

The SACC logo is rendered in a bold, white, sans-serif font with a blue glow effect. It is positioned in the upper right quadrant of the image, above the main title. The background features a blue wireframe architectural structure with a perspective view, and a large, faint '2021' is visible in the bottom left corner.

SACC

2021 中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2021

数字转型 架构重塑

IT168.com

ChinaUnix

ITPUB

云上会议 网络直播 | 2021.5.20-2021.5.22

MQ平台在VIPKID的架构演进实践

VIPKID 基础架构存储平台

主讲人

石 鹏

VIPKID存储平台团队负责人

10+年专注于分布式存储、高并发、高可用，曾就职于摩托罗拉、爱奇艺

2019年5月加入VIPKID，目前负责VIPKID存储中间件平台，包括Redis平台、Kafka平台、RocketMQ平台、ElasticSearch搜索平台、数据库访问平台、对象存储平台等



目录

Content

01

平台简介

RocketMQ：业务类
Kafka：日志类

02

MQ平台一期方案

做了哪些Features
存在哪些不足

03

MQ平台二期方案

加了Proxy,怎样解决MQ消息丢失、重复，保证顺序
Proxy层怎样保证高可用
基于Proxy增强MQ平台原生能力

04

收益总结

避免了多语言客户端迭代的风险和人工成本
流量调度和容灾自动化

01

MQ平台简介

- ▶ RocketMQ : 业务类场景
- ▶ Kafka : 日志类场景

02


MQ平台一期方案

- ▶ 我们做了什么
- ▶ 存在哪些不足



Feature1：封装Java版RocketMQ客户端

- 可动态更新客户端配置
- 避免用户自维护客户端稳定性风险
- 监控数据上报

Feature2 : 提供RocketMQ的管理后台



在线少儿英语

 石鹏
 

[VKMQ接入](#)
[接入流程](#)
[2.2.1 Spring接入](#)
[2.2.1 Java接入](#)
[消费不到消息怎么办?](#)
[报警我该怎么处理?](#)
[启动获取不到配置怎么办?](#)
[更多FAQ](#)
[我想轻轻告诉你: 你们](#)

应用管理

VKMQ管理

品 topic管理

生产者管理

消费者管理

消息管理

消息轨迹

报表

报警管理

文档

VKMQ文档

VKMQ知了社区

* topic

topic名称


查询

+ 新增 topic


[怎么授权别人访问这个topic?](#)
[无法消费?](#)


详情	topic	owner	操作
>	vkmq-topic	nodata - root	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	eim-fin-contract-topic	nodata - liwei5	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	room-line-doc-event-default-topic	nodata - liwei5	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	course-account-hour-consume-topic	nodata - root	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	order-center-deposit-refund-change	nodata - root	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	course-account-finish-type-topic	nodata - root	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>
>	order-center-deposit-confirm	nodata - root	<div>我要生产消息</div> <div>我要消费topic</div> <div>堆积详情</div> <div>发消息</div> <div>报警</div>

Feature3：提供Kafka的管理后台



在线少儿英语



 石鹏(存储平台组,管理员)

Dashboard

集群管理

接入与审批

用户管理

审计日志

系统管理

配置管理

应用管理

Proxy服务管理

报警规则

报警模板

资源组管理

集群列表

Topic列表

我的Topic

添加集群

#	集群名称	所属端	资源组数量	Broker数量	Topic数量	消费者组数量	集群版本	创建时间	操作
1	arch-vkcat	基础架构	1	5	26	9	2.0	2020-06-11 12:16:58	编辑 删除
2	arch-g1	基础架构	1	3	88	34	2.0	2020-06-11 14:29:55	编辑 删除
3	databus-ten-wyj	基础架构	1	3	45	40	1.1.0	2020-06-11 14:39:59	编辑 删除
4	big-data-ali-kafka	大数据	1	3	576	148	1.1.0	2020-06-11 15:45:10	编辑 删除
5	databus-prt-ali	家长端	1	3	15	2	1.1.0	2020-06-11 15:50:04	编辑 删除
6	online-class-qos-kafka-biz	在线教室	1	5	377	107	2.0	2020-06-11 15:56:36	编辑 删除
7	elk-cal-ali	基础架构	1	3	96	28	0.10.0	2020-06-15 14:56:13	编辑 删除

