07 | 消息积压了该如何处理？

2019-08-06 李玥

你好，我是李玥。这节课我们来聊一聊关于消息积压的问题。

据我了解，在使用消息队列遇到的问题中，消息积压这个问题，应该是最常遇到的问题了，并且，这个问题还不太好解决。

我们都知道，消息积压的直接原因，一定是系统中的某个部分出现了性能问题，来不及处理上游发送的消息，才会导致消息积压。

所以，我们先来分析下，在使用消息队列时，如何来优化代码的性能，避免出现消息积压。然后再来看看，如果你的线上系统出现了消息积压，该如何进行紧急处理，最大程度地避免消息积压对业务的影响。

优化性能来避免消息积压

在使用消息队列的系统中，对于性能的优化，主要体现在生产者和消费者这一收一发两部分的业务逻辑中。对于消息队列本身的性能，你作为使用者，不需要太关注。为什么这么说呢？

主要原因是，对于绝大多数使用消息队列的业务来说，消息队列本身的处理能力要远大于业务系统的处理能力。主流消息队列的单个节点，消息收发的性能可以达到每秒钟处理几万至几十万条消息的水平，还可以通过水平扩展Broker的实例数成倍地提升处理能力。

而一般的业务系统需要处理的业务逻辑远比消息队列要复杂，单个节点每秒钟可以处理几百到几

千次请求，已经可以算是性能非常好的了。所以，对于消息队列的性能优化，我们更关注的是，**在消息的收发两端，我们的业务代码怎么和消息队列配合，达到一个最佳的性能。**

1. 发送端性能优化

发送端业务代码的处理性能，实际上和消息队列的关系不大，因为一般发送端都是先执行自己的业务逻辑，最后再发送消息。**如果说，你的代码发送消息的性能上不去，你需要优先检查一下，是不是发消息之前的业务逻辑耗时太多导致的。**

对于发送消息的业务逻辑，只需要注意设置合适的并发和批量大小，就可以达到很好的发送性能。为什么这么说呢？

我们之前的课程中讲过Producer发送消息的过程，Producer发消息给Broker，Broker收到消息后返回确认响应，这是一次完整的交互。假设这一次交互的平均时延是1ms，我们把这1ms的时间分解开，它包括了下面这些步骤的耗时：

- 发送端准备数据、序列化消息、构造请求等逻辑的时间，也就是发送端在发送网络请求之前的耗时；
- 发送消息和返回响应在网络传输中的耗时；
- Broker处理消息的时延。

如果是单线程发送，每次只发送1条消息，那么每秒只能发送 1000ms / 1ms * 1条/ms = 1000条消息，这种情况下并不能发挥出消息队列的全部实力。

无论是增加每次发送消息的批量大小，还是增加并发，都能成倍地提升发送性能。至于到底是选择批量发送还是增加并发，主要取决于发送端程序的业务性质。简单来说，只要能够满足你的性能要求，怎么实现方便就怎么实现。

比如说，你的消息发送端是一个微服务，主要接受RPC请求处理在线业务。很自然的，微服务在处理每次请求的时候，就在当前线程直接发送消息就可以了，因为所有RPC框架都是多线程支持多并发的，自然也就实现了并行发送消息。并且在线业务比较在意的是请求响应时延，选择批量发送必然会影响RPC服务的时延。这种情况，比较明智的方式就是通过并发来提升发送性能。

如果你的系统是一个离线分析系统，离线系统在性能上的需求是什么呢？它不关心时延，更注重整个系统的吞吐量。发送端的数据都是来自于数据库，这种情况就更适合批量发送，你可以批量从数据库读取数据，然后批量来发送消息，同样用少量的并发就可以获得非常高的吞吐量。

