kafka

1. Kafka 是一个分布式的基于发布/订阅模式的可持久化的消息队列（Message Queue），主要应用于大数据实时处理领域。 （AB系统要通信，AB系统不直连，中间经过消息中间）
2. 消息队列的好处：解耦，可恢复性（部分失效不影响整体），缓冲，削峰，异步
3. kafka基础框架



Kafka

架构

Producer A

Producer B

Topic A

Partition 0

Topic A

Partition 1

Broker1

Leader

Follower

ReplicationA

/0

ReplicationA

/1

Topic A

Partition 0

Topic A

Partition 1

Broker2

Leader

Follower

Consumer A

Consumer B

Kafka Cluster

Broker3

Partition0

message0

message1

Consumer C

Zookeeper

Consumer group

message to A

-

0

message to A

-

1

message to B

-

0

message

from A

-

0

message

from A

-

1

message

from B

-

0

生产者生产消息

Kafka

集群管理消息

消费者消费消息

Zookeeper

注册消息

0.9

版本之前

offset

存储在

ZK

0.9

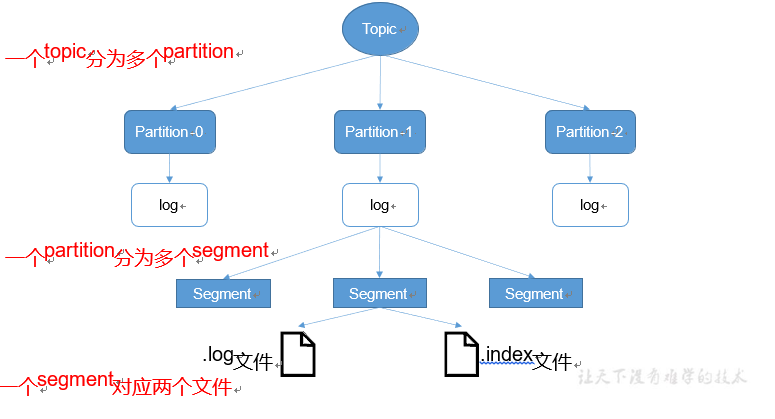
版本及之后

offset

存储本地（）

0.9版本之前offset是存放在zk中后来改为存储在本地kafaka的某个主题里原因：消费者本身是和kafka连接在获取数据的同时还要跟zk连接，消费者是以拉去的模式来获取消息的。拉取的速度太快和zk打交道太频繁。（记住消息存放在磁盘，默认保存7天）

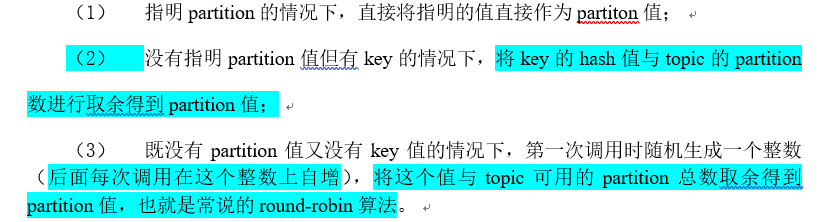
1. 消费者组是逻辑上的一个订阅者（把组当成一个大的消费体，这样做是为了提高并发）（效率最好的情况是消费者组里面的消费者个数和topic的的分区个数相等的时候）
2. 一个 topic 可以分为多个 partition（负载均衡），每个 partition 是一个有序的队列
3. 一个 leader 和若干个 follower（leader和follower不在同一个机器上）
4. kafka创建的副本数不能大于机器的台数，这个hdfs是有区别的，hdfs是只最大的副本数即使及其不够也行，在后面扩机器的时候会相应的增加到你指定的副本数
5. 严重注意，kafka向消息队列中写数据的时候不能保证全局有序性，只能保证分区有序性。
6. kafka的文件存取机制



为防止 log 文件过大导致数据定位，效率低下，Kafka 采取了分片和索引机制，将每个 partition 分为多个 segment。每个 segment 对应两个文件——“.index”文件和“.log”文件。

在同一个segement中进行二分查找，结合起始偏移量和数据大小快速拿到自己想要的数据

1. kafka分区的原因：方便在集群中扩展，可以提高并发。
2. 分区的原则



1. 数据可靠性保证：ack(3种)+副本同步策略（半数和全部）+ISR（看同步时间）+hw+leo(ack是保证生产者，hw和leo保证副本（这个副本是向consumer开放的）)

|  |  |  |  |
| --- | --- | --- | --- |
| 方案 |  | 优点 | 缺点 |
| 半数以上完成同步，就发送 **ack** |  | 延迟低 | 选举新的 leader 时，容忍 n 台节点的故障，需要 2n+1 个副本 |
| 全部完成同步，才发送 **ack （kafka选中这个）** |  | 选举新的 leader 时，容忍 n 台节点的故障，需要 n+1 个副本 | 延迟高 |

