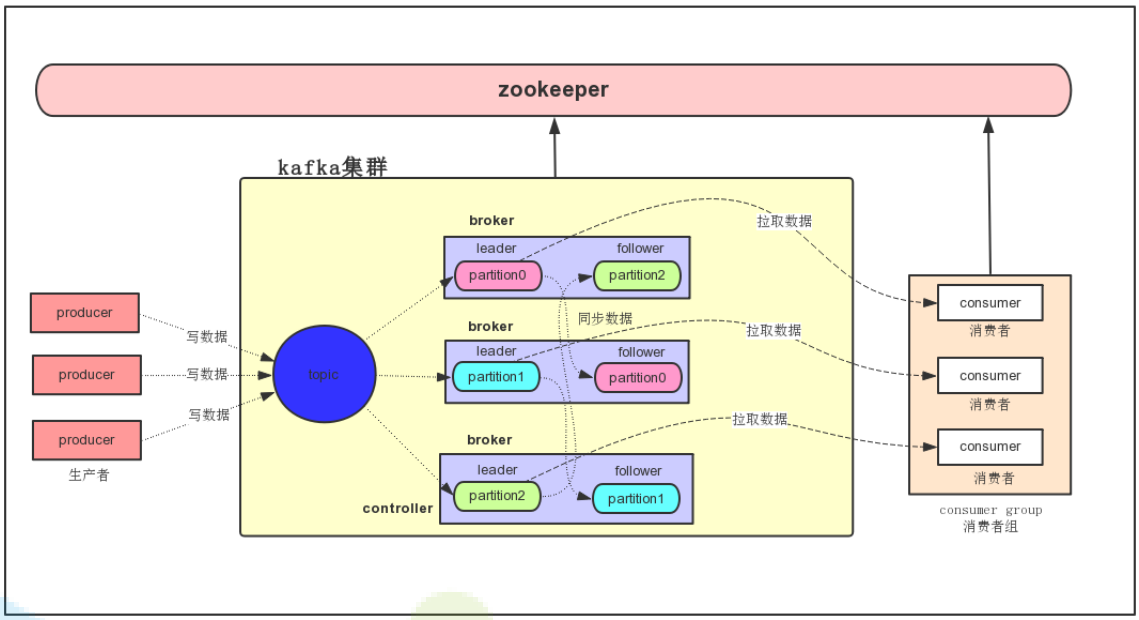
Kafka笔记

1 为什么有消息系统

异步、解耦、流量削峰

2 Kafka系统架构



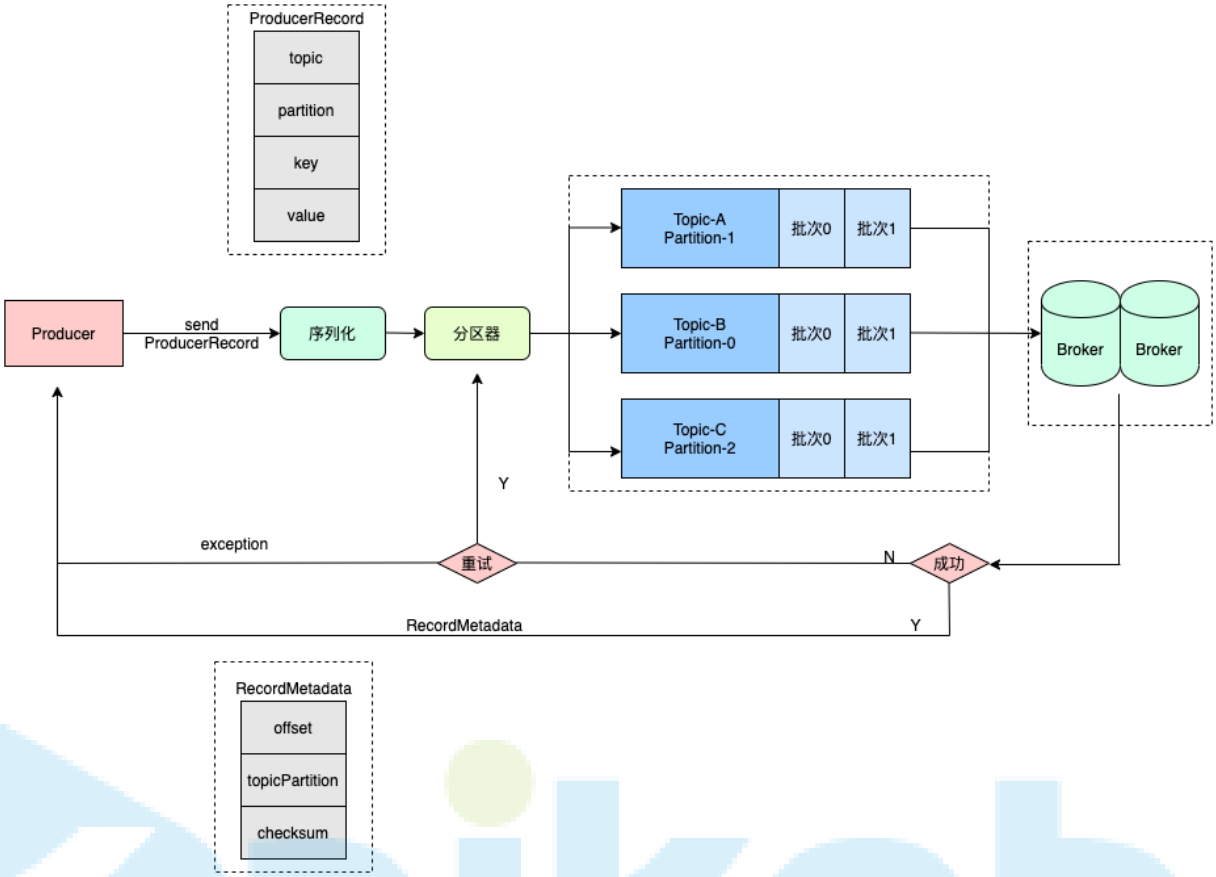
3 Kafka高吞吐率实现

- 顺序读写：Kafka将消息写入到了分区partition中，而分区中消息是**顺序读写**的。磁盘顺序读写性能要远高于内存的随机读写。
- 零拷贝：生产者、消费者对于kafka中消息的操作是采用**零拷贝**实现的。
- 批量发送：Kafka允许使用批量消息发送模式。
- 消息压缩：Kafka支持对消息集合进行压缩。
