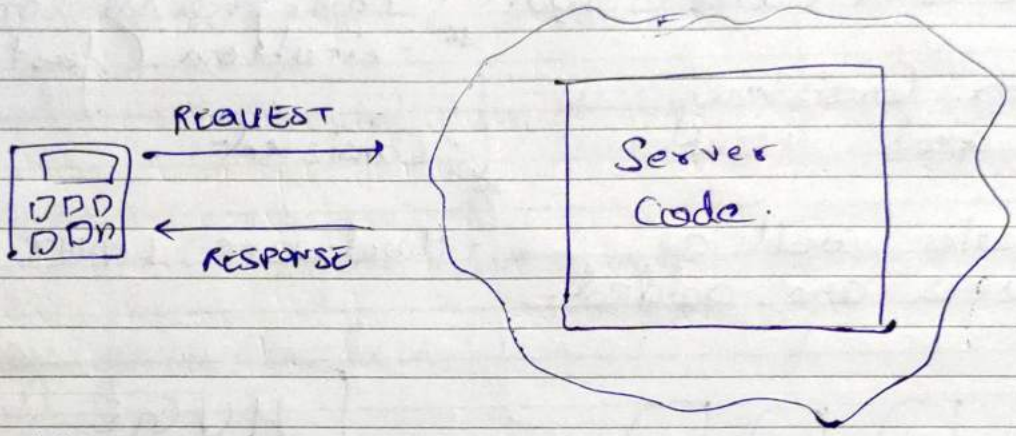
13/10

System Design

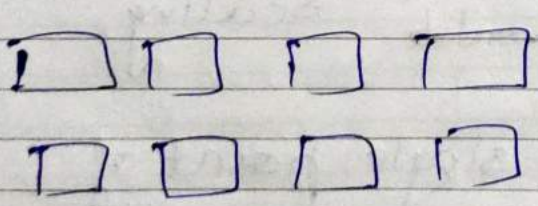


Scalability

① Buy BIGGER Machine. ⇒ VERTICAL SCALING.

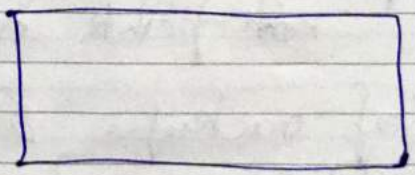
② Buy MORE Machine. ⇒ HORIZONTAL SCALING.

HORIZONTAL



- Load Balancing Required.

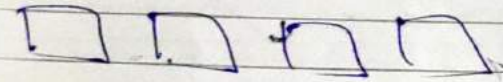
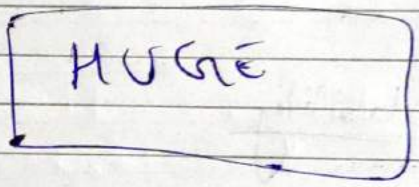
VERTICAL



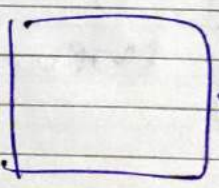
- N/A.

☆ → Good Qualities.

Resilient → preventing failures.

☆ Resilient	• Single pt. failure.
• Network calls (slow).	• Inter process communication (fast) ☆
• Data Consistency is a real issue.	• Consistent ☆
• Scales well as ☆ users are added.	• Hardware LIMIT.
	

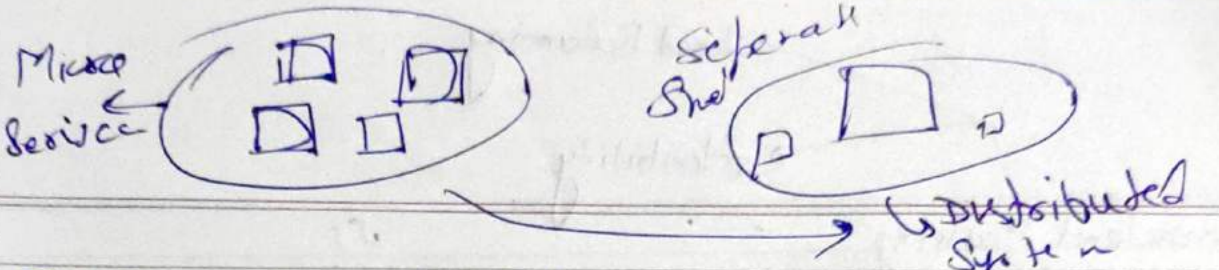
PIZZA SHOP EXAMPLE

1. Optimise processes and increase throughput using the same resource. }  1 chef. → Vertical Scaling.

2. Preparing before hand at non-peak hour.] → CronJob

3. Keep backups & avoid single point of failure.] → Backups

4. Hire more resources.] → Horizontal Scaling.



5⊙ Micro-Service Architecture

6⊙ Distributed System (Partitions).

7⊙ Load Balancer (~~votes~~ req to diff servers).

- Scalable
- Fault tolerance
- Extensible

8⊙ Decoupling → Seperation of responsibility } handle. seperate System more responsibly.

9⊙ Logging & ~~Alerts~~ Metric

10⊙ Extensible.

HIGH LEVEL DESIGN (HLD)

ALLOWER

→ Order Overload. → Recruitment
→ Complexity. → Seperation of Concerns
→ Mishaps → Fault tolerance.

Load Balancing.

Balancing load among servers.
- This uses Consistent hashing.

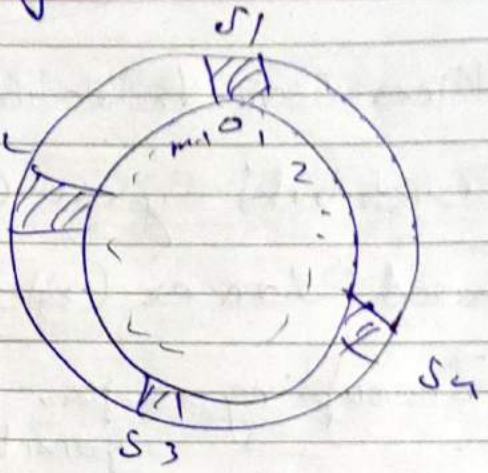
Using hash functions to map & request to √ N-servers.

