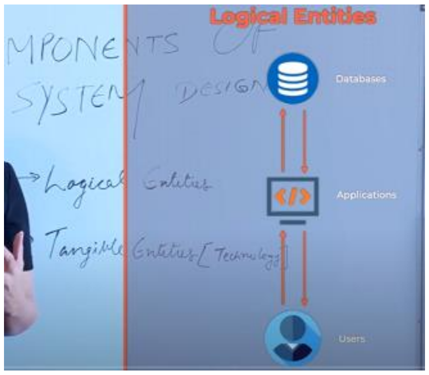
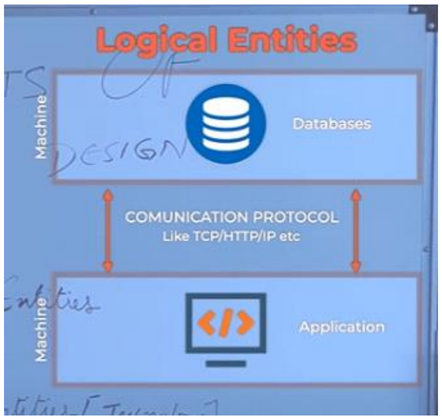
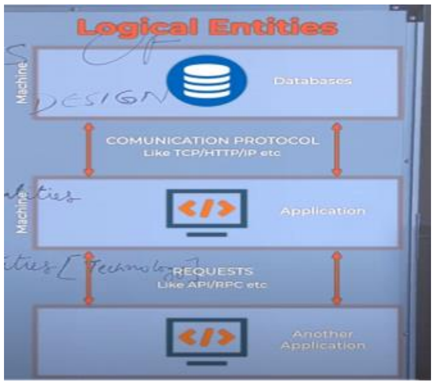
System design

Monday, February 8, 2021 3:48 AM

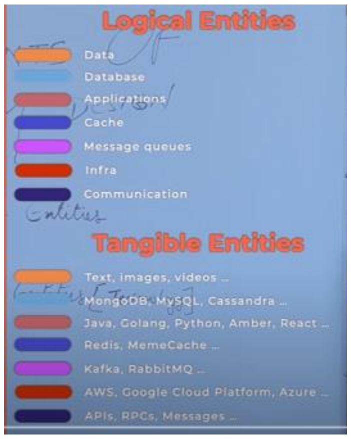




This two thing are on different machine that happen using communication protocol
Tcp/http



Logger don't have presentation layer

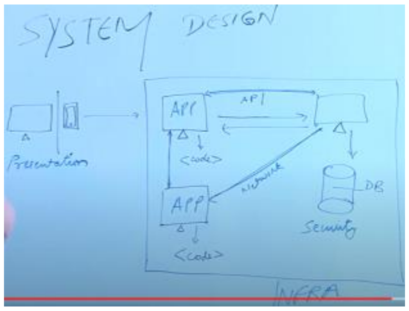


Logical Entities

🟧	Data
🟦	Database
🟥	Applications
🟦	Cache
🟪	Message queues
🟥	Infra
🟦	Communication

Tangible Entities

🟧	Text, images, videos ...
🟦	MongoDB, MySQL, Cassandra ...
🟥	Java, Golang, Python, Amber, React ...
🟦	Redis, MemeCache ...
🟪	Kafka, RabbitMQ ...
🟥	AWS, Google Cloud Platform, Azure ...
🟦	APIs, RPCs, Messages ...

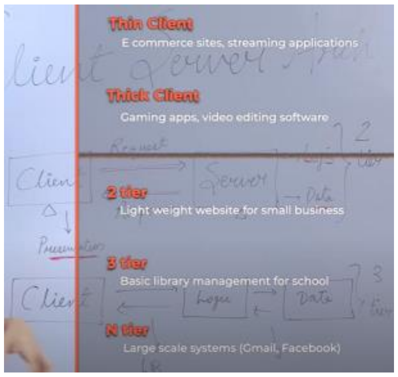


Thick client-> Outloook
Logic & processing sit on client side
And vise versa->Netflix-thin client

Client->Logic->Database->3 tier

Load balancer and proxy between
Client and logic layer or server layer
Caching layer between logic layer & database layer

Very logic manipulation store -> 2tier



Thin Client
E commerce sites, streaming applications

Thick Client
Gaming apps, video editing software

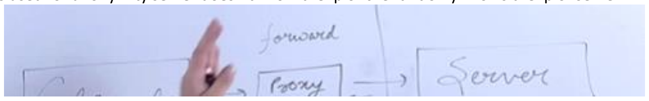
2 tier
Light weight website for small business

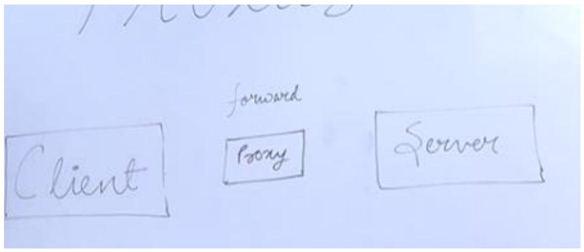
3 tier
Basic library management for school

N tier
Large scale systems (Gmail, Facebook)

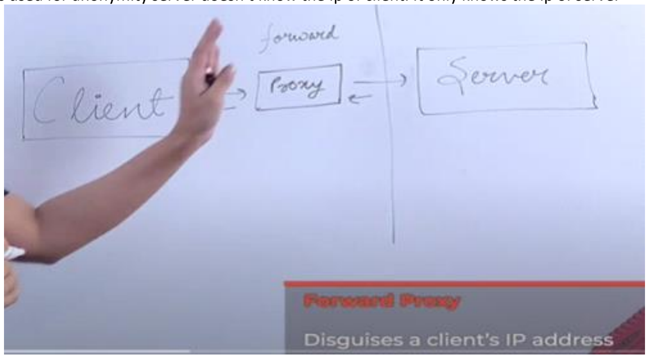
****
**
On behalf if we want to find appartment tell our assistant to find the apartment
Proxy--> Client server -> Forward ->Server

Forward proxy-> Client is never talking to the server. Proxy behalf of client talking to server.
This is used for anonymity server doesn't know the ip of client. It only knows the ip of server

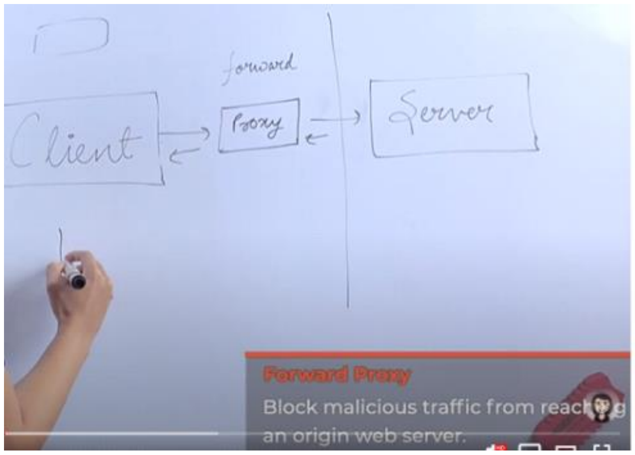




This is used for anonymity server doesn't know the ip of client. it only knows the ip of server

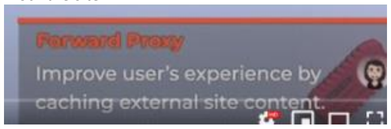


Forward Proxy
Disguises a client's IP address



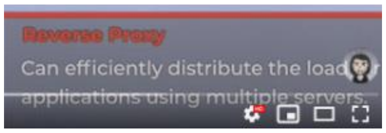
Forward Proxy
Block malicious traffic from reaching an origin web server.

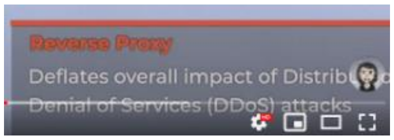
In institution where we block malicious traffic which need to go though proxy
Block the site

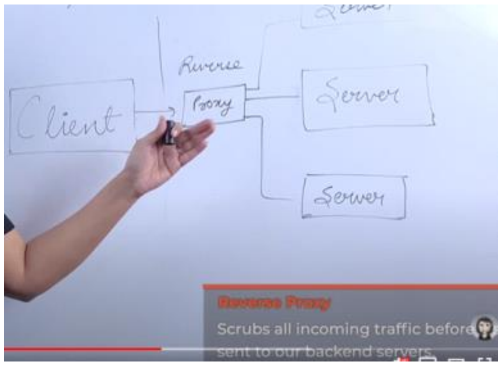

Forward Proxy
Improve user's experience by caching external site content.