- Page Cache(页缓存):

4 ZooKeeper作用

ZooKeeper 一种常用的**分布式协调服务**，Kafka 用到了它的**数据发布和订阅、命名服务、分布式协调/通知、集群管理、Master 选举**等功能。

5 Kakfa发送流程



6 Kafka消费者offset提交方式

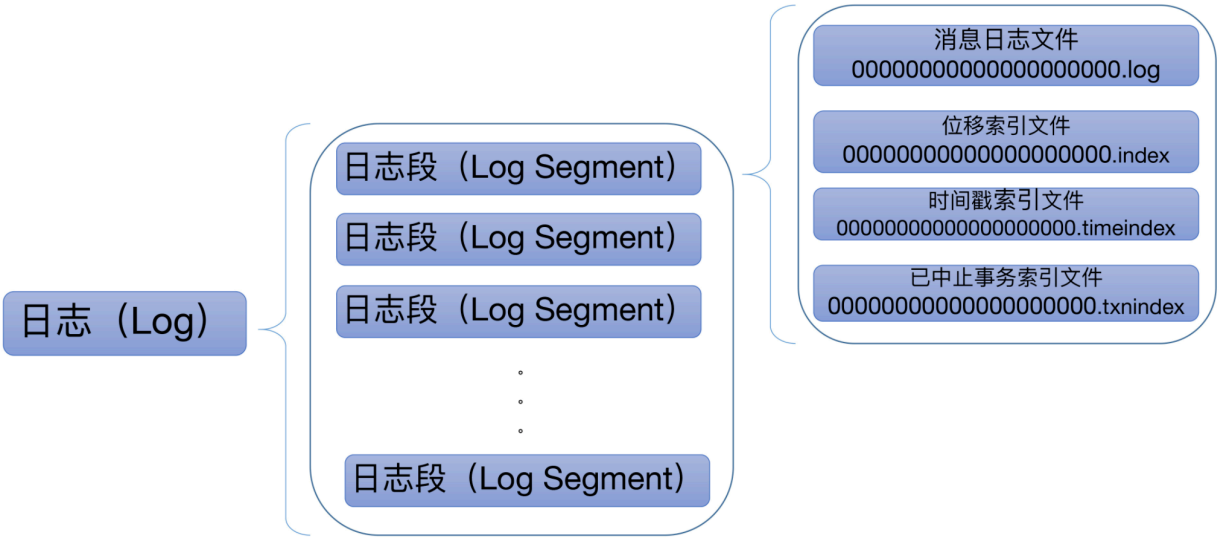
提交方式		实现方式
自动提交	自动提交	props.put("enable.auto.commit", "true");
手动提交	同步提交	consumer.commitSync();
	异步提交	consumer.commitAsync();

