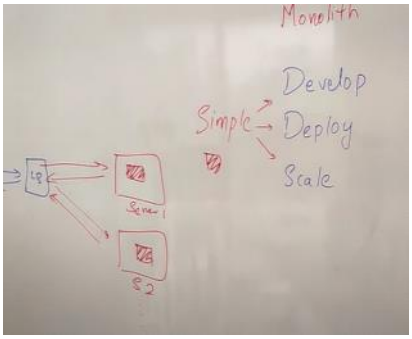
Monday, February 15, 2021 3:44 AM

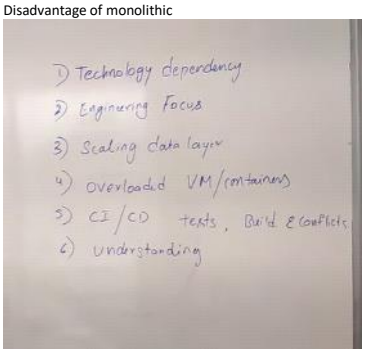
Monolothic design-> Every developer works in same aaplication and put it in one server. Package it. Because all are presnt in same package.



It is easy to scale. We just need to copy the same package and distirbute the load.

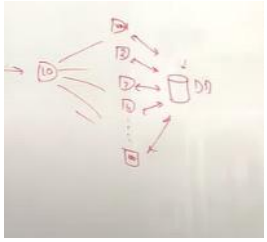
We need to put load balancer between two server it will be distributed by load balancer. Example round robbin technique. Keep adding package on server

Disadvantage of monolithic



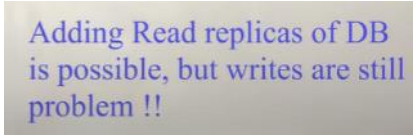
Technology depency-> We have to be dependent on same technology if near future someone provide better feature we cannot use

Engineering focus->Team who only focus on order management. But it in this it is diifcult. They are tighlt coupled



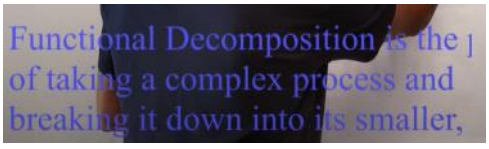
There can be lots of virtual machine but it has only one DB which Has too suffer as it might have transactional update

All read and write used to happens on single databases heavily bombarded



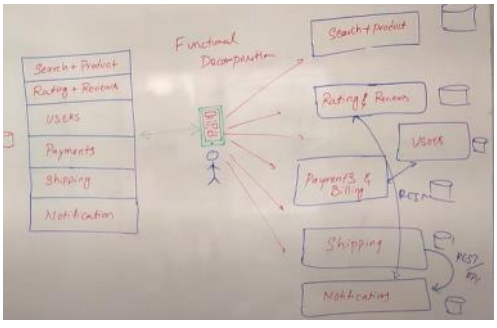
C1 &CD & Test-> It has to test all code as it is tightly coupled. Takes Long time

Microservice



1	whole monolithic breakdown in functional decomposition
	Eexample->Payment & notification wil be one module

Serving there own purpose
There are individuall packaged



Now if we want to search then we have to see in the search +product module.
Earlier module used to interact with same databases join was easier

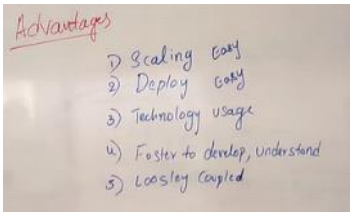
Sepearted deployed

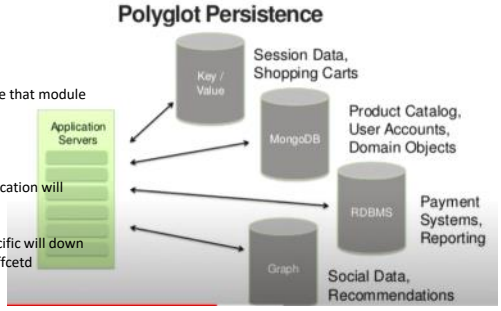
If we want to intearct then we have to do using rest api calling or rpc

Now every service should have there own databases

Now different module can use different databases as per there convinces. Like search can use elastic search and notfication can use rdbms

If we have too many data then we can use nosql databases





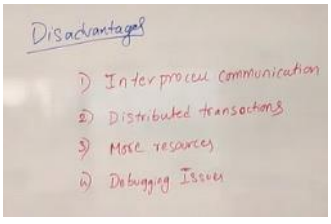
Scaling-> If user searching more then we can scale that module

Deploy time reduce as it is small application.
Start time also reduce

Losely coupled-> Changing one part of code application will Not affect other part

Instead taking down whole application. Only specific will down
Exapmple if search was down then other is not affcetd

High availbility lots of traffic



Data duplication can be also there

****

How to scale->

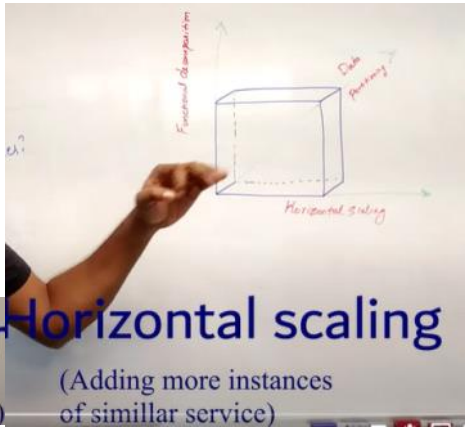
Like user search more but only choose one or two to buy

****

How to scale->

Like user search more but only choose one or two to buy

Scaling rule



Like search service can be decomposed in more service

Data partioning can be same like table partion
Example- Currently it used to searhc from A to Z name
Then we cannot add one server-> then one will handle to A to n
And other o to z

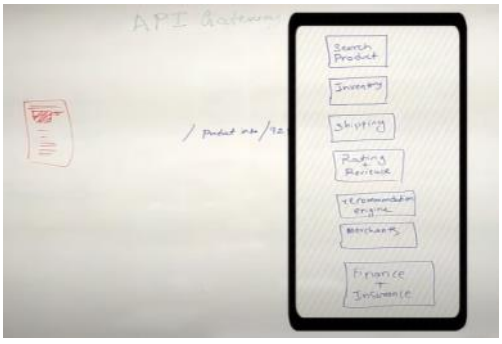
****

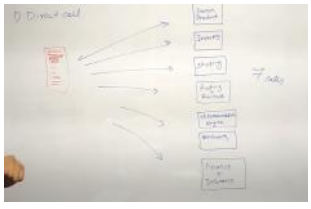
MICROSERVICES ARCHITECTURE | API GATEWAY |

Direct call

If this was monolithic api and want to access then we have to only expose one api which contains all the information in one json file. Only one call

Decmposed evrything deployed
Then we have to call each module individual or one by one.

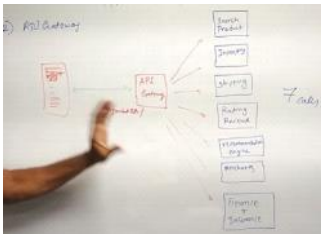




In this we have to make 7 calls to call each api
It is tightly coupled

If we want rating & reviews in two microservice then
We have to update client code also

Api Gateway call



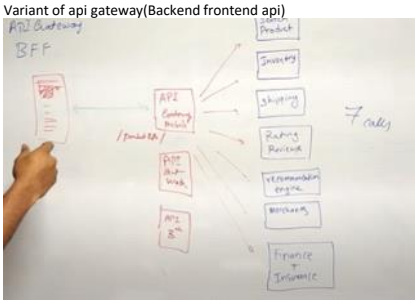
Then we have one more layer between client and server which called as API gateway. It is like front facing of all microservice we have in server.

If client want to call microservice the it cannot access directoly. Client has to make call to api gateway. Then api gateway will call all 7 to get info

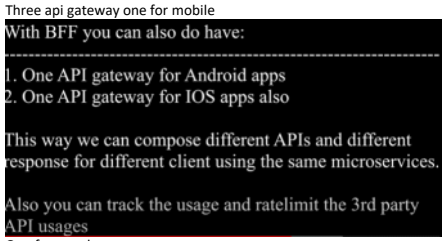
It same as one Api in monolithic which is decomposed in 7 Api call.

Api gateway can call 7 parallel to retrive data which is faster.

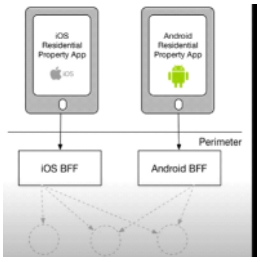
Variant of api gateway(Backend frontend api)



Three api gateway one for mobile



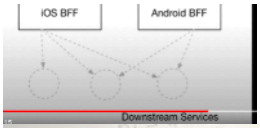
One from web
One from third party



Advantage

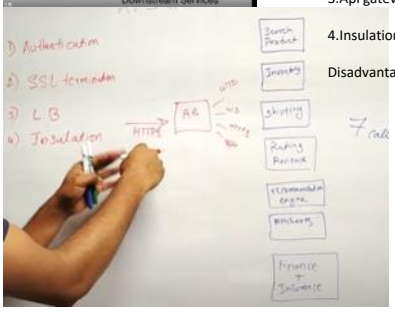
1.Authentication-> Can be happening authentication to api gateway rathera than all microservice authentication

2 Ssl termination-> Making call from client to api gateway using HTTPs. It doesn't have to be microservice till all microservice. It should only verify in api gateway.
All internal call should be http. Or web socket, or rpc
In internal world we can use anything for calll but for exteranl word we have to call HTTPs



2 Ssl termination-> Making call from client to api gateway using HTTPs. It doesn't have to be microservice till all microservice. It should only verify in api gateway.
All internal call should be http. Or web socket, or rpc
In internal world we can use anything for calll but for exteranl word we have to call HTTPs

3.Api gateway install Load balancer if we increase number of server of search product or scale it load balance between them

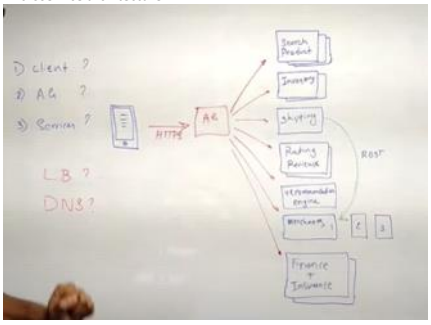


4.Insulation-> No one from outside can acces from it. It is possible to decompose microservice in future. No effect to client

Disadvantage-> Hopping increase. Complex increases

****

Microservice architecture



Search product has 3 instances

Inventory has 2 instances

Client wants to talk different api services?
If we are not using api gateway then client will directly talks to api service.

We cannot configure static ports. Because this instance are very dynamic

If we connect using Api gateway then who will update the instance in api gateway

What if any instance are down? Who will be update ?

One microservice can call other microsrervice??

Loadbalance also need to infrom all the instances which are running.