Feature4：提供统一的监控告警

- 客户端上报监控信息到监控平台
- 收集Broker监控信息到监控平台
- 统一对接告警平台做IM、邮件、短信、电话等告警

问题1：怎样提供MQ多语言客户端支持

- 同时维护多语言MQ客户端：升级成本高、风险大

思考：

- 怎样以最小的代价提供MQ多语言客户端？

问题2：客户端升级需多业务配合，代价高

- 客户端更新需要所有业务应用升级

思考：

- 怎样规避业务端一起升级带来的代价？

问题3：由于历史原因多个Kafka版本难统一

- 不同业务应用使用Kafka版本不同，怎样统一？

思考：

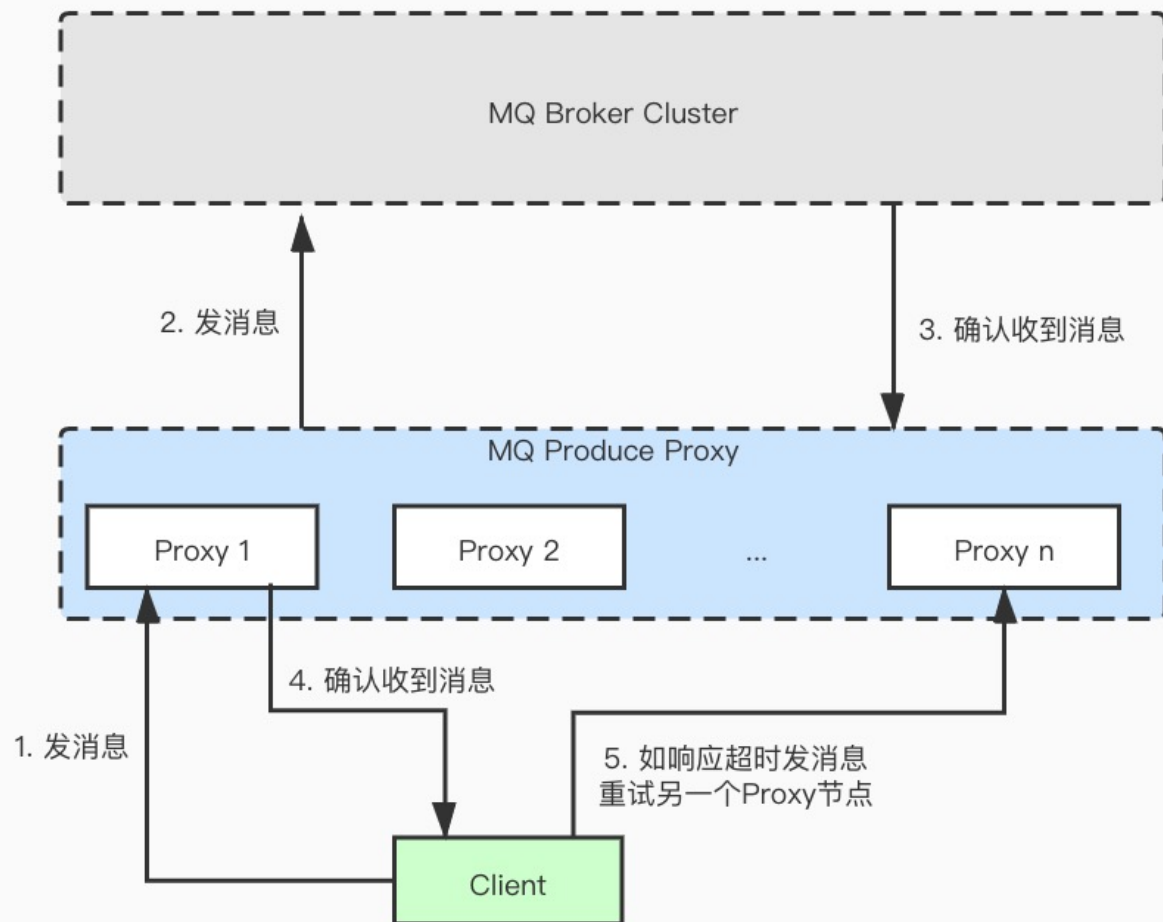
- 怎样统一Kafka版本且无稳定性风险？

03

MQ平台二期方案

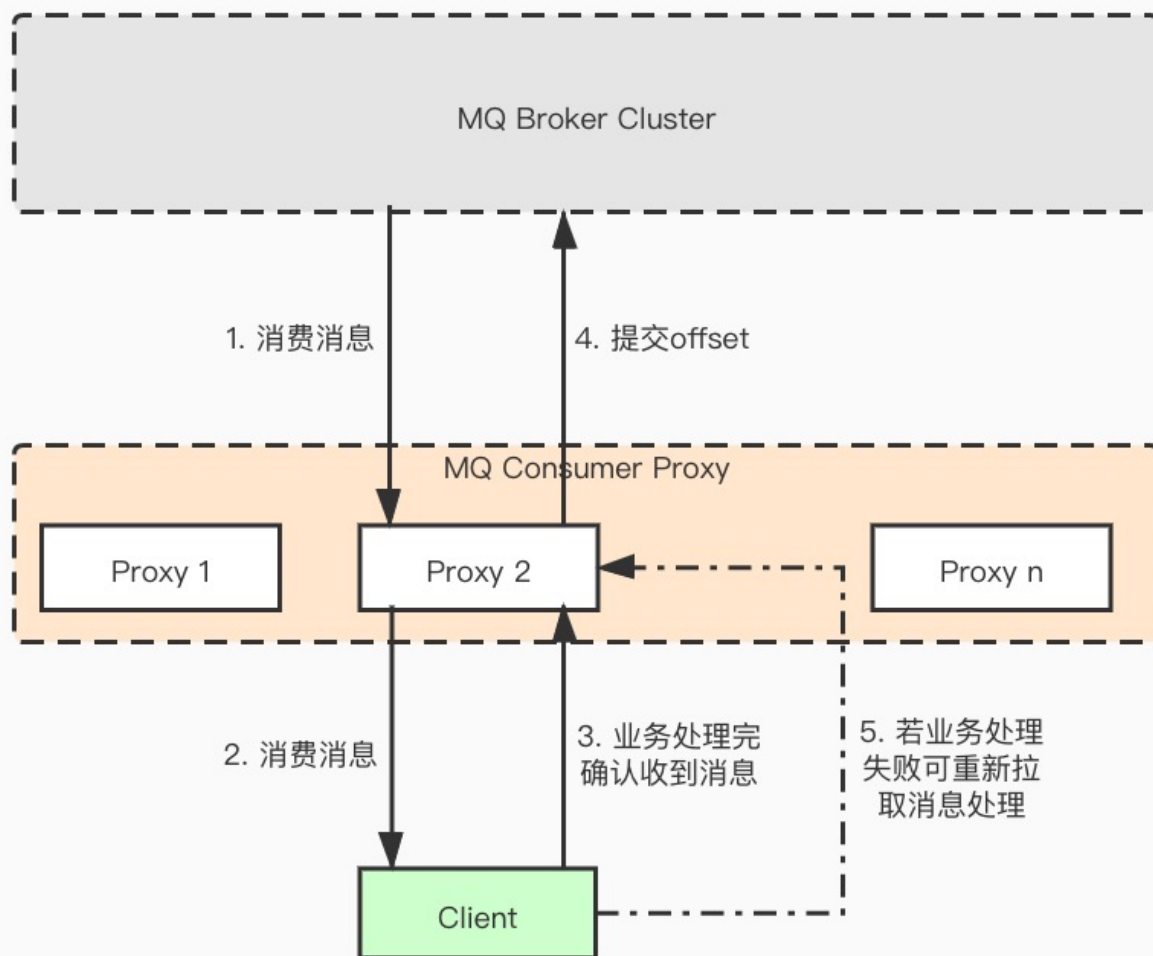
- ▶ 基于Proxy重构MQ平台的架构
- ▶ 加了Proxy后如何避免消息丢失、重复，保证顺序
- ▶ Proxy层的高可用方案
- ▶ 基于Proxy怎样做到MQ集群容灾自动化智能化
- ▶ 基于Proxy提供MQ数据自动化迁移工具
- ▶ 基于Proxy增强Kafka消息过滤能力

