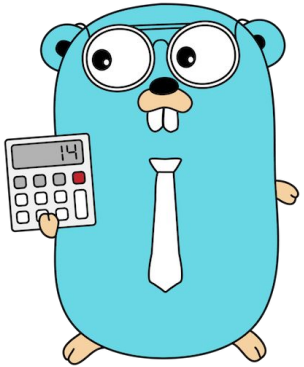
IN4331 Web-scale Data Management

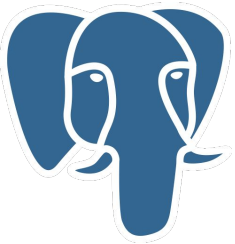
Group 7

Florentin Arsene
Miruna Bețianu
Pietro
Mozafar Shah
Pepijn Klop 4684311

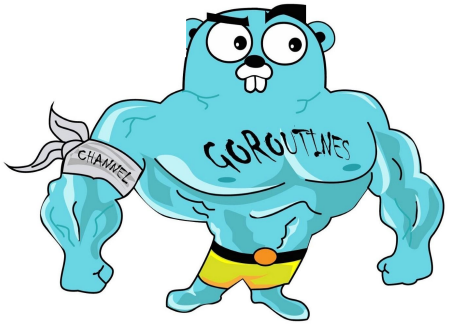
Tech Stack



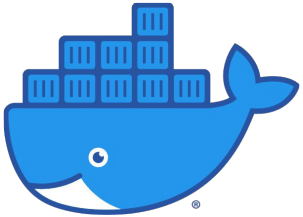
+ Fiber =

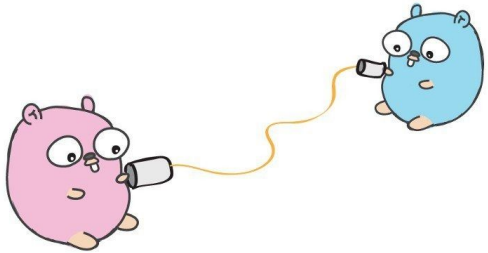


kubernetes

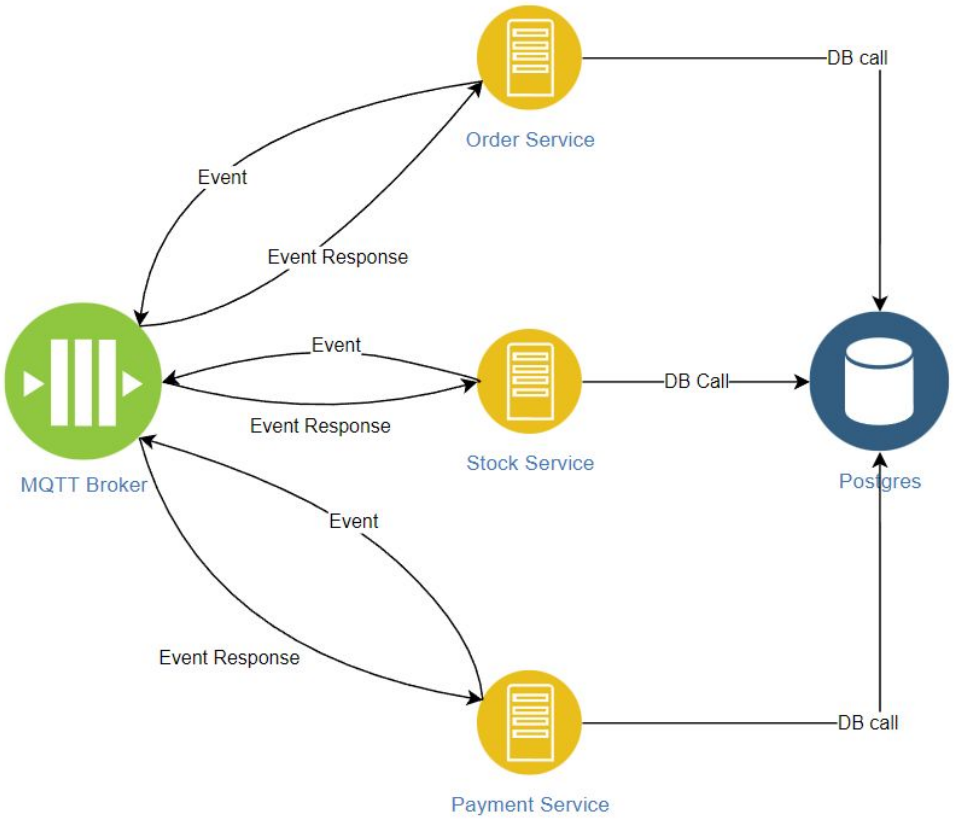


MQTT



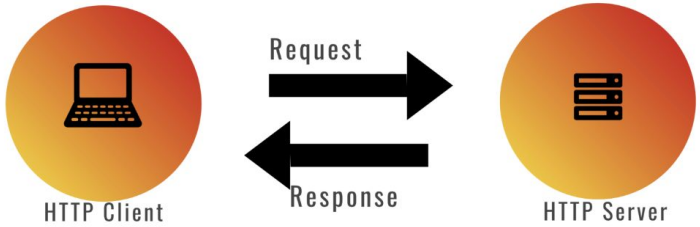


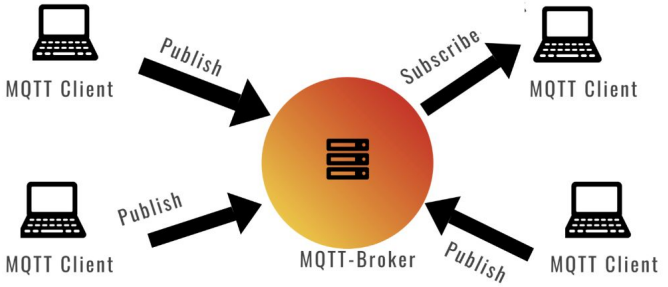
Overall Design Architecture Diagram



Orchestration VS Choreography

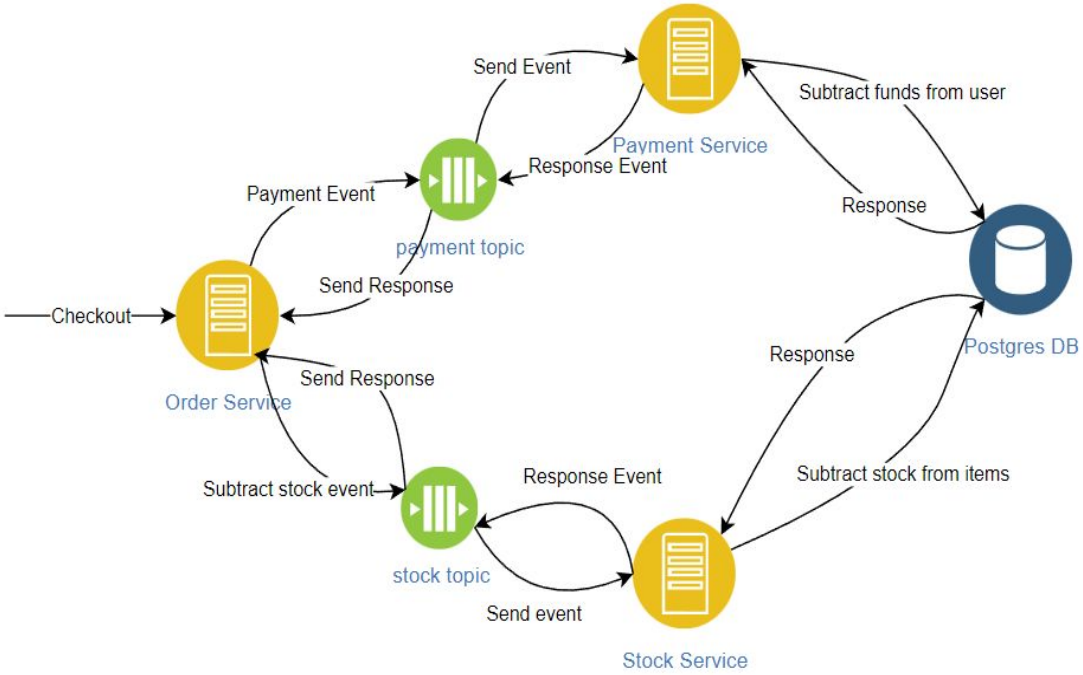
Why we switched from HTTP calls





Method of transaction execution

