# 1、背景引入：很多同学看不懂kafka参数

今天给大家聊一个很有意思的话题，大家知道很多公司都会基于Kafka作为MQ来开发一些复杂的大型系统。

而在使用Kafka的客户端编写代码与服务器交互的时候，是需要对客户端设置很多的参数的。

所以我就见过很多年轻的同学，可能刚刚加入团队，对Kafka这个技术其实并不是很了解。

此时就会导致他们看团队里的一些资深同事写的一些代码，会看不懂是怎么回事，不了解背后的含义，这里面尤其是一些**Kafka参数的设置**。

所以这篇文章，我们还是采用老规矩画图的形式，来聊聊Kafka生产端一些常见参数的设置，让大家下次看到一些Kafka客户端设置的参数时，不会再感到发怵。

# 2、一段Kafka生产端的示例代码

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("buffer.memory", 67108864);

props.put("batch.size", 131072);

props.put("linger.ms", 100);

props.put("max.request.size", 10485760);

props.put("acks", "1");

props.put("retries", 10);

props.put("retry.backoff.ms", 500);

KafkaProducer<String, String> producer = new KafkaProducer<String, String>(props);

[IMG_257](https://www.cnblogs.com/technologykai/articles/javascript:void(0);)

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("buffer.memory", 67108864);

props.put("batch.size", 131072);

props.put("linger.ms", 100);

props.put("max.request.size", 10485760);

props.put("acks", "1");

props.put("retries", 10);

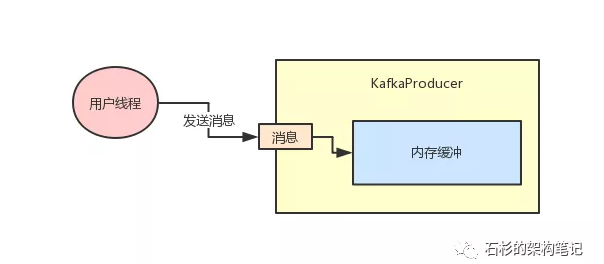
props.put("retry.backoff.ms", 500);

KafkaProducer<String, String> producer = new KafkaProducer<String, String>(props);

# 3、内存缓冲的大小

首先我们看看“buffer.memory”这个参数是什么意思？

Kafka的客户端发送数据到服务器，一般都是要经过缓冲的，也就是说，你通过KafkaProducer发送出去的消息都是先进入到客户端本地的内存缓冲里，然后把很多消息收集成一个一个的Batch，再发送到Broker上去的。



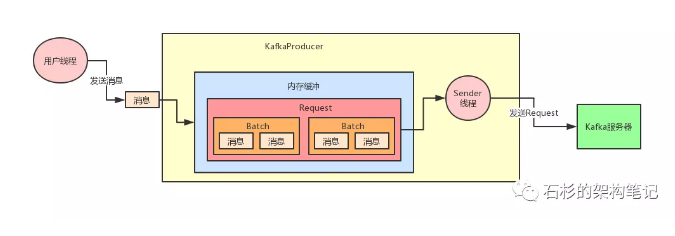
所以这个“buffer.memory”的本质就是用来约束KafkaProducer能够使用的内存缓冲的大小的，他的默认值是32MB。

那么既然了解了这个含义，大家想一下，在生产项目里，这个参数应该怎么来设置呢？

你可以先想一下，如果这个内存缓冲设置的过小的话，可能会导致一个什么问题？

首先要明确一点，那就是在内存缓冲里大量的消息会缓冲在里面，形成一个一个的Batch，每个Batch里包含多条消息。

然后KafkaProducer有一个Sender线程会把多个Batch打包成一个Request发送到Kafka服务器上去。



那么如果要是内存设置的太小，可能**导致一个问题**：消息快速的写入内存缓冲里面，但是Sender线程来不及把Request发送到Kafka服务器。

这样是不是会造成内存缓冲很快就被写满？一旦被写满，就会阻塞用户线程，不让继续往Kafka写消息了。

所以对于“buffer.memory”这个参数应该结合自己的实际情况来进行压测，你需要测算一下在生产环境，你的用户线程会以每秒多少消息的频率来写入内存缓冲。

比如说每秒300条消息，那么你就需要压测一下，假设内存缓冲就32MB，每秒写300条消息到内存缓冲，是否会经常把内存缓冲写满？经过这样的压测，你可以调试出来一个合理的内存大小。

