system desgin

system : collection of technology which server sets of users to full fill their requirement

design : artitecture of that system based on the users requriement which serveses people to full fill theire requirement

system desgin : is process of understanding users requirement and artituce a sytem based on those requirement to serve people to full fill their requirement

component of system design :

i)DNS : dns is domain name system

ii)Load balancer

iii)database

iv)key value stores (like distrubeted hash table) – it uses for fast lookup

v)blob storage (blob means : binary large object storage) – it is basically storage solution for unsturctrued data

vi)CDN (content delivery newtwork) – it is gerographically disturbuted group of newtrok or server to provide fast delivery of content .

* It reduce the lookup time.

vii)cach : it is storge where we keep repatedly read and write data , which redcue the operational latency .

why system design ?

we need system design because :

i)to make our application reilable (like our application not down , falult tolerant and consistent)

ii)to make our our application effective (to meet business requirement)

iii)to make our system maintainable (to make our system scalable, like we should be able to do scale up and scale down suppose right now i have 100 active user which are useing my application but what happened when 1000 people are useing my application)

Scalaliblity : suppose right now i have application which have 1000 active user but suppose suddenly i have 1000 user satarted useing my application on regular basis so our application should be scalable to that new user without slowing down like it should be resposnive for the new user with the same speed that is calledd scalebility .

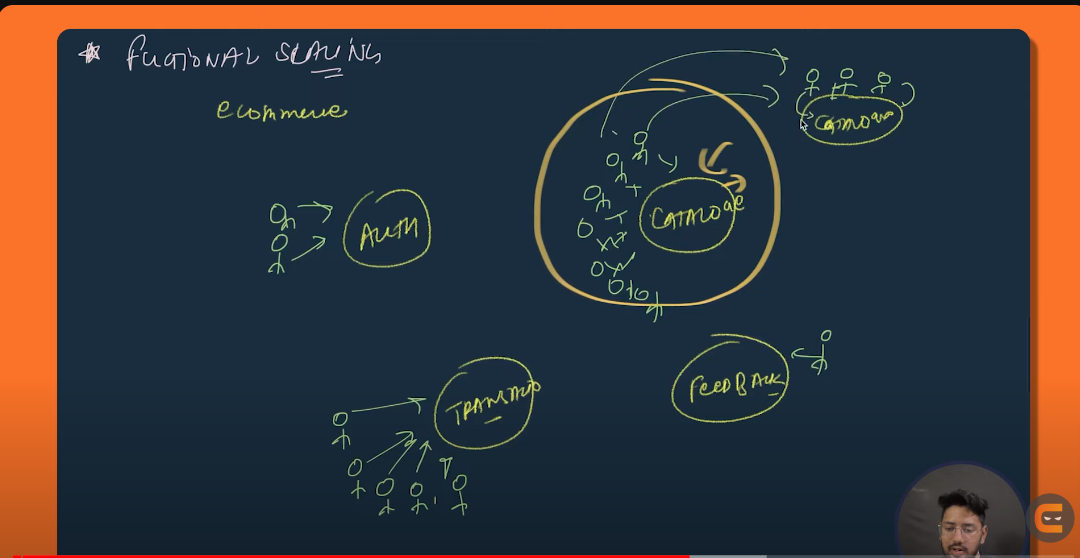
scaleibility is directly propertional to the perfromance of our application

vertical scalling : buying bigger machine

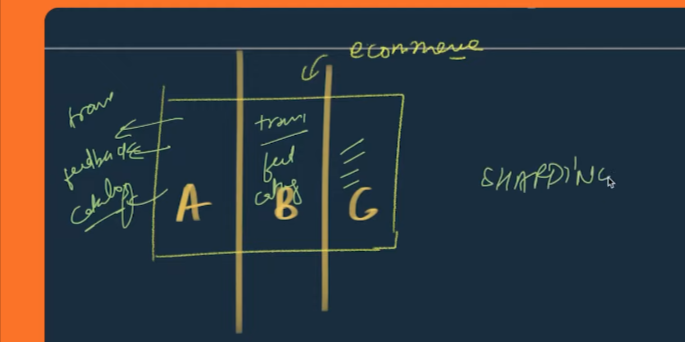
horzinatal scaling : buying more machine