2. 消费端性能优化

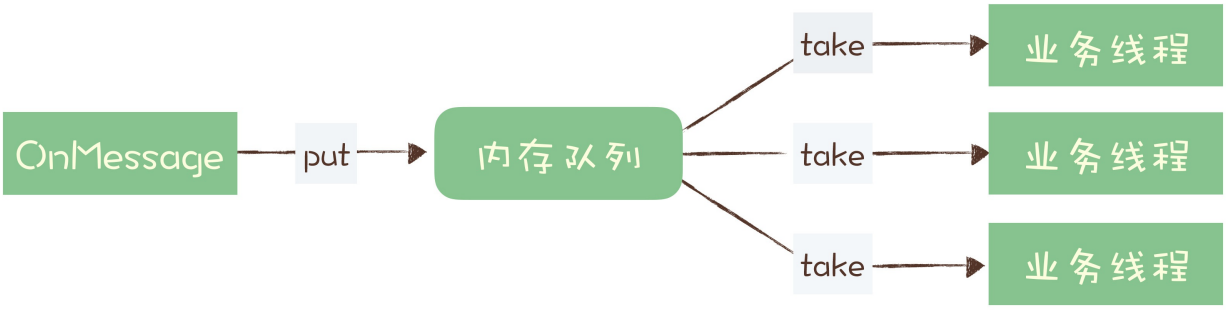
使用消息队列的时候，大部分的性能问题都出现在消费端，如果消费的速度跟不上发送端生产消息的速度，就会造成消息积压。如果这种性能倒挂的问题只是暂时的，那问题不大，只要消费端的性能恢复之后，超过发送端的性能，那积压的消息是可以逐渐被消化掉的。

要是消费速度一直比生产速度慢，时间长了，整个系统就会出现问题，要么，消息队列的存储被填满无法提供服务，要么消息丢失，这对于整个系统来说都是严重故障。

所以，我们在设计系统的时候，**一定要保证消费端的消费性能要高于生产端的发送性能，这样的系统才能健康的持续运行。**

消费端的性能优化除了优化消费业务逻辑以外，也可以通过水平扩容，增加消费端的并发数来提升总体的消费性能。特别需要注意的一点是，**在扩容Consumer的实例数量的同时，必须同步扩容主题中的分区（也叫队列）数量，确保Consumer的实例数和分区数量是相等的。**如果Consumer的实例数量超过分区数量，这样的扩容实际上是没有效果的。原因我们之前讲过，因为对于消费者来说，在每个分区上实际上只能支持单线程消费。

我见到过很多消费程序，他们是这样来解决消费慢的问题的：



它收消息处理的业务逻辑可能比较慢，也很难再优化了，为了避免消息积压，在收到消息的OnMessage方法中，不处理任何业务逻辑，把这个消息放到一个内存队列里面就返回了。然后它可以启动很多的业务线程，这些业务线程里面是真正处理消息的业务逻辑，这些线程从内存队列里取消息处理，这样它就解决了单个Consumer不能并行消费的问题。

这个方法是不是很完美地实现了并发消费？请注意，这是一个非常常见的错误方法！ 为什么错误？因为会丢消息。如果收消息的节点发生宕机，在内存队列中还没来及处理的这些消息就会丢失。关于“消息丢失”问题，你可以回顾一下我们的专栏文章《05 | 如何确保消息不会丢失？》。

消息积压了该如何处理？

还有一种消息积压的情况是，日常系统正常运转的时候，没有积压或者只有少量积压很快就消费掉了，但是某一个时刻，突然就开始积压消息并且积压持续上涨。这种情况下需要你在短时间内找到消息积压的原因，迅速解决问题才不至于影响业务。

导致突然积压的原因肯定是多种多样的，不同的系统、不同的情况有不同的原因，不能一概而论。但是，我们排查消息积压原因，是有一些相对固定而且比较有效的方法的。

能导致积压突然增加，最粗粒度的原因，只有两种：要么是发送变快了，要么是消费变慢了。

大部分消息队列都内置了监控的功能，只要通过监控数据，很容易确定是哪种原因。如果是单位时间发送的消息增多，比如说是赶上大促或者抢购，短时间内不太可能优化消费端的代码来提升消费性能，唯一的方法是通过扩容消费端的实例数来提升总体的消费能力。

如果短时间内没有足够的服务器资源进行扩容，没办法的办法是，将系统降级，通过关闭一些不重要的业务，减少发送方发送的数据量，最低限度让系统还能正常运转，服务一些重要业务。

还有一种不太常见的情况，你通过监控发现，无论是发送消息的速度还是消费消息的速度和原来都没什么变化，这时候你需要检查一下你的消费端，是不是消费失败导致的一条消息反复消费这种情况比较多，这种情况也会拖慢整个系统的消费速度。

如果监控到消费变慢了，你需要检查你的消费实例，分析一下是什么原因导致消费变慢。优先检查一下日志是否有大量的消费错误，如果没有错误的话，可以通过打印堆栈信息，看一下你的消费线程是不是卡在什么地方不动了，比如触发了死锁或者卡在等待某些资源上了。

