15 | Kafka如何实现高性能IO？

2019-08-27 李玥

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 7

谢谢老师，今天讲到的点，我会在课下去读源码并写出文章

2019-08-27

作者回复

期待

2019-08-27

微微一笑 5

老师好，有些疑问希望老师解答下：

①rocketMq有consumeQueue，存储着offset，然后通过offset去commitlog找到对应的Message。通过看rocketmq的开发文档，通过offset去查询消息属于【随机读】，offset不是存储着消息在磁盘中的位置吗？为什么属于随机读呢？
②rocketMq的某个topic下指定的消息队列数，指的是consumeQueue的数量吗？
③性能上，顺序读优于随机读。rocketMq的实现上，在消费者与commitlog之间设计了consumeQueue的数据结构，导致不能顺序读，只能随机读。我的疑惑是，rocketMq为什么不像kafka那样设计，通过顺序读取消息，然后再根据topic、tag平均分配给不同的消费者实例，，这样消息积压的时候，直接增加消费者实例就可以了，不需要增加consumeQueue，这样也可以去除consumeQueue的存在呀？我在想consumeQueue存在的意义是什么呢？
哈哈，我的理解可能有些问题，希望老师指点迷津~

2019-08-27

作者回复

A1：这个过程就是随机读的过程。所有对文件的读写最终都要指定一个位置，都是按位置去读。随机读和顺序读的区别是，读取的数据是不是在文件中连续的一段。

A2：是的。

A3：RocketMQ的consumerQueue文件和Kafka的index file作用是差不多的，都是log文件（保存真正的消息）的索引，消费的时候，都需要先读索引，再读log，这个方面，两者并没有什么不同。它们存储设计的真正的差异的是log文件的设计，RocketMQ每个Broker只有一组log文件，而Kafka是每个分区一组log文件，你可以想一下，这两种设计各有什么优点和缺点。

另外，随机读和顺序读并没有严格的区分，不是非黑即白的。即使是最理想的顺序读，那它读第一个字节也是需要寻址的，这是不是一次随机读呢？随机读的时候，只要不是每次只读一个字节，你在读第二个字节的时候不就是顺序读吗？

所以，不用纠结这个概念，只要我们能做到读取数据的时候，尽量读连续的整块的数据，尽量减少寻址次数，性能就会更好。

2019-08-30

timmy21 2

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？

2019-08-27

作者回复

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。

2019-08-27

linaw 1

尝试回答微微一笑的问题，老师有空帮忙看下哦
老师好，有些疑问希望老师解答下：
①rocketMq有consumeQueue，存储着offset，然后通过offset去commitlog找到对应的Message。通过看rocketmq的开发文档，通过offset去查询消息属于【随机读】，offset不是存储着消息在磁盘中的位置吗？为什么属于随机读呢？
②rocketMq的某个topic下指定的消息队列数，指的是consumeQueue的数量吗？
③性能上，顺序读优于随机读。rocketMq的实现上，在消费者与commitlog之间设计了consumeQueue的数据结构，导致不能顺序读，只能随机读。我的疑惑是，rocketMq为什么不像kafka那样设计，通过顺序读取消息，然后再根据topic、tag平均分配给不同的消费者实例，，这样消息积压的时候，直接增加消费者实例就可以了，不需要增加consumeQueue，这样也可以去除consumeQueue的存在呀？我在想consumeQueue存在的意义是什么呢？
哈哈，我的理解可能有些问题，希望老师指点迷津~
①顺序读写是从头开始进行读写，比随机读比，不需要进行数据的位置定位只要从头开始进行读写，随机读需要进行数据位置的定位，如果能知道位置，通过位置进行随机读也会很快，rocketmq就是这样来优化io的随机读，快速读数据不一定是顺序读，也可以根据位置的随机读。
②rocketmq是messageQueue的数量，老师我有个好奇点rocketmq内部为什么要分读写队列，还有messageQueue内部没有存放消息，而是由消息message存放queueId和topic，消费者在消费的时候应该会有个consumerQueue才对，但是我在rocketmq代码里没有找到。
③rocketmq在内部用到consumeQueue，因为consumeQueue内部无需存放真正的消息，只要存储消息在commitLog的offset的位置、消息的storeSize，每次要消费的时候只要拿到位置和大小，就可以读到消息，并且无需每次根据topic和tag进行平均分配。