Reverse proxy->

If same proxy acts behalf of server
In this client doesnot know the ip of server
Caching the response from the server
It is also used for sso


Reverse Proxy
Can efficiently distribute the load applications using multiple servers.


Reverse Proxy
Deflates overall impact of Distributed Denial of Services (DDoS) attacks



Reverse Proxy
Scrubs all incoming traffic before sent to our backend servers.

If reverse proxy fail then it become bottle neck.
Important security , privacy, for handling traffic

****

Data & Data flow

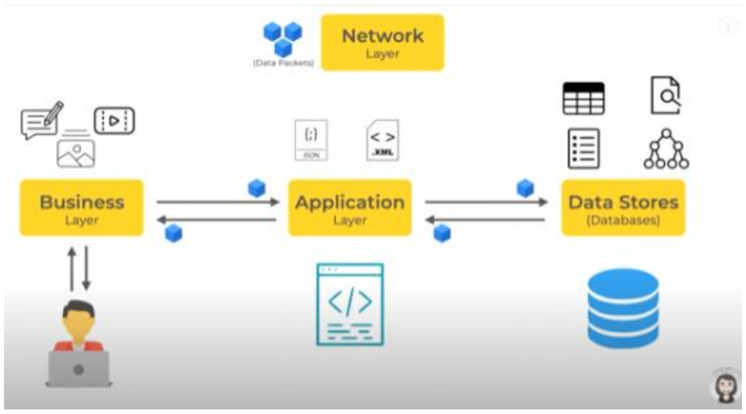
Example of building-> Core if no people in building. Which is centric to people. People
Livecore in building.

In same way data may get inside the system & manipulated in the system & computed int
The system and served backed to the people for use. People create acess or only
Access the data and cannot create it. Example youtube people cretae data and access it

Business layer-> Interact using images text videos

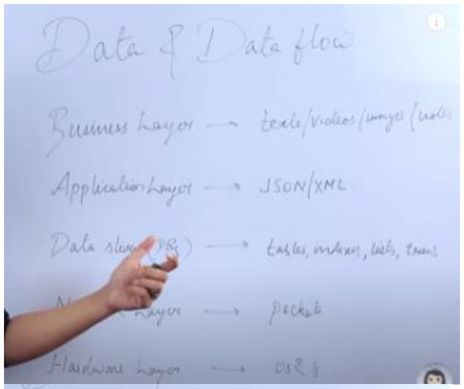
Application layer->Transformed into xml or json

Datastore-> Store where data retrivel can be efficently



Data travel using network layer using packets at hardware layer only 0 and 1

Exchange and retrivel and secure the data in effective manner



Data & Data flow

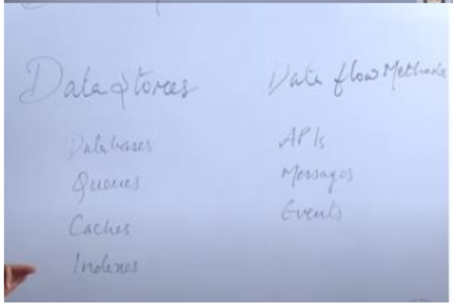
Business Layer → texts/videos/images/(video

Application Layer → JSON/XML

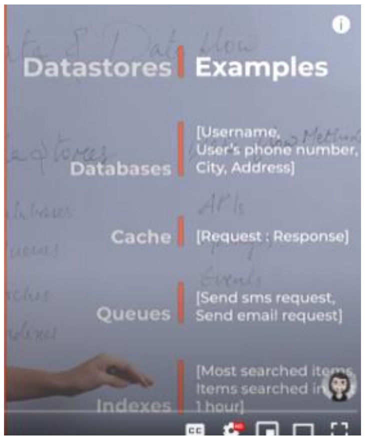
Data stores (DBs) → tables, indexes, lists, trees

N[illegible] Layer → packets

Hardware Layer → 0s & 1s



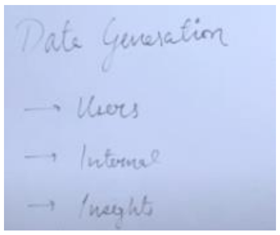
Datastores	Data flow Methods
Databases	APIs
Queues	Messages
Caches	Events
Indexes	



Datastores	Examples
Databases	[Username, User's phone number, City, Address]
Cache	[Request : Response]
Queues	[Send sms request, Send email request]
Indexes	[Most searched items, Items searched in 1 hour]

Data flow or travel from application to cache or server

Data generation-> Internal-> Logs-> which system generate own-> data about data


Data Generation

→ Users

→ Internal

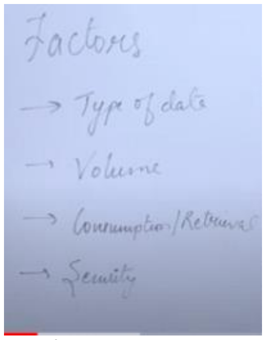
→ Insights

Users -> Create data

Insights->User interact with system then example if he buys something it will give bill
Or Netflix they will provide the search history profile history

****

Factors of data->


Factors

→ Type of data

→ Volume

→ Consumption/Retrieval

→ Security

Type of data->Text image->Like storing databases

Volume ->System which store tera byte or gigabyte

Consumption/ Retrivel-> where data is read allot and write is low
Or where consumption is less and write is more. Retrivel is high

Security-> System like transcation where data security is priority
It is ok user cannot login but it cannot comprimse the securtiy

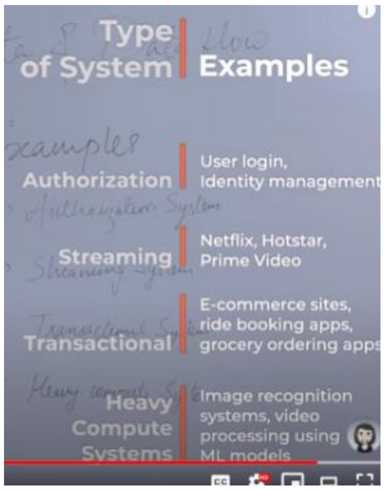
Examples->

Authorization System- User login where volume of data might be low only credential store but
Level of security will be high

Streaming- where data retrival is also high and data volume is also high-> Net flix

Transactional-> ecommerce|banking-> Journey of transaction it shouldn't fail
Journey of data is important if we want get headphone it might lead to different order

Heavy compute-> Image rcognition, Video processing. Camera take lots of video
Lot of data computed

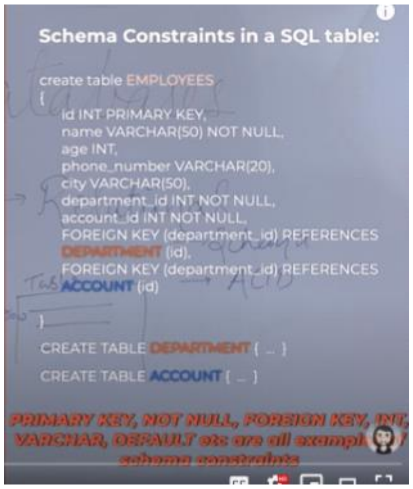


Type of System	Examples
Authorization	User login, Identity management
Streaming	Netflix, Hotstar, Prime Video
Transactional	E-commerce sites, ride booking apps, grocery ordering apps
Heavy Compute Systems	Image recognition systems, video processing using ML models

****

Databases->Different property differn volum different retrival

Relational-> Schema-> How data to be structures ->Ex employe table Department table



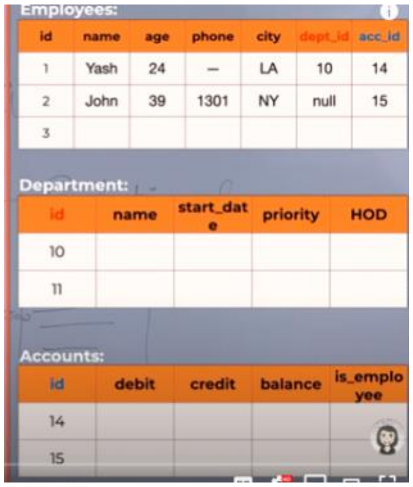
Schema Constraints in a SQL table:

```
create table EMPLOYEES
{
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    age INT,
    phone_number VARCHAR(20),
    city VARCHAR(50),
    department_id INT NOT NULL,
    account_id INT NOT NULL,
    FOREIGN KEY (department_id) REFERENCES
    DEPARTMENT (id),
    FOREIGN KEY (department_id) REFERENCES
    ACCOUNT (id)

}
CREATE TABLE DEPARTMENT { ... }

CREATE TABLE ACCOUNT { ... }
```

