分布式消息队列Qbus介绍

web平台部　代兵

2013.07.20

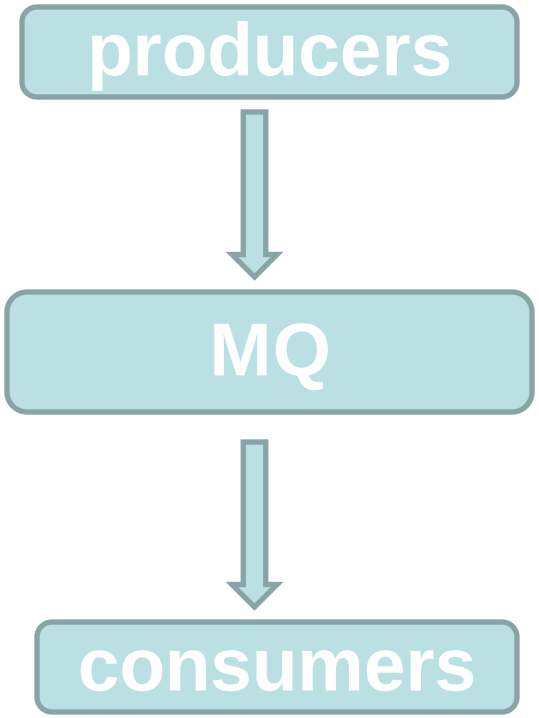
Agenda

- 基本背景
- 内部架构
- 问题及解决
- QA

神马是消息队列

消息队列

- 为程序或组件提供**异步**的通信机制
- 是在消息的传输过程中保存消息的**容器**
- 解耦
  - 发送者和接收者不必同时在线
  - 发送接收不必了解对方



两种模型

- 点对点（Point-to-Point）
- 订阅（Publish/Subscribe）

Gearman弱爆了

- 弱分布式。水平扩展性差
- 无订阅
- 可靠性差（使用外部存储性能较差）
- 运维工具不完善

So 我们需要另外一个靠谱的。。。

我们想要的

- 支持分布式
- 支持订阅
- 高可靠性
- 高性能
- 开发语言
- 社区活跃度，周边运维工具
- 用户使用成本等
- kiss原则

货架上有的

	RabbitMQ	ActiveMQ	beanstalkd	gearman	kafka
水平扩展性	较复杂	较复杂	一般	一般	容易
协议	AMQP	AMQP	类memcached	自有	自有
Pub/sub	支持	支持	不支持	不支持	支持
可用性	好	好	一般	一般	好
性能	一般	一般	较高	较高	高
语言	Erlang	Java	C	C	scala
代码量	较多	很多	较少	较少	较少
活跃度	很好	很好	好	好	好
用户量	很多	很多	较少	较多	一般
易用性	较复杂	较复杂	简单	简单	一般
运维工具	较完善	较完善	较少	较少	较少
对Php客户端的支持	好	好	好	好	一般

特点

Qbus

- 以kafka为原型
- 分布式
- 持久化
- 支持订阅
- 高可靠
- 高吞吐、高性能，与消息总量无关
- 支持消息批量处理
- 支持压缩
- 支持可选可靠级别

