情况说明

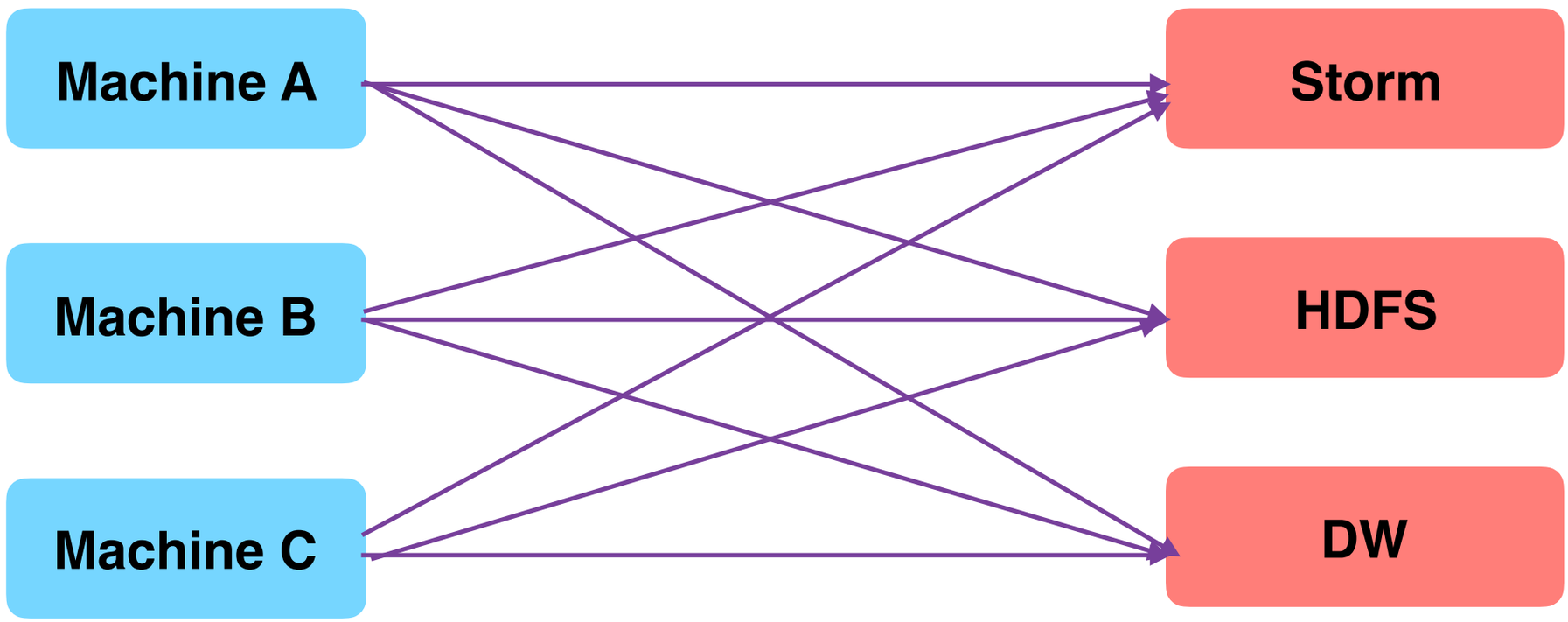
- **分享主题**：**从 MQ 说到 Kafka**
- **时间地点**：2017-08-23（周三）18:30，2F - War Room会议室
- **面向对象**：
  - 数据组：Kafka 的接入方
  - 关注 MQ 的工程师
- **主要内容**：
  - MQ 的**通用知识**
    - MQ 的价值
    - MQ 的核心思路
  - Kafka 的**设计原理**
    - 基本原理：并发性、有序性、高可靠
    - 使用实践：并发调优、常见问题

从 MQ 说到 Kafka

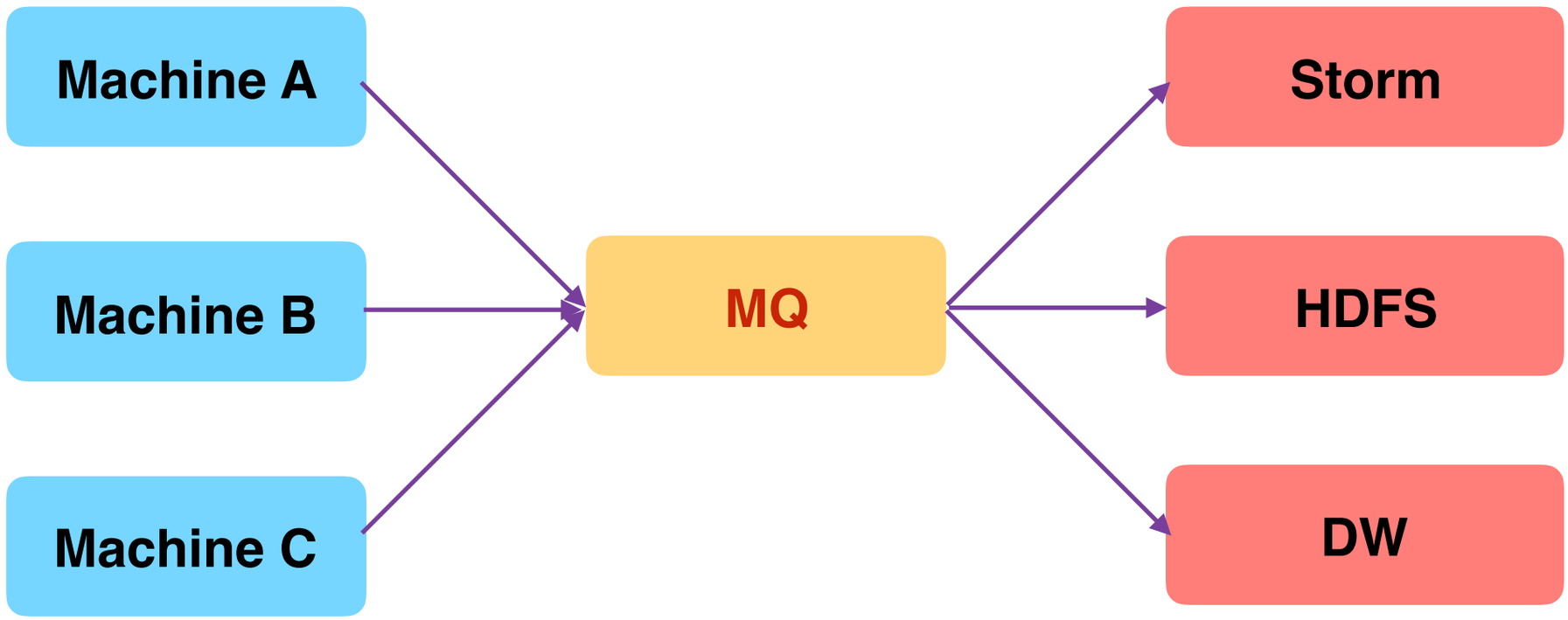
—— 郭宁@摩拜
guoning@mobike.com
2017/08/23

目录

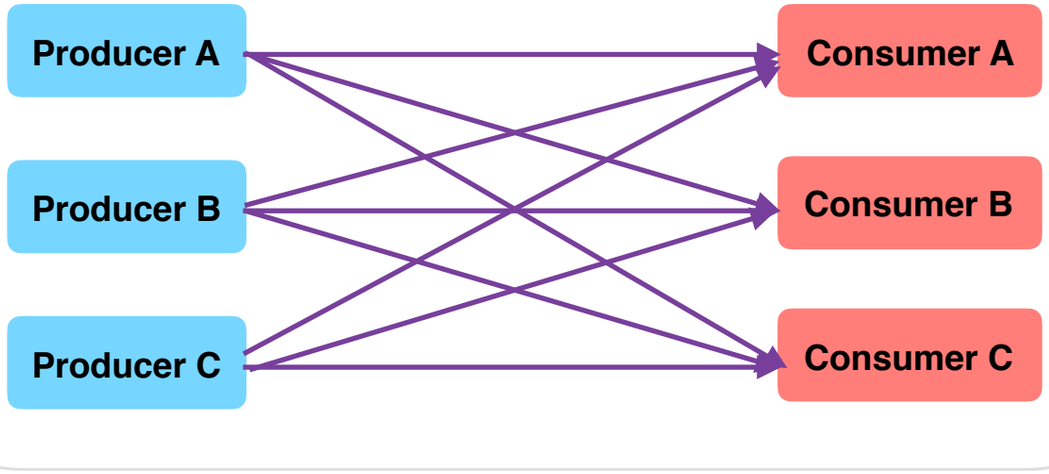
MQ有什么用？



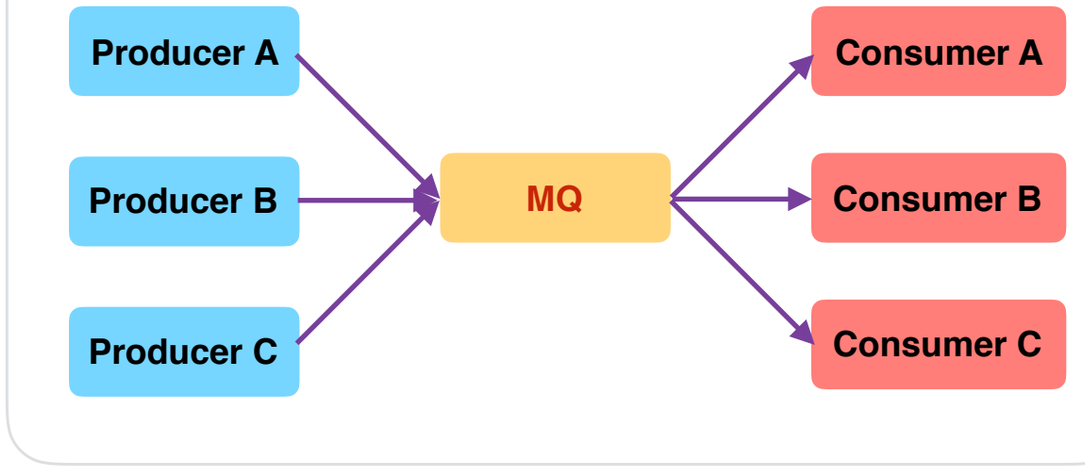
MQ有什么用？



MQ 有什么用？



VS.



