图文 57 精益求精：消费者到底是根据什么策略从Master或Slave上拉



详情 评论

**精益求精：消费者到底是根据什么策略从Master或Slave上拉取消息的?**

石杉老哥重磅力作：《互联网java工程师面试突击》（第3季）**【强烈推荐】**：




全程真题驱动，精研Java面试中**6大专题的高频考点**，从面试官的角度剖析面试

（点击下方蓝字试听）

**《互联网Java工程师面试突击》（第3季）**

---

**1、Broker读写分离架构的回顾**

上一次我们已经分析了消息消费的整个流程和原理，这次我们来看一个比较关键的问题，也是很多人都关注的问题，就是Broker实现高可用架构的时候是有主从之分的

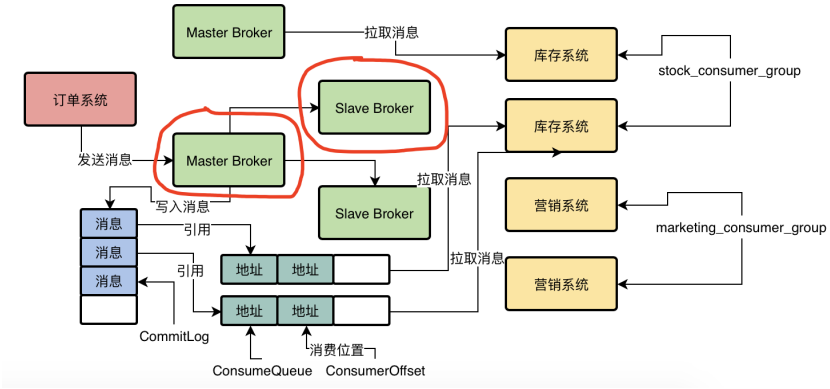
之前我们提到过，消息消费，可以从Master Broker拉取，也可以从Slave Broker拉取，具体是要看机器负载来定。

所以很多人都会有一个疑问，那到底什么时候从Master Broker拉取，什么时候从Slave Broker拉取。

所以我们先来简单回顾一下，之前我们对Broker的读写分离架构是怎么描述的。

之前我们说过，刚开始消费者都是连接到Master Broker机器去拉取消息的，然后如果Master Broker机器觉得自己负载比较高，就会告诉消费者机器，下次可以从Slave Broker机器去拉取。

我们看下面的图，图里示意是说Master Broker和Slave Broker都可以拉取消息。

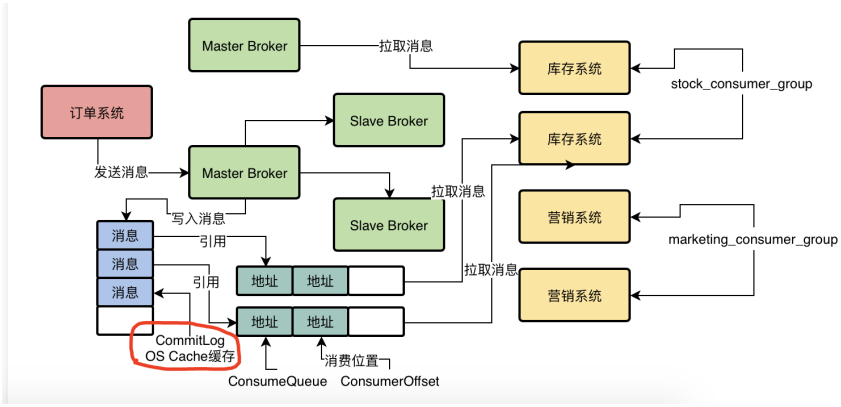



**2、CommitLog基于os cache提升写性能的回顾**

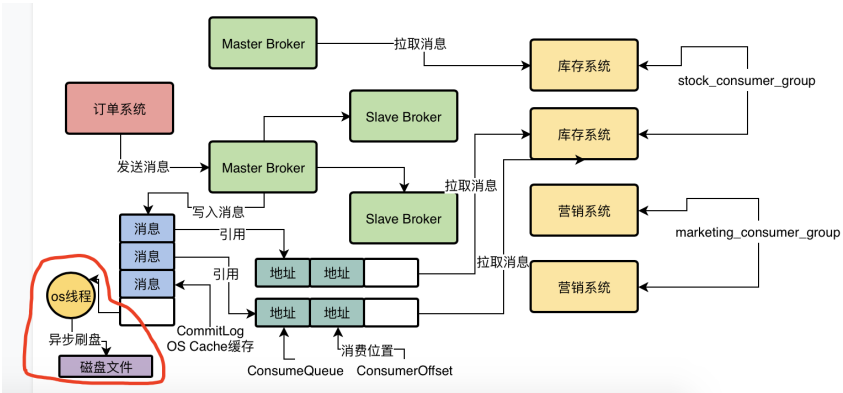
接着如果大家要搞明白到底什么时候从Master Broker拉取消息，什么时候从Slave Broker拉取消息，首先得搞明白一个很关键的问题，那就是拉取消息的时候必然会先读取ConsumeQueue文件，这个ConsumeQueue文件的读取是如何优化的？

