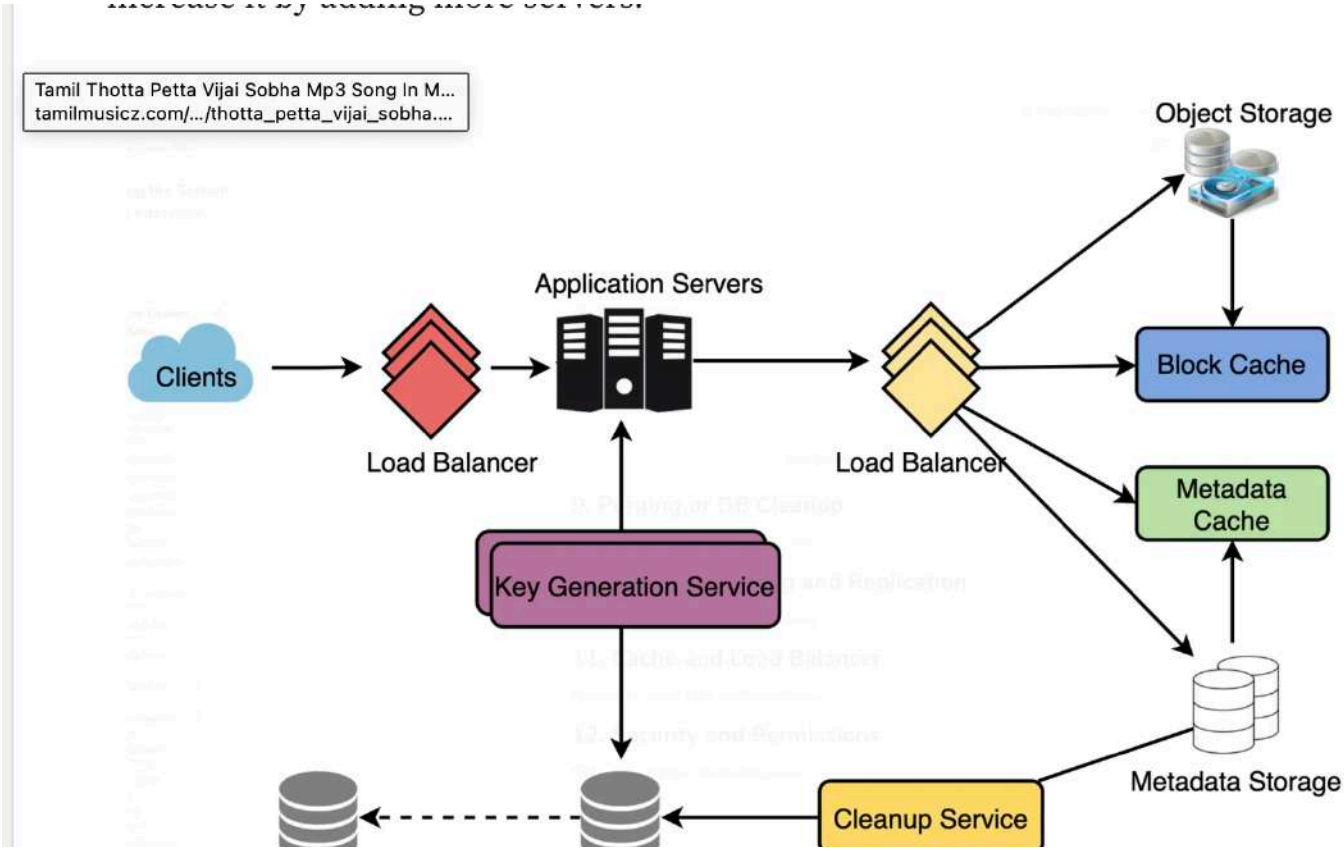



Hadoop
Cluster
Weekly
Oozie/Cron Jobs
Notifications
Service
Redis
Cluster
Color Coding
UI - App or
Website
Database /
Data Store /
Third Parties
Rest Service /
Spark Jobs/
Kafka Consumer

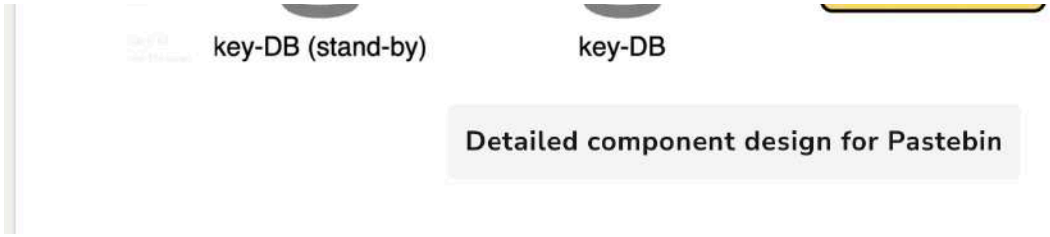


Proximity Service Design|Yelp
https://astikanand.github.io/techblogs/high-level-system-design/design-yelp-or-nearby

Pastebin system design:

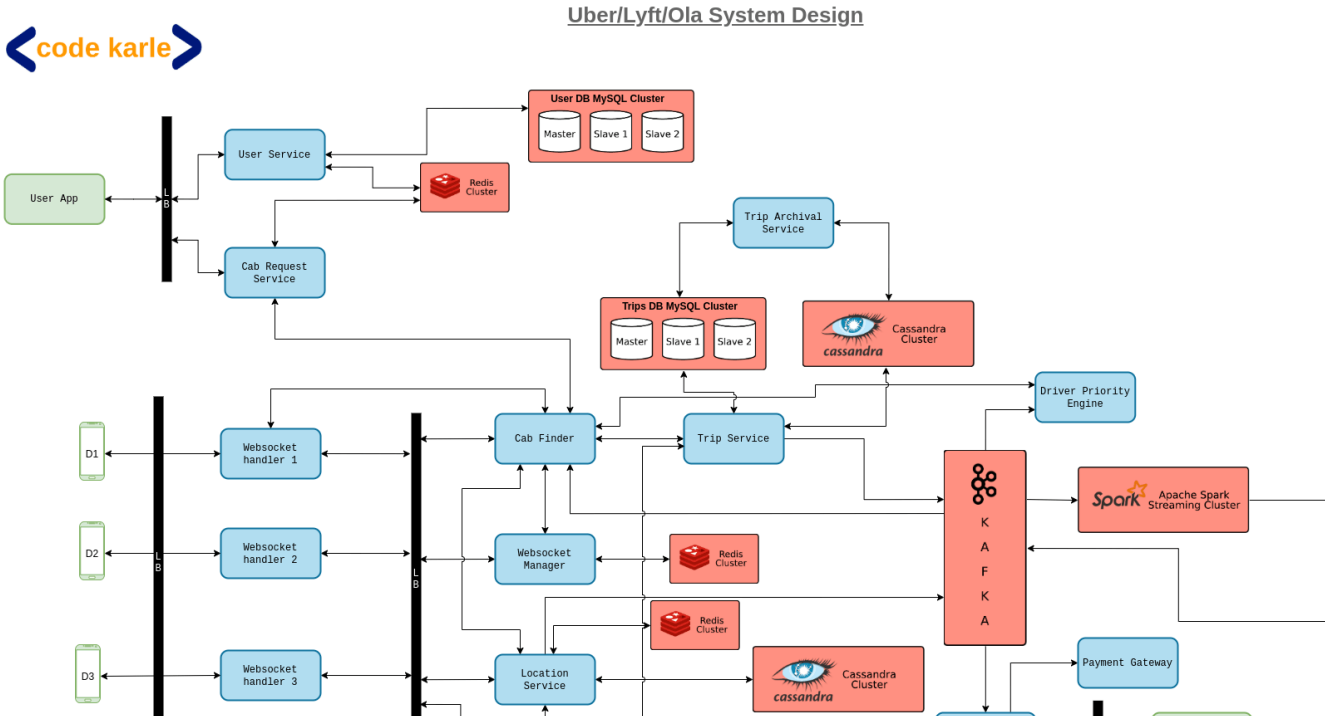


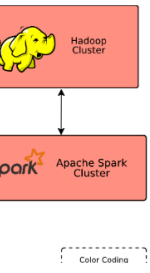

Load Balancer /
Authentication +
Authorization
Layer

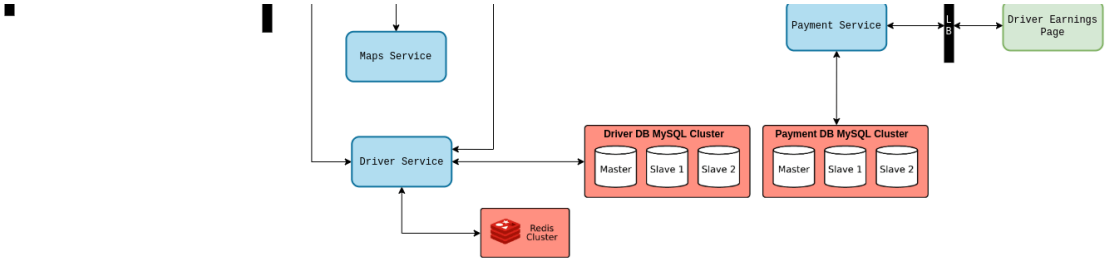


Uber System design

1. [UBER System design | OLA system design | uber architecture | amazon interview question](https://www.geeksforgeeks.org/system-design-of-uber-app-uber-system-architecture/)
2. https://github.com/codekarle/system-design/blob/master/system-design-prep-material/architecture-diagrams/Uber%20System%20Design.png




Hadoop
Cluster
Apache Spark
Cluster
Color Coding



Google Search Engine design | ElasticSearch

How Google searches one document among Billions of documents quickly?

Google indexing search

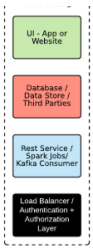
a. https://eileen-code4fun.medium.com/system-design-interview-mini-google-search-6fd319cd66ca
b. https://medium.com/double-pointer/system-design-interview-search-engine-edb66b64fd5e

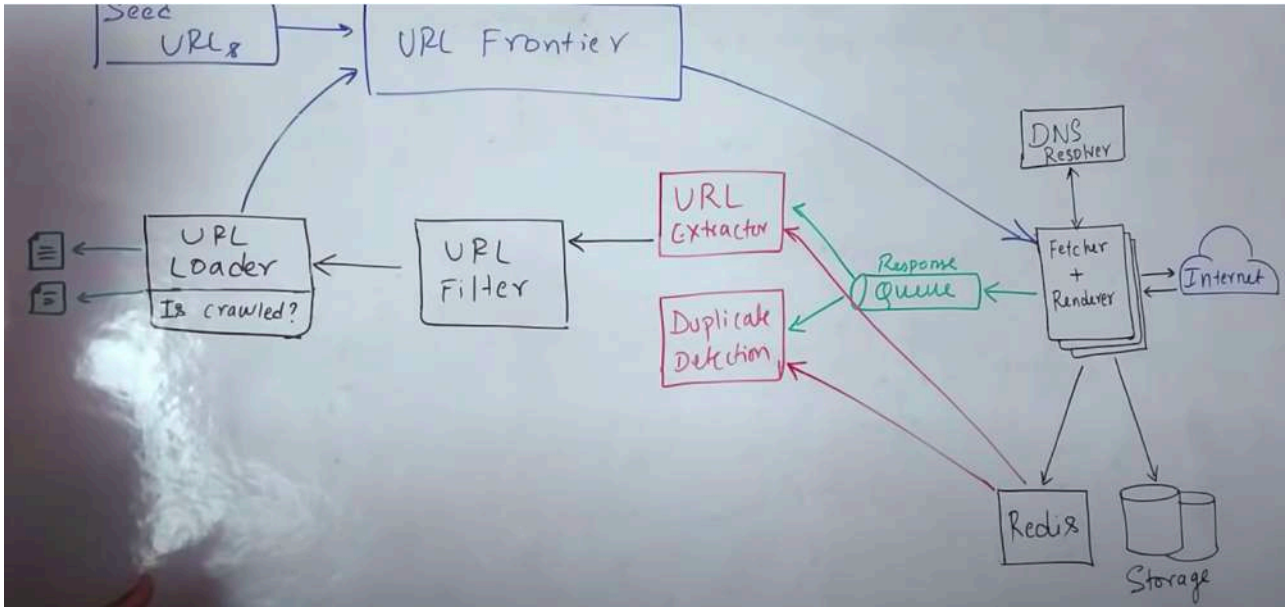
Google crawler:

https://medium.com/double-pointer/top-5-videos-for-web-crawler-system-design-interview-75b7ac9c04ce

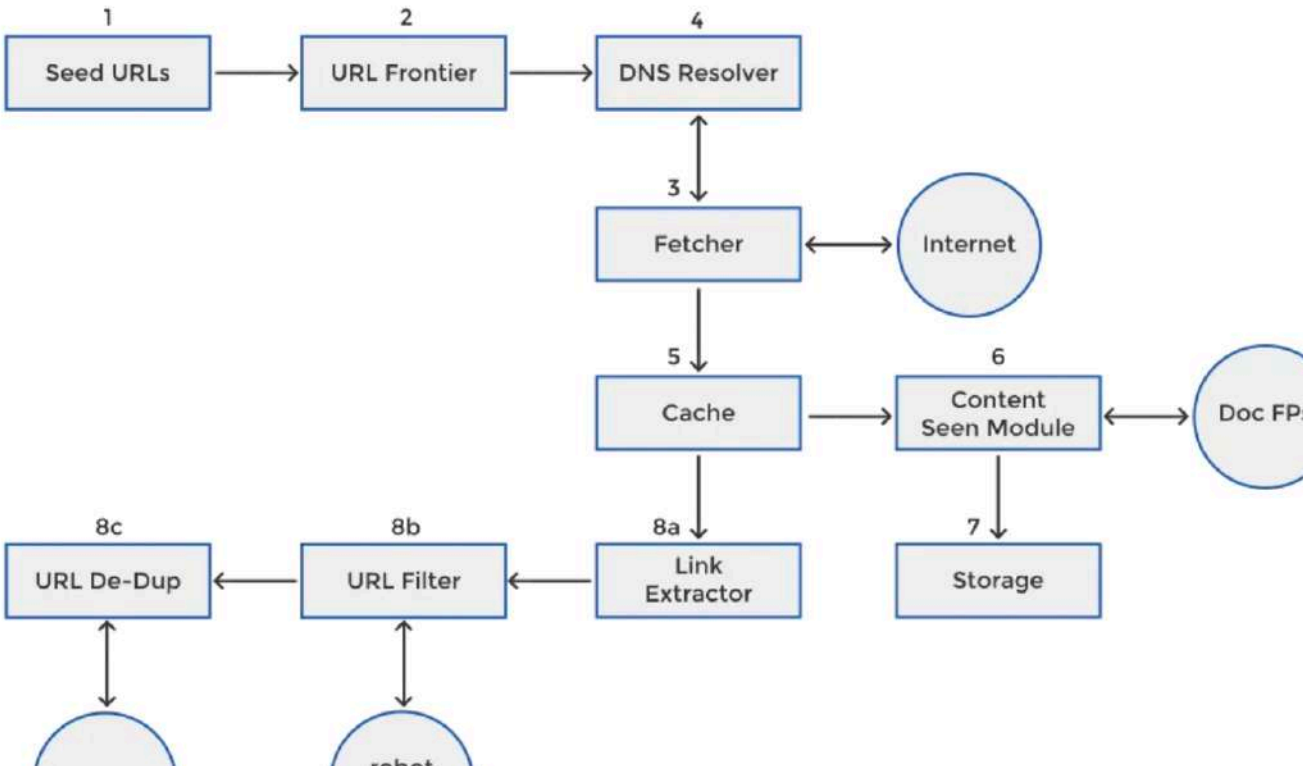
System Design distributed web crawler to crawl Billions of web pages | web crawler system design

https://www.enjoyalgorithms.com/blog/web-crawler


UI - App or Website
Database / Data Store / Third Parties
Rest Service / Spark Jobs/ Kafka Consumer
Load Balancer / Authentication + Authorization Layer