	可扩展性	可靠性
Producer	😀	😷
Consumer	😀	😀

MQ 有什么用？

- 其他应用场景：
  - 平滑突发峰值：系统突发处理能力不足，平均处理能力可以，MQ 平滑突发峰值
  - 异步任务：耗时的任务

- 更多参考：
  - Top 10 Uses For A Message Queue

小结

- MQ有什么用?
  - 可扩展性
  - 可靠性
  - 平滑突发峰值
  - 异步任务

- **核心**思路：**解耦**
- 基本**模型**：生产者– –消费者

Kafka 设计原理

- 整体结构

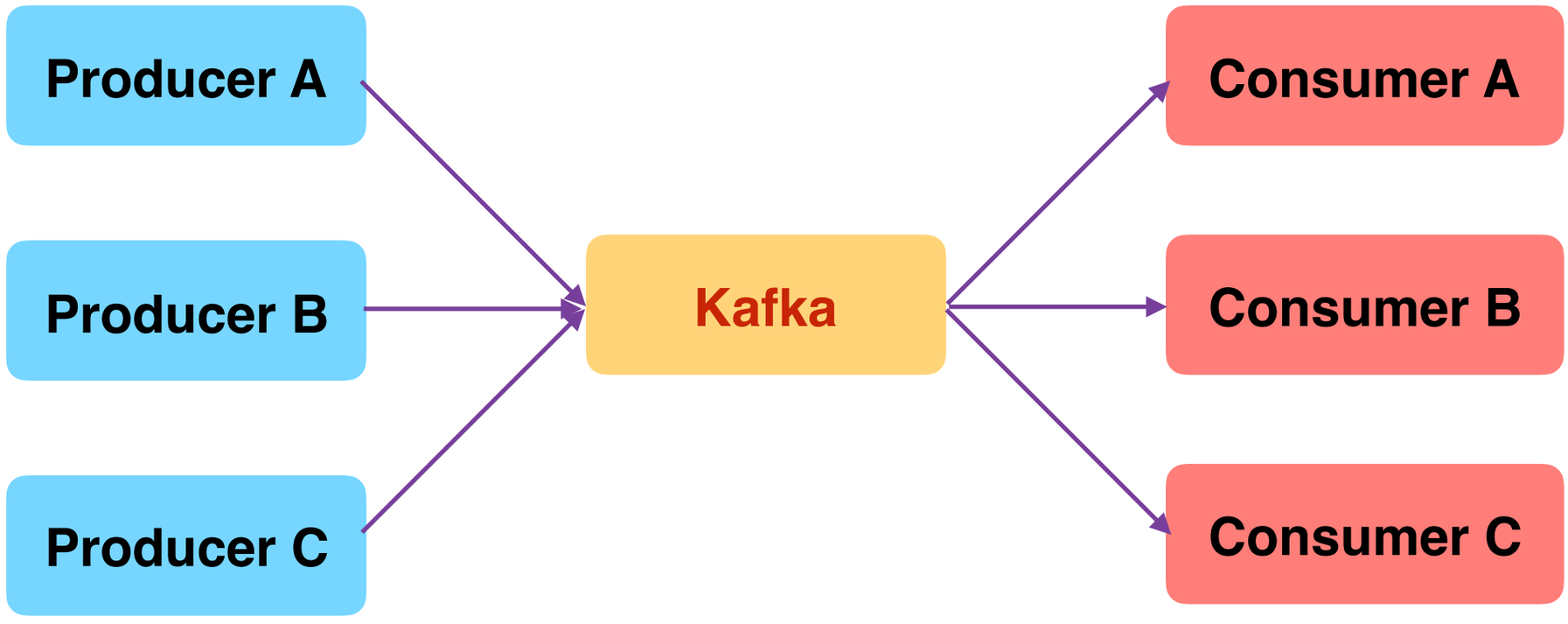
- 核心原理：（关键术语）
  - topic
  - consumer group
  - partition
  - replica

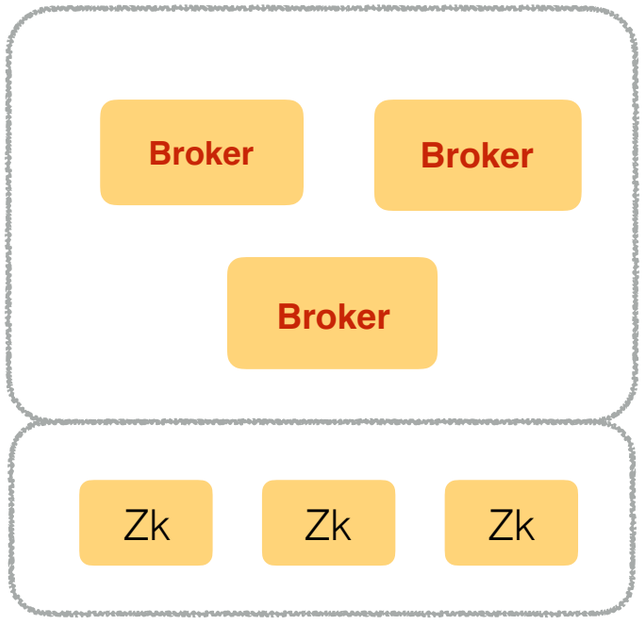
- 关键实现细节

Kafka 是什么?

Kafka 怎么实现的?

Kafka 整体结构

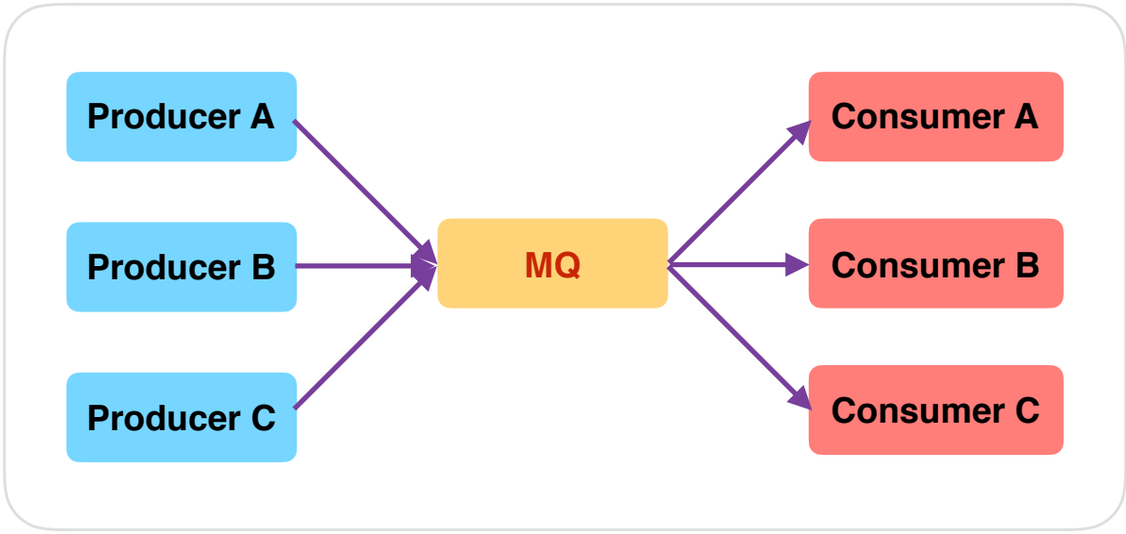

Producer A
Producer B
Producer C
Kafka
Consumer A
Consumer B
Consumer C


Broker
Broker
Broker
Zk
Zk
Zk

Kafka 核心原理

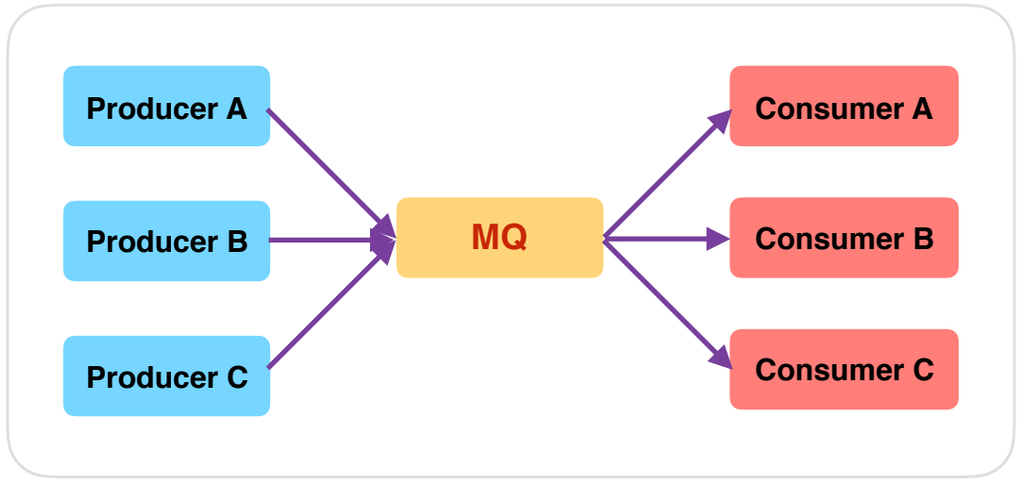
- topic
- consumer group
- partition
- replica