7 partition的作用

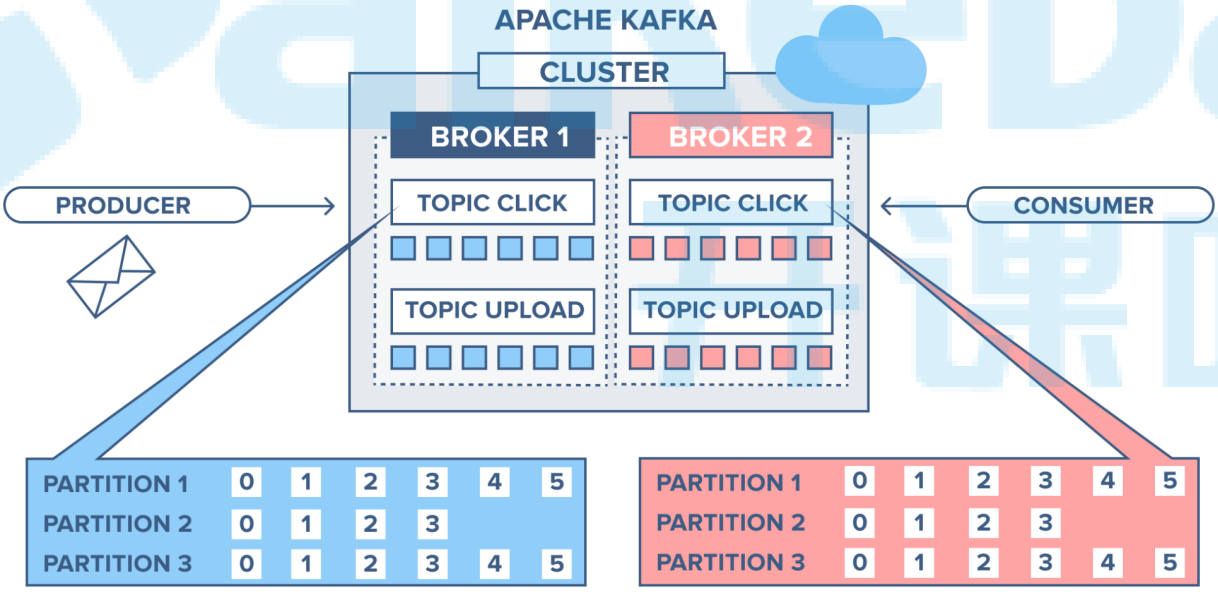
消息存储扩容：一个文件的存储大小是有限的，但在集群中的多个文件的存储就可以大大增加一个topic能够保存的消息数量。

并行读写：通过多个partition文件存储消息，意味着producer和consumer可以并行的读写一个topic。

8 segment文件组成



9 Broker、Topic、Partition 及 Segment 间的关系



一个 Topic 的消息可以被存放到多个 Broker 中，一个 Broker 中可以存放一个 Topic 的多 个 Partition，而一个 Partition 中可以存放很多的 Segment，一个 Segment 中可以存放很多的消息。

10 Kakfa发送者分区选择策略

Producer负责将消息发送到Kafka集群的某一个topic中。同时Producer发送消息时能够指定partition号，从而将消息持久化到特定的partition中。

- 如果没有指定具体的partition号，那么Kafka Producer可以通过一定的算法计算出对应的

partition号。

- 如果消息指定了key，则对key进行hash，然后映射到对应的partition号
- 如果消息没有指定key，则使用Round Robin轮询算法来确定partition号，这样可以保证数据在所有的partition上平均分配。

另外，Kafka Producer也支持自定义的partition分配方式。客户端提供一个实现了`org.apache.kafka.clients.producer.Partitioner`的类，然后将此实现类配置到Producer中即可。