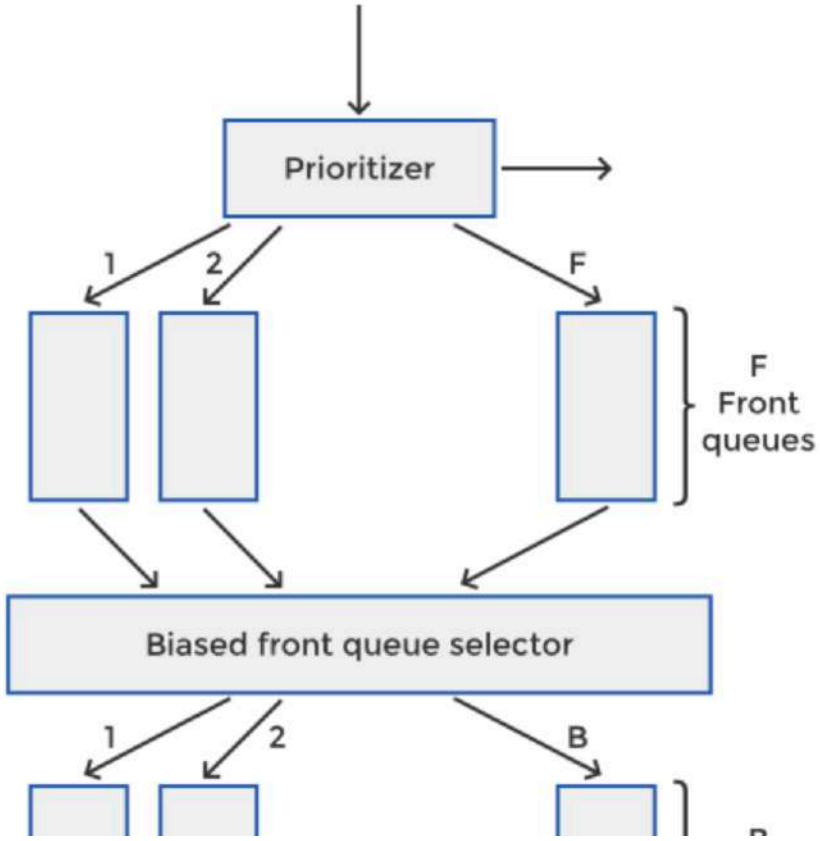
Design Diagram

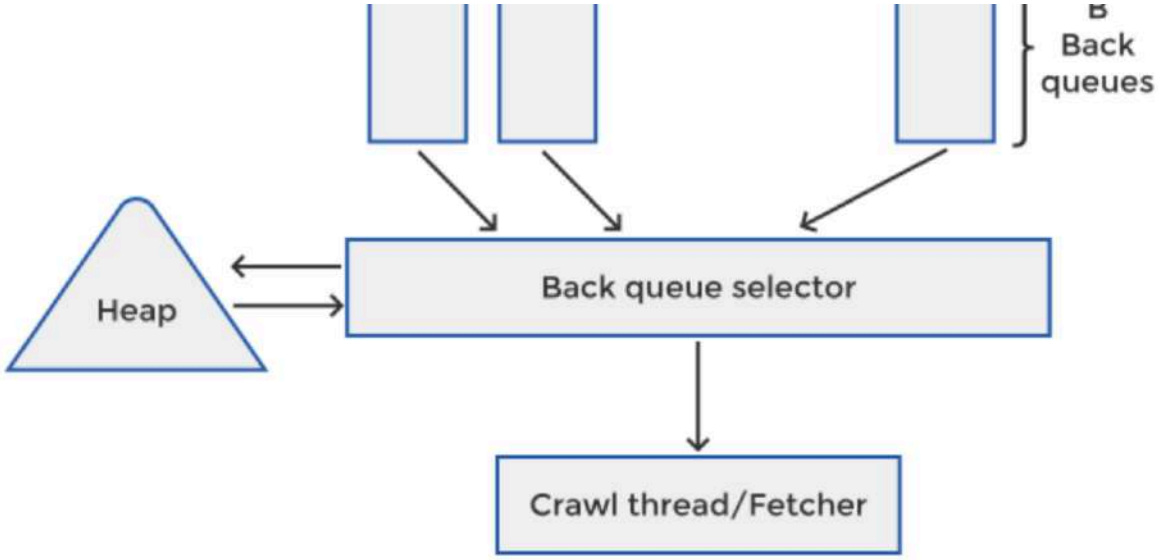


Design Diagram

- Seed URLs
- URL Frontier -> pushes the URL into the queue which will be fetched by URL fetcher
- URL Fetcher -> contacts DNS resolver
    - Download the contents
    - Cache it
    - Content seen module -> compare It with Doc FPS to check the uniqueness of it
- Link Extractor
- URL Filter
- URL De-Dup Test
    - All the passed URLs will get added to URL frontier again
    - Test will be conducted using bloom filter for faster validation
- Crawler vs scraper
    - Crawler - scans the entire page and extracts the URLs out of it
    - Scraper -> scans the particular portion of the page and extract certain required information only.

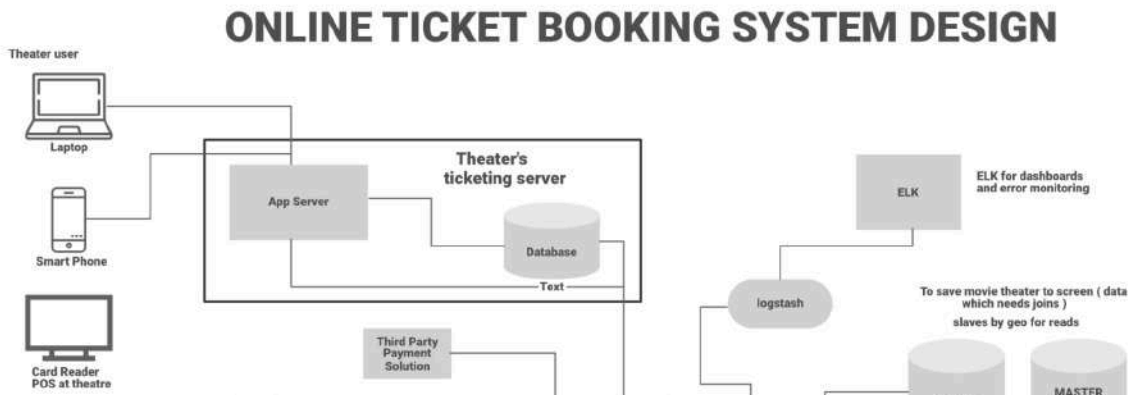
URL Frontier Architecture Diagram

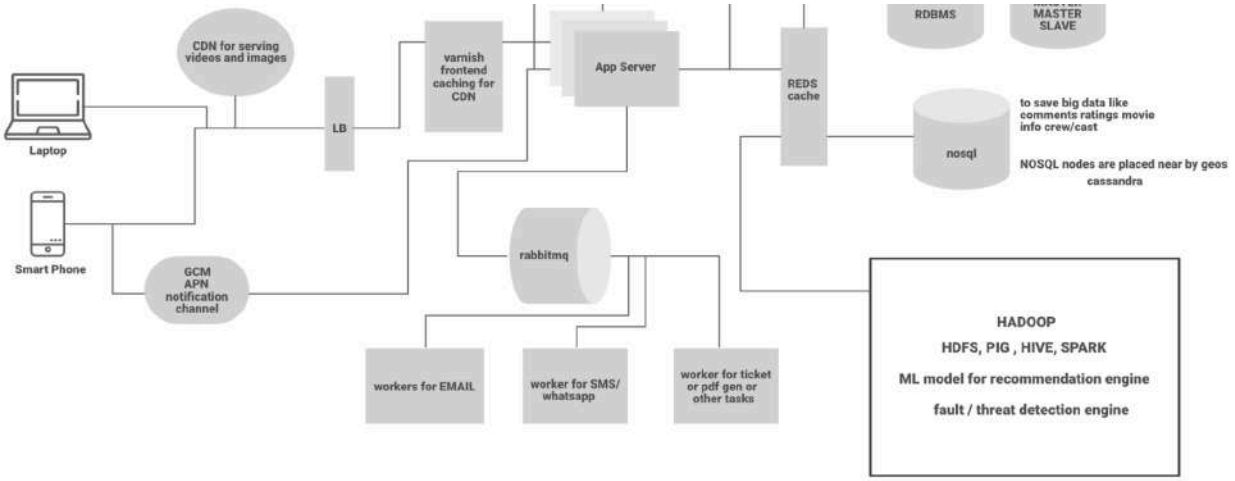




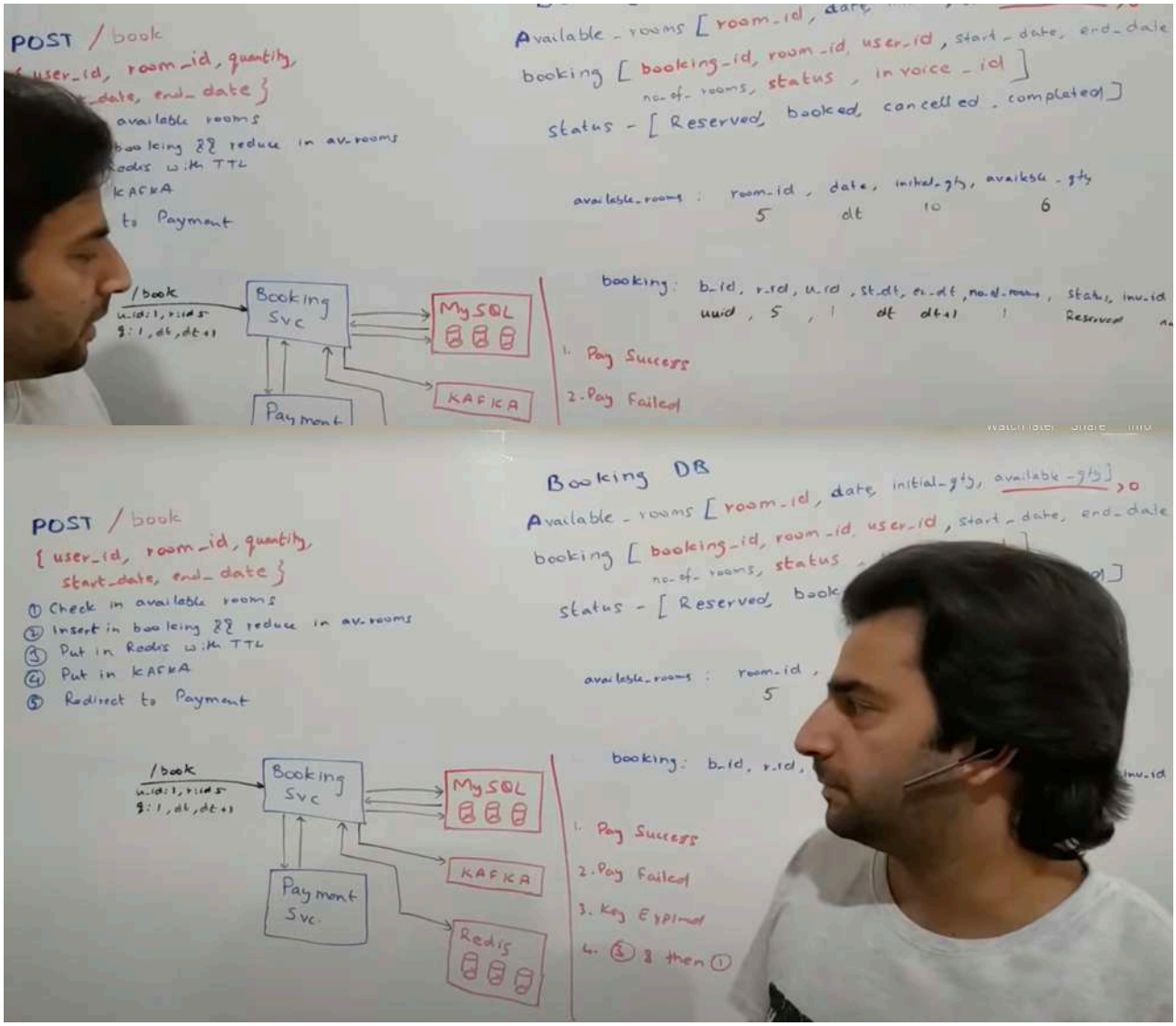
BookMyShow Design

[BOOKMYSHOW System Design, FANDANGO System Design | Software architecture for online ticket booking]
https://www.geeksforgeeks.org/design-bookmyshow-a-system-design-interview-question/
https://www.codekarle.com/system-design/Airbnb-system-design.html