- **问题1**：不同业务数据，分开存放
  - OS：CPU、内存、网络
  - APP：pv、响应时间

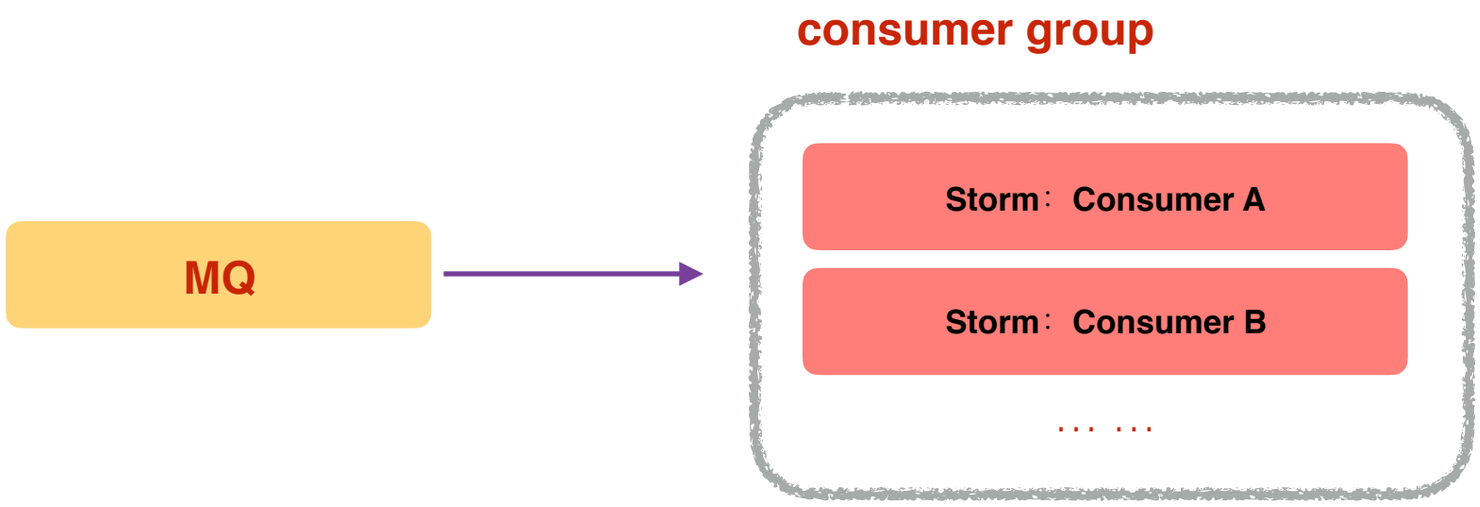


- **解决方式**：topic
  - 同一种数据放入同一个 topic
  - 不同数据，通过 topic 分离

- **问题2**：消息的单播、多播?
  - 单播：一条 msg，一个 consumer
  - 多播：一条 msg，多个 consumer

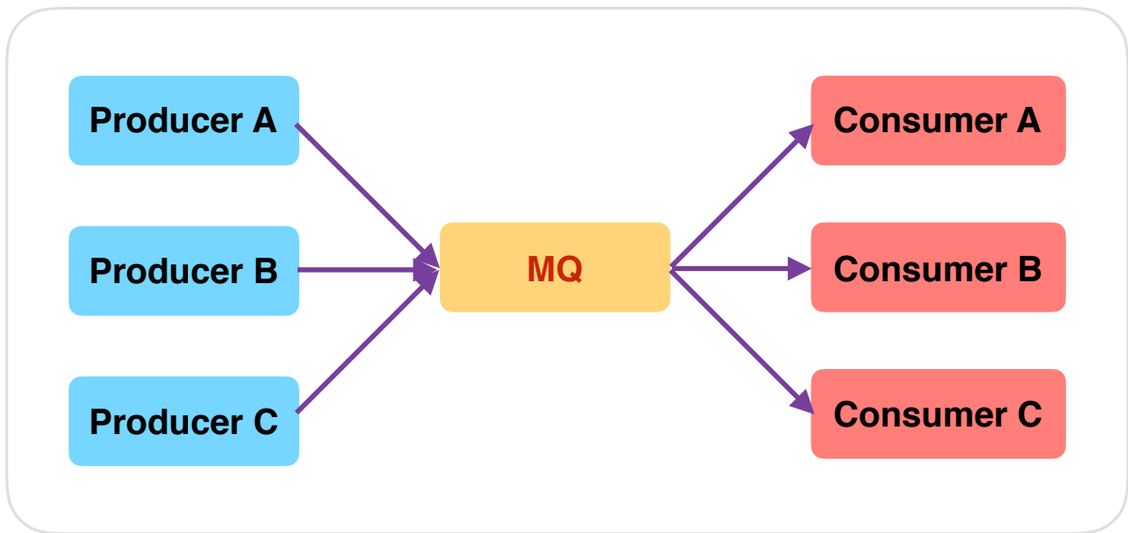


- **解决方式**： consumer group
  - 一个 msg，送入多个 consumer group
  - 一个 consumer group 中，只能有一个 consumer 处理某一条 msg



- **问题3**：并发效率
  - 同一类业务数据，多 producer，多 consumer
  - 并行处理?

- **解决方式**： partition
  - 一个 topic，划分为 多个 partition
  - partition 之间能够并行处理



Producer

Producer

Producer

partition A

		6	5	4	3	2	1	0

partition B

		4	3	2	1	0

... ...

Topic A

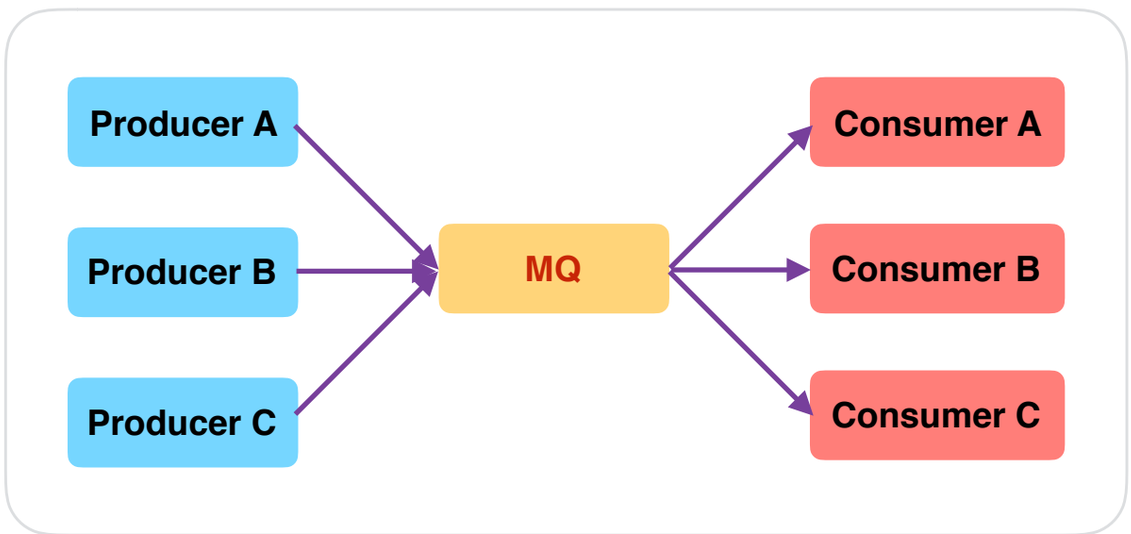
Consumer

Consumer

Consumer

- **问题4**：msg 之间的顺序保证
  - 推荐系统中，时间序列挖掘
  - 严格限制，msg 之间的先后顺序

- **解决办法**： partition 内部有序
  - 将有顺序要求的 msg，送入同一个 partition
  - 将时间戳，放入 msg 内



Producer

Producer

Producer

partition A

		6	5	4	3	2	1	0

partition B

		4	3	2	1	0

... ...

Topic A

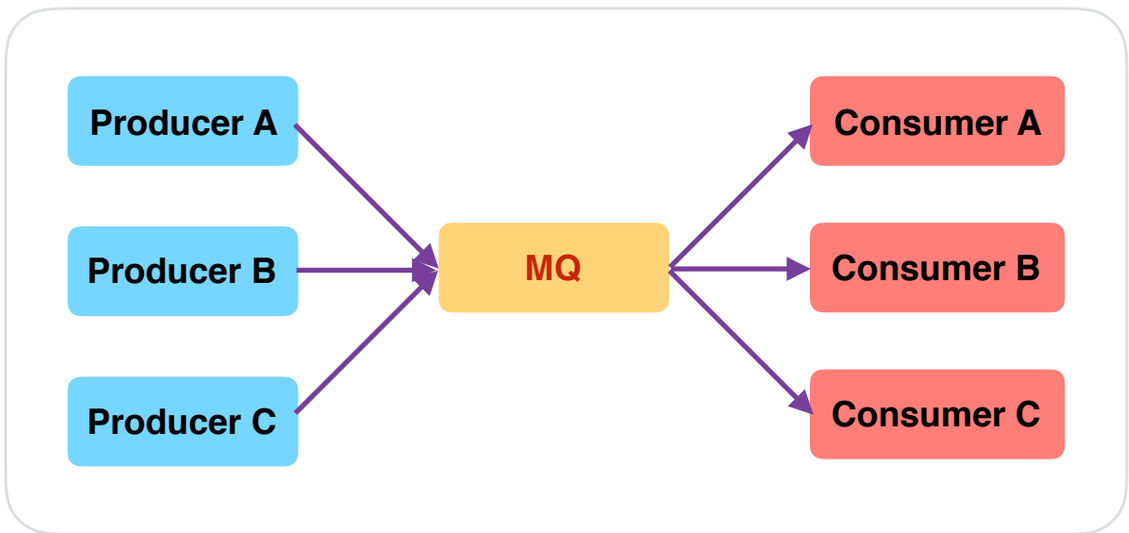
Consumer

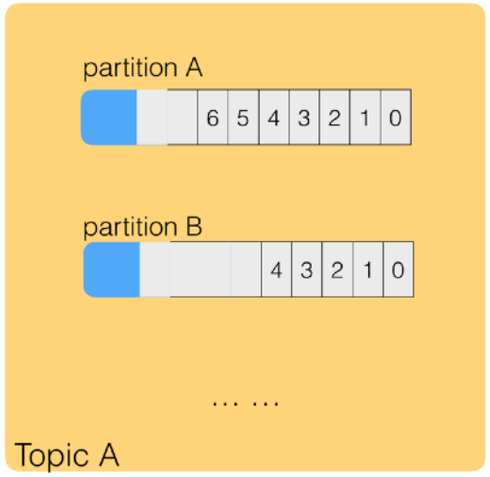
Consumer

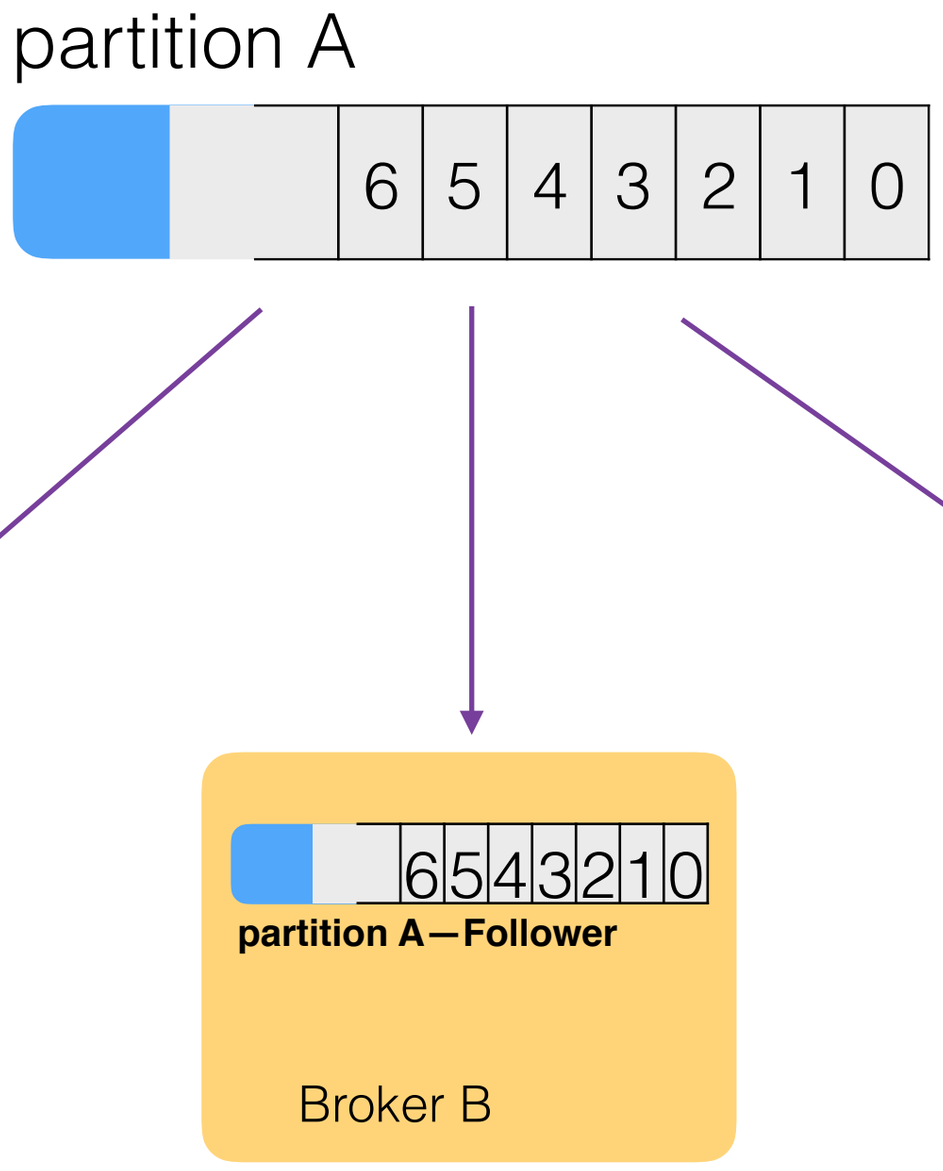
Consumer

- **问题5**：Kafka 的可靠性
  - Kafka 集群，由broker组成
  - 某个 broker 宕掉，数据是否会丢失?