horzantal scaling is also known as homogenous scaling

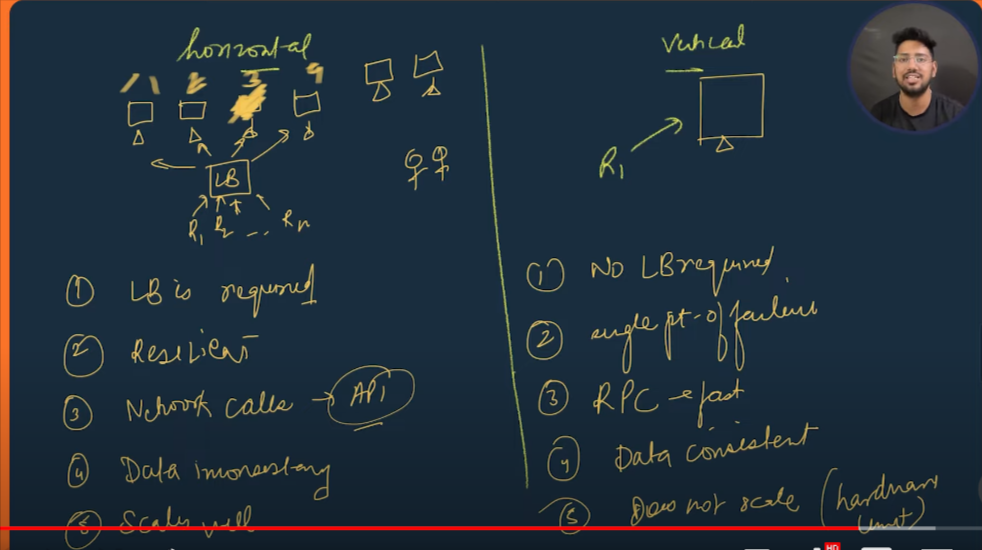
functional scaling : scaling at the functional level



sharding : equally separating the database into some partation it is like a horzintal scaling



diffrence between horzintal scaling and vertical scaling



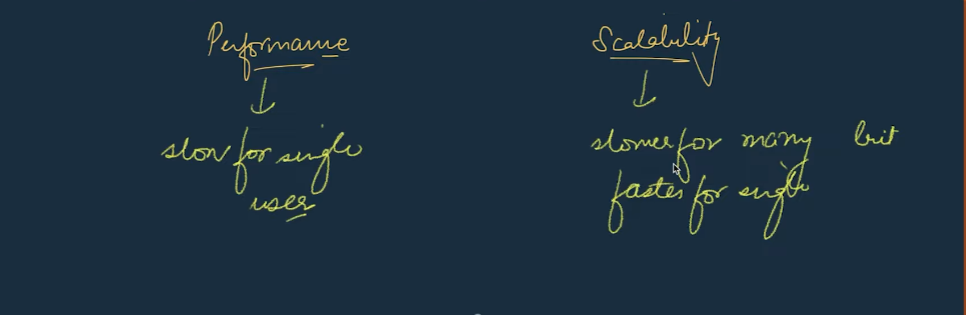
2. lb:load balancer

3. if any of our machine get falult we can migrate the request to other machine so horzintal scalling is resilent and vertical scaling is single point of failure

4. rpc : remote procedure calling (calling at application level )

note : in real word we use mixter of horzintal and vertical scaling

diffrence between performance issue and scalibility issue :



Disturubited cache

in short : cache minimize the operatonal latency by :

i)reduce the no fo network call

ii)reduce Db load

in application we have :

i)cilent side

ii)server side

iii)data base

in normal circumstances our application works fine , but when the no of users incerease there will be increase in no of request on database,internal qurey so there will increase in load on the application which reduces the performance of the application .

In order to keep the perfomrance of our applicaiton we use chahe to reduce the detoratation of the perfomrance of the application .

Q. why not to store everthing in cache ?

i) because there will be increase in serach space (like we will going to have so much data to search upon) so there will be heigh operatonal cost

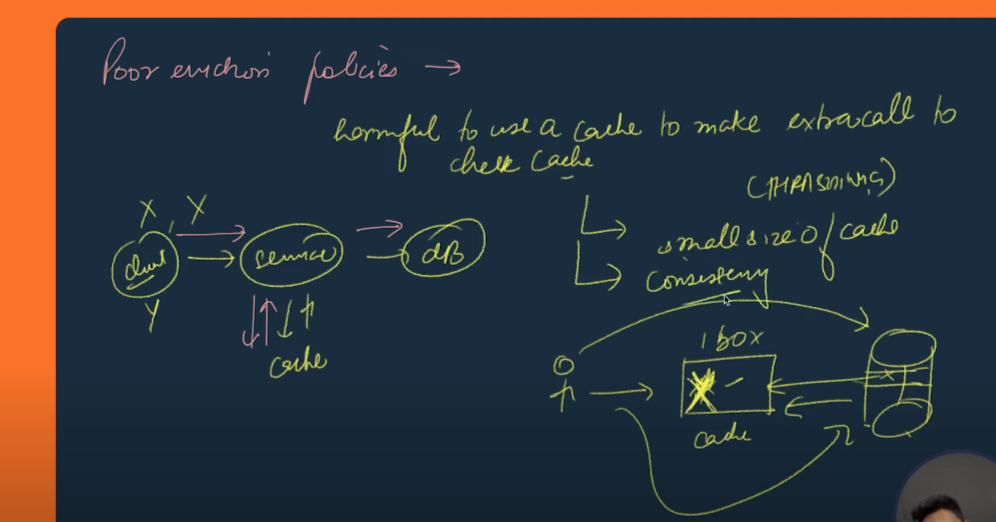
ii)the hardware that used in cache is very expensive