→ Load Balancing.

→ Flexibility.

Consistent Hashing.

We use cyclic structure to reduce request change on when servers are added.



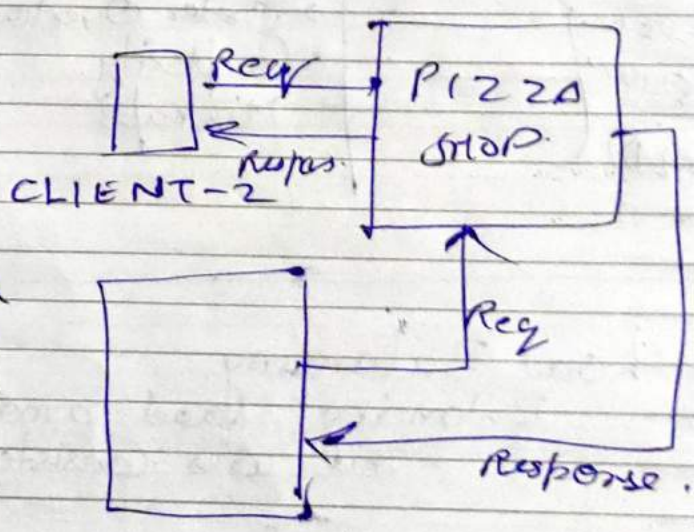
We can use k-hash function.

this avoids skewed load on each server.

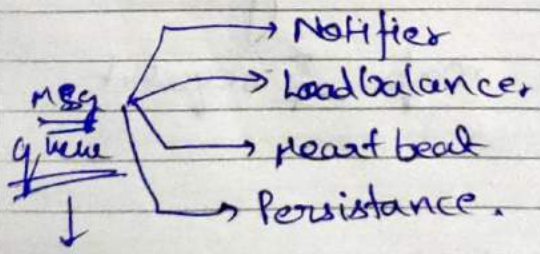
Extensively used in → distributed systems.
→ web caches.
→ Data Bases.

MESSAGE QUEUE

This architecture is asynchronous.

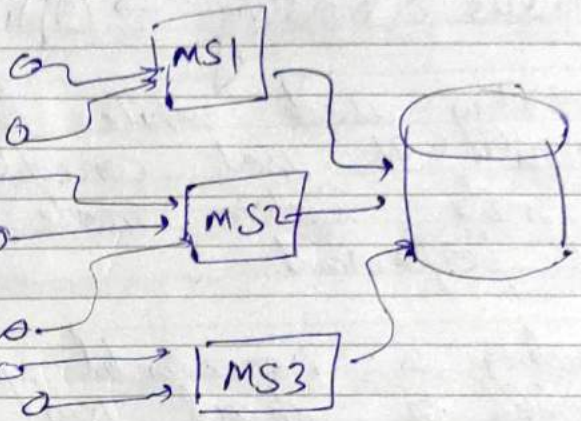


Principles of load balancer also ensures that there is no duplicates req. on a server.



Eg: Rabbit MQ, JMS, ZeroMQ.

MONOLITH

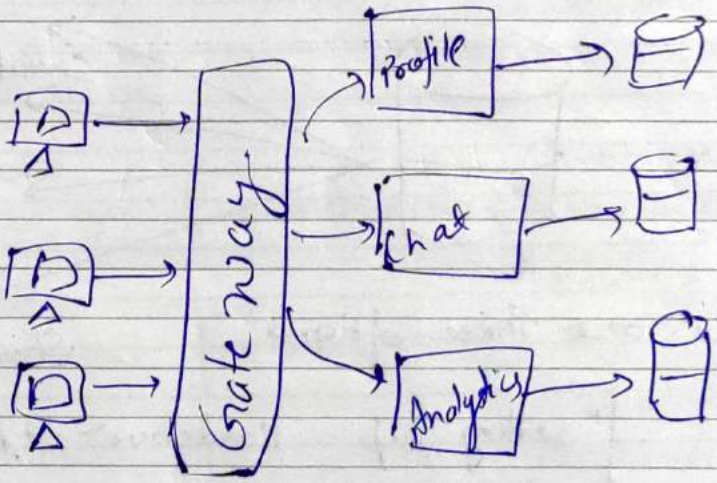


Advantages
- Good for small teams.
- Less Complex.
- Less Duplication.
- Faster (due to procedural) call.

Disadvantages
- Complicated Deployment.
- Not decoupled.
- Single pt. of failure.

(Eg: StackOverflow)

MICROSERVICES



Advantages
- Scalability.
- Easier for new team members.
- Easier to reason about (which is used more).

Disadvantage
- Harder to design.
- Needs skilled Architects.

Good for large scale systems.

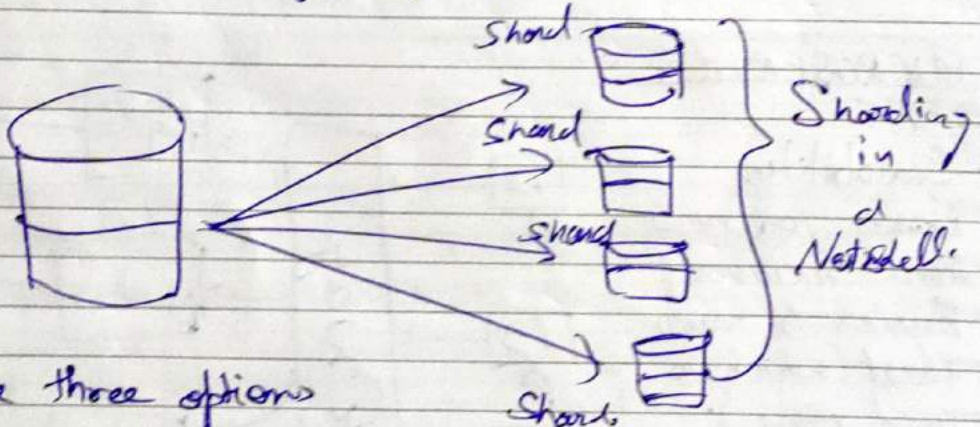
Eg: Google, facebook.

