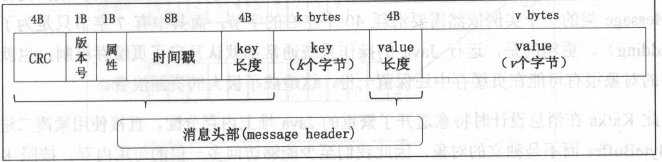
**Kafka笔记**

1. **基本概念**
   1. **消息格式**



key: 消息键，对消息做分区(partition)时使用

value：消息实体

timestamp:消息发送时间戳

* 1. **消息保存格式**

消息采用二进制字节数组而不是java对象的形式保存，避免了空闲字节的空间浪费。

**2. topic和partition**

生产者将消息发送到topic中，其实topic只是逻辑上的概念，并不会实际的存储消息。消息实际是存储在partition上的，partition的引入实际上是为了实现提高系统吞吐量。

Partition实际上是一个有序的消息序列。用户唯一能做的就是在消息序列的尾部追加消息，序列中的每条消息都会被分配一个序号，这个序号就是位移。位移用来定位序列中的消息。

注：在消费者端也存在位移的概念，这两个位移不是同一个，消费者端的位移是指：随着消息被消费，这个位移会不断的往后移动，但是不会超过分区中最新的消息。

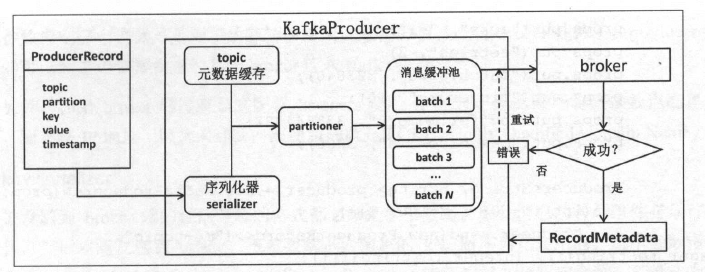
消息的定位:<topic，partition，offset>

**3. replica**

为了防止partition上的数据丢失，每个partition都会至少有一个replica(副本)，副本是不会在同一个机器上的，其次，不同的replica是有区分的，分为master和follower。follower是不负责向生产者和消费者提供服务的。他的唯一作用就是备份数据。它会被动的从master中获取数据，当master挂了，再从剩下的followers中选举出一个作为新的master。

1. **生产者**

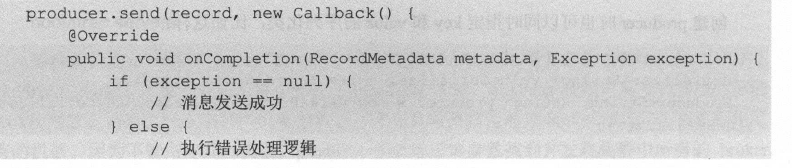
用户主线程先将消息封装进PrducerRecord类实例中，然后将其序列化之后交给partitioner,在确认需要发送的分区之后，将其放到内存缓冲区中，之后，Sender线程会从该缓冲区中取出数据封装进一个批次(batch)中统一发送到broker。



**2.1 消息发送方式：**

1. **异步发送**

生产者消息发送默认都是异步的，kafka提供一个Future对象供用户稍后获取发送结果。



其中参数metadata和exception不会同时为空，当消息发送成功，exception为空，发送失败，metadata为空。

1. **同步发送**

Future .get（）无限等待结果返回，即实现同步发送的效果

* 1. **acks参数**

生产者发送消息之后，会等待(会有等待超时时间)leader broker返回的消息，以确认消息是否被成功提交。收到broker的消息之后，再去决定发送下一条消息。Kafka可以保证消费者永远不会消费到未提交状态的消息。

Broker什么时候向生产者返回确认消息直接影响到kafka的系统吞吐量。而acks就是决定broker返回确认消息的时机的参数。acks参数有3种取值：

1. acks=0,生产者完全不管broker端的处理结果，一条消息发完，就可以直接进行下一条数据的发送了，这种情况吞吐量很高，但是生产者并不会知道是否发送成功，回调也就失去了作用。
2. acks=-1或者all：在这个取值情况下，不仅会将消息写入本地日志，还会等ISR中其他所有的副本都成功写入各自的日志后，才会将响应结果返回给producer。这种情况下的吞吐量最低但是消息也能最大程度上保证不会丢失。
3. acks=0,这种取值是一种折中的办法，也是默认的取值。当生产者将消息发送给leader broker之后，leader broker会将消息保存在本地，然后将响应结果发送给生产者，而不会等待ISR中其他副本写入消息。

**ISR**：每个partition都会有一个ISR，它由leader partition维护。其实就是一个列表，列表中的副本和master副本保持同步。

* 1. buffer.memory参数

生产者会先将消息放入缓冲区中(并不是batch)，然后由另外的一个线程(IO线程)从缓冲区中取出消息执行发送，当消息发送速率超过了IO线程，则发送线程会被阻塞消息发送，一段时间后，若消息发送线程还是超过IO线程，那么发送线程会抛出异常。也可以通过设置max.block.ms让producer阻塞而不会抛异常。

* 1. compression.type 参数

决定发送端是否压缩消息。

* 1. retries参数

生产者内部消息重发机制。这个参数设置不好容易导致消息重复。

* 1. batch.size参数

kafka是批量发送消息的，这个参数就是设置发送批次的大小，当然，也并不是每次都是等batch满了之后才会发送消息，linger.ms就是设置这个的。

* 1. linger.ms

控制消息延时发送的，当消息存入batch中超过一定的时间，就不等batch满了也会被发送出去。

* 1. max.request.size

发送消息的最大大小。

* 1. request.timeout.ms

生产者接收broker响应的超时时间。当超过这个时间还没有返回，则认为消息发送失败，回调中抛出超时异常。

1. 分区

默认采用的是轮训的方式去分区的，但是可以通过实现Partitioner接口，去自定义分区的策略。

1. 消息序列化

消息发送出去之前，会被序列化，消费者拿到之后会解序列化。

1. Producer拦截器

在生产者这一端，可以在消息发送前(序列化和计算分区前)被拦截、在ack返回前被拦截。

1. 消息压缩

Kafka支持将多条消息压缩为一条消息发送出去。

KatkaProducer是否是线程安全的

1. **消费者**