Pulsar在滴滴的实践与探索

李松松 2024-09

CONTENTS

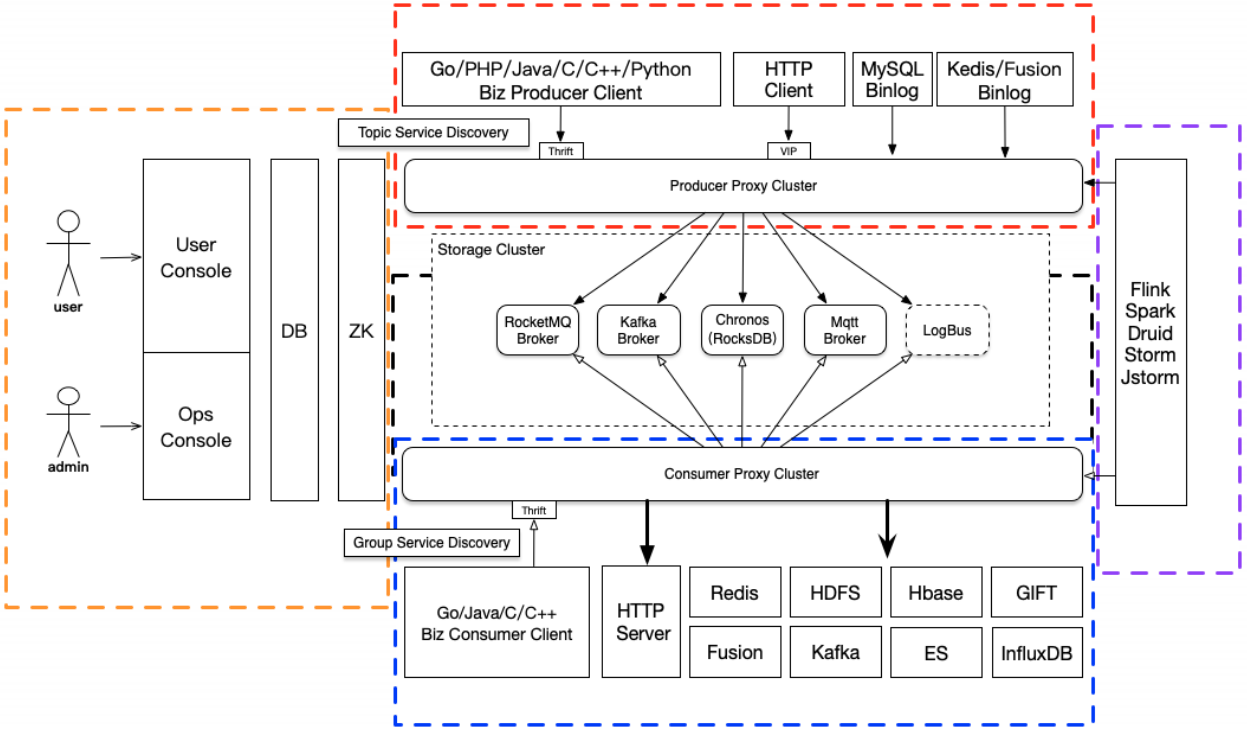
引入Pulsar的背景和初衷

DDMQ简介

- DDMQ消息队列系统
  - Kafka-->RocketMQ-->Pulsar
- 特点
  - 功能：毫秒级实时、秒级延迟消息等
  - 稳定性：可用性99.996%+
  - 业务场景：网约车、顺风车、两轮车、能源、金融等业务
  - 集群规模：
    - 万级别的Topic/Sub数量
    - 千万级QPS峰值
    - 万亿级别的消息流转量/天
    - 千级机器数量、几十个集群