Database Sharding → Optimisation Technique.

A strategy used while designing server side systems. It used concepts like sharding to make system more scalable, reliable and performant.

Sharding is horizontal partitioning of data according to shard key. The shard key determines which database the entry to be persisted is sent to.

Some standard strategies for this are reverse proxies.



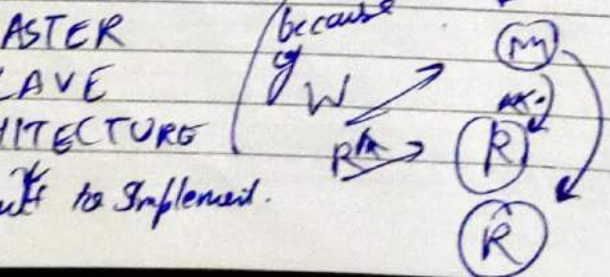
There are three options

- Setting up Hardware → Food → Money
- Adding replica → Problem → "Eventual Consistency"
- Sharding

MASTER SLAVE ARCHITECTURE

difficult to implement.

because of



Shard → pieces of DB.

User Id / Customer ID is a good candidate for Shard ID.

We need a hash func^n which is crafted in such a way that we can introduce the concepts of versioning.

In shard we have a table which map entries to databases.

↓

What if this goes down?

Increased complexity as data is split b/w different databases.

↓

Need to introduce intermediate layer

↓

Responsible for mapping.

2 ways

↑

Partition Mapping.

NON UNIFORMITY OF DATA.

Routing Layer.

Re-sharding is freq in this case.

↓

{ Occurs when one client has lot of data }

Pros

- Scalability.
- Availability & fault tolerance

Cons.

- Complexity

we need to build.

Leads to "Scatter Gather"

Codec → A way in which we compress videos.

Best practices:-

- Creating Index on Shards
  - ↳ Meaning: People in New York >50 years will lie in one Shard

---

⊙ HOW NETFLIX UPLOADS NEW VIDEO?

There are several things involved.

1. Different Format (F no. of formats)
   - → high
   - → Low
   - → Med

   ] → codec.

2. Different resolutions (R no of resolu^n)
   - → 1080p
   - → 720p
   - → 480p

No of videos processed = F * R.

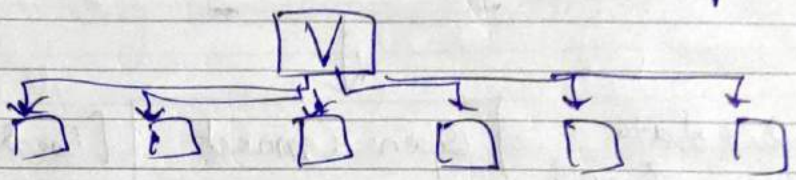
Automatically adjust video quality

↳ Adaptive Bitrate streaming

Risk: What if System shuts down? etc.

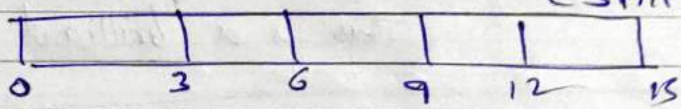
Solution: What Netflix does:?

→ Break a video into multiple chunks.



→ Then we process each chunk.

Video file: (3min each).



What if it breaks on some action scene.

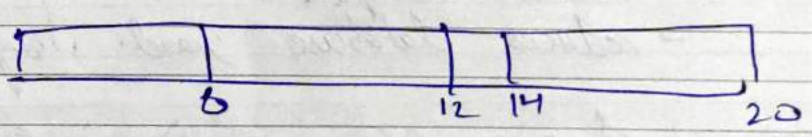
So what it does is ↴

NOT to break on TIMESTAMP

↓

INSTEAD break on Scenes.

New file:

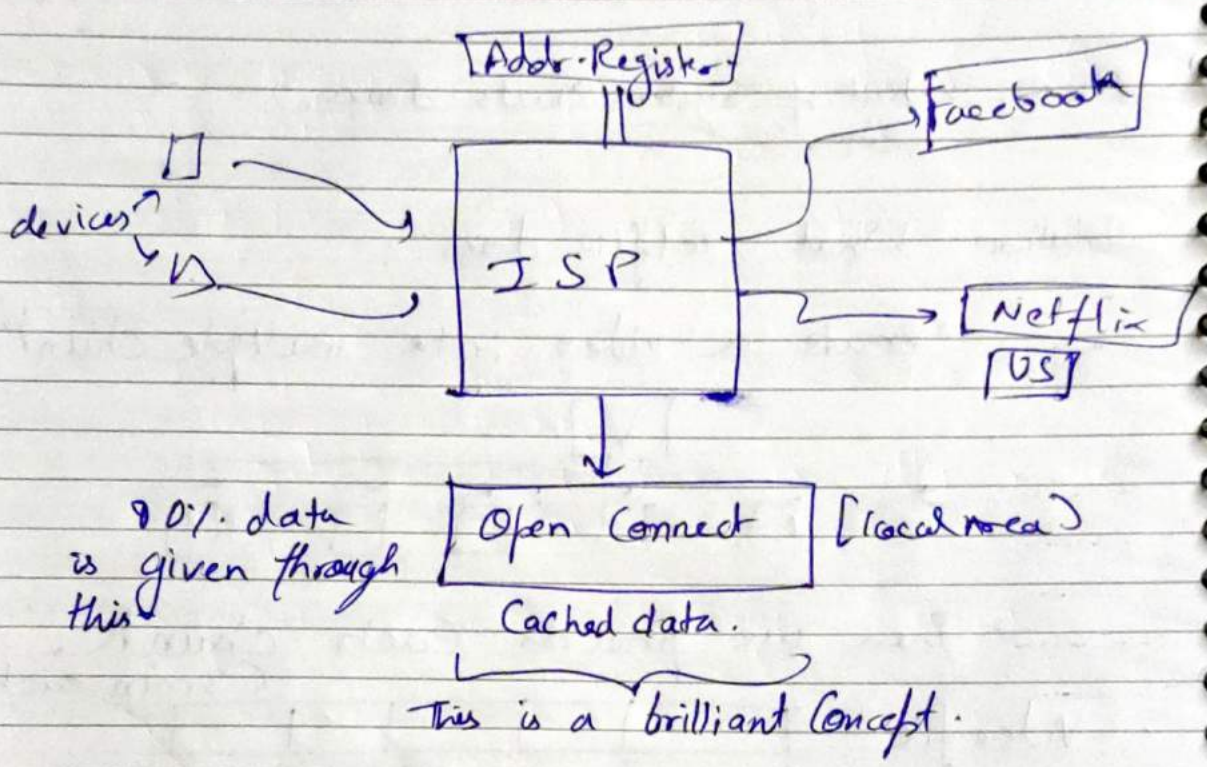


Uses recomendo" algorithm with 2 scenarios.

→ Sparse

→ Dense.

} uses Amazon S3 to Store chunks of Video.



~~TINDER as a MICROSERVICE Architecture~~

~~Da~~ Distributed Caching.

Why :-

- To save network calls.
- Avoids recomputations.
- reduces database load.

} speeds up responses.

Cache stored in SSD, which is extension of data on cache is counter productive

increase search time.

Cache should have most relavent information

Cache

Policy → The way you decide to load and delete data from cache.

Eg: LRU - Least recently used.
- FIFO - LIFO.

LRU

Sliding Window based policy → developed by Google.

Thrashing: When you save and delete data from cache without using it.

Where to place

Problems :-
- Extra calls.
- Thrashing.
- Consistency.

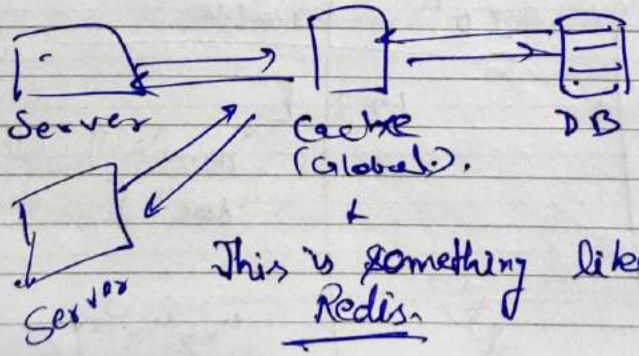
Near to database

Near to Servers.

Near Server

→ If small cache → Keep In memory cache
↳ If server fail, cache fails.
↳ but its faster, and simple.

Global cache



- Slighly Slower.
- More Accurate.
- We can scale redis cache.

This is something like Redis.

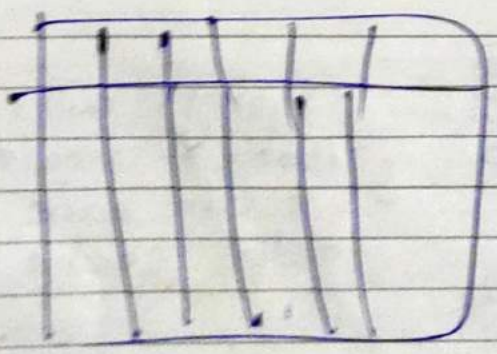
① Write-through → While updating info, update on cache as well, then DB
② Write-back.
↳ Update DB and then cache.

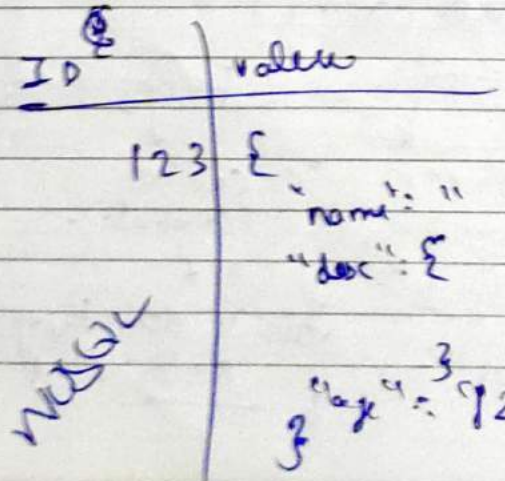
Problem:-
→ What if there is other servers with in-memory cache.
So updating only one is silly, we DONT use this.

→ In write-back, the only issue is speed.

How can we do better?
↳ Use both, depending on the info.
Critical data → Use Write-back
Normal data → Write-through.

SQL and NO SQL





Benefits of NOSQL.

- Insertions and retrievals are cheaper.
- Schema is easily changed.
- Build for scale.
- Good for Metric (finding aggregate).

Drawbacks of NOSQL

- Not build for Updates.
- Consistency Issue. (ACID not Guaranteed)
  - ↳ Main reason By why financial system don't use these NOSQL databases.
- Read time is slower.
- Relations are not implicit.
- Joints are HARD.

Eg: Same ID with diff data.
↑ SQL give ACID property

Consistency

C A P	A C I D
Consistency	Atomicity
Availability	Consistency
Partition tolerance	Insolation
	Durability

- Two Generals Problem.
- Two phase commit.

Database Design Tips.

choice of DB ⟹ It doesn't impact funtional requirements

Factors :
- → Types of Data → structured / Non-structured.
- → Query Pattern
- → Amt. of Scale.