- **解决办法**： replica
  - 每个 partion 都存储多份，分布在不同 broker 上
  - replica 的角色，分为 leader、follower



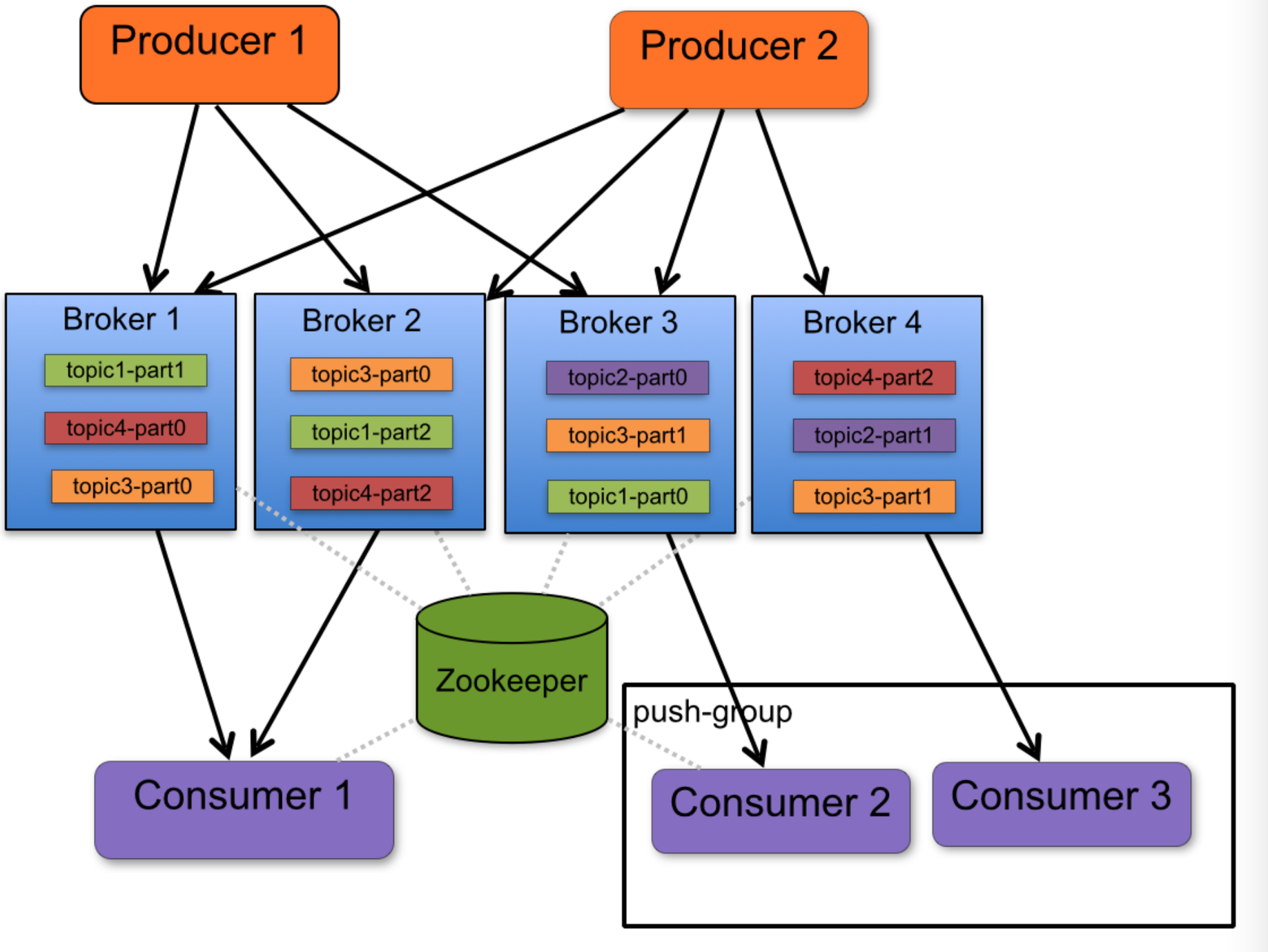

partition A
6 5 4 3 2 1 0
partition B
4 3 2 1 0
… …
Topic A


partition A
6 5 4 3 2 1 0
6 5 4 3 2 1 0
partition A—Leader
Broker A
6 5 4 3 2 1 0
partition A—Follower
Broker B
6 5 4 3 2 1 0
partition A—Follower
Broker C

小结

- 整体结构：
  - **producer**: 向kafka指定topic发送数据的程序称作producer
  - **consumer**: 从kafka指定topic消费数据的程序称作consumer
  - **broker**: kafka集群中每个实例称作 个broker，由唯 id标识
- 核心原理：
  - **topic**: kafka维护的消息种类,每一类消息由一个topic标识
  - **consumer group**：解决单播、多播问题
  - **partition**: 每个topic可以分成多个区，分布在不同的broker上
  - **replica**: 每个topic可以设置副本数，所有的副本称作replica
    - **leader**: 所有的副本中只有leader处理读写，其他的follower从leader同步
    - **isr**: replica中能够跟上leader的实例称作isr

小结

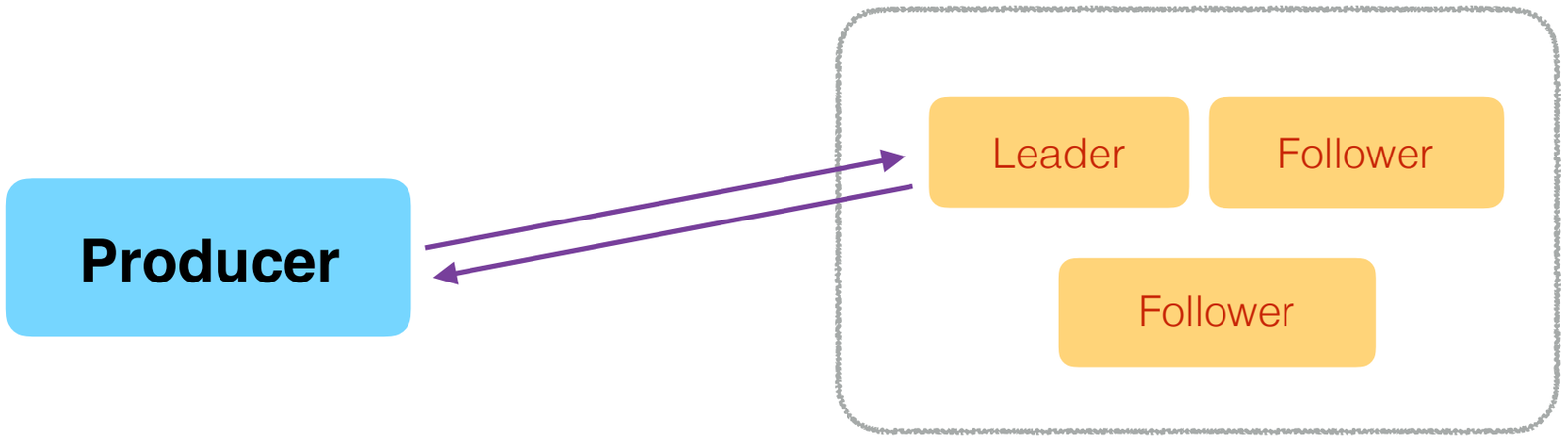

Producer 1
Producer 2
Broker 1
topic1-part1
topic4-part0
topic3-part0
Broker 2
topic3-part0
topic1-part2
topic4-part2
Broker 3
topic2-part0
topic3-part1
topic1-part0
Broker 4
topic4-part2
topic2-part1
topic3-part1
Zookeeper
push-group
Consumer 1
Consumer 2
Consumer 3