PRIMARY KEY, NOT NULL, FOREIGN KEY, INT, VARCHAR, DEFAULT etc are all exampl[illegible] schema constraints



Employees:

id	name	age	phone	city	dept_id	acc_id
1	Yash	24	—	LA	10	14
2	John	39	1301	NY	null	15
3						

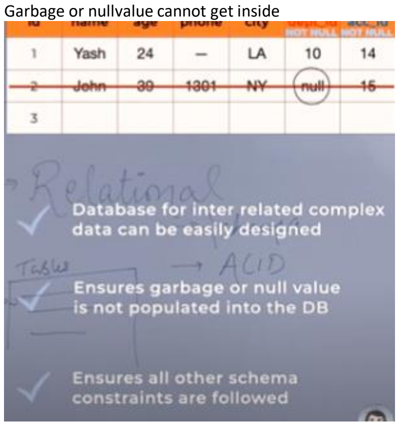
Department:

id	name	start_date	priority	HOD
10				
11				

Accounts:

id	debit	credit	balance	is_employee
14				
15				

How foreign key combined together all table-> One employee need one department and one accounts. It shouldn't be null this is schema constarint
Represent complex data
Garbage or nullvalue cannot get inside



id	name	age	phone	city	dept_id NOT NULL	acc_id NOT NULL
1	Yash	24	—	LA	10	14
~~2~~	~~John~~	~~39~~	~~1301~~	~~NY~~	~~null~~	~~15~~
3						

Relational

✓ Database for inter related complex data can be easily designed

✓ Ensures garbage or null value is not populated into the DB

✓ Ensures all other schema constraints are followed

ACID Propetries->>>>Atomcity-> Transction happen or nothing
Constientcy- Given at any time state time are consistent. Give same value to all time
Isolation-> Read write-> Read will not about write

Durability-> Whatever change is done it is logged properly and persisted in disk
Banking application->

Things which cannot stored.
Where schema not fixed. Where coloumn might changed-> Table size grow. So join can be expensive

Scaling is problem-> It is difficult to divide the table into two parts
1-10000
10001-5000

Nonrelational->

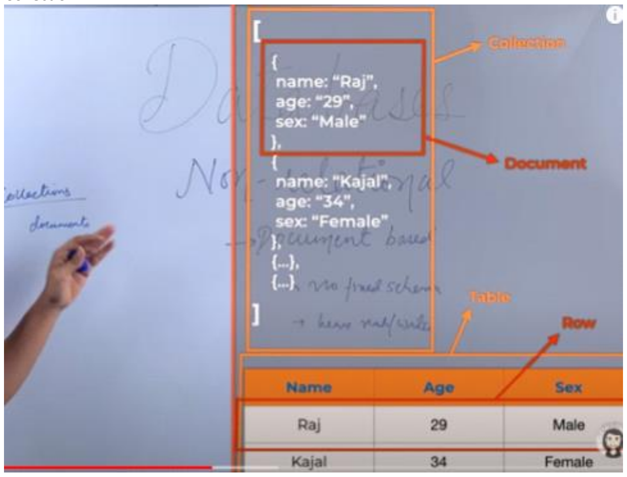
Key value stores

Caching-> Reddis> They provide quick access

Document based-> When we are not know the schema. Heavy reads and write

When we want query the whole data in one go. We don't join. It decrease complexity

Collection

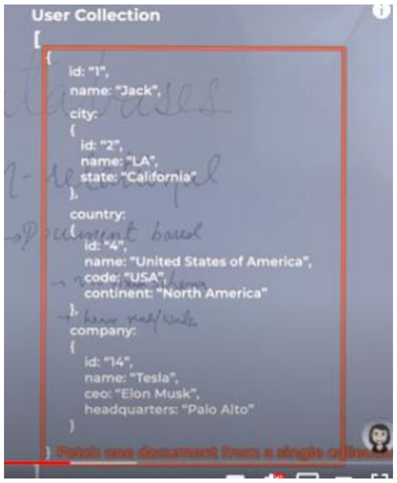


```
[
  {
   name: "Raj",
   age: "29",
   sex: "Male"
  },
  {
   name: "Kajal",
   age: "34",
   sex: "Female"
  },
  {...},
  {...}
]
```

Name	Age	Sex
Raj	29	Male
Kajal	34	Female

Product details->ecommerce site->

Large data-> if we put all information in document



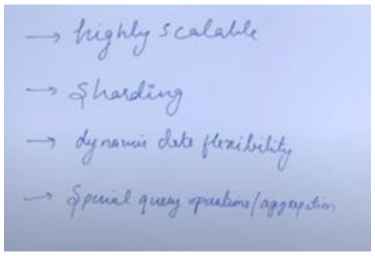
User Collection

```
[
  {
    id: "1",
    name: "Jack",
    city:
    {
      id: "2",
      name: "LA",
      state: "California"
    },
    country:
    {
      id: "4",
      name: "United States of America",
      code: "USA",
      continent: "North America"
    },
    company:
    {
      id: "14",
      name: "Tesla",
      ceo: "Elon Musk",
      headquarters: "Palo Alto"
    }
  }
```

Fetch one document from a single collection

Downside of document based db-> It may give null value
DB don't provide acid property

Used of nosql

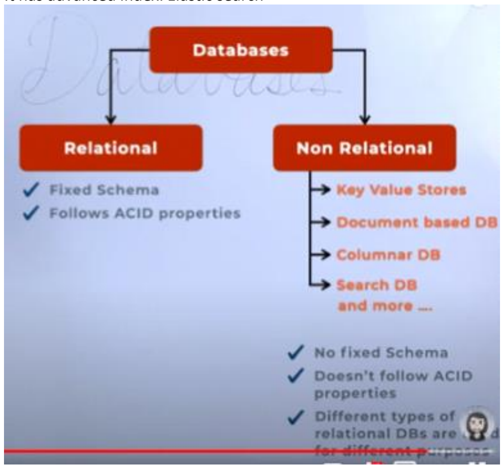


→ highly scalable
→ sharding
→ dynamic data flexibility
→ Special query operations/aggregation

Column dbs-> Midway of relational and document db
Who have fixed schema bit don't support acid properties
 Event data-> Heavy interaction. So that analytics done-> Like music or dislike
Or sending data in 10 sec. Particular db

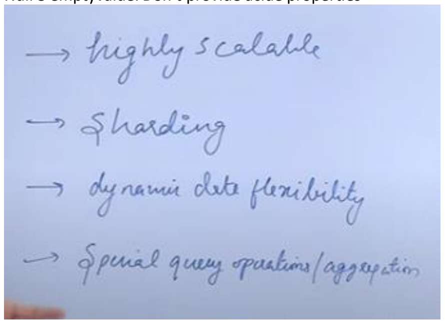
Cassandra
Hbase
Sylla

Search dbs->booking from flight, booking from hotel
It has index.. Find the chapetr or page
It has advanced index. Elastic search



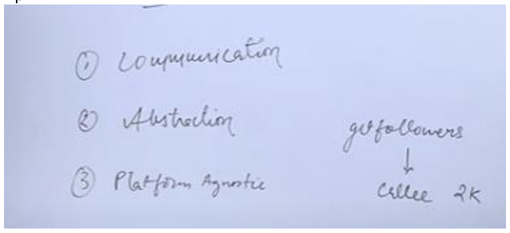
Caching solution is made using non relational db. Memcahe they are quite fast

Document based db when no fixed schema
HEAVy read and writes
Coloum or row can be change over the time
Null 8 emptyvalue. Don't provide acide properties



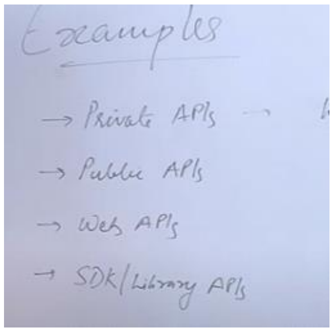
→ highly scalable
→ Sharding
→ dynamic data flexibility
→ Special query operations/aggregation