Service discovery-> It is a pattern by which discovering all the running instance of microservices

Service register-> A service register is database which has list of all microservices.. And instances . It is a database contains address of server

How service registry get latest instances of any microservices instance?
Two way-> Self registry-> Microservice automatically register if we scale shipping microservice. Update microservices-> Talks to service register talk to port. Which added to service register. Network adress also updated. After every 10 sec or 1 minute microservice will updating locating in service registory

In this microservice will not talk service registry go and ask to each microservices. What is your adress port.
When cluster add then event ragister will talk to that microservice and it updated in service registry

If api gateway wants to talk any microservice then it will fetch from service registry and fetch that

If client want to talk directly to microservice.
Then client will talk to first service regsitry and ask for latest microservice of merchant of product service. So it will give 3 address and port so. Client can use loadbalance and then it can use it

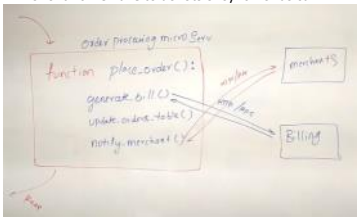
If sometime providng the address service registry all microservice goes down then it should have threshold value to client if 10 sec don't work then again take latest copy from service registry

In server discovery it talks to api gateway
Api gateway talks to service registry

****

Interprocess communictaion-> In this how one microservice talk to other microservices

In monolithic we have to do lots of synchronus call



Request

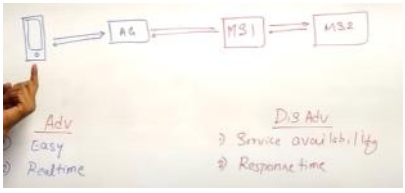
Response

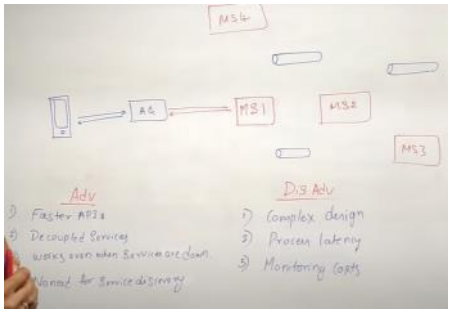
Synchronus intercommunication call-> Calling different microservices synchronusly

When client calls to api gateway then api gateway calls the microservce A and microservices A call microservices B
Microservice A waits until microservice B sends back the response.

So we always expect microservice B up and running.
So this disuise or break the microservice architecure. Because we want the microservice to be independent

If there are so many microservice call 1 to 2 to 3 to 4 to5 so it increase the time. Latency added each microsercixe calling





Asynchronus service model

Each microservice has own databases.
And each microservice wants that there database should get updated

If client give order and it talks to api gateway and api gateway to MS1(order microsrvice) then all microservices wanted to know

So MS1(order microservice) put on the queue 1 and all the microservice which is listening to that queue will get updated and other microservice will get updated.

Asynchronulsy communication happen using the queues .

Request and response is faster.. As Microservice1 execute and write order on queue1 or queue2 then it need to send back the response to client without talking other response

So if Microservice2 is down but microservice1 will load data on queue when microservice2 will get up the data which is loaded will be consume by microservice 2 when it will up

So no microservice is tightlt coupled to each microservice

We also don't need to use service discovery as microservice1 doesn't need any address of other microservice. Just it has to put on queue. We only need queue which can be hardcoded

Disadvantages
If queue is full and microsrevice is not sacled or down then it can leak the data

Additional delay we doesn't gurantee when all the work will be finished by queue

********** **********

Circuit breaking is a design pattern which are used when there are services interacting with other services for some type of network calll like HTTP and Rpc calls

Example-> Time out exception bad we need to handle this

M1->M2->m3 talking synchronauslly. If it is working then we need to take response from each and return back to client.
But there are sometime when services are not avialbale they goes down
What happen M3 is down.
So all Microservice ccalling m3 will get error. We don't want to send back the error.

Consider M2 has mechanism retry of M3 5 times..in same way m1 will have retry of M2 5 times.

So if M3 haveliy loaded it cannot process new request even though it is up.

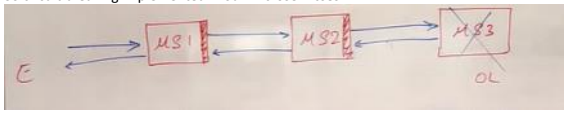
If M2 retry 5 times it make M3 life more hell because there is already loaded on top of it M2 is calling 5 more times.

Even they have returned error but user retrying getting error he might refresh it

So this will not allow M3 to recover itslef

So to handle better way

So circuit breaking implemented in our microservicses



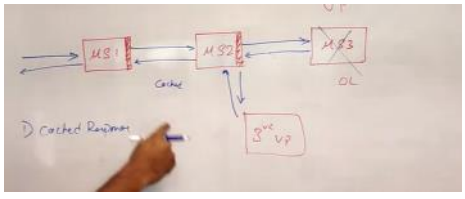
So how circuit braking will heall the microservices3
When request goes one by one then we immediately get to know microservice 3 is not working properly

Circuit breaker will get to know the service which returned just has returned error

Circuit breaker used to rember cache and it used to returned cache response

Cahce response is the success response which we get previous time from that Microservice 3
So we get succesful response which is cached responsed

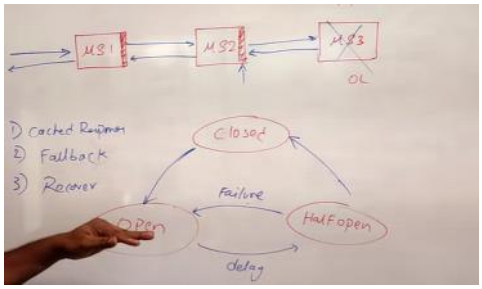
1. Cached response
2. Redirect call to other microservice which is back microservice 3 (Fallback Mechanism)
3. It lets microservice to recover



Third party call when circuit brekaer get to know that m3 is down
We can actually give time till which we don't talk to Microservice

So after 5 minutes time out. Then again circuit breaking will send the request to m3
If it send succesful request then it will start old configuration and remove 3rd party microservice or cahche response

3. It lets microservice to recover



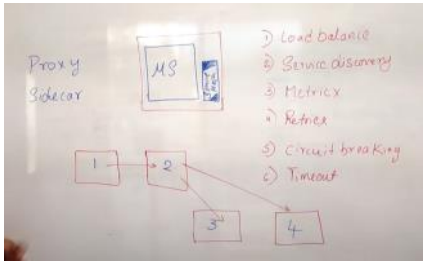
So after 5 minutes time out. Then again circuit breaking will send the request to m3
If it send succesful request then it will start old configuration and remove 3rd party microservice or cahche response

Open is disconnected

Hystix

****

Service messh->



1. There are lots of challenges when one service talking to other service
2. Challenges we facing while communcating microservices

From one microservice to another microservices how do we do load balancing. If there are two instance of microservice then we have to know which one we have to connect

What is the Ip address and port which we need to connect

We also understood service discovery and service client will have all the latest network address of microservices

But we need to query service registry first and findout and select one that's how load balancing work.
For load balancing we need to choose random like random or round robbin

M1 -M2 -M3- M4 which has different instances running on

So same problem everywhere. We need to find service registry of each microservices.

Collecting metrics is also problem

We have to collect what is response time and what is request time
How many are failed how many were passed
In total how many request we are making between service out
Who is going to collect this all information

M1 want service registry for m2
M2 wanr service registry for m3
And so on

Problem 1st Load balancing 2nd Service discovery

Who will do circuit breaking who will let service retry all the microservices

Sophiscated we have to build each request responsse eacha activity

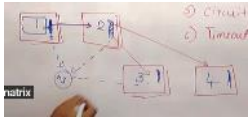
If m1 request something m2 doesn't respond back . So it dosent need we have to keep waiting till m2 closes the connection. We have to give time out if developer forget give time out
Who will make timeout if service doesn't give time in particular time or developer forget to give timeout

I

So same problem occurs in each microservices

So it is advised not to implement everywhere

Service mesh is the solution->
Actually service mesh run along the all microservices or instances which help to do all of these things.
So when we deploy microservice then we have to deploy service mesh also
It runs parellely to all microsrevcies



Every service has one service mesh
Service mesh used to follow proxy design pattern
Its agent or imidator

It's like client access youtube then it goes to proxy server then proxy call youtube.
Blocking for the sites

Service mesh acts as proxy

Whenever m1 talks to m2 so m1 calls to service mesh and service mesh call m2

M1 ->MESH---------->M2->MESH

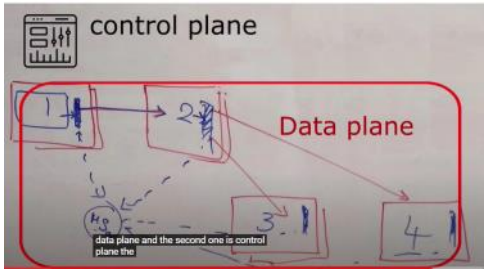
In that way we don't need to do service registry first to identify what is the IP address and host name
We don't need to worry about service registry or load balanceing if there are 100 of instance of m2 are presnet
We don't need to log every metrics when request started when request stopped
Metrics will be calculated by mesh

So all mesh of microservice sends back to mesh server at centralized place.

So actually it solve all the problem it also take care of time out

After time out service automatically closes the connections

Also do service breaking it also do retry



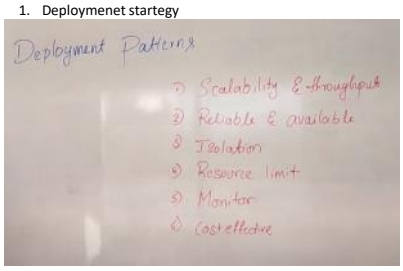
Service mesh has two plane.
1.control plane
2. Data plane
Control plane is the centralized hub or single hub which acts like a control panel which you can actually configure configuration for all of the proxies which are side loaded in all microservice and instances

Data plane is the group of proxy basically
This proxies doent know the existense of other proxy
They work independlty
Proxy actually do load balancing service discovery

WE GET RELIABILITY IN ALL MICROSERVICE INSTANCES

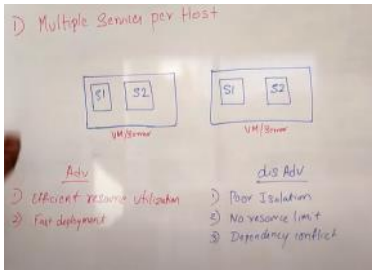
****

1. Deploymenet startegy



1. Server
2. Containers
3. Cloud
4. Virtual machine
   So it can be used by ui

So before deployment we need to set deployment goals



Running multiple service in single host
More running service in single virtual machine

If it is written java then both server can be served by tomcat server
It is traditional

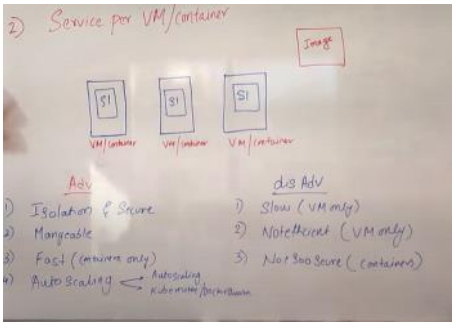
If we want to scale out then add one more virtaul machine and run the same copy of service
Then we need to do load balance

It is utilizing hardware very cost effective. As one service may be serving low traffic but other service may be using that vm effectively

If we start a server then we can deploy differnet services at same time.