小结

这节课我们主要讨论了2个问题，一个是如何在消息队列的收发两端优化系统性能，提前预防消息积压。另外一个问题是，当系统发生消息积压了之后，该如何处理。

优化消息收发性能，预防消息积压的方法有两种，增加批量或者是增加并发，在发送端这两种方法都可以使用，在消费端需要注意的是，增加并发需要同步扩容分区数量，否则是起不到效果的。

对于系统发生消息积压的情况，需要先解决积压，再分析原因，毕竟保证系统的可用性是首先要解决的问题。快速解决积压的方法就是通过水平扩容增加Consumer的实例数量。

思考题

课后请你思考一下，在消费端是否可以通过批量消费的方式来提升消费性能？在什么样场景下，适合使用这种方法？或者说，这种方法有什么局限性？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

精选留言

约书亚 11

批量消费有意义的场景要求：1.要么消费端对消息的处理支持批量处理，比如批量入库 2. 要么消费端支持多线程/协程并发处理，业务上也允许消息无序。3. 或者网络带宽在考虑因素内，需要减少消息的overhead。

批量消费的局限性：1. 需要一个整体ack的机制，一旦一条靠前的消息消费失败，可能会引起很多消息重试。2. 多线程下批量消费速度受限于最慢的那个线程。

但其实以上局限并没有影响主流MQ的实现了批量功能。

2019-08-06

岁月安然 7

1、要求消费端能够批量处理或者开启多线程进行单条处理

2、批量消费一旦某一条数据消费失败会导致整批数据重复消费

3、对实时性要求不能太高，批量消费需要Broker积累到一定消费数据才会发送到Consumer

2019-08-06

作者回复

2019-08-07

iLeGeND 5

老师好，我一直理解，消息积压不是一种正常的现象吗？来不及处理的消息先在消息队列中存着，缓解下游系统的压力，让上下游系统在时间上解偶，，听了今天的课，感觉理解的不太一

样，希望老师解答一下

2019-08-06

作者回复

消息积压是正常现象，积压越来越多就需要处理了。

就像一个水库，日常蓄水是正常的，但下游泄洪能力太差，导致水库水位一直不停的上涨，这个就不正常了。

2019-08-06

蓝魔、 3

消费端通过丢入队列，并使用多线程来提高并发，老师讲到可能丢消息，但也要是某种条件成立，比如开启了自动提交，如果改为手动提交完全可以避免丢消息，只是可能出现重复消费问题，至少不会丢

2019-08-06

C J J 2

一、如何预防消息积压？

1、发送端提高并发及批量大小；

2、消费端增加实例且同步宽容分区；

二、如何处理消息积压？

1、消费端扩容；

2、服务降级；

3、异常监控。

2019-08-08

Jxin 2

1.无法提升消费业务效率（仅受消费业务自身逻辑影响），但可以提高mq中堆积消息消费的整体吞吐量（批推比单推mq耗时较短）。

2.数据增量同步，监控信息采集。（非核心业务的稳定大数据流操作）。

3.批处理意味数据积累和大数据传输，这会让单次消费的最长时延变长。同时批量操作为了保证当前批量操作一致性，在个别失败的情况下会引发批量操作重试。

2019-08-06

作者回复

总结的非常好！

2019-08-07

linqw 2

尝试回答下课后习题，老师有空帮忙看下哦

消费端进行批量操作，感觉和上面的先将消息放在内存队列中，然后在并发消费消息，如果机器宕机，这些批量消息都会丢失，如果在数据库层面，批量操作在大事务，会导致锁的竞争，并且也会导致主备的不一致。如果是一些不重要的消息如对日志进行备份，就可以使用批量操作之类的提高消费性能，因为一些日志消息丢失也是可以接受的。

2019-08-06

作者回复

非常好！

2019-08-07

13761642169 1

消息本来就有限流或者削峰填谷，这个也是使用消息队列的作用，老师您讲的不严谨。

2019-08-09

godtrue 0

1：请教几个小问题

1-1：目前那些消息队列产品实现了批量消费？

1-2：批量消费的实现原理是怎用的？类似文中的例子直接放入一个内存队列中，然后再开启多线程批量消费嘛？

2：课后思考

2-1：在消费端是否可以通过批量消费的方式来提升消费性能？

可以

2-2：在什么样场景下，适合使用这种方法？

不担心重复消费，网络环境比较稳定，消费者端机器比较稳定

2-3：这种方法有什么局限性？

首先，使用的消息队列产品支持批量消费

然后，一次取多条消息，性能节省在连接建立上，消息量大在传输时会多耗一点，且消息者只能消费完一批消息，才能返回确认，会耽误点时间取下一批消息

最后，如果一批消息的消费过程中有一个失败了，假如一次消费1000条消息，消费第1000条时失败了，则整批消息都算失败了，那会重试，则前面999条消息都会重复消费，如果都有幂等控制也没什么，不过这种情况会使单台消息消费成功的时间加长许多。