11 总结Kafka中Consumer Group中的Consumer与其要消费的Topic中的partition间的数量关系。

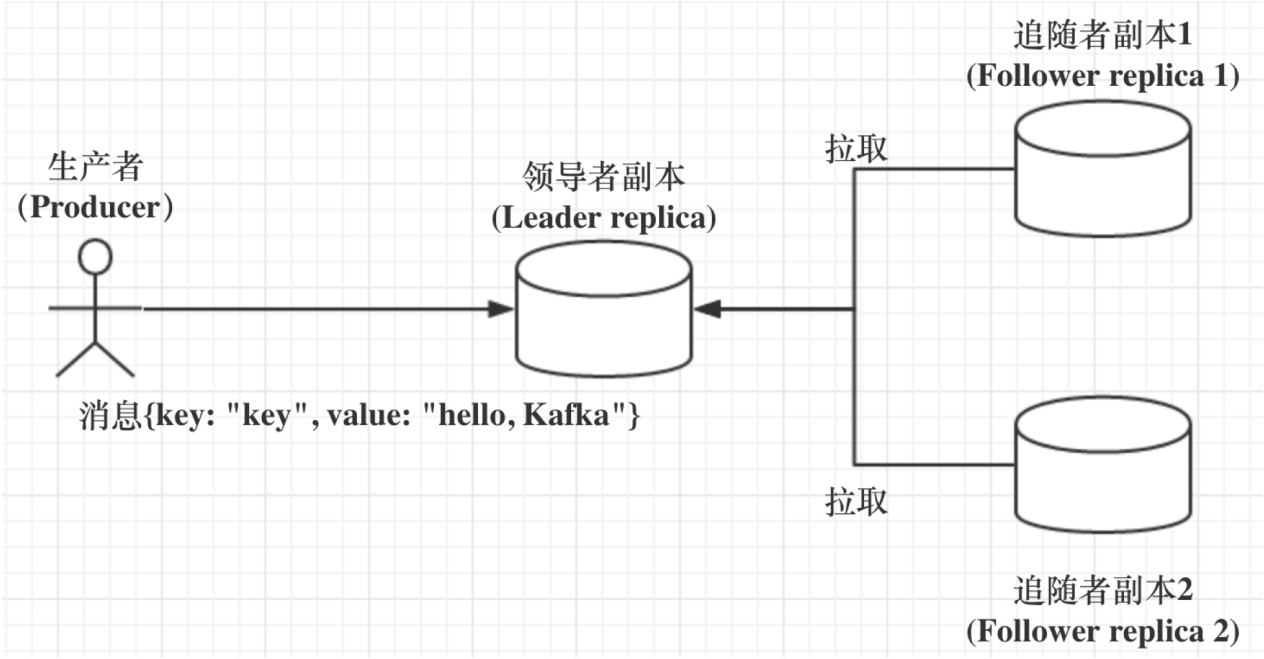
Consumer Group中组内consumer与其要消息的Topic的partition的关系是1:n。在稳定状态下，一旦为某组内 consumer 分配了某一个或几个 partition 后，就不会发生变化了;反过来说，一旦为某 partition 分配 了组内 consumer，就不会再为其分配其它组内 consumer 了。

12 Kafka 中一个 partition 只允许一个 Consumer Group 中的一个组内 Consumer 进 行消费。请简述这样设计的优劣。

Kafka 中一个 partition 只允许一个 Consumer Group 中的一个组内 Consumer 进 行消费。而一个组内 Consumer 可以消费同一 Topic 的多个 parittion。这样设计的好处是，对 于每个 parition 中消费消费的偏移量控制简单。不足是，无法让同一个组内的 consumer 均匀消费消息，因为消息在同一 Topic 的 parition 中的存放并不是平均的。一旦组内 consumer 与 parition 的消费关系确立，则可能会导致某些组内 consumer 需要消费的消息量很大，有的组内 consumer 可能无消息可消费。

13 请描述Kafka副本工作机制

基于领导者的副本机制的工作原理如下图所示：



14 为什么追随者副本是不对外提供服务?

1.方便实现“Read-your-writes”。

所谓 Read-your-writes，顾名思义就是，当你使用生产者 API 向 Kafka 成功写入消息后，马上使用消费者 API 去读取刚才生产的消息。

2.方便实现单调读（Monotonic Reads）。

什么是单调读呢？就是对于一个消费者用户而言，在多次消费消息时，它不会看到某条消息一会儿存在一会儿不存在。

15 Rebalance 触发条件

1、组成员数发生变更。
2、订阅主题数发生变更。
3、订阅主题的分区数发生变更。

16 Rebalance 缺点

1、消费暂停
2、 Rebalance 的设计是所有 Consumer 实例共同参与，全部重新分配所有分区。
3、Rebalance 速度慢。

17 Controller作用

controller 除了是一个普通的 broker 之外，还是集群的总扛把子，它负责副本 leader 的选举、topic 的创建和删除、副本的迁移、副本数的增加、broker 上下线的管理等等。

18 消费分区分配算法

Kafka中提供了多重分区分配算法（PartitionAssignor）的实现：

RangeAssignor、RoundRobinAssignor、StickyAssignor

PartitionAssignor接口用于用户定义实现分区分配算法，以实现Consumer之间的分区分配。

Kafka默认采用RangeAssignor的分配算法。

19 消息写入算法

消息生产者将消息发送给broker，并形成最终的可供消费者消费的log，是一个比较复杂的过程。

1）producer向broker集群提交连接请求，其所连接上的任意broker都会向其发送**broker controller**的通信URL，即broker controller主机配置文件中的listeners地址