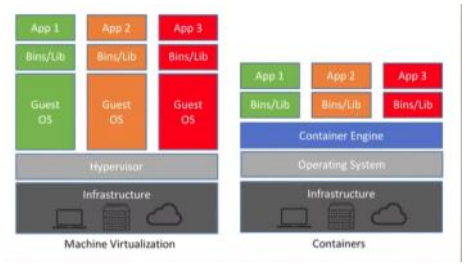
Poor isolation. Both service are on same server. That may effect other server.
Ex1. S1 creating temp file but S2 server might delete those files
Resource consumption. If s1 uses all resource.
So we cannot limit per service

Depency conflict. If both service in java. If one service using same package differnet version and other another. So we have one more isolation layer on top of each service



So if we want to scale then we have to scale one more VM and scale out

So if we want to scale up then we have just to take image and deploy it other vm



So basically we give images and scale it on based on many fatcors

1. No of request coming in
2. CPU

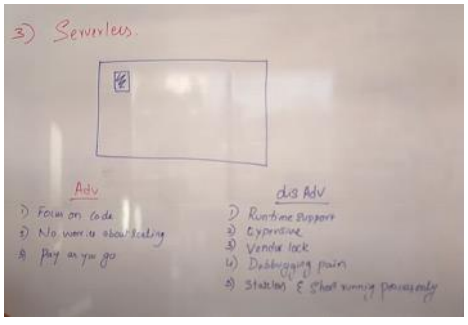
Advatnages

1. Isolation & secure -> We are actually running one instance of machine . That means that service cannot touch other service if it is Running on same server or hypervisor. So it isolated.

2. It is very easy to deploy
3. Containers is fast. Containers is lightweight. Virtual machine is not so fast
4. Autoscaling is fast. We already have image we just spin up to scale up

Disadvnatge
1.Slow(VM) -> As it runs entire operating systems. It takes time to copy. It take to booting up
2.Not efficent(Vm)-> So it consume lots of resource to run operating system

Servless



Login to console-> Paste code or write code . Up and running on cloud.

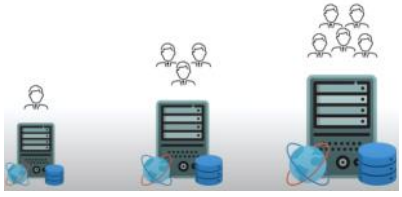
**********MicroService Architecture is over**********

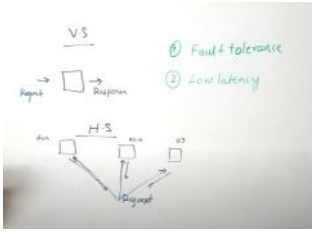
****

Distributed-> A group of computer which works to accomplish the common goals
Example->There is one service which take word and convert into pdf
So we have one server to accomplish the task
So first month-> 1000request /second
So second month->100000 request/second
User experince degarded as it takes more time convert the word to pdf

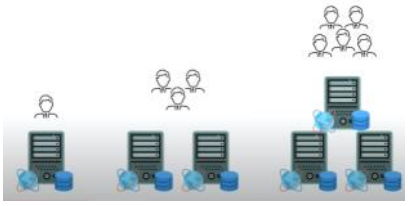
So 3rd month we buy 128 gb ram new processor upgrade the server

Vertical Scaling





Horizontal Scaling



In horizontal scaling if request goes
Down we just need to decommision the service
And sell them off

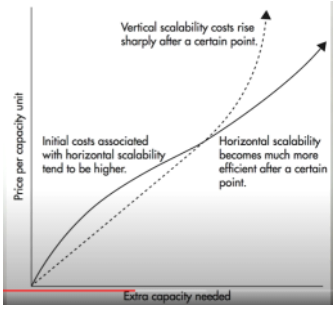
And none of them when he was upgrading the system was down or switched off

1. If any problems occur then distributed system can tolerate that

In horizontal scaling if server fail then it has nothing to serve request
As it has only one machine.

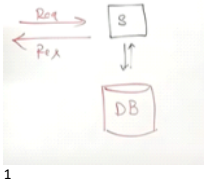
In vertical scaling we keep server in different reqion to process the request
If server down in asia then request will redirected to other server.
Load on server will increase. Still service will not go down
If some machine fail then there are other machine who can take the response
SO it can give nearest distributed system and serve the nearest server

But in one machine so latency and time taken will be more



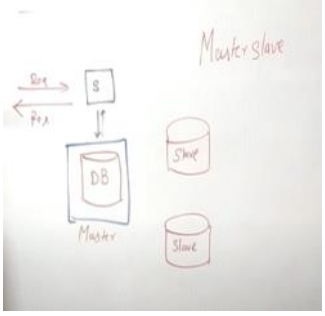
****

Distributed datastores | RDBMS scaling problems | CAP theorem



1Million db request. Then write and read will also become slower. App server will slow.

1
Horizontally it is difficult to divide db
Easy to scale vertical db(Increase memory)
Cache using key value (nosql)
Document based db(When schema is not fixed)
Colum db(Fixed schema but not acid property)mixed rdbms



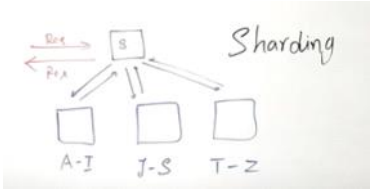
Posiible way to scale server using master & slave
In this we have multiple instance of db

1. Master
2. Slave
3. So any writes will happen that will happen to master. SO any read will be from slave db
4. So any data written on master should be replicated to slaves. Which happen ASYNCHRONUSLY

5. Problem: If user wrote something and read concurrenlty. Then write will happen to master. But it may have not written slave database from where read operation happens. User wont get it. Inconsistency database

CAP theorem
C-> Consistencey

2nd stratgey sharding



1. All the instances of db will have same operation. All are master in themselves
2. Data are divided into segment A-I J-S T-Z. Figuring of key is important. Any name start to A-I in db1 which is called as key
3. In this way we have also scale by read and writes n times.

Problem-> Most Names start with A and I. And les name start with J-P
This will create hotspot and most of data will be present in DB1 heavily loaded

Further how to divide and scale db1 -> A to I-> A to F , G to I-> In this we have to take down database and divide and take dump switch it back

In this cases we have to monitored always which database is loaded. Which is wrong and don't want to do as it is tdious process

We will require one more layer which joins the databases A to I to S TO Z

Consistency-> First machine & second machine. A user writing and reading from database. Both machine can talk to each other.Either read σ write

CAP theorem

1.Consisteny
2.Partition
3.Availability

If write something then next time if we retrive then we should get same value. That is called consisitent
If we get different or old value caleed non consistent.

Table value -1 and write ->2 if we read 2 conssitent else inconsitent

Both machine will have same machine. It should update in both machine or all replica

Availbality-> Even one machine or some machine is down then some machine or instance should be available that is called as avialbility

Partition taulrence-> Considering we have two db. Talking each other. Partiton tell us that. Even two system between not able to connect then als o it should be up and running
Still able to read and write

In reality we cannot have all this property either can have both two combination

Highly available and partition taulrence.

1.In this we cannot have consistent setup.

Example if we do transcation. Even if one system goes down still we can read and write on second database.

Considered User trying to update in database 2 so it will updated in database 1

Even though db1 and db2 is disconnected. So now if he update data db2 if he connects that database then he reads as same valu e

If connected to db1. which is not updated due to failure in communcation between them
It will give old value.

MYSQL->CP system.

Example instagram->Leave consistnce. Availabilty and pratince taulernce
This is era of big data. Ton of post. Billion post. We cannot expect consisitency. We want instagram run all time.

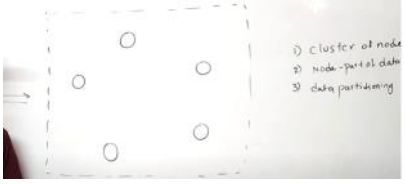
If I post the picture. If my frnd not able to see picture in few second it's ok. We can get consistent over a time.

NOSQL , Cassendra
Eventually consistenty. Slow consistency very slowly.. Partion taulernce high priority

RDMB->ACID compilance

********** **********

How distributed datastore works(basics)?



Couple of machine all is called as node. Cluster of node or collection of node is called as distributed datastore

Each node is responsible for saving or holding part of the database which we want to save in database.
Every machine is treated equally. No master slave concept

Every machine responsibility to saving some part of data
One more important responsibliyt is keeping backup of some other node
Means DB1 has some data backup of DB2 and db3

If DB3 goes down then we have a part of data db3 which is svaed in DB1
So we have backed all data. Even though machine goes down we have data.

So this is called as data partioning. Nosql datastore works in same way.

So each machine can be any part of the world

Incase of failure or scale it up

So if we write data in DB1.
DB2 some backup
DB3 some backup

Distributed is fault tolernce

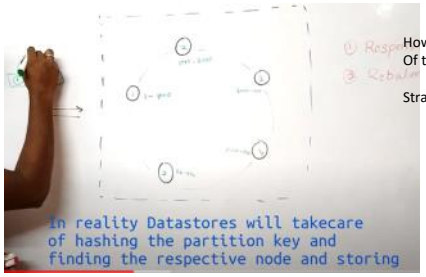
Rebalancing. How nodes should get to know what is my responsibilty or what part of data should I store

Distributed is fault tolernce

Rebalancing. How nodes should get to know what is my responsibilty or what part of data should I store of others.
Consistent hashing plays important role in this.
Partition key-> Hash ranging from 1 to 1000 will be keeping data whose hash become DB3

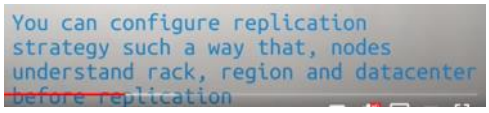
So I want to save some row.
Take the data and partiton key and hash to it and put into db as backup

In casendra there is replication factor which telss that how much time data should be replicated



How database know if we store CAT in db1 then who or which db will take backup
Of that data

Strategy
Next available in clockwise->
Or Next two node
Combination of hashing and backiup
Which make resilant and fault talrence



Rebalancing-> If we want to scale others. This 5 cluster can handle upto almost handle 1 million data

So we get to know sesson will increase data.

We add new node and give range to them to store then it again will reshuffle all its hashing or key or range

So they reshuffle automatically the range. How does they lean about it

Gossip protocol helps to reshuffle the range of datastore when new datastore is added.

Each node continously talking to know status of each node. That's' how each node know about primary responsibility and secondary responsibility. Evry cluster know about other cluster responsibility

All node talking all nodes. So they are up to dated.
So as soon as we add cluster will know that cluster is added they reshuffle.

So data loaded to that range is also transfer to new node.

Some machine goes down. There responsibility to store data from 5000 to 10000. Other will node get to know that this db is no more presnet in cluster. So it reshuffle all the value between them.

Reddis also uses distrubted store. HDFS also which keep backup .
So it break files intho 64 mb and store into different machine acrross
Piece of file are stored in all nodes as backup. (They use master). Single point failure of master. Zookeper should be there.. To detect mastre is up and running if master fails then it should give responsibilty to others.
(Cassendra check)

****

Distributed Locks | System design basics

Mutex
Semaphore
Locks

A lock allows only one thread to enter the part that's locked and the lock is not shared with any other processes.
A mutex is the same as a lock but it can be system wide (shared by multiple processes).
A semaphore does the same as a mutex but allows x number of threads to enter, this can be used for example to limit the number of cpu, io or ram intensive tasks running at the same time.

It is used to lock critical section
Semaphores to count how many connections are opend on particular tab in chrome.

Mutex->Multiple process are running on machine and all trying to log in single file
That's where we need to lock read and write file.

Multiple server working for same purpose for the same website.

I have a website I cant just serve information for the same website from the same a single.
If more people coming to my server then I have to serve more number of data.

Request can be serve from any of the machine.

