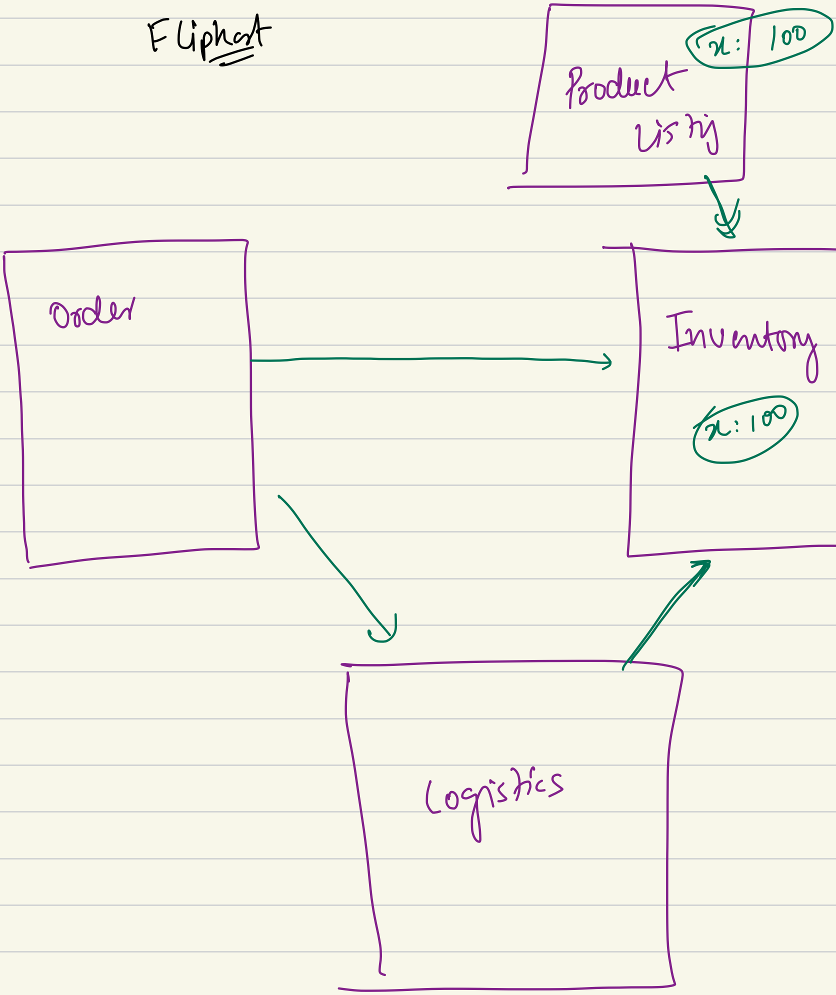
23|Jan|2024

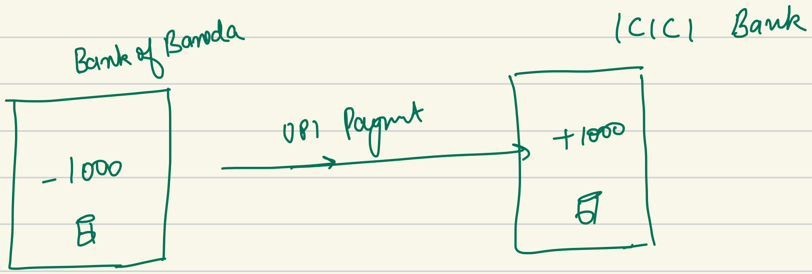
Microservices -3

Consistency in Microservices

Flipkart



Eg 2:



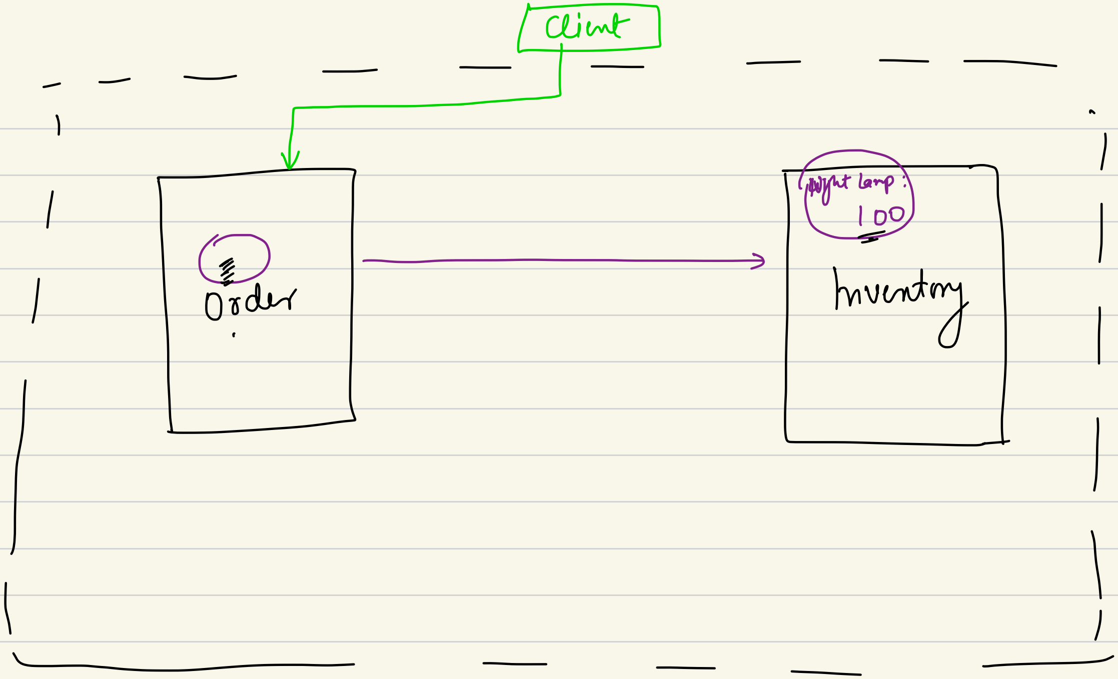
* * * How can we make our microservices handle data consistency better??

① Two Phase Commit / 2PC

② Distributed Transactions — SAGA Pattern

ⓐ orchestrated

ⓑ choreographed



TWO PHASE COMMIT

Phase one

Before

I Check the feasibility of each of the participating services to perform that transaction