Always use some caching solution.

How it works → we have a key & value.
Eg: Redis
Memcached.
etcd.

File System (for static content, eg, image, video, etc)

for this case we use BLOB Storage → Eg: Amazon S3 + CDN. (along with S3)

CDN → Distribute content in various locations. Worldwide

Primarily stored in S3 and cached in Data centers

Text Searching Capability. (Product like Netflix)
Seach movie.

↓

Text Search Engine. ⟶ Eg: ElasticSearch or Solr

↓

Solr are build on top of Apache Lucene.

↓

Provide searching capability

Also support FUZZY search

↓

O. AIRPORT } Identify this
AIRPROT }

The above are NOT databases, these are search engines.

Database Gives Guarantee that data will not be Lost.

For Metric Data.

↓

Time-Series Database. ⟶ Extenn of RDBMS.

↓

Sequential update in append-only mode

We query for last few min/hours etc.

↳ Eg: Influx DB
open TSDB

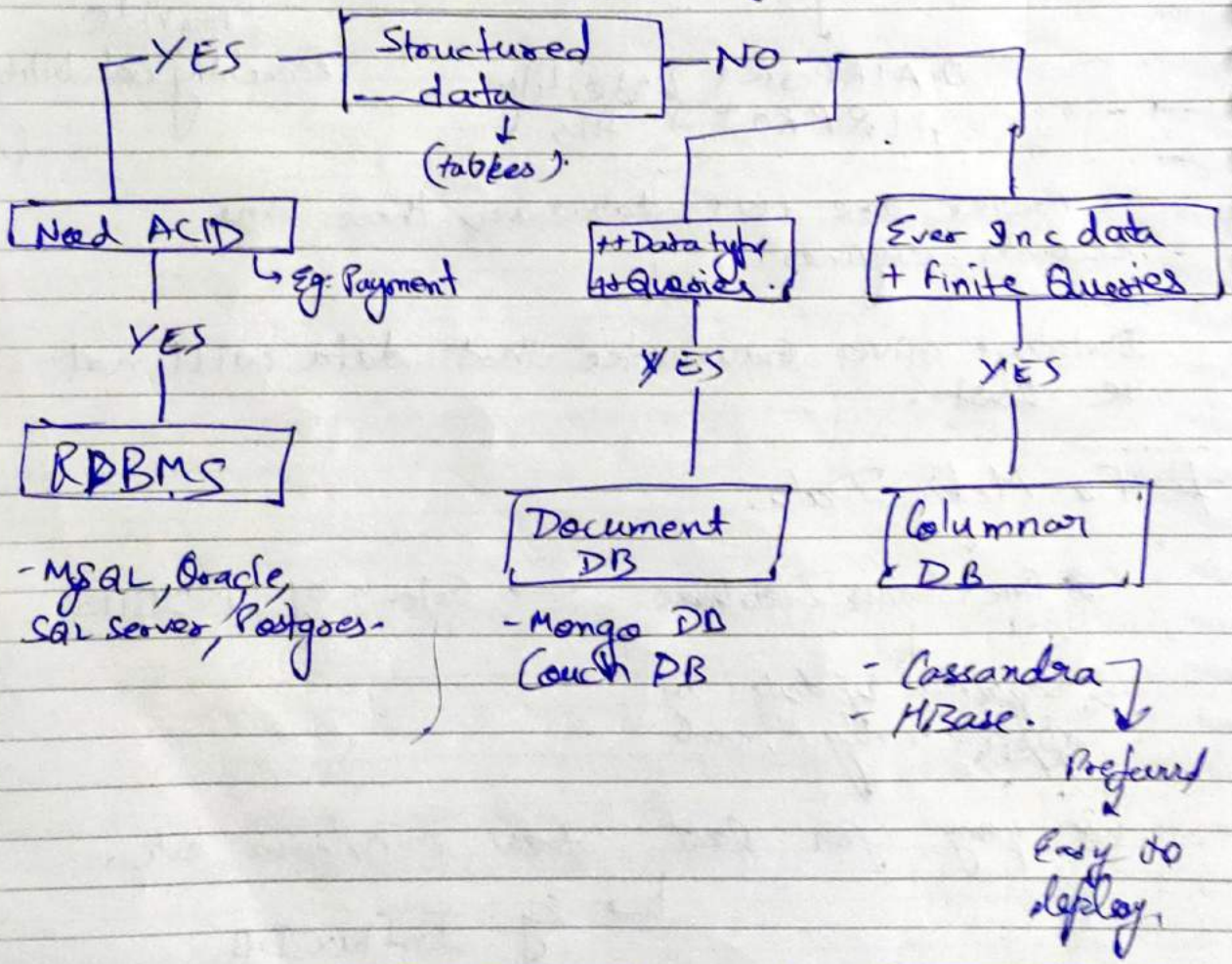
Soln Pattern

- Sharding Data
- Replication type
- Write ahead Logging.
- Seperating data & Metadata storage.
- Load distributions.

Large DB for Analytics (Amazon, Uber, etc)

We need DATA WAREHOUSE → Used for offline reporting.
↳ eg: Handloop
↳ for batch (Slow) processing.

In RDBMS → last option is Postgres.



low cost → Go for open source.
High cost → Production Service.

Helps us choose tech & requirements

Questions to ask.

- → Location.
- → Type of Data.
- → Type of Queries.
- → Scalability.
  - Users / Customers.
  - Scale (read & write)
  - Performance.
  - Cost

Extra things

- → Alerts
- → Monitoring
- → Backup.
- → Logs.
- → Caching (Netflix : Open Connect)
- → Shardn + Master-Slave.
- → Loadbalancer / message Ques.
- → Images (FileSystem) + CDN.

To ensure no lag, low latency & high availability.

→ CPU Utiliza
Disk Utiliza
throughput
log in Kafka
↓
add more nodes

Non-functional (Write down).