使用场景 —— 数据同步

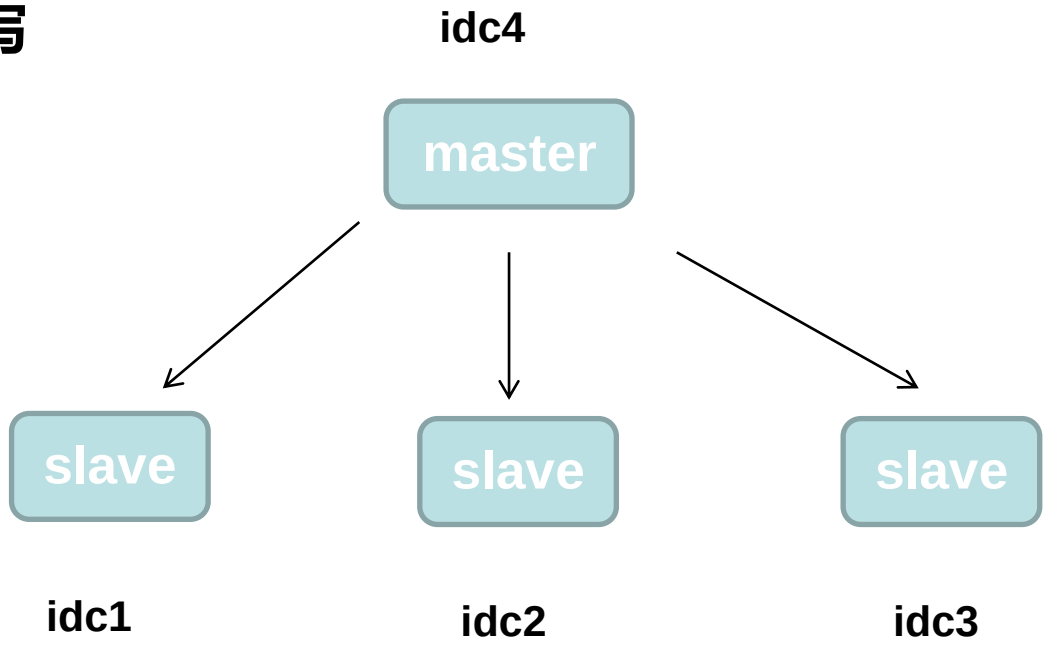
多IDC间数据同步常见方案

1. **master/slave**
2. **multi-master**
3. **multi-write(Paxos)**

Master/slave方案

问题

- Master中心化导致跨机房写
- 单点风险(failover)
- Idc太多？



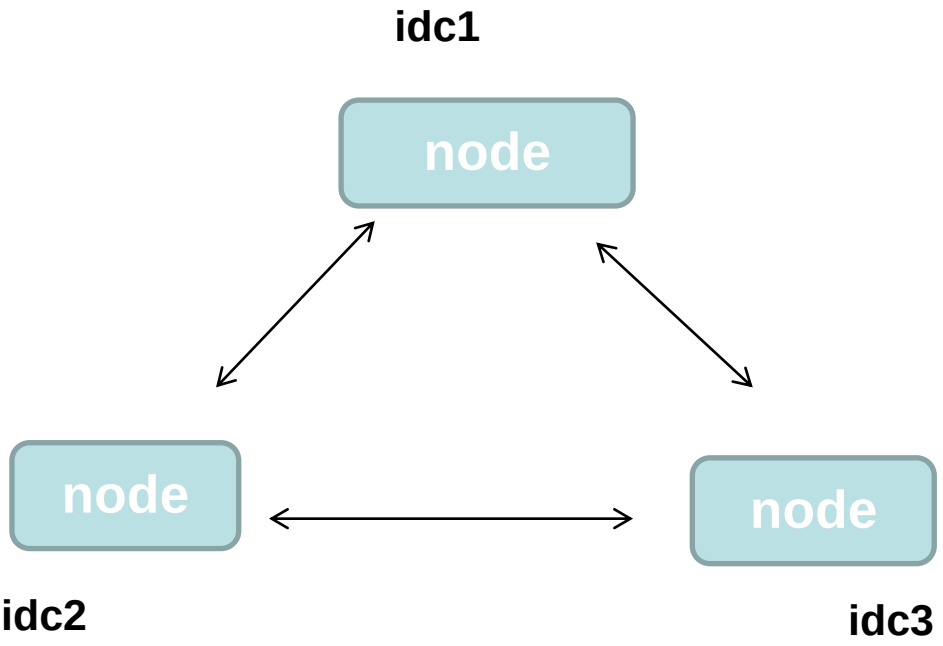
Multi-write方案

特点：

强一致写

问题

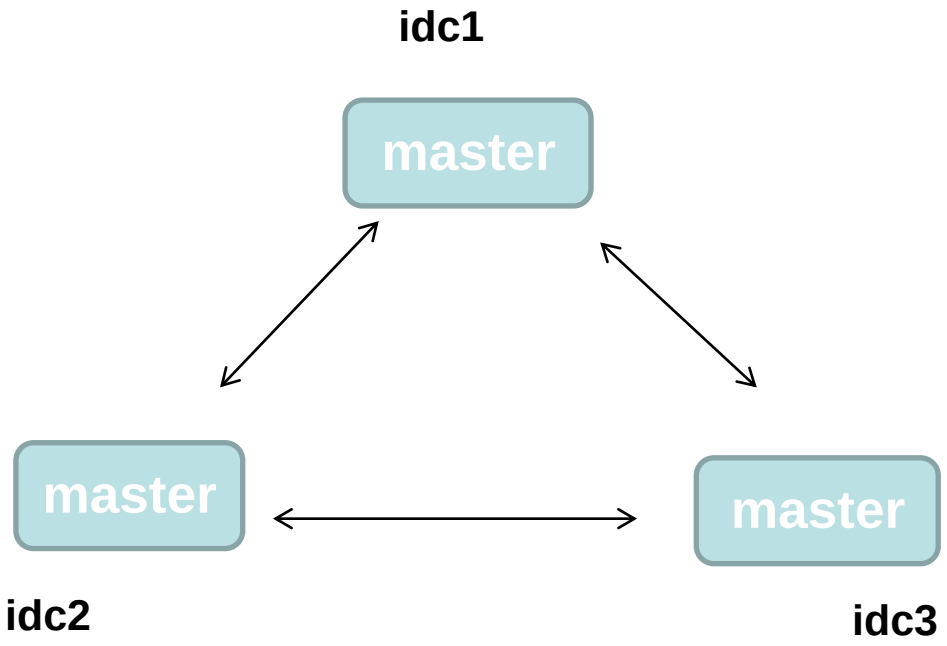
- **延迟大**



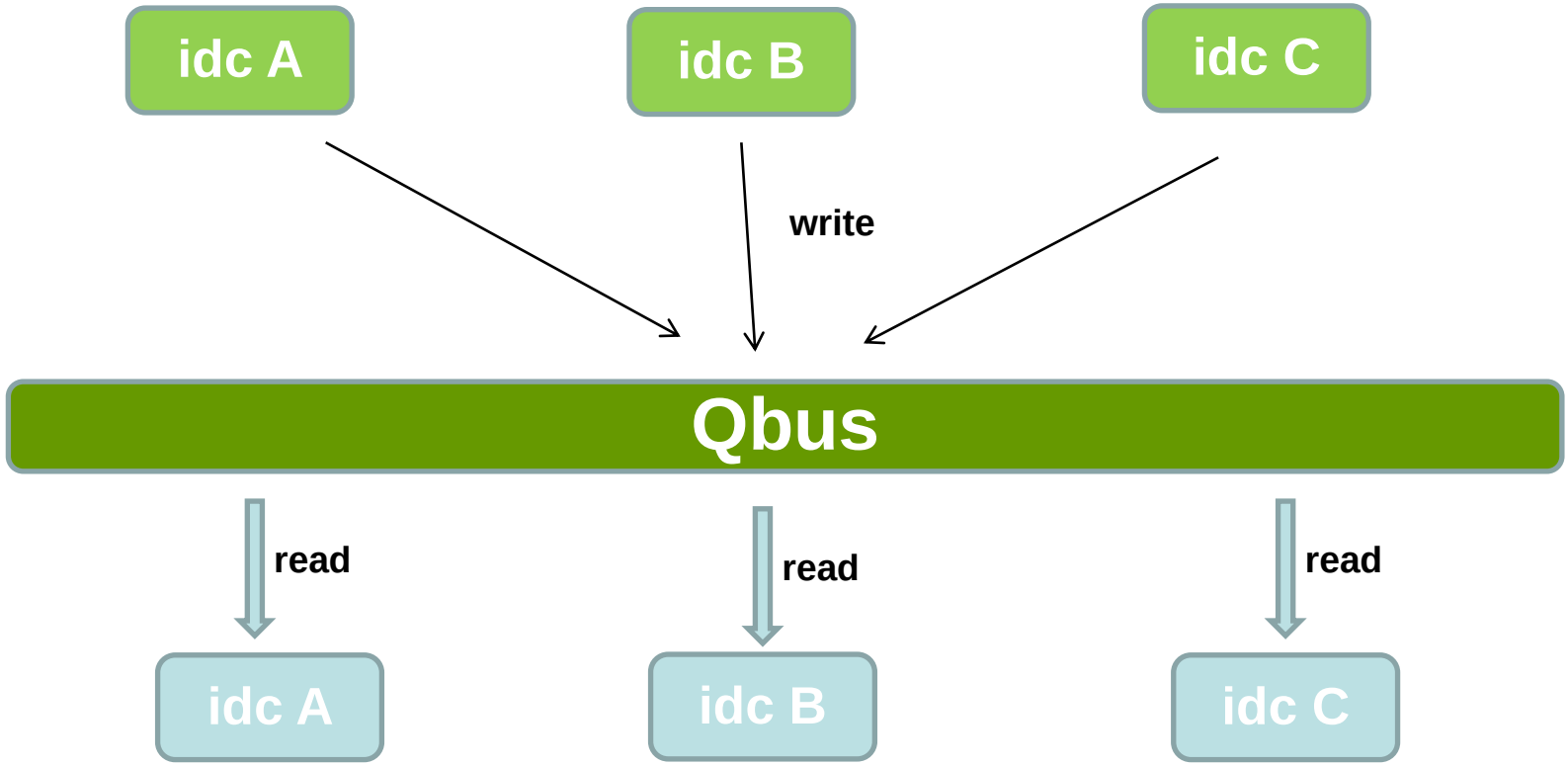
Multi-master方案

问题

- 冲突如何解决



Qbus解决方案



- write延迟小
- 可靠性高
- 各idc完全独立

使用场景

还可以用来

- **收发消息**
  作为普通的消息系统来使用
- **分布式任务**
  作为任务分发器
- **收集日志**
  收集业务日志进行实时存储、分析
- **监控服务，保护网站安全**
  监控服务访问情况，防止用户对网站进行无限制的抓取数据等

用的咋样

Server:
各IDC ~20个集群 70+台机器

Client:
~2000台

使用业务:
十几个业务和产品

使用场景:
业务消息，Idc之间数据同步，监控，实时统计分析，日志收集

处理量:
单个集群 每天 50亿条 2T

一个例子

其中一个集群五台broker一天统计情况

统计周期:从 20130513 09:00:10 到 20130514 09:00:08

各 broker 的接收发送情况

brokerid(host,port)	总的 fetch 次数	总的 produce 次数	总的接收流量	总的发送流量
[illegible])	12134951	228968431	110781048233(103.17G)	111773694526(104.10G)
[illegible]	12335887	228821873	110737394074(103.13G)	111667469562(104.00G)
[illegible]	16412927	228653503	110767253299(103.16G)	111659952138(103.99G)
[illegible]	16219813	229115786	110913563202(103.30G)	111759980289(104.08G)
[illegible]	15944650	228164151	110590645112(103.00G)	115914533676(107.95G)