2019-08-28

作者回复

关于为什么分多个队列，我在之前的课程中提到过，和kafka分区一样，主要是为了能并行消费，提升消费性能。另外还有一个作用是，多个队列（分区）可以分布到多个节点上，提升主题整体的可用性。

2019-08-30

海罗沃德

1

Kafka既然是批量處理消息，那麼是怎麼實現Kafka的實時數據流計算呢？

2019-08-27

作者回复

这里面的批量处理和大数据中讲的“流和批”是二个不同的概念。

大数据中的“批量计算”是相对于“流计算”来说的，它指的是，一个计算任务处理一批数据，这批数据处理完了，这个计算任务就结束了。

我们这里的说的批量处理消息，是相对一条一条处理来说的，成批的处理会显著提升性能。

即使是在Flink或Storm这种纯正的流计算平台中，它对流数据进行传输、计算也是批量处理的。

2019-08-29

linqw

1

1、老师有个疑问，kafka在发送时，都会在客户端进行攒一波，然后过一定的时间，或者达到一定的大小发送出去，批量发送的时候，是把一批同一个topic下的分区的消息进行批量发送么？还是不管是属于同一分区的消息都进行批量发送，broker端是不会对批消息进行拆分成每一条，那这样消费端消费到的消息不是有可能有不是订阅的分区么？

2、学习到现在，有个感想，很多事情看似很简单，但是实际再做的时候都没那么简单，很多都得持之以恒，多思考、多实践、多动手，不然的话很多都是看懂，真正在使用的时候还是不知道如何下手。把很多小事、简单的事情做好本身就不是个简单的事情，目前有个想法打算把开源rocketmq读完，代码上写上注释和理解。

3、老师我一直有个疑惑点，如何才能当上架构师了，一方面硬核实力技术过硬，有整体的大局观，老师能否以你自身的经历给我们解惑下了

2019-08-27

作者回复

只有相同分区的消息才能组成同一个批消息。你的第三个问题太大了，改天有时间可以专题聊一下。

2019-08-30

业余草

1

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。

2019-08-27

作者回复

你可以分享一下，在使用Kafka的时候遇到了哪些问题。

2019-08-27

leslie

1

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。

2019-08-27

作者回复

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。

2019-08-27

龍蝦 0

Kafka Producer 调用同步 send() 成功返回，其实没法保证消息已经成功发送到 Kafka 服务器？

2019-09-11

作者回复

是这样的。

在Kafka中，这个Send是一个异步方法。如果要确保发送成功，你必须在提供的回调方法中去检查发送结果。

或者你也可以调用producer.send(record).get()来同步获取发送结果。

2019-09-12

张洪阆 0

如果没攒够数据量是否就一直不发呢，有没有超时机制？

2019-09-07

作者回复

有的，你可以看一下这个配置：linger.ms

2019-09-09

asdf100 0

Pagecache的作用和mysql里的innodb buffer作用基本一样的，都是读取时从buffer里取数据，若果数据不存在，则发生缺页，再从磁盘读取，放入buffer，都有加速客户端读取的效果，包括写操作。

2019-09-07

asdf100 0

建议 零复制 这款增减一个图吧，看图在理解的话就方便的多了，谢谢！

2019-09-07

asdf100 0

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。

客户端读取消息是，有个位置信息，这个信息在信息写入时，只在pagecache里出现还没有落盘时，位置信息就已经有了吗？这个位置信息怎么来的，原理又是什么？

2019-09-07

作者回复

对于消息队列中的消费位置，不同的MQ实现不一样，有的是在写入磁盘之前生成的，有的是在写入之后生成的，这个位置不是文件中的偏移量，一般是一个逻辑位置，含义是：“这个分区中的第几条消息”。