要搞明白这个ConsumeQueue文件的读取是如何进行性能优化的，我们又得先回顾一下之前讲过的CommitLog文件写入的优化原理，其实他本质就是基于os cache来进行优化的

也就是说，broker收到一条消息，会写入CommitLog文件，但是会先把CommitLog文件中的数据写入os cache(操作系统管理的缓存)中去，如下图。




然后os自己有后台线程，过一段时间后会异步把os cache缓存中的CommitLog文件的数据刷入磁盘中去，如下图。




就是依靠这个写入CommitLog时先进入os cache缓存，而不是直接进入磁盘的机制，就可以实现broker写CommitLog文件的性能是内存写级别的，这才能实现broker超高的消息接入吞吐量。

**3、一个很关键的问题：ConsumeQueue文件也是基于os cache的**

所以接下来我们就可以看一个很关键的问题了，那就是ConsumeQueue会被大量的消费者发送的请求给高并发的读取，所以ConsumeQueue文件的读操作是非常频繁的，而且同时会极大的影响到消费者进行消息拉取的性能和消费吞吐量。

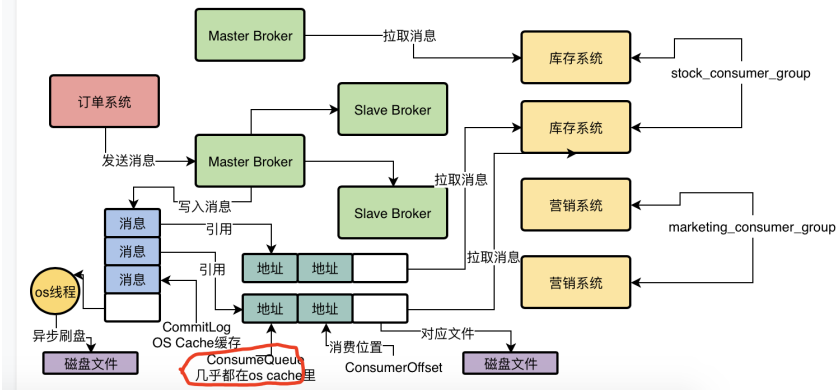
所以实际上broker对ConsumeQueue文件同样也是基于os cache来进行优化的

也就是说，对于Broker机器的磁盘上的大量的ConsumeQueue文件，在写入的时候也都是优先进入os cache中的

而且os自己有一个优化机制，就是读取一个磁盘文件的时候，他会自动把磁盘文件的一些数据缓存到os cache中。

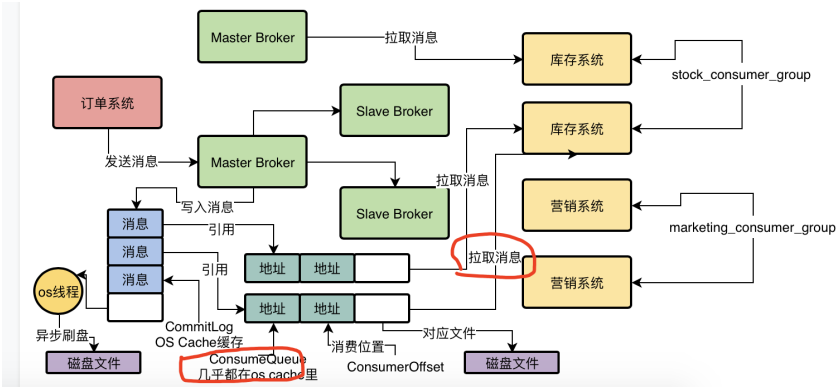
而且大家之前知道ConsumeQueue文件主要是存放消息的offset，所以每个文件很小，30万条消息的offset就只有5.72MB而已。所以实际上ConsumeQueue文件们是不占用多少磁盘空间的，他们整体数据量很小，几乎可以完全被os缓存在内存cache里。

