



下载APP



13 | Java生产者是如何管理TCP连接的？

2019-07-02 胡夕

Kafka核心技术与实战

[进入课程 >](#)



讲述：胡夕

时长 12:14 大小 9.81M



你好，我是胡夕。今天我要和你分享的主题是：Kafka 的 Java 生产者是如何管理 TCP 连接的。

为何采用 TCP？

Apache Kafka 的所有通信都是基于 TCP 的，而不是基于 HTTP 或其他协议。无论是生产者、消费者，还是 Broker 之间的通信都是如此。你可能会问，为什么 Kafka 不使用 HTTP 作为底层的通信协议呢？其实这里面的原因有很多，但最主要的原因在于 TCP 和 HTTP 之间的区别。

从社区的角度来看，在开发客户端时，人们能够利用 TCP 本身提供的一些高级功能，比如多路复用请求以及同时轮询多个连接的能力。

所谓的多路复用请求，即 multiplexing request，是指将两个或多个数据流合并到底层单一物理连接中的过程。TCP 的多路复用请求会在一条物理连接上创建若干个虚拟连接，每个虚拟连接负责流转各自对应的数据流。其实严格来说，TCP 并不能多路复用，它只是提供可靠的消息交付语义保证，比如自动重传丢失的报文。

更严谨地说，作为一个基于报文的协议，TCP 能够被用于多路复用连接场景的前提是，上层的应用协议（比如 HTTP）允许发送多条消息。不过，我们今天并不是要详细讨论 TCP 原理，因此你只需要知道这是社区采用 TCP 的理由之一就行了。

除了 TCP 提供的这些高级功能有可能被 Kafka 客户端的开发人员使用之外，社区还发现，目前已知的 HTTP 库在很多编程语言中都略显简陋。

基于这两个原因，Kafka 社区决定采用 TCP 协议作为所有请求通信的底层协议。

Kafka 生产者程序概览

Kafka 的 Java 生产者 API 主要的对象就是 `KafkaProducer`。通常我们开发一个生产者的步骤有 4 步。

第 1 步：构造生产者对象所需的参数对象。

第 2 步：利用第 1 步的参数对象，创建 `KafkaProducer` 对象实例。

第 3 步：使用 `KafkaProducer` 的 `send` 方法发送消息。

第 4 步：调用 `KafkaProducer` 的 `close` 方法关闭生产者并释放各种系统资源。

上面这 4 步写成 Java 代码的话大概是这样子：

```
1 Properties props = new Properties ();
2 props.put("参数 1", "参数 1 的值");
3 props.put("参数 2", "参数 2 的值");
4 .....
5 try (Producer<String, String> producer = new KafkaProdu
6         producer.send(new ProducerRecord<String, St
7         .....
8 }
```

这段代码使用了 Java 7 提供的 try-with-resource 特性，所以并没有显式调用 producer.close() 方法。无论是否显式调用 close 方法，所有生产者程序大致都是这个路数。

现在问题来了，当我们开发一个 Producer 应用时，生产者会向 Kafka 集群中指定的主题（Topic）发送消息，这必然涉及与 Kafka Broker 创建 TCP 连接。那么，Kafka 的 Producer 客户端是如何管理这些 TCP 连接的呢？

何时创建 TCP 连接？

要回答上面这个问题，我们首先要弄明白生产者代码是什么时候创建 TCP 连接的。就上面的那段代码而言，可能创建 TCP 连接的地方有两处：Producer producer = new KafkaProducer(props) 和 producer.send(msg, callback)。你觉得连向 Broker 端的 TCP 连接会是哪里创