Why we will need distributed locking system

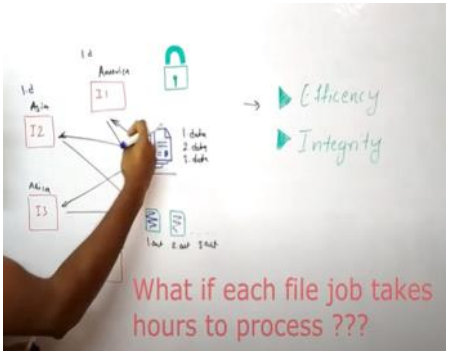
4 instance is availble all are in different part of world.one Logs file read the file and figure out some information

Write back to another place.

3 input files 1.data 2.data 3.data
3 Output file 1.out 2.out 3.out

Now 2 instacnce on same file they both write in same file.
It is uncessary that two system working on same file. We don't want this. We want to make it efficent

It should be one machine should pick one file and process it



Integrity -> Corruptness of the file. In this two instance are working on same file it may corrupt the output file

Only solution for this is to lock file as soon as any insatnce start using that file.

So other insatnce will pick other files.

It is single point failure lock. If lock server fails then it will delete all the records

What if a file take one day to process the file. Like traing machine learning models

We don't want lock if single point failure. We don't want to use db as there will be lots of read and write so this will not be efficent

We need to distribute lock system as well more machine working for that purpose
Which replicate same data asynchronuslly.

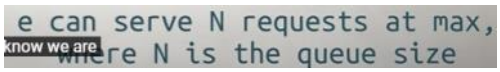
No lock system is also distributed it will also take lots of time sync the data

Properties of lock

1. Mutual Exclusion-> We don't want two or more process to acquire same log
2. Dead lock free
3. Fault tolernce

Leaky Bucket

1 request per second
If queue is empy it sit in bucket. Extra request will overflow



Last time user used that service and avaliable tokens 5. It stored in user id

1. Fetch Token-> Fetch token for that user
2. Update token-> Once it access the token then it should update the token
   We need to reset token after every 5 minutes. If exceed drop or reject request

There can be race condition if same user request from parallel

****

Rate limiting-> 10 times the normal traffic which is from the bots
If we want to give api which is 10 api per minutes 100 api fer hours.

Both problem solution is Rate Limiting. By providing protections by overuse of Api. How user can access the API when the user request quota is finished
Either by dropp or rejecting the request

1. It is important for user experince. Some user overusing api so other user experince will get affected. So we need to rate thelimit of api
2. People might use brute forc or promocode so we need to protect it from them
3. As application auto scaled then it might rise the cost. May be user want to use api for fun and bombarding the requset

User based limit .. That how many given user you are going to allow per minute.

Concurrent-> Given user how many parrallel session or parrelel connection allowed to reduce ddos attack

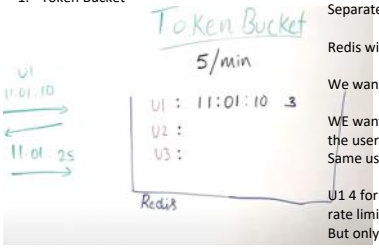
Location or Ip address-> Running event to given location. In that all other location should be rate limited and that location should not be rate limited

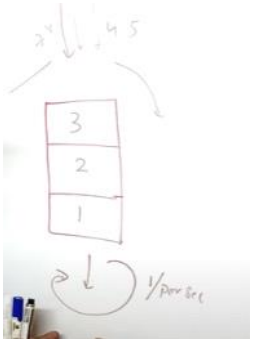
Server->Defined server for certaion kind of service rate limiting on that service

Alogorithim to rate limit

1. Token Bucket

4.





User-> Having distributed system which has load balancer and rate limit

Separate for region 1 and region 2 rate limit for differnet region

Redis will hold the keypair data and for rate limiting for user

We want to enforce global rate kimit

WE want to store local rate limit but whole system we need to control rate limit we don't which region the user using the rate limit
Same user from different regions both request will given load balncer and rate limiter

U1 4 for both rate limiter replicated in rate limiter. If two request from different region and get latest rate limit now 4 request is serveed.
But only 5 request we need to server there may inconstency of data.

WE can use this using sticky session. By redirecting the user to particular server..or same region.
It is not well balnced or fault tolerated. As all request to particular server then it might get down

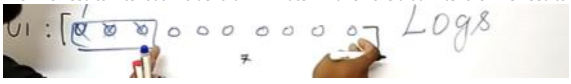
We can use locks.. Race conditon by using lock.
If user tries to request then load balancer put lock on particular on rate limit till when one rl is uing no other rate limiter should have worked till other rate lock is unlcoked

There is always latecny as when we sync data

Fixed window counter algorithim
In this we have counter we keep counting at every request.
When counter exceed rate limit we will drop the request

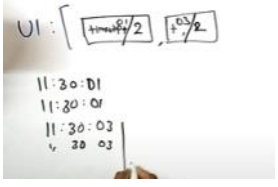
10Request/ Minute
59 sec 10 requet come there traffic is not smottheend and next sec 10 request more come.
That is bad

Sliding logs-> Key on user id -> Then we will take big list. We keep to adding entries to big array. Each enrty also contaion tiemstamp. At which request come.
Then we need to filter out in older than 1 minute . And remove that. After then we need to count the request.



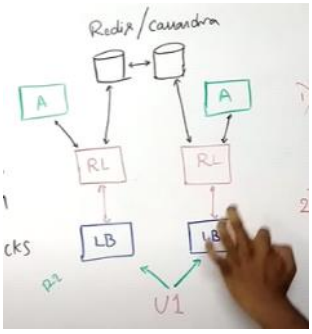
If we have 11 entries then we need. We are using more memory

Sliding window counter-> Ui entries->If we get same entry at same second then we keep counter 2 and
Then if at other second then we keep time stamp. If counter exceed we can remove. Each entries now we have less entries



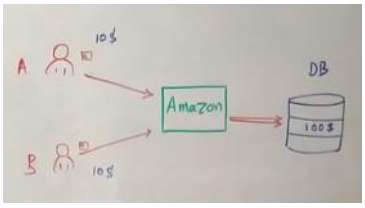
This algortihtim works better in single server setup. But when we have distributed server then it will not work fine

Inconsistency -> In rate limiting the data
Race condiiton->



****
*

Transactional locks

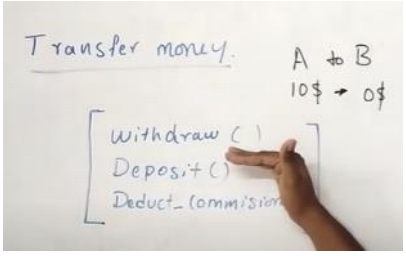


If two person spend 10 dollar from same account then balance shpuld be 80 dollars.
If balance is 90 dollar then there is bug

This is happening due to transaction. Even though this transaction update same record in db.
A request to deduct from db then that particular row wil be locked.

B will not allowed to touch db while A is working. Either one will be picked first.

Transaction is unit work which either happen full or nothing.



A has 10 dollar
 B has 0 dollar

Withdraw()
Deposit()
Deduct_commision()

If we were not using transaction. 5 dollar A to B withdraw from account a and transfer to B deduct commision. Defintely update in A

If no paralle is not happening

What happen error / or server failure-> We withdraw from A deposit in B some function give error. If 5 dollar is not deposited to anywhere then 5 dollar will be gone.

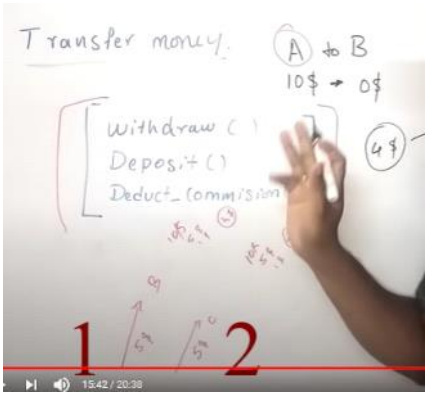
If we say db that all these steps covered with transaction. Everything execute perfectlty otherwise scrapped

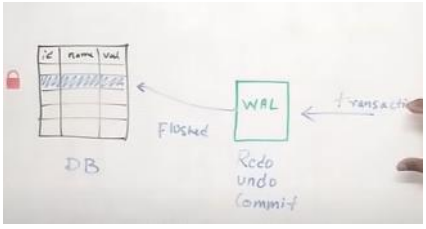
If anything error rollback if everything occurs fine we will do commit.

Concurrency-> Same thing happening at same point of time

Two time A give to B. Withdraw function will be executed. Both operation see all have same balance.

So transaction should be isolated or atomic. So still one thread will access account A





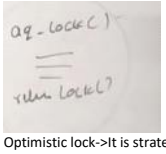
Before updating to db a lock will be aquire to db

There will be policy that no one will update while one transaction is going on or but it can allow to read

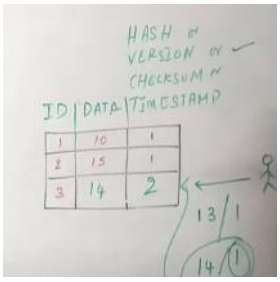
WAL Is inmemory database

****

Pessimistic lock-> Usual lock-> First db is locked then we do opeartion then db is release. Then this will be used by other method or thread is called pessimetic lock
Databases uses cases take pessimistic lock



Optimistic lock->It is strategy in which we read a record and also record the timestamp hash version or checksum



A and B both want to modify the data

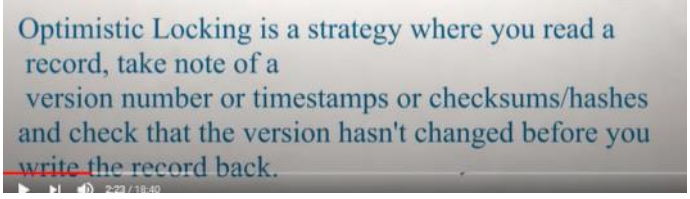
Then B take the data id= 3 just incrememnt the data 13 to 14 then version will be updated from 1 to 2

In next case A and B both take Id 3 at same time so A is fast increment 14 by 3 and while updating in db just see the version
If it is 2 then he will update and increment the version by 1
So when B going to increment 2 on 14 and he check version which is 3 so he will not update because data might get corrupted

So now he has to recompute the things. Then he will take 17 / 3 and do computation

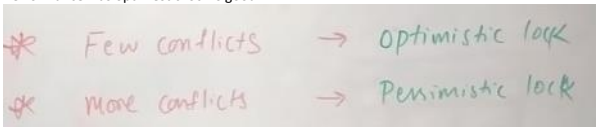
It is optimistic lock it was ok if some people was updating the lock.

If wikipedia or stackoverflow then we use optimistic lock.



In pessimestic lock.. Until I finished the work no one is else need to use this.
In optimesti lock every wone can use at a time. Just need to be updated

Performance was optimestic lock is good



In this type of lock there should be time out to avoid deadlock. It might be A user waiting for B user

****

How Google searches one document among Billions of documents quickly?

They crawl billion of web pages

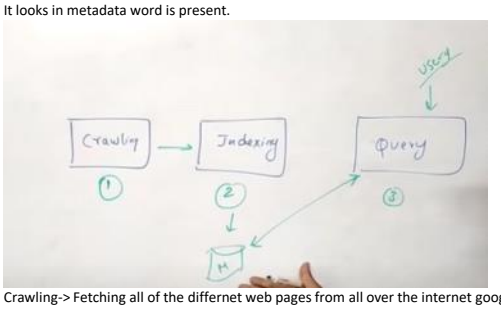
1. Low latency -> Search should be very fast
2. High Throughput-> System should give lots of query at any given point of time even though the thougsoand or billlion users are searching some queries in that sysetm

If we want to search the word then primitive computer will search iteartive it will go to all files and search it
It has to open each file and then search each line and then close. It take lots of time.