加了PProxy如何避免Kafka消息丢失



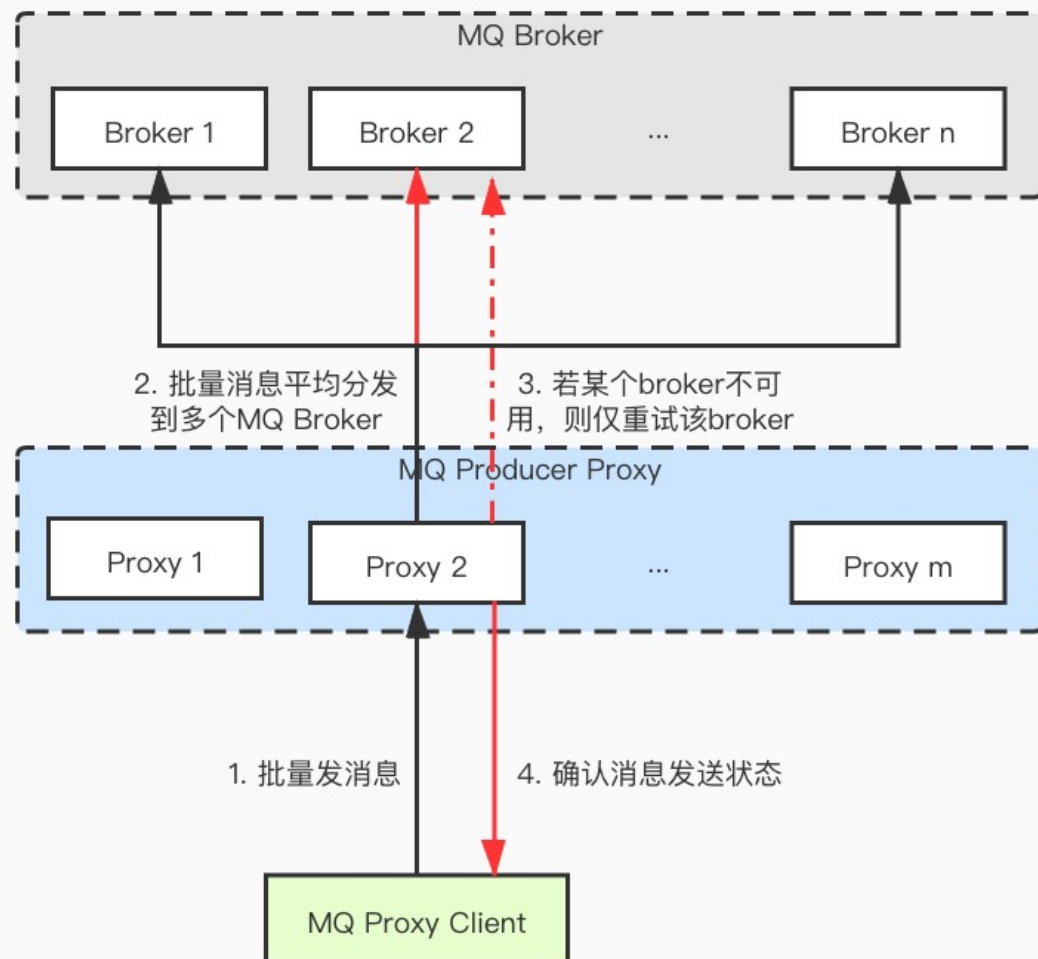
- 收到确认才算发成功
- 若超时则换Proxy重试
- 会重不会丢

加了CProxy如何避免Kafka消息丢失



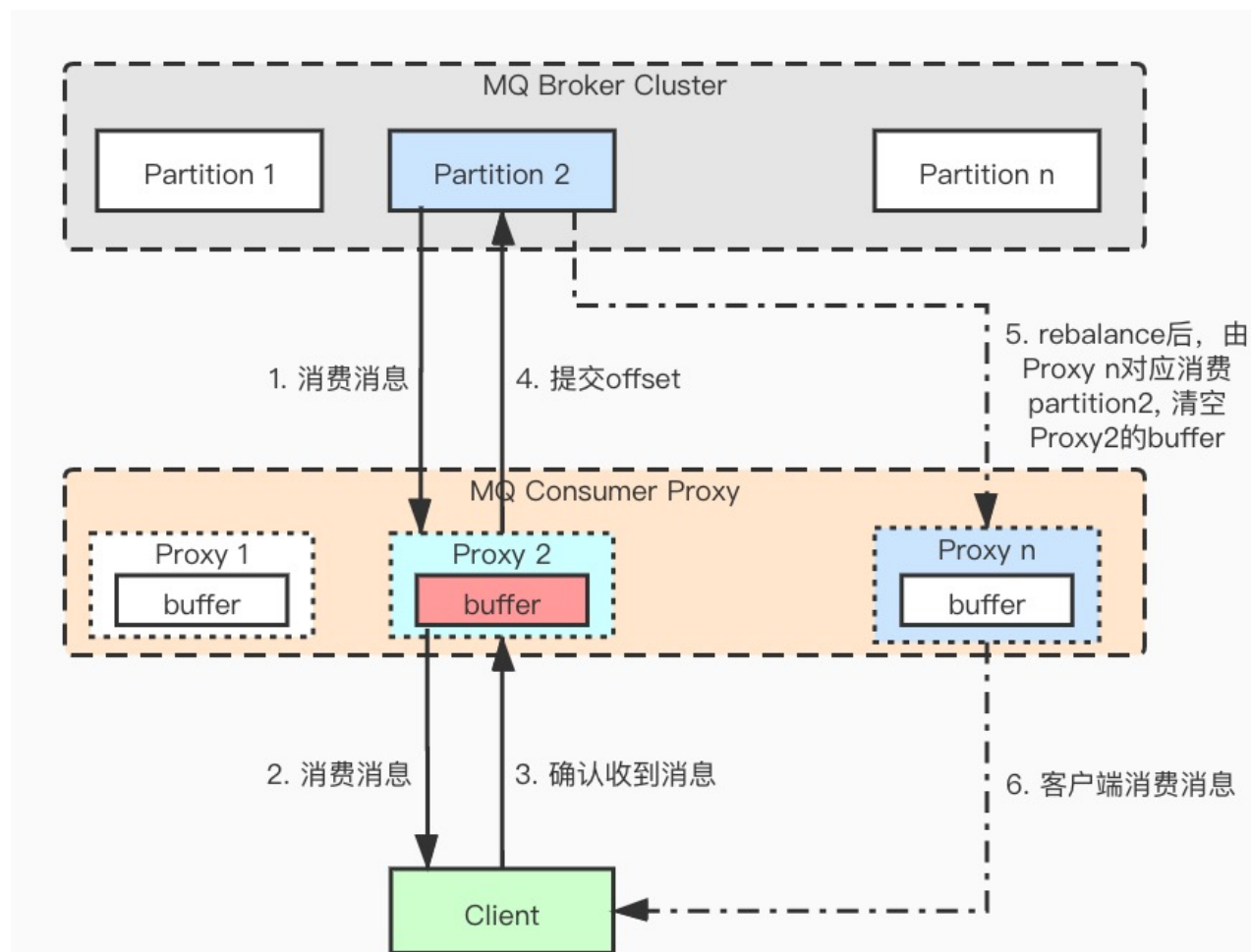
- 向broker提交offset后才算消费成功
- 业务处理失败可从CProxy上重新拉取