Kafka 关键实现细节

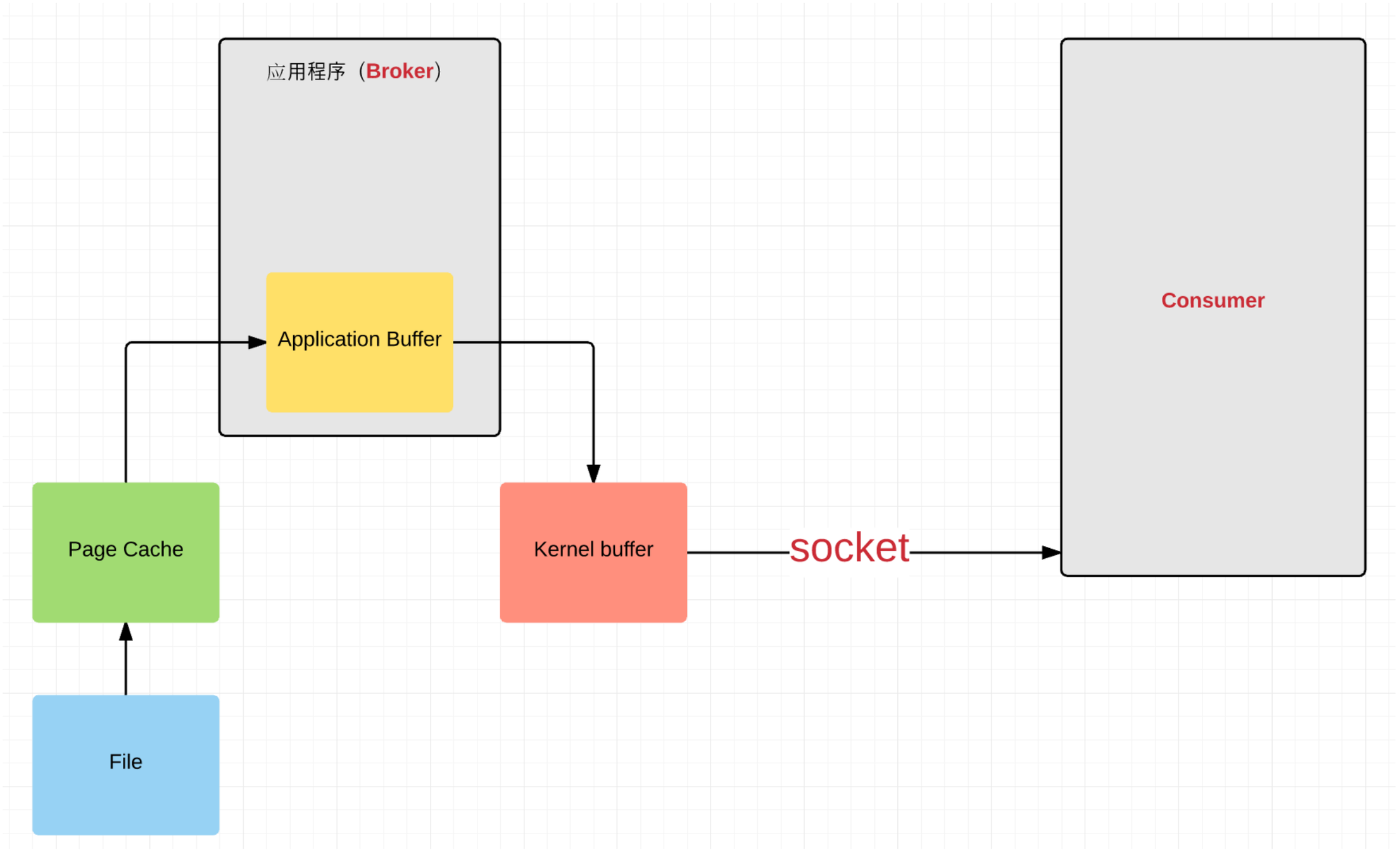
Kafka 相对其他 MQ 的关键实现细节？

- Producer：
  - 支持同步复制和异步复制
  - 批量发送：Nagle 策略，利用缓冲区、超时时间，合并小的数据包 （batch.size，16KB）
    - 减少网络IO 次数
    - 增加有效 payload 比例，提升有效吞吐量
- Broker：
  - 利用sendfile系统调用，zero-copy，批量传输数据
  - 消息磁盘持久化，不在内存中 cache，充分利用磁盘顺序读写优势
    - broker 没有 cache 压力，因此，更适合支持 pull 模式
- Consumer：
  - pull 模式，broker 无状态，consumer 自己保存 offset
    - 配置过期时长（broker）
    - 多次回放数据

