When we make system smaller and smaller it becomes to MS.

Pro: easy for enginer to work on these service. RAMP up quickly. You can scale up and scale down each independent service indepedently. I can throw machine to a particular service when it becomes bottleneck.

CON: now u don’t have single service which does all these….

For every task u have to do network call. Consistency a challenge, costly. Context of user journey is spread across many service, observer, debug is challenge.

So we nee dto setup observability. Warm path for log, hot path for

Trace id I need to combine log for many service..

SOA: is earlier level arch. One service, you create a middleware has some independent service, all of them will coupled each other, part of same Business logic, share a same DB. People don’t use these days SOA.

MS is.. new and adv.

SPrangler fit: from monolith to mnicro service.

The microservice you create will be such that those services, are now act differently. For other use case u can use these service for other job as well. Every service is independently.

ESB: enterprise service bus..

Event driven arch: makes MS powerfull…

How SM can talk to each other… HTTP call, event driven arch…

A SM does its job, when need to talk it ads a message in a queue. Service takes the message from the queue. Swiggy example. Order service donet make a call to payment 1-1 manner. I populate to message queue. Whosoever is interested payment, notification etc.. they can poll the message from there.. SM becomes powerfull when you cuple them with Event driven arch. FK use MQ for use case… event driven makes MS strog, reason 1 service becomes very decoupled from other service. Reading properly or not, error or not, other service will not be effected. They do their job.. make it decoupled..

**Consistency in case of MS:**

When multiple servce,m each service will have own resource such as DB, hard to keep them in sync and consistency together is a challenge..

Example: if I have Flpkart has a order service, when a order is created or gets delivered or modified, order service will be interacting with Client.

Another service: inventory service, where you maintain the inventory, for the SKU I have these many item in this warehouse..

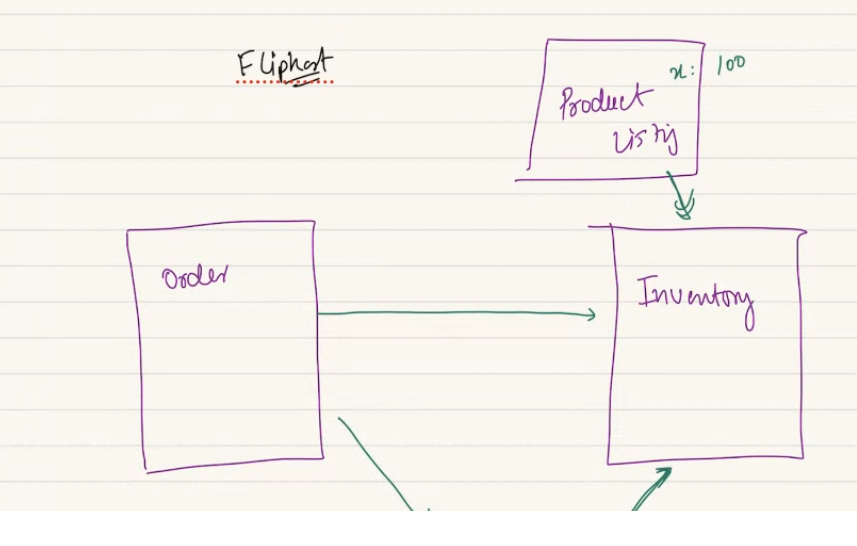
Logistic service: ensure that delivery happens, reaches to people who paid for it.

Whenever a user want to placxe a order, order service wil have to talk to inventory service, to figure out if the product is there or not in inventory service. Do u have the product or not? Can be at beginning..

Product listing service: need to talk to inventory, product can be listed as available or not.

When someone place order. Order service can talk to inventory, also gives a estimated time of delivery….to find delievrty itme 2-4 days they talk to logistic, logistic talks to inventory,,, ifproduct is available in delihi warehouse..as custome from delhi…

Every service to act properly, they need to talk to each other.



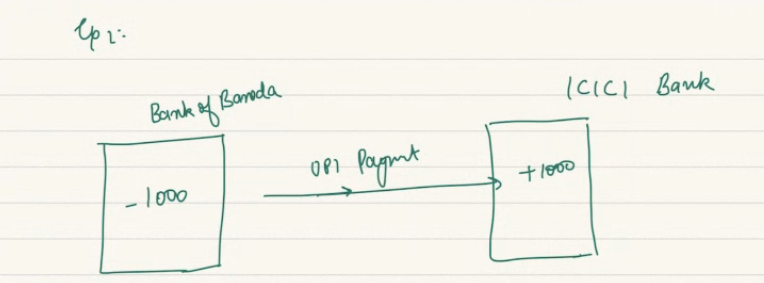
Also we need to persist the product listing and inventory data together. They have to match… problem of keeping data persistence with each other is a challenge. When we have distributed data the problem of consistency emerges as communication happens over network andnot realiable. In MS it becomes multifolds..