1. At Least Once + 幂等性 = Exactly Once，Broker 端会对<PID, Partition, SeqNumber>做缓存，当具有相同主键的消息提交时，Broker 只会持久化一条。 但是 PID(生产者id) 重启就会变化，同时不同的 Partition 也具有不同主键，所以幂等性无法保证跨分区跨会话(会话就是连接)的 Exactly Once。
2. Ack决定数据丢不丢的问题（ack有3个策略），ISR是争对副本同步策略，LEO,HW
3. 分区分配策略：range(12345678->(123,456,78)),roundrobin()(12345678->(147,258,36))

当消费者的个数发生变化的时候，增多或减少就触发，range是按主题分，roundrobin是按组分

1. kafka的高速读写数据：分布式分区（读写并行），顺序磁盘，零复制技术
2. Kafka 集群中有一个 broker 会被选举为 Controller（controller的选举是抢资源，谁先抢到谁当），controller负责管理集群 broker 的上下线，所有 topic 的分区副本分配和 leader 选举等工作。
3. Producer事务：解决单次会话里的问题，保证精准一次性写入磁盘。为了实现跨分区跨会话的事务，需要引入一个全局唯一的 Transaction ID（客户端传进去的），并将 Producer 获得的PID 和Transaction ID 绑定。这样当Producer重启后就可以通过正在进行的Transaction ID 获得原来的 PID。
4. offset的维护：什么东西唯一决定一个offset,由组+主题+分区决定（如果组内某个消费者挂了，其他成员能接着消费）。’
5. 如果消费者向没有创建的主题中发送数据的话，会默认创建一个主题，1个分区，1个副本。
6. 拦截器：第一个 interceptor 会在消息发送前将时间戳信息加到消息 value 的最前部；第二个 interceptor 会在消息发送后更新成功发送消息数或失败发送消息数。
7. 1.Kafka 中的 ISR(InSyncRepli)、OSR(OutSyncRepli)、AR(AllRepli)代表什么？

AR: kafka分区中，维护了一个AR列表，其中包括了所有的分区的副本编号，AR分为ISR和OSR(isr+osr=ar)

IDR：同步列表，只有当所有的ISR内的副本都同步了leader中的数据，数据才能被提交，才能被消费者访问

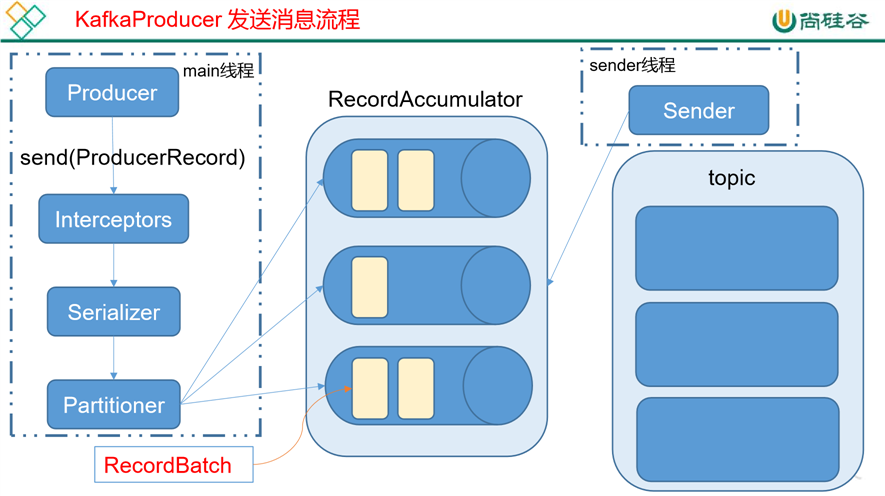
osr: 非同步列表，OSR内的副本是否同步了leader的数据，不影响数据的提交，OSR内的follower只是尽力的去同步leader，数据版本可能落后

1. Kafka 中的分区器、序列化器、拦截器是否了解？它们之间的处理顺序是什么？

interceptor 使得用户在消息发送前以及 producer 回调逻辑前有机会对消息做一些定制化需求，比如修改消息等。第一个 interceptor 会在消息发送前将时间戳信息加到消息 value 的最前部；第二个 interceptor 会在消息发送后更新成功发送消息数或失败发送消息数。