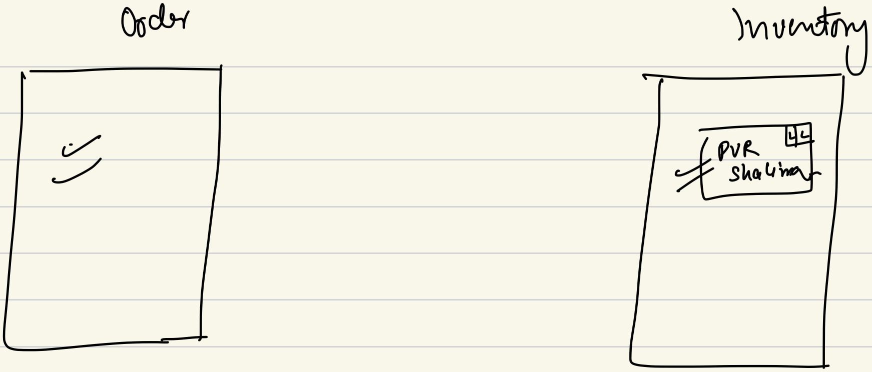
+

Take a lock over that data point

Phase Two

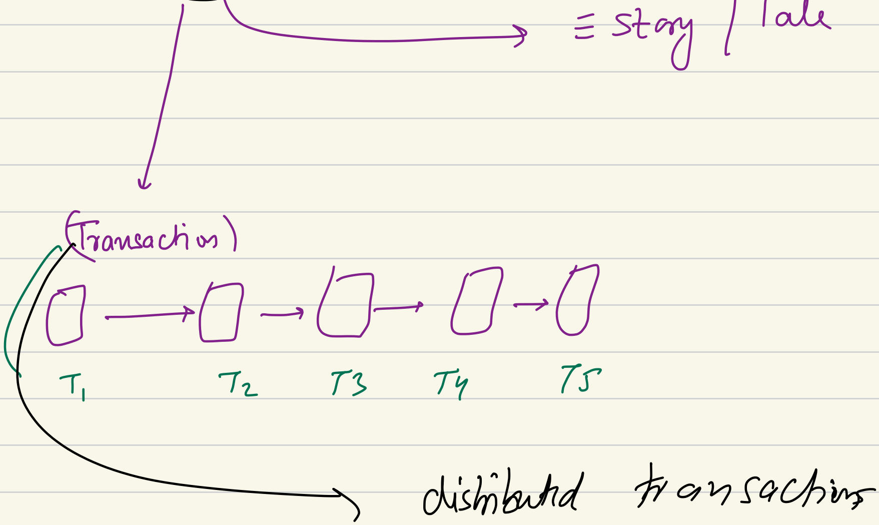
II Once every participating service has taken the requisite lock, we can finally COMMIT The transaction

commit

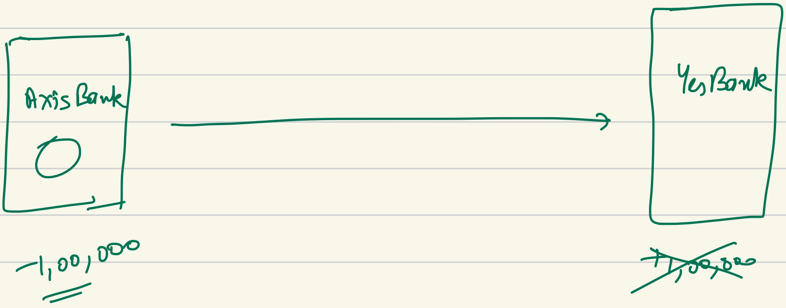


2PC is considered an anti-pattern

② SAGA Pattern ≡ story / Tale



Example:

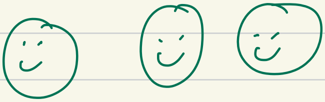


Orchestration in case of SAGA

One service acts as the orchestrator service

↳ Take the responsibility to make sure all distributed transaction pieces are completed.

In the case some distributed transactions failed, the orchestrator service will ensure that every other distributed transaction should be rolled back.