Its not simple master-slave arch, not a simple solution, these problem become even bigger, its not 1 servce, many service. Consistency problems on steroids.

Another example:

If u are having bank account in BOB bank. U wanna make payment.. UPI payment to some other user.. that user has a UPI maintained by icici bank upi. So I pay from BOB to other bank. There can be other intermediate app. Phoen pay- cred application… many stackholder present here…

So I have to send 1000 rupee. Need added and subtracted from here and there.

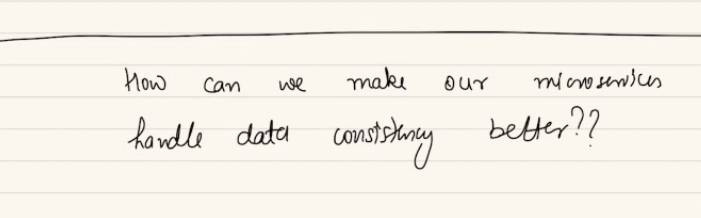


All independent price , each one of them are own service on their own, can be diff bank, diff stack holder. Each of them are service,..

If data in service A says money deducted and in B says not received.. so inconsistency can eb very big. Each has own DB, BOB and ICIC DB can be out of sync…

**Consistency is a problem in Microservices.**

By design, by theory each MS has own independent database..

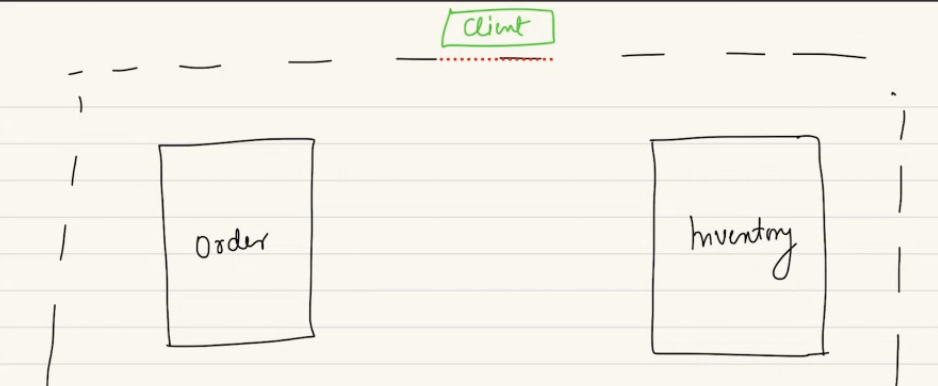


Multiple ways:

1. TWO Phase Commit, aka 2PC
2. Distributed Transaction, SAGA Pattern. Inside saga we have orchestrated or choreographed.

**TWO Phase Commit:**

Order and inventory service, each service will have own arch with it, app service, CH LB, diff DB, etc. between 2 service will have API gateway..



When client try tp place order, talk to order service, via GWY, API GWY, Order service.

Order service need to talk to inventory also reduce quantity by 1. For now lets consider it’s a synchronized call, HTTPS call, RPC, GRPC call. You are making API calls.

Also I need to check if we have 100 quentity in inventory..

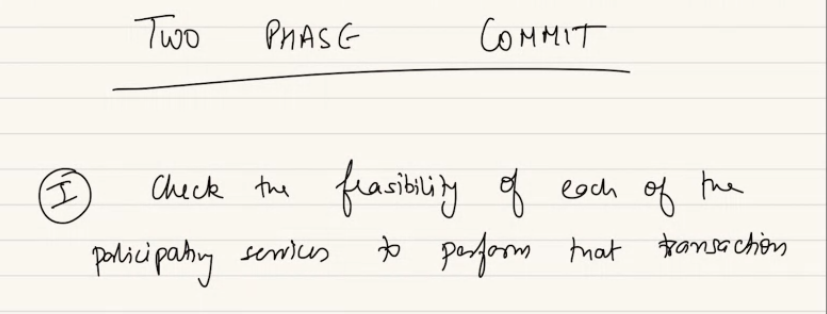
2 PHC says, whosoever is going to be involved in this transaction..(a Transaction is a sequence of steps you combines together and give guarentee of ACID, either all or none will happen.) here a transactuion need to happen. Order, no of product should decreament all need to be done. Cant be left in some part of job is done, some didn’t… all these were easy if single DB you have. We don’t have single service, some task hasspens in service A, some in B service.. problem of maintaining in here is tuff…

People use 2PHC, call it anti-pattern and hence does less frequently…

***2 PC: there are 2 phases to the commit.***

given we have 2 diff MS involved for this transactio to be successful, we can use 2 phc. First of all I should check with both service, are they aeady to place a order.. in 1st phase..