Api

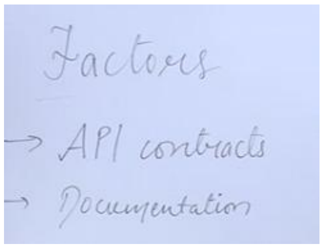


① Communication
② Abstraction
③ Platform Agnostic

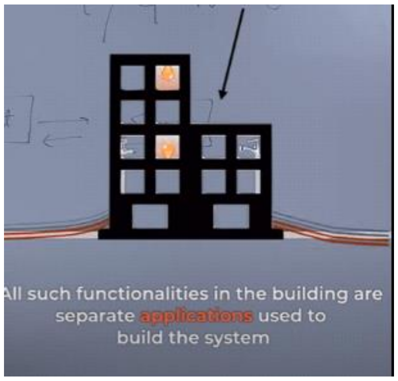
getfollowers
↓
Cellee 2K



Examples
→ Private APIs
→ Public APIs
→ Web APIs
→ SDK/Library APIs



Factors
→ API contracts
→ Documentation



All such functionalities in the building are separate applications used to build the system

Water suuplies electric supplies which server or fullfill the requirments

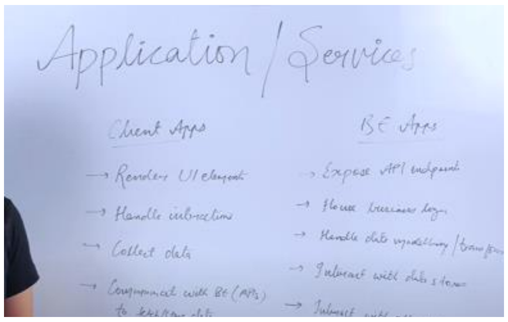
Deskotop or web app
Collect data
User interaction
Send the data to backend

Backend -> Handling all the notes
Handling user profile
Subscription or handling paymnets

Application service perfomr fulfill certain task or certain responsibity or fetaure for that developed

Framework do common task which is needed most time which make life easier of devloper

How applictaion talk to each other is language ignostic it doent matter which language it is written



Application / Services

Client Apps
→ Render UI element
→ Handle interactions
→ Collect data
→ Communicate with BE (APIs) to fetch/save data

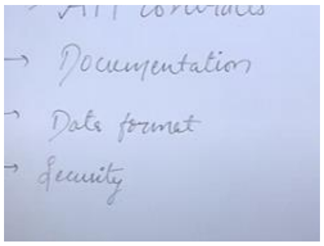
BE Apps
→ Expose API endpoint
→ House business logic
→ Handle data modelling / transform
→ Interact with data stores
→ Interact with

Data transformation and modelling and save to db

Monitoring-> it should not use many resource reliable

When one piece of code has to interact with another piece of code

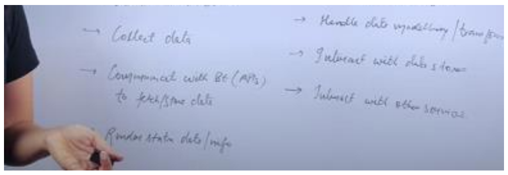
It can be same machine or other same machine
Send data



Api security is imp-> It can hit multiple time and make down the system

Throttling
Rate limiting
Can wrong input. Bring system down by hitting api multiple time

To write api
RPC
Soap
Rest


Collect data
Communicate with BE (APIs) to fetch/save data
Render static data/info
Handle data validation / transform
Interact with data stores
Interact with other services

Private api-> Like payment application. We doent know how that application is running that service

Public api-> Google map api, Weather api

Web api->Running on cloud. Post photo

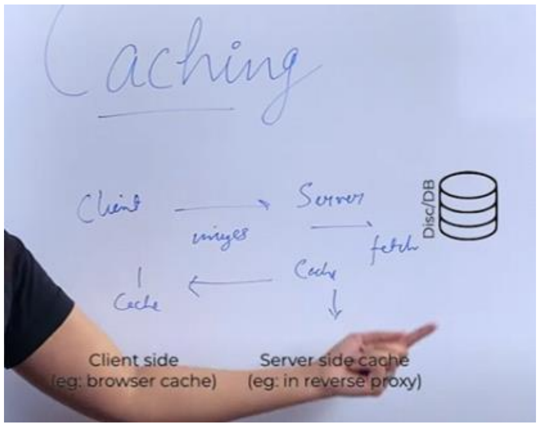
Threading library-> Lock relase fork

When one piece of code has to interact with another piece of code

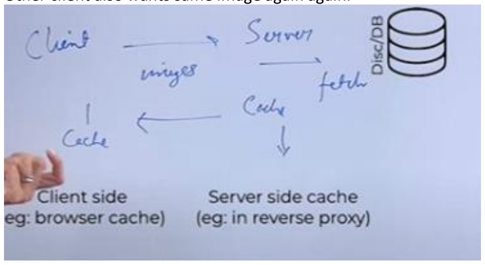
It can be same machine or other same machine
Send data
Fetch data

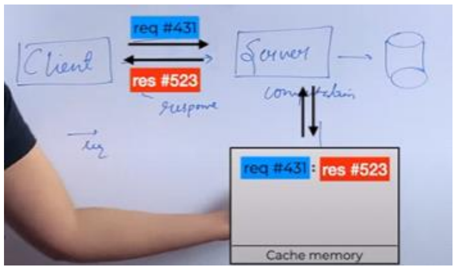
Interface-> In this code would know how to call other code or endpoint.
But it doesn't how other code is doing
Abstarction-> Freedom of implementation

Platform agnostic-> It doent worry about language



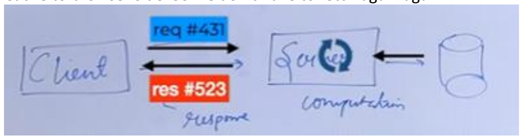
Images fetching from db is expensive. So take one time then after that cache it.
Other client also wants same image again again.





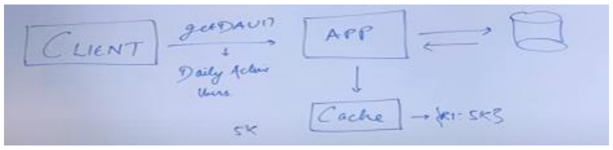
Cache to browser side. So we don't have to fetch again again

Cache hit if cache is presnet



Fethcing data and computation is redundant here

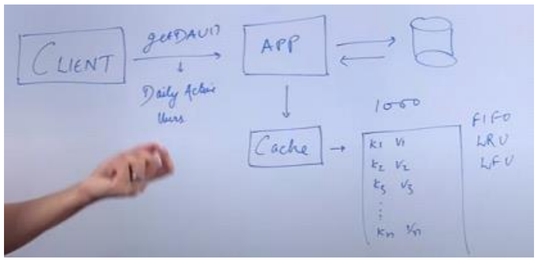
Data form cache should be invalidated or removed from cache. If any new data come which has new value but there is old stale value present in cache should be invalidated. So we have to updating with new value is called cache invalidation.



How to decide when to remove
1.by keeping the expiry time. TTL-> Time we enter when data will expire
If ttl is 10 minutes. T0 min-> 5K T8 min->5K form cahce
If t11-> then it will invalidate the cache and fill with new data

If we keep low ttl then there data will vanish fast so db fetching and computation will happen frequenlty
If we keep high ttl then people get stale data for longer time

Two problem-> Is it ok to keep stale data in db or increase number of hits on db by putting low ttl



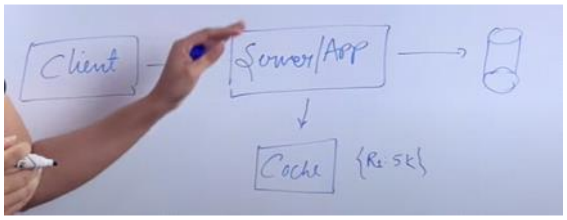
Application code can also update the cache vvalue when new value is updated in application code

1. Removing the value from cache
2. Updating the value from cahce is also two way

It may happen the limit of cache would be 1000 keys
In this existing key should be evicted so that there will be room for new key

Cache patterns

1.Cache aside patterns-> In this cache always to application never talks to server
User activity. Aplication find from db then store it inot db if value comes then application need to check in db is value presnet in cache
Cache never talking to db. Aplication code is only talking to cache.