Airbnb/Bookings.com design

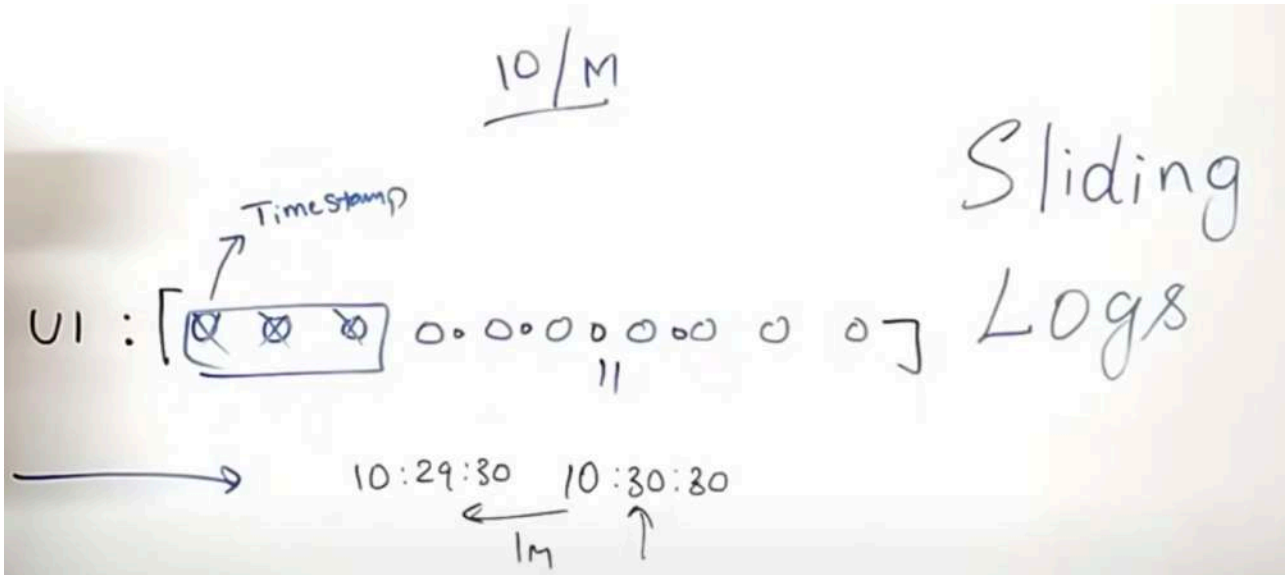


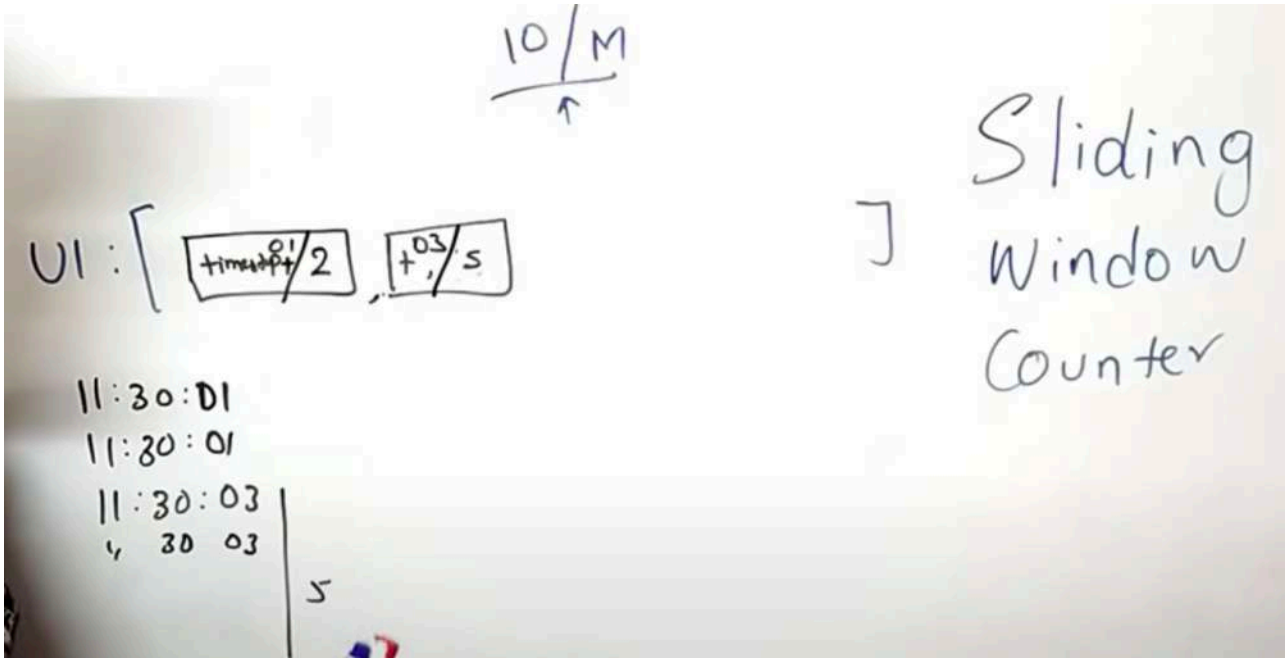
Rate Limiting system design | TOKEN BUCKET, Leaky Bucket, Sliding Logs

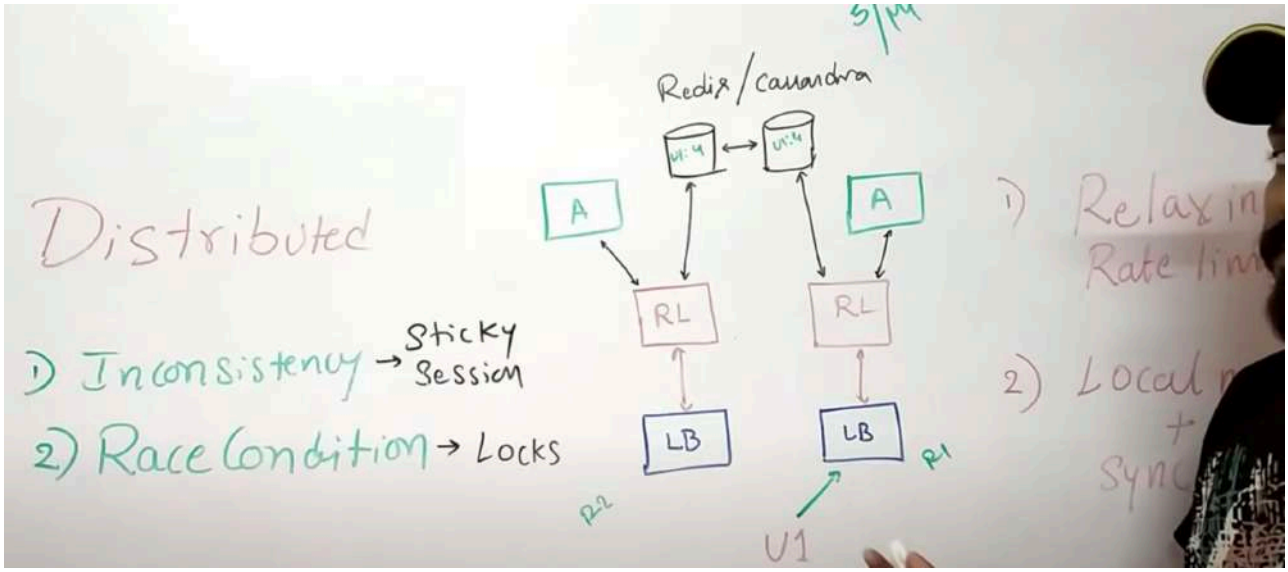
[Rate Limiting system design | TOKEN BUCKET, Leaky Bucket, Sliding Logs]

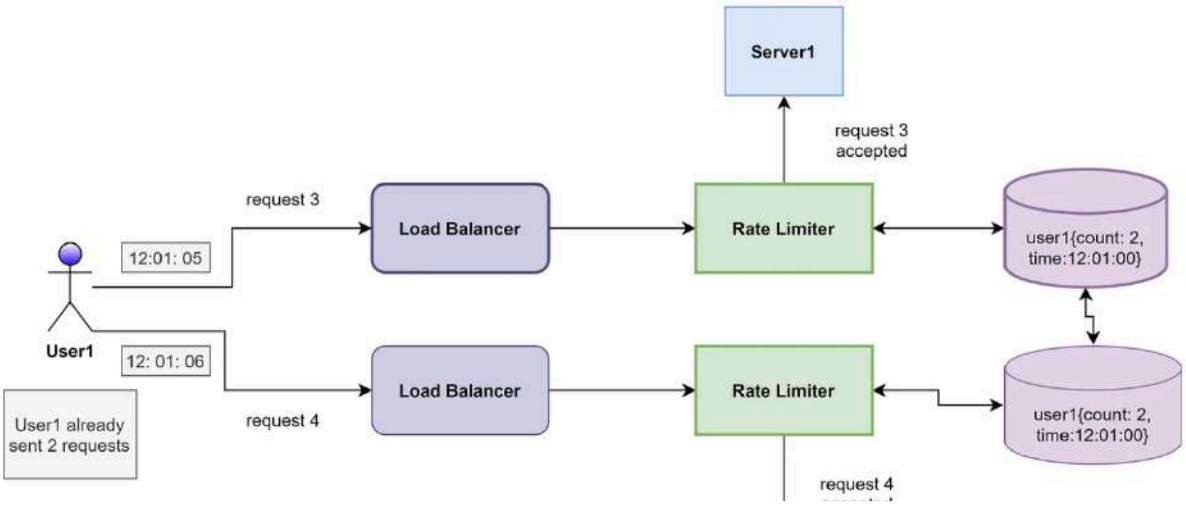
https://towardsdatascience.com/designing-a-rate-limiter-6351bd8762c6
https://medium.com/geekculture/system-design-basics-rate-limiter-351c09a57d14
https://www.codementor.io/@arpitbhayani/system-design-sliding-window-based-rate-limiter-157x7sburi

- Rate limiter algorithms
  - Fixed window counter
  - Sliding logs
  - Sliding Window Counter(efficient one)
- Distributed Rate limiter
  - Problem:
    - Inconsistency: sticky session
    - Race condition: locks
  - Solution:
    - Relaxing rate limiter
    - Local cache + sync to server periodically




10/M
Sliding
Window
Counter
U1 :
timestamp 01/2
+03/5
11:30:01
11:30:03
30 03
5


Distributed
1) Inconsistency → Sticky Session
2) Race Condition → Locks
Redis / cassandra
A
A
RL
RL
LB
LB
P1
P2
U1
1) Relax
Rate lim
2) Local
+
Sync


Server1
request 3
accepted
request 3
Load Balancer
Rate Limiter
user1{count: 2,
time:12:01:00}
12:01: 05
User1
12: 01: 06
User1 already
sent 2 requests
request 4
Load Balancer
Rate Limiter
user1{count: 2,
time:12:01:00}
request 4

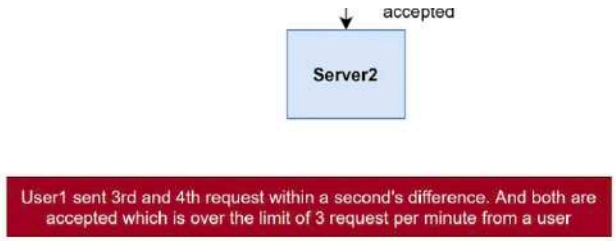


Figure1: Problems in a distributed environment (Image by Author)

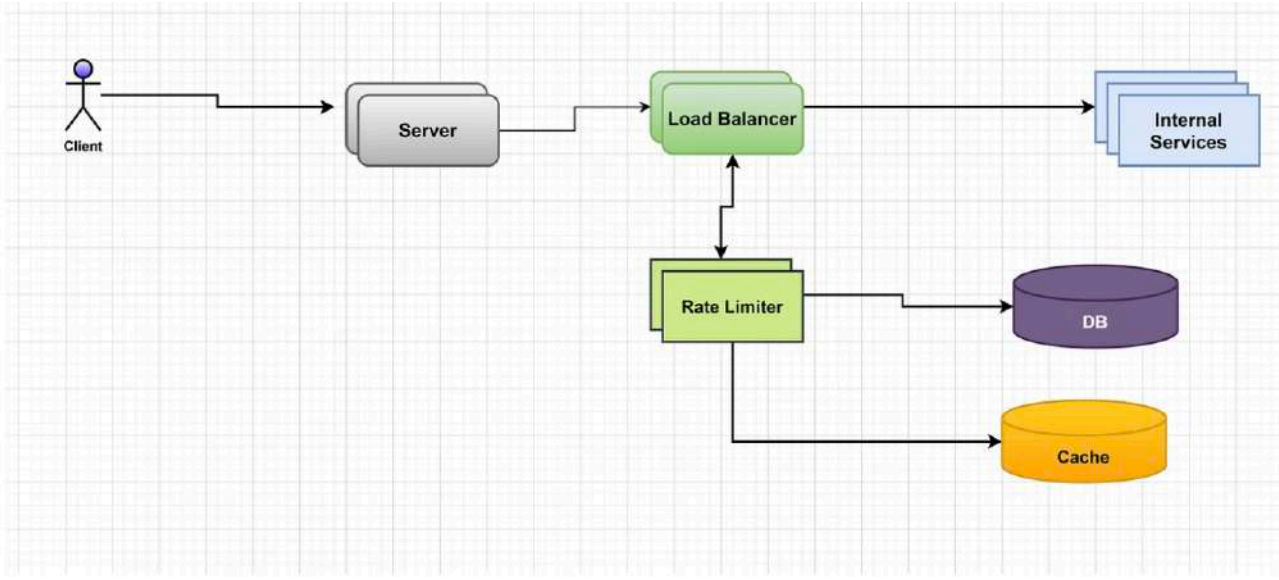
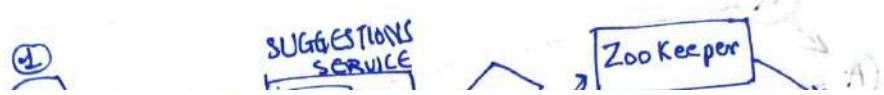


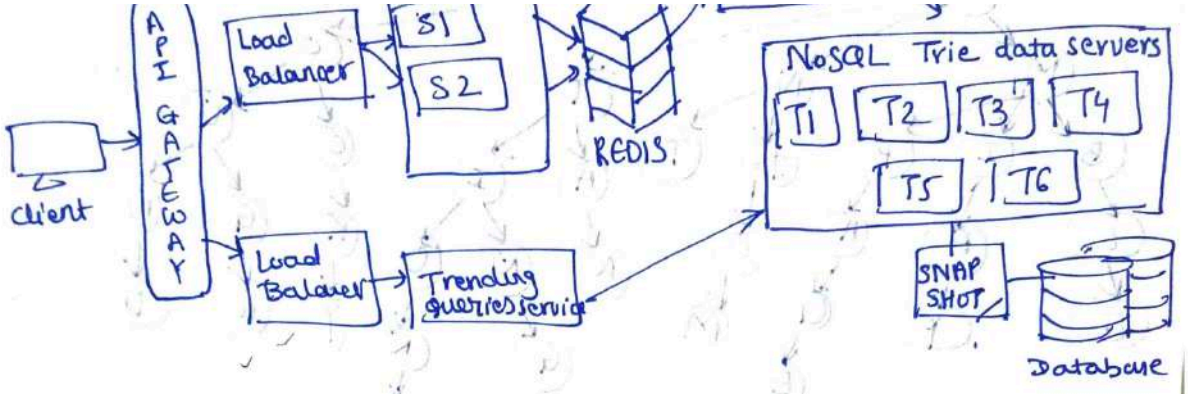
Figure 1: Final Design of Rate Limiter (Image by Author)