各个 topic 的生产消费情况

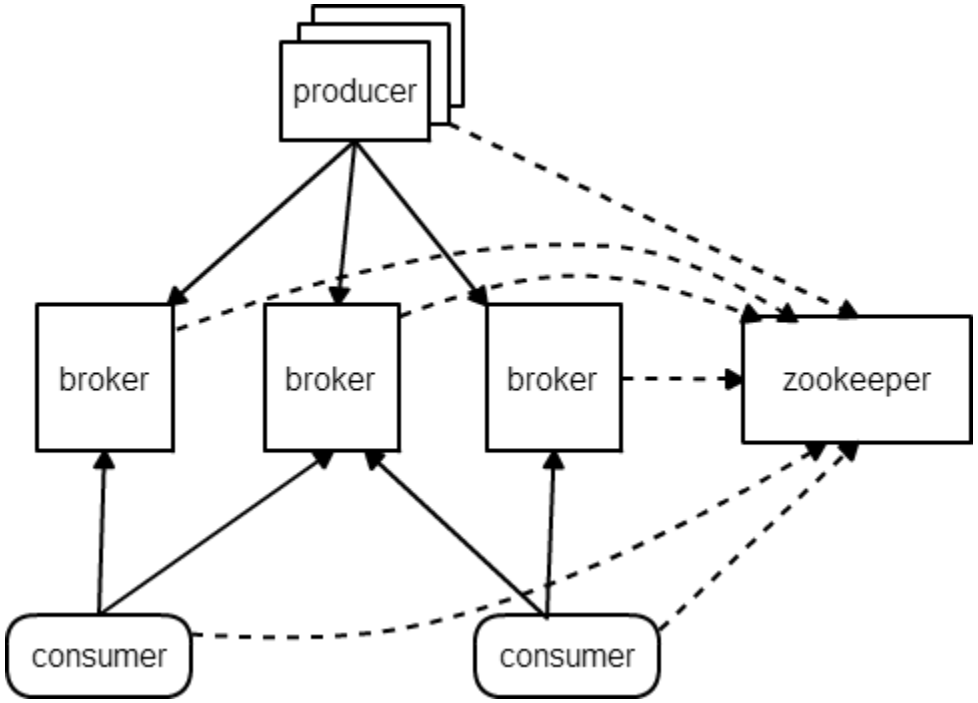
topic	group	生产消息的总数量	生产消息的总大小	消费的总大小
[illegible]	[illegible]oup	2587007	1105069736(1.03G)	1105069736(1.03G)
[illegible]	[illegible]oup	2587007	1105069736(1.03G)	1105069736(1.03G)
[illegible]	[illegible]	3258461415	235884423163(219.68G)	235873986998(219.67G)
[illegible]	[illegible]	3258461415	235884423163(219.68G)	2956386(2.82M)

架构那点事儿

- 分布式架构
- Broker（服务端）设计
  - 无状态
  - 高性能
- 负载均衡实现
- 订阅的实现方式

内部架构 - 分布式

- 四个角色：producer, broker, consumer, zookeeper
- broker: 即server。存放消息的服务器
- broker: shared nothing
- producer向某个topic发布消息，consumer订阅某个topic的消息