加了PProxy怎样避免消息重复



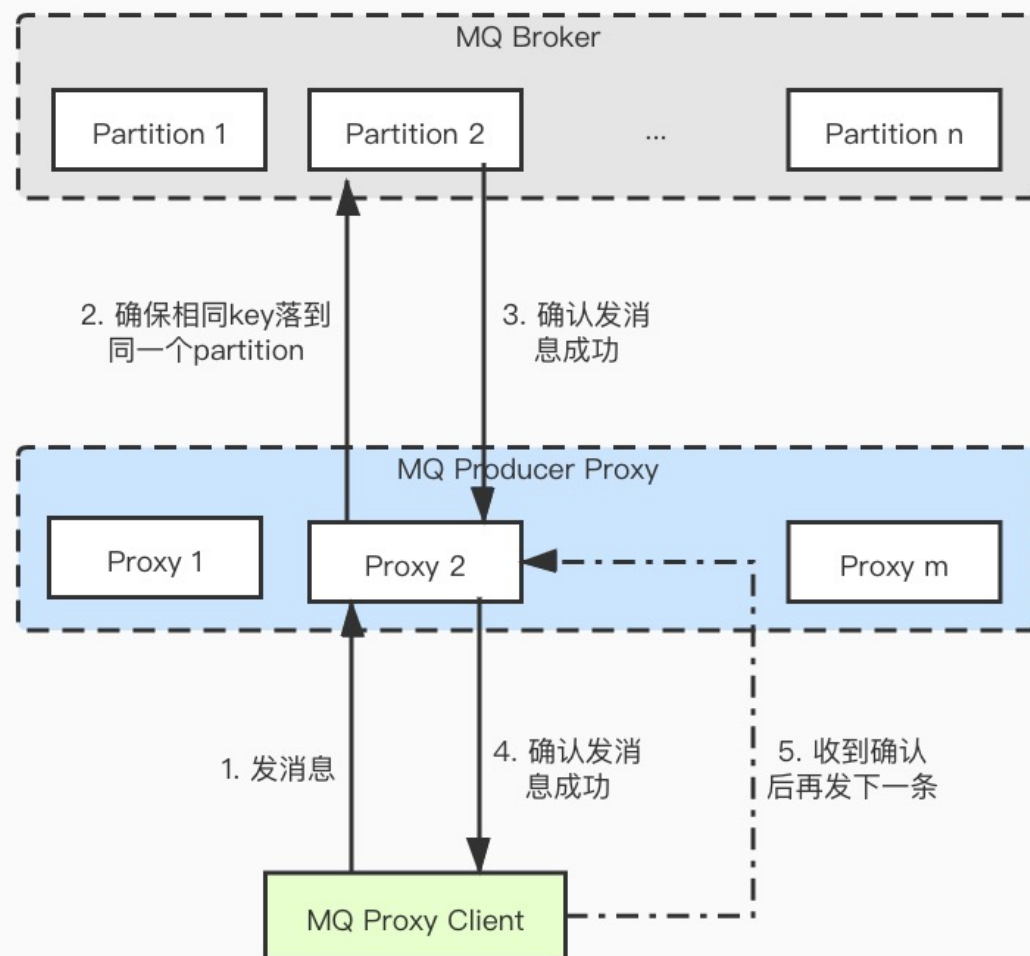
- 发批量消息：平均负载到每个broker
- 向单broker发送失败时仅重试该broker对应的消息