建的呢？前者还是后者，抑或是两者都有？请先思考 5 秒钟，然后我给出我的答案。

首先，生产者应用在创建 `KafkaProducer` 实例时是会建立与 Broker 的 TCP 连接的。其实这种表述也不是很准确，应该这样说：**在创建 `KafkaProducer` 实例时，生产者应用会在后台创建并启动一个名为 `Sender` 的线程，该 `Sender` 线程开始运行时首先会创建与 Broker 的连接。**我截取了一段测试环境中的日志来说明这一点：

```
[2018-12-09 09:35:45,620] DEBUG  
[Producer clientId=producer-1] Initialize  
connection to node localhost:9093 (id: -2  
rack: null) for sending metadata request  
(org.apache.kafka.clients.NetworkClient:1  
084)
```

```
[2018-12-09 09:35:45,622] DEBUG  
[Producer clientId=producer-1] Initiating  
connection to node localhost:9093 (id: -2  
rack: null) using address  
localhost/127.0.0.1  
(org.apache.kafka.clients.NetworkClient:9  
14)
```

[2018-12-09 09:35:45,814] DEBUG
[Producer clientId=producer-1] Initialize
connection to node localhost:9092 (id: -1
rack: null) for sending metadata request
(org.apache.kafka.clients.NetworkClient:1
084)

[2018-12-09 09:35:45,815] DEBUG
[Producer clientId=producer-1] Initiating
connection to node localhost:9092 (id: -1
rack: null) using address
localhost/127.0.0.1
(org.apache.kafka.clients.NetworkClient:9
14)

[2018-12-09 09:35:45,828] DEBUG
[Producer clientId=producer-1] Sending
metadata request
(type=MetadataRequest, topics=) to
node localhost:9093 (id: -2 rack: null)
(org.apache.kafka.clients.NetworkClient:1
068)

你也许会问：怎么可能是这样？如果不调用 send 方法，这个 Producer 都不知道给哪个主题发消息，它又怎么能知道连接哪个 Broker 呢？难不成它会连接 bootstrap.servers 参数指定的所有 Broker 吗？嗯，是的，Java Producer 目前还真是这样设计的。

我在这里稍微解释一下 bootstrap.servers 参数。它是 Producer 的核心参数之一，指定了这个 Producer 启动时要连接的 Broker 地址。请注意，这里的“启动时”，代表的是 Producer 启动时会发起与这些 Broker 的连接。因此，如果你为这个参数指定了 1000 个 Broker 连接信息，那么很遗憾，你的 Producer 启动时会首先创建与这 1000 个 Broker 的 TCP 连接。

在实际使用过程中，我并不建议把集群中所有的 Broker 信息都配置到 bootstrap.servers 中，通常你指定 3~4 台就足够了。因为 Producer 一旦连接到集群中的任一 Broker，就能拿到整个集群的 Broker 信息，故没必要为 bootstrap.servers 指定所有的 Broker。

让我们回顾一下上面的日志输出，请注意我标为橙色的内容。从这段日志中，我们可以发现，在 KafkaProducer 实例被创建后以及消息被发送前，Producer 应用就开始创建与两台 Broker 的 TCP 连接了。当然了，在我的测试环境

中，我为 bootstrap.servers 配置了 localhost:9092、localhost:9093 来模拟不同的 Broker，但是这并不影响后面的讨论。另外，日志输出中的最后一行也很关键：它表明 Producer 向某一台 Broker 发送了 METADATA 请求，尝试获取集群的元数据信息——这就是前面提到的 Producer 能够获取集群所有信息的方法。

讲到这里，我有一些个人的看法想跟你分享一下。通常情况下，我都不认为社区写的代码或做的设计就一定是对的，因此，很多类似的这种“质疑”会时不时地在我脑子里冒出来。

拿今天的这个 KafkaProducer 创建实例来说，社区的官方文档中提及 KafkaProducer 类是线程安全的。我本人并没有详尽地去验证过它是否真的就是 thread-safe 的，但是大致浏览一下源码可以得出这样的结论：KafkaProducer 实例创建的线程和前面提到的 Sender 线程共享的可变数据结构只有 RecordAccumulator 类，故维护了 RecordAccumulator 类的线程安全，也就实现了 KafkaProducer 类的线程安全。

你不需要了解 RecordAccumulator 类是做什么的，你只要知道它主要的数据结构是一个 ConcurrentMap<TopicPartition, Deque>。