When problem happens. When new value updated in db in that case cache should be invalidated
Or ttl. Or there will be code in application whenever db is updated it invalidate the cache or update the value

ADVantage when cache wholes goes down the application can keep serving request. It may slow down

Disadvantage-> Long expiry or updating logic to cache

2. Read through pattern

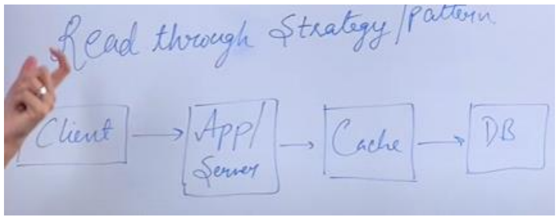
   Cache sit between application and db
   Application server always talking to cache never to the db

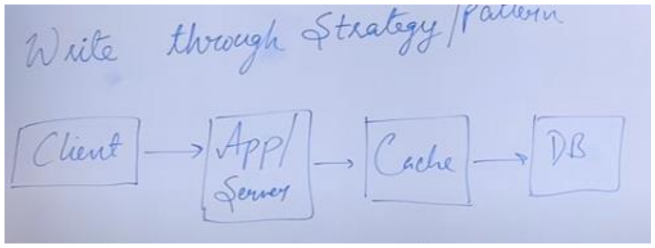
   So whenever first data will come then there will be cache miss and fetch data from db and update in cache and send request to applciation

   Lots of read it help alot

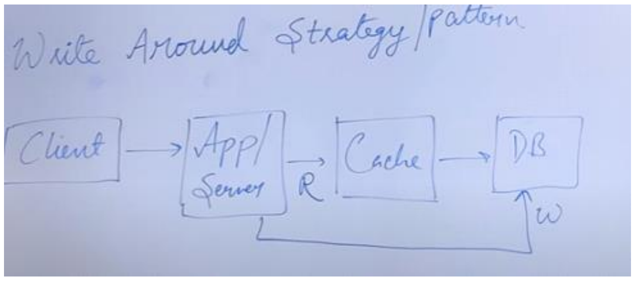
   Disadvanteg-> cache mis first time

   In cache aside pattern-> Application modified can be put in cache but
   In rEad through whatever data is taken from db is put in cache. Modeling is less





Write through. Cache responsbility to write the data and replciate to db
Extra layer of latency

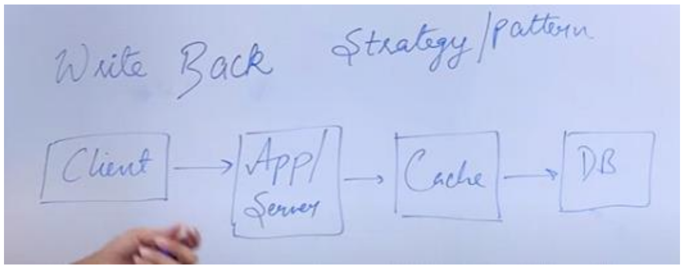


Write through

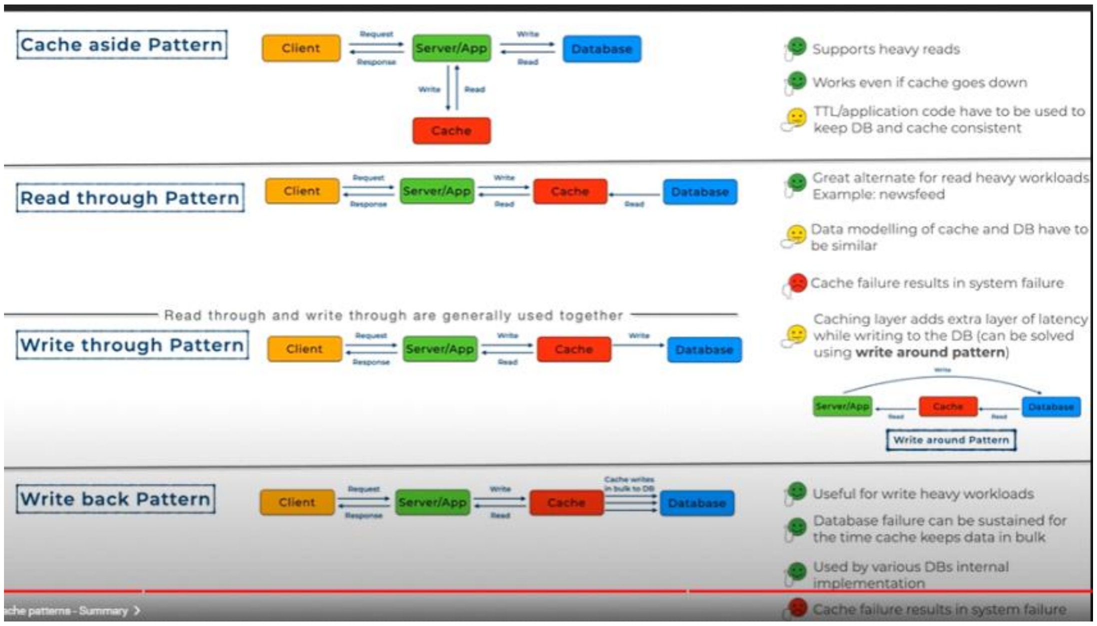
App direclty talks to DB while writing but in reading it talks to Cache

Save hopping to write

When there is load of write is heavy



Accpet the writes it update in db after sometimes

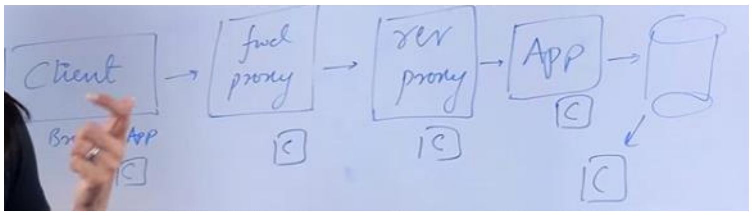


Supports heavy reads
Works even if cache goes down
TTL/application code have to be used to keep DB and cache consistent

Great alternate for read heavy workloads Example: newsfeed
Data modelling of cache and DB have to be similar
Cache failure results in system failure

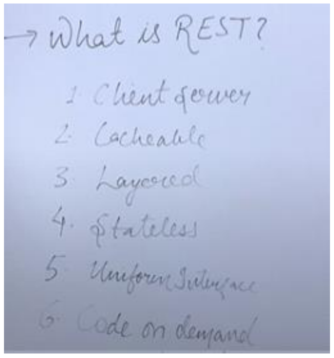
Caching layer adds extra layer of latency while writing to the DB (can be solved using **write around pattern**)

Useful for write heavy workloads
Database failure can be sustained for the time cache keeps data in bulk
Used by various DBs internal implementation
Cache failure results in system failure



****

Rest



→ What is REST?

1. Client server
2. Cacheable
3. Layered
4. Stateless
5. Uniform Interface
6. Code on demand

Works in client server architecture

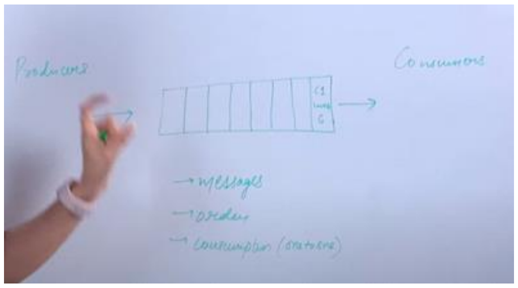
Stateless-> One server get request from multiple clients

All client is same for server. It doesn't matter for server whose client is calling or not. They will give same info. They don't store address of client.
There might be some example where serevr store the clients address

It should always follow same protoclos and rule. Example if they have implemented using rest then it shouldn't be implmented using other protocol

Path parameter-> If we are looking for particular resource /book/1
1000 books-> /books?limit=20& offset=0-> Query paramtere

Quueues



If customer want invoice for 6 months then we don't need to process and give at exact time.

We just tell give me 6 months invoice it will goes to queue when queue will take that event it will process for that account and give it back to user in mail. This are all independent if million of user asking invoice at same time. So queue will take and store it and process each one and give back to user

Consumer can handle lots of request

When request of user increases then it we can scale
Now rate consumtion will increase as there are more servers

If componnent is down then message remains in the queue it doesn't happen anything to queue

Once message is consumed by consumer then it is removed from queue

Order of message-> Example chat app where order of message matters a lot

M3 failed in unorder then it would be pushed in to retry queue . Consumer can take other m messgae

We have to take one event from queue do some task on each event like

M3 failed in unorder then it would be pushed in to retry queue . Consumer can take other m messgae

We have to take one event from queue do some task on each event like modeling or emailing.

