15-Kafka如何实现高性能IO?

你好，我是李玥。

Apache Kafka是一个高性能的消息队列，在众多消息队列产品中，Kafka的性能绝对是处于第一梯队的。我曾经在一台配置比较好的服务器上，对Kafka做过极限的性能压测，Kafka单个节点的极限处理能力接近每秒钟2000万条消息，吞吐量达到每秒钟600MB。

你可能会问，Kafka是如何做到这么高的性能的?

我们在专栏“进阶篇”的前几节课，讲的知识点一直围绕着同一个主题：怎么开发一个高性能的网络应用程序。其中提到了像全异步化的线程模型、高性能的异步网络传输、自定义的私有传输协议和序列化、反序列化等等，这些方法和优化技巧，你都可以在Kafka的源代码中找到对应的实现。

在性能优化方面，除了这些通用的性能优化手段之外，Kafka还有哪些“独门绝技”呢?

这节课，我来为你一一揭晓这些绝技。

使用批量消息提升服务端处理能力

我们知道，批量处理是一种非常有效的提升系统吞吐量的方法。在Kafka内部，消息都是以“批”为单位处理的。一批消息从发送端到接收端，是如何在Kafka中流转的呢?

我们先来看发送端，也就是Producer这一端。

在Kafka的客户端SDK（软件开发工具包）中，Kafka的Producer只提供了单条发送的send()方法，并没有提供任何批量发送的接口。原因是，Kafka根本就没有提供单条发送的功能，是的，你没有看错，虽然它提供的API每次只能发送一条消息，但实际上，Kafka的客户端SDK在实现消息发送逻辑的时候，采用了异步批量发送的机制。

当你调用send()方法发送一条消息之后，无论你是同步发送还是异步发送，Kafka都不会立即就把这条消息发送出去。它会先把这条消息，存放在内存中缓存起来，然后选择合适的时机把缓存中的所有消息组成一批，一次性发给Broker。简单地说，就是攒一波一起发。

在Kafka的服务端，也就是Broker这一端，又是如何处理这一批一批的消息呢?

在服务端，Kafka不会把一批消息再还原成多条消息，再一条一条地处理，这样太慢了。Kafka这块儿处理的非常聪明，每批消息都会被当做一个“批消息”来处理。也就是说，在Broker整个处理流程中，无论是写入磁盘、从磁盘读出来、还是复制到其他副本这些流程中，**批消息都不会被解开，一直是作为一条“批消息”来进行处理的。**

在消费时，消息同样是以批为单位进行传递的，Consumer从Broker拉到一批消息后，在客户端把批消息解开，再一条一条交给用户代码处理。

比如说，你在客户端发送30条消息，在业务程序看来，是发送了30条消息，而对于Kafka的Broker来说，它其实就是处理了1条包含30条消息的“批消息”而已。显然处理1次请求要比处理30次请求要快得多。

构建批消息和解开批消息分别在发送端和消费端的客户端完成，不仅减轻了Broker的压力，最重要的是减少了Broker处理请求的次数，提升了总体的处理能力。

这就是Kafka用批量消息提升性能的方法。

我们知道，相比于网络传输和内存，磁盘IO的速度是比较慢的。对于消息队列的服务端来说，性能的瓶颈主要在磁盘IO这一块。接下来我们看一下，Kafka在磁盘IO这块儿做了哪些优化。

使用顺序读写提升磁盘IO性能

对于磁盘来说，它有一个特性，就是顺序读写的性能要远远好于随机读写。在SSD（固态硬盘）上，顺序读写的性能要比随机读写快几倍，如果是机械硬盘，这个差距会达到几十倍。为什么呢？

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，这个寻址需要比较长的时间，因为它要移动磁头，这是个机械运动，机械硬盘工作的时候会发出咔咔的声音，就是移动磁头发出的声音。

顺序读写相比随机读写省去了大部分的寻址时间，它只要寻址一次，就可以连续地读写下去，所以说，性能要比随机读写要好很多。

Kafka就是充分利用了磁盘的这个特性。它的存储设计非常简单，对于每个分区，它把从Producer收到的消息，顺序地写入对应的log文件中，一个文件写满了，就开启一个新的文件这样顺序写下去。消费的时候，也是从某个全局的位置开始，也就是某一个log文件中的某个位置开始，顺序地把消息读出来。