TopicPartition 是 Kafka 用来表示主题分区的 Java 对象，本身是不可变对象。而 RecordAccumulator 代码中用到 Deque 的地方都有锁的保护，所以基本上可以认定 RecordAccumulator 类是线程安全的。

说了这么多，我其实是想说，纵然 KafkaProducer 是线程安全的，我也不赞同创建 KafkaProducer 实例时启动 Sender 线程的做法。写了《Java 并发编程实践》的那位布赖恩·格茨（Brian Goetz）大神，明确指出了这样做的风险：在对象构造器中启动线程会造成 this 指针的逃逸。理论上，Sender 线程完全能够观测到一个尚未构造完成的 KafkaProducer 实例。当然，在构造对象时创建线程没有任何问题，但最好是不要同时启动它。

好了，我们言归正传。针对 TCP 连接何时创建的问题，目前我们的结论是这样的：**TCP 连接是在创建 KafkaProducer 实例时建立的**。那么，我们想问的是，它只会在这个时候被创建吗？

当然不是！**TCP 连接还可能在两个地方被创建：一个是在更新元数据后，另一个是在消息发送时**。为什么说是可能？因为这两个地方并非总是创建 TCP 连接。当 Producer 更新了集群的元数据信息之后，如果发现与某些 Broker 当前没有连接，那么它就会创建一个 TCP 连接。同样地，当要发

送消息时，Producer 发现尚不存在与目标 Broker 的连接，也会创建一个。

接下来，我们来看看 Producer 更新集群元数据信息的两个场景。

场景一：当 Producer 尝试给一个不存在的主题发送消息时，Broker 会告诉 Producer 说这个主题不存在。此时 Producer 会发送 METADATA 请求给 Kafka 集群，去尝试获取最新的元数据信息。

场景二：Producer 通过 `metadata.max.age.ms` 参数定期地去更新元数据信息。该参数的默认值是 300000，即 5 分钟，也就是说不管集群那边是否有变化，Producer 每 5 分钟都会强制刷新一次元数据以保证它是最及时的数据。

讲到这里，我们可以“挑战”一下社区对 Producer 的这种设计的合理性。目前来看，一个 Producer 默认会向集群的所有 Broker 都创建 TCP 连接，不管是否真的需要传输请求。这显然是没有必要的。再加上 Kafka 还支持强制将空闲的 TCP 连接资源关闭，这就更显得多此一举了。

试想一下，在一个有着 1000 台 Broker 的集群中，你的 Producer 可能只会与其中的 3~5 台 Broker 长期通信，但

是 Producer 启动后依次创建与这 1000 台 Broker 的 TCP 连接。一段时间之后，大约有 995 个 TCP 连接又被强制关闭。这难道不是一种资源浪费吗？很显然，这里是有改善和优化的空间的。

何时关闭 TCP 连接？

说完了 TCP 连接的创建，我们来说说它们何时被关闭。

Producer 端关闭 TCP 连接的方式有两种：**一种为用户主动关闭；一种是 Kafka 自动关闭。**

我们先说第一种。这里的主动关闭实际上是广义的主动关闭，甚至包括用户调用 `kill -9` 主动“杀掉”Producer 应用。当然最推荐的方式还是调用 `producer.close()` 方法来关闭。

第二种是 Kafka 帮你关闭，这与 Producer 端参数 `connections.max.idle.ms` 的值有关。默认情况下该参数值是 9 分钟，即如果在 9 分钟内没有任何请求“流过”某个 TCP 连接，那么 Kafka 会主动帮你把该 TCP 连接关闭。用户可以在 Producer 端设置 `connections.max.idle.ms=-1` 禁掉这种机制。一旦被设置成 -1，TCP 连接将成为永久长连接。当然这只是软件层面的“长连接”机制，由于 Kafka

创建的这些 Socket 连接都开启了 keepalive，因此 keepalive 探活机制还是会遵守的。

值得注意的是，在第二种方式中，TCP 连接是在 Broker 端被关闭的，但其实这个 TCP 连接的发起方是客户端，因此在 TCP 看来，这属于被动关闭的场景，即 passive close。被动关闭的后果就是会产生大量的 CLOSE_WAIT 连接，因此 Producer 端或 Client 端没有机会显式地观测到此连接已被中断。