大家看下面的图，我们示意了ConsumeQueue文件几乎都是放在os cache里的。




所以实际上在消费者机器拉取消息的时候，第一步大量的频繁读取ConsumeQueue文件，几乎可以说就是跟读内存里的数据的性能是一样的，通过这个就可以保证数据消费的高性能以及高吞吐

下面的图示意了消息拉取的时候，都是从os cache里读取ConsumeQueue的数据的。




返回
前进
重新加载
打印

**4、第二个关键问题：CommitLog是基于os cache+磁盘一起读取的**

接着我们来看第二个比较关键的问题，在进行消息拉取的时候，先读os cache里的少量ConsumeQueue的数据，这个性能是极高的，然后第二步就是要根据你读取到的offset去CommitLog里读取消息的完整数据了。

那么大家可以思考一下，这个从CommitLog里读取消息完整数据是如何读取的？是从os cache里读取？还是从磁盘里读取？

**答案是：两者都有**

因为CommitLog是用来存放消息的完整数据的，所以内容量是很大的，毕竟他一个文件就要1GB，所以整体完全有可能多达几个TB。

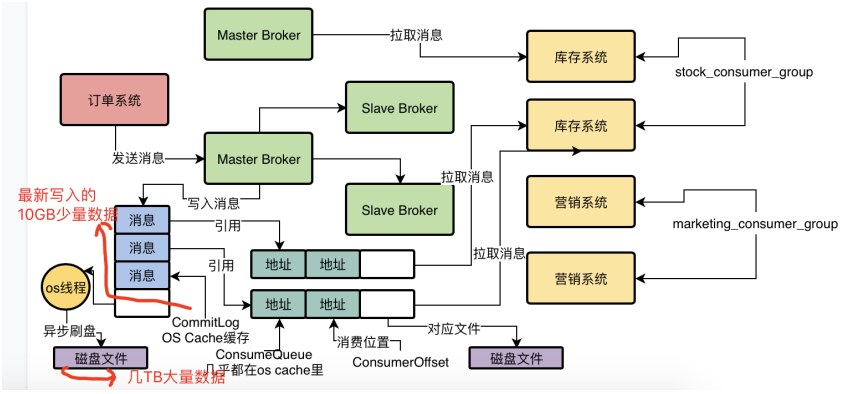
所以你思考一下，这么多的数据，可能都放在os cache里吗？

明显是不可能的，因为os cache用的也是机器的内存，一般多也就几十个GB而已，何况Broker自身的JVM也要用一些内存，留个os cache的内存只是一部分罢了，比如10GB~20GB的内存，所以os cache对于CommitLog而言，是无法把他全部数据都放在里面给你读取的！

也就是说，os cache对于CommitLog而言，主要是提升文件写入性能，当你不停的写入的时候，很多最新写入的数据都会先停留在os cache里，比如这可能有10GB~20GB的数据。

之后os会自动把cache里的比较旧的一些数据刷入磁盘里，腾出来空间给更新写入的数据放在os cache里，所以大部分数据可能多达几个TB都是在磁盘上的

我们看下面图里的示意，就是这个意思。




所以最终结论来了，当你拉取消息的时候，可以轻松从os cache里读取少量的ConsumeQueue文件里的offset，这个性能是极高的，但是当你去CommitLog文件里读取完整消息数据的时候，会有两种可能。

**第一种可能**，如果你读取的是那种刚刚写入CommitLog的数据，那么大概率他们还停留在os cache中，此时你可以顺利的直接从os cache里读取CommitLog中的数据，这个就是内存读取，性能是很高的。

**第二种可能**，你也许读取的是比较早之前写入CommitLog的数据，那些数据早就被刷入磁盘了，已经不在os cache里了，那么此时你就只能从磁盘上的文件里读取了，这个性能是比较差一些的。

**5、什么时候会从os cache读？什么时候会从磁盘读？**

接着我们看下面一个问题，那么什么时候会从os cache读？什么时候会从磁盘读呢？

其实这个问题很简单了，如果你的消费者机器一直快速的在拉取和消费处理，紧紧的跟上了生产者写入broker的消息速率，那么你每次拉取几乎都是在拉取最近人家刚写入CommitLog的数据，那几乎都在os cache里。

但是如果broker的负载很高，导致你拉取消息的速度很慢，或者是你自己的消费者机器拉取到一批消息之后处理的时候性能很低，处理的速度很慢，这都会导致你跟不上生产者写入的速率。

比如人家都写入10万条数据了，结果你才拉取了2万条数据，此时有5万条最新的数据是在os cache里，有3万条你还没拉取的数据是在磁盘里，那么当后续你再拉取的时候，必然很大概率是从磁盘里读取早就刷入磁盘的3万条数据。

