为什么Kafka那么快

Original 2016-07-18 fireflyc 写程序的康德

网上有很多Kafka的测试文章，测试结果通常都是“吊打”其他MQ。感慨它的牛B之余我觉得必要仔细分析一下它如此快速的原因。这篇文章不同于其他介绍Kafka使用或者技术实现的文章，我会重点解释——为什么真快。（当然不是因为它用了Scala！！！！）

生产者（写入数据）

生产者（producer）是负责向Kafka提交数据的，我们先分析这一部分。
Kafka会把收到的消息都写入到硬盘中，它绝对不会丢失数据。为了优化写入速度Kafak采用了两个技术，**顺序写入**和**MMFile**。

顺序写入

因为硬盘是机械结构，每次读写都会寻址->写入，其中寻址是一个“机械动作”，它是最耗时的。所以硬盘最“讨厌”随机I/O，最喜欢顺序I/O。为了提高读写硬盘的速度，Kafka就是使用顺序I/O。

上图就展示了Kafka是如何写入数据的，**每一个Partition其实都是一个文件**，收到消息后Kafka会把数据插入到文件末尾（虚框部分）。
这种方法有一个缺陷——**没有办法删除数据**，所以Kafka是不会删除数据的，它会把所有的数据都保留下来，每个消费者（Consumer）对每个Topic都有一个offset用来表示**读取到了第几条数据**。

上图中有两个消费者，Consumer1有两个offset分别对应Partition0、Partition1（假设每一个Topic一个Partition）；Consumer2有一个offset对应Partition2。这个offset是由客户端SDK负责保存的，Kafka的Broker完全无视这个东西的存在；一般情况下SDK会把它保存到zookeeper里面。(所以需要给Consumer提供zookeeper的地址)。
如果不删除硬盘肯定会被撑满，所以Kakfa提供了两种策略来删除数据。一是基于时间，二是基于partition文件大小。具体配置可以参看它的配置文档。

Memory Mapped Files

即便是顺序写入硬盘，硬盘的访问速度还是不可能追上内存。所以Kafka的数据并**不是实时的写入硬盘**，它充分利用了现代操作系统**分页存储**来利用内存提高I/O效率。
Memory Mapped Files(后面简称mmap)也被翻译成**内存映射文件**，在64位操作系统中一般可以表示20G的数据文件，它的工作原理是直接利用操作系统的Page来实现文件到物理内存的直接映射。完成映射之后你对物理内存的操作会被同步到硬盘上（操作系统在适当的时候）。

通过mmap，进程像读写硬盘一样读写内存（当然是虚拟机内存），也不必关心内存的大小有虚拟内存为我们兜底。
使用这种方式可以获取很大的I/O提升，**省去了用户空间到内核空间**复制的开销（调用文件的read会把数据先放到内核空间的内存中，然后再复制到用户空间的内存中。）也有一个很明显的缺陷——不可靠，**写到mmap中的数据并没有被真正的写到硬盘，操作系统会在程序主动调用flush的时候才把数据真正的写到硬盘。**Kafka提供了一个参数——producer.type来控制是不是主动flush，如果Kafka写入到mmap之后就立即flush然后再返回Producer叫**同步**(sync)；写入mmap之后立即返回Producer不调用flush叫**异步**(async)。
mmap其实是Linux中的一个函数就是用来实现内存映射的，谢谢Java NIO，它给我提供了一个mappedbytebuffer类可以用来实现内存映射（所以是沾了Java的光才可以如此神速和Scala没关系！！）

消费者（读取数据）

Kafka使用磁盘文件还想快速？这是我看到Kafka之后的第一个疑问，ZeroMQ完全没有任何服务器节点，也不会使用硬盘，按照道理说它应该比Kafka快。可是实际测试下来它的速度还是被Kafka“吊打”。**“一个用硬盘的比用内存的快”，**

这绝对违反常识；如果这种事情发生说明——它作弊了。
没错，Kafka“作弊”。无论是**顺序写入**还是**mmap**其实都是作弊的准备工作。

如何提高Web Server静态文件的速度

仔细想一下，一个Web Server传送一个静态文件，如何优化？答案是zero copy。传统模式下我们从硬盘读取一个文件是这样的

先复制到内核空间（read是系统调用，放到了DMA，所以用内核空间），然后复制到用户空间(1,2)；从用户空间重新复制到内核空间（你用的socket是系统调用，所以它也有自己的内核空间），最后发送给网卡（3、4）。

Zero Copy中直接从内核空间（DMA的）到内核空间（Socket的），然后发送网卡。
这个技术非常普遍，The C10K problem 里面也有很详细的介绍，Nginx也是用的这种技术，稍微搜一下就能找到很多资料。

Java的NIO提供了FileChannle，它的transferTo、transferFrom方法就是Zero Copy。

Kafka是如何耍赖的

想到了吗？Kafka把所有的消息都存放在一个一个的文件中，**当消费者需要数据的时候Kafka直接把“文件”发送给消费者。**这就是秘诀所在，比如：**10W的消息组合在一起是10MB的数据量，然后Kafka用类似于发文件的方式直接扔出去了，如果消费者和生产者之间的网络非常好（只要网络稍微正常一点10MB根本不是事。。。家里上网都是100Mbps的带宽了），10MB可能只需要1s。所以答案是——10W的TPS，Kafka每秒钟处理了10W条消息。**
可能你说：不可能把整个文件发出去吧？里面还有一些不需要的消息呢？是的，Kafka作为一个“高级作弊分子”自然要把作弊做的有逼格。Zero Copy对应的是sendfile这个函数（以Linux为例），这个函数接受

- out_fd作为输出（一般及时socket的句柄）
- in_fd作为输入文件句柄
- off_t表示in_fd的偏移（从哪里开始读取）
- size_t表示读取多少个

没错，Kafka是用mmap作为文件读写方式的，它就是一个文件句柄，所以直接把它传给sendfile；偏移也好解决，用户会自己保持这个offset，每次请求都会发送这个offset。（还记得吗？放在zookeeper中的）；数据量更容易解决了，如果消费者想要更快，就全部扔给消费者。如果这样做一般情况下消费者肯定直接就被**压死了**；所以Kafka提供了的两种方式——Push，我全部扔给你了，你死了不管我的事情；Pull，好吧你告诉我你需要多少个，我给你多少个。

总结

Kafka速度的秘诀在于，它把所有的消息都变成一个的文件。通过mmap提高I/O速度，写入数据的时候它是末尾添加所以速度最优；读取数据的时候配合sendfile直接暴力输出。阿里的RocketMQ也是这种模式，只不过是用Java写的。

单纯的去测试MQ的速度没有任何意义，Kafka这种“暴力”、“流氓”、“无耻”的做法已经脱了MQ的底裤，更像是一个暴力的“数据传送器”。**所以对于一个MQ的评价只以速度论英雄，世界上没人能干的过Kafka，我们设计的时候不能听信网上的流言蜚语——“Kafka最快，大家都在用，所以我们的MQ用Kafka没错”。在这种思想的作用下，你可能根本不会关心“失败者”；而实际上可能这些“失败者”是更适合你业务的MQ。**