Indexing is preprocessing helps to identify your resource. Arrays and dictionary
In indexing we can find in 0(1)
They used to use B tree

If computer is not in use they use preprocessing they used to do index and keep metadata so that when we searhc anything then it don't need to go each file open and close to find That word

It looks in metadata word is present.

1. 

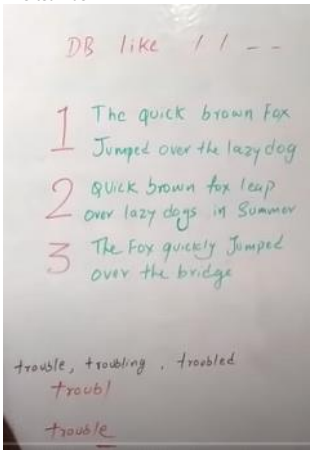
2. Crawling-> Fetching all of the differnet web pages from all over the internet google has distributed crawler called spider. It is kind of distributed means
That so many computers are basically crawling over different sites using links available in page.

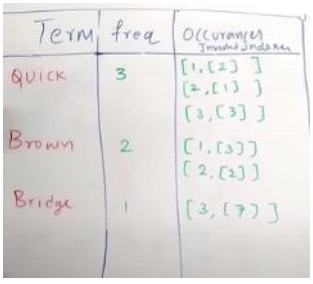
3. Indexing-> All these file fed into indexer . Indexing process all over the pages
Store metadata in db

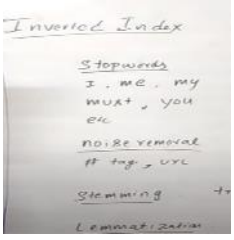
4. Then user can do query and value given from metdata. And it give data.

If we were creating with db then we would have make two coloumn one id and one string . Then we have to use like keyword to search
So it will not process fast in db

Inverted Index







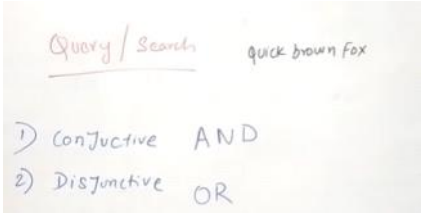
Stemming is process to removing suffixes exampl quick quicks quickly s and ly

Lemmatization -> It give actual keyword as it know all tense and ing
Like trouble and troubling

HTML tags url hastags are noise removal

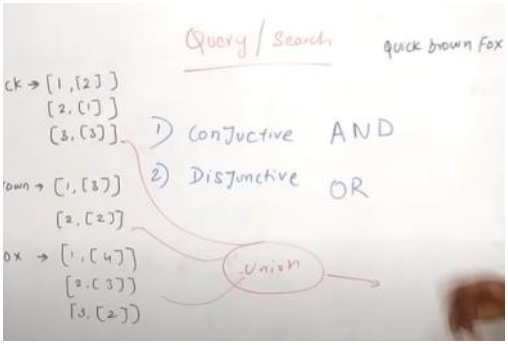
Then document is clean

Metdata is created above for quick brown fox



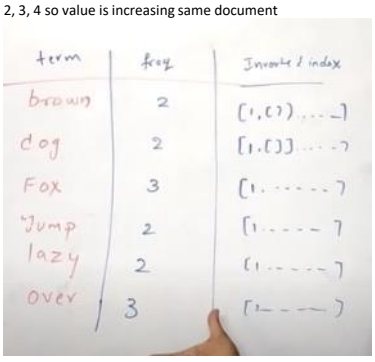
First I am intrested which document have all the word. Later any of word present
Page rank do on google

Order of word is also important.



Same ordering as well.. Conjunctive query. In increasing order

2, 3, 4 so value is increasing same document



Search quick . So we will give all the coloumn which has word quick.
Search jump.

Instead of this we can use bit represnetation it will save data and it will also fetch data fast

If we want to search prefix like jum* then we need to keep sort the term and then we can do binary search on it find it quickly

System design basics: Real-time data processing

Sream processing-> A real data is coming we need to process it to create meaningful business requirment.
Moniotring system. How many succesful request and response and how many 404 or 500 error as such.
We cannot do in one machine. We go for stream processing.

We have 100 file in other server.
We need one system who need to keeps on reading and push it to one particular data structure.
We need to some time data queue or messaging queueing system or let's think it is kafka

Kafka is queue districbuted and reliable.

From where our stream processor will read our msg. Rate at which it is pushing 1000 line log.

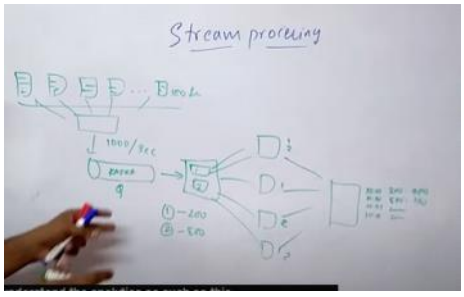
With a single machine it is very diffcult to process 1000 message because we need to parse every line
It will diffciult

We need to have so many server. Again we need master or driver where it recive message. It knows what are all the jobs we n eed to execute basically. We know how many job we have to execute example job1 , job2
Job1 say we need to count 200 so it send to particular task executor computer which will do that job
Job2 say we need to count 500

Driver keeps on reciving message and keeps on pushing it. Executor process it.

After process we need to write in one centralized place or database
10 pm 20 sec we saw 100 request 200 500

Every second log will be there and there will be analytics going on

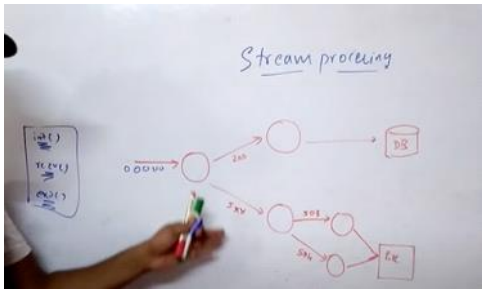


Spark does microbatching in master.
It waits for 2 second and collect all data and send for further process.
So 1000 files are converted in to one type of file RDD -> resillent distributed data
It has relaibility it can be recover it keep duplicate in different machnine

May rate increase to 4000 per second just we need to add more machine. Driver will take task to sending the message to machin e

Alerting system
Real time monitoring

So data push on queue we need to see 200 and 5**
If 200 request then we need to write in db
If 5** then we need to process 500 or 503



All are queue.
We need to do partition in queue. Stream of data will be divided into more queue so there is no duplication

****

Distributed system

Multiple system are connected via network high level problems.
This whole system looks like one system

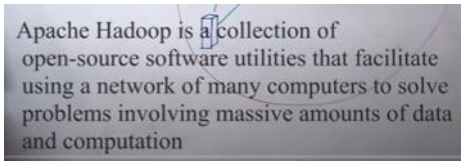
If nasa release any picture it is in petabyte. We cannot save in one machine.

So we divide the data in samll chunks and giving the machine small piece to each one

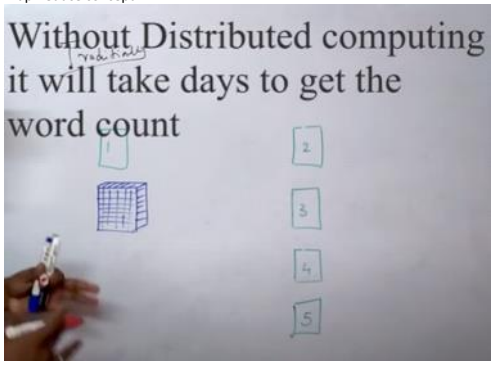
Collect and process it in one pc.

If we want to count words of books.

Then we divide books in batches and give count to each of server then we process and take the value



Map Reduce concept->



If we have millions of book and we need to find the word count

I have 5 machine. Then we need to split 5 million in 1million chunks.
Then we would send chunk to different machine

So we have code for word count in each machine to compute all of this.

I have result all in machine. Then we need to collect the result in machine.

I need to write one more code to merge the number of each result. So lots of manual work

What if there have some problem. One machine failed. Obivously cannot have final result

We need to get know when that machine is failed when we will go to collect result.

If that machine then ideally that machine data will be lost. So we need to store data somewhere

Relaibility problem

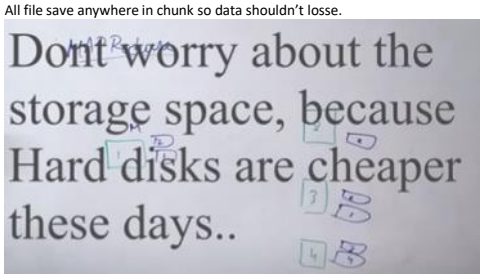
What if one machine is faster and one is slow?
Then machine will take more time to compute. So we need to wait for that machine to get completed. Even it take 10 hours to complete and all other machine is finished there work.
It may be very tidious for 100 machine to keep track of all machine

MAP REDUCE-> MaP REDUCE DOENT ALLOW TO SAVE DATA AT ONE PLACE. So it automatically in different computer

Master node is like a leader node. Which will take care to distribute data

Automaticallly save chunks of data in all server. So it also keep multiple of data redundant.

All file save anywhere in chunk so data shouldn't losse.



Even if one machine fail then we have chunks of data.

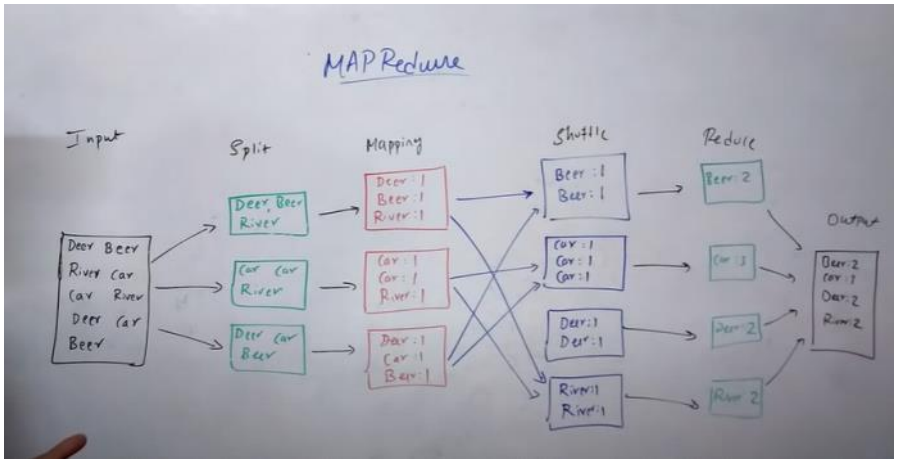
We simply go to master node and submit the code . Automatically copy the code and transfer to all machine

Master know how many machine is in cluster.
It all knows who has how many power. It has power to give equal data to there capacity
We need to worry which machine is faster or slower

Even one machine fail the job distibuted to other machine.

We need to colect the result. Master will collect the result. Then it will aggreagte the result

Manual task is removed which is done by master.



Executing big problem in parallel in distributed.
Framweork will split in chunks..

****

System design basics: Learn about Distributed file systems

NTFS-> It is way the data store in storage memory block to retrive data faster and efficently

It knows the sector where file is stored. Some file system used to compress the data.