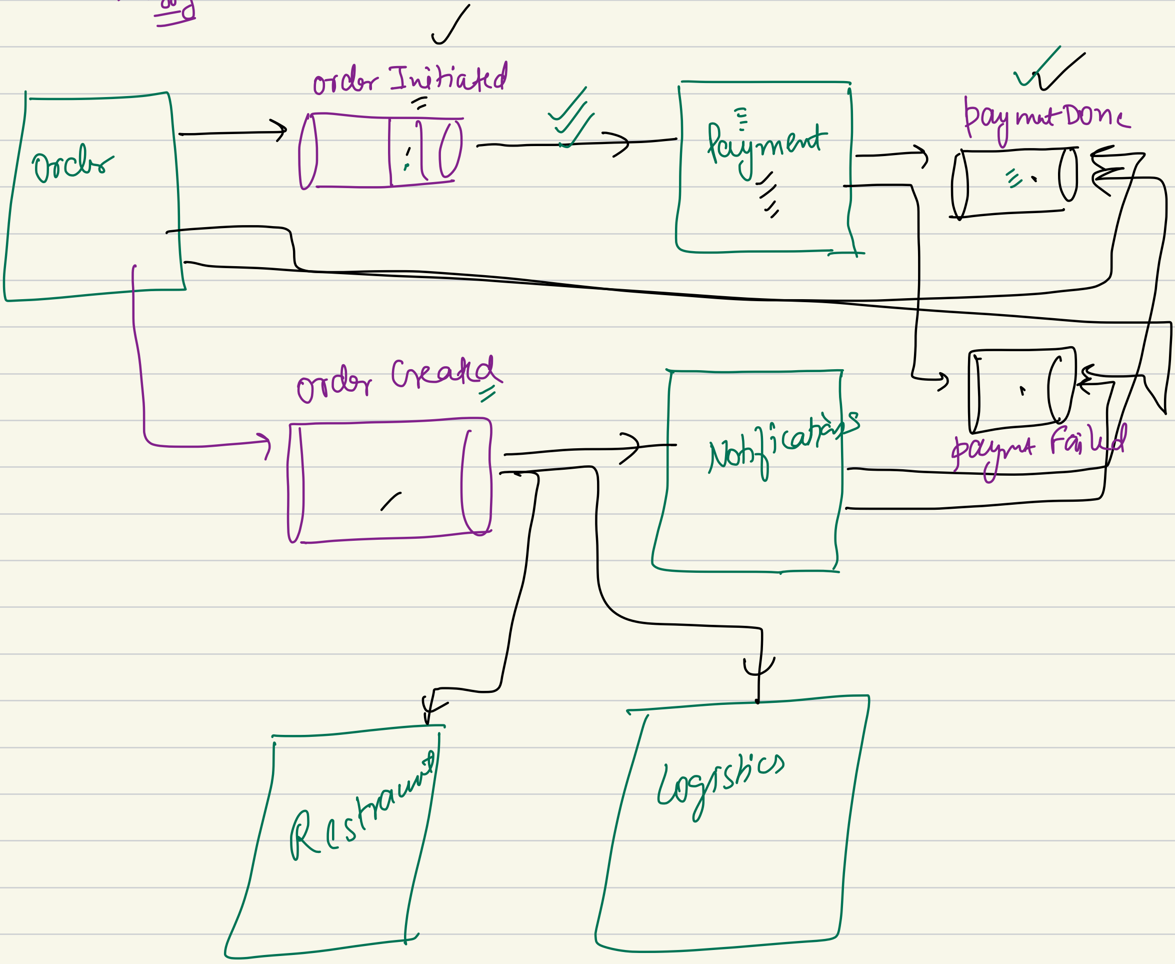


SAGA Pattern – Chorographed Way

→ Asynchronous Communication

Event Driven Architecture

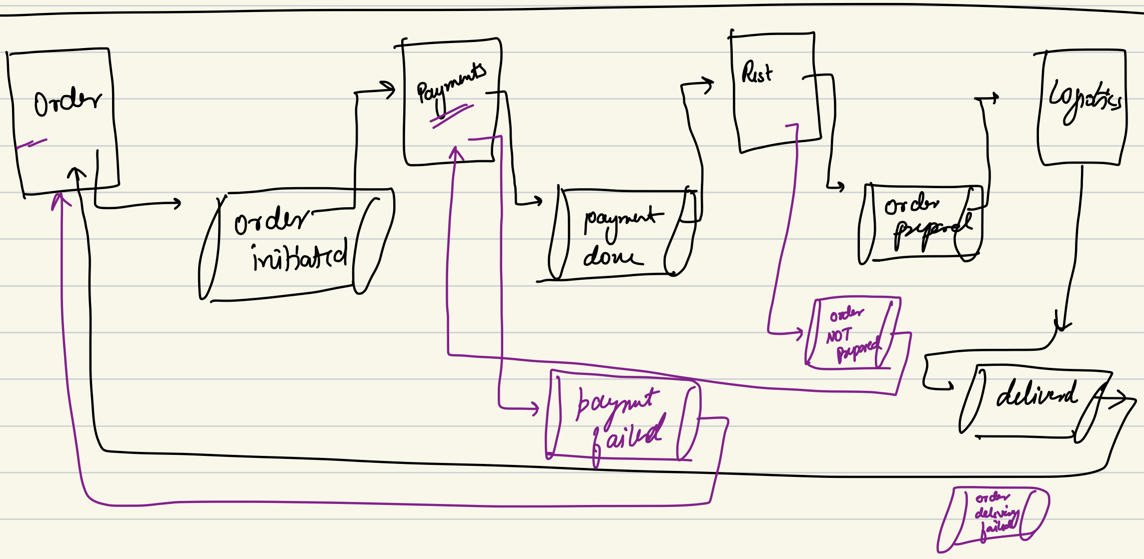
Swiggy


order Initiated
Order
Payment
paymnt Done
order Created
Notifications
paymnt Failed
Restraunt
Logistics