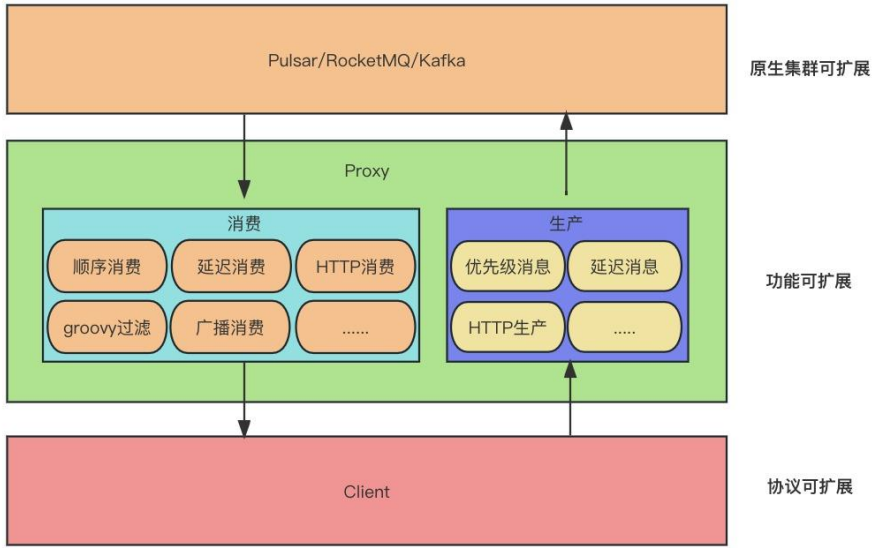
DDMQ架构

- 五大模块
  - 用户运维控制台
  - PProxy+生产Client
  - 存储引擎
  - CProxy+消费Client
  - 大数据流式Connector



DDMQ proxy优势

- 提升系统可扩展性
  - 原生集群可扩展：兼容不同消息队列如Kafka、Rmq、Pulsar
  - 功能独立性：集成一系列特性
  - 协议灵活性：定制客户端与服务端之间的通信协议，协议转换
  - 多语言客户端兼容性：thrift实现私有协议，多语言兼容，易维护



DDMQ proxy优势

- 提升系统的可维护性
  - 迁移Topic所属集群(kafka->rocketMq->Pulsar->…)
  - 服务端参数控制，提高系统灵活性
  - 降低客户端复杂性，降低客户端升级频率