Polices of caching :

i)LRU caching

ii)LFU caches

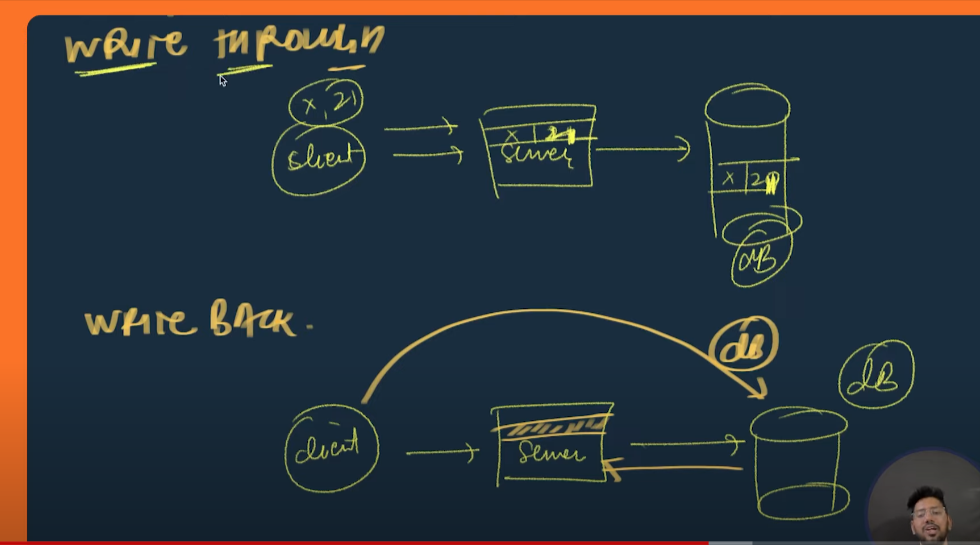
Poor evichens polices :



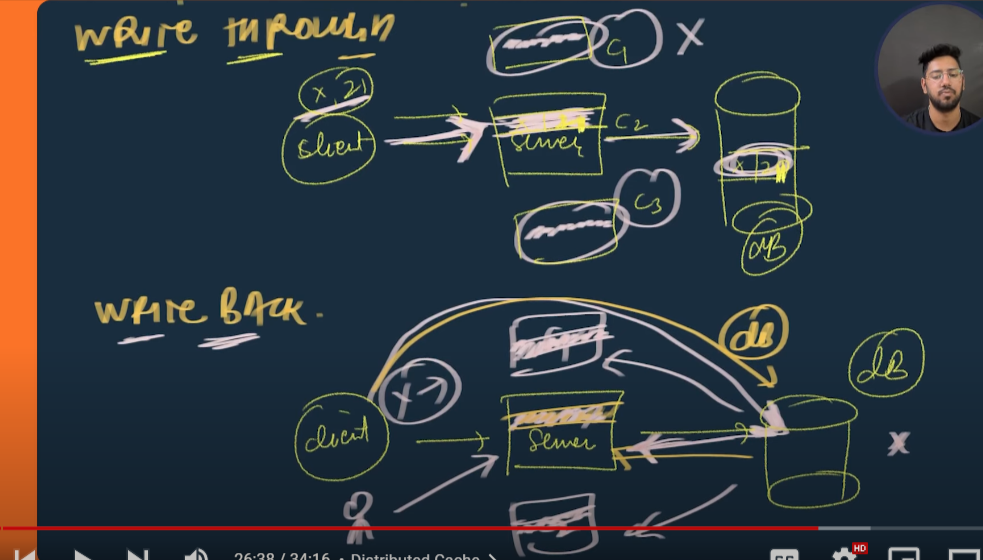
How to update data in distrubted cache ?