2） 当producer指定了要生产消息的topic后，其会向broker controller发送请求，请求当前topic中所有partition的leader列表地址

3） broker controller在接收到请求后，会从zk中查找到指定topic的所有**partition的leader**，并返回给producer

4） producer在接收到leader列表地址后，根据消息路由策略找到当前要发送消息所要发送的partition leader，然后将消息发送给该leader

5） leader将消息写入本地log，并通知ISR中的followers

6） ISR中的followers从leader中同步消息后向leader发送ACK

7） leader收到所有ISR中的followers的ACK后，增加HW，表示消费者已经可以消费到该位置了

8) 若leader在等待的followers的ACK超时了，发现还有follower没有发送ACK，则会将该follower从ISR中清除，然后增加HW。

20 高水位的作用

HW，HighWatermark，高水位，表示Consumer可以消费到的最高partition偏移量。

1）定义消息可见性，即用来标识分区下的哪些消息是可以被消费者消费的。

2) 帮助 Kafka 完成副本同步,保证leader和follower之间的数据一致性。

21 发送的可靠性机制

生产者向kafka发送消息时，可以选择需要的可靠性级别。通过acks参数的值进行设置。

acks=0

该方式效率最高，吞吐量高，但可靠性最低。其可能会存在消息丢失的情况。

acks=1

只要集群的 Leader 节点收到消息，生产者就会收到一个来自服务器的成功响应。

acks=all

这种模式可靠性最高，很少出现消息丢失的情况。但可能会出现部分follower重复接收的情况

22 消费过程解析

生产者将消息发送到topic中，消费者即可对其进行消费，其消费过程如下：

1) consumer向broker集群提交连接请求，其所连接上的任意broker都会向其发送broker controller的通信URL，即broker controller主机配置文件中的listeners地址

2) 当consumer指定了要消费的topic后，其会向broker controller发送poll请求

3) broker controller会为consumer分配一个或几个partition leader，并将该partitioin的当前offset发送给consumer

4) consumer会按照broker controller分配的partition对其中的消息进行消费

5) 当消费者消费完该条消息后，消费者会向broker发送一个该消息已被消费的反馈，即该消息的offset

6) 当broker接到消费者的offset后，会更新到相应的__consumer_offset中

7) 以上过程一直重复，直到消费者停止请求消息

8) 消费者可以重置offset，从而可以灵活消费存储在broker上的消息

23 消息投递语义

Kafka支持三种消息投递语义

- **At most once**：至多一次，消息可能会丢，但不会重复
- **At least once**：至少一次，消息肯定不会丢失，但可能重复
- **Exactly once**：有且只有一次，消息不丢失不重复，且只消费一次。

24 重复消费的场景

在配置自动提交，出现重复消费的场景有以下几种：

1、消息在阈值时间内没有消费完毕

2、Consumer 在消费过程中，应用进程被强制kill掉或发生异常退出。

3、Rebalance场景下

25 高效文件存储设计特点

- Kafka把topic中一个parition大文件分成多个小文件段，通过多个小文件段，就容易定期清除或删除已经消费完文件，减少磁盘占用。
- 通过索引信息可以快速定位message
- 通过index元数据全部映射到memory，可以避免segment file的IO磁盘操作。
- 通过索引文件稀疏存储，可以大幅降低index文件元数据占用空间大小。
- 顺序写: 操作系统每次从磁盘读写数据的时候，需要先寻址，也就是先要找到数据在磁盘上的物理位置，然后再进行数据读写，如果是机械硬盘，寻址就需要较长的时间。kafka的设计中，数据其实是存储在磁盘上面，一般来说，会把数据存储在内存上面性能才会好。但是kafka用的是顺序写，追加数据是追加到末尾，磁盘顺序写的性能极高，在磁盘个数一定，转数达到一定的情况下，基本和内存速度一致

  随机写的话是在文件的某个位置修改数据，性能会较低。
- 零拷贝

26 日志清理策略

kafka log的清理策略有两种:**delete,compact**,默认是delete,

delete：一般是使用按照时间保留的策略，当不活跃的segment的时间戳是大于设置的时间的时候，当前segment就会被删除

compact: 日志不会被删除，会被去重清理，这种模式要求每个record都必须有key，然后kafka会按照一定的时机清理segment中的key，对于同一个key只保留最新的那个key.同样的，compact也只针对不活跃的segment6 监控工具