- Scalable (1000 req/sec)
- High Performant ()
- High Available

- Recall CAP (talk about ()
- Cost (Everything)

1. → Question to ask
2. → func^{n}al requirement & non-functional (write)
3. → High level archite^{n}
4. → Define Data Model.
5. → Database Choice
6. → (Connected Scale)

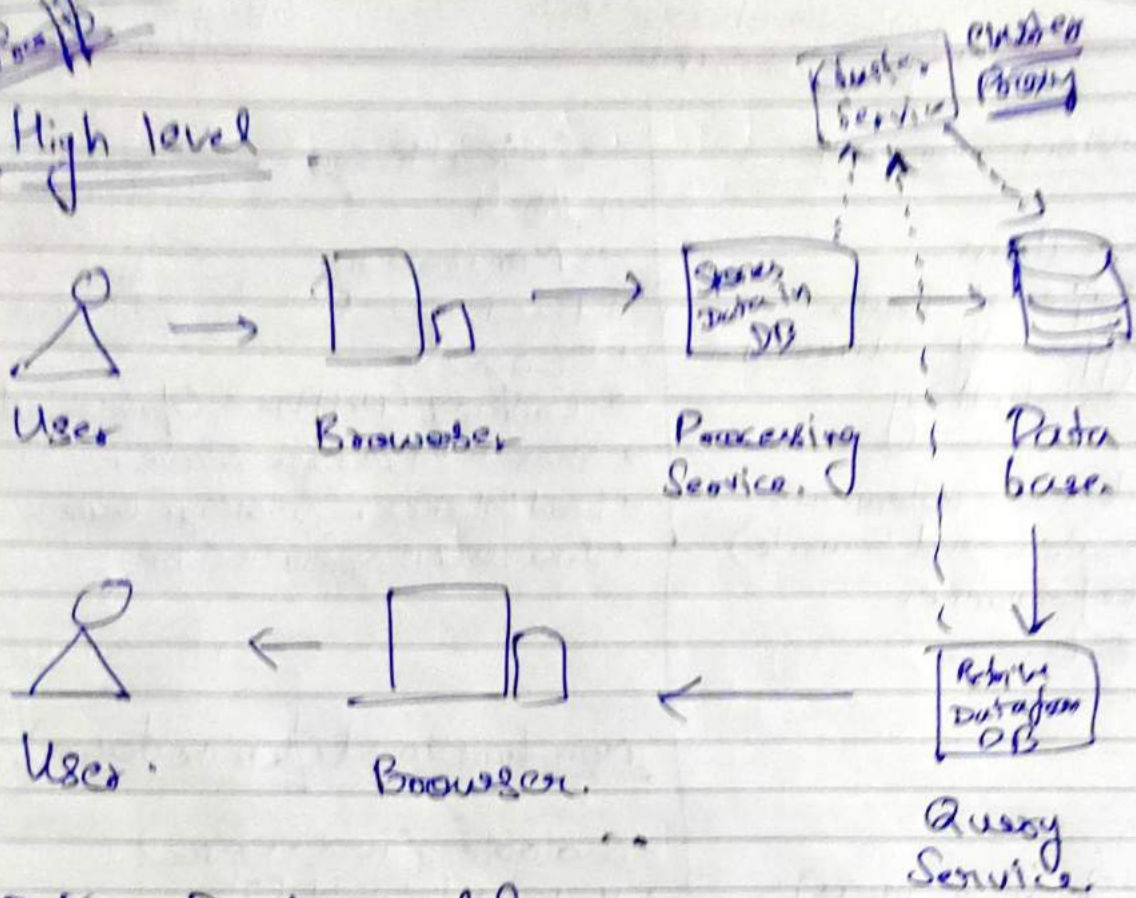
Highlevel

Next Page

When trade off, they try to get best of both worlds

Push

High level



Defining Data model

table / Document / etc → Pros & cons of Each

Stream data processing ← Every Click → Individual Event.

Low latency & Quick response

Every Min → Batch data processing.

Aggregate data

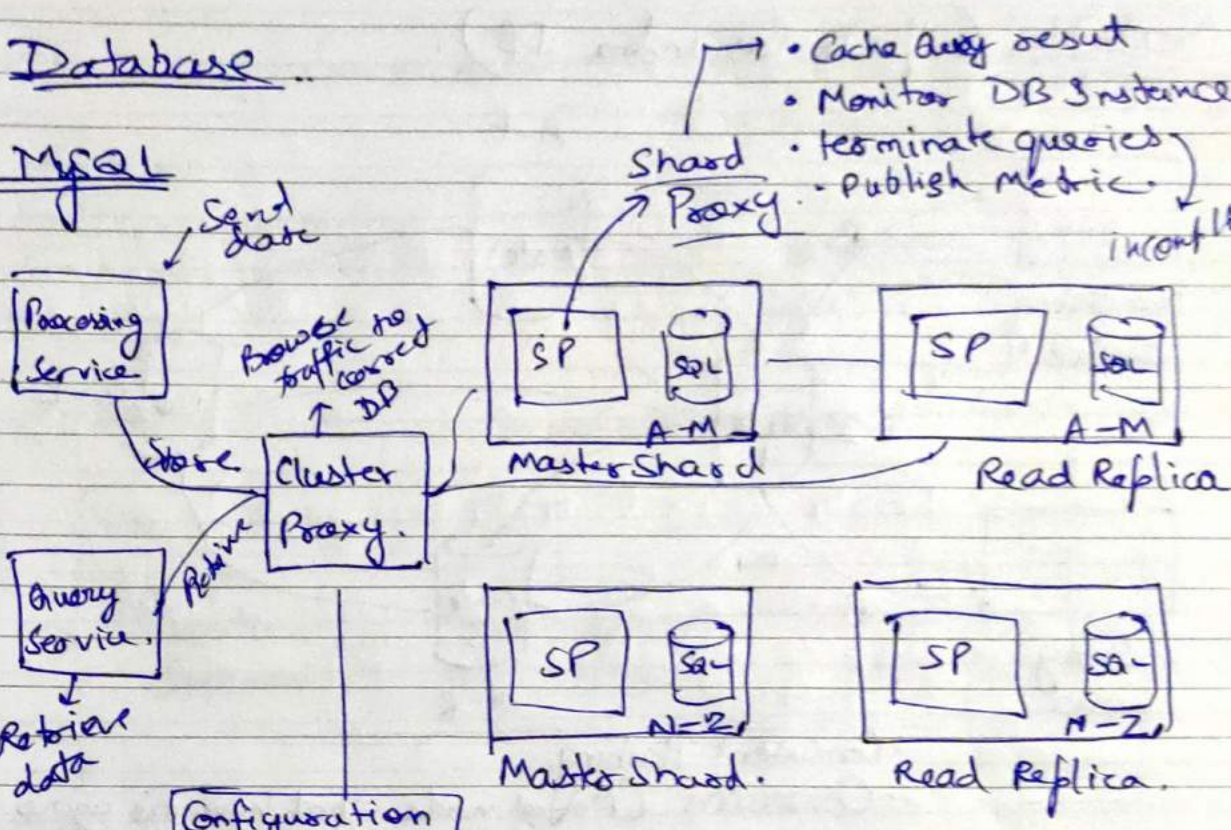
Takes some time to reflect.

