在分布式消息模块的最后 2 个课时中，我将对消息队列中应用最广泛的 Kafka 和 RocketMQ 进行梳理，以便于你在应用中可以更好地进行消息队列选型。另外，这两款消息队列也是面试的高频考点。

所以，这一课时我们就一起来看一下，Kafka 是如何实现高性能的。

Kafka 的高性能

不知道你有没有了解过自己电脑的配置？

我们一般会认为高性能是和高配置联系在一起的，比如大内存比小内存快，8 核的机器比 4 核的机器快。我身边也有一些朋友是攒机爱好者，对各种硬件配置如数家珍。

对于服务器来说，家用电脑的性能与配置的关系也同样适用——价格更昂贵的服务器会有更好的性能——这并不是一件需要大张旗鼓去讲述的事情。但 Kafka 所实现的高性能不需要太高配置的机器，它使用普通服务器就能实现 TB 级别的传输性能。这一点也是 Kafka 对外宣传的一个特性，也正是因为这一点，Kafka 被广泛运用于大数据处理、流式计算、各类日志监控等需要处理海量数据的场景。

Kafka 实现高性能的手段，是面试中经常被问到的问题。下面我从 Kafka 的磁盘读写、批量优化、零拷贝等方面，对 Kafka 的高性能特性进行分析。

分析 Kafka 的高性能会涉及操作系统的一些知识，比如文件系统、PageCache等，作为大学计算机专业的必修课，这些概念就不展开了。如果你觉得这方面比较生疏，可以回顾下操作系统课程的相关知识，找一些经典教材来学习。

磁盘顺序读写

Kafka 消息是存储在磁盘上的，大家都知道，普通的机械磁盘读取是比较慢的，那 Kafka 文件在磁盘上，如何实现高性能的读写呢？

Kafka 对磁盘的应用，得益于消息队列的存储特性。与普通的关系型数据库、各类 NoSQL 数据库等不同，消息队列对外提供的主要方法是**生产和消费**，不涉及数据的 CRUD。所以在写入磁盘时，可以使用顺序追加的方式来避免低效的磁盘寻址。

我们知道，数据存储在硬盘上，而硬盘有机械硬盘和固态硬盘之分。机械硬盘成本低、容量大，但每次读写都会寻址，再写入数据（在机械硬盘上，寻址是一个物理动作，耗时最大）；SSD 固态硬盘性能很高，有着非常低的寻道时间和存取时间，但成本也特别高。

为了提高在机械硬盘上读写的速度，Kafka 使用了顺序读写。在一个分区内，Kafka 采用 append 的方式进行顺序写入，这样即使是普通的机械磁盘，也可以有很高的性能。

除了顺序读写，在提到磁盘写入的时候，还有一个问题避免不了，那就是何时进行刷盘。

在 Linux 系统中，当我们把数据写入文件系统之后，其实数据是存放在操作系统的 page cache 里面，并没有刷到磁盘上，如果服务器宕机，数据就丢失了。

写到磁盘的过程叫作 Flush。刷盘一般有两种方式，一种是依靠操作系统进行管理，定时刷盘，另一种则是同步刷盘，比如调用 fsync 等系统函数。

同步刷盘保证了数据的可靠性，但是会降低整体性能。Kafka 可以配置异步刷盘，不开启同步刷盘，异步刷盘不需要等写入磁盘后返回消息投递的 ACK，所以它提高了消息发送的吞吐量，降低了请求的延时，这也是 Kafka 磁盘高性能的一个原因。

批量操作优化

批量是一个常见的优化思路，比如大家熟悉的 Redis，就实现了 pipeline 管道批量操作。Kafka 在很多地方也应用了批量操作进行性能优化。

Kafka 的批量包括批量写入、批量发布等。它在消息投递时会将消息缓存起来，然后批量发送；同样，消费端在消费消息时，也不是一条一条处理的，而是批量进行拉取，提高了消息的处理速度。

除了批量以外，Kafka 的数据传输还可以配置压缩协议，比如 Gzip 和 Snappy 压缩协议。虽然在进行数据压缩时会消耗少量的 CPU 资源，但可以减少网络传输的数据大小、优化网络 IO、提升传输速率。

Sendfile 零拷贝

零拷贝是什么？它是操作系统文件读写的一种技术。

零拷贝不是不需要拷贝，而是减少不必要的拷贝次数，这里会涉及 Linux 用户态和内核态的区别。

用户进程是运行在用户空间的，不能直接操作内核缓冲区的数据。所以在用户进程进行系统调用的时候，会由用户态切换到内核态，待内核处理完之后再返回用户态。

传统的 IO 流程，需要先把数据拷贝到内核缓冲区，再从内核缓冲拷贝到用户空间，应用程序处理完成以后，再拷贝回内核缓冲区。这个过程中发生了多次数据拷贝。

为了减少不必要的拷贝，Kafka 依赖 Linux 内核提供的 Sendfile 系统调用。 在 Sendfile 方法中，数据在内核缓冲区完成输入和输出，不需要拷贝到用户空间处理，这也就避免了重复的数据拷贝。在具体的操作中，Kafka 把所有的消息都存放在单独的文件里，在消息投递时直接通过 Sendfile 方法发送文件，减少了上下文切换，因此大大提高了性能。

MMAP 技术

Kafka 是使用 Scala 语言开发的。Scala 运行在 Java 虚拟机上，也就是说 Kafka 节点运行需要 JVM 的支持，但是 Kafka 并不直接依赖 JVM 堆内存。如果 Kafka 所有的数据操作都在堆内存中进行，则会对堆内存造成非常大的压力，影响垃圾回收处理，增加 JVM 的停顿时间和整体延迟。

因此，除了 Sendfile 之外，还有一种零拷贝的实现技术，即 Memory Mapped Files。

Kafka 使用 Memory Mapped Files 完成内存映射，Memory Mapped Files 对文件的操作不是 write/read，而是直接对内存地址的操作。如果是调用文件的 read 操作，则把数据先读取到内核空间中，然后再复制到用户空间。 但 MMAP 可以将文件直接映射到用户态的内存空间，省去了用户空间到内核空间复制的开销，所以说 MMAP 也是一种零拷贝技术。

那 MMAP 和上面的 Sendfile 有什么区别呢?

MMAP 和 Sendfile 并没有本质上的区别，它们都是零拷贝的实现。零拷贝是一种技术思想，除了我们说到的这两种，还有DMA，以及缓冲区共享等方式，感兴趣的同学可以去扩展了解一下。

总结

这一课时讲解了 Kafka 如何实现高性能，介绍了顺序读写、批量优化、零拷贝等技术，对于大部分业务开发的同学，这部分知识了解即可。

Kafka 的高性能实现原理，在很多地方都有应用，比如 Netty 中也有零拷贝技术。Linux 中，一切皆文件，Netty 关注的是网络 IO 的传输，Kafka 等存储关注的是文件 IO 的传输，但在操作系统中都是 IO 操作，在优化手段上非常类似。

另外，上面提到的 Sendfile 可以大幅提升文件传输性能，在 Apache、Nginx 等 Web 服务器当中，都有相关的应用。感兴趣的同学可以了解下 Netty 等网络组件的性能优化方式，欢迎留言进行分享。

精选评论

****华:**

> 干货满满！