加了CProxy怎样避免消息重复



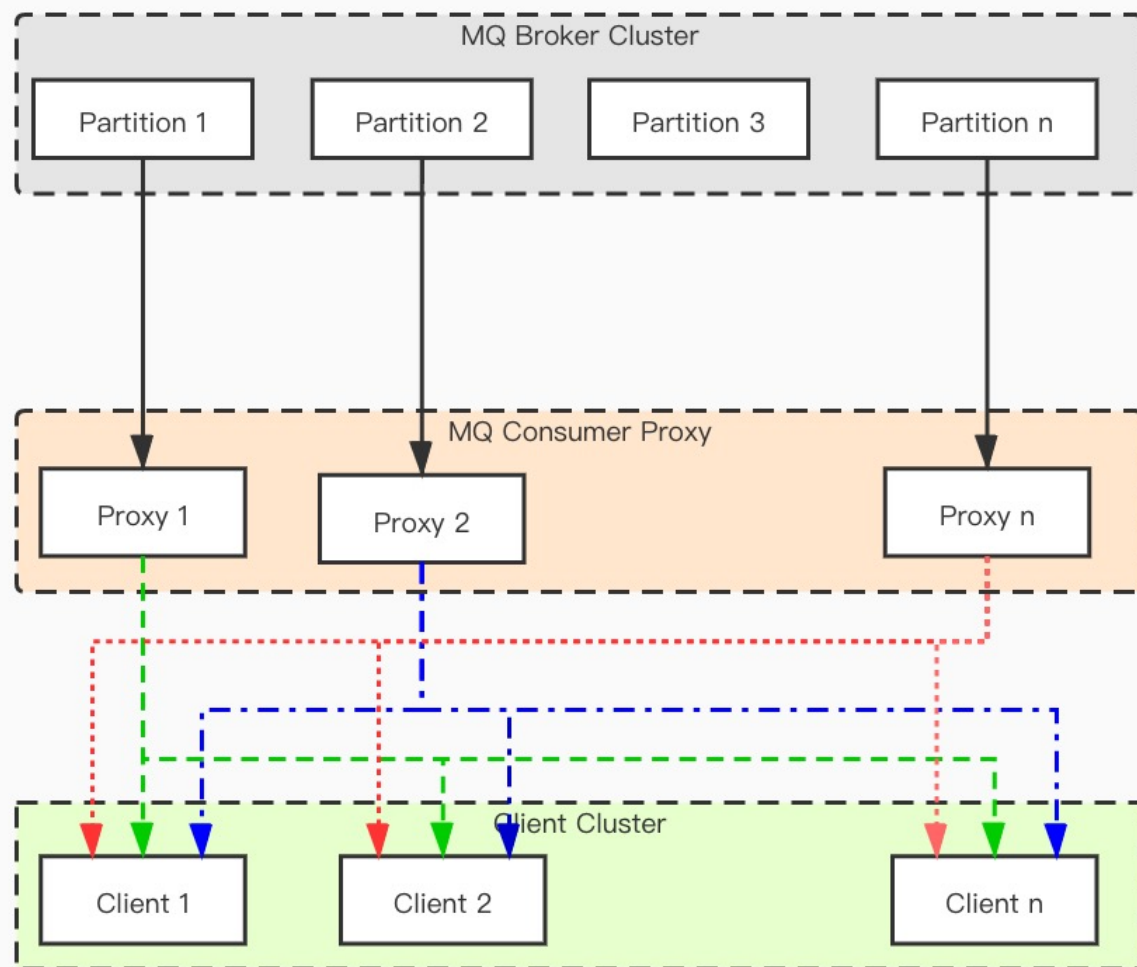
- Consumer&partition rebalance : 清空CProxy上的buffer
- 清空buffer仍可能重复: 消息已被client拉取, rebalance后, 新的CProxy节点仍会消费此消息