Auto Complete System design

[Amazon interview question: System design / Architecture for auto suggestions | type ahead]

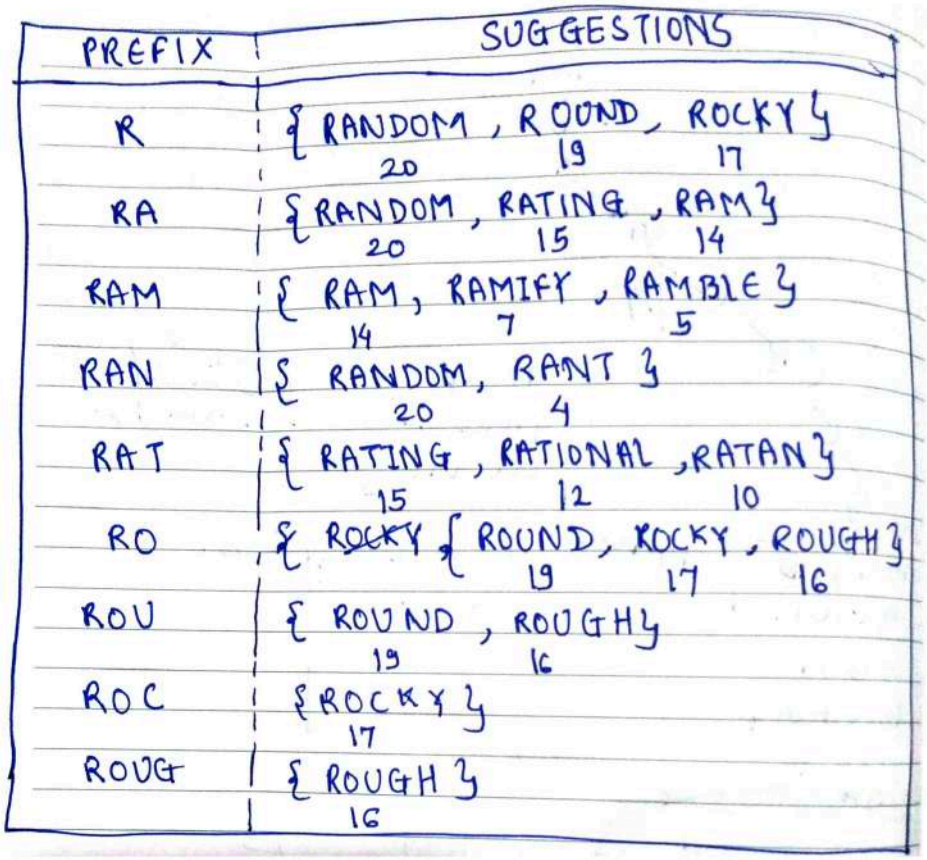
https://systemdesignprep.com/autocomplete

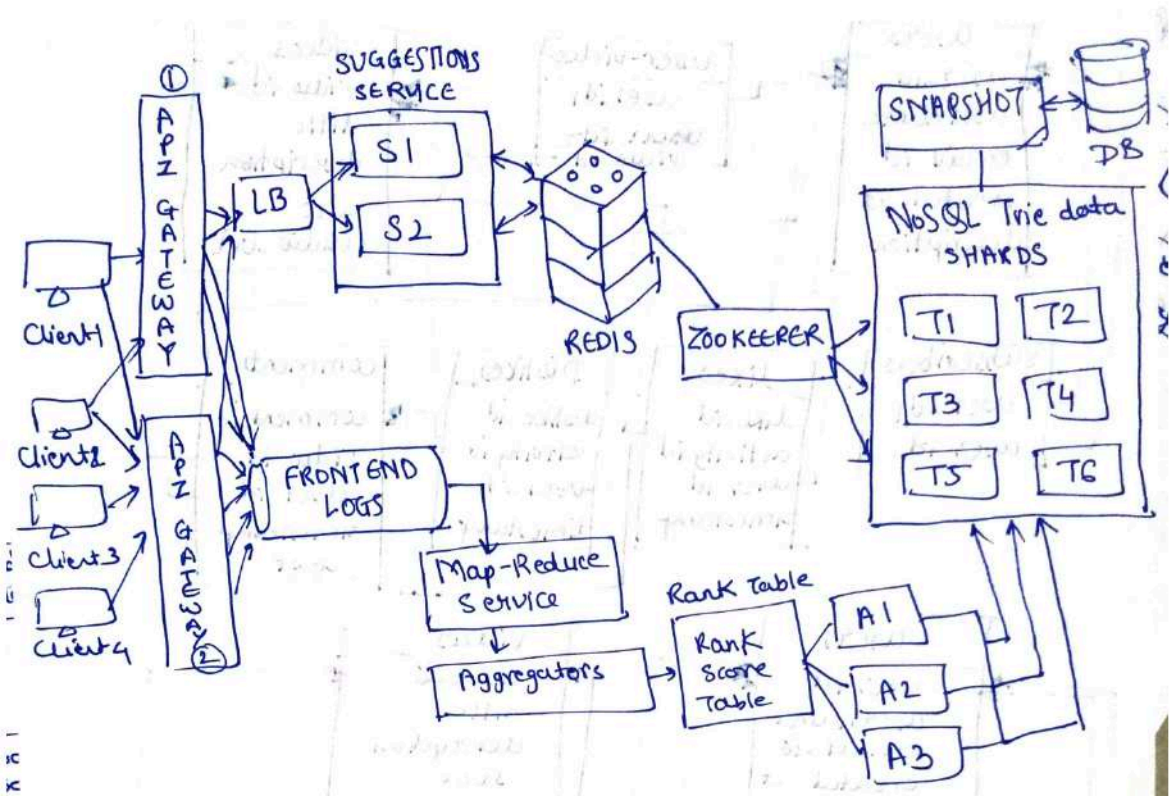




QUERY	TIME (must be in timestamp)	WEIGHT
RAM	6th July 12:00 pm to 1:00 pm	173848
RAMIFY	6th July 1:00 pm to 2:00 pm	34889
RATING	6th July 2:00 pm to 3:00 pm	256488
ROUND	5th July	8234
ROCKY	4th July	2235

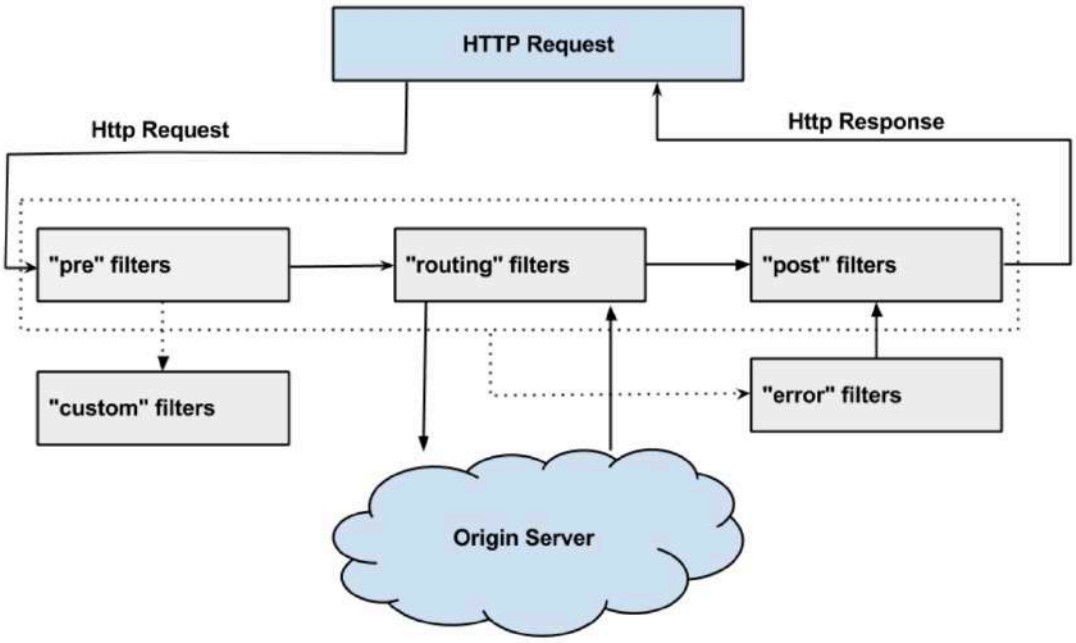
The key-value structure in the cache will be as prefix(Key)- suggestions(Value). It will look like-



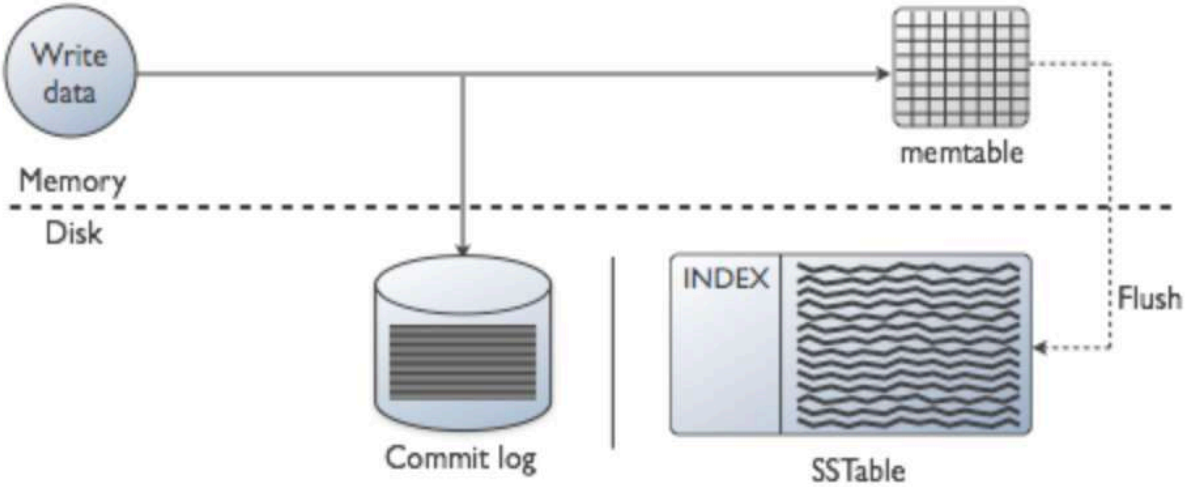


Zuul API Gateway

https://medium.com/geekculture/zuul-api-gateway-2bcdf4dd33e6



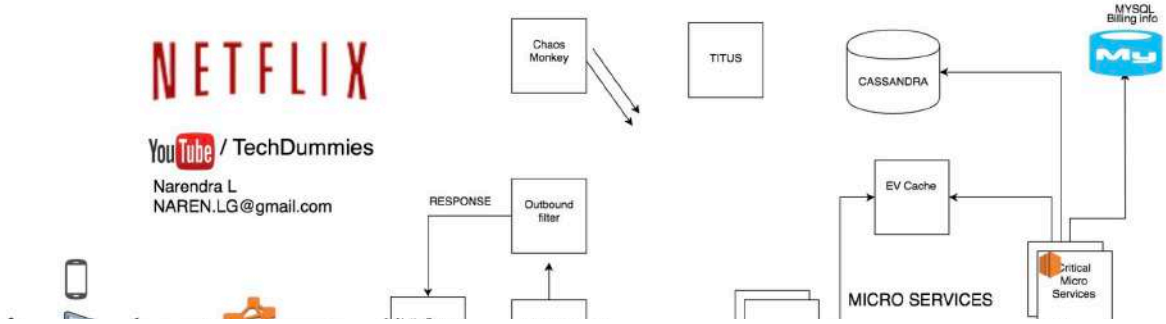
Why Cassandra writes are fast:
https://blog.devgenius.io/why-writes-in-cassandra-are-so-fast-ae4ad9413902