小结

我们来简单总结一下今天的内容。对最新版本的 Kafka (2.1.0) 而言，Java Producer 端管理 TCP 连接的方式是：

1. KafkaProducer 实例创建时启动 Sender 线程，从而创建与 bootstrap.servers 中所有 Broker 的 TCP 连接。
2. KafkaProducer 实例首次更新元数据信息之后，还会再次创建与集群中所有 Broker 的 TCP 连接。
3. 如果 Producer 端发送消息到某台 Broker 时发现没有与该 Broker 的 TCP 连接，那么也会立即创建连接。
4. 如果设置 Producer 端 connections.max.idle.ms 参数大于 0，则步骤 1 中创建的 TCP 连接会被自动关闭；如

果设置该参数 $= -1$ ，那么步骤 1 中创建的 TCP 连接将无法被关闭，从而成为“僵尸”连接。

开放讨论

对于今天我们“挑战”的社区设计，你有什么改进的想法吗？

欢迎写下你的思考和答案，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

 极客时间

Kafka 核心技术与实战

全面提升你的 Kafka 实战能力

胡夕

人人贷计算平台部总监
Apache Kafka Contributor



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 12 | 客户端都有哪些个常见但是很高级的功能?

下一篇 14 | 幂等生产者和事务生产者是一回事吗?

精选留言 (23)

写留言



柠檬C

2019-07-02

应该可以用懒加载的方式，实际发送时再进行TCP连接吧，虽然这样第一次发送时因为握手的原因会稍慢一点

1

3



KEEPUP

2019-07-02

KafkaProducer 实例只是在首次更新元数据信息之后，创建与集群中所有 Broker 的 TCP 连接，还是每次更新之后都要创建？为什么要创建与所有 Broker 的连接呢？

展开

1

2



明翼

2019-07-02

我的想法这样的，先用客户端去连配置的第一broker server，连不上就连接第二个，一旦连上了，就可以获取

元数据信息了，这是创建produce实例时候动作，只发起一个TCP连接。再send时候发现没连接的再连接，至于其他的都还是很合理的。



👍 2



JoeyLi666

2019-07-04

老师，最近使用kafka，报了个异常：

Caused by:

org.apache.kafka.common.KafkaException: Record batch for partition Notify-18 at offset 1803009 is invalid, cause: Record is corrupt (stored crc =...

展开 ✓

作者回复: 也可能是网络传输过程中出现的偶发情况，通常没有什么好的解决办法。。。



💬 1

👍 1



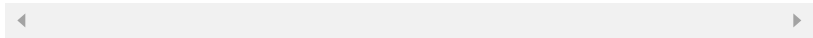
南辕北辙

2019-07-03

老师您好，我在本地分别用1.x版本和2.x版本的生产者去测试，为什么结果和老师的不一样呢。初始化KafkaProducer时，并没有与参数中设置的所有broker去建立连接，然后我sleep十秒，让sender线程有机会多运

行会。但是还是没有看到去连接所有的broker。只有当...
展开 ∨

作者回复: 我不知道您这边是怎么实验的, 但是我这边的确
会创建新的TCP连接~~

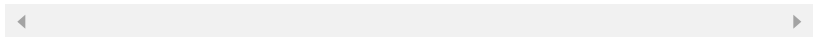


kursk.ye

2019-07-02

试想一下, 在一个有着 1000 台 Broker 的集群中, 你的
Producer 可能只会与其中的 3~5 台 Broker 长期通信,
但是 Producer 启动后依次创建与这 1000 台 Broker 的
TCP 连接。一段时间之后, 大约有 995 个 TCP 连接又被
强制关闭。这难道不是一种资源浪费吗? 很显然, 这里...
展开 ∨

作者回复: 嗯嗯, 欢迎不同意见。Kafka对于创建连接没有做
任何限制。如果一开始就创建所有TCP连接, 之后因为超时的
缘故又关闭这些连接, 当真正使用时再次创建, 那么为什
么不把创建时机后延到真正需要的时候呢? 实际场景中
将TCP连接设置为长连接的情形并不多见, 因此我说这种设计
是可以改进的。