i)WRITE THROW CACHE

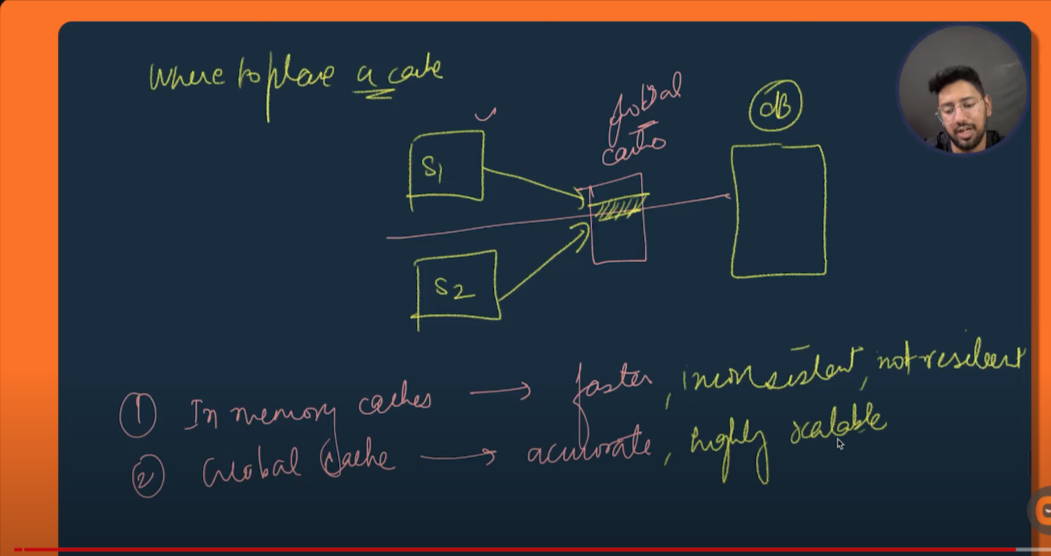
ii) WRITE BACK CACHE



Problem with the write TROW AND WRITE BACK :



where to place cache :



DataBase Sharding :

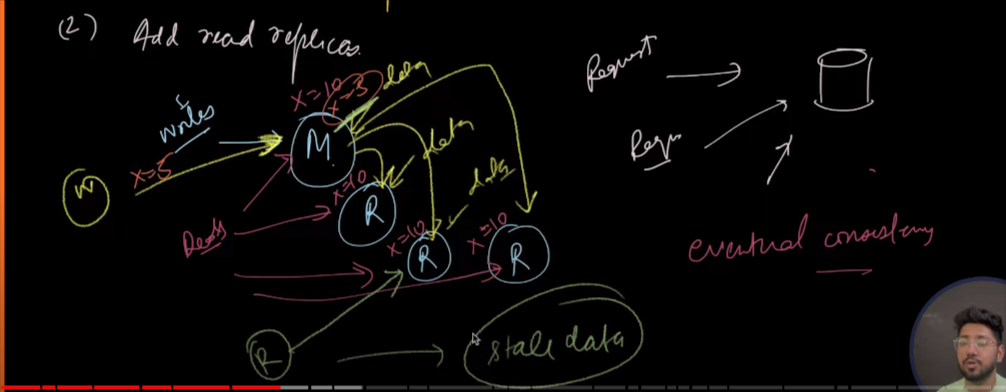
i) it is optimization technique which performance of our system

ii) it optimizes database performance at a very large scale

options through which we can improve the performance of our application :

i)scalling hardware (but it has some limitation like at after some point we will not be able to increase the limit of our hardware)

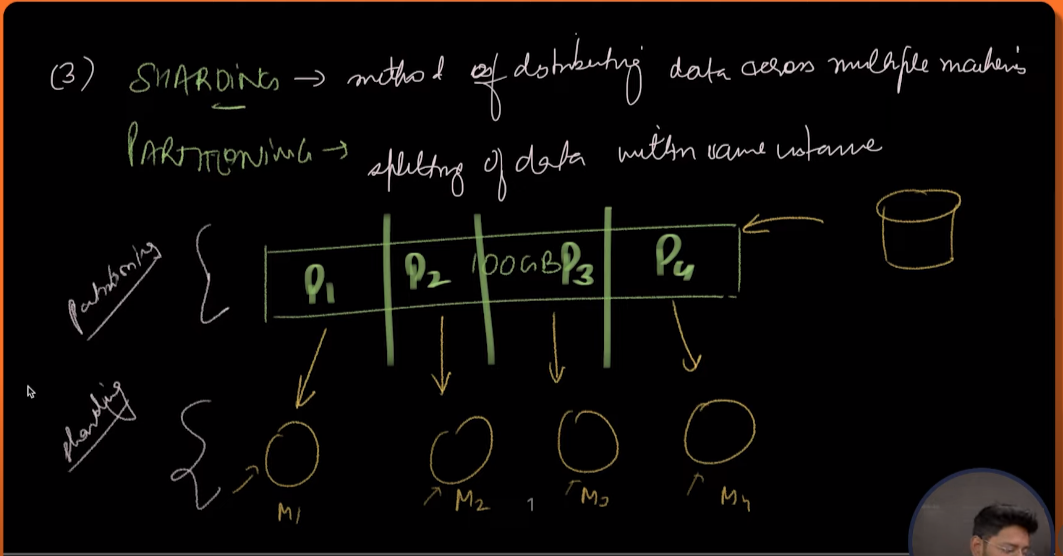
ii)add read replica (most of the time we observe that there are much request to read the data than the write the data like instagram)