这样一个简单的设计，充分利用了顺序读写这个特性，极大提升了Kafka在使用磁盘时的IO性能。

接下来我们说一下Kafka是如何实现缓存的。

利用PageCache加速消息读写

在Kafka中，它会利用PageCache加速消息读写。PageCache是现代操作系统都具有的一项基本特性。通俗地说，PageCache就是操作系统在内存中给磁盘上的文件建立的缓存。无论我们使用什么语言编写的程序，在调用系统的API读写文件的时候，并不会直接去读写磁盘上的文件，应用程序实际操作的都是PageCache，也就是文件在内存中缓存的副本。

应用程序在写入文件的时候，操作系统会先把数据写入到内存中的PageCache，然后再一批一批地写到磁盘上。读取文件的时候，也是从PageCache中来读取数据，这时候会出现两种可能情况。

一种是PageCache中有数据，那就直接读取，这样就节省了从磁盘上读取数据的时间；另一种情况是，PageCache中没有数据，这时候操作系统会引发一个缺页中断，应用程序的读取线程会被阻塞，操作系统把数据从文件中复制到PageCache中，然后应用程序再从PageCache中继续把数据读出来，这时会真正读一次磁盘上的文件，这个读的过程就会比较慢。

用户的应用程序在使用完某块PageCache后，操作系统并不会立刻就清除这个PageCache，而是尽可能地利用空闲的物理内存保存这些PageCache，除非系统内存不够用，操作系统才会清理掉一部分PageCache。清理的策略一般是LRU或它的变种算法，这个算法我们不展开讲，它保留PageCache的逻辑是：优先保留最近一段时间最常使用的那些PageCache。

Kafka在读写消息文件的时候，充分利用了PageCache的特性。一般来说，消息刚刚写入到服务端就会被消费，按照LRU的“优先清除最近最少使用的页”这种策略，读取的时候，对于这种刚刚写入的PageCache，命中的几率会非常高。

也就是说，大部分情况下，消费读消息都会命中PageCache，带来的好处有两个：一个是读取的速度会非常快，另外一个是，给写入消息让出磁盘的IO资源，间接也提升了写入的性能。

ZeroCopy：零拷贝技术

Kafka的服务端在消费过程中，还使用了一种“零拷贝”的操作系统特性来进一步提升消费的性能。

我们知道，在服务端，处理消费的大致逻辑是这样的：

- 首先，从文件中找到消息数据，读到内存中；
- 然后，把消息通过网络发给客户端。

这个过程中，数据实际上做了2次或者3次复制：

1. 从文件复制数据到PageCache中，如果命中PageCache，这一步可以省掉；
2. 从PageCache复制到应用程序的内存空间中，也就是我们可以操作的对象所在的内存；
3. 从应用程序的内存空间复制到Socket的缓冲区，这个过程就是我们调用网络应用框架的API发送数据的过程。

Kafka使用零拷贝技术可以把这个复制次数减少一次，上面的2、3步骤两次复制合并成一次复制。直接从PageCache中把数据复制到Socket缓冲区中，这样不仅减少一次数据复制，更重要的是，由于不用把数据复制到用户内存空间，DMA控制器可以直接完成数据复制，不需要CPU参与，速度更快。

下面是这个零拷贝对应的系统调用：

```
#include <sys/socket.h>
ssize_t sendfile(int out_fd, int in_fd, off_t *offset, size_t count);
```

它的前两个参数分别是目的端和源端的文件描述符，后面两个参数是源端的偏移量和复制数据的长度，返回值是实际复制数据的长度。

如果你遇到这种从文件读出数据后再通过网络发送出去的场景，并且这个过程中你不需要对这些数据进行处理，那一定要使用这个零拷贝的方法，可以有效地提升性能。

小结

这节课，我们总结了Kafka的高性能设计中的几个关键的技术点：

- 使用批量处理的方式来提升系统吞吐能力。
- 基于磁盘文件高性能顺序读写的特性来设计的存储结构。

- 利用操作系统的PageCache来缓存数据，减少IO并提升读性能。
- 使用零拷贝技术加速消费流程。

以上这些，就是Kafka之所以能做到如此高性能的关键技术点。你可以看到，要真正实现一个高性能的消息队列，是非常不容易的，你需要熟练掌握非常多的编程语言和操作系统的底层技术。