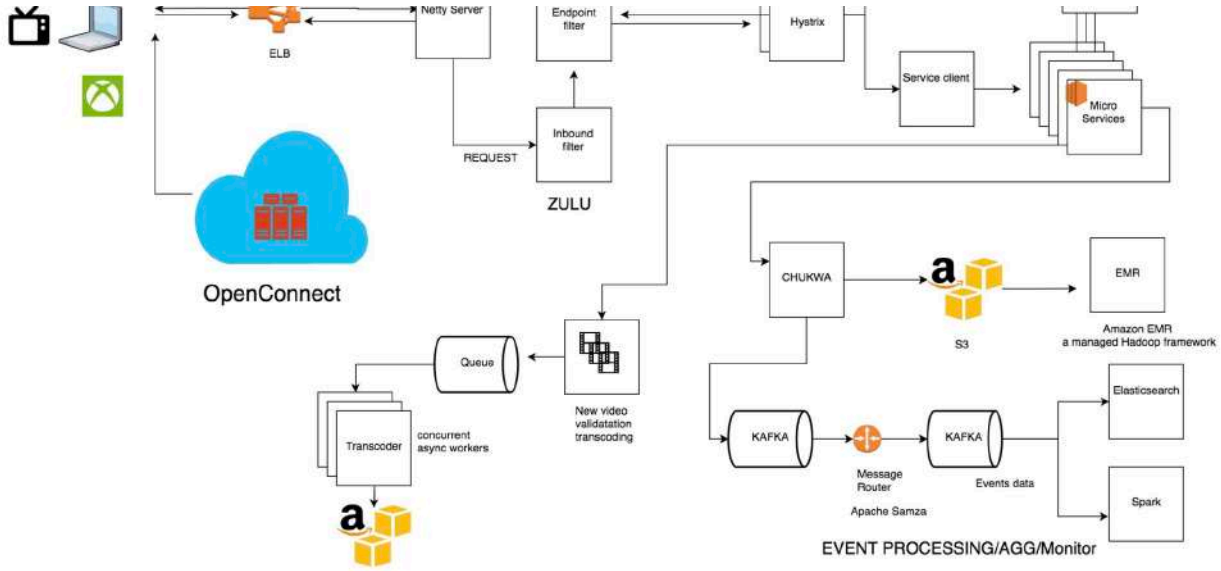


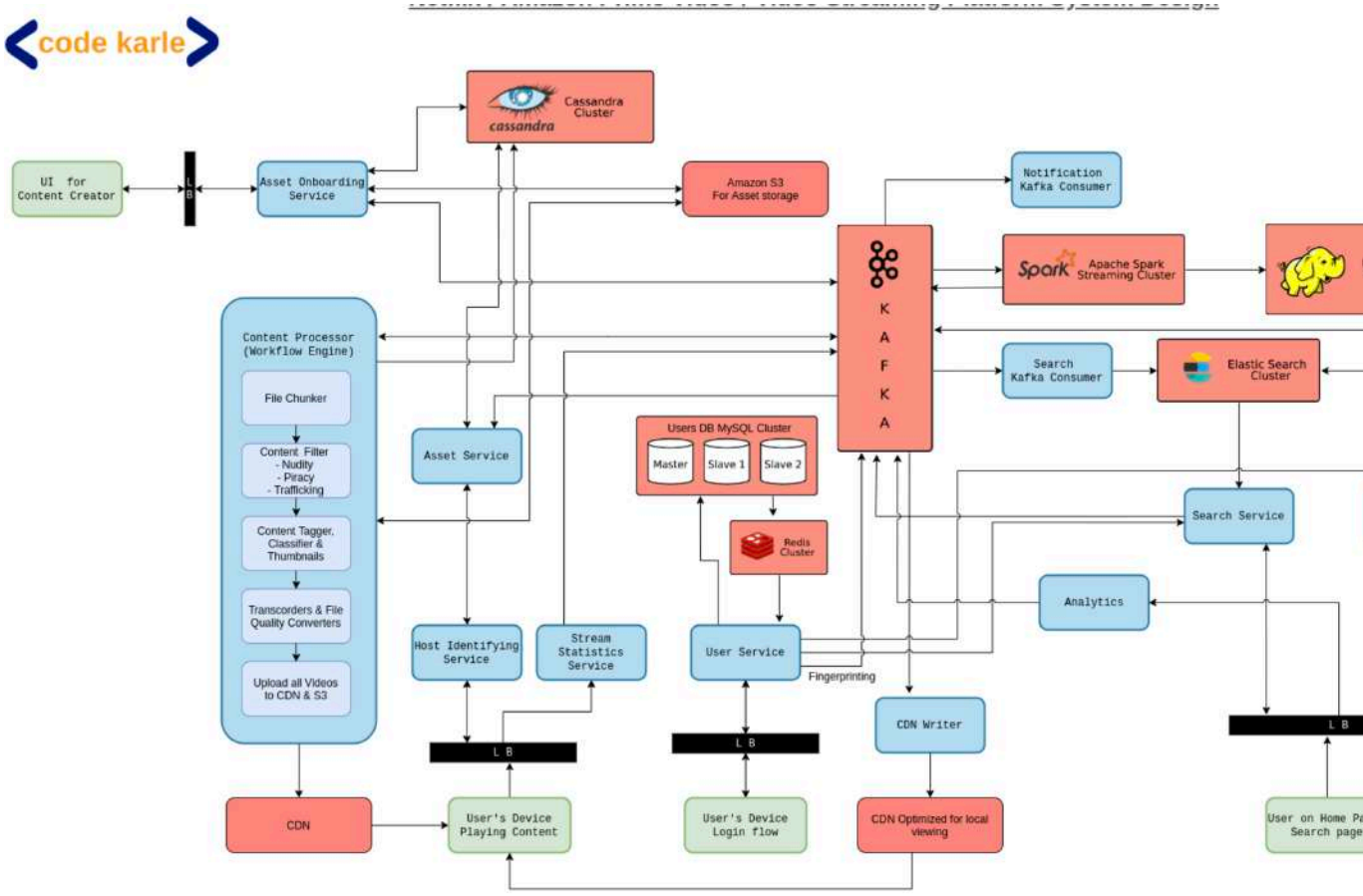
Write Process (Obtained from DataStax docs)

Netflix Design

3. NETFLIX System design | software architecture for netflix
4. https://www.geeksforgeeks.org/system-design-netflix-a-complete-architecture/
5. https://medium.com/@narengowda/system-design-dropbox-or-google-drive-8fd5da0ce55b
6. https://www.codekarle.com/images/Netflix.png



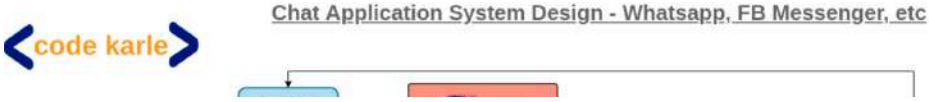


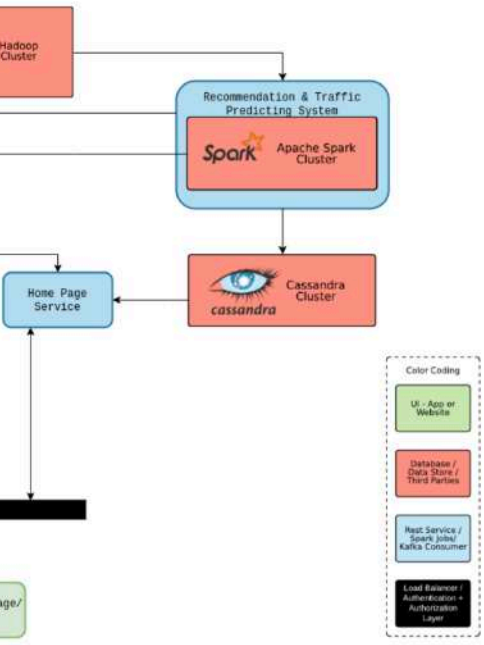


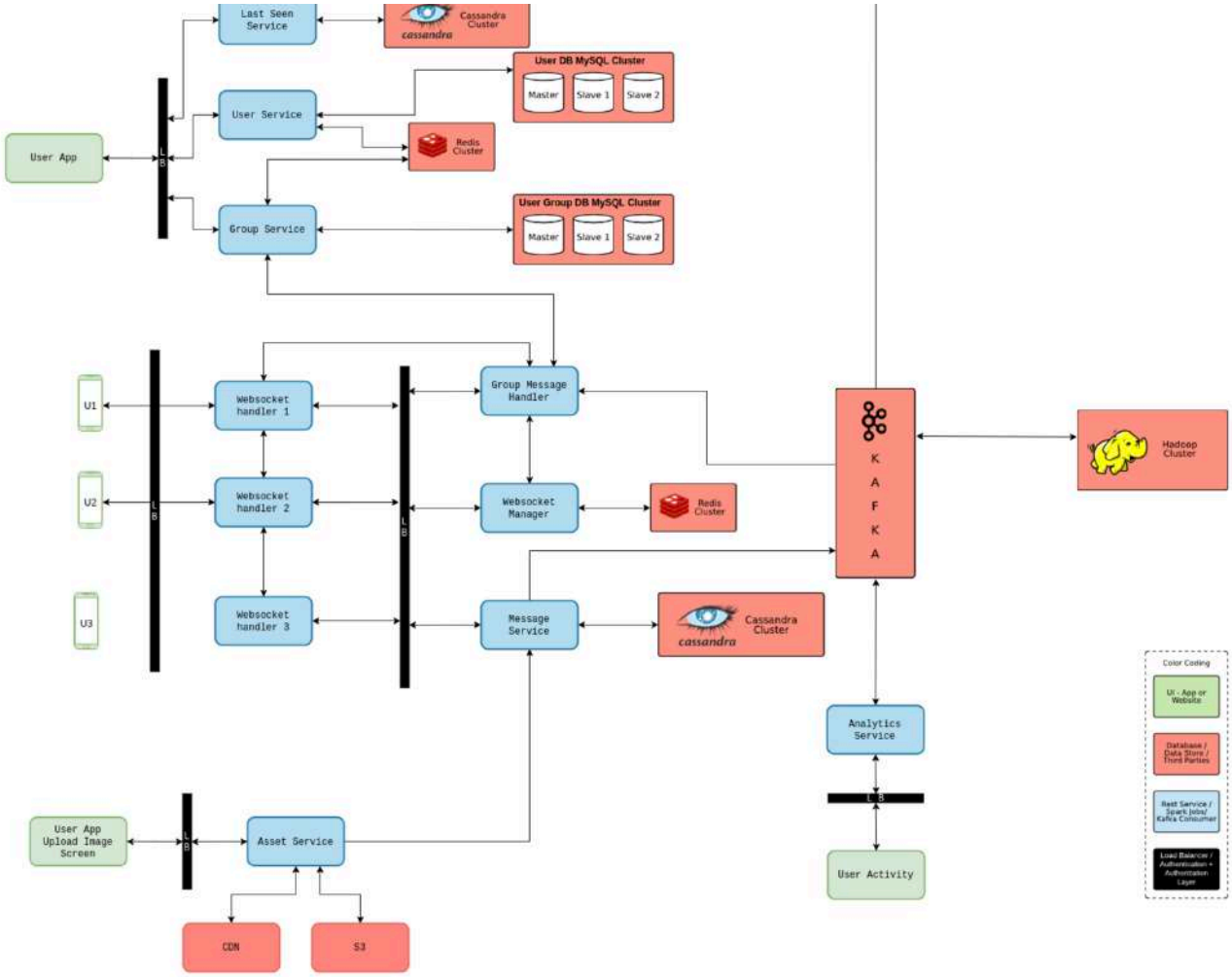
Whatsapp System Design

https://medium.com/geekculture/system-design-basics-rate-limiter-351c09a57d14

https://www.codekarle.com/system-design/Whatsapp-system-design.html




Hadoop Cluster
Recommendation & Traffic Predicting System
Spark
Apache Spark Cluster
Home Page Service
cassandra
Cassandra Cluster
Color Coding
UI - App or Website
Database / Data Store / Third Parties
Rest Service / Spark Jobs/ Kafka Consumer
Load Balancer / Authentication + Authorization Layer



Elastic search design

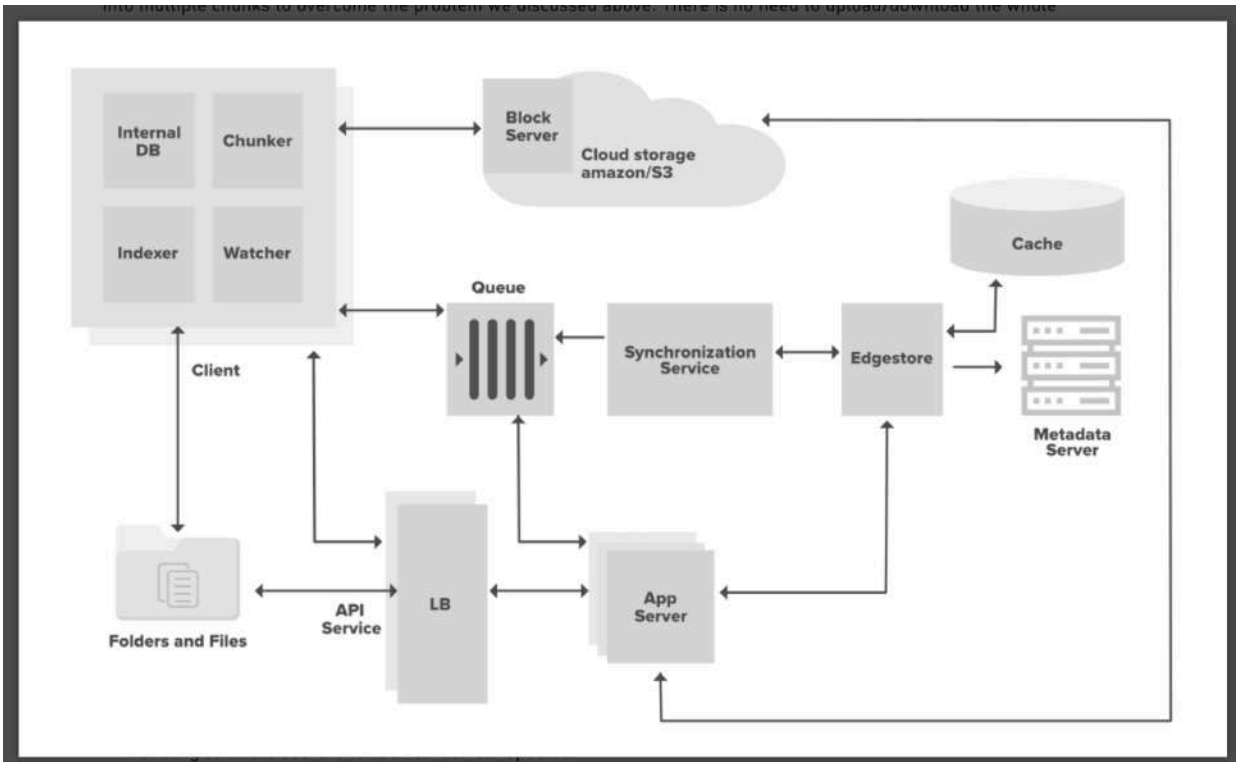
a. https://medium.com/geekculture/elasticsearch-internals-4c4c9ec077fa
b. https://thoughts.t37.net/designing-the-perfect-elasticsearch-cluster-the-almost-definitive-guide-e614eabc1a87