If we want to do bulk operation then we can put in queue and do at own speed

When we want to take one message to do multiple process then we can use publish subscriber

Once message is consumed by consumer then it is removed from queue

Order of message-> Example chat app where order of message matters a lot
When two party talking over tunnel or channel. Those message should be delivered in same order

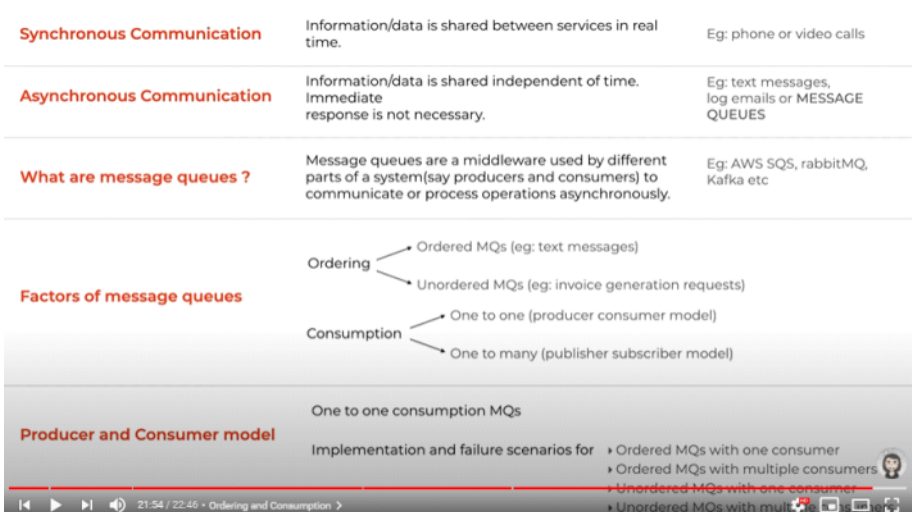
Invoice generation-> Inorder will also work fine

If m3 in queue is blocked or failed then it will not allow other messgae after that to pass before acknowldigning
Processing will stop

So if M1 taken by C1 and it fails couldn't acknowldge queue that it has process the queue. So consupmtion will stop heere

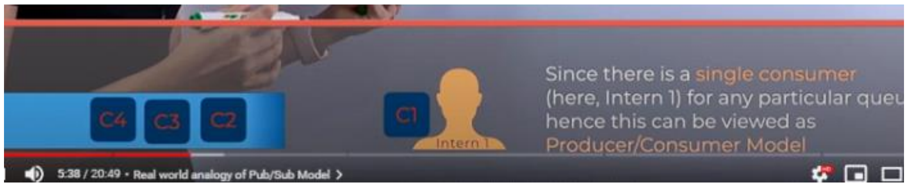
Downside of FIFO

Unordered queeue.



Synchronous Communication	Information/data is shared between services in real time.	Eg: phone or video calls
Asynchronous Communication	Information/data is shared independent of time. Immediate response is not necessary.	Eg: text messages, log emails or MESSAGE QUEUES
What are message queues ?	Message queues are a middleware used by different parts of a system(say producers and consumers) to communicate or process operations asynchronously.	Eg: AWS SQS, rabbitMQ, Kafka etc
Factors of message queues	Ordering → Ordered MQs (eg: text messages); Unordered MQs (eg: invoice generation requests) Consumption → One to one (producer consumer model); One to many (publisher subscriber model)	
Producer and Consumer model	One to one consumption MQs Implementation and failure scenarios for	• Ordered MQs with one consumer • Ordered MQs with multiple consumers • Unordered MQs with one consumer • Unordered MQs with multiple consumers

21:54 / 22:46 • Ordering and Consumption


Since there is a single consumer (here, Intern 1) for any particular queu hence this can be viewed as Producer/Consumer Model
C4
C3
C2
C1
Intern 1
5:38 / 20:49 • Real world analogy of Pub/Sub Model

Annoucmenet happens in hospital to interns
To call interns at particular place

To call doctors

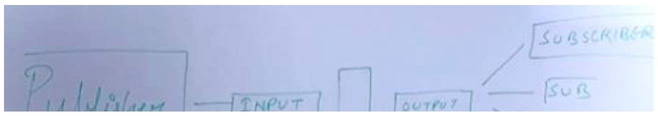
To call people when fire take place

What is Pub/Sub Messaging ? (and related terminologies)

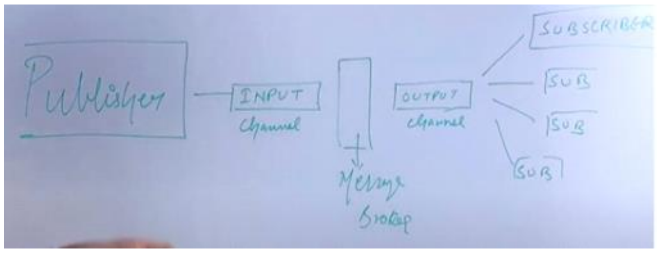
In any system who publish the message who told that any event has happened like order has been placed.
Publisher publish whoever wants to react will react
Input channel. Store the message
Output channel. There format can be changed. Categorsied in different topics.

So it may possible that susbcribver don't need to react on every message otherwise it will be noise

Consumer choose only those message who are concerned with

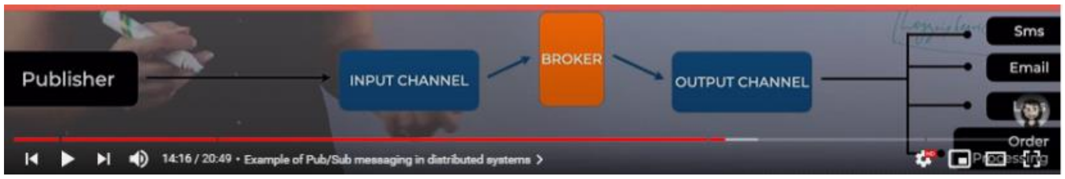


Message broker-> Publisher published the data may it may need to modifed
It may have to enrcihed with other message.
It can be divided into differne categories in different topics.
Like who will see the patient who have ear problem they will categorised by message broker
That doctor will be on 4 th floor



Message broker-> Publisher published the data may it may need to modifed
It may have to enrcihed with other message.
It can be divided into differne categories in different topics.
Like who will see the patient who have ear problem they will categorised by message broker
That doctor will be on 4 th floor

Subcriber listen which they concern



Ecommerce website where we can buy things

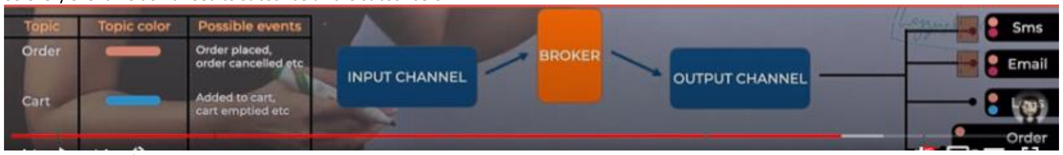
So publisher send message order place with the order id

Broker will take order id.. Then it will take more information. Add more data and send it to output channels

1. Whar order is place
2. Who order it
3. What time it creatde it

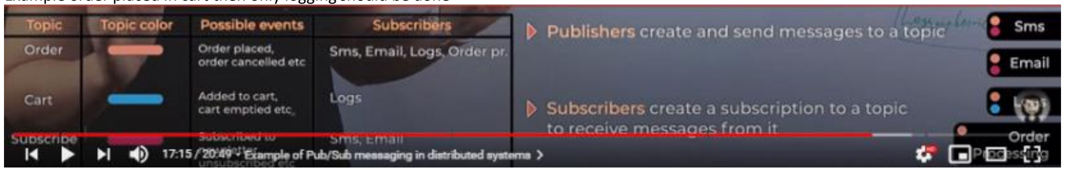
One order is published different channels reacted it

So every event we don't need to subscribe all the subscribers



Topic	Topic color	Possible events
Order	🟥	Order placed, order cancelled etc
Cart	🟦	Added to cart, cart emptied etc

Other service don't need to react.
Example order placed in cart then only logging should be done



Topic	Topic color	Possible events	Subscribers
Order	🟥	Order placed, order cancelled etc	Sms, Email, Logs, Order pr.
Cart	🟦	Added to cart, cart emptied etc,	Logs
Subscribe	🟥	Subscribed to newsletter, unsubscribed etc	Sms, Email