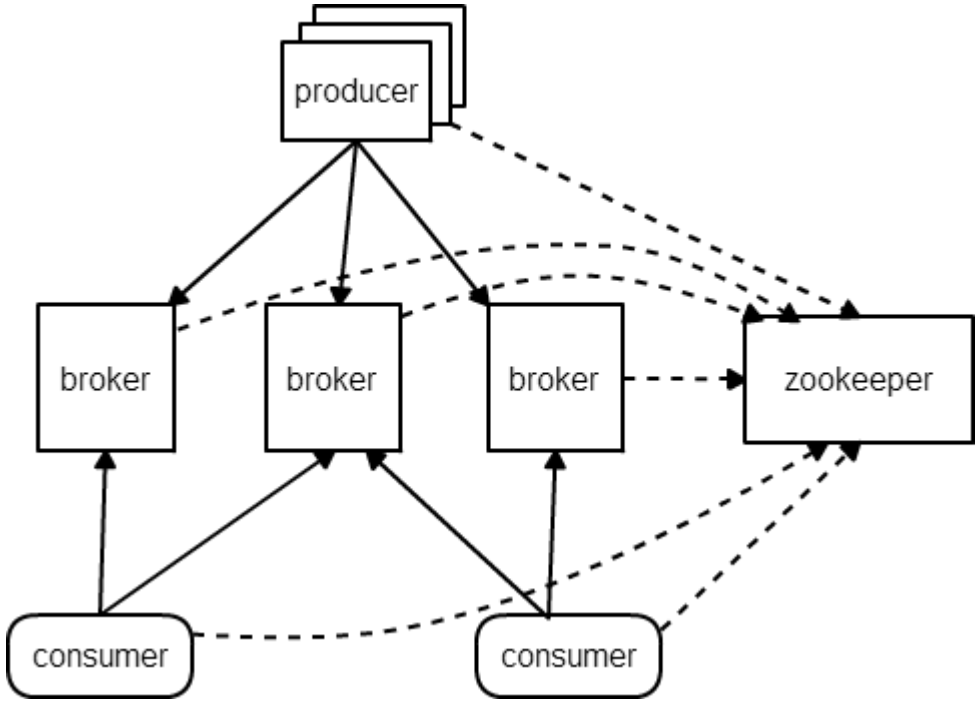


分布式数据管理与协调系统

- 配置管理/集群管理/分布式锁/。。。
- 类似文件系统**树状**结构来存储数据

在qbus系统中，zookeeper用来管理

- 所有broker的信息
- 所有topic的meta信息
- 所有consumer信息
- 消息的消费状态



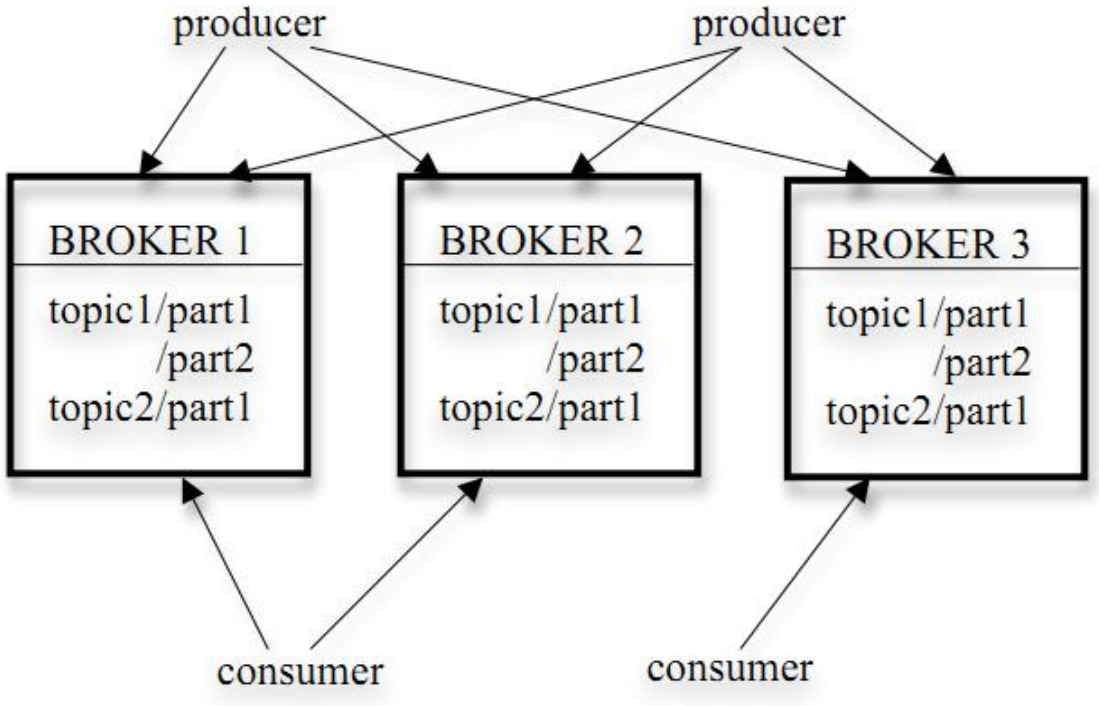
传统Broker的实现需要维护

- 每条消息消费状态
- Consumer的信息
- 每条消息消费完后删除

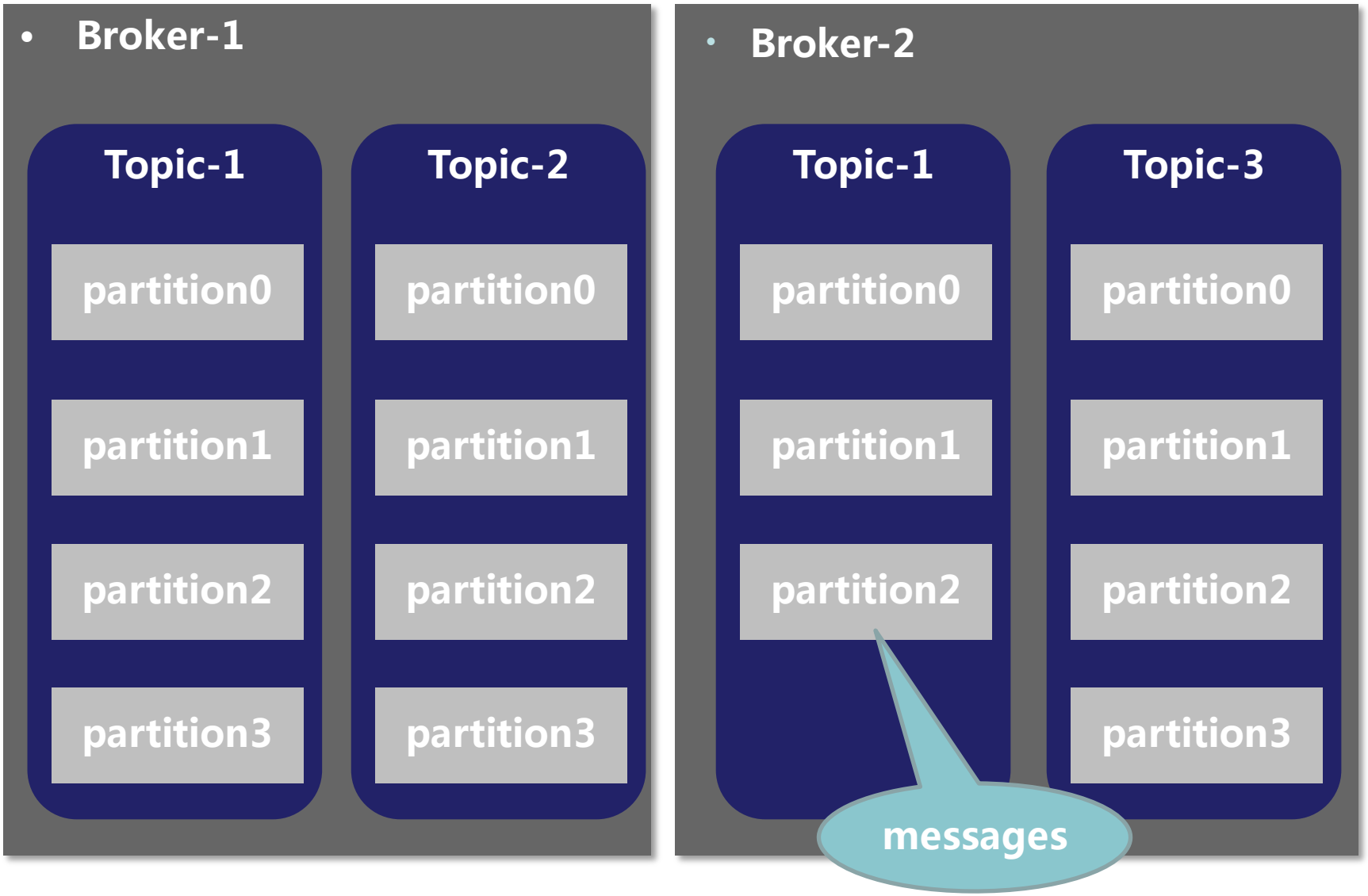
Qbus Broker特点——无状态

Broker

- 接收producer发来的消息，持久化存储，接收consumer的请求
- 以topic来进行消息管理，每个topic包含多个part（ition）
- 一个Topic即一个队列，存储同类型的消息
- 多partition用来增加consumer的并发度



Broker中消息存储结构

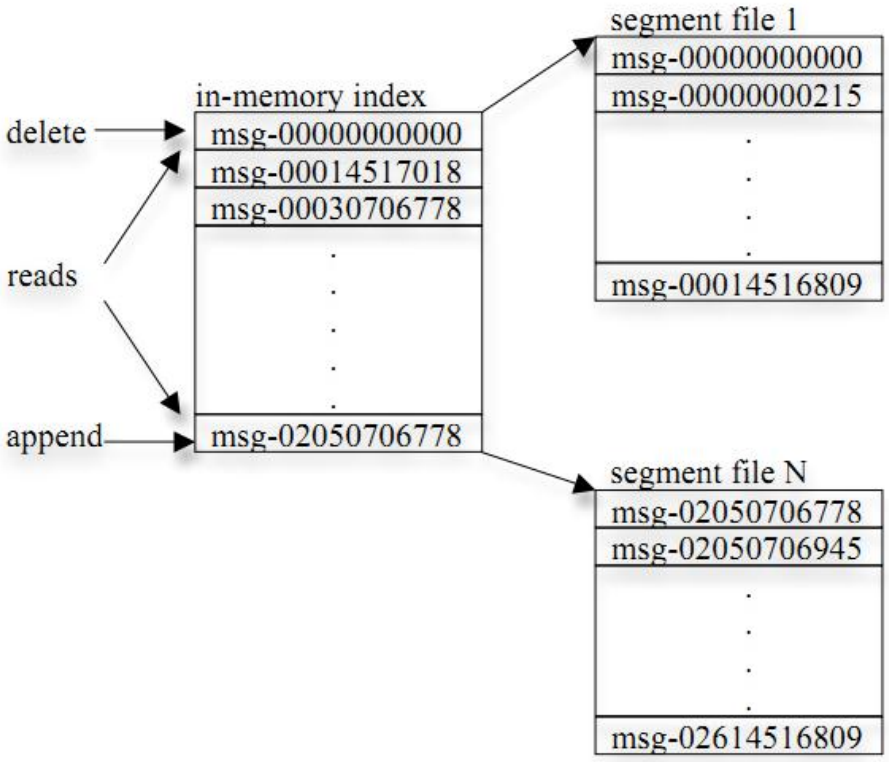


Broker中消息存储结构

Partition中的消息如何删除？

Broker中消息存储结构

- 每个partition对应一个逻辑log，由多个segment组成
- segment以第一条消息在这个partition中offset为文件名
- 新的消息追加到最后一个segment文件末尾
- offset即为消息id，由id可直接定位到消息的存储位置
- 定期以Segment为单位删除消息



Broker的高性能

性能如此高？Why?