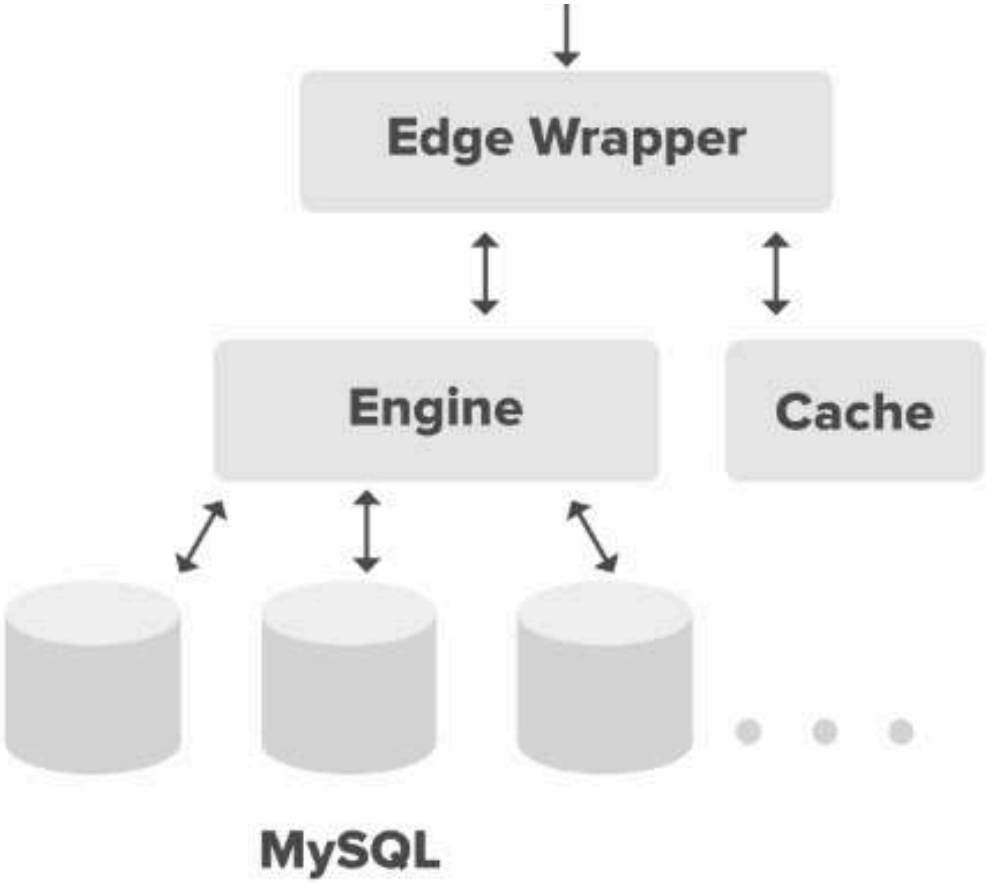
Time Series Database

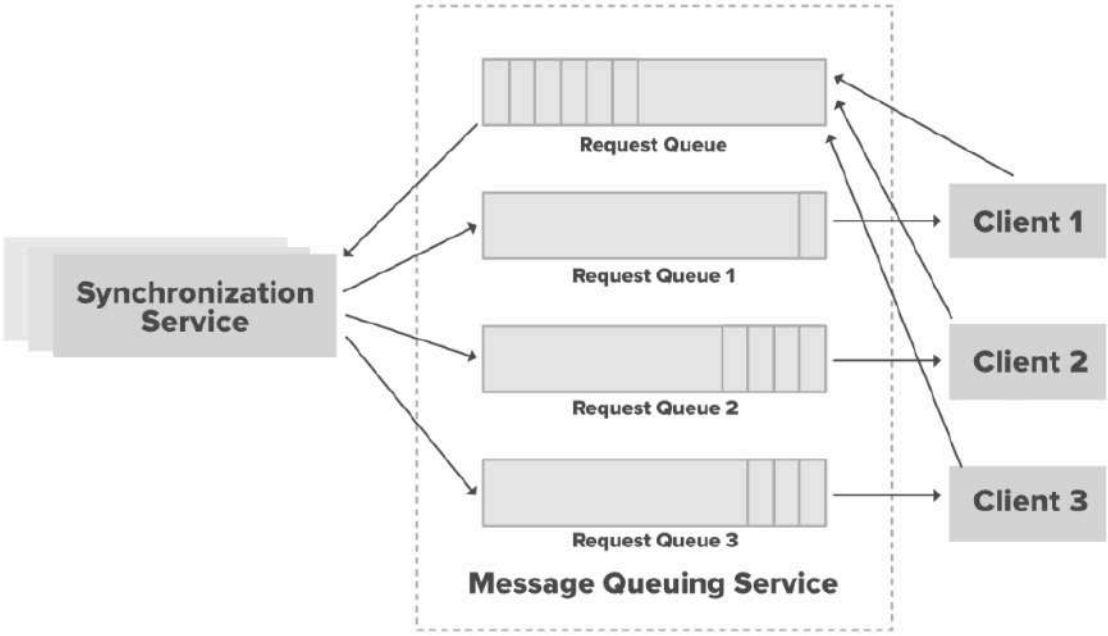
Suppose we are trying to build a **metric** tracking system, we will need something called a time-series database. Time-series databases are in a way an extension of Relational databases but unlike a standard relational DB, time-series databases will never be randomly updated. It will be updated sequentially in an append-only format. Also, it will have more bulk reads for certain time range as opposed to random reads, eg. how many people watched codekarle videos in the last 1 week, 10 days, 1 month, 1 year, and so on. Some examples of time series databases are **OpenTSDB, InfluxDB**, etc.

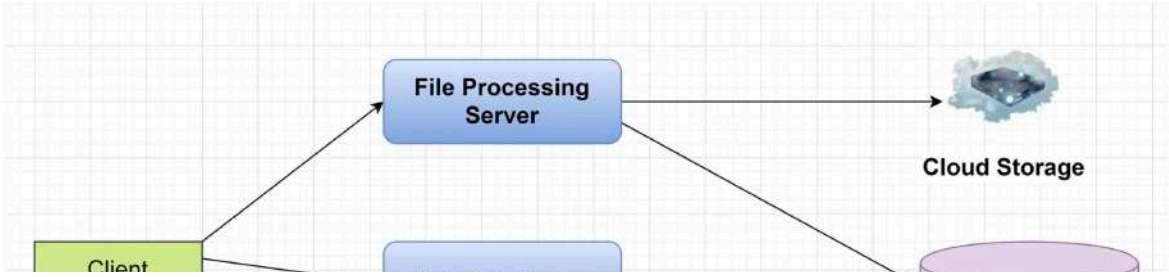
7. **Google Drive System Design**
   a. https://medium.com/@narengowda/system-design-dropbox-or-google-drive-8fd5da0ce55b

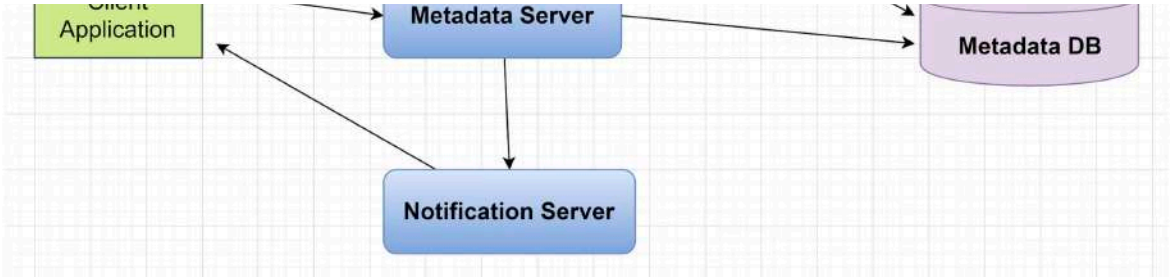
b. [Dropbox system design | Google drive system design | System design file share and upload]
c. [https://www.geeksforgeeks.org/design-dropbox-a-system-design-interview-question/]




Edge Wrapper
Engine
Cache
MySQL


Request Queue
Request Queue 1
Request Queue 2
Request Queue 3
Synchronization Service
Client 1
Client 2
Client 3
Message Queuing Service


File Processing Server
Cloud Storage
Client



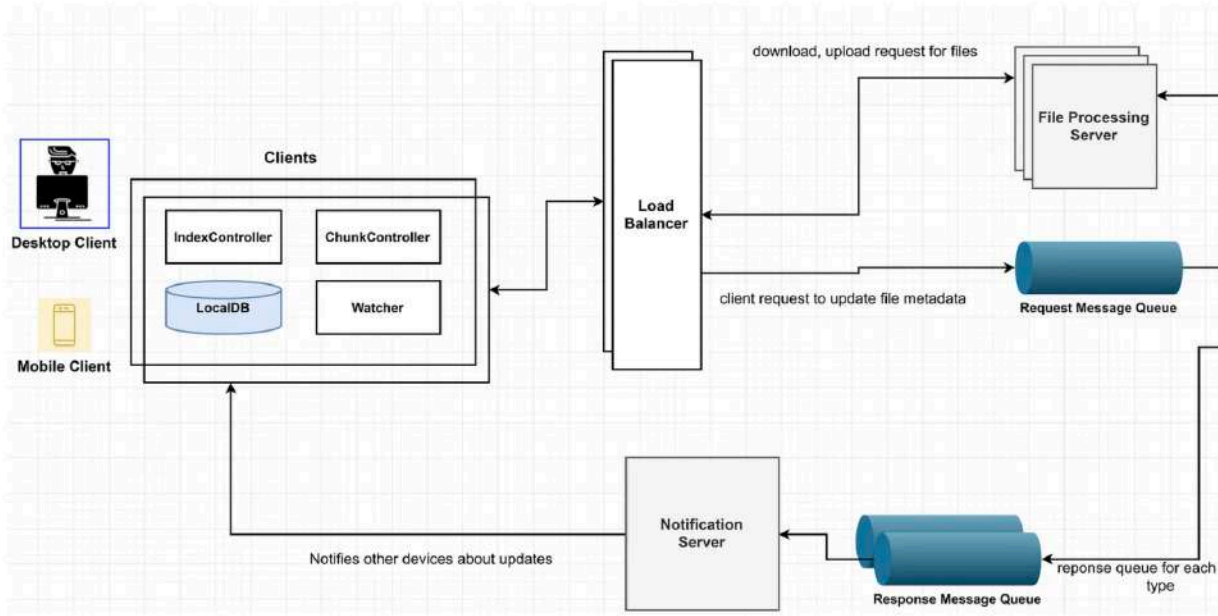
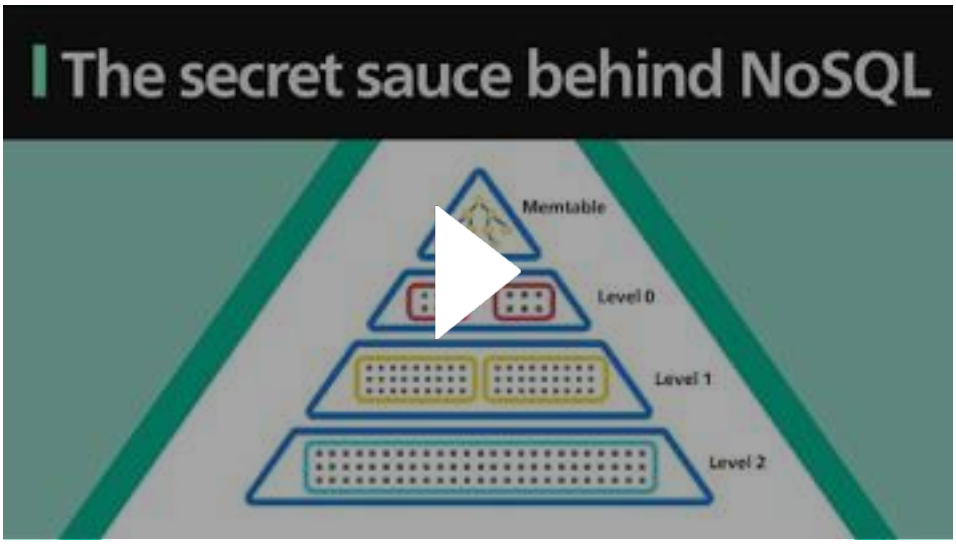


Figure: System Design of Google drive (Image by Author)

Why no-sql DB is so fast

[The Secret Sauce Behind NoSQL: LSM Tree]

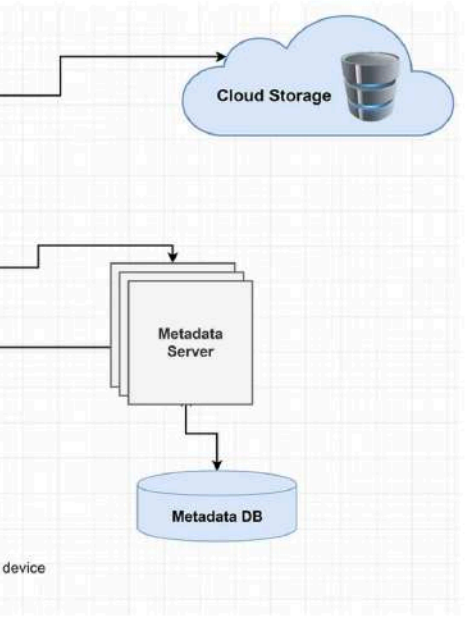


https://www.linkedin.com/pulse/data-structures-powering-our-database-part-2-saurav-prateek/?trk=pulse-article_more-articles_related-content-card

Memtable - in memory storage

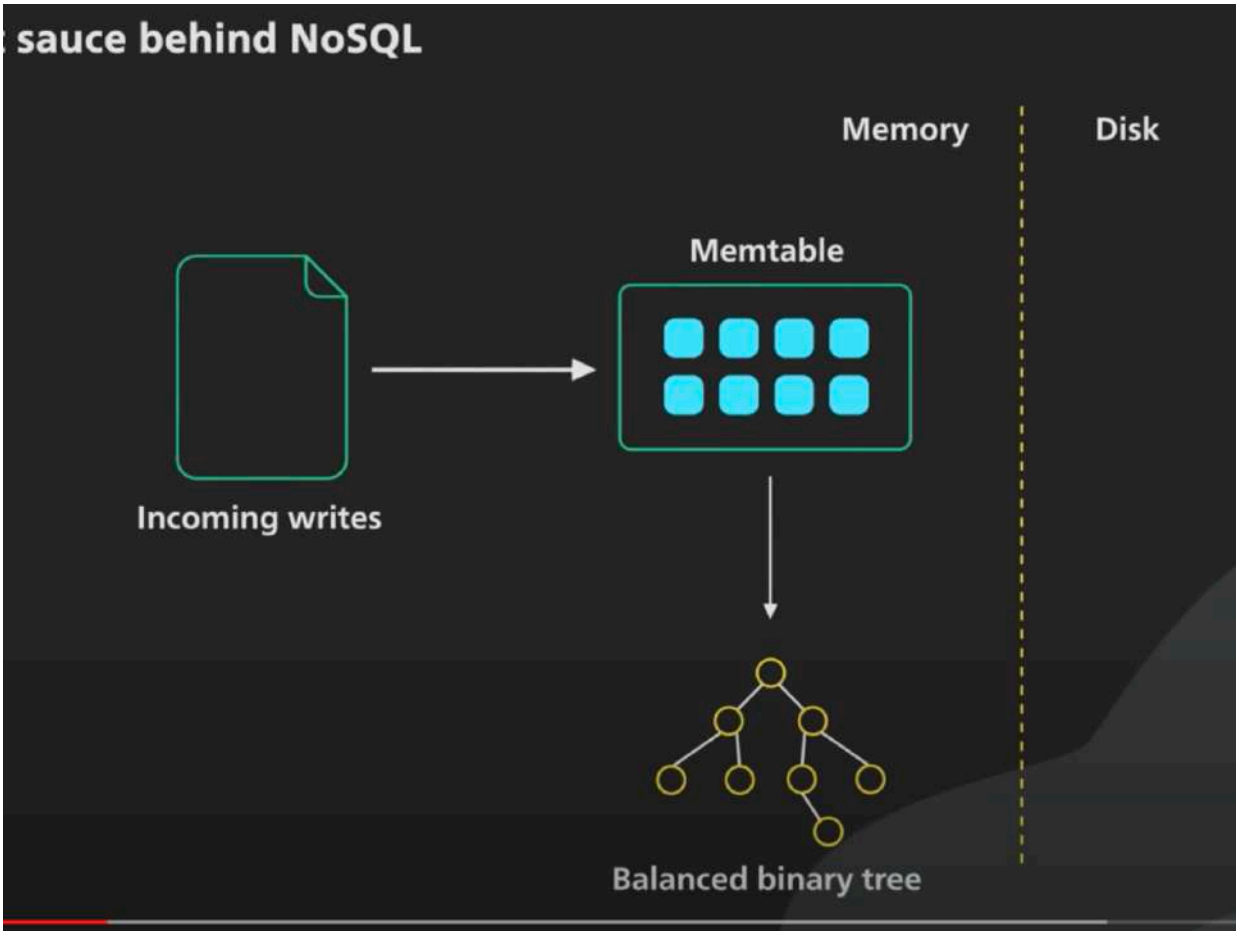
Organizes the data in Balanced binary tree for efficient search