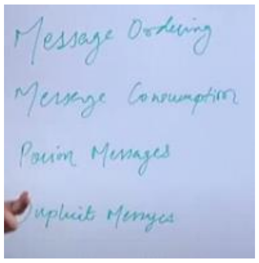
- Publishers create and send messages to a topic
- Subscribers create a subscription to a topic to receive messages from it

Sms
Email
Logs
Order Processing
17:15 / 20:49 • Example of Pub/Sub messaging in distributed systems

Just publisher push and don't need to follow up

Publisher and subscriber not depened on each other
It can be scaled and behave as distibuted architecture when message increases

In pubsub order is not gurantee or defined

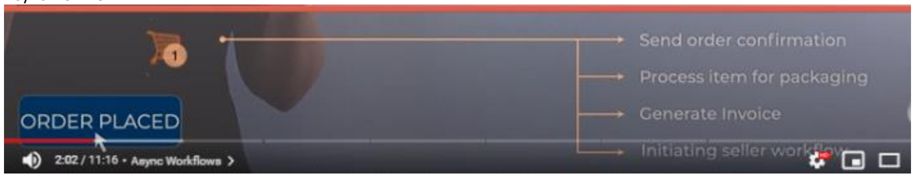

Message Ordering
Message Consumption
Poison Messages
Duplicate Messages

To provide ordering we can use priority queue

Repeated message can also be there. We need to remove or malformed message

Feature

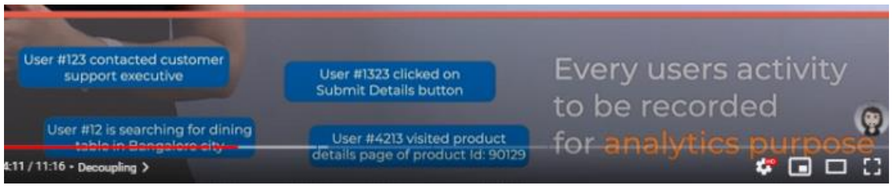
Async workflow


1
ORDER PLACED
Send order confirmation
Process item for packaging
Generate Invoice
Initiating seller workflow
2:02 / 11:16 • Async Workflows

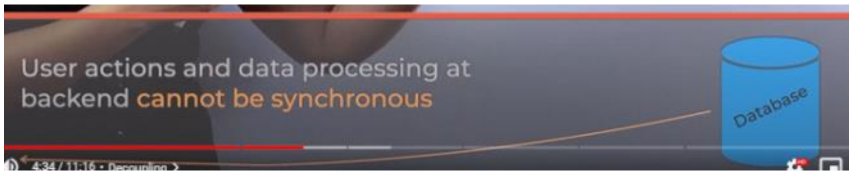
No dependency. Just publish all the subscriber which are related to that will react according to it. Decoupld
No is dependen on other

Some are synchronous. Which need to happended just . Like add order in cart That cannot happened in async.

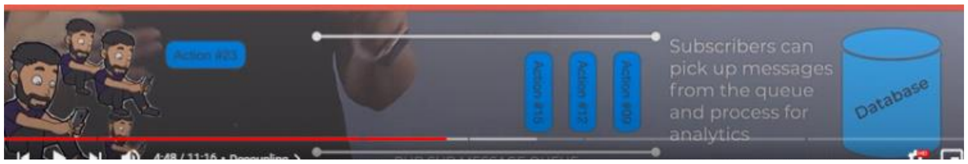
Decoupling


User #123 contacted customer support executive
User #1323 clicked on Submit Details button
User #12 is searching for dining table in Bangalore city
User #4213 visited product details page of product Id: 90129
4:11 / 11:16 • Decoupling

Every users activity to be recorded for analytics purpose

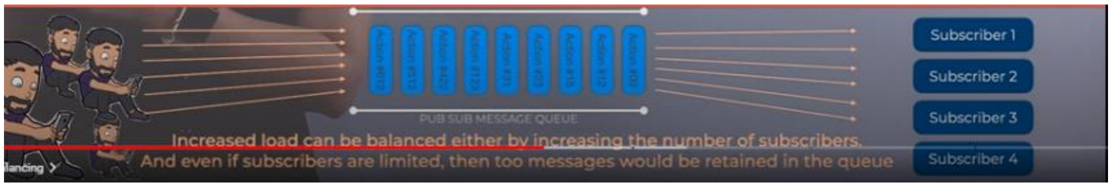


User actions and data processing at backend cannot be synchronous



Subscribers can pick up messages from the queue and process for analytics

Load balancing



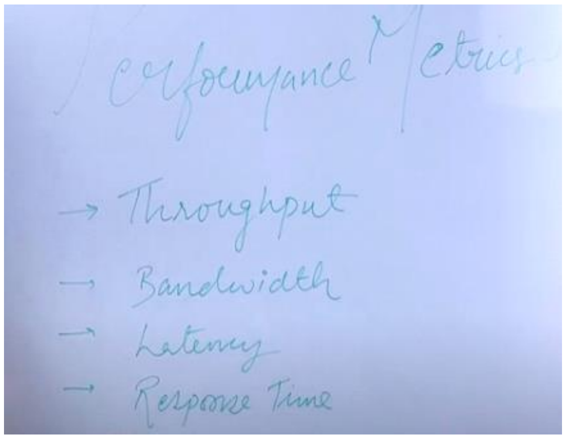
Increased load can be balanced either by increasing the number of subscribers.
And even if subscribers are limited, then too messages would be retained in the queue

If there heavy load it can block db so to avoid thie we use load balancer

Defered processing. The event which are not useful just put in retry queue.
Do some task at later time give some time do it after some time

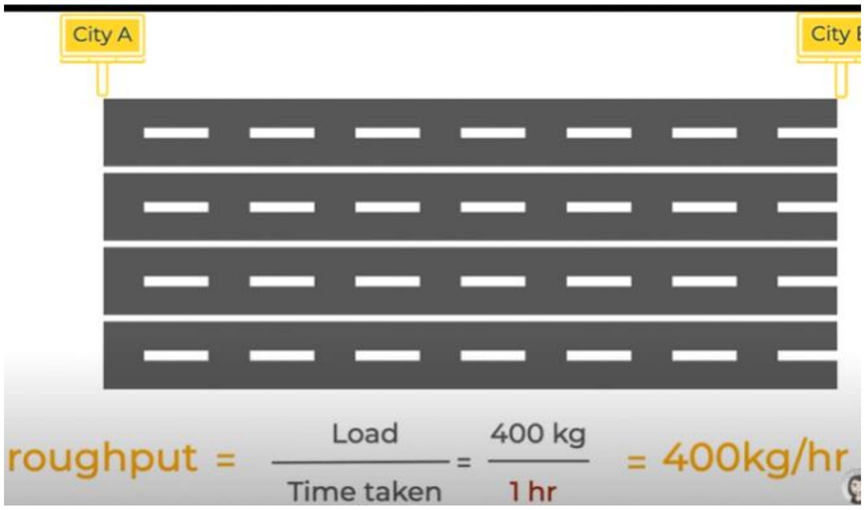
Data streaming.
The sensor which take and contionusly putting on queue can be easily handled by pub sub patern

Performance metrics



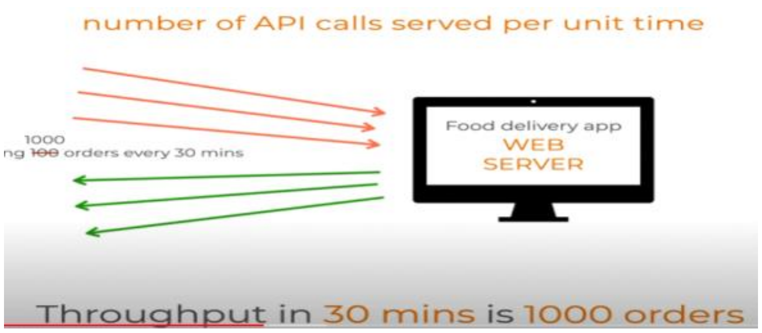
Performance Metrics
→ Throughput
→ Bandwidth
→ Latency
→ Response Time

Throughput- Some amount of work done in particular time.
20 onions in 1 hour throughput is 20