接着之前在os cache里的5万条数据可能又被刷入磁盘了，取而代之的是更新的几万条数据在os cache里，然后你再次拉取的时候，又会从磁盘里读取刷入磁盘里的5万条数据，相当于你每次都在从磁盘里读取数据了！

**6、Master Broker什么时候会让你从Slave Broker拉取数据？**

经过了上述大量的铺垫之后，我们终于可以解释这个关键性的问题了，到底什么时候Master Broker会让你从Slave Broker拉取数据？

其实这个问题我们上一个小节已经解释了一部分了，假设此时你的broker里已经写入了10万条数据，但是你仅仅拉取了2万条数据，下次你拉取的时候，是从第2万零1条数据开始继续往后拉取的，是不是？

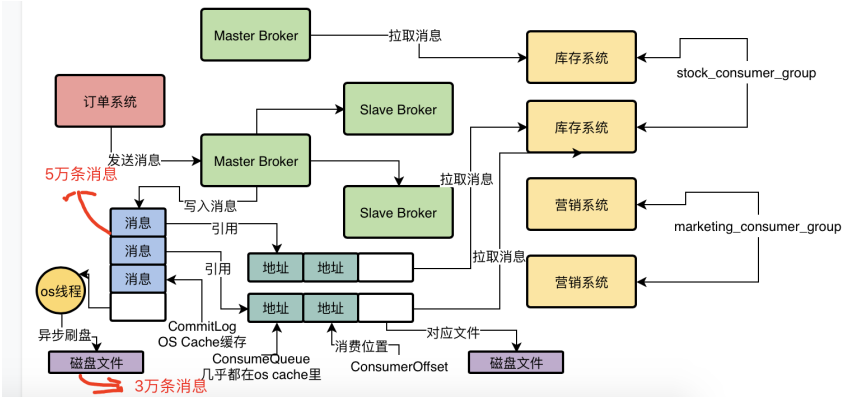
也就是说，此时你有8万条数据是没有拉取的！

然后broker自己是知道机器上当前的整体物理内存有多大的，而且他也知道自己可用的最大空间占里面的比例，他是知道自己的消息最多可以在内存里放多少的！比如他心知肚明，他最多也就在内存里放5万条消息而已！

因为他知道，他最多只能利用10GB的os cache去放消息，这么多内存最多也就放5万左右的消息。

然后这个时候你过来拉取消息，他发现你还有8万条消息没有拉取，这个8万条消息他发现是大于10GB内存最多存放的5万条消息的，那么此时就说明，肯定有3万条消息目前是在磁盘上的，不在os cache内存里！

我们看下面的图，就分别示意了os cache里和磁盘上你没拉取的消息数量。




所以他经过上述判断，会发现此时你很大概率会从磁盘里加载3万条消息出来！他会认为，出现这种情况，很可能是因为自己作为master broker负载太高了，导致没法及时的把消息给你，所以你落后的进度比较多。

这个时候，他就会告诉你，我这次给你从磁盘里读取3万条消息，但是下次你还是从slave broker去拉取吧！

以上就是这个关键问题的解答，本质是对比你当前没有拉取消息的数量和大小，以及最多可以存放在os cache内存里的消息的大小，如果你没拉取的消息超过了最大能使用的内存的量，那么说明你后续会频繁从磁盘加载数据，此时就让你从slave broker去加载数据了！

**7、对今日内容的一点总结**

今天我们给大家分析了一下消费者在拉取消息的时候到底是怎么选择Broker的，是从Master Broker还是Slave Broker去拉取，这里牵扯到了os cache，内存读，磁盘读，内存数据刷入磁盘等一系列的问题，包括了以下一些点：

Broker读写分离架构的回顾
CommitLog基于os cache实现写入性能优化的回顾
ConsumeQueue基于os cache实现读取性能优化的原理
CommitLog是基于os cache+磁盘一起进行读取的原理
到底什么时候从os cache里读取CommitLog？什么时候从磁盘里读取？
Master Broker什么时候回让你去Slave Broker读取？



## End



---

**狸猫技术窝其他精品专栏推荐：**

《从零开始带你成为JVM实战高手》
《21天Java 面试突击训练营》（分布式篇）（现更名为：**互联网Java工程师面试突击第2季**）
互联网Java工程师面试突击（第1季）

---

**重要说明：**

**如何提问**：每篇文章都有评论区，大家可以尽情在评论区留言提问，我会逐一答疑

**如何加群**：购买了狸猫技术窝专栏的小伙伴都可以加入**狸猫技术交流群**

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群？》**（购买后可见）**