When you have to perform a Transaction Spread across multiple services in an event driven architecture;

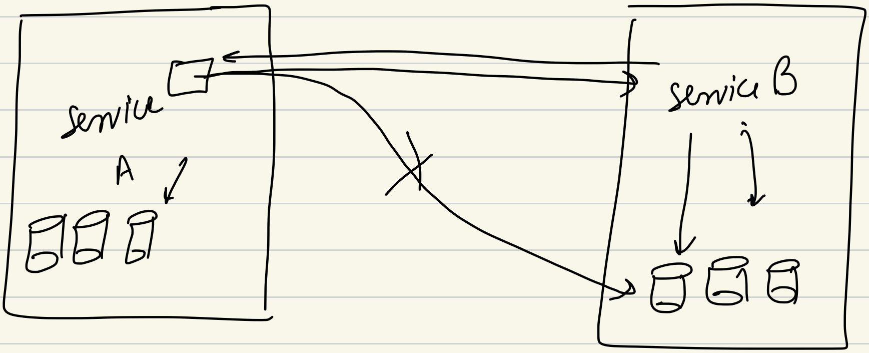
we can happily use Choreography pattern of SAGA

→ distributed transaction + Compensatory transaction / Compensatory event

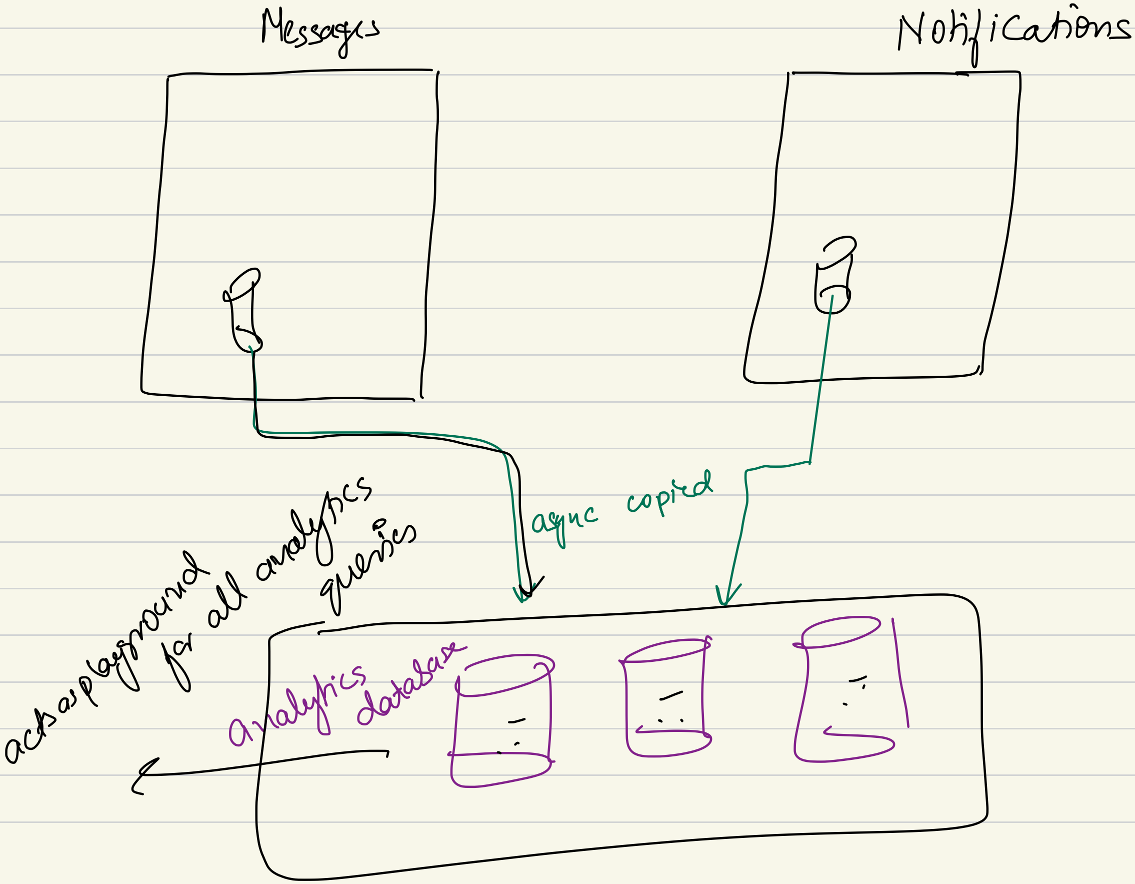


Orchestration	Choreography
SPOF ≡ orchestrator	reliable, elegant
scaleable	Scaleable
faster	slower
easier reconsile debugging	debugging is tough good observality solutions ☺ are required

CQRS ≡ Command query responsibility
Segregation


service A
service B

Analytics

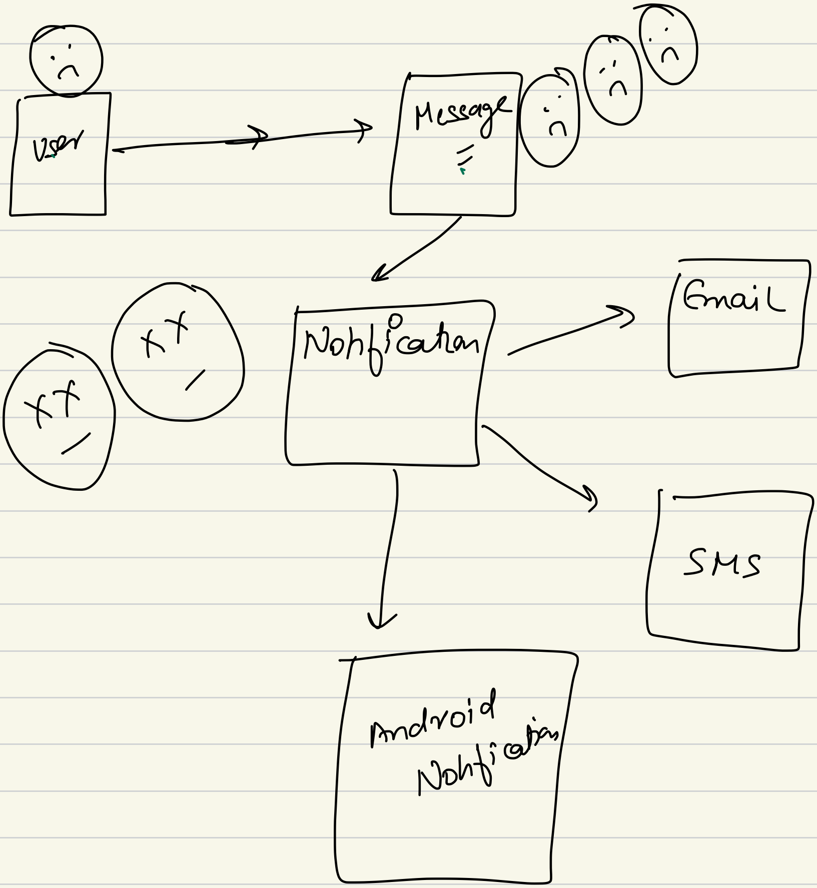

Messages
Notifications
async copied
analytics database
acts as playground for all analytics queries

CIRCUIT BREAKER

Solution

→ sync calls b/w microservices

Problem:



Given in sync communication, services are NOT decoupled,

failure at one can cause issues to cascade to other services 😕

Circuit Breaker

machine errors = Δ

error

Regular Error)Failure

user busy, user not found,

bad request

Machine Failures

Timeout, connection reset,

(machine issue)

I #machine errors / min > Threshold

★ open | Circuit Break

☹

↑ bad

II Half Open → small % of Traffic

↓ good

III #machine errors / min < Threshold

★ closed circuit ☺

Interviews

✓ Blue Green Deployment

✓ Canary Deployment

✓ A/B Depl =

ESB