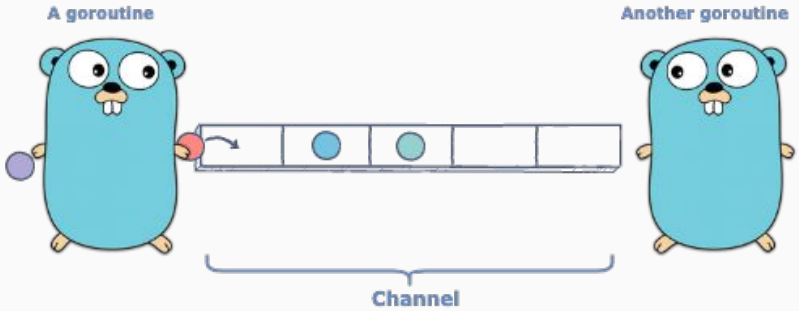
- Each transaction spans multiple services
- Transaction split up in local transactions
- Each service is responsible for guaranteeing local consistency
- Transaction is successful when all local transactions are successful.



Guarantees offered by transactional design

- Compliance with ACID principles
- Consistency of data across the services
- Concurrency

How? Go routines!



Method used for consistency

- Built-in methods:
  - PostgreSQL (synchronous commit)

- Implementation methods:
  - Message broker:
    - Messages
    - Publisher/Subscriber pattern
  - Single database

Method of scalability

Current implementation:

- Kubernetes