Why we need?

I have cpu 500 gb. If file is critical we don't want to loose it.
We have to buy one more machine.
We will divide and put in different machine with replica. So that even if one fail there should be data

How do we know where is my file backed up. Where is my file in. When I want whole file

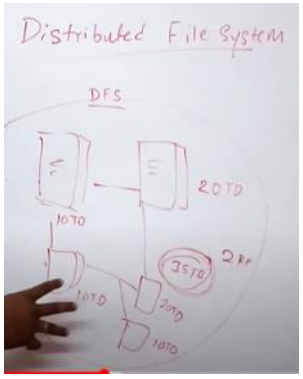
NTFS Fat were design to handle only one machine data.

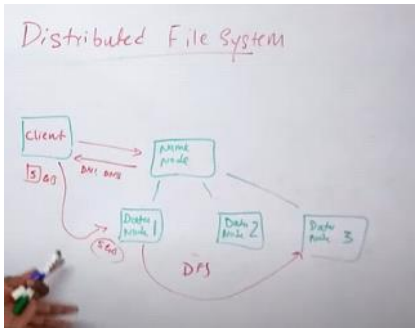
In cluster we 3 machine. All are interconnected.
How do we store 30 tb file. So distributred system will automatcally break into chunks of data

And automatically filling wherver the space is available.

When I query for file then it automatically take all the data and combine and give us the whole data

I can also provide replication factor. If enough space is there then more copies are stored in case of failure of one server it doesnot affect much.
Please replicate in 2 copies of 35 tb .
Distributed file system is clever. So we get all file even if server fails.





Name node-> Doent store data. It acts like master.
It knows the address where file location is.
It actually check the health of data nodes.

It also take care when machine fail then move data in other server.

And copy the file as per the replication factors
Configure for 64mb. So if any file uploaded it will brken down to 64 mb and put across the data node

Client library to access the data which is used for read and write from namenode

Name node gives which node is free. And please go ahead and write on data node 1. And also tell the node 1 to replicate on no de 3

So one node goes down then it tell data node to replicate the data which was on server which is down. To maintain replication factor

There is rack in same location two server then data not replicated in same rack

Distibuted file system also know it shouldn't put on same rack or same location

Wherever we want to copy safely redudncy.

Namenode also take care of health and replication

****

System design basics: What is asynchronous processing?

From <https://www.youtube.com/watch?v=BFcNDPt6SIE&list=PLkQkbY7JNJuAhePp7E_WSpfFqjQp6RniV&index=11>

Synchronus

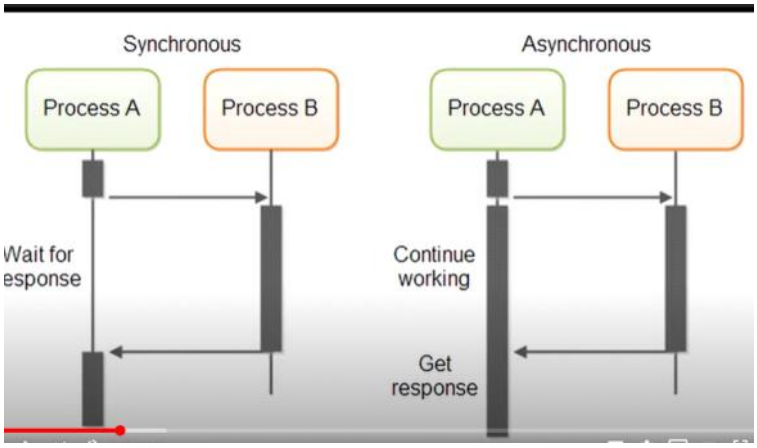
Code1
|
Testapi()(Takes 20 sec) If it is synchronus then it will block thread for 20 sec
|
Code2()-> This will processed after 30 sec only

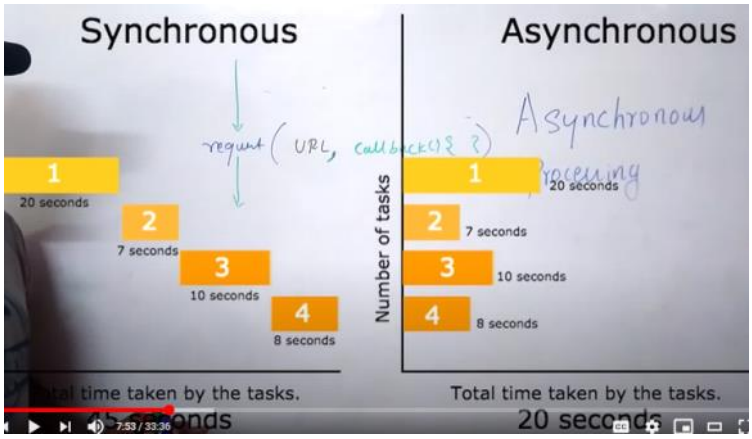
File IO Input IO->TestApi() calling google.com then it will go to other repsonse. Thread will wait till we get response

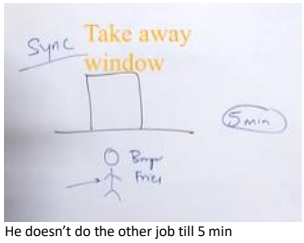
If my code dosent depend on testapi() then we neednot to wait..

ASynchronus

Code1
|
Testapi() Thread execute it and again do further process
|
Code2()





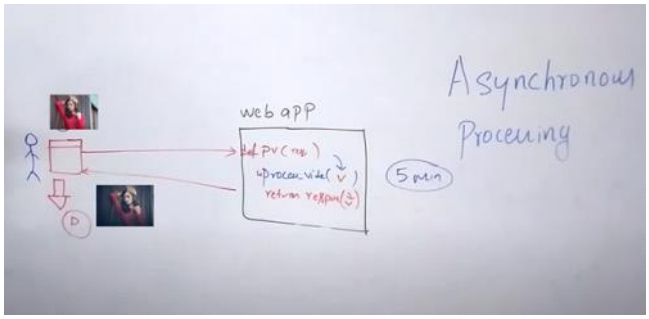


He doesn't do the other job till 5 min
AFTER taking the burger he took the banana and apples. 5 minutes
Total 10 minutes

In asynchronusly he will not wait after giving order.
May burger guy send msg that collect your burger as notification(May be queue)

IO call cpu till response come.

Send video to me. On web app.. Call process video.



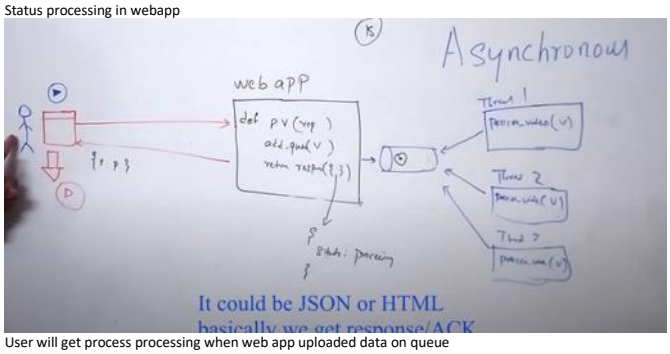
User has to wait.. What if user close web
He has to wait.. What if processing taking 15 minutes. User has to wait 15 minutes

If he waited 14 minutes at end minute power gone or something happened.
Then everything is lost

User has submit and do whetever he wants to do. And the video should be there in his account aafter processed

Thread will be waiting.. Message in queue. Move all function of web app to thread

Web app will use request handler. Add to the queue and return the response.
Status processing in webapp

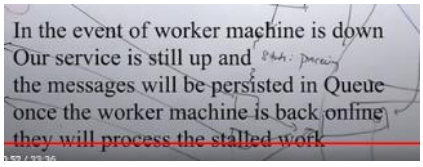


User will get process processing when web app uploaded data on queue

Now user can go and do his work.
Thread will wait for messgae. And when message come thread will taken by one thread. This thread will be blocked for 15 minut es.
Now some more thread is free. Max 3 people can upload.. If more people uploaded then it will be on queue when one threaad wil l get free then they will process it.

If thread done processing either he should put in db or store somehwre and update db that status is completed.
Or send sms or on mail. We can put gpu in thread

Now we can handle more people.



Same in facebook and youtube same pattern. They process and find refrnce from queue and download it

Data corruption and Merkle trees

From <https://www.youtube.com/watch?v=rsx1nt2bxf8&list=PLkQkbY7JNJuAhePp7E_WSpfFqjQp6RniV&index=13>

There is one big file we need to copy into different servers efficently.
This should be ensure that files are not corrupted and intact.

If files get corrupted then we need to find and correct it
We should less time to solve the problem.

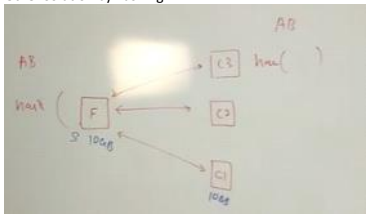
Bitorent cassandra dropbox merkel tree is used

File get corrupted while downloading and uploading.

1. Either problem with software hardware
2. Intenitally someone modified file
3. Virus and malware modifed file

Some algorithim to use solve corrupt problem.

We can check two file by character by character.
Other solution by hashing.



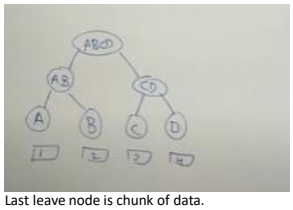
If we get hash of f and get hash of c3 which is replicated. Calculate the hash if both macthes then it is ok otherwise corrup ted

If one bit get corrected then also hashing matching fail we have to download whole file to compare.

Merkel tree-> First we need file and breaks this file into chunks.
Smaller the chunk more data.
Large the chunk less data

Each chunk has same size.
We need to take hash of all chunk. Then we will take from leave nodes.

From bottom we need to combine to make parent add the hash for both leaves make parent.



Last leave node is chunk of data.

Now we copy the file.. To server. We also want to know which part of file is corrupted

We need to download that chunk which is corrupted not whole. So it make efficent
(log n)

We need to make merkel in client side also. Same thing which we do at our side.

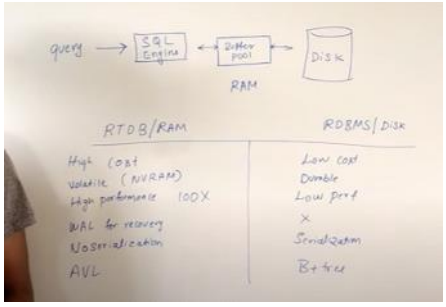
We check which portion get corrupted.

Compare node by node to get to know file is corrupted where. If we find error then copy the file which is corrupted.

Same will be done in bit torrent.

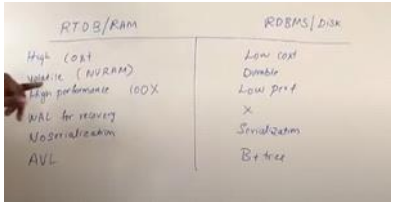
In Memory databases internals for system design interviews

From <https://www.youtube.com/watch?v=zkACt4NYkU4&list=PLkQkbY7JNJuAhePp7E_WSpfFqjQp6RniV&index=14>



Bufferpool of ram to make much faster
Frequenctly or cache put in bufferpool
If write having then both place need to be updated

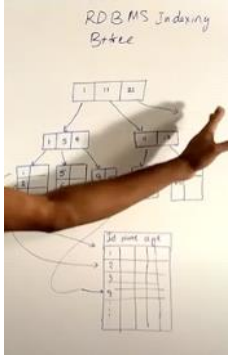
If we query read write on ram database then 100 times gain faster
Wal file to store in disc



Ram do index on range query

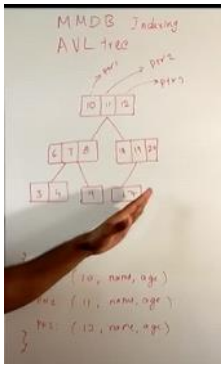
RDMS do indexing .