In 1st: ensure all participatory service are ready to make the change. Take a lock over corresponding DB, entry which will change if order is placed.. I wil take a lock on data point.. also a lock on order service where write no of unit.. count of FBA warehouse should be locked, so no one can play spoil sport here… which ever datapoint are gonna updated take a lock on them so no one can come and destroy for you.



Hey can you do this, also take a lock on data point..

If I wanna place a order, I check in inventory if I have at least 1 product is there… also I check if person is eligible to place a order, is a bad, blockesd order..

How a service knows another service: in logic in order service, it send sdownstream request to inventory .. order service knows about it…INV informed and take a lock.. order will take a lock internally… every service who are participating take sa lock which is required for this service.

Why lock: inventor count can be a 0 by time I process. Once I know I am ready for the transactio to happen.

**Explain again:**

User kevin has placed a order of 5 piece of a iphone, at the same time order service need to talk to inventory, ensure 5 piece kept aside for kevin. Also logistic service have to be told, these 5 have to delivered to kevin.. lets talk about ordr an d inventory only.. order need to ensure order should be placed, also if inventory size available.

If I need 2 corner seat, I also check if I have 2 seat available.

Once you get confirmation that lock is taken over data point.. 2nd phase says

2 phase says I don perfrom txn directly. But process involve in 2 steps.. 1st which service is participant each serv should take lock … I need to have a lock on no of ticket I booked… both of these service should take a lock on their resource, same way invenmtory service… PVR cinema… should block the 2 corner seat.. I prepare the txn by taking lock…

2nd step: once you have the info, once every participating service has taken the requsite lock we can finally commit the txn. Then I can initiate the txn commit…

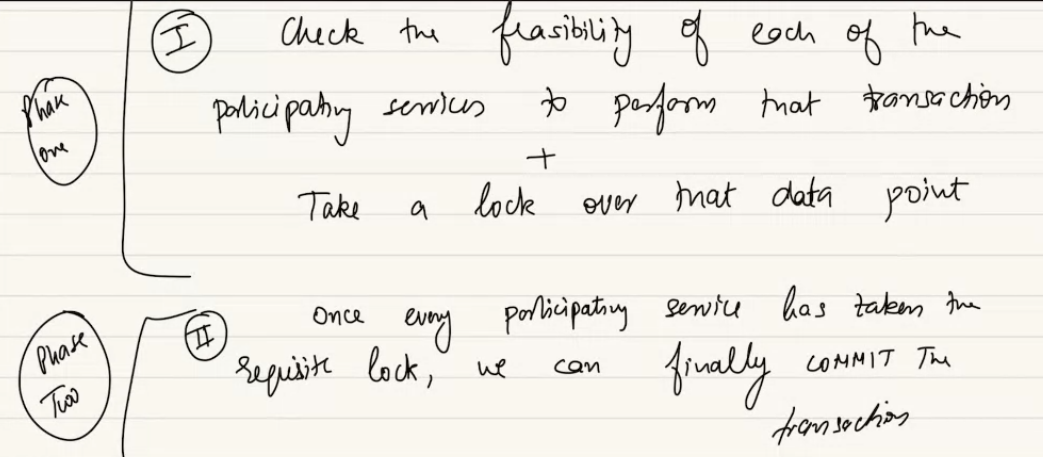
1st: take lock

2nd: you execute..

These are indenpendt service, if you directly commit the txn, probably one part can fail other part of txn can fail.. if u don’t take a lock first and directly perform the operation, some service an complete and other service cant complete the txn.. that’s why we take 2 phc commit not a single direct commit.

How 2 service can communicate that I have taken lock: 1 particular service who willinitaite this. 2 models people follow, service who is initiating the txn will be onbe who is orcharestrating.. did he taken lock ok. Then I can start commit..

Or 2PC txn manager, diff set of machine act as tXN manager, when 2PC requd, they act as arbitertator.. similar we can find in SAGA…



Lock on not an whole table, but you can take a lock on a element or row..

If there is one order and lock has been taken, what if another order comes in between? Fail as lock is taken by someone else..

If I cant do other half, I roll it back…

In order if I commit, and in inventory I could not.. order done 2nd phase, but inventory could not do 2nd phase. I retry and I rollback. And release the lock.. rollback after retries

Once txn is not completed, failed, it gets automatically rollback..retry measn if one part of txn fails the other part completes, we retry couple of times. 2-3 times, then may be I can initiate rollback.. that is when it releases the lock.. we can configure the retry.. 0-1-2-3- times… your won use case..

If 3 service we take lock on 3 service before going to 2nd phase.. in reali time system .. its difficult to use in scale. Not a first choice..