Broker的高性能

- 顺序读写磁盘效率很高
  - ✓ 顺序读/写速度 > 300MB/s
  - ✓ 写消息Append代价为O(1)
  - ✓ 从指定位置offset读消息代价O(1)
- 性能与消息总量无关
- zero-copy
- 支持消息批量处理
- 支持压缩

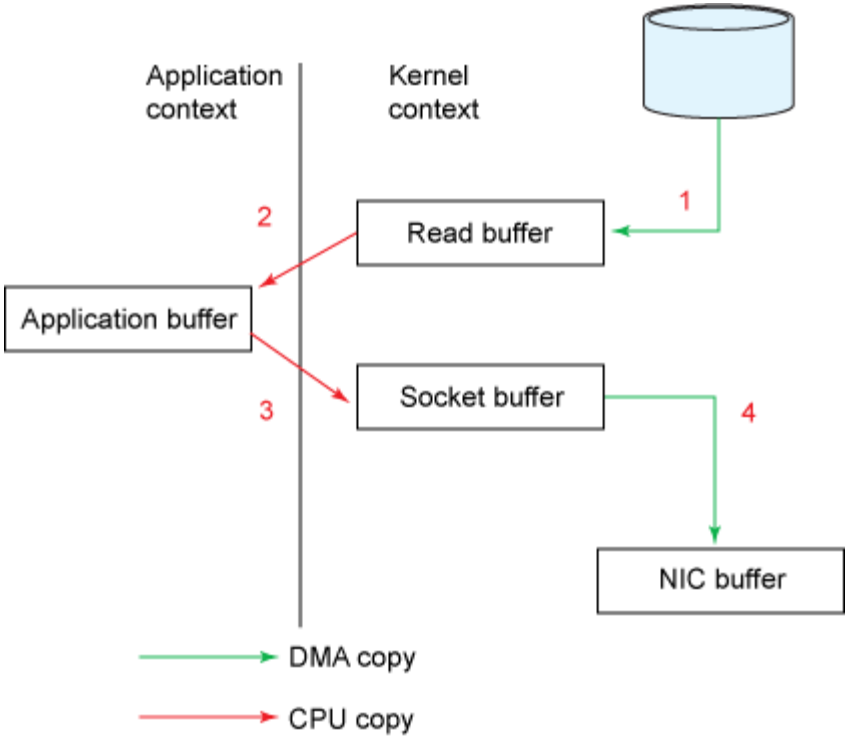
zero copy

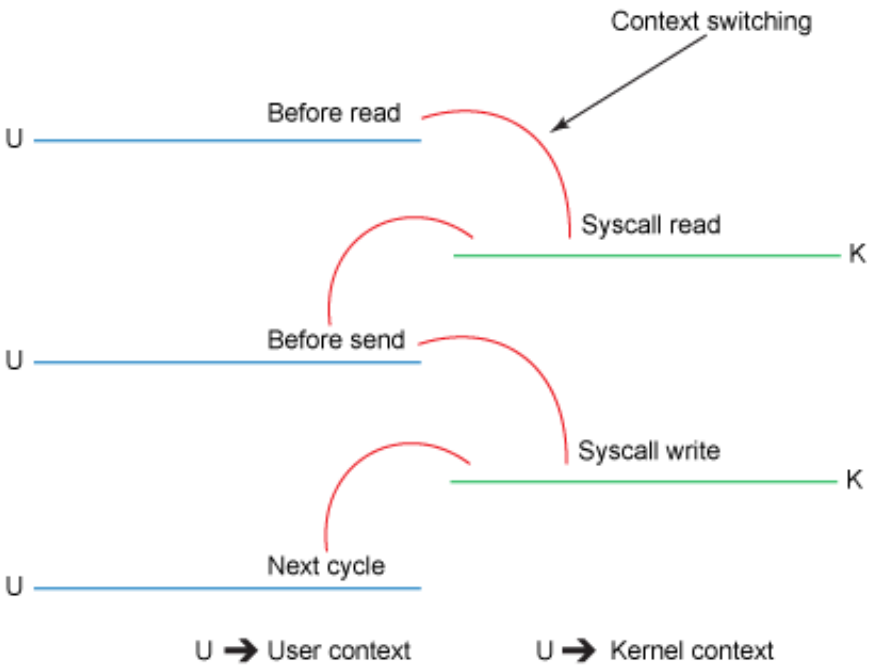
- 传统读 - 发送步骤

  read(fileDesc, buf, len);

  send(socket, buf, len);

- 代价
  - 2次拷贝
  - 2次系统调用



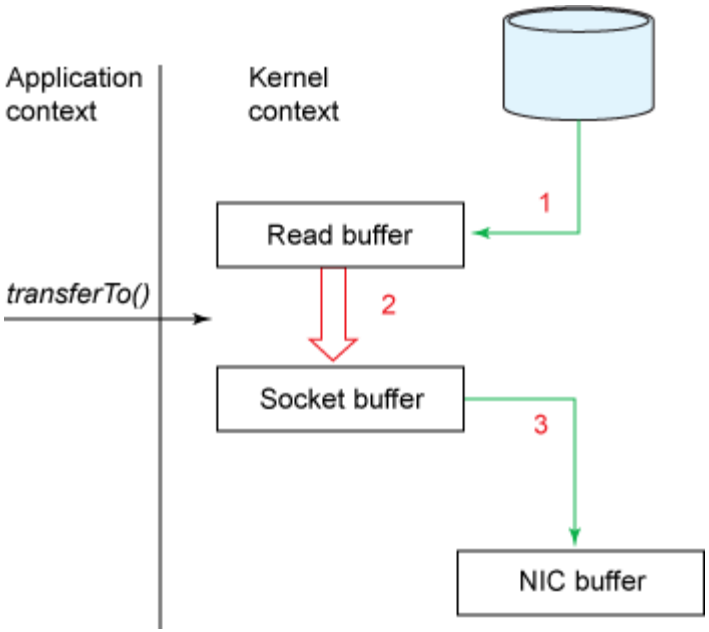


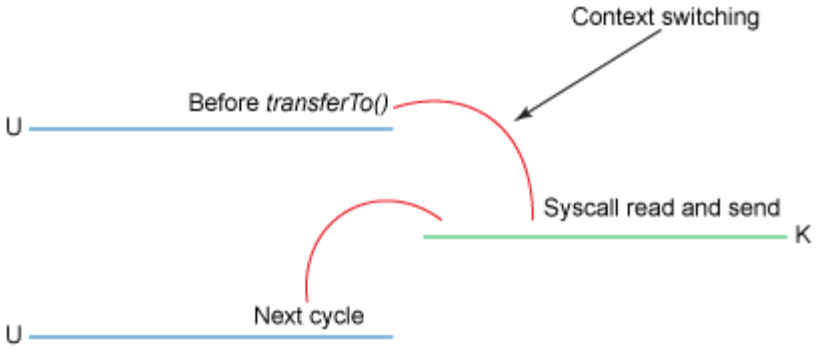
zero copy

- zero-copy技术

  sendfile(int out_fd, int in_fd, off_t *offset, size_t count);