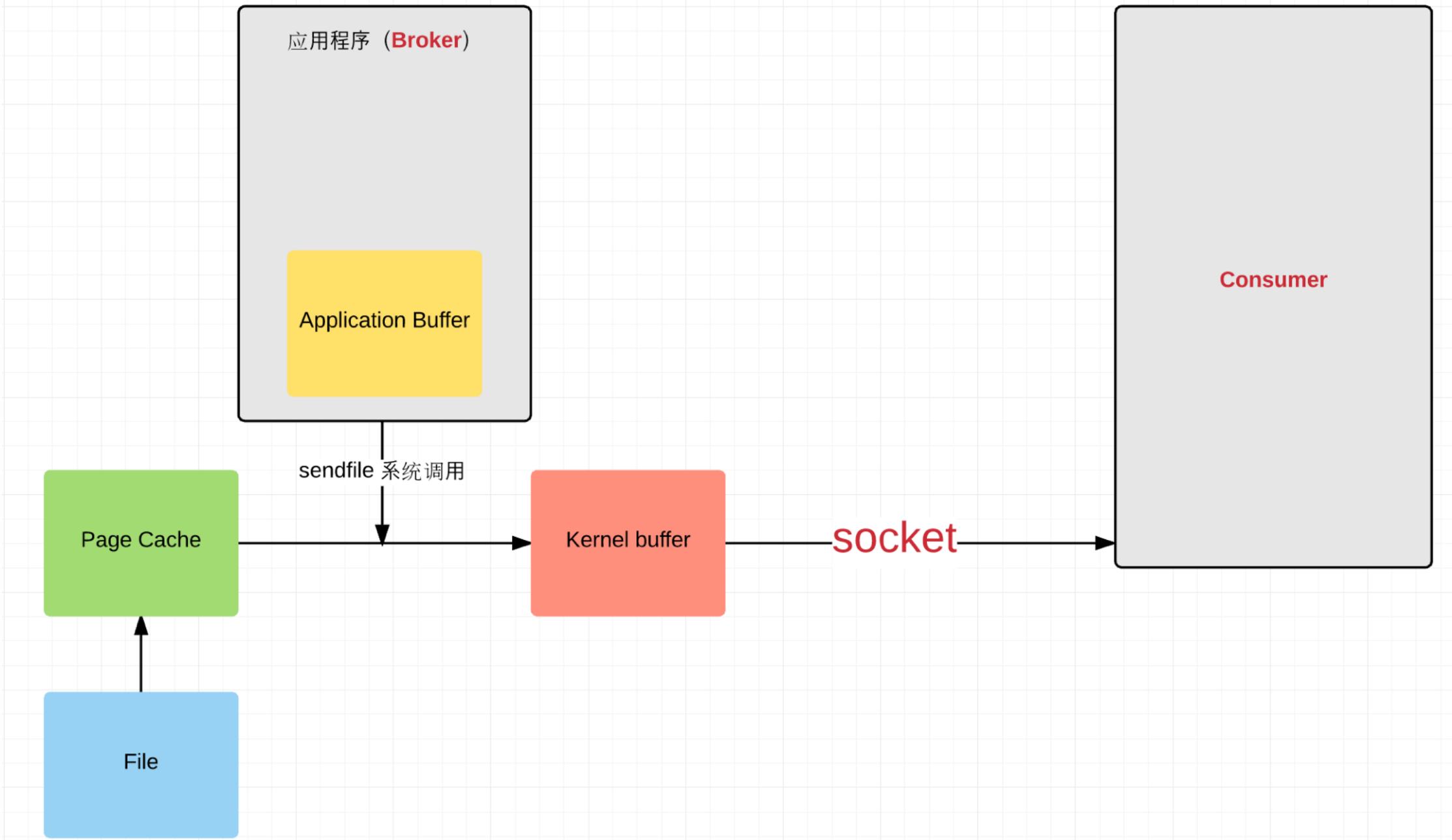
Kafka 典型特点— 同步复制、异步复制



Kafka 典型特点— sendfile 系统调用

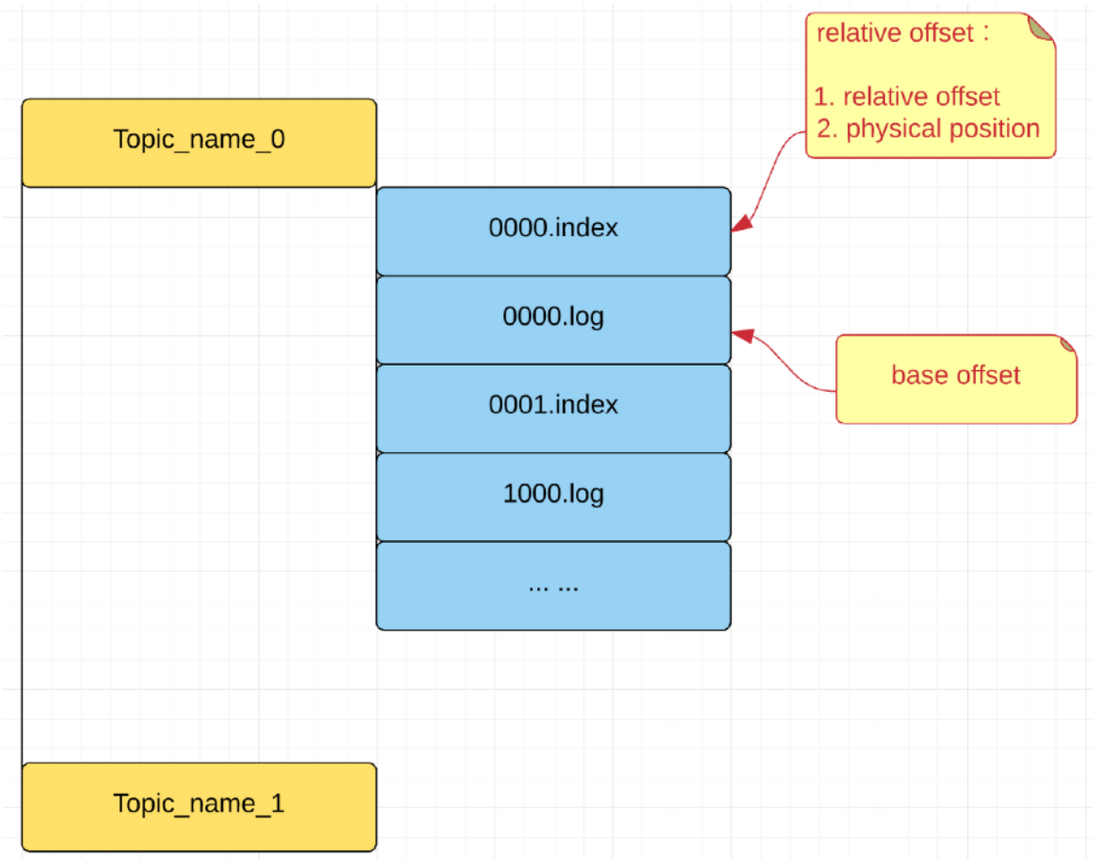


Kafka 典型特点— sendfile 系统调用



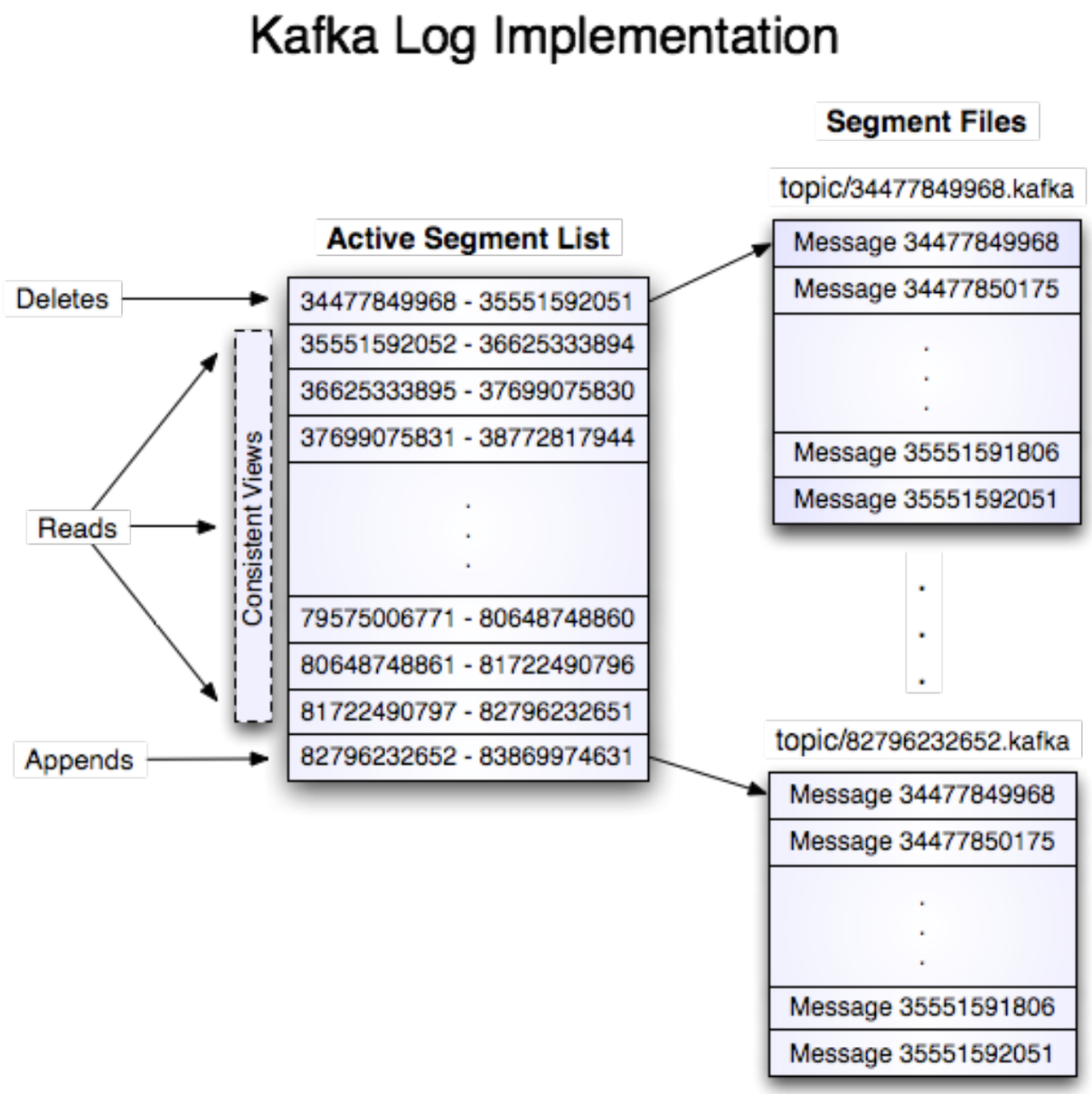
Kafka 典型特点— 磁盘持久化

- Kafka - topic - partition 的存储关系：
  - 一个 partition 对应一个文件夹
  - 一个 partition 分为多个 segment
    - segment 命名：offset.log
    - segment 对应一个 index 文件



- 核心问题：如何快速 seek 到任意 offset

Kafka 典型特点— 磁盘持久化



小结

- Kafka 关键实现细节？
  - **Producer**：
    - 支持同步复制和异步复制
    - 批量发送（Nagle 策略）
  - **Broker**：
    - sendFile 系统调用
    - 磁盘持久化：分段+索引
  - **Consumer**：
    - pull模式
    - broker 无状态

Kafka 使用实践

- 使用实践：
  - 吞吐量 *vs.* 数据实时性：API 简介
  - 并发效率：数据倾斜
- 数据丢失，典型问题：
  - 数据丢失 1：producer 侧
  - 数据丢失 2：consumer 侧
- Mobike 内部实例：
  - ElasticSearch Consumer

Kafka API 简介：吞吐量 vs. 实时性

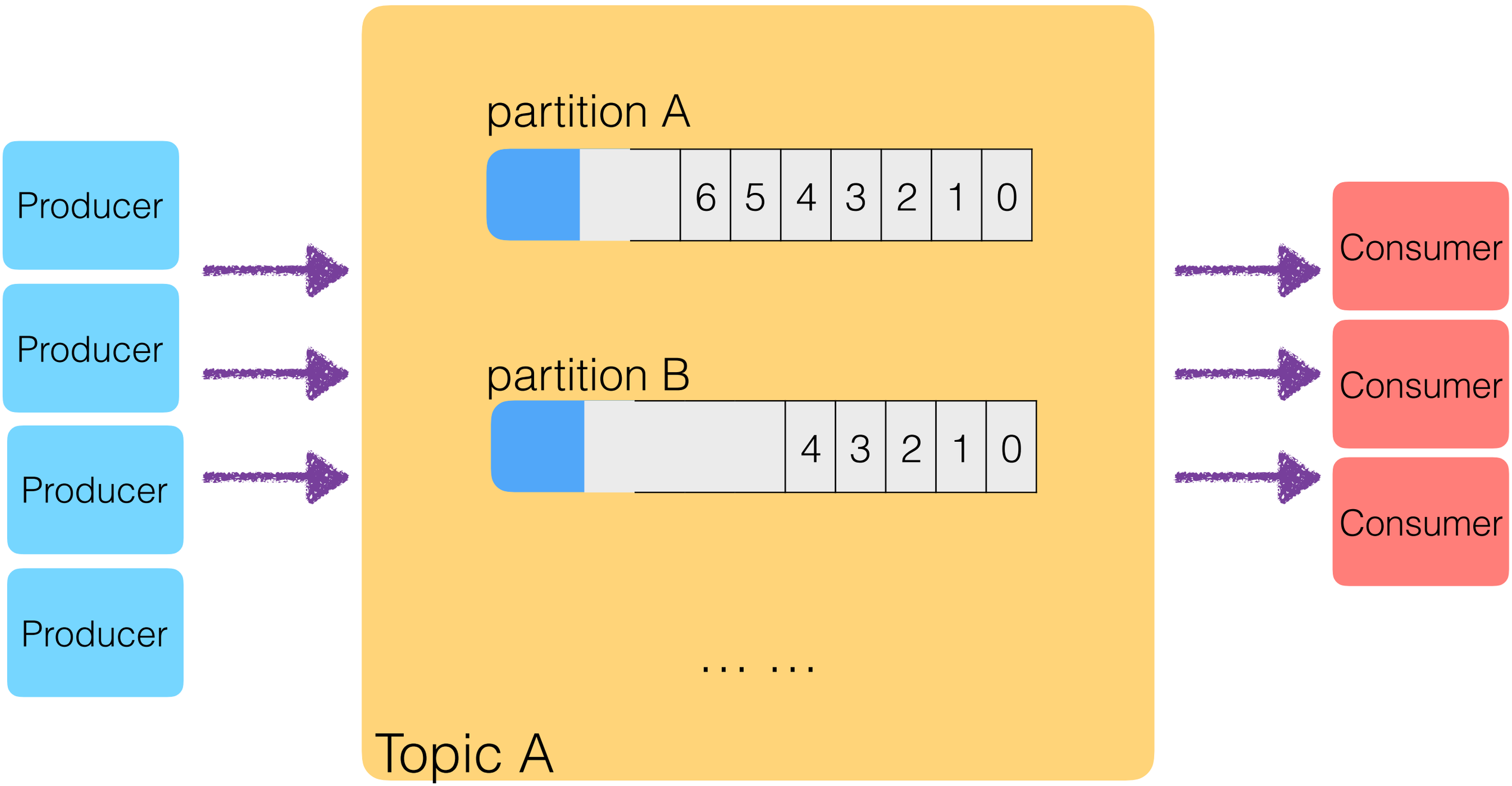
- Producer API:

```
props.put("bootstrap.servers", "localhost:4242");
props.put("acks", "all");
props.put("retries", 0);
props.put("batch.size", 16384);
props.put("linger.ms", 1);
props.put("buffer.memory", 33554432);
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
```

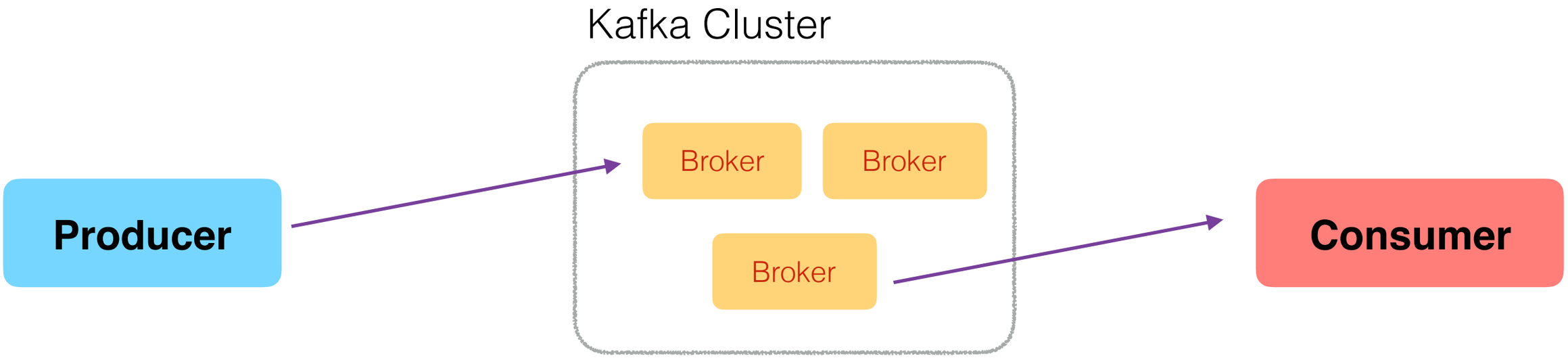
- Consumer API:
  - old Consumer API：scala 语言
    - old High-level Consumer API
    - old Simple Consumer API
  - new Consumer API：Java 语言（0.9.0+）

```
Properties props = new Properties();
props.put("zookeeper.connect", a_zookeeper);
props.put("group.id", a_groupId);
props.put("zookeeper.session.timeout.ms", "400");
props.put("zookeeper.sync.time.ms", "200");
props.put("auto.commit.interval.ms", "1000");
```

并发效率：数据倾斜


Producer
Producer
Producer
Producer
Producer
partition A
6 5 4 3 2 1 0
partition B
4 3 2 1 0
... ...
Topic A
Consumer
Consumer
Consumer

数据丢失典型场景

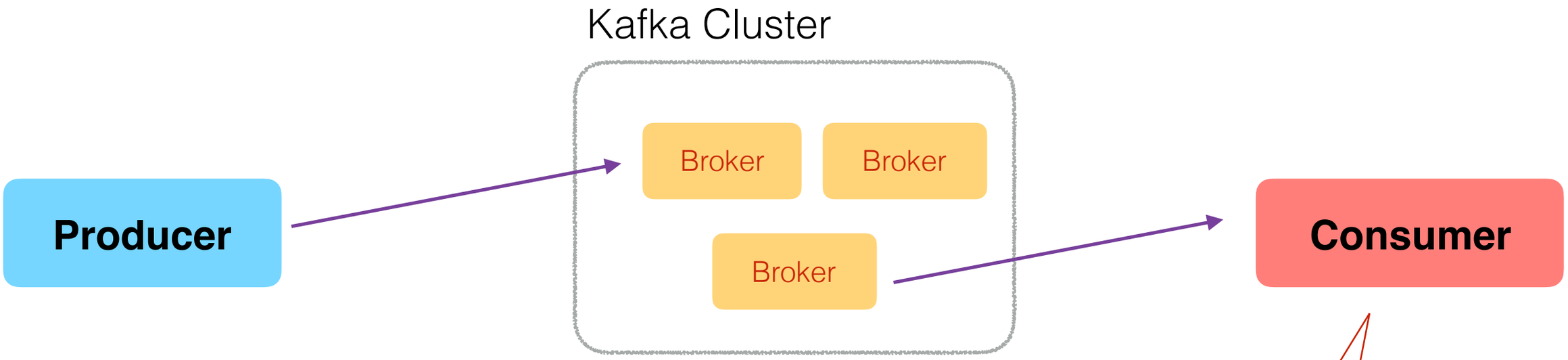


消息分发语义	含义	补充
at least once	至少一次	Kafka 默认语义
at most once	至多一次	
exactly once	精确一次	Kafka 0.11.0 开始支持