While writing MS, we ont have any logic, when order is communicate with inventory, it preservs. How it locks the table… is it code or a manager who does it? We can configure in called MS, when a txn need to do across service… distributes .. if A is orchrestred service, A calls other service as a preparetd step.. they have to extend , expose API which can make a prepare call… prepare for upcoming txn.. what resurce you need take a lock… orchraster also take s alock. Once bothe party is prepared then can initiate the 2nd phase…

If 2 user order same lamp.. the lock on 1 user or other user have to wait.. if I take lock on over the count.. we block on taking the lock. Ideally we should take lock on 5 instance of lamp.. sp other 95 instance is still bookable. Once txn is completed or fails lock is realeases. The instance is up for grap. Or they are sold..

Way -1 : crud aggregated value, take a lock so no one elase can take lock. Or we have diff entries or logical ways 95 is available.. 5 is under process. dding extra column locked quantity. The more better the better performance, the bigger lock, the bad performance.

2PC is 1st way of handling consistency on MS. This is anti-pattern. We advise against.. anti pattern, people commonly use but it is not best way. Advised way to go about it.

In 2pc commit you take a aggressive lock, you go and lock something. how long you take lock on.. if a txn tke 5 min, taking lock for long is not best for me. Can be stretch over 2-3-5 MS. Bad perforce. Overhead is high service is slow, degrade user experience.

2PC has adv: commit will happen once we are sure. red bus, BMS does 2 PC in backend. Makes service slower, UE compromised... in banking they use some part of 2PC.

Go to irctc video he shared.. does beautifully,

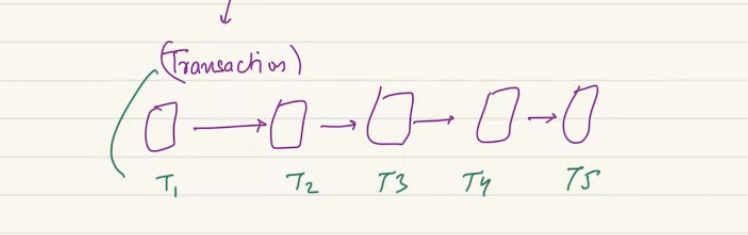
PRO: not even for a small time you have done something that you could not have done, you always books something which is definitely available. As you took aggressive lock. You will be in your limit.

3PC why not: these 2 phase make slow,, no ned to go and use 3 PC here..

***If we don’t want 2PC: SAGA Pattern:***

Saga means Story… beautiful tale, long tale a story..

Saga pattern says, assume a txn …. I need to runa txn across multiple services.. imagine that it can be broken down to smaller txn, t1, t2, ..t5.. smaller txn per service, each of them should be represent that part of the story for ..



If you break a txn to smaller, the idea of txn is lost.. as we cant commit a small part of it..

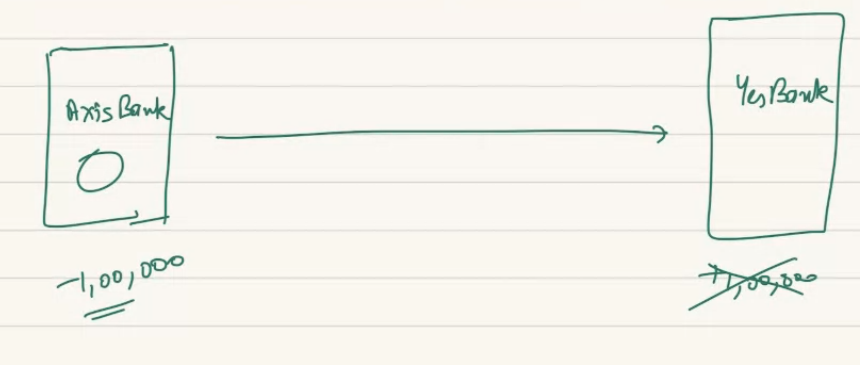
In Distributed system imagine your TXN also can be distributed. As a distributes txn each piece .. if something fails..

***Example:***

A axis bank suer makes a txn to yes bank… its not possible to runa 2 phase commit here.. take a lock on yes and axis tke a lock.. can be UPI involved.. or phonepe app.. so PP take a lock on PPE, ask axis to take a lock.. that might not be possible. Can be possible in one side.. ic an take a lock on my bank account balance when I pay 1 lakh. I can ensure that. I cant ensure other person is ready to take alock and then I go 2nd phase..

That’s why you don’t try and control every part of txn, you devide into smaller part of txn.. I can take lock on this service in txn manger. Once 1 akh is subtracted .. as a diff distributed manner I add 1 lakh to yes bank.. if it fails I should have a way to come back to my axis later. Reconciliation it calls. If deducted it should be compensante for..

So you let a txn to happen without taking lock,, if something fails in process,, the person who failed, whould have a way of triggering a rollback. This is saga pattern.. a pattrn of distriiby=uted txn..