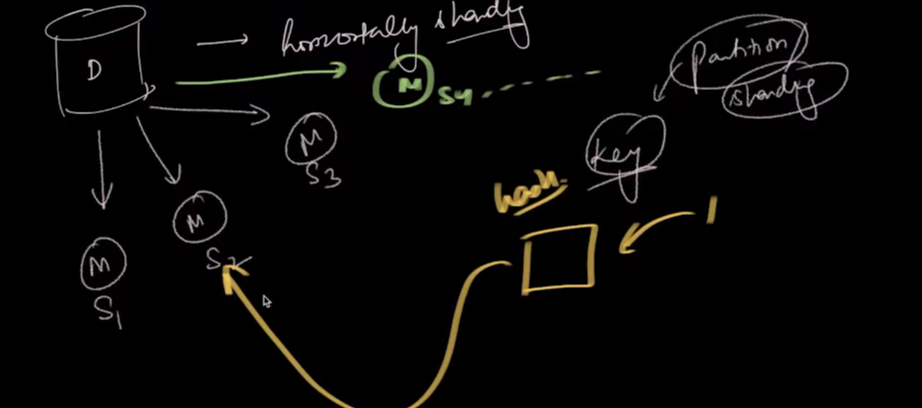
disadvantages : problem of eventual consistency

SHARDING : method of distrubuting data across multiple machine , it occur at database level

PARTATIONING : spliting of data withing same instance , it occur at datalevel



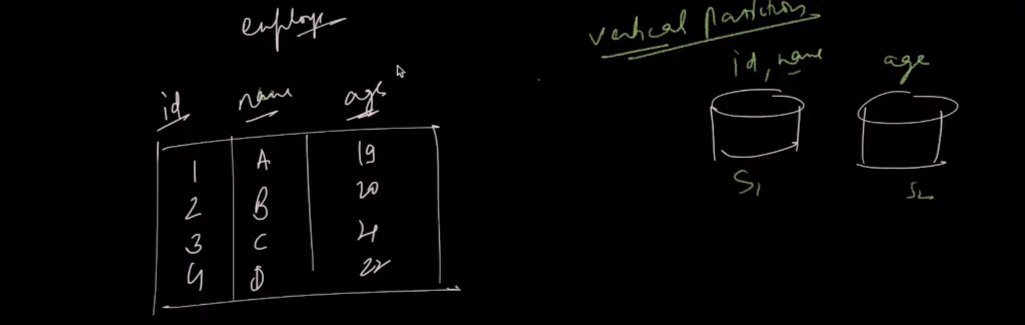
How sharding occur :



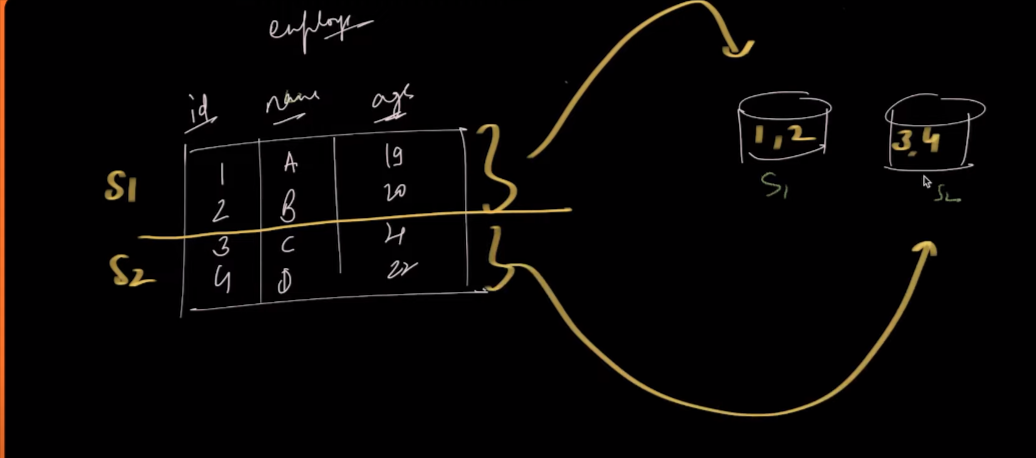
partationing :