引入Pulsar的初衷

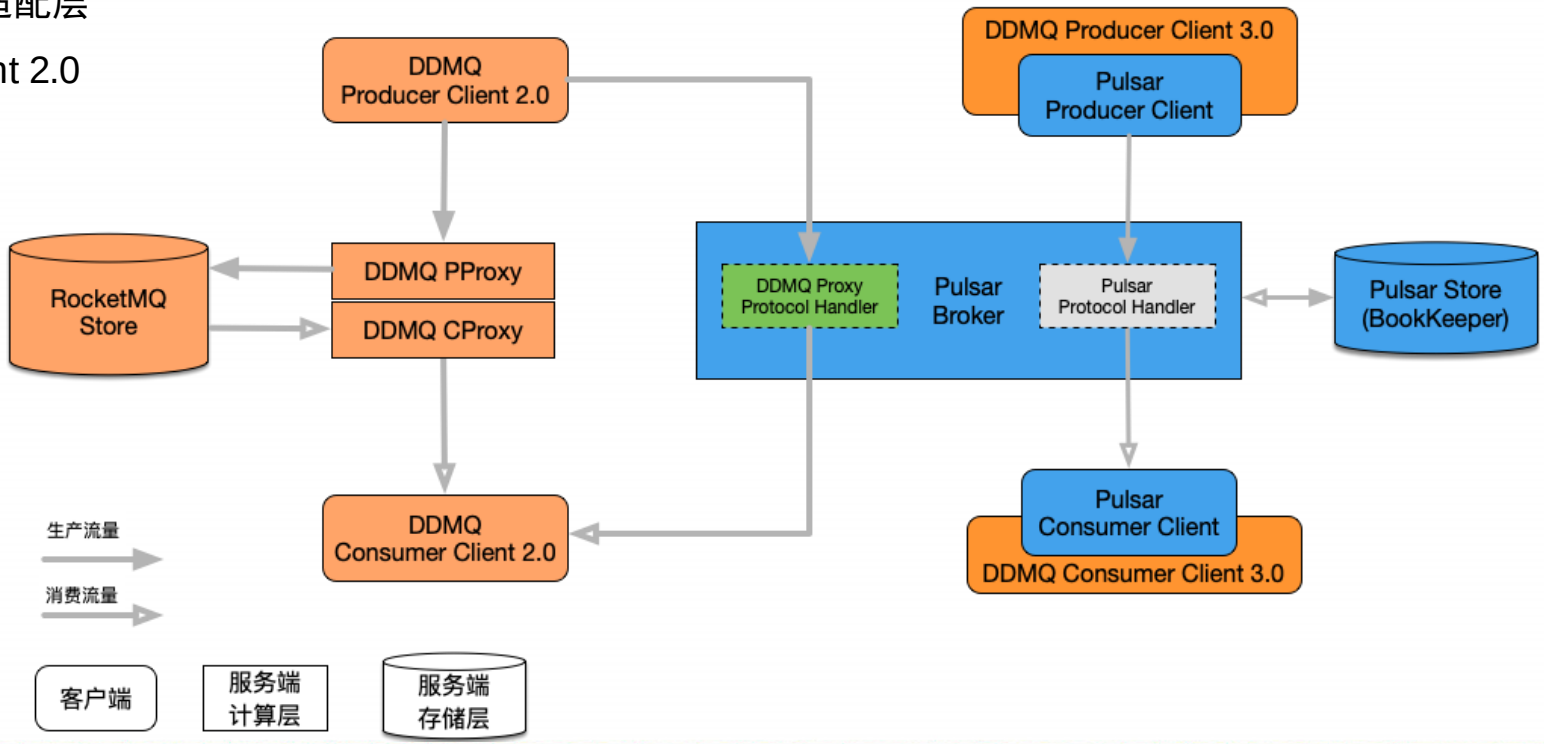
- 云原生架构升级-存算分离
  - Broker无状态，动态负载均衡，快速伸缩的支持更友好
  - Bookie无主从结构，副本并发写，单点IO抖动时用户感知到的延迟影响更小
  - 减少文件系统依赖，PageCache的依赖更小，回溯场景友好
- 运行成本降低
  - Cpu开销更低。相同负载下，Pulsar的CPU使用量是DDMQ的30%~50%
  - 写入延迟毛刺小。高负载下，Pular的99分位延迟是DDMQ的30%~40%
  - 端到端延迟小。高负载下，Pular的99分位延迟是DDMQ的20%
- 客户端
  - 多语言、高性能、富客户端
- 业务长尾需求
  - 社区合作、引入更多的可能性