You can combine both Approaches.

- Scalability
- Availability
- Performance

Database..

MySQL

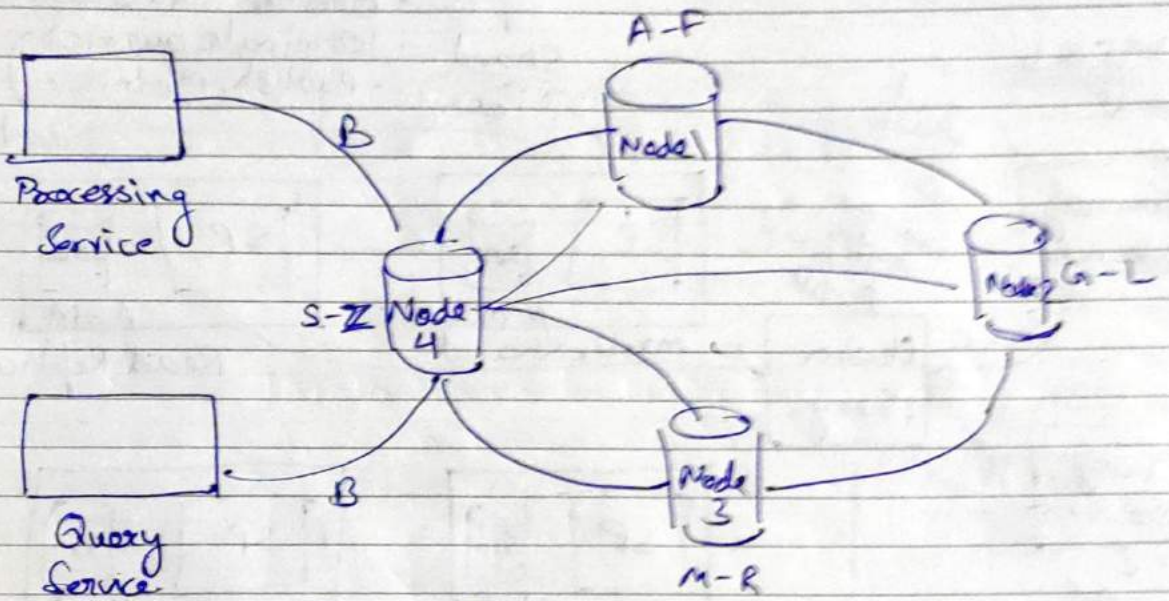


Above Design addresses all things

Scalability
Availability
Perform

Problem of Master-Slave is Reason for "eventual Consistency".

NoSQL (Apache Cassandra DB).



- Consistent Hashing
- Quorum (No of node that have to agree on response).

#NoSQL are of 4 types :-

- Column
- Document
- Key-value
- Graph.

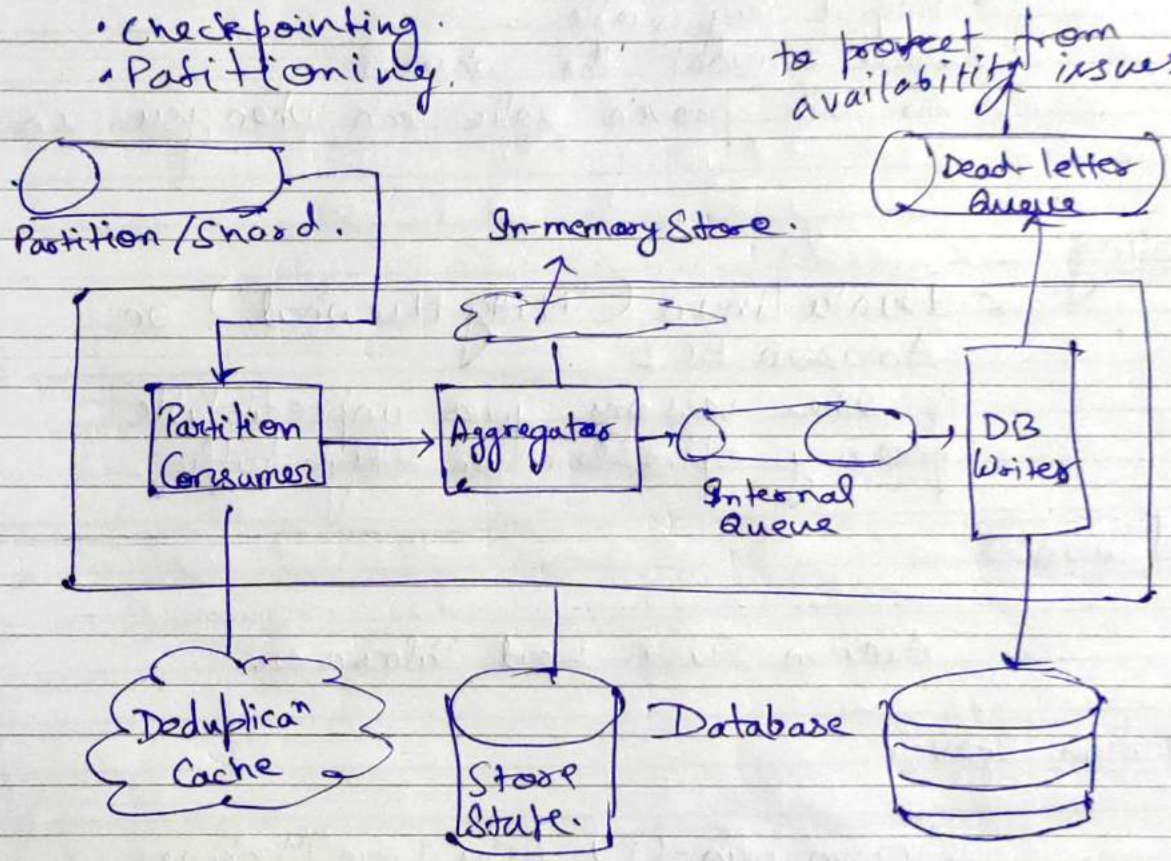
BUILD SOLUTION IN FIRST 20 MINS..