i)vertical partationing : spliting on the basis of column

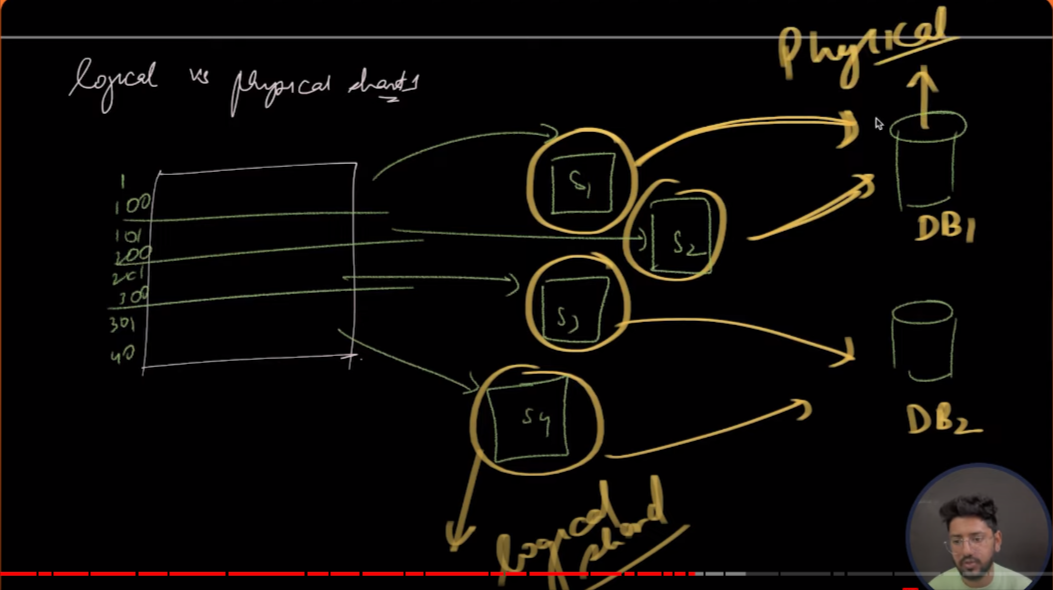
drawbacks : it is less reliable



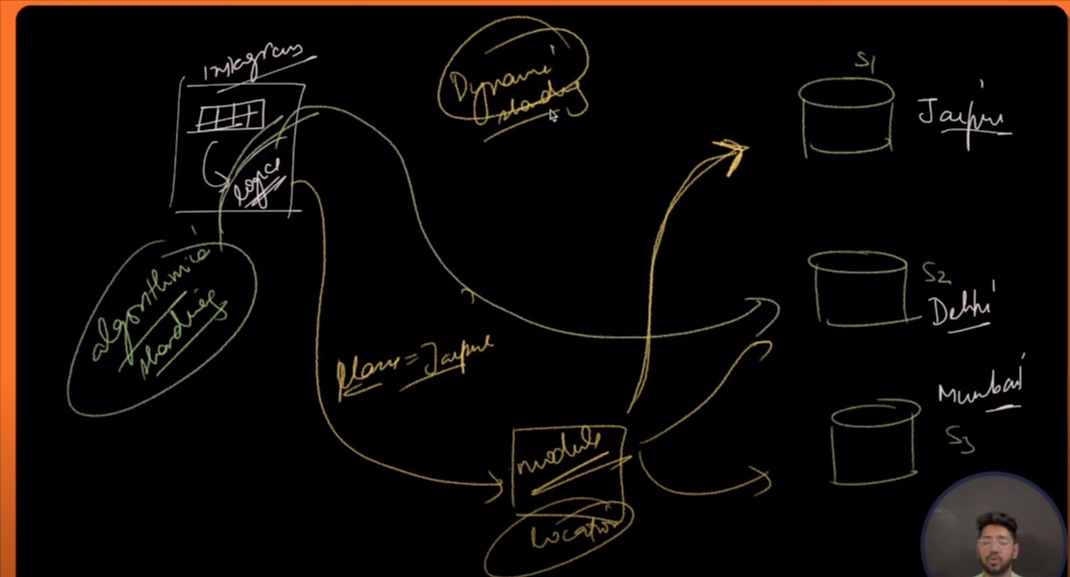
ii)Horizintal partationing :



LOGICAL SHARD VS PHYSICAL SHARD :



ALOGRITHMIC SHARDING AND DYNAMIC SHARDING :



pros and cons of SHARDING :

pros –

a)scalability ( with the horizonal scaling we are able make our appliciation scalble)

b)avalibility and fault tolerent ( previsouly we only one instace of databse it there will any kind of fault the whole data lost but with the sharding we have multiple instace of database )