而数据在文件中的位置，是在写入之前就确定的，不管你用什么语言，写文件最终大多使用的是2个系统调用write或者pwrite，在调用这两个方法之前，其实位置已经是确定了。

PageCache映射的是文件中的一部分数据，那这些数据必然有一个固定地址（无论数据是否已经写到磁盘上了），否则操作系统也没办法做这个映射。

2019-09-09

asdf100 0

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。

这样的他，一批消息在多个消费端的情况下，这批消息这能落到一个消费端的吧？

2019-09-07

作者回复

只能由一个消费者来消费，因为在消费者收到消息解开这个“批消息”之前，这个批消息就是一个不能分割的整体。

2019-09-09

看不到de颜色 0

有一点疑惑还请老师解答一下。kafka为了保证消息丢失，客户端在发送消息时有三种acks可供选择。那如果kafka消息客户端都采用异步批量发送，那这三种参数还有意义吗？

2019-09-05

作者回复

我在之前的课中讲到过，在发送端，你需要在发送消息之后，检查发送结果，如果发送失败再进行重试或者执行其他的补偿。

这个原则同样适用于Kafka，并且和是否批量发送没关系，也和同步发送还是异步发送也没关系。

即使是异步发送，或者批量发送，只要你检查了发送结果（异步发送需要在回调方法中检查结果），并且发送结果是发送成功，就可以保证消息不丢。

2019-09-06

康师傅

0

请教一下

我的理解是：顺序写是针对某个分区而言的，那么如果单个节点上的topic数量很多，或者分区数很多，从整体来看应该还是会有很多的随机IO，因为会切换写不同的文件，这种情况下整体性能是不是就不高了？

这种场景下，除了增加节点，将分区分布到多个节点上，是否还有其他有效提升性能的办法？

2019-09-02

作者回复

是的，所以Kafka在分区非常多的情况下，性能是不如RocketMQ的。这种情况，除了想办法减少一些分区，确实没什么好的办法。

2019-09-03

牛牛

0

老师、想请教下 RocketMQ的queue

1. 生产者和消费者都有自己的Queue吗 ？ 还是生产者的Queue就是消费这的Queue呢 ？ 如果各自有独立的Queue、这两者之间有什么联系没 ？

2. ReadQueue 和 WriteQueue 的作用是什么呢 ？ 这两个Queue都是针对某个topic而言还是针对Producer和Consumer而言呢 ？ ReadQueue和WriteQueue的数量可以不一致么 ？

3. 真正存储消息的地方是Queue、而Broker只是负责消息的路由？Broker的slave和master上包含相同的数据信息、对吗？

有点儿乱了、希望老师或者路过的同学指点下 ^.^、先多谢啦~~~

2019-09-02

作者回复

生产者和消费者都没有自己的队列，这些队列在逻辑上属于主题，物理上保存在Broker上。或者你说的ReadQueue和WriteQueue是客户端在收发消息时候的缓存吗？这个是存在的，作用仅仅是在收发消息的时候缓存一部分消息而已。

关于队列相关的概念，你可以再重新看一下03和08这两节课，里面有比价详细的阐述。

2019-09-03

嘉木

0

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。

这个sendfile是不是从一个fd到另外一个fd的复制都是可以用的？

2019-09-02

作者回复

是的，并不限于磁盘到socket缓冲区。

2019-09-03

K-Li

0

老师，我知道kafka是攒一波消息后进行批处理的，那么在consumer消费到一条消息后如果处

理失败需要commit offset为上一条消息来重新消费的话是这么做到下一次来的就是刚刚处理失败的那条数据而不是"一批"里的下一条？

2019-08-30

作者回复

实际上是无法保证的，所以有可能会有重复消息。

2019-08-31

K-Li

0

老师，我知道kafka 是攒一波消息后做为一批来处理的。那消费端如果消费一条消息后处理失败要重新消费，这时候要重新commit offset

2019-08-30