拦截器-》序列话器-》分区器

1. Kafka 生产者客户端的整体结构是什么样子的？使用了几个线程来处理？分别是什么？



在消息发送的过程中，涉及到了两个线程——main 线程和 Sender 线程，以及一个线程共享变量——RecordAccumulator。 main 线程将消息发送给 RecordAccumulator，Sender 线程不断从 RecordAccumulator 中拉取消息发送到 Kafka broker。

1. “消费组中的消费者个数如果超过 topic 的分区，那么就会有消费者消费不到数据”这句话是否正确？ 正确，这样浪费资源
2. 消费者提交消费位移时提交的是当前消费到的最新消息的 offset 还是 offset+1？

是offset+1,offset存放的下一个要消费的数据

1. 有哪些情形会造成重复消费（这是consumer）？不是数据重复(不是ack，ack是生产问题)

先处理数据，在提交offset（数据处理完了，offset没有提交）

1. 那些情景会造成消息漏消费？不是数据丢失

先提交后处理（先提交offset，但是数据没有处理）

1. 当你使用 kafka-topics.sh 创建（删除）了一个 topic 之后，Kafka 背后会执行什么逻辑？

* 会在 zookeeper 中的/brokers/topics 节点下创建一个新的 topic 节点，如：

/brokers/topics/first

* 触发 Controller 的监听程序
* kafka Controller 负责 topic 的创建工作，并更新 metadata cache

1. topic 的分区数可不可以增加？如果可以怎么增加？如果不可以，那又是为什么？

可增，不可减。可增，新的分区可以投入使用，不可减的原因是已经存在的数据不能解决。

1. 13.Kafka 有内部的 topic 吗？如果有是什么？有什么所用？

有，给普通消费者存offset

1. Kafka 分区分配的概念？ range(所有场景，面向主题)，**RoundRobin（面向组）**
2. 简述 Kafka 的日志目录结构？

index和log(先通过二分查找定位到index文件，扫描这个index文件，找到这个文件在log中具体的位置)

1. 如果我指定了一个 offset，Kafka Controller 怎么查找到对应的消息？

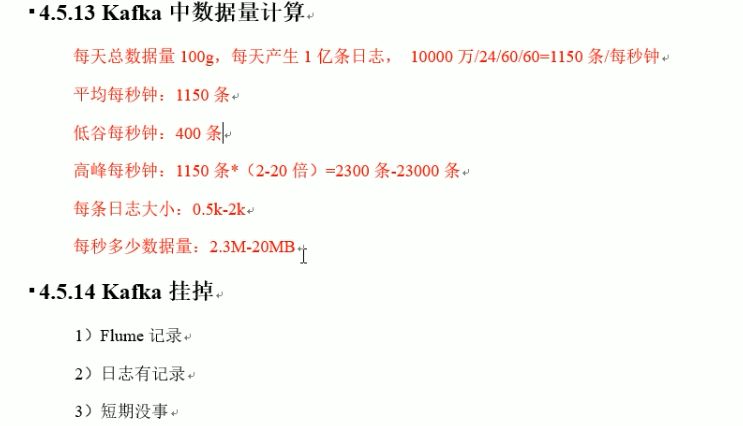
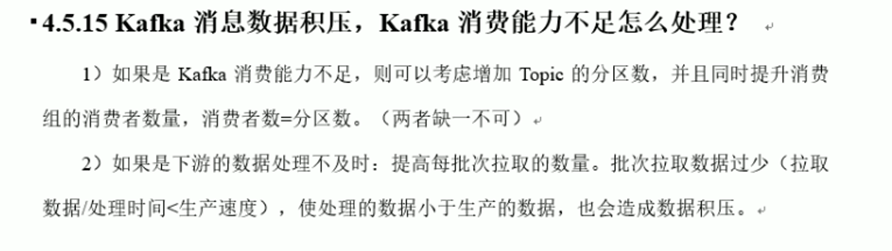
index和log(先通过二分查找定位到index文件，扫描这个index文件，找到这个文件在log中具体的位置)

1. Kafka 中有那些地方需要选举？这些地方的选举策略又有哪些？

controller(抢资源)，leader(ISR(一个看同步时间，一个看同步条数))

1. 失效副本是指什么？有那些应对措施？

失效leader

1. kafka压测：kafka官方自带压力测试脚本，CPU，内存，网络IO，一般网络IO达到瓶颈
2. kafak机器数量，日志存放时间，硬盘大小：2\*n+1，7天，每天数据量\*7天
3. 
4. 
5. 