DDMQ如何融合Pulsar

DDMQ on Pulsar

架构融合关键点

- Proxy协议适配层
- DDMQ client 2.0



DDMQ on Pulsar

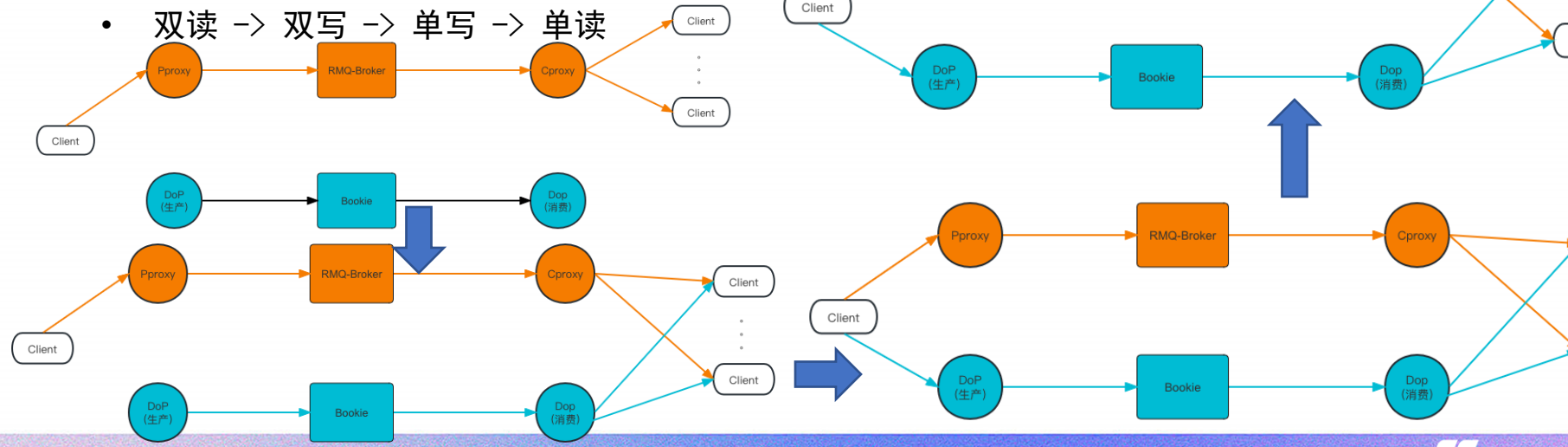
线上流量切流

目标：

- 稳定性高：毫秒级抖动，业务感知小
- 适配成本低：业务无需改造

方案：

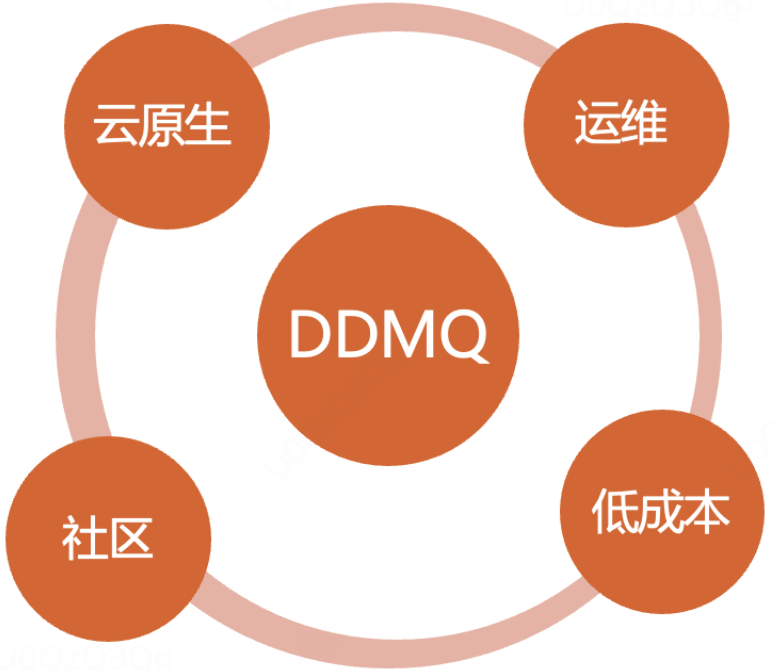
- 双读 -> 双写 -> 单写 -> 单读



DDMQ on Pulsar

现状

- 云原生化架构升级
- 运行成本降低
- 运维人效提升
- 自动运维平台搭建
- 承载超过50%的线上流量



DDMQ应用Pulsar: 痛点和解决方案

存储层

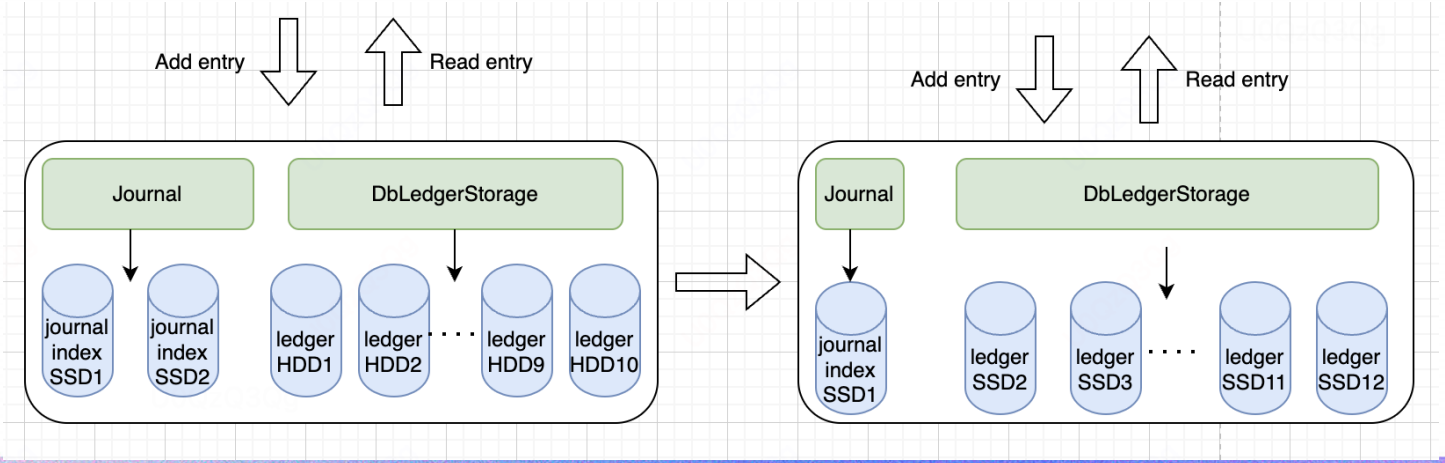
原有存储架构痛点

- 单物理机单bookie下，当前吞吐和保存时间下，

机器存储利用率低

- HDD随机读时，单机吞吐和读写延迟表现不稳定