- Event based architecture

Bottlenecks:

- Limited resources of local machine
- No load balancing
- Single database

Method for fault-tolerance

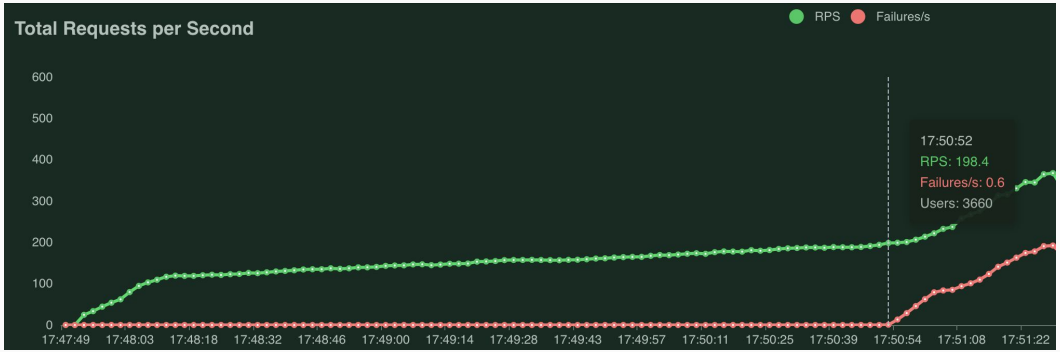
Database:
- Postgres fault-tolerant by design
- Transaction log for automatic crash recovery

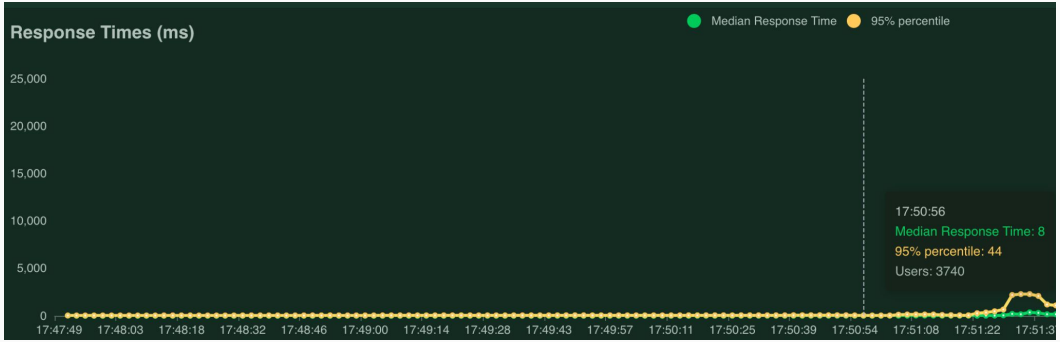
Service APIs:
- Error Detection
- Exception Handling

Further improvements:
- Synchronous Replication
- Data block checksums
- Backup mechanisms

Results

Max 200 RPS / 3600 users on a single CPU core
Median response time of 8ms on every endpoint





Conclusion

- Stable
- More than 200 RPS causes trouble
- Fault-tolerant
- Potential scalability and database improvements