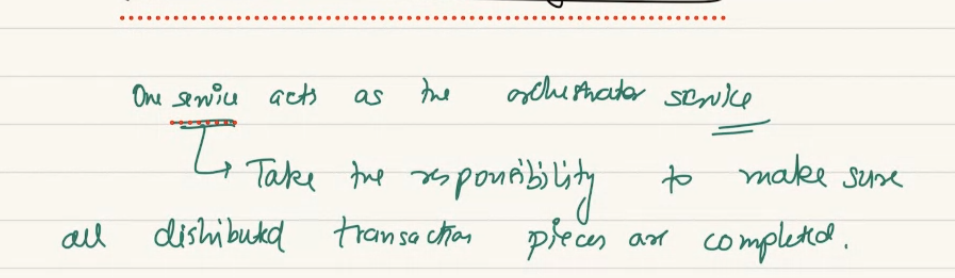


2 ways we can do.. orchestrated and choreograph:

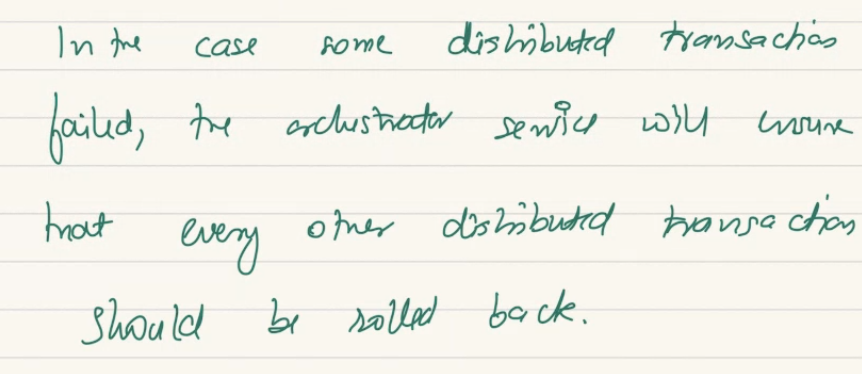
**Orchestration in SAGA:**

1 service acts a s the orchestrator service. Means this service is gonna take responsibility to make sure all distributed transaction pieces are completed…

If one step failed in between the OS will take responsibility to roll back..



In case of some distributed txn fail… the orchestrator service, will ensure that every other distributed txn should be rolled back..



In UPI payment eco system there is upi intermediatory in between like intermediate API.. NPCI is the org that manages or orchestrate the UPI for me.. I interact with NPCI and make my UPI payment.. so I am trying to make a payment to abhishek… the NPCI keep a track.a nd when failed NPCI send the req to may bank and says this pending and didn’t reach to recipient and please revoke, rollback the txn..

Intermediatory is responsible for rollback if something fails inbetween. This is orchestrator way of doing SAGA. Saga is happening in orchestration way… NPCI ensure when UPI txn prform the sender interacts with me. I lock this user try to pay to this bank..

I ensure recipant got the money, if not I start rollback. Until completely completed, it keeps active so can ask for rollback..

There would be 1 orchestration service who handle.. we can add machines…

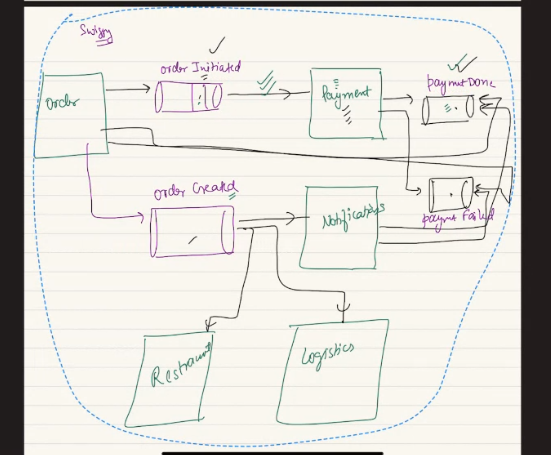
NPCI is orchestrator. NPCI charges bank when a payment happens. Also gets offset no of txn you receive. But NPCI gets fee, the higher load, more no of server to be added,and cost is shared by bank for free UPI service..

Is ONDC orchastrator for paytm and amazon or filpkart? Ondc is orchestrator for order delivery. Ondc is playtform for delivery. When I made payment the restaurant gets money and delivery food to me. Act as intermediatory.

Pls suggest some tools for HLD ,as like we have mermaid for LLD

<https://draw.io/>

Choreography:



In swiggy order, payment, notification, logistic, resturent service.. Simply writes to order initited queue, payment service reads from order initiate queue.

Can from same queue another service read same set of message? Yes. Idea is msg will stay fro TTL, many consumer group will have offset and go and read next message. Consumer group can reset the offset to reread.. if TTL not expired..