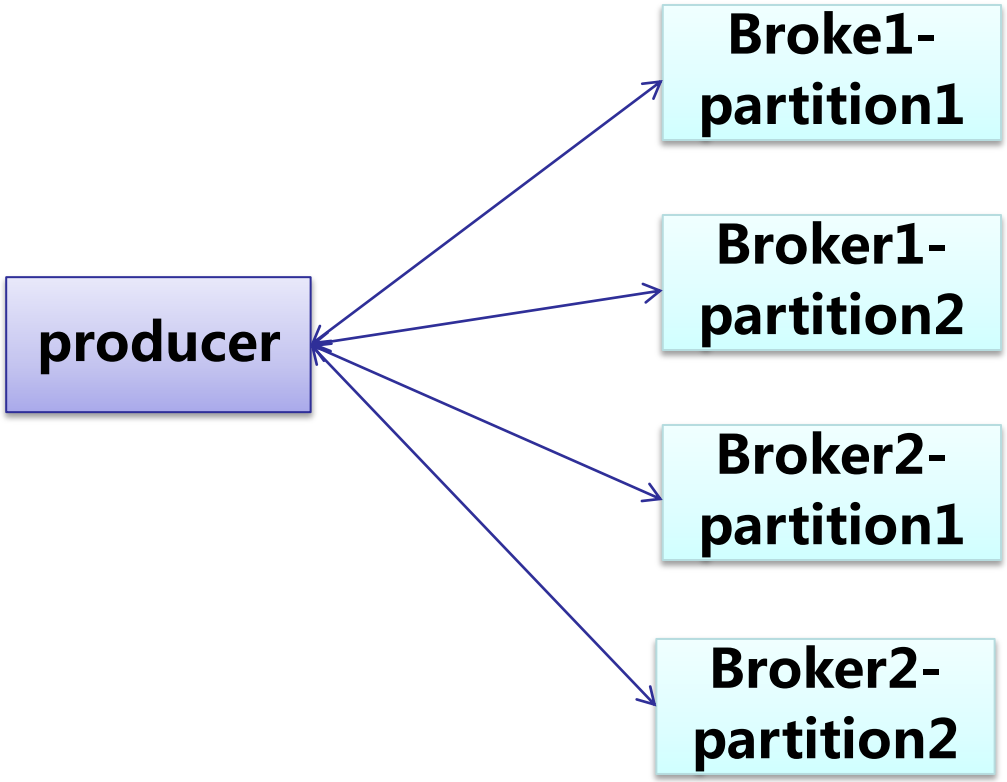
- 代价
  - 1次拷贝
  - 1次系统调用
  - 比read-send方式节省60 - 70%时间





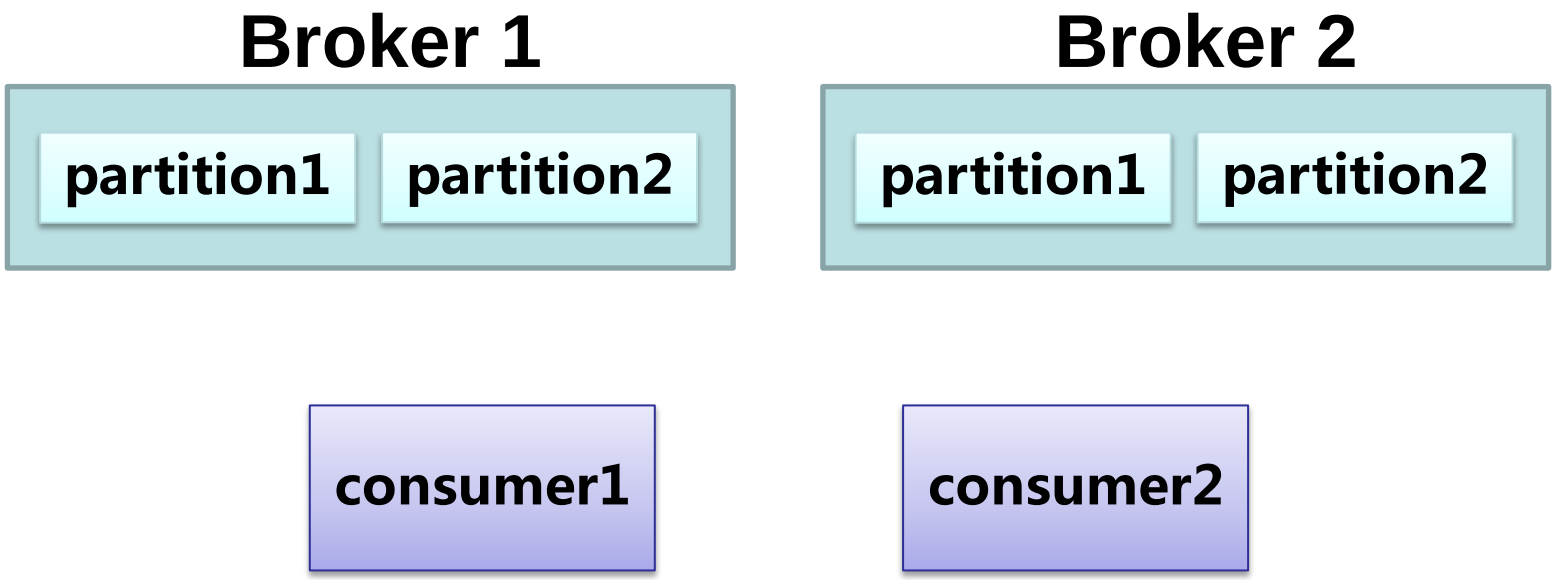
发送方式

- layer-4 load balancing —— 完全随机选择broker与partition
- zookeeper-based load balancing —— 指定发送到broker-partition