这些优化的方法和技术，同样可以用在其他适合的场景和应用程序中。我希望你能充分理解这几项优化技术的原理，知道它们在什么情况下适用，什么情况下不适用。这样，当你遇到合适场景的时候，再深入去学习它的细节用法，最终就能把它真正地用到你开发的程序中。

思考题

课后，我希望你去读一读Kafka的源代码，从我们这节课中找一两个技术点，找到对应的代码部分，真正去看一下，我们说的这些优化技术，是如何落地到代码上的。在分析源代码的过程中，如果有任何问题，也欢迎你在留言区和我一起讨论。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言：

- 每天晒白牙 2019-08-27 06:59:19
  谢谢老师，今天讲到的点，我会在课下去读源码并写出文章 [2赞]

  作者回复2019-08-27 09:24:50
  期待

- 业余草 2019-08-27 08:30:28
  只说了它的优点，其实它的缺点也很明显。把确定也顺便解释解释。 [1赞]

  作者回复2019-08-27 09:26:45
  你可以分享一下，在使用Kafka的时候遇到了哪些问题。

- leslie 2019-08-27 06:16:46
  老师的课程学到现在开始越来越费力了：一堂课学完笔记量已经直线上升了；对于今天的课程读完后有些困惑之处烦劳老师可以指点迷津：
  1.客户端发送者的发送给服务器端的时候：其实是写入一个Packge或者说一个log包，然后服务器端处理完这个包之后，作为一个批处理，处理完成后给客户端的消费者消费者解包之后依次获得处理结果；是这样么。
  2.关于PageCache：刘超老师的课程中曾经提及其实消息队列主要运作在缓存层，常驻缓存就是为了节约查询时间；老师早先在开课的时候提过不同的消息队列其实特性不同，Kafka擅长或者说充分利用的是PageCache，其它如RockeMQ呢？我们如何扬长避短
  主要是基于以下两方面：一方面是-其实现在大量的服务器是在云端的，无论是Amaze云、腾讯云、阿里云其实共同的特性都是CPU和IO稳定性或者使用率并非真实会引发一些看似极高的是使用率真实情况却并非有那么高，另外一方面-其实任何消息队列的推出都是基于当下，如果想基于当下的消息队列做些二次开发或者特性改进需要做些什么或者准备些什么呢?操作系统、计算机组成原理，还有什么？望老师能提点1、2.
  跟着老师学到现在发现确实学好这门课可能比老师最初说的要求还要高：老师的课程跟到现在，觉得自己已经在最初的目标的路上了，谢谢老师的提点；期待老师的后续课程。 [1赞]

  作者回复2019-08-27 09:32:16
  对于第一点，你的理解是没问题的。

  第二个问题，我的建议是，平时注重学习积累，哪怕我只是开发一个CRUD，也要认真的做好每个细节，把涉及到的知识搞清楚。而不是照葫芦画瓢跟网上抄一个能work的就行了。对于二次开发这个事儿，先解决目的的问题。不能为了二次开发而二次开发，一定是遇到一个什么问题，经过思考，二次开发是最佳的解决方案，这样才需要做二次开发。

  至于涉及到哪些知识，我们这门课中讲的这些基础的东西大概率你会用到，其它的可以靠日常积累和快速学习来解决。

- timmy21 2019-08-27 01:19:02
  老师，我有两个疑问想请教一下：1. 我们平常打开文件写入数据是顺序写吗？2. 还有如何进行随机写？是seek到某个位置开始写？但这样的话文件数据不是会被覆盖吗？ [1赞]

  作者回复2019-08-27 09:22:38
  A1：是的。
  A2：是的，不同的编程语言API不太一样，但都提供了类似将指针移动到文件中某个位置的功能。
  A3：会被覆盖。