Ofr order initiated queue, payment is consumer .. paynet service wil initiate payment, it will process, eitwhe success or fail payment, if pass put in payment done queue. Notification service will read and order service will read, if payment done, OS will put in order created queue me message done.

After I put msg in order created queue, notification, logistic, resturent service will read from it. This is idea of event driven rchitecture.

Every service wil interact with MQ, not with other service. A kafka queue and topic gets the job here.

ADV: a programetic way…

No service is gonan worry about what other service is doing.. order don’t need to worry if payment service is up or not.. I don’t worry who reads what I am writing. So I have decoupled each service. No need to response to other calls.

In an event driven arch I can do districbuted txn and saga using Coreograpphy..

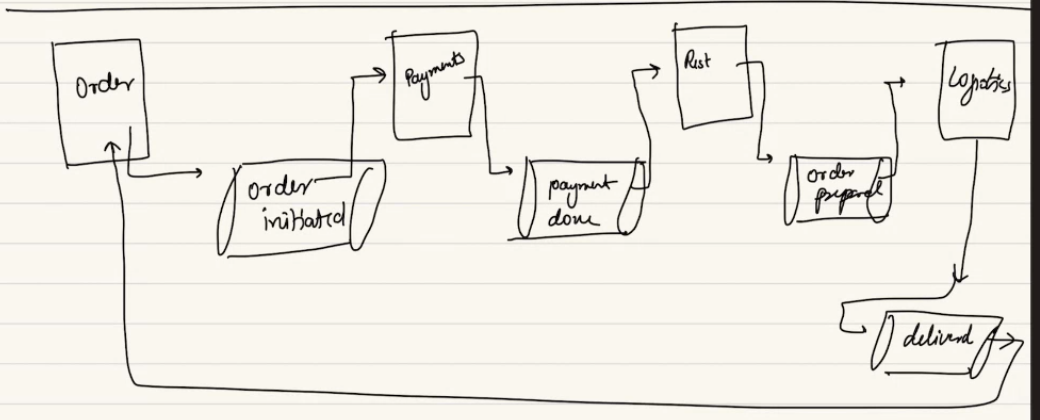
In evernt driven arch: when you have to perform a txn spread across multiple services in an event driven architecture.. we can happily.. use choreography..pattern of SAGA.

The idea is normal saga like you can do distributes txn, each sevice will sdo small txn. All txn will have a compensatory txn/ composentaory event.. using this idea you can easily use distributed txn in event driven arch.

Example:

Order service, once order is initiate I put a message in order initiate uee. Payment service read from order initiate queue. Once payment service read from it, I will process the pauymnet. If payment was successful..it will put the message in payment done queueu. Once payment is done.. next service is resturent service, as it has to prepare the order.. I insert in prder prepared queue, logistic service will read.

Once you deliver the ordere oin delivered queue. Order service will read from it and makrk order as completed.



We can add ore service.. like notificsaltion.

All of these should be aprt of 1 txn, I cant use 2pc.. assume txn done prepare food, and something fails. Delvery partner go back to resturent.. cant roll back..so 2pc is not good way..

So 1st part order service assume my job done and put in q.

Payment process an dput in pay done q.

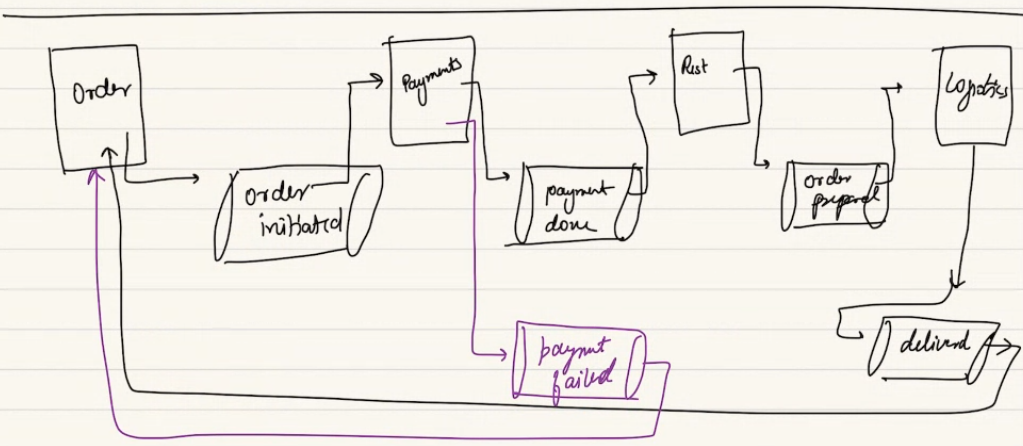
Resturent takes process and puts in r

Each of thm perfom inpedent txn and ot worry about anything else.

If something fails in between:

I need a way for everything happened in between undone.. so we have idea of compensatory.