加了PProxy怎样保证顺序消息



- 顺序消息的原理：顺序消息落到同一个partition, 并串行发消息
- Proxy发消息也要保证串行

加了CProxy怎样保证顺序消息



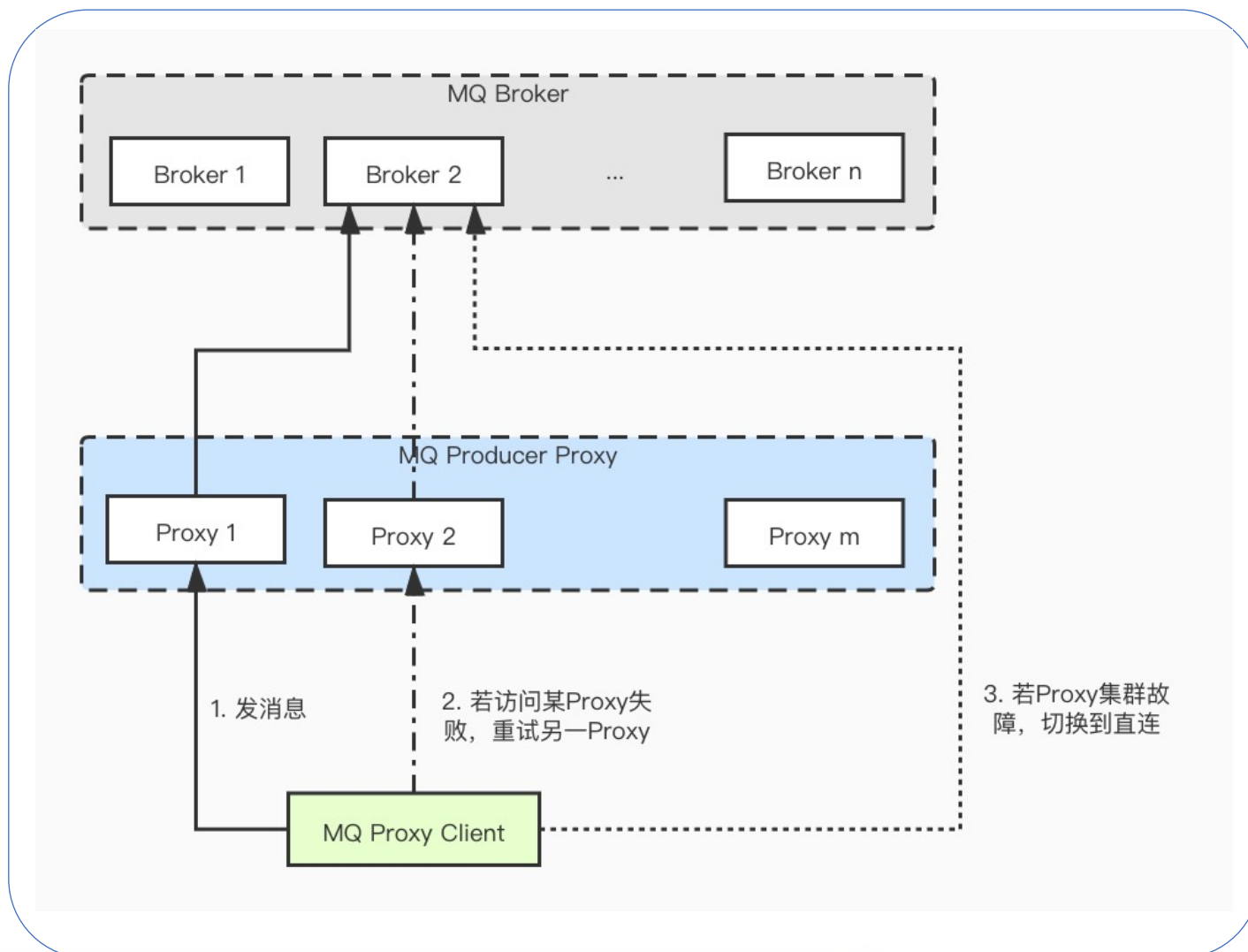
问题：

- Partition与Cproxy consumer一一对应。
- 但无法保证一个客户端仅对应一个Cproxy consumer

解决方案：

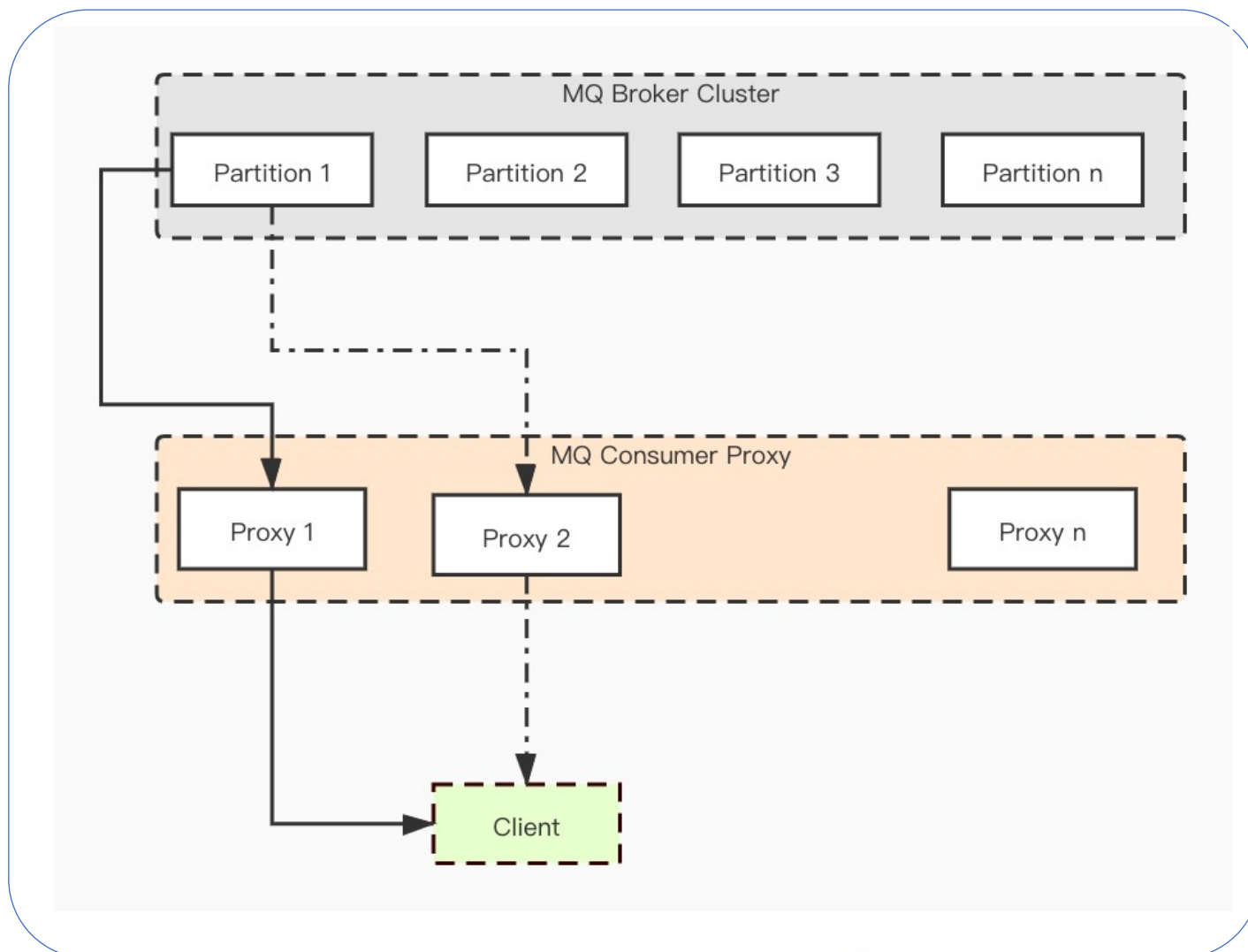
- Client确认处理完CProxy的消息后，CProxy consumer的后面缓存的消息才能被其他客户端拉取