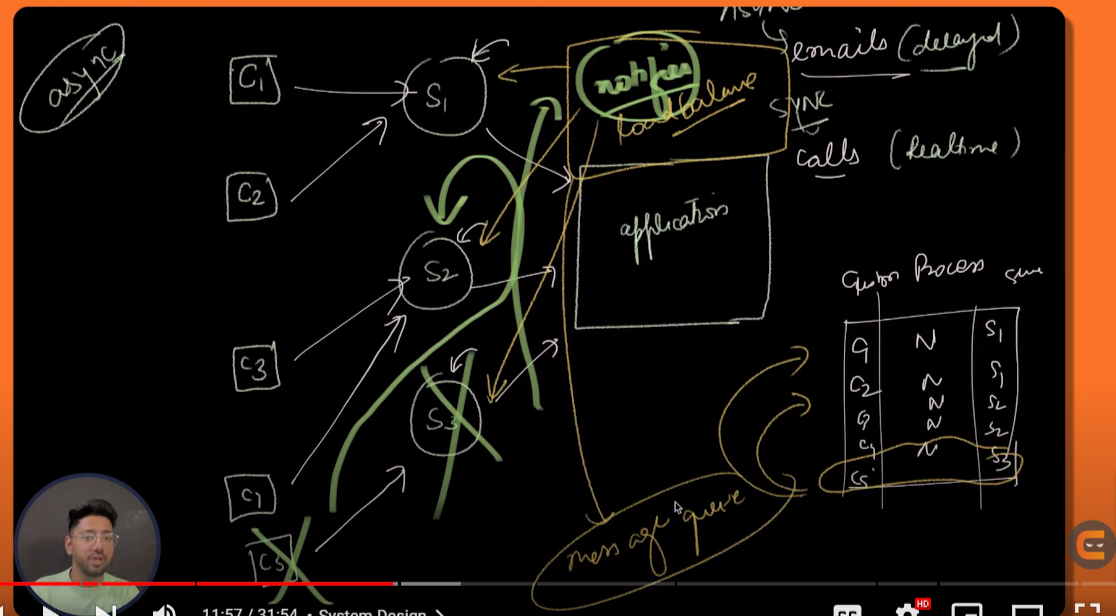
cons-

a)complexity :

i) complixity interm of partationing of data and maping

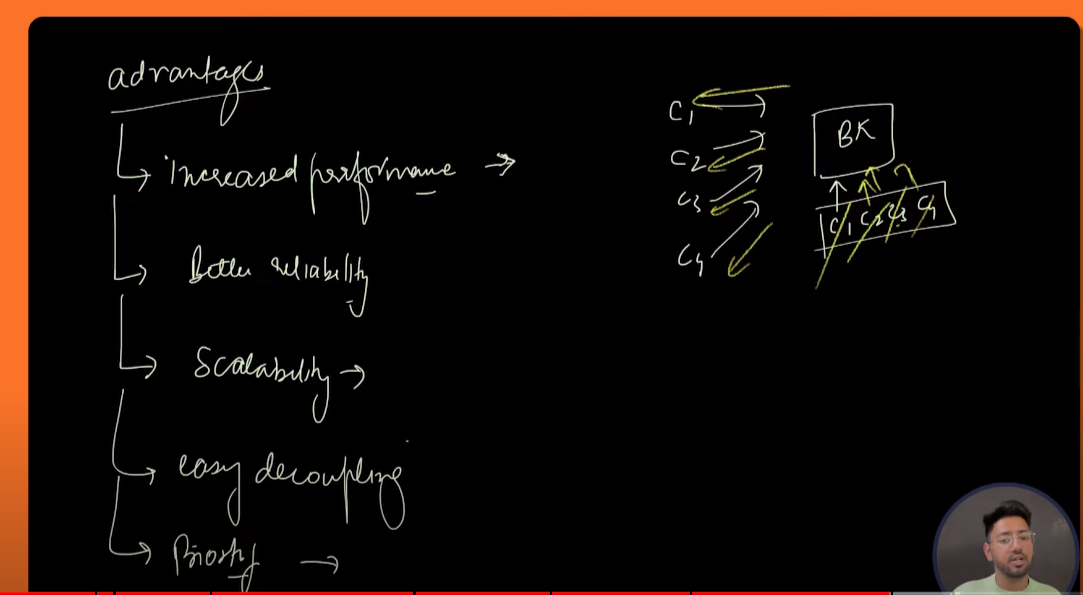
ii)complixity in term of routing of data to diffrent shard

