Distributed Message Queue

xxx

March 2023

1 Step1

Understand the problem and establish the design scope
traditional MQ: does not retain messages once delivered
Delivery semantics:

- at most once
- at least once
- exact once

procedure:

- producer send msgs to MQ
- consumer consumes msgs from MQ
- msg can be repeated or once
- historical data can be truncated (do not lose it in 2 weeks)
- msg size kbs
- deliver in the same order
- delivery semantics

2 Step2

Propose High-Level Design
Msg model:

- Point-to-point:
  one msg into MQ and consumed by a single customer

- Publish-Subscribe:
  topics, partitions, brokers
  if writing in order, we can guarantee it received in order
  consumer group: one in the group receives one topic partition

High-Level Architecture:
Apache zookeeper

3 Step3

Design Deep Dive
data storage: read heavy and write heavy
no very suitable database: use write-ahead log (WAL) sequentially on disk
Batching:
can happen in producer, MQ, consumer
Cosumer flow:
How to get msg from MQ:

- push mode:
  MQ pushes to consumer w/o knowing if consumer has capacity
- pull mode:
  consumer actively read from MQ and add to offset

MQ replication:
in-sync replica (ISR):
a broker has several replicas to guarantee availability
producer send to one lead broker
this broker forward the msg to other replicas:
get ACK from its replica:

- ACK = all
  make sure all the replica has received the msg
- ACK = 1
  this 1 comes from the lead, meaning producer does not monitor how the lead forwards to replicas
- ACK = 0
  producer sends msg, and does not wait for MQ to respond

Delivery semantics:

- at-most once

  - producer send to MQ once, and deletes it
  - consumer fetch from MQ, and send response to MQ before consumer processes data. So even if consumer process fails, it cannot fetch the same msg from MQ again.

- at-least once
  - producer sends msg to MQ and wait for AKC ¿ 0
  - consumer fetches msg and process the data. After processing success, send response to MQ to move offset.

- exactly once