# 4、多少数据打包为一个Batch合适？

接着你需要思考第二个问题，就是你的“batch.size”应该如何设置？这个东西是决定了你的每个Batch要存放多少数据就可以发送出去了。

比如说你要是给一个Batch设置成是16KB的大小，那么里面凑够16KB的数据就可以发送了。

这个参数的默认值是16KB，一般可以尝试把这个参数调节大一些，然后利用自己的生产环境发消息的负载来测试一下。

比如说发送消息的频率就是每秒300条，那么如果比如“batch.size”调节到了32KB，或者64KB，是否可以提升发送消息的整体吞吐量。

因为理论上来说，提升batch的大小，可以允许更多的数据缓冲在里面，那么一次Request发送出去的数据量就更多了，这样吞吐量可能会有所提升。

但是这个东西也不能无限的大，过于大了之后，要是数据老是缓冲在Batch里迟迟不发送出去，那么岂不是你发送消息的延迟就会很高。

比如说，一条消息进入了Batch，但是要等待5秒钟Batch才凑满了64KB，才能发送出去。那这条消息的延迟就是5秒钟。

所以需要在这里按照生产环境的发消息的速率，调节不同的Batch大小自己测试一下最终出去的吞吐量以及消息的 延迟，设置一个最合理的参数。

# 5、要是一个Batch迟迟无法凑满怎么办？

要是一个Batch迟迟无法凑满，此时就需要引入另外一个参数了，“linger.ms”

他的含义就是说一个Batch被创建之后，最多过多久，不管这个Batch有没有写满，都必须发送出去了。

给大家举个例子，比如说batch.size是16kb，但是现在某个低峰时间段，发送消息很慢。

这就导致可能Batch被创建之后，陆陆续续有消息进来，但是迟迟无法凑够16KB，难道此时就一直等着吗？

当然不是，假设你现在设置“linger.ms”是50ms，那么只要这个Batch从创建开始到现在已经过了50ms了，哪怕他还没满16KB，也要发送他出去了。

所以“linger.ms”决定了你的消息一旦写入一个Batch，最多等待这么多时间，他一定会跟着Batch一起发送出去。

避免一个Batch迟迟凑不满，导致消息一直积压在内存里发送不出去的情况。**这是一个很关键的参数。**

这个参数一般要非常慎重的来设置，要配合batch.size一起来设置。

举个例子，首先假设你的Batch是32KB，那么你得估算一下，正常情况下，一般多久会凑够一个Batch，比如正常来说可能20ms就会凑够一个Batch。

那么你的linger.ms就可以设置为25ms，也就是说，正常来说，大部分的Batch在20ms内都会凑满，但是你的linger.ms可以保证，哪怕遇到低峰时期，20ms凑不满一个Batch，还是会在25ms之后强制Batch发送出去。

如果要是你把linger.ms设置的太小了，比如说默认就是0ms，或者你设置个5ms，那可能导致你的Batch虽然设置了32KB，但是经常是还没凑够32KB的数据，5ms之后就直接强制Batch发送出去，这样也不太好其实，会导致你的Batch形同虚设，一直凑不满数据。

# 6、最大请求大小

“max.request.size”这个参数决定了每次发送给Kafka服务器请求的最大大小，同时也会限制你一条消息的最大大小也不能超过这个参数设置的值，这个其实可以根据你自己的消息的大小来灵活的调整。

给大家举个例子，你们公司发送的消息都是那种大的报文消息，每条消息都是很多的数据，一条消息可能都要20KB。

此时你的batch.size是不是就需要调节大一些？比如设置个512KB？然后你的buffer.memory是不是要给的大一些？比如设置个128MB？

只有这样，才能让你在大消息的场景下，还能使用Batch打包多条消息的机制。但是此时“max.request.size”是不是也得同步增加？

因为可能你的一个请求是很大的，默认他是1MB，你是不是可以适当调大一些，比如调节到5MB？

# 7、重试机制

“retries”和“retries.backoff.ms”决定了重试机制，也就是如果一个请求失败了可以重试几次，每次重试的间隔是多少毫秒。

这个大家适当设置几次重试的机会，给一定的重试间隔即可，比如给100ms的重试间隔。