Real time and DB indexing-> Enabling the indexing query faster.



1 to 11 in one table
12 to 25 to one db

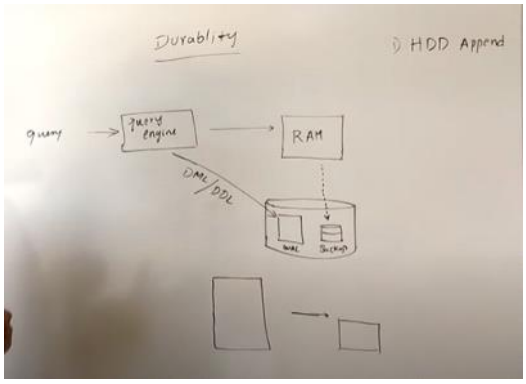
Offset 9 will we go to table row 9



Avl rule greater side and smaller size. Height diff 1

Soretd order log(n)

We cannot use hashing becaue for range query we need indexing query



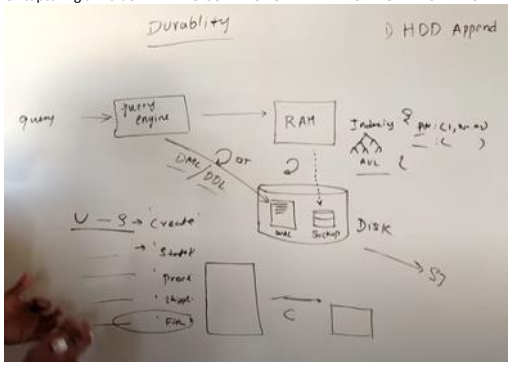
Query to make faster. Write ahead log wal file. Append is fatser in hardsisk.

Query engine has to do two thing1

1. It has to write data into data structer example indexing data sturcture. Or hashtable
2. It write the query ddl and dml into file wal file. Write ahead log. All the query update and creation sent into wal.
3. We also need to back up of db in wal. All db should be taken snapshot and stored in db.

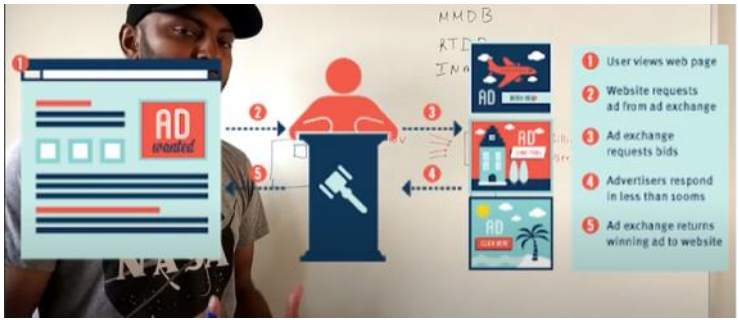
If ram restarted then replay all the query from beginning in wal file. So it will reproduce all the data.

Recording the row or updating the status. So we need to make to do something to make compact. Instead
Of capturing all value. IF WE DO COMAPCTION WE WILL CAPTURE LAST VALUE AND DELETE PREVIOUS FILE.



High avialibility
Backup and latest value.

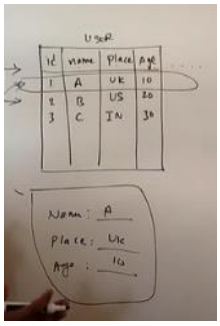
If 1 tb 2 tb then we have to go for sharding



****

Row or coloum orinted database

User management system

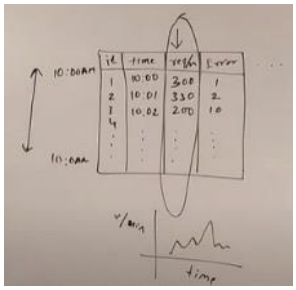


We query using id and find all the data. After changing the row we again put back to table which is called transactional workload.

In table where there is lots of data where we fetch one or more coloumn.
With any given duration.
If I want to see all the trafic for past 24 hours.

Here we are accessing using row wise

All infromation are stored together If we want to get all coloumn then it will take 0(n)



Here we access coloumn wise.

Give me all the coloumn from this duratin to this duration

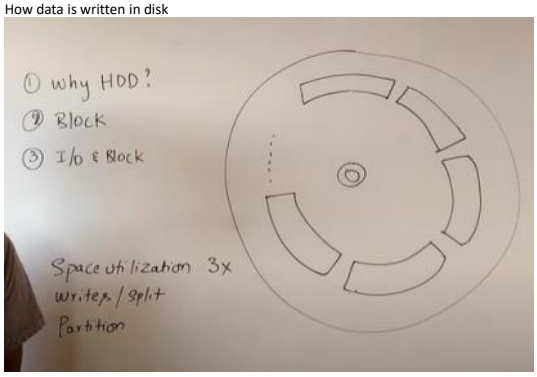
How to save that so that efficently take out the value.

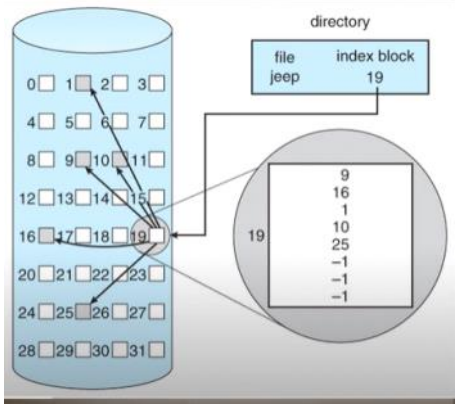
Here data stored by column wise ex

300 330 200

We are fetching only low.

How data is written in disk

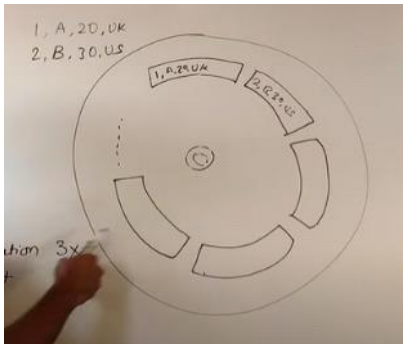




We take block so that system distrubte data efficently to harddisk.

Data stored and read in sequential manner which is better.

In case of row orinted.
1, A, 20, UK
2, B, 30, US



So example in write ahead log.. They keep data storing in last. But in this data stored in sequntial so it is difficult to re tive and go back

If we have to find id then we have to read whole block so more io operation. And more data is loaded on ram and discarding th e data.
It is good for transancational but not for reading coloumn as it will read all data.
If datatype are different there are hetregenious so we cannot use data compression to compress data
So if any new record.. It take all and put in sectore.

In column db it take one coloumn and store data in sequential order.
Space utilization is also there.

Even if new data in coloumn then we can fit into that sector.

If all data in one coloumn then we can use specific coloumn compression to store data

If coloumn then it will take 3 io operation and if row operation then it will take 1 io operation.

In partition one of column data stored in one machine other coloumn in other machine

EFFICIENT COUNTING USING BITMAPS FOR SYSTEM DESIGN

From <https://www.youtube.com/watch?v=8ZgRW0DNus4&list=PLkQkbY7JNJuAhePp7E_WSpfFqjQp6RniV&index=17>

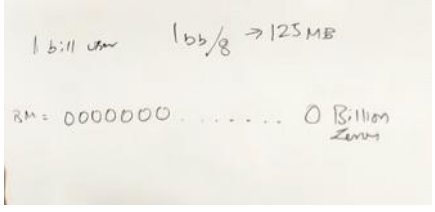
Daily active user and daily inactive user 1 billion user.
I want to store in efficent memory. High performant.

1 billion user the key of user id is integer then it take 4 byte then 1 billion will take 4gb memory just to save key if in h ashtable

Then we can use bit map.

Bitmap is just array of bits.. 7 bits. This datastructure can used to solve this problem.

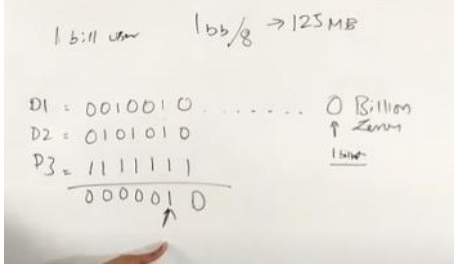
If we want to get active user
Then we will take exactly 9 bits equal to 1 billion



Each billion map to user.
If n th user login then n bit should be on.
Count the all bit of one that will be active user

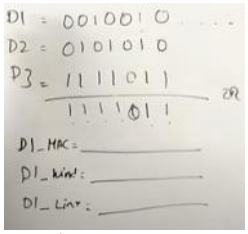
Count all bit of zero to get inactive user.

If we want to get last 10 day user active then we just need to save in memory

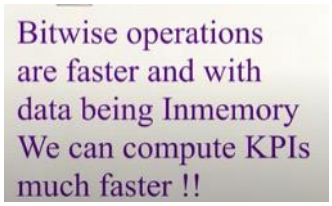


Give user who is active any day.then do the or.

If we want to calculate the mac os and phone. Then each will have bitmap;



By using bitmap save memory

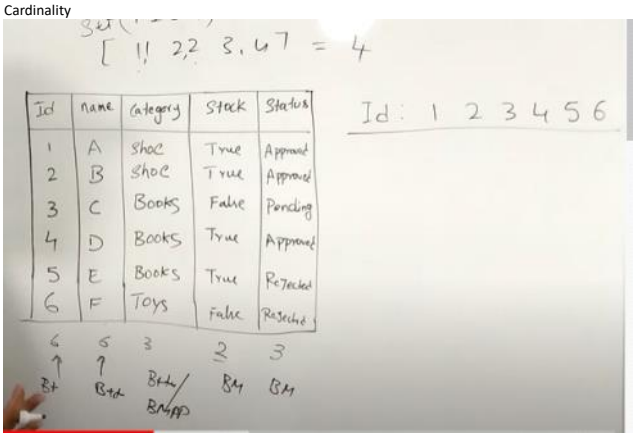


LEARN BITMAP INDEXES

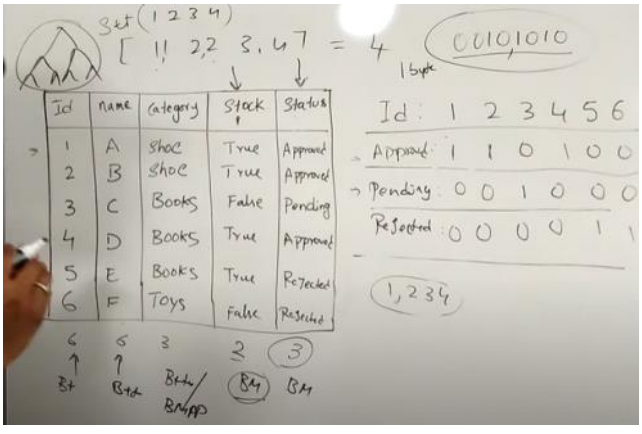
From <https://www.youtube.com/watch?v=5-JYVeM3IQg&list=PLkQkbY7JNJuAhePp7E_WSpfFqjQp6RniV&index=18>

Better than avl tree and b+ tree

Cardinality



Wherever there is high cardinality it is not advised to use bitmap.
Wherever there is low cardinality it is advised to use bitmap indexing.