目标：高吞吐低读写延迟



元数据层

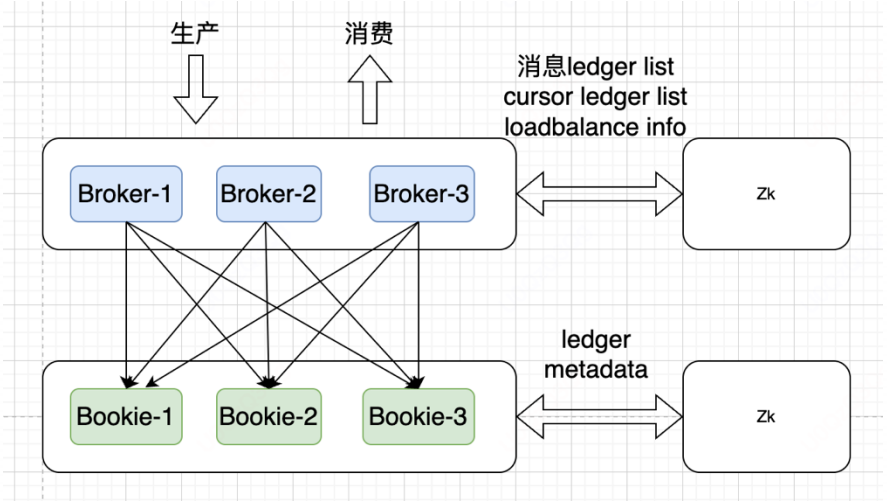
zk弱依赖

背景：

- DDMQ使用pulsar 2.8.2版本，强依赖zk
- zk切主后无法响应或zk断连后未及时重连，造成session expired，触发shutdown策略
- zk抖动或宕机后，导致生产失败

方案：

- Broker或bookie Zk断连后不滚动数据ledger
- Broker或bookie Zk断连后不切换cursor ledger
- Broker或bookie Zk断连后不裁剪分区的ledger列表
- Broker或bookie Zk断连后停止动态负载均衡

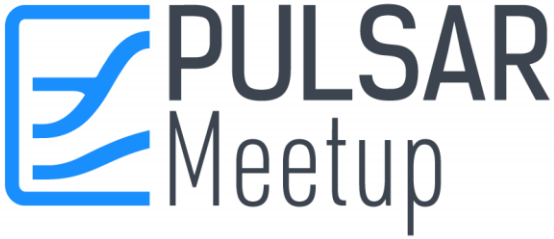


未来规划

未来规划

- 升级至新的稳定版本
- 完善bookie下线方案
- 拥抱Pulsar原生特性(offload，延迟消息等)
- 积极融入Pulsar周边生态(connector的使用等)

加入我们



Thanks