数据丢失典型场景

- 1. Producer 发送数据：
  - 一个 broker 收到数据后，返回 ack，但在向其余 broker 同步数据之前，当前节点宕机，数据丢失
- 2. Consumer 处理数据：
  - consumer 获取数据后，告知 broker 数据获取成功，此后，consumer 处理数据期间宕机，数据丢失
- 3.特别说明：
  - 合理配置 Producer 和 Consumer ，会保证：at least once 语义，数据不会丢失

数据丢失典型场景



Offset 无效时，默认会「批量丢弃数据」：

1. 参数：auto.offset.reset
2. 取值说明：earliest、latest*、none

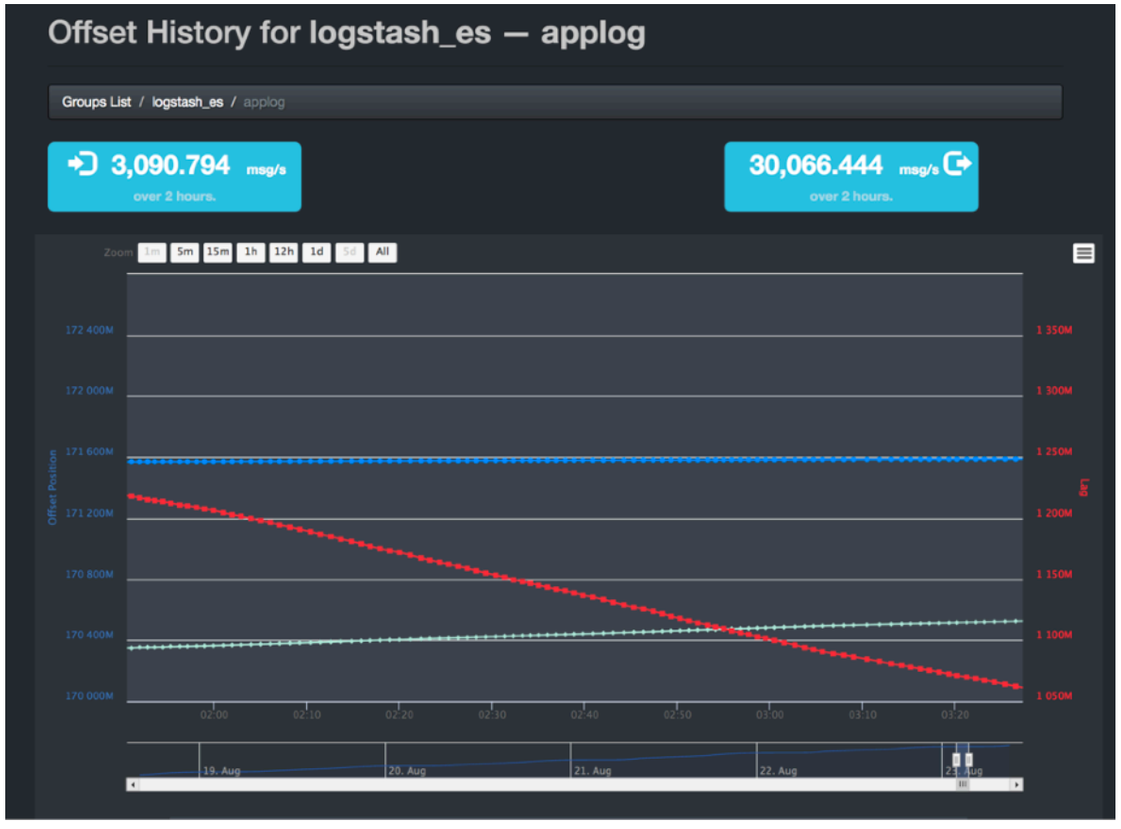
Mobike 内部实例：ElasticSearch

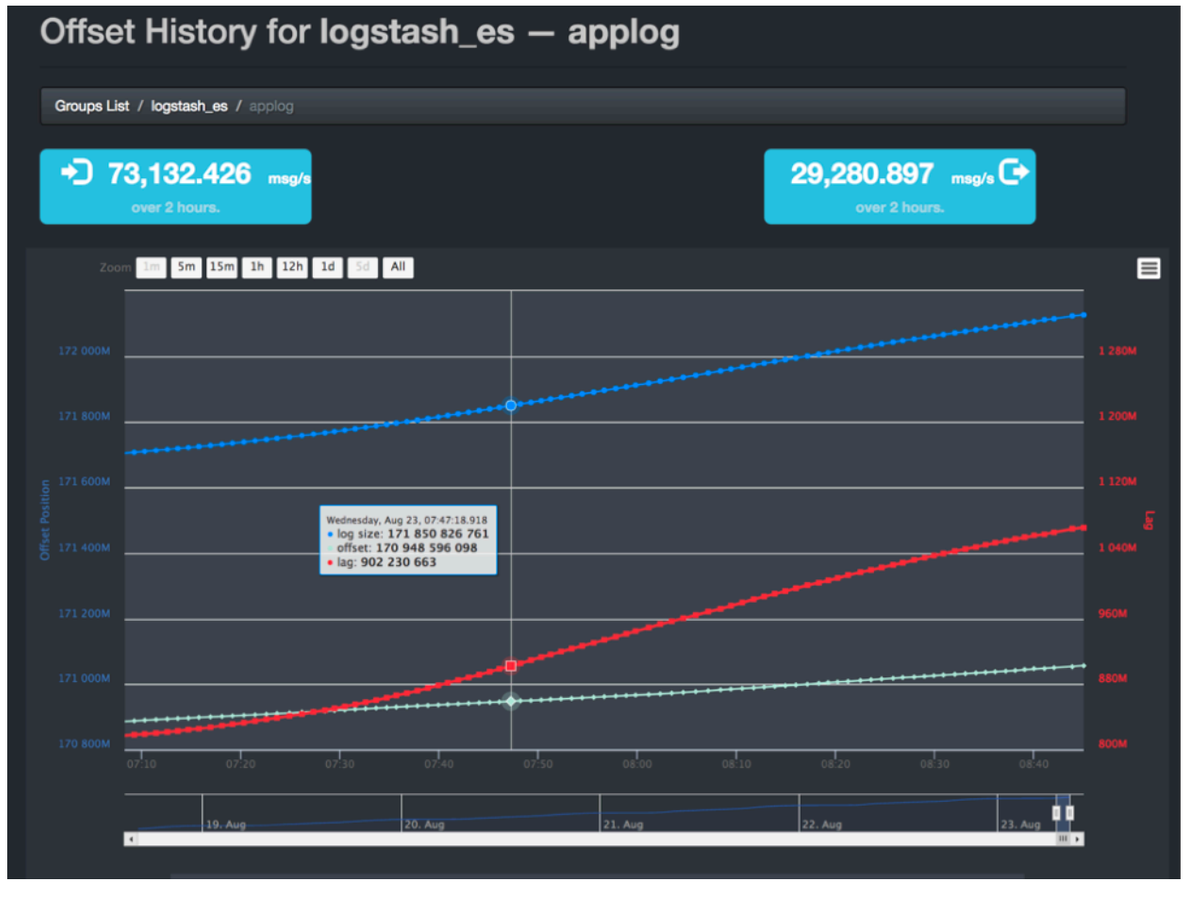
- 背景：
  - Kafka 上数据存活时间：24h
  - partition 数量：36
  - consumer 数量：3

生产者—消费者	业务低峰（2点~3点）	业务高峰（7点~8点）
生产者：applog	0.3w msg/s	7.3w msg/s
消费者：ElasticSearch	3w msg/s	2.9w msg/s

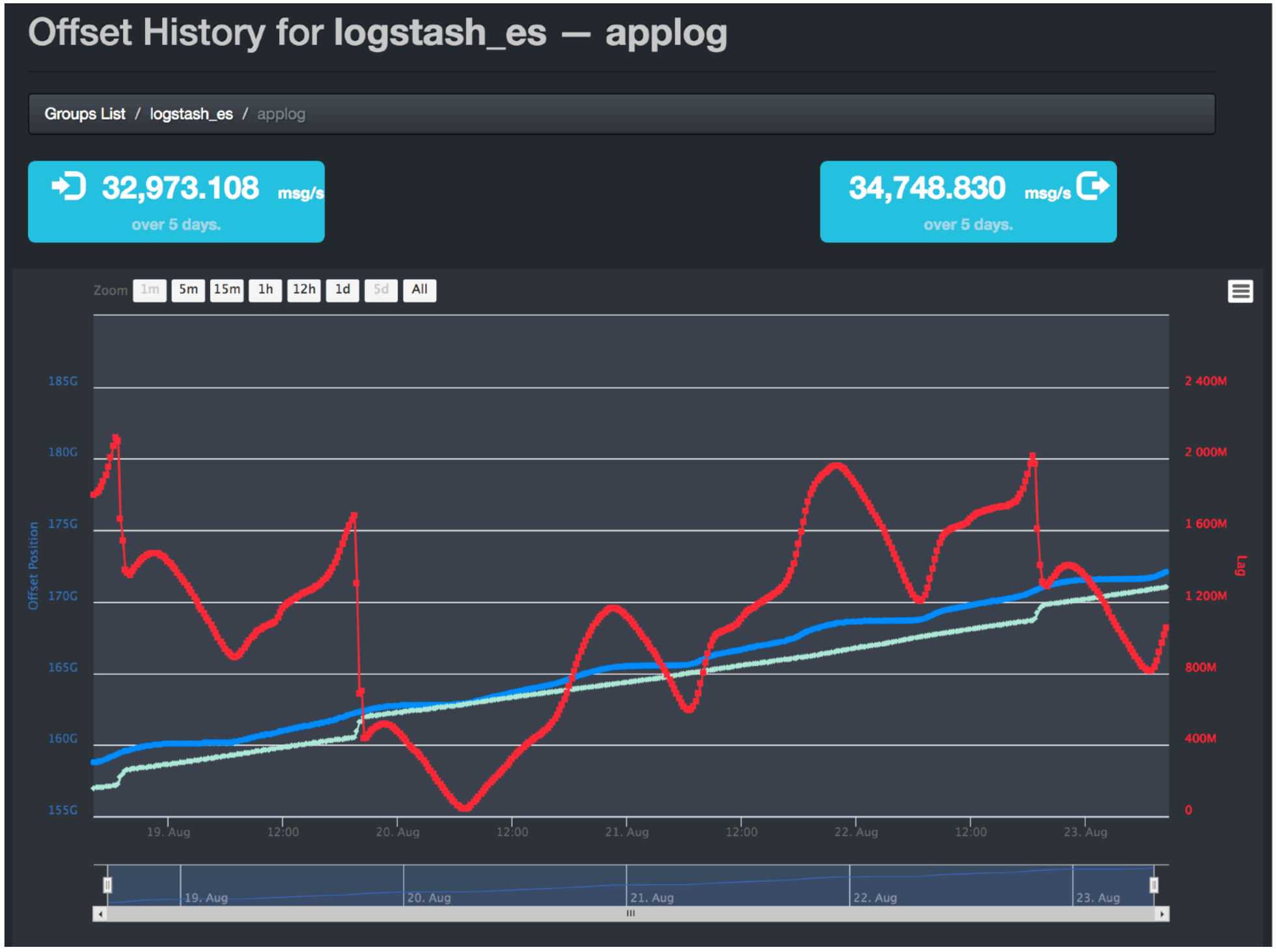
- 批量丢数据，**本质**：
  - Consumer 的**消费能力不足**，消息积压在 MQ 中，同时，会产生消费延迟。