For each of service I should have compensatory queue says payment failed. If a order which is initiate, if fails it write in payment fail queue.. once they write in payment fail, order service should read, when it ses payment fails, it should fail the order. Will not go deeper..



Before putting in payment failed. Payment service can retry fro no oftimes. Once no of retry exhauseted then it outs in compensatory queue.

If resturent was unable to prepare the order.. mayda.. they need to be a sad case scenario.. they cant prepare the order. And will have a different queue hey order not prepared.. once resturent puts event in compensatory queue, everyone behind that should read from it and undo.. payment reads and refund the money.. once done, they will let the prev service order service will roll back will do.

If delivery eats the food, he says my bike broke down and puts messge order not delivered.. when order delivery failed the message put in queue and prev

A->B->C  
C->B->A

**Which one is Better:**

Diff between orchestration vs choreography..

Choreography is realiable (more distributed, q in between, responsible for reading now or later).. Elegant, scalable.

Orchestration: SPOF, NPCI, if orchestrator failes its done. SPOF. Orchestrator can scalble.. but if you want your service to be fast, which one of them will make it fast for me.. the orchestrator is faster as I have a centralized orchestrator keep a track of everyting…

In case of choreography: reads later and so slightly slower..

Orchestraion: is faster…easier to reconcile, as centralized place me thinsg are kept. NPCI can centralized controller and can look at log , recored at single place…and debug easily.. debugging si easy.

Coreograohy: debug is tuff. As its distributed in nature. No one has a track of everyting that happends. Info is stored in chain. I need goddo observability required.. some solutions are …. So we can debug. Where its stauckk…..

for design choice, which one to pick when.. which scenario... what t pick..?

for txn in MS you will recall 3 discussion, 2PC route or saga router.

2PC is anti and only be picked when you have significant requirement.. happen or not..

Saga: prev is done and rolled back later… if a resturent has no maida to begin with, why did I even start the booking and taking to resturent… if I want a situation only be taken to a place when its available, I might not wanna say I will book that and ultimately I realize someone has booked that.. to implement BMS with SAGA, you take payment and at inventory service you realize its not available….

Saga handles by everything compensate and rollback… that’s nota god use case fo r a BMS example.. for BMS its bad UE. As they made payment asn endup seta is booked and I am refunding… you have to chooce. To go with 2PC or saga..

If SAGA, free flow evernt driven arch or central driven orchestration..

**CQRS: a concept of Microservice:**

Means …Command Query Responsibility Segregation.. you are doing some sort of segregating.. separating thing…

Lets look at reality of microservice,..

Service A has a own DB, and service B has DB.. service A can access its DB easily and vice versa. But Service A cant access DB of B, rather they need to make API call or GRPC call to other service, based on API B exposed….

I interact with API or grpc call..

In reality you have use case where you need much more details access to somebody else’s DB.

Example comes from analytics, system usually has good analytics to look at user request, to make logical claim and optimized.

If I have service in FB to message go and come.. another service handles all notfications.. if I have messages and all the notification come to me.. a user want to see all notification/ message he got.i will go to messages service and read all message there.. if a user wanna check how many notification I have.. so now I have 2 diff service, … I want to run some analysis…

How many users are there who has more message and less notification or vice versa.. it need infor from notification and from message..

2 service has each DB.. for some analysis or analytical job. For that I need to combine or join the data stored in service A with B.

Jon message data with user data and notification data with user data and combine them.. this becomes very difficult for MS. Or any query when I need to runa query on someone elses DB, as I can access based on API they exposed. They only Expose based on normal use case. Not expose PAI for join ot to give entire DB. Not expose API which are randoem in nature. But in such scenario when data is spead… then we follow CQRS..

When I do open query, merge db, I don’t do in MS as they don’t expose such API for me.. I should follow the idea of common service analytics DB.. whenre whatever data is written in message db which is reqd by analytics in async manger..

What data from notify you should put in async maner and out in analytic DB.. and this analytics DB acts a common playground for analytical Query..

In MS arch you need to copy data to a centralized place if you wanna run as many playfground query as needed..

MS rwesponsibile for normal q.

Open ended q use DB in async manner.

Who wants toi run on this they connect to common DB and runanalytics query on this…

It’s a SQL DB, it need not be join.. it was from another DB but I am copying to a common DB.. copy and set up pipeline in async manner.. copy in centralized….

so data is persisted regularly... or occasionaly?? Async manner..

***Circuit breaker:***

CB only required if you have sync calls between micro services..

***Problem statement***: CB is solution here

There is a user service, it interacts with messge service when he wants to send message.. MS interacts with a notification service which send a notification to a suer, interacts to email service.. also intercats with SMS service and android notification service these diff service talk to each other.. they are diff MS in case of FB.