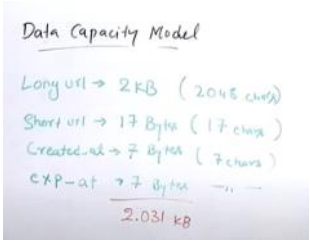
To get all approved

****

Url shortner: What is length ? What is voulme of traffic?

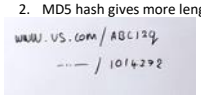
Url shortner: Data capacity model

Long url: www.india.com/7character



Two algorithim we can use to shorten

1. B62
2. MD5 hash gives more lengthy characters-> Only first 7 character/ There can be collision



Base 62 A-Z 26
a-z 26
0-9 10
62^7-> 3.5 trillion

Base 10 0-9

10^7 million combination

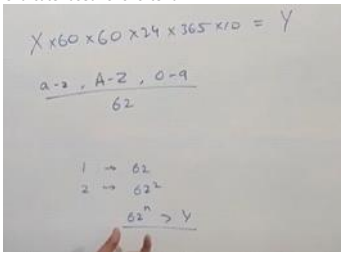
We can use rdbms/ nosql

Tehcnique base 64 for 7 characters
Long url -> To short url insert into db
It can be that short url might already present we have to ensure it is not used by db
Check in db if it is present or not. It can work in single server. Multiple app server

Very low latency
Very high availbility

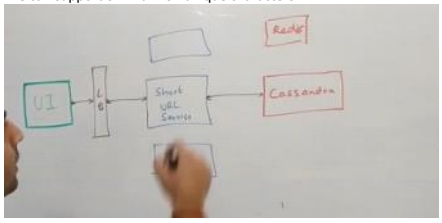
What will traffic?
How many unique url will come?
Save till next 10 years

X number of request comes every second
Charcater used for shorter url



We have to come to one number basically which need to be greater than y

We can support 62^7 trillion unique characters



Ui takes long url
Then it will call short url service
Store in database and return short url

If there are muliple server so it short url service can generate 111 on all service when it is using load balancer
So we end with the collision

If two service generate same short url then we have problem it called colliosion

So we cannot have two same short url for two long url.

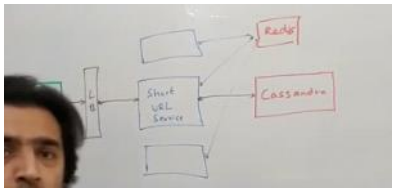
Possible solution can be we can check in database then provide the short url
Then it is not efficent.

So we need to do by way that no collision will be there

Reddis cluster-> Reddis make sure that it should return unique result. So every time just increment the number and return the response.

There is problem everyone is connecting to reddis. So we have reddis huge ammout of load

It can be single point failure.



One reddis doent scale to latency requirment. Then it will start choking the whole system
Now multiple redis.
So machine talk to one reddis. And other to other.

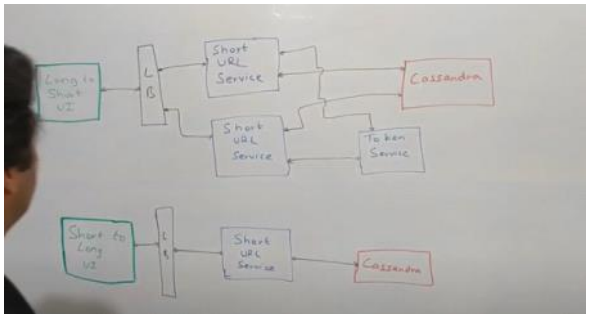
So if both reddis start with same number then again it is problem.

Somehow we make sure that reddis doent produce same number

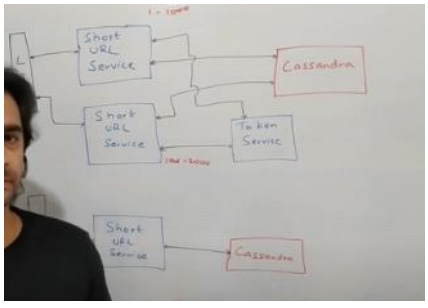
Give series to other reddis.

What if we need to add third reddis then it will difficult to add series. Now we need somebody to manage what series given to whom.

So generating unique url's withouth using reddis.



Now we will intoduced token service.
So we need to make sure noone of the machine generate same number. One of the simplest ways to that ranges to the service.



Mysql Transaction get a record and get first assigned token range and return that
So it run on mysql then it make sure that it will be unique

Massive traffic-> So will allocate million of range to request
So it will distributed at different data center so it doenst become single point failure

Thers is no track of recor of range. If any service goes down or shut down then it might be that Token service get wasted so we need to keep record

When short url hityou get request in short url you hit dtaabasase and give long url

Why we have use cassandra. Handlling 3.5t so other database will give problem we need to shard but cassendra will work fine.
Mysql with enough sharding

So this design will not give which is top url used what type of geographical url is coming.
We can use ip address which database it is coming then it will be put on kafka. It will increase latency

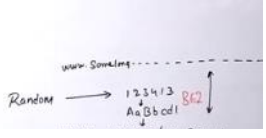
WWW.US.COM/ABC12Q-> This abc12q locate the long url in db

B62
MD5 Hash-> Give long url and it gives short url. There can be lot of collision data corruption.
Precuationary check already present in db.

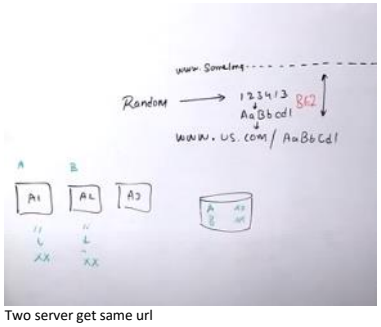
Hard to scale rdbms because lot of read write. Sharding will be hard

Nosql-> If we write something it take time to replicate on node.
High avialble and easly scalble

Reddis is in memory database very fast

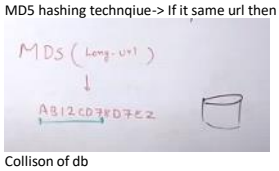


Reddis is in memory database very fast



Two server get same url
DB level protection
But it is not be avialble to rdbms

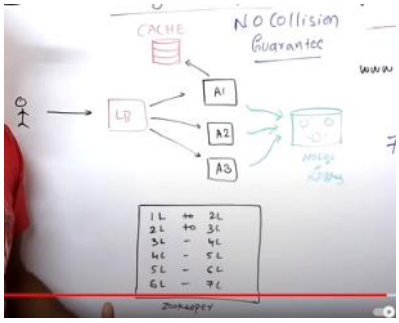
MD5 hashing technqiue-> If it same url then it will always give same hashcode



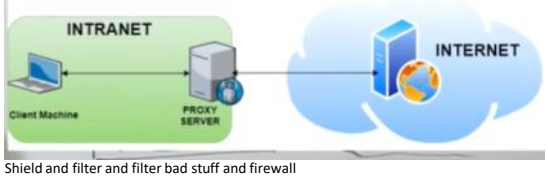
Collison of db

Counter-> Thread protected

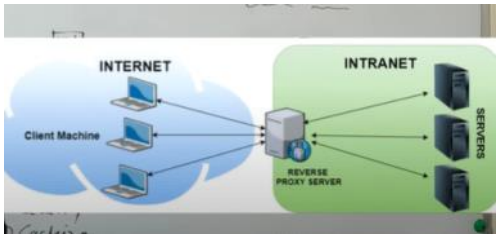
Zookeeper... Multiple counters..Manged distributed server. Cordination service
Zookeper is class teahcer keeeps of record who are the student. If student join the class it knows. It give unique id.
They also keep who is absent in class or present class. Teacher also elect who is leader.
Cordination service



Proxy->Child ask parent icecream parent go outside bring icecream
Security client



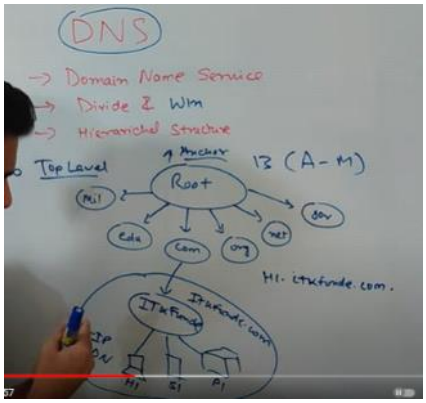
Shield and filter and filter bad stuff and firewall
Better management lot of client can have one proxy server
Caching->
Encruption and decrytption
Ip address encryption



Protect the server
Instead exposing put reverse proxy
Reverse proxxy and can also load balance the website
Caching-> same page
Ip address of server masked

Rverse proxy can be load balancer

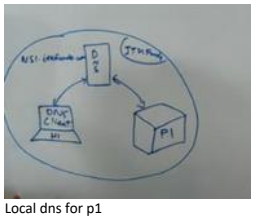
DNS-> Every computer is remember by ip address. Human cannot remember ip address.
Either it has to reach IP address or domain name H1.ITKEFUNDE.COM.



Bottom level approach

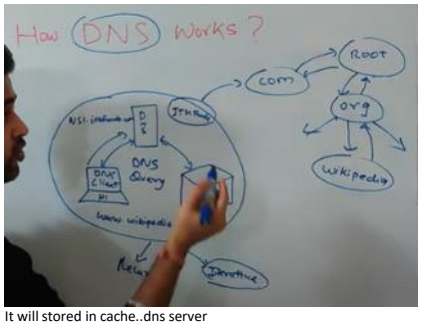
HOST file in window-> In which we maintain IP address.
First window check in host file then it only go to dns server

Every machine trying to connect client has DNS client installed then



Local dns for p1

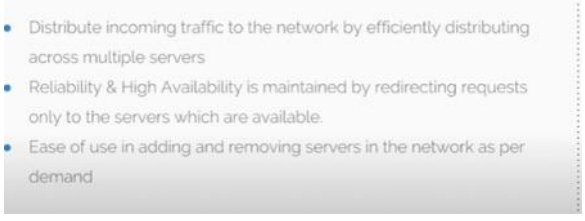
If we visit in new address. DNS local server will respond nothing answer.
Recursive or iterative
Then client connect to .com

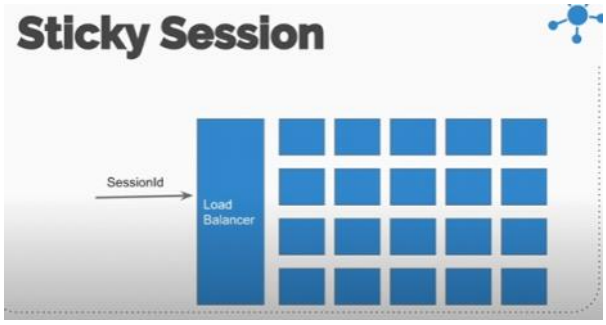


It will stored in cache..dns server

Load Balancer-> Distribute incoming traffice by efficiently distributing acrross multiple serve

Reliability & High Availability is maintained by redirecting request only to the servers which are avialbale





Client server cache the session id which server has generated so it will use that session only

Common cache.