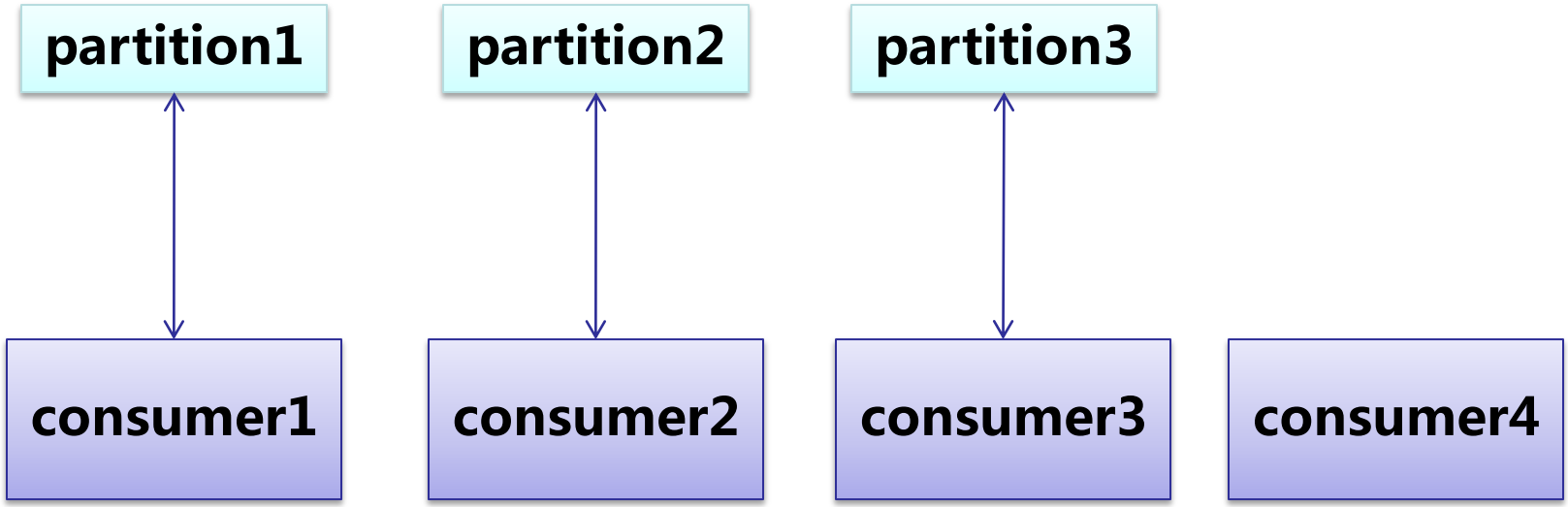


Consumer

- M个partition如何被N个consumer消费？
- 如何保证每条消息消费仅消费一次？
  - ✓ Broker变动
  - ✓ Consumer变动

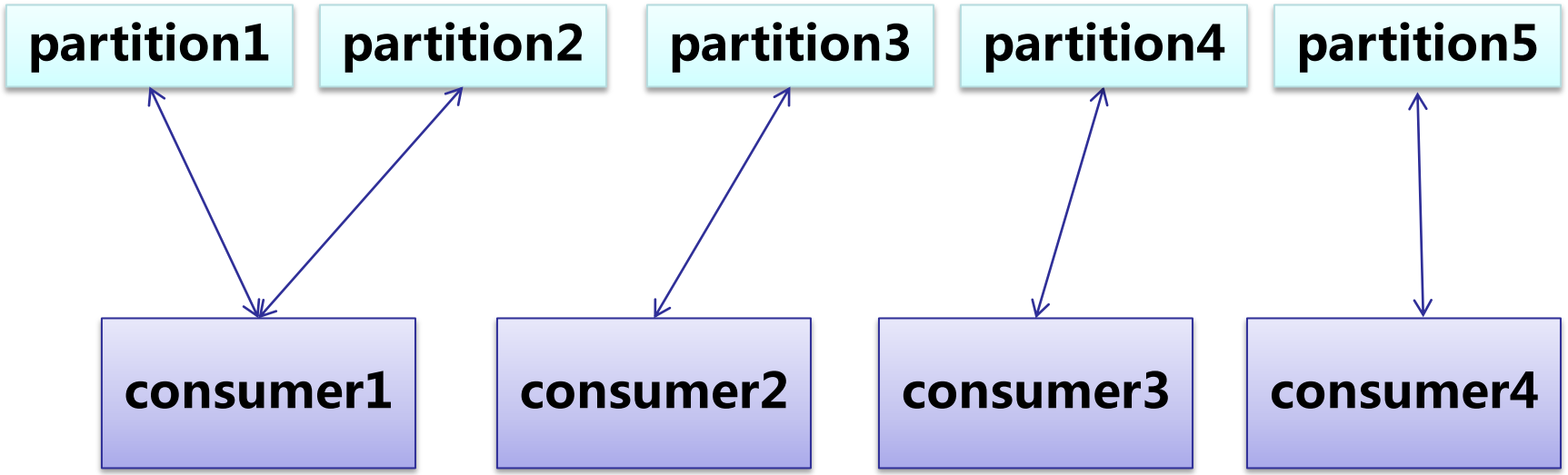


Consumer负载均衡



- 每个分区只能被一个消费者消费
- 多余的消费者不参与消费

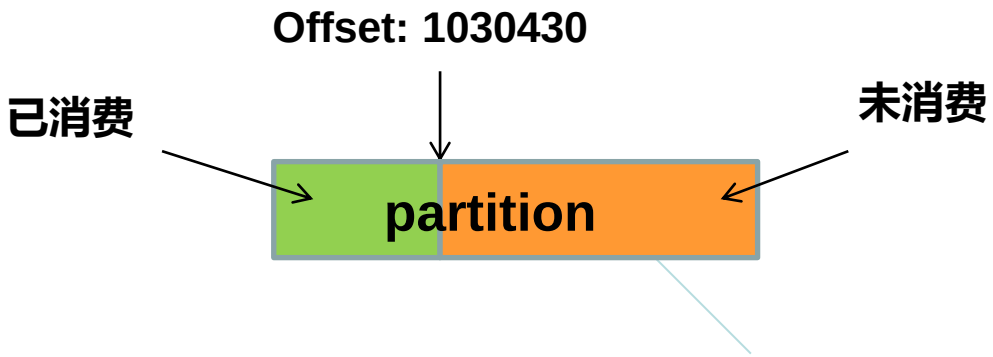
Consumer负载均衡



- 当分区数目(n)大于消费者数目(m)的时候， 则有n%m个消费者需要额外承担1/n的消费任务。
- n足够大的时候，仍然可以认为负载平均分配

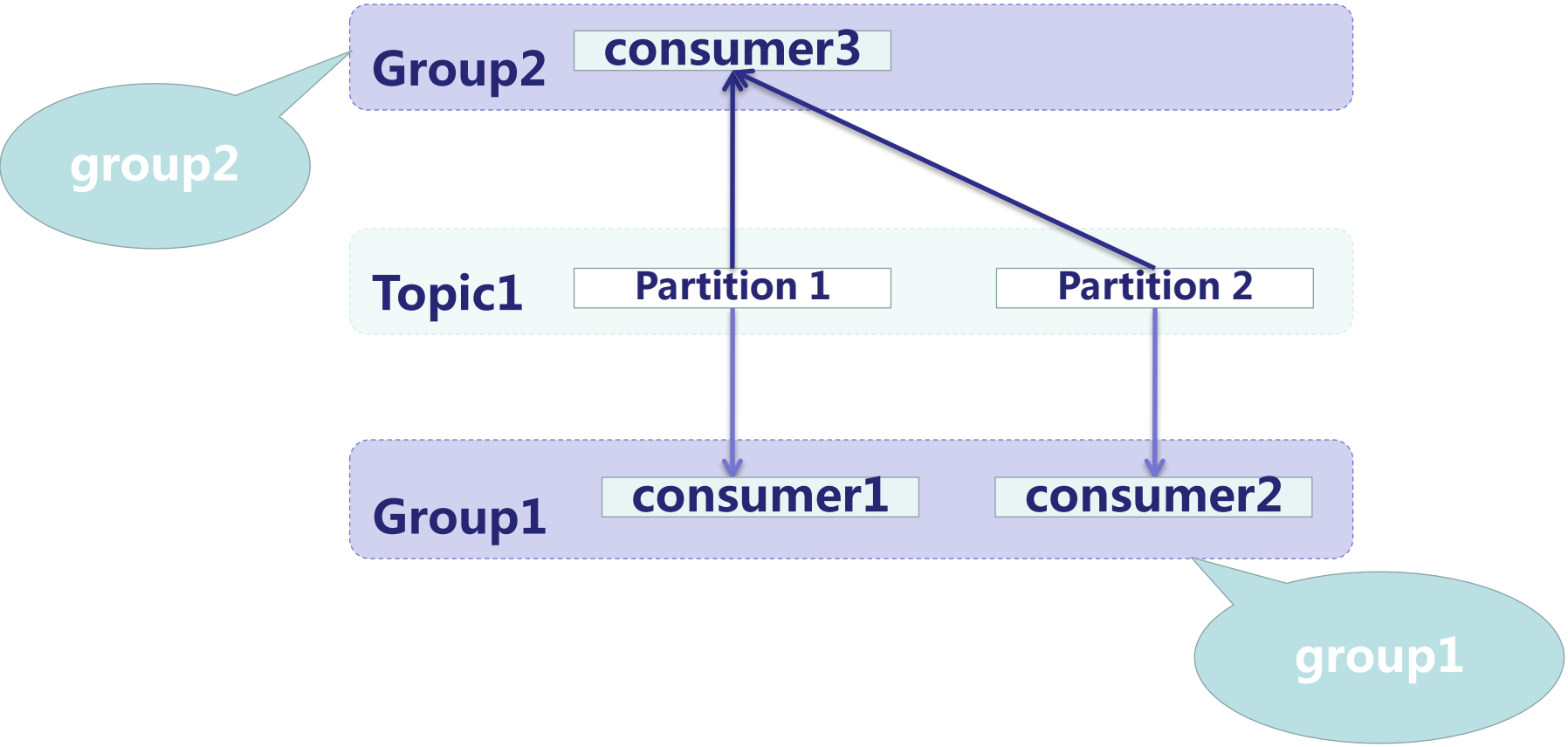
Consumer消费过程

- 每个partition消费到的位置(offset)保存在zookeeper上
- 每条消息的消费状态由partition的offset来决定



实现订阅方式

- 一种消息有多个业务逻辑来消费多次怎么办？
- 使用Group
- 每个消费组在zookeeper上有自己独立的一份partition消费的offsets



设计理念

将broker逻辑适当转移到consumer，使broker非常简单高效

增加消息协议

问题：

Qbus丢消息？

解决：