They are sync calls: make a HTTP call or rest call or GRPC call.. sends request and expects a respose call..

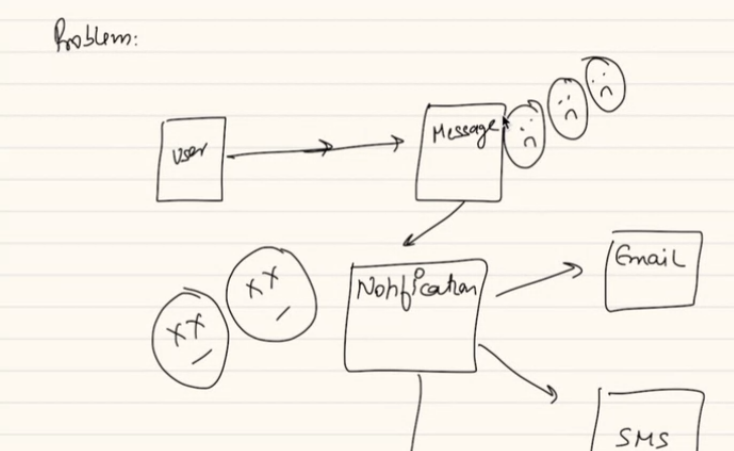
In event driven architecture.. decoupled.. async call, independent, fault tolerance..one service don’t depend on another.. in sync calls that a problem.. when user send a request to messages if the message service doesn’t come with ack. User service in wating,c nat close loop. Same way msg service asks notification please send this notic.. if notification says I could not I retry.. allthe responsible of retry and keep connection open.. all of these is on the services..

Service are responsible for keep connect open and retry.. so each service is depend on bunch of other svice,a nd not independent..

If notification is down mesg service is also feel the brunt. Conection hanging. Lot of resource used in bearing for brunt given by failure of notification and message can fail.. imcat can reach to user service.

Given in SYNC communication: services are not decoupled, failure at one can cause, issues, to cascade.. to other services.. problem of cascading failure… if you are conentced in electrical circuit any applies has problem. Other appliance will bear the brunt. As all of them are conencte din circuit..

With discussed cascading in CH class..



Problem of cascading failure.. the solution is circuit breaker.. if one device goes wwrog, other device cabn bear the brunt. Same as domino, replete effect..

MCB we have at home. To save everting.. a circuit breaker. Same idea is used here. If something goes wrong, other will wait for the service to come up so entire service don’t go down..

circuit breaker also called resilience

**Concept of Circuit breaker:**

Tehis soln says the service that call other services, should keep an track on no of error you are encountering. Error are happening. Erro fo 2 type. I keep only 1 type of error..

1. Regular failure/ error: things like user busy, user not found, bad request.. user not exist..
2. Non-Transient error: connection timeout, connection reset when lost, internal server error. RAM is out, CPU is OUT. Machine related errors..

Erro si a data/regular or machine error..

Keep a track of no f machine level or error, how many times you see app , machine is down.. keepa track of that… how manytimes you see, timeout, app not responding.. keep a track.. if you realize if no of machne erro you are getuing per minute, or hour.. if tahts beyond threshold, means system si down. The service you call is down.. it means yu should break circuit there.. tahts my circuit is OPEN.. I broken the circuit, not closed. Loop is open..

User ser vice realise that message service is down…so they shoud stop sending.. I don’t wanna talkto service which is down.. if user comes back. I say I am not available as I cant communictaie to downstream service.. so you open/break the service..

Break = open..

If everything is fine.. no of machine erro are less.. then I say my circui is closed circuit.. all good. I skekp sending message to downstream,..

My immediate predecessor wil open and their predecessor will open up the circuit when something down in downstream..

Once a circuit is broken: you don’t directly close it..

We go half open.. upstream service can send small % of traffic to check if they all good. If they don’t get response go backto abd case. Or if they get response its happy case.

Good case me go to closed circuit..

FaALLBACK: you can send, can configure when you need to.. when yu are handling circuit breaker and realize wrong in downstream,m. in some case you can send fallback.. thee is a service that tells no of user driver in some location.. the service is donw…

Can use CACHED data… fallback..tyo c diff response, if that circumstant satidfy your use case.

You open in tranisitory maner.. don’t jump to closed tstae. Go back to intermediatory state aka half open. Send small % of traffic. If fine then convert to close circuit. Or go back to open circuit..

Discover: ZK… whn lot of services..

Half open traffic control: confg you made for circuit breaker pattern that make sure if downstream system….

..

;;

..

**Are we not covering all the popular microservices design pattern or are they not asked in interviews like  
Service Registry  
Externalized Configuration  
Database per Service  
BulkHead**

All pattern(SAGA,2PC,Circuit) you explained today, do we need to implement while working in Microservice architecture or is there any framework giving all the implementation of the pattern.