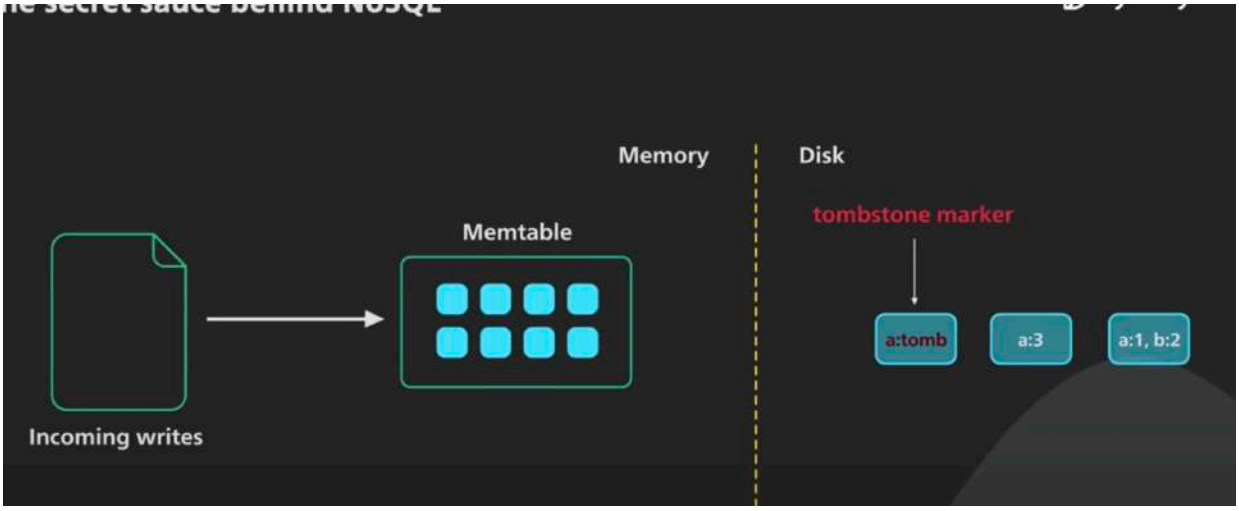
SSTable - **Sorted string table** for permanent(immutable) storage on disk


Cloud Storage
Metadata
Server
Metadata DB
device

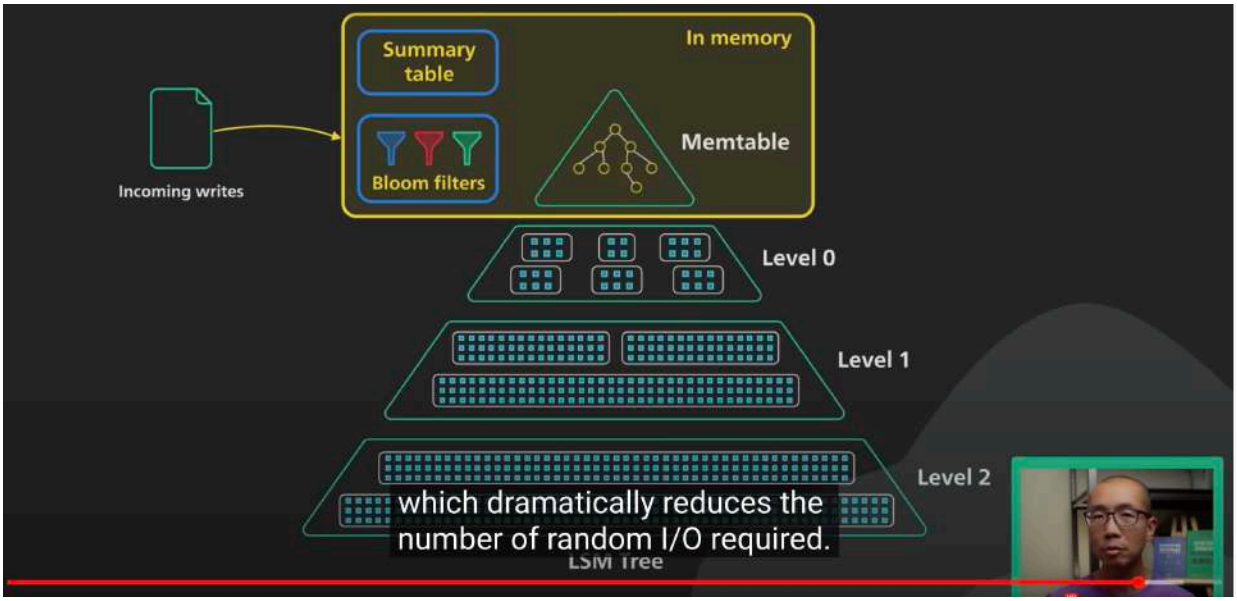
using sequential I/O process
It adds the data in append only format instead of modifying the data
Most recent Sstable will be in front always
And compaction occurs asynchronously to get merged in **LSM tree**
Bloom filter - for efficient key lookup in each levels in LSM tree
Deletion process:

Data will be deleted by marking it as tombstone
And it will be deleted after the **grace** period





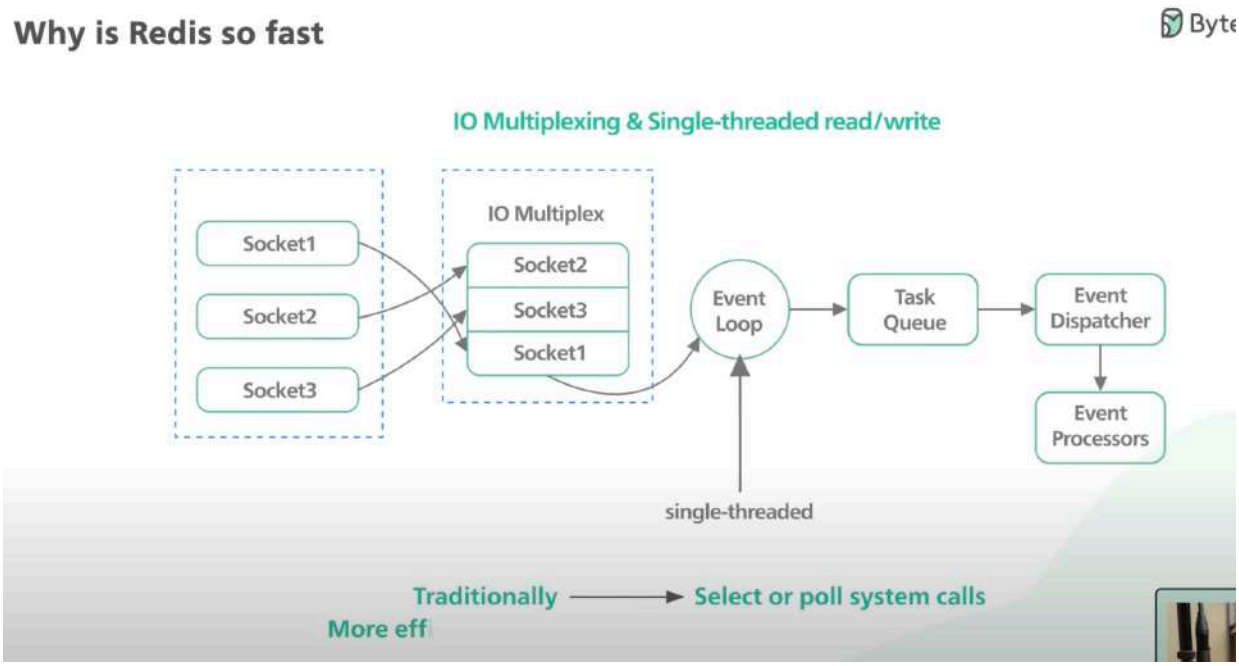
The secret sauce behind NoSQL ByteByteGo



LSM Tree - Log structured merge tree

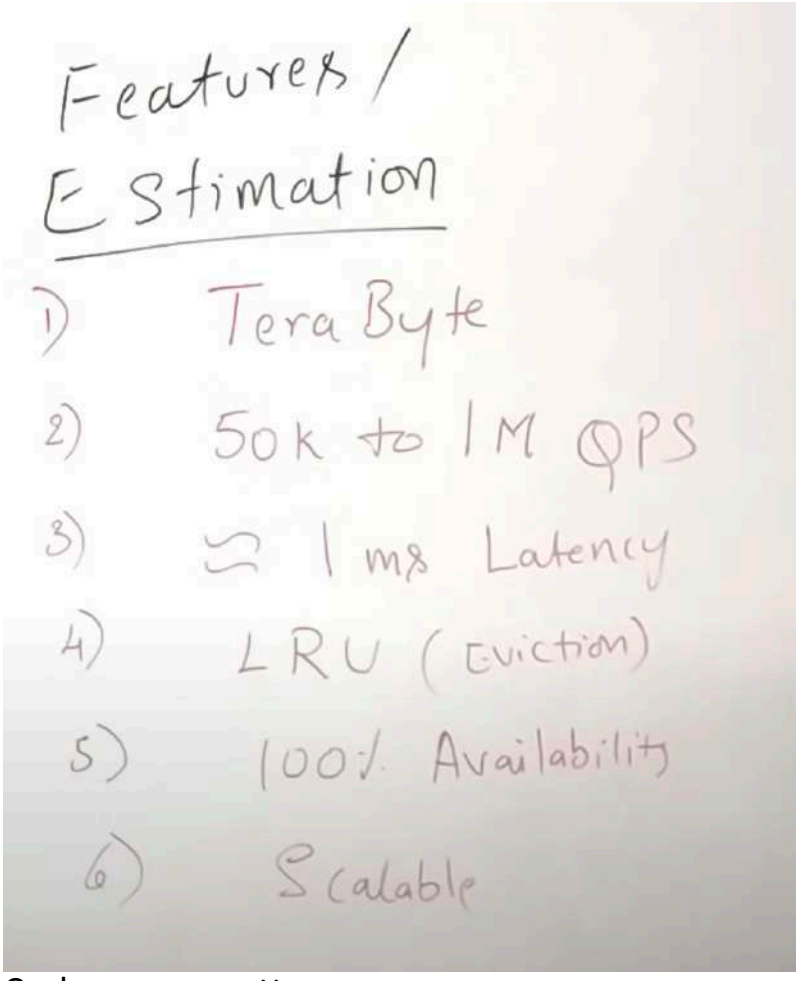
Redis:
[System Design: Why is single-threaded Redis so fast?]

i. In Memory Storage
ii. IO multiplexing and single-threaded implementation
iii. Optimized lower-lever data structures



Caching best practices:
TTL
Validity

High hit rate
Cache miss less



Cache access patterns
Write through - store in cache first and then store in db
Write around - skip cache and store in db, so whenever cache miss happens, it will be stored in cache
Write back -> asynchronously sync cache data to db

Cache eviction policy
LRU
LFU

Core logic of cache
Hashtable
Store in key-value pairs
So key can be hashed

Fault tolerant:
Log reconstruction
Regular interval snapshot

For availability
Have replicas
[Redis system design | Distributed cache System design]

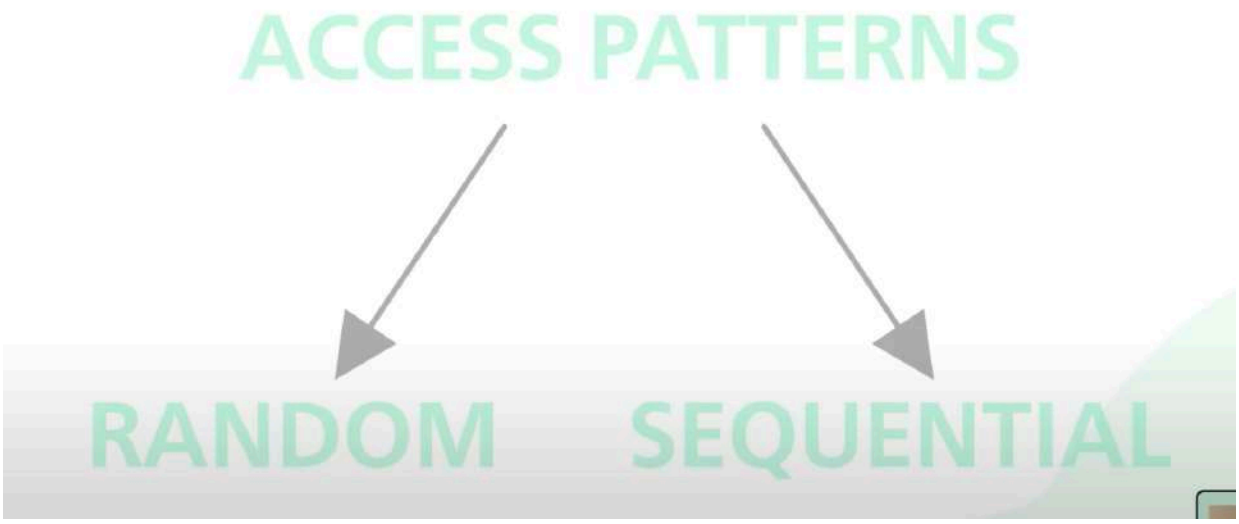
Why kafka has so higher throughput

[System Design]: Why is Kafka fast?