PProxy 如何保证高可用



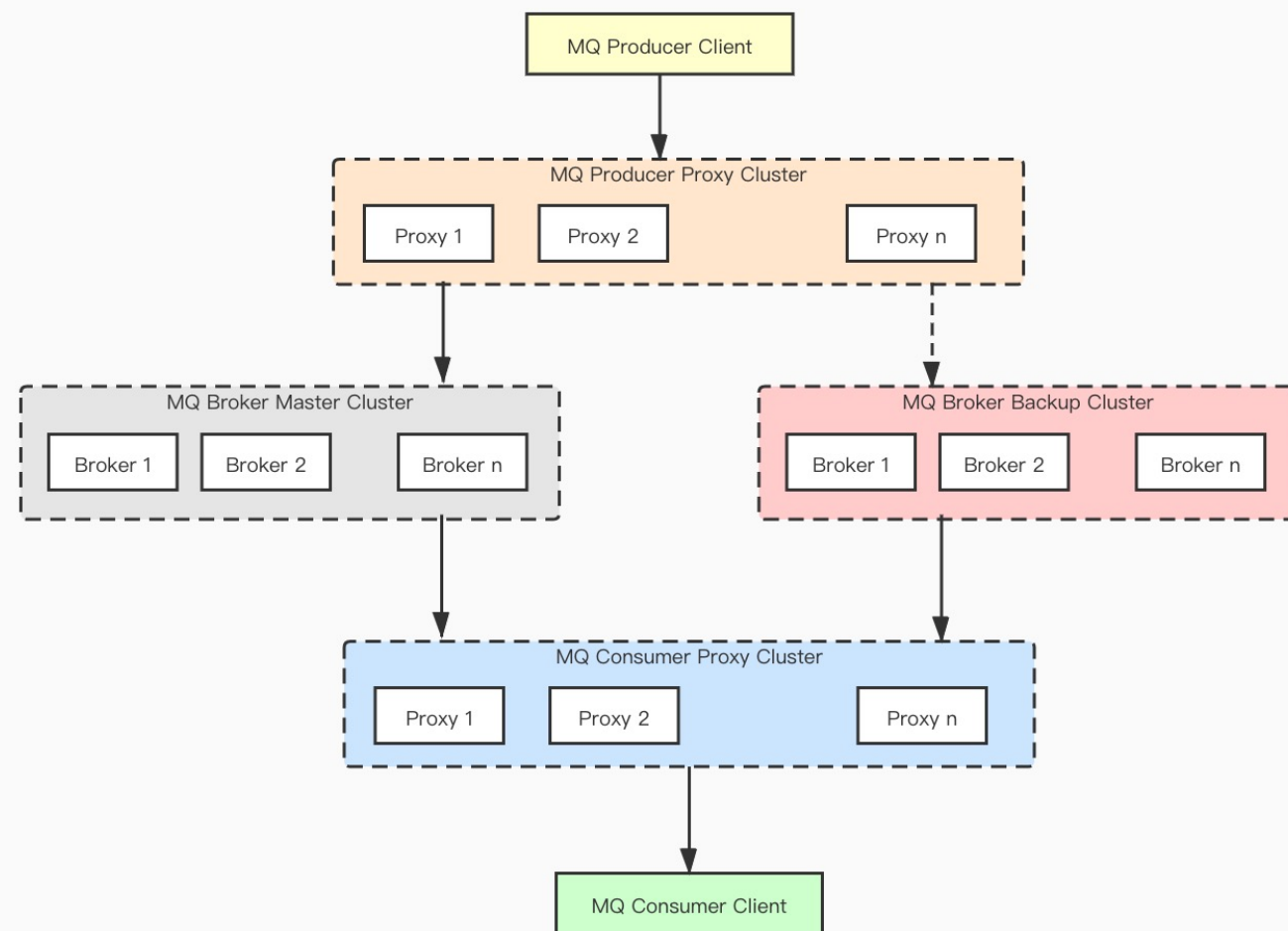
- 发消息，PProxy单节点访问失败，自动重试另一节点
- 若PProxy集群故障，业务客户端直连MQ Broker

CProxy如何保证高可用



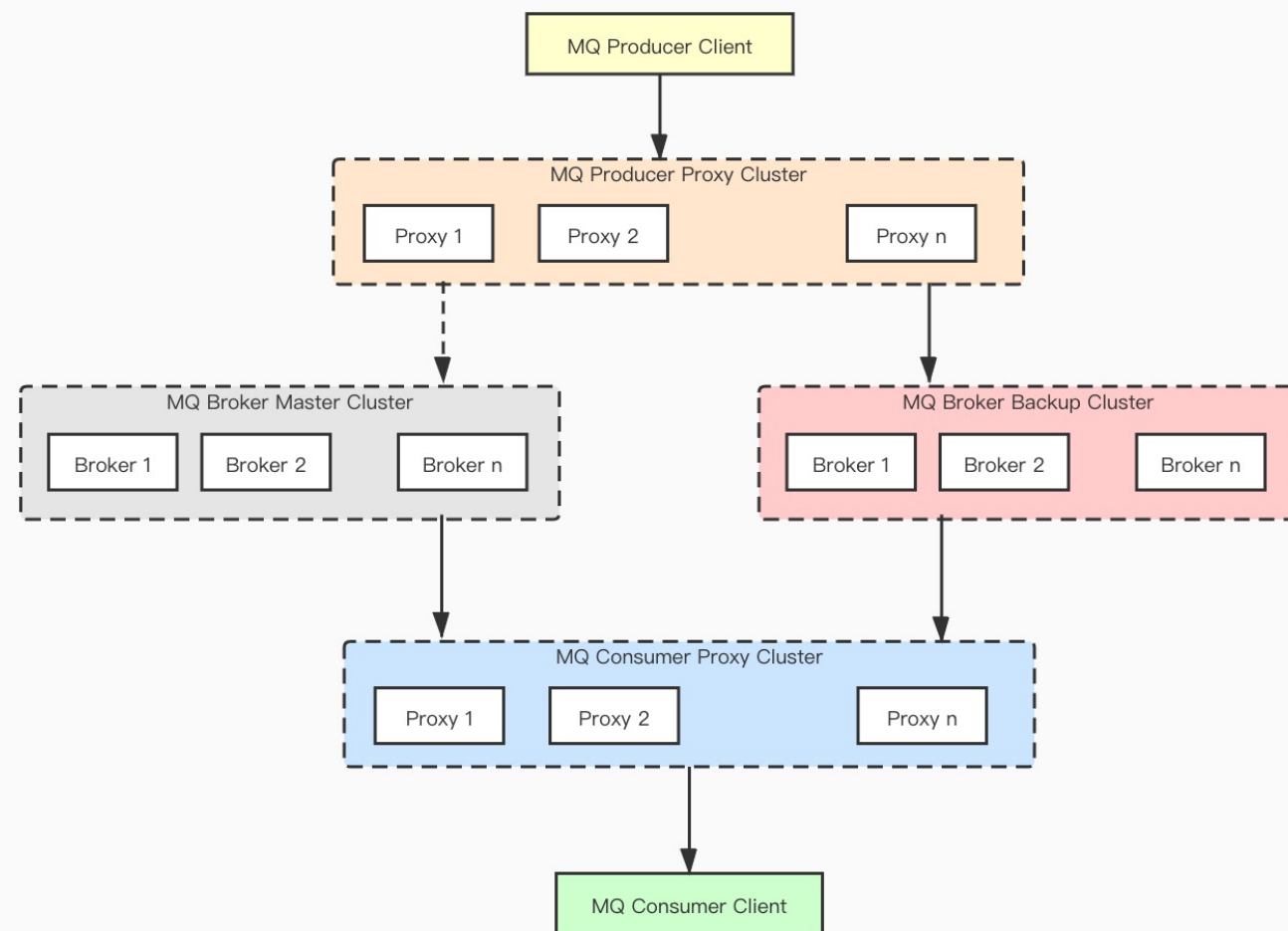
- 单台CProxy故障：消息可能会重（Client拉取到消息，CProxy未提交offset）但不会丢
- CProxy集群故障：消费MQ要等CProxy恢复后