增加新的协议，发送消息时支持可选ACK，解决速度与可靠之间的权衡

消息发送的问题

- 网络异常时，producer发送失败时怎么办？
- 跨idc发送出现timeout
- 实时处理线上的日志文件？

解决：

增加agent角色

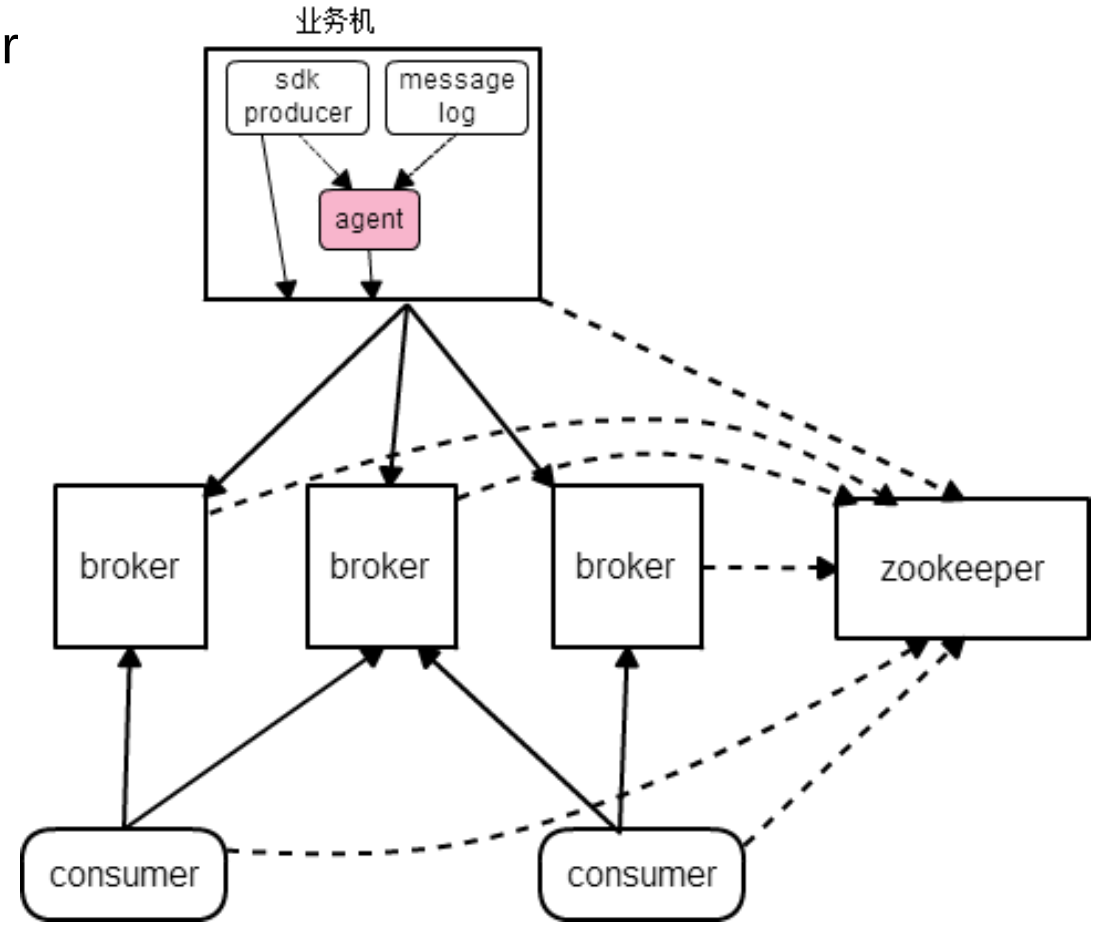
agent

agent作用：

- 将日志文件实时push到broker
- 支持sdk异步发送

特点:

- 通过zookeeper中心化控制
- 配置从zookeeper上获取
- 可以实时查看当前处理状态
- 支持压缩
- 。。。



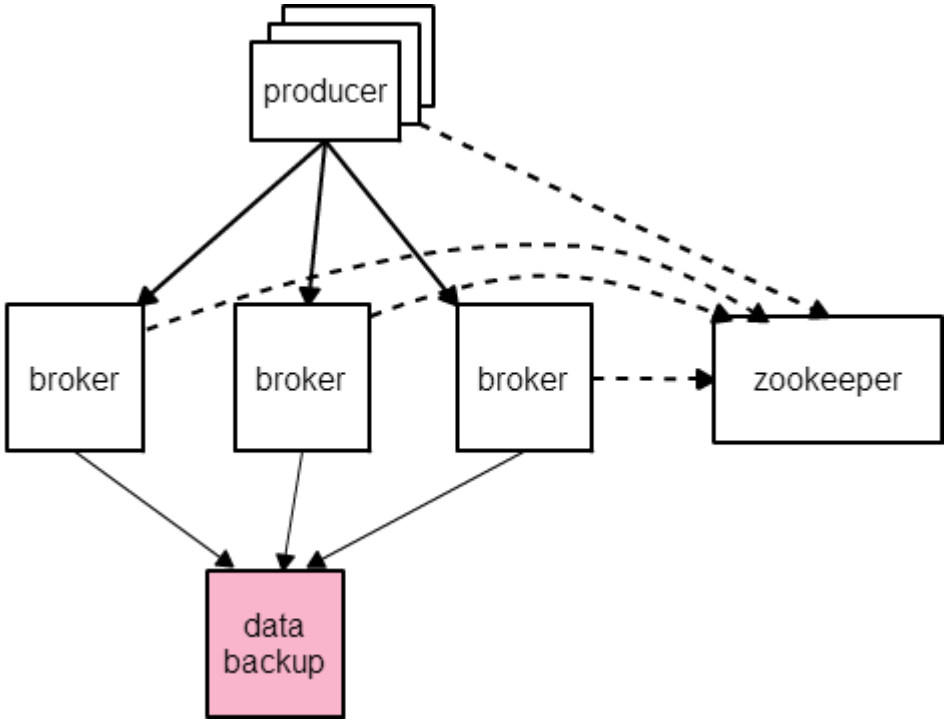
可靠性保证

服务可靠性

- 机器宕机
- 网络中断
- 机房断电

数据可靠性

- 机器宕机
- 磁盘坏掉



性能测试

环境:
普通虚机
CPU: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
MEM: 4G
DISK: SAS 15000

参数:
message size = 100 bytes
batch size = 100 messages
fetch size = 1MB

1 发送: 吞吐量约100MB/s，10w/s。
2 消费: 单consumer约10-50w条/s

Thanks!

Q&A

email: daibing@360.cn
Weibo: @durbin