诗泽

2019-07-02

看来无论在bootstrap.servers中是否写全部broker 的地址接下来producer 还是会跟所有的broker 建立一次连接



1



电光火石

2019-07-07

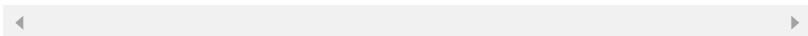
谢谢老师。有几个问题请教一下：

1. producer连接是每个broker一个连接，跟topic没有关系是吗？（consumer也是这样是吗？）
2. 我们运维在所有的broker之前放了一个F5做负载均衡，但其实应该也没用，他会自动去获取kafka所有的...
展开 ∨

作者回复: 1. 也不能说没有关系。客户端需要和topic下各个分区leader副本所在的broker进行连接的

2. 嗯嗯，目前客户端是直连broker的

3. 光看描述想不出具体的原因。有可能是频繁rebalance、long GC、消息corrupted或干脆就是一个已知的bug



燃烧的M豆



2019-07-04

c 包在v1.0.0 已经对这个问题采取了对策。所有使用基础 c 包 librdkafka 都因此受益

Sparse connections

In previous releases librdkafka would maintain ope...

展开 ∨



流浪的阳光

2019-07-03

老师、各位专家好，请问kafka适合做两个系统之间的转账处理吗？

另外，请问kafka的使用案例中，最多支持过什么数量级的消费者和生产者？



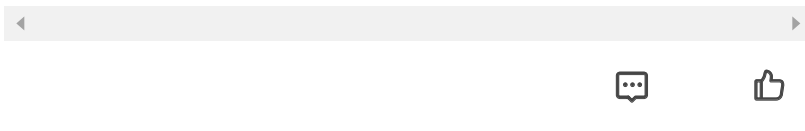
ban

2019-07-03

如果多个线程都使用一个KafkaProducer实例，缓冲器被填满的速度会变快。

老师看评论这句话不太理解，多个线程共用一个实例，没有再new新的实例，为什么缓冲器很快填满，不是利用原有的实例的吗。

作者回复: 一个KafkaProducer实例会开辟一块内存缓冲区, 如果多个线程共用这部分buffer, 自然buffer被填满的速度就会快很多。我是这个意思。。。



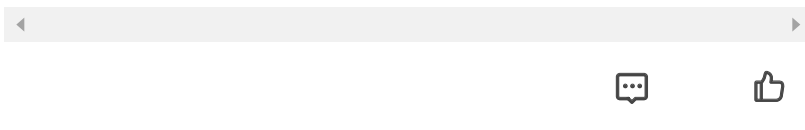
Li Shunduo

2019-07-03

请问Producer会和同一台broker建立多个TCP连接吗?

展开 ∨

作者回复: 有可能, 可能用于不同的目的, 通常最终会收敛成一个



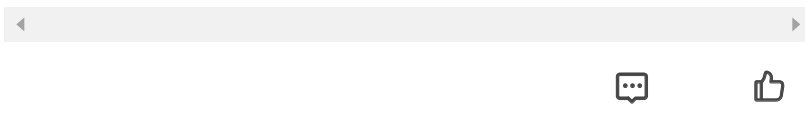
Geek_Sue

2019-07-03

胡老师, 我想请教一下。

connections.max.idle.ms这个值如果设置为-1, 按照您文章里所说, KafkaProducer会创建bootstrap.servers中全部tcp连接, 如果是1000个, 那么就是说这1000个连接永远不会关闭了?

作者回复: 如果 broker端和clients端的
connections.max.idle.ms都是-1, 就真是永不关闭了。



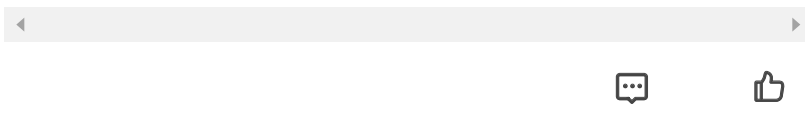
飞翔

2019-07-03

KafkaProducer 实例首次更新元数据信息之后, 还会再次创建与集群中所有 Broker 的 TCP 连接。这里如果我有1000个broker, 就建立1000个链接? 为啥不是只根据元信息只建立有我要发的主题的broke的链接?