基于Proxy实现MQ跨集群容灾



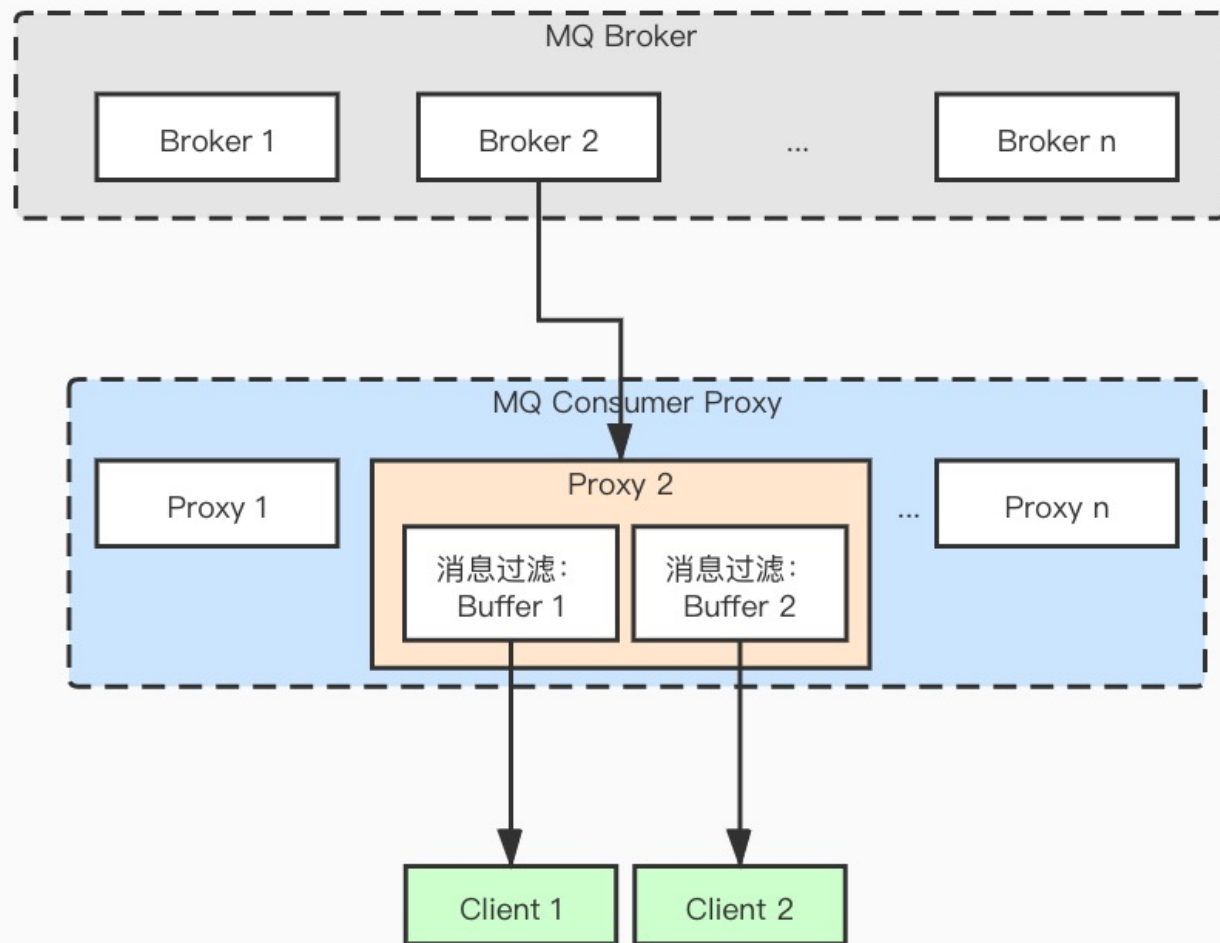
- 发消息，集群故障自动发现，自动流量调度，业务无感
- 主备MQ Broker集群，单写双消费
- 通过滑动窗口检测超阈值触发熔断，自动检测是否恢复
- 自动流量调度

基于Proxy提供集群数据迁移工具



- 切换写集群
- 单写双消费
- 原集群数据消费完毕后下线

基于CProxy实现MQ消息过滤



- 需求：MQ Broker中数据格式无法改变
- CProxy对数据预处理，消费完并过滤好等Client来拉取，提升性能
- 避免业务无意义消费全量消息后做过滤

后续展望

- 合并RocketMQ和Kafka的管理后台
- 合并Kafka和RocketMQ的客户端为Proxy的客户端
- 基于Proxy实现MQ的双机房双活的流量调度

04

MQ平台收益总结

- ▶ 不必浪费人力同时维护多语言客户端
- ▶ 消除客户端升级带来的人力成本和稳定性风险
- ▶ 统一服务端MQ版本
- ▶ 业务层对存储层无感知，一种客户端可兼容多种MQ
- ▶ 自动流量调度&故障容灾
- ▶ 可更轻量增强MQ原生功能

The background is a deep blue with a complex wireframe pattern of rectangular shapes, resembling a stylized city skyline or a digital grid. A bright, glowing blue light source is positioned behind the word 'THANKS', creating a lens flare effect that radiates across the image. A diagonal line of light also cuts across the scene from the top left. In the upper left, there are some small, faint geometric shapes. The word 'THANKS' is written in a large, bold, white sans-serif font, centered horizontally and slightly above the vertical middle. The overall aesthetic is futuristic and high-tech.

THANKS