MESSAGE QUEUE : it is intermediate between produce and consumer



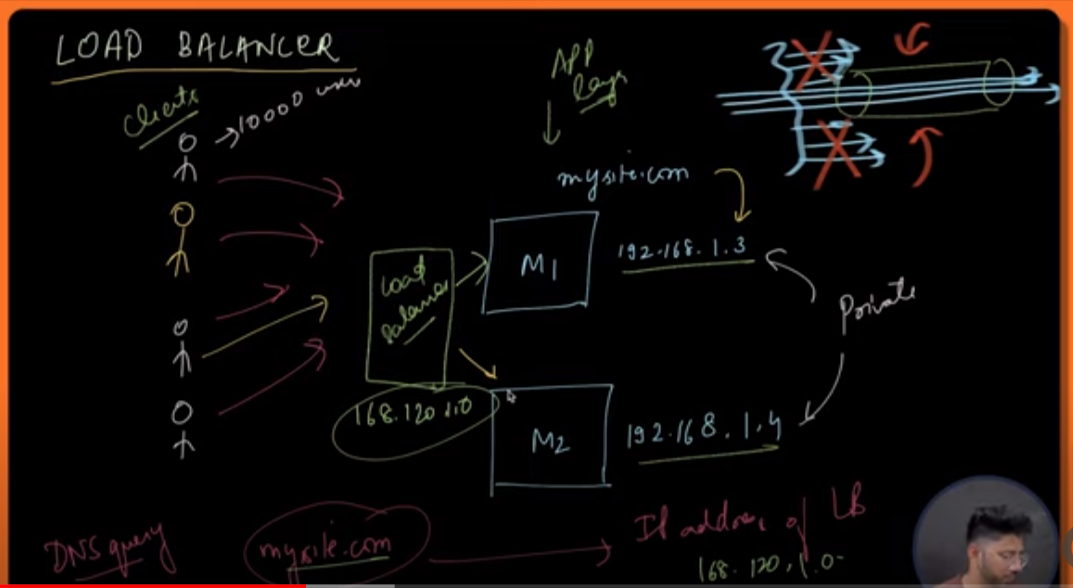
Above thing is called message queue. Ex : domino’s

Advantages of message queue :



Eg of message queue : AWS , KAFKA

Load Balance ->



Ways to which load balancer passes the request :

i)static alogorithm – in static we do not care of the backend server state .

eg :

1. round robin algorithm - what we do is we genrate a algorithm like using modulo so what we do is whatever no of request comes we will going to take the modulo by the no of server prsent we donot rely on the backend server.
2. Source IP hash : in this method what we do is what we have a hash function in that hash function we pass the source ip as the parameter then whatever the hash function return we take the mod by the no of server persent .

ii) dynamic algorithm – in dynamic algorithm we take backend server state into the consideration .

1. In dynamic alogorithm we take care of the backend server state into the considration , what we do is we check we server has the least no of request on that server we migrate the current request .
2. Resource allocation : in this method we check which server has the least memory utilization and cpu utilization on the basis of that we pass the request to that server .

ADVANTAGES OF LOAD BALANCER :

i) Scalability – if need we can add more server .

ii)Abailability – like ever few quantam of time the notifer will going to check the health of the server like which server is responding and which not so because of that request will going to only avilable serve only .

Type of LOAD BALANCER :

i)HARDWARE LOAD BALANCER

a) these are basically machine and they are expensive

ii)SOFTWARE LOAD BALANCER :

1. These are coloud services like (AZURE,AWS)

SQL AND NO SQL DATABASES :

SQL PROPERTY :

i)schema : schema define the shap of data sturctre that we are going to used to store our data in the database . it can be at database level or it can be at data level (like column datatype).

Advantages :

i)ACID Compliance

ii)sturcutred dataset

iii)reduce chances of error due to well sturctued schema)

Disadvantages :

i)can not be horzintally scale becauase of the regid structre )

ii)reading data might be sometime expesive ( because we have to fectch data from different tabel using forien key or any other relation).

iii)strict shema (like for every data set we need to define the data type other properties )

ii)ACID proerties :

ACID : stands for : atomicity,consistency,isolation,durability)

Acid property granatte that database transcition are processed are relibile .

Transaction : it is a database operation , where either all the operaton can be excuted or none of them get excuted .

Eg : bank transcition

When to use Relational database :

i)when we have prior information we should not need to change our shcema like strict shema

ii)when data need to accurate

iii)when there will be many-to-many relationship (like student , adress , and many more )

eg of relational database :

i)SQL

ii) POSTGRSQL

iii)Oracle

Non relational database :

i)non relational database are sclable databases like we donot have fixed schema

ii)data set are in form of collection document

iii)there is no acid property so there might be chance of data inconsistency

type of non relational databases :

i)key-value pair

ii)