展开 ∨

作者回复: 嗯嗯, 目前不是这样设计的。



rm -rf ☺

2019-07-02

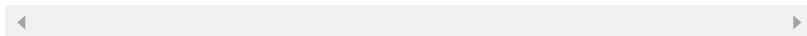
您好, 老师, 你说的第二种关闭producer的方法是一个被动关闭, 发起方是客户端。

1. 这个客户端指的是producer客户端? 所以broker此时可能会出现close wait, 是这个意思吗?
2. 还有这句 “因此 Producer 端或 Client 端没有机会显...

展开 ✓

作者回复: 1. 是这个意思

2. 我是想说consumer端也是这个原理。因此加上了clients

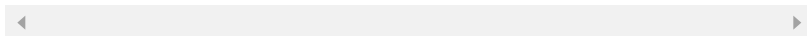


曾轼麟

2019-07-02

老师下面就有一个问题，KafkaProducer是建议创建实例后复用，像连接池那样使用，还是建议每次发送构造一个实例？听完这讲后感觉哪个都不合理，每次new会有很大的开销，但是一次new感觉又有僵尸连接，KafkaProducer适合池化吗？还是建议单例？

作者回复: KafkaProducer是线程安全的，复用是没有问题的。只是要监控内存缓冲区的使用情况。毕竟如果多个线程都使用一个KafkaProducer实例，缓冲器被填满的速度会变快。



趙衍

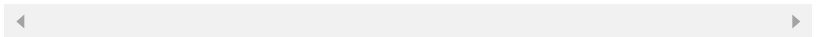
2019-07-02

老师好，看了今天的文章我有几个问题：

1.Kafka的元数据信息是存储在zookeeper中的，而Producer是通过broker来获取元数据信息的，那么这个过程是否是这样的，Producer向Broker发送一个获取...

展开 ∨

作者回复: 1. 集群元数据持久化在ZooKeeper中，同时也缓存在每台Broker的内存中，因此不需要请求ZooKeeper
2. 就我个人认为，的确有一些不高效。所以我说这里有优化的空间的。

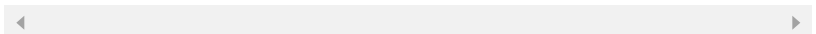


吴宇晨

2019-07-02

老师你好，kafka更新元数据的方法只有每5分钟的轮训吗，如果有监控zk节点之类的，是不是可以把轮询元数据时间调大甚至取消

作者回复: Clients端有个参数metadata.max.age.ms强制刷新元数据，默认的确是5分钟。新版本Clients不会与ZooKeeper交互，所以感觉和ZooKeeper没什么关系。。。



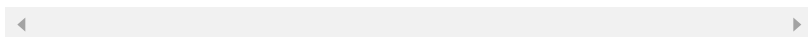


Imtoo

2019-07-02

Producer应该只跟Controller节点更新元数据，以及相关的topic机器交互数据，而不应该跟所有的机器创建连接；有个疑问，当 Producer 更新了集群的元数据信息之后，如果发现与某些broker没有连接，就去创建，为什么会去创建跟producer无关的连接？

作者回复: 有可能也会连接这些Broker的。Clients获取到集群元数据后知道了集群所有Broker的连接信息。下次再次获取元数据时，它会选择一个负载最少的Broker进行连接。如果发现没有连接也会创建Socket，但其实它并不需要向这个Broker发送任何消息。



张庆

2019-07-02

主要的资源浪费应该是在第一次获取元数据的时候创建所有的连接，应该是这个地方可以做一些优化吧，可以做一个最小初始化数量，从元数据中随机获取配置的最少数据，然后进行初始化。然后向Broker发送消息的时候在去判断，如果没有连接就创建连接，这样应该可以折中一下吧。

