Day 5 Kafka

Wednesday, 24 January 2024 6:03 PM

Banking → Transactions → Bank statement (This is more important)

Async

→ → Notifications → Mail

Uber → Invoice → Accounts → Mail

YouTube → watch, like, comment → upload

Amazon → Order placement (Instant) → Delivery system (Async)

These can be done delay in Async

Problem:

Low Delivery service is bombarded with more instant requests from OMS

Solution:

Queue

- Fast read requests and then respond on
- we need a replica of Queue in case first Queue dies

This is called Distributed Message Queue

Message Queue

→ Async communication (a bit slightly in processing is fine)

→ Load Balancing (By controlling the throughput)

→ Rate Limiting

(consumer can process at its own rate)

→ Decoupling components

→ Scaling (we can keep adding as if Mq Q)

→ Fault tolerance & Reliability (as we have backup)

This is decoupled as producer & consumer doesn't care about each other

Distributed Message Queue:

→ Availability

→ Mq persistence

→ Scalable

→ Geographical Distribution

Above is one to Many communication which is PubSub

Above is one to one communication which is Producer Consumer

Pub Sub
one to Many

PubSub: Kafka, RabbitMQ

Design a Distributed Message Queue

Topic

→ Encode
→ Log
→ Text
→ AVRO

→ Send Notification

how is those whether is Encoding or Notification message?

This is called Topics in kafka which is category of a message

Many servers in kafka is called Brokers

kafka is a cluster of servers

Topic → Encode
Parallelism
→ 1080
→ 720
→ 480

→ Notification
→ Order
→ Sub-org

To achieve parallelism, Topics are divided into Partitions. Partitions will be in different servers to achieve Parallelism

Msgs → Topic → Parallelism

⇓

Partition

To maintain Availability, we have Replicas in different brokers

Backups → Partition placed in another server

If main broker goes down, Then leader Election happens

RC

Categories ⟹ Topics

Servers ⟹ Brokers ⟹ Clusters

Topic ⟹ Partitions ⟹ (different servers)

⇓

Parallelism

Backup aka Replica

↳ different servers

Consumers consuming from same Topic is called Consumer Grouping.

cloudkarafka.com gives free tier kafka for learning

'key' hashes to a partition

'offset' is like an index number inside a partition

Msgs: [

{ key : , value : }

{ }

{ }

]

Topic } Mandatory
Value

Key
Partition } optional
Offset

Kafka usecase:

→ Notifications

→ Real time Data → Analytics
→ Personalized recommendation
→ Fraud detection

→ log Aggregator → ⟹ S3

Rabbit MQ:

Producer — Consumer

Routing keys

Exchange (Broker)

kafka always Persists messages but RabbitMQ needs some more settings for Persistance.

So Most Companies prefer kafka