3：本节小结

3-1：消息积压怎么产生的？

要么生产者太快了，要么消费者太慢了，这是相对的，但无论如何根本就是消费者的消费速度＜生产者的生产速度。

3-2：消息积压咋弄？

3-2-1：就让他积压，不管

3-2-2：有时为了缓解消费者的压力，还会故意让他积压

3-2-3：优化消费者逻辑，比如：将慢逻辑移出去，再将消息倒一道手

3-2-4：扩容，消费者扩容，broker也扩容，保持同一个主题下的分区和消费者实例一一对应，充分发挥消费者的性能

3-2-5：某些情况可以使消费者侧批量消费消息的方式，来提高性能

2019-08-21

作者回复

第一个问题，主流的消息队列都有实现，区别是，Kafka它是自动异步批量发送，而其它消息队列会提供一个类似batchSend(Msg [] messages)的批量发送的方法，由使用者决定是否批量发

送以及批量大小。

第二个问题我们后面的课程会有专门的讲解。

2019-08-21

一叶知秋 0

rabbitmq和rocketmq有什么常用的监控工具吗？

2019-08-20

小蚂蚁 0

老师，不好意思，我不明白为什么在每个分区上实际上只能支持单线程消费。

2019-08-19

作者回复

关于基础概念这块儿，你可以再看一下08答疑，里面会有更详细的解释。

2019-08-20

快快 0

老师，kafka扩容时会发出reblance吧

2019-08-19

作者回复

不仅是扩容时候，只要是consumer和partition有一方的数量变化，都会触发reblance。

2019-08-19

Panda 0

Consumer 的水平扩展能力也很重要

2019-08-16

lecy_L 0

消息积压处理：

1、发送端优化，增加批量和线程并发两种方式处理

2、消费端优化，优化业务逻辑代码、水平扩容增加并发并同步扩容分区数量

查看消息积压的方法：

1、消息队列内置监控，查看发送端发送消息与消费端消费消息的速度变化

2、查看日志是否有大量的消费错误

3、打印堆栈信息，查看消费线程卡点信息

2019-08-16

作者回复

2019-08-18

sky 0

还遇到一种消息积压的情况，mq那边hash算法有问题，大量消息被分配到某一个队列上了

2019-08-12

小涛

0

讲的太好了！

2019-08-12

Stenvien

0

如果消费者消费异常，即使多次消费也无法成功处理（如消息格式异常），导致一直无法成功ack此条消息，这种场景一般要怎么处理？

我想到有2种：
1. 不做任何处理，消费者会一直卡在此消息的处理上，那么后面的所有消息都没机会处理了，只能靠监控发现消费延迟，发出告警，人工修复。这种处理方式会导致一条有问题的消息就影响了整个业务
2. 数据库存储此异常的消息，并发告警，人工修复，仍然ack此消息，继续消费后面的消息。但是，若对消息的处理顺序有依赖，若没有成功处理此异常消息，消费的后面的消息的处理可能会有问题

是否有更好的处理方式？

2019-08-11

作者回复

有的消息中间件提供了“死信队列”的功能，它会自动把这种反复消费都失败的消息丢到一个特殊的死信队列中，避免一条消息卡主队列的情况。

2019-08-12

我瑟瑟的方法

0

在扩容 Consumer 的实例数量的同时，必须同步扩容主题中的分区（也叫队列）数量，确保 Consumer 的实例数和分区数量是相等的。

——////——

老师这一步怎么保证啊，需要业务consumer团队联系消息中间件团队一起运维配合吗？

2019-08-11

作者回复

是的。

2019-08-12

我瑟瑟的方法

0

在扩容 Consumer 的实例数量的同时，必须同步扩容主题中的分区（也叫队列）数量，确保 Consumer 的实例数和分区数量是相等的。

————-

老师。这一步该怎么保证？需要业务团队消费方

2019-08-11

DeathKnightH

0

批量消费的问题，文中指出使用内存队列会导致丢消息，再参考之前关于消息丢失和消息重复的两节课课，我有一个初步的想法，还是把消息放在broker中，消费者端实现幂等的消费操作，在每次消费完成后返回给broker确认消息。
用请求确认机制保证消息的可靠性，同时使消费操作幂等来保证不重复消费

2019-08-09