Mobike 内部实例：ElasticSearch





Mobike 内部实例：ElasticSearch



小结

- Kafka API：吞吐量 vs. 实时性
- 数据倾斜：Producer 侧，调整路由策略
- 消息丢失：是否会丢消息，哪个地方会丢失?

全文小结

- MQ 的作用
- Kafka 的设计原理
  - Kafka 基本原理
  - Kafka 关键技术细节
- Kafka 实践

参考资料

- Top 10 Uses For A Message Queue
- Kafka: a Distributed Messaging System for Log Processing *NetDB'11, Jun. 12, 2011*
- http://kafka.apache.org/documentation/ Kafka 0.11.0 文档
- http://kafka.apache.org/0100/documentation.html Kafka 0.10.0 文档
- http://kafka.apache.org/090/documentation.html Kafka 0.9.0 文档
- http://kafka.apache.org/082/documentation.html Kafka 0.8.2 文档
- Kafka-技术文档-消息分发语义

附录

- Kafka vs. RocketMQ

- Kafka: stream & connect

- Kafka 集群规划

Kafka vs. RocketMQ

alibaba / RocketMQ

Watch 768 | Star 3,978 | Fork 2,234

Code | Issues 1 | Pull requests 15 | Projects 3 | Wiki | Insights

Compare: rmq_vs_kafka

New Page

Revert Changes

Showing 27 changed files with 657 additions and 173 deletions.

Unified | Split

122 ■■■■■ rmq_vs_kafka.md

View

```
... ...  @@ -1,122 +0,0 @@
1        -## RocketMQ与Kafka对比（18项差异）
2        -
3        -淘宝内部的交易系统使用了淘宝自主研发的Notify消息中间件，使用Mysql作为消息存储媒介，可完全水平扩容，为了进一步降低成本，我们认为存储部分可以进一步优化，2011年初，Linkin开源了Kafka这个优秀的消息中间件，淘宝中间件团队在对Kafka做过充分Review之后，Kafka无限消息堆积，高效的持久化速度吸引了我们，但是同时发现这个消息系统主要定位于日志传输，对于使用在淘宝交易、订单、充值等场景下还有诸多特性不满足，为此我们重新用Java语言编写了RocketMQ，定位于非日志的可靠消息传输（日志场景也OK），目前RocketMQ在阿里集团被广泛应用在订单，交易，充值，流计算，消息推送，日志流式处理，binglog分发等场景。
4        -
5        -为了方便大家选型，整理一份RocketMQ与Kafka的对比文档，文中如有错误之处，欢迎来函指正。[vintage.wang@gmail.com](vintage.wang@gmail.com)
6        -
7        -
```

Kafka vs. RocketMQ

RocketMQ vs. ActiveMQ vs. Kafka

Messaging Product	Client SDK	Protocol and Specification	Ordered Message	Scheduled Message	Batched Message	BroadCast Message	Message Filter	Server Triggered Redelivery	Message Storage	Message Retroactive	Message Priority	High Availability and Failover	Message Track	Configuration	Management and Operation Tools
ActiveMQ	Java, .NET, C++ etc.	Push model, support OpenWire, STOMP, AMQP, MQTT, JMS	Exclusive Consumer or Exclusive Queues can ensure ordering	Supported	Not Supported	Supported	Supported	Not Supported	Supports very fast persistence using JDBC along with a high performance journal, such as levelDB, kahaDB	Supported	Supported	Supported, depending on storage,if using kahadb it requires a ZooKeeper server	Not Supported	The default configuration is low level, user need to optimize the configuration parameters	Supported
Kafka	Java, Scala etc.	Pull model, support TCP	Ensure ordering of messages within a partition	Not Supported	Supported, with async producer	Not Supported	Supported, you can use Kafka Streams to filter messages	Not Supported	High performance file storage	Supported offset indicate	Not Supported	Supported, requires a ZooKeeper server	Not Supported	Kafka uses key-value pairs format for configuration. These values can be supplied either from a file or programmatically.	Supported, use terminal command to expose core metrics
RocketMQ	Java, C++, Go	Pull model, support TCP, JMS, OpenMessaging	Ensure strict ordering of messages,and can scale out gracefully	Supported	Supported, with sync mode to avoid message loss	Supported	Supported, property filter expressions based on SQL92	Supported	High performance and low latency file storage	Supported timestamp and offset two indicates	Not Supported	Supported, Master-Slave model, without another kit	Supported	Work out of box,user only need to pay attention to a few configurations	Supported, rich web and terminal command to expose core metrics

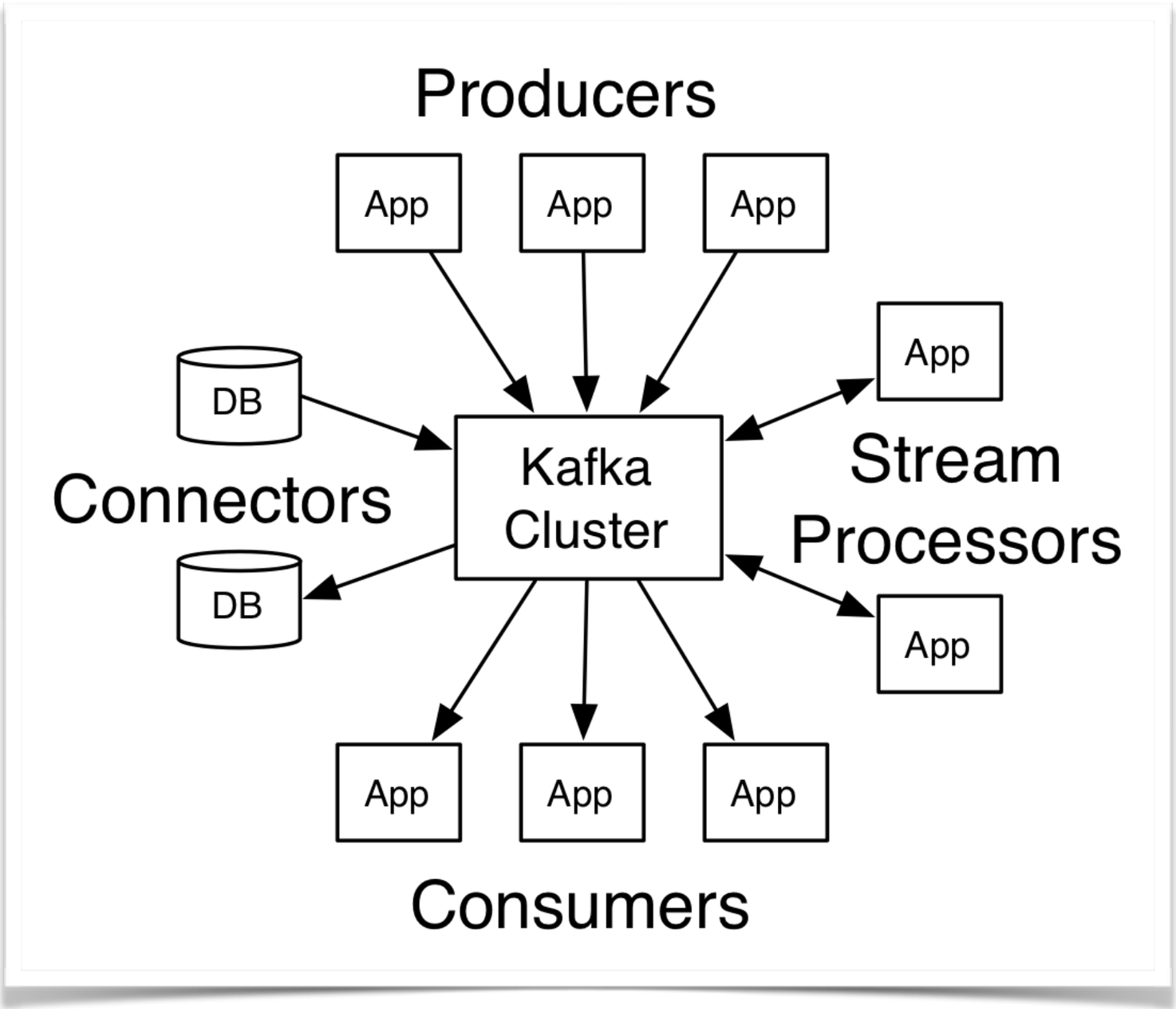
Kafka *vs.* RocketMQ

- 详细资料：

  - http://rocketmq.apache.org/docs/motivation/

  - https://github.com/alibaba/RocketMQ/wiki/rmq_vs_kafka 18 个差异

Kafka: Stream & Connect



Kafka：Stream & Connect

- **Kafka 最新组件**：stream & connect
  - 版本：**Kafka 0.10.0** 引入 stream 和 connect
- **stream**
  - **作用**：从 topic A 读取数据，处理之后，送入 topic B
  - **定位**：
    - 提供了通用操作的封装，例如聚合、过滤，编写 Processor Topology
    - 数据源：Kafka、外部数据源（微服务）
  - **特点**：数据还写回 Kafka
- **connect**：
  - **作用**：从外部数据源，向 Kafka 导入数据；将 Kafka 数据，导出
  - **定位**：
    - 数据同步，增量数据，也会同步，connect 运行后，会一直存在

Kafka 集群规划

- 容量规划 & 部署结构 & 性能调优：
  - 网络、磁盘
  - Topic 下 partition 数量、单 broker 上 partition 数量
- 性能监控：
  - OS 级别
  - Topic-Consumer 级别