Latency Throughput → Numbers

- Learn 'adv/disadv. of all database designs

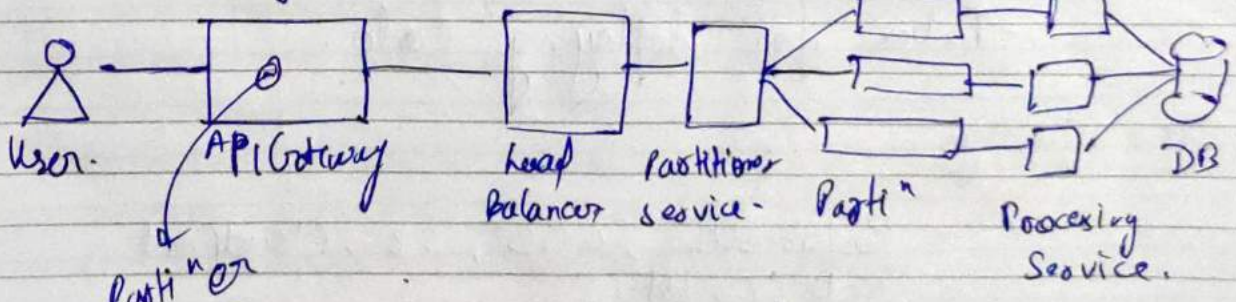
Processing Service
- Scalable = Partitioning
- Reliable = Replica^n & checkpting
- Fast = In-memory.

- Checkpointing.
- Partitioning.



~~# Data Ingestion~~

Data Ingestion Path



Instagram

Poinciples

- Keep it very Simple.
- Don't reinvent the wheel.
- Go with proven solic tech when you can

Hosting

- Ubuntu Linux ("Natty Narwhal") on Amazon EC2

Ps: Other versions gives unpredictable freezing episodes on high traffic.

Load Balancer

- Amazon Elastic Load Balancer.

Application Server

- Amazon high-CPU Extra Large Machines
- WSGI → Gunicorn (Easy to configure) which is less CPU extensive
- Fabric for deploying code.

Data Storage

- PostgreSQL, EBS*, S3, Cloudfont
- ~~Redis (a bit)~~

90% Customers
↳ Use 10% Workflows

Caching

- Redis, Memcached
- Amazon Elastic cache.

Task / Push Notifica^n Queue

- Gearman for Task Queue
- pyapns for push notification.

Monitoring

→ for Graph Metric → Munin.
→ for External Monitor^n → Pingdom.
→ for hard^n notifi^n → PagerDuty
→ for Py Error → Sentry

Run a script whenever we send a new msg.

```
CREATE TRIGGER name
WHEN EVENT
EXECUTE
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure
```

Skeleton

```
CREATE TRIGGER sendMsg
WHEN INSERT
ON Message FOR EACH ROW
EXECUTE stored_procedure.
```

System design interview is not about building a working system, it's about measuring my aptitude

⊙ For Storing Images → File ✓
→ Blob ✗
↳ Binary large object - Storing in Database as Blob.

What DB provides :-
1) Mutability
2) Transaction → ACID
3) Indexes (search)
4) Access control.
→ NOT Required for Images.

Adv of File System :- → BEST
1) Cheaper 2) Faster 3) CDN.

In this we need to store URL in DB.

Protocols to Use
- Internal → GRCP
- Externally → HTTP
- Video → RTMP → Live streaming

Cassandra, Elastic Search → KNOW ABOUT IT

Learn # about trade offs.

FOCUS ON CORE

- Ensure high availability.
- Improves cost efficiencies.
- Helps define & meet SLA's.
- Provide insights.

Why

2 hr → 0.4 GB
= 400 MB
200 MB = 1 hr

Capacity planning: Youtube.

1 B × 1/1000 × 10 mins = 10^7 min

$= 10^7 \times 3$ MB

= 30 TB

Amt. of data to keep raw footages of vid on storage disk.

200 MB → 1 hr.

200/60 MB → 1 min

3 MB/min

We need in multiple resolution.

We need copy for fault tolerance & redundancy

For 3 copies of data

→ 90 TB → 180 TB → 0.2 PB

Alternate approach → We can store video in form of image frames.

- Estimate daily data usage
- Estimate what the cache & how many system of 16 GB's are required.
- Estimate no of processor to process video.