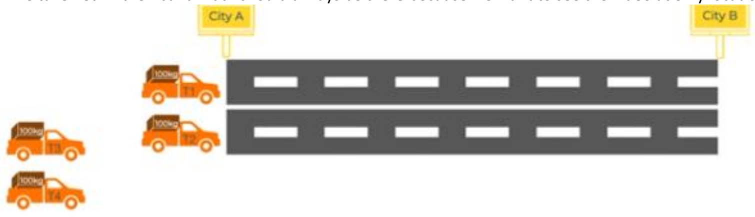


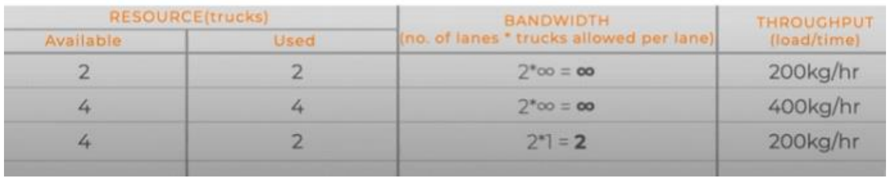
How can throughput can be increased.
1 way is that capacity of truck can be increased.

number of API calls served per unit time

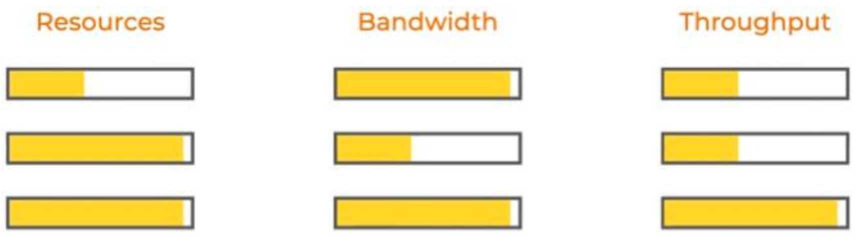

number of API calls served per unit time
1000
ng ~~100~~ orders every 30 mins
Food delivery app
WEB
SERVER
Throughput in 30 mins is 1000 orders

Bandwitdh-> Data getting transferred on connected networks on multiple location.
If we take netflix then bandwidth should always be there because we want to see the video at evry location without lagging





RESOURCE(trucks)		BANDWIDTH	THROUGHPUT
Available	Used	(no. of lanes * trucks allowed per lane)	(load/time)
2	2	2*∞ = ∞	200kg/hr
4	4	2*∞ = ∞	400kg/hr
4	2	2*1 = 2	200kg/hr



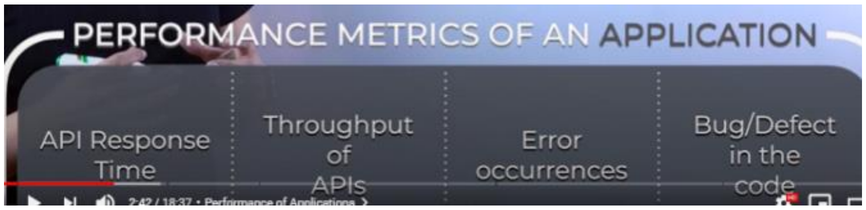
When we say system can serve 1000 request / sec. that means
If api responsd time is slow then like 2s then even we have bandwith but we will not able to utilize the request

Bandwith reponse time throughput to increase performance

How to use performance of metrics to find the performance of appliction.

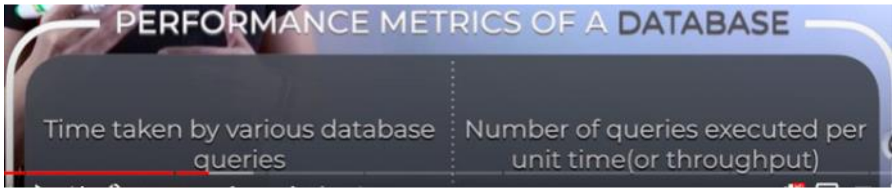
We need api response time. In that how much time this api are responding

How muhc error is prodcuing. Logs system show lots of error . It affect performance of system



PERFORMANCE METRICS OF AN APPLICATION

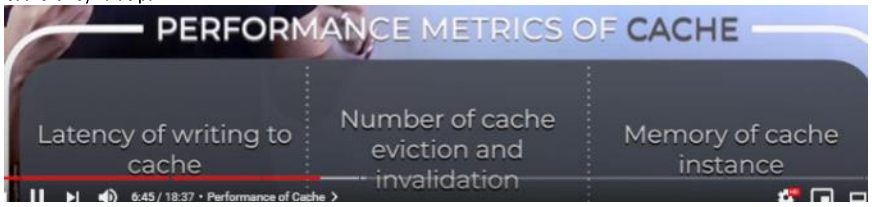
API Response Time	Throughput of APIs	Error occurrences	Bug/Defect in the code



PERFORMANCE METRICS OF A DATABASE

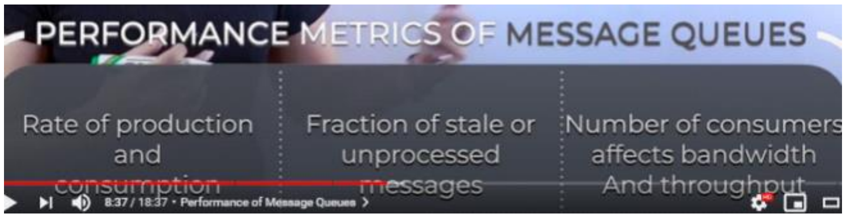
Time taken by various database queries	Number of queries executed per unit time(or throughput)

Cache is key value pair



PERFORMANCE METRICS OF CACHE

Latency of writing to cache	Number of cache eviction and invalidation	Memory of cache instance



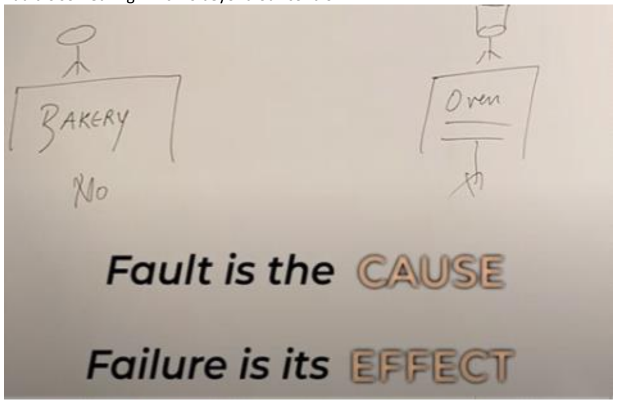
PERFORMANCE METRICS OF MESSAGE QUEUES

Rate of production and consumption	Fraction of stale or unprocessed messages	Number of consumers affects bandwidth And throughput

Fault vs failure

If we order cake in bakery and bakery oven gets break on that day. Then we go to evening to collect cakes but we get response no we cannot give you cake;

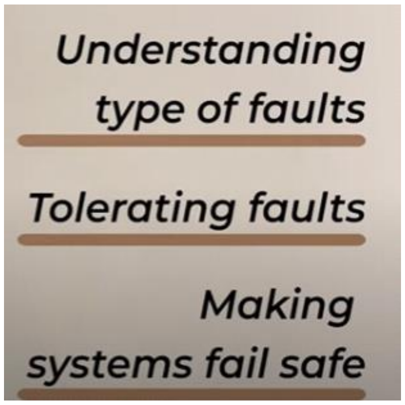
Fault is something which is beyond our control.



Even customer will not satisfy but let him choose the cake which is already present

We can have multiple ovens

Fault-> Network fault Network time out Hardware issue Memmory issue. Software bugs



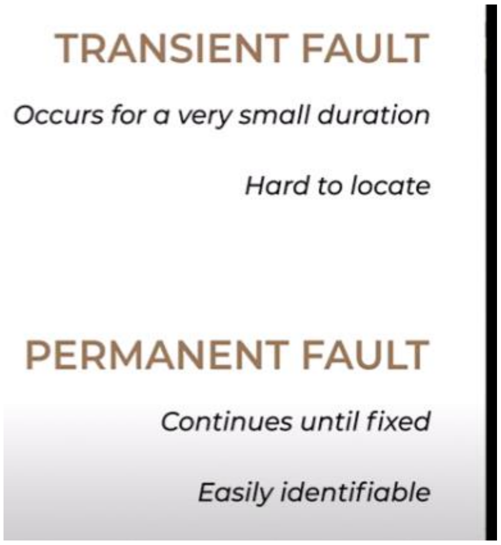
Understanding type of faults

Tolerating faults

Making systems fail safe

Application running out of meemory or high cpu usage then it will not able to serve request 4000
Software Bug

In harwdare faults we can have replication



TRANSIENT FAULT

Occurs for a very small duration

Hard to locate

PERMANENT FAULT

Continues until fixed

Easily identifiable