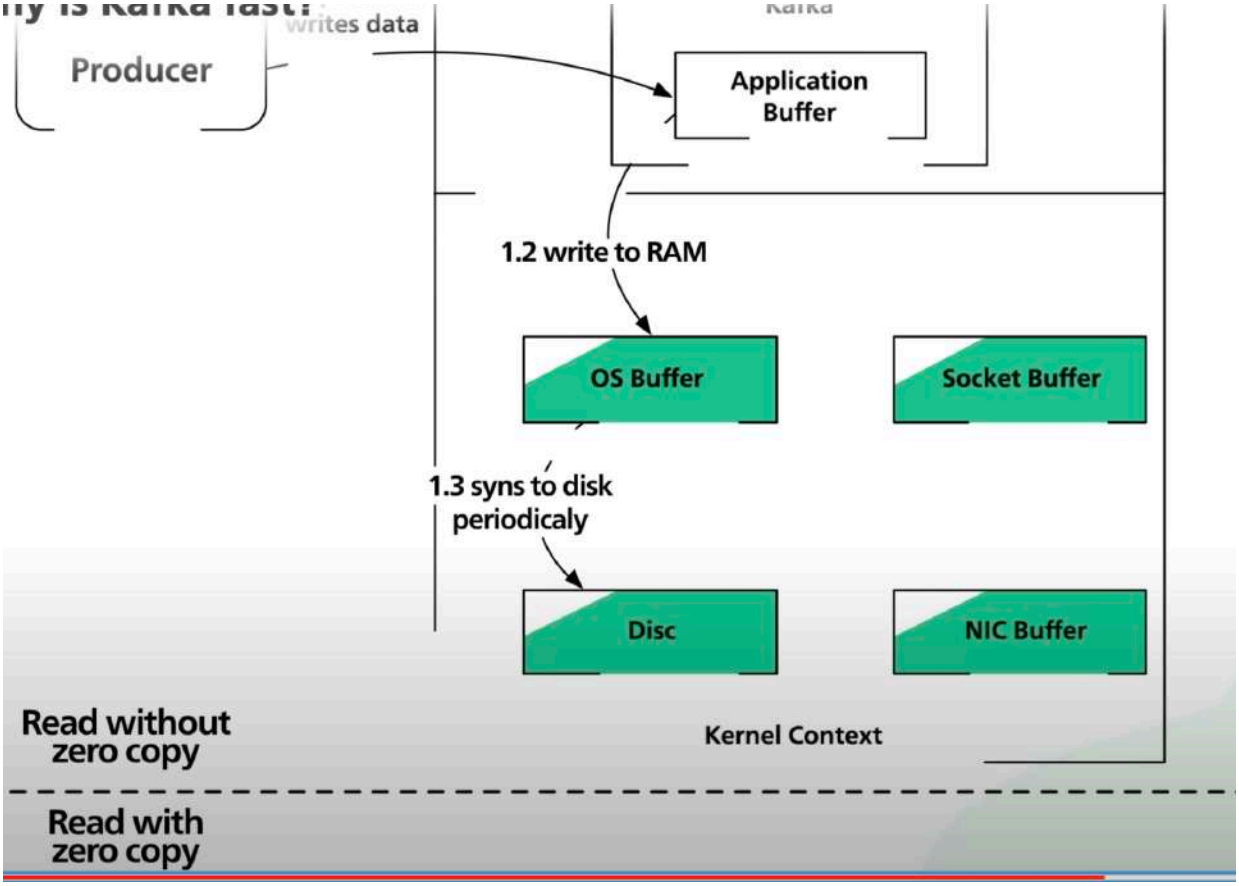
- **Sequential I/O disk access patterns**
  - 2 types of disk access patterns
    - Random access patterns
    - Sequential access patterns
      - Since kafka is adding data using append only logs(AOL),
      - It will be stored in disk quickly using sequential

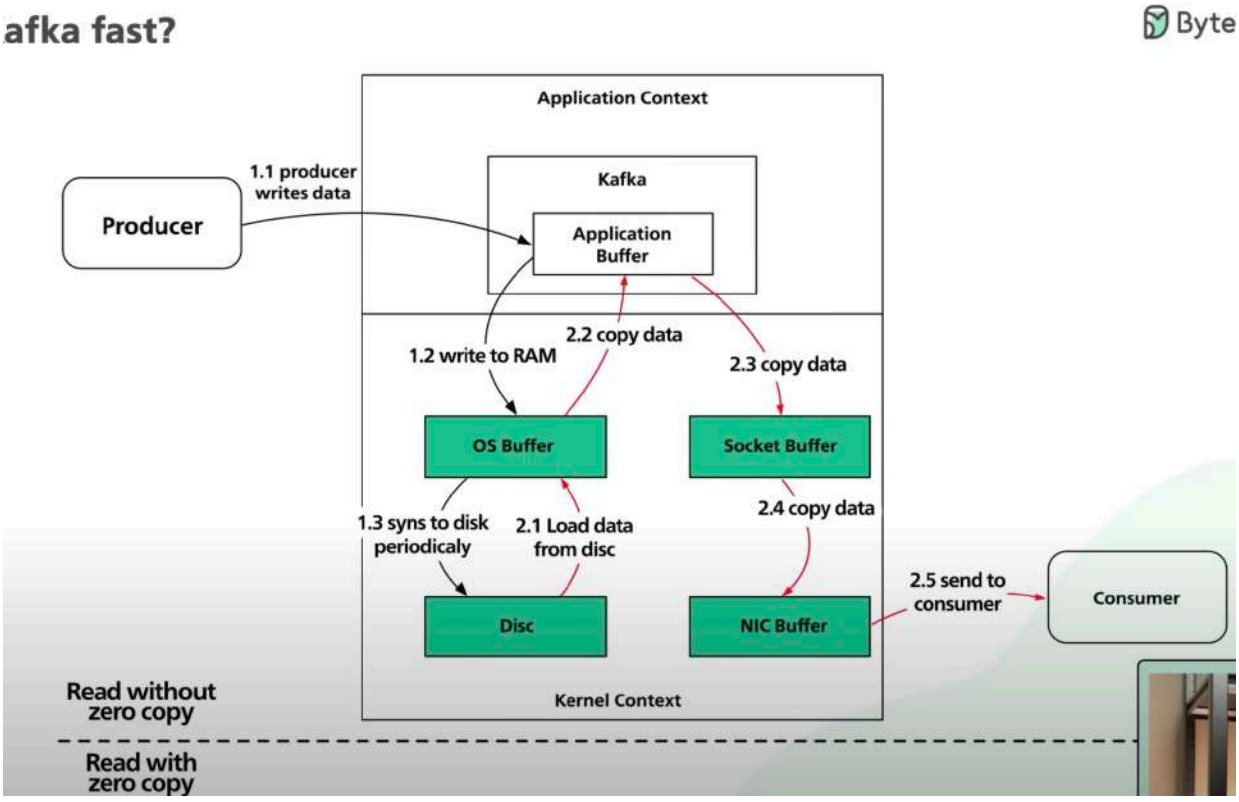
access

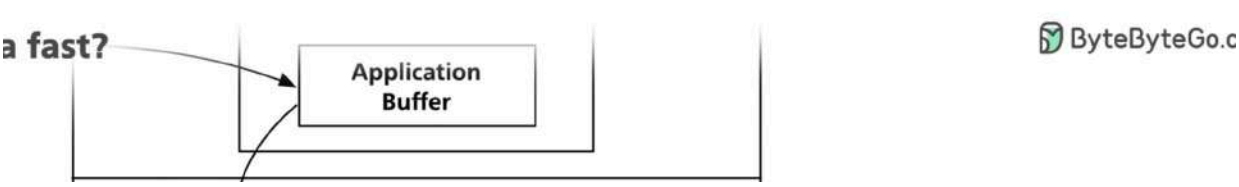
- **HDD is cost effective** than SDD, so retention becomes cost effective since it uses more HDD
- **Zero Copy logic** by copying data from disk to network
- Kafka application will directly copy the data from disk to network interface card(NIC) buffer using DMA(direct memory access) or zero copy
- In traditional systems, it will be copied from
  - disk buffer to OS buffer(RAM)
  - OS buffer to Kafka application buffer
  - Kafka application buffer to Socket buffer
  - Socket buffer to NIC buffer

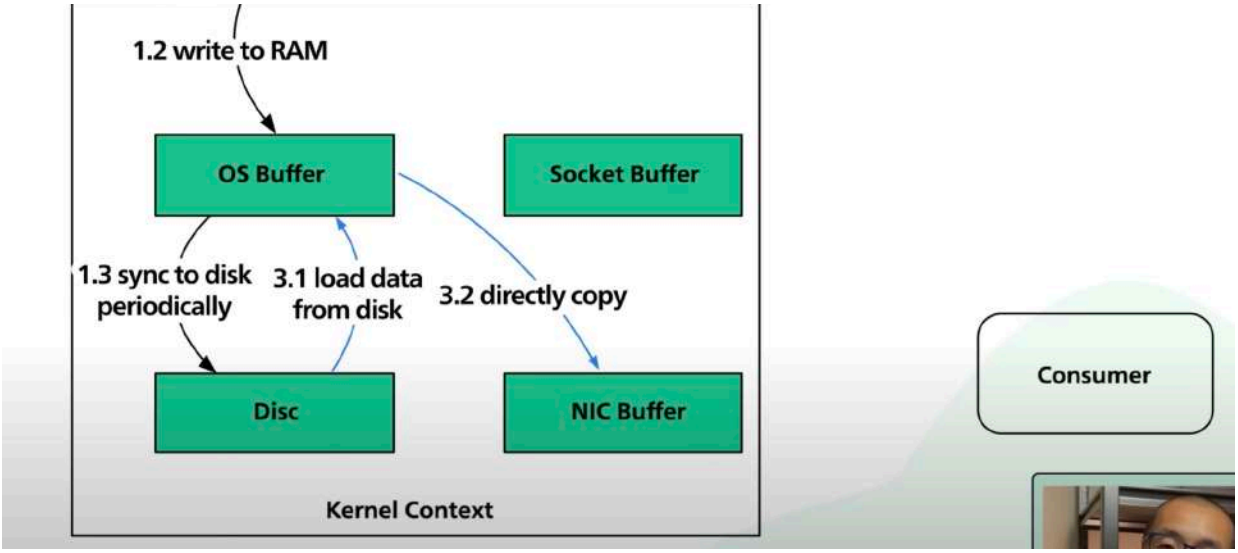


hy is Kafka fast? producer


writes data
Producer
Application Buffer
1.2 write to RAM
OS Buffer
Socket Buffer
1.3 syns to disk periodicaly
Disc
NIC Buffer
Read without zero copy
Kernel Context
Read with zero copy


afka fast?
Byte
Application Context
Producer
1.1 producer writes data
Kafka
Application Buffer
1.2 write to RAM
2.2 copy data
2.3 copy data
OS Buffer
Socket Buffer
1.3 syns to disk periodicaly
2.1 Load data from disc
2.4 copy data
2.5 send to consumer
Consumer
Disc
NIC Buffer
Read without zero copy
Kernel Context
Read with zero copy


a fast?
Application Buffer
ByteByteGo.c



https://medium.com/event-driven-utopia/understanding-kafka-topic-partitions-ae40f80552e8

Partitions:

- Topic is materialized event stream
- Partitions are the single log file where records are written to it in an append-only fashion

autoOffset:

- Earliest - when consumers joins first time, it will fetch historical data also
- Latest - it will poll only the new incoming data after it joins
- https://dzone.com/articles/apache-kafka-consumer-group-offset-retention
- https://stackoverflow.com/questions/48320672/what-is-the-difference-between-kafka-earliest-and-latest-offset-values
- __consumer_offsets -> {partition, consumerGroup, commited offset)
- kafka auto.offset.reset earliest vs latest

- Partition Key:
  - Partition Key will be passed to **hashfunction** which will decide to which partition the message has to be routed
  - If there is no partition key, it will be routed based on round-robin assingment

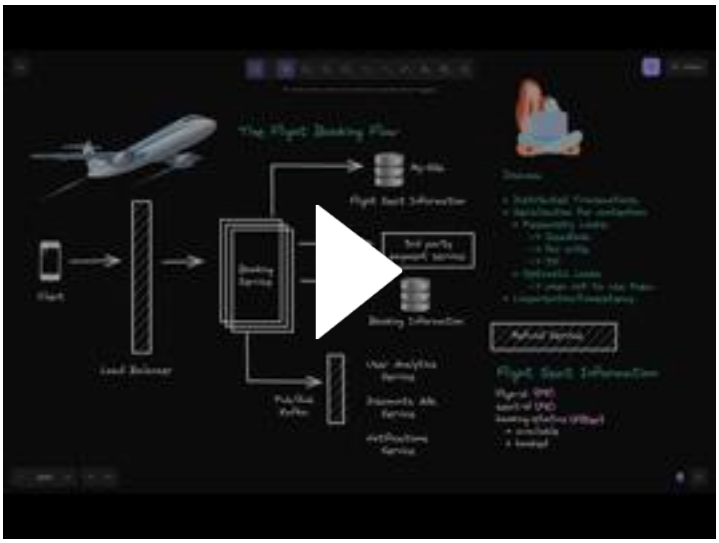
Bloom filter:

https://medium.com/@prateektiwari.in/bloom-filter-b195799a2496
https://www.geeksforgeeks.org/bloom-filters-introduction-and-python-implementation/

Data Corruption and Merkle Trees

Data corruption and Merkle trees

Airline Booking System design
[Airline Reservation System - Distributed Transactions, Serialisation, Linearisation, Consistency]

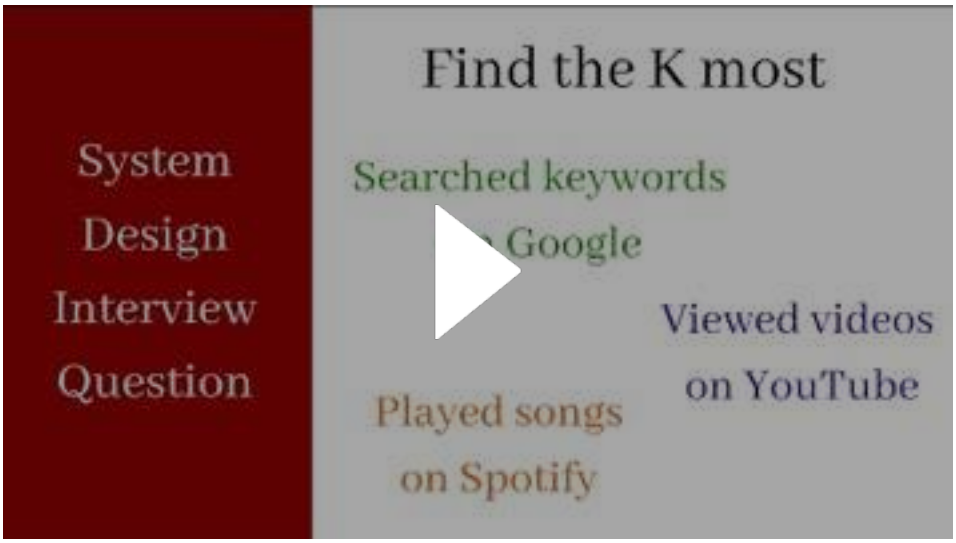


Pessimistic Locks
Optimistic Locks

When there are more parallel operations to book a same seat, it will result in success of 1 transaction and all other computed transactions have to be rolled back completely.
It results in user bad experience and wastage of resources/computations

Design Key-Value Store DB - Amazon Dynamo DB
[9. DESIGN A KEY-VALUE STORE | Amazon System Design Interview Quest. | HLD of Key-Value DB & DynamoDB]

Leadership board:
https://levelup.gitconnected.com/how-we-created-a-real-time-leaderboard-for-a-million-users-555aaa3ccf7b
https://systemdesign.one/leaderboard-system-design/
System Design Interview - Top K Problem (Heavy Hitters)



Count-Min Sketch
Count min sketch | Efficient algorithm for counting stream of data | system design components


PPLICATIONS
FOR SYSTEM
DESIGN