01 | 为什么需要消息队列?

李玥 · 消息队列高手课

你好，我是李玥。今天我们来讲讲为什么需要消息队列，消息队列主要解决的是什么问题。

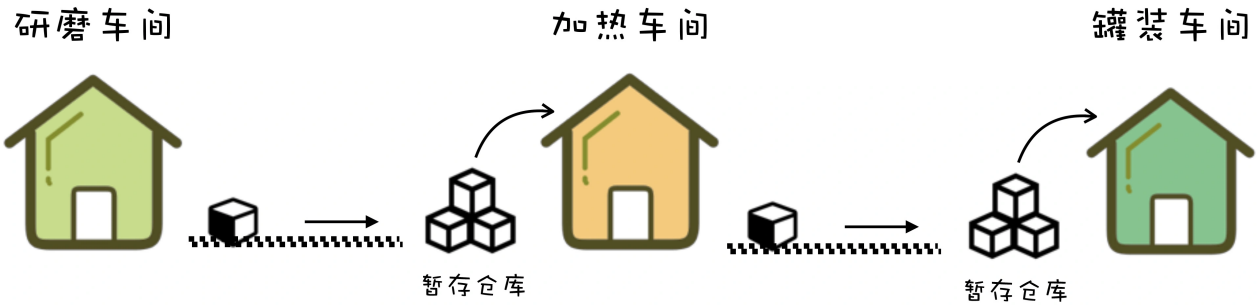
消息队列是最古老的中间件之一，从系统之间有通信需求开始，就自然产生了消息队列。但是给消息队列下一个准确的定义却不太容易。我们知道，消息队列的主要功能就是收发消息，但是它的作用不仅仅只是解决应用之间的通信问题这么简单。

我们举个例子说明一下消息队列的作用。话说小袁是一家巧克力作坊的老板，生产出美味的巧克力需要三道工序：首先将可可豆磨成可可粉，然后将可可粉加热并加入糖变成巧克力浆，最后将巧克力浆灌入模具，撒上坚果碎，冷却后就是成品巧克力了。

最开始的时候，每次研磨出一桶可可粉后，工人就会把这桶可可粉送到加工巧克力浆的工人手上，然后再回来加工下一桶可可粉。小袁很快就发现，其实工人可以不用自己运送半成品，于是他在每道工序之间都增加了一组传送带，研磨工人只要把研磨好的可可粉放到传送带上，就可以去加工下一桶可可粉了。 传送带解决了上下游工序之间的“通信”问题。

传送带上线后确实提高了生产效率，但也带来了新的问题：每道工序的生产速度并不相同。在巧克力浆车间，一桶可可粉传送过来时，工人可能正在加工上一批可可粉，没有时间接收。不同工序的工人们必须协调好什么时间往传送带上放置半成品，如果出现上下游工序加工速度不一致的情况，上下游工人之间必须互相等待，确保不会出现传送带上的半成品无人接收的情况。

为了解决这个问题，小袁在每组传送的下游带配备了一个暂存半成品的仓库，这样上游工人就不用等待下游工人有空，任何时间都可以把加工完成的半成品丢到传送带上，无法接收的货物被暂存在仓库中，下游工人可以随时来取。传送带配备的仓库实际上起到了“通信”过程中“缓存”的作用。



传送带解决了半成品运输问题，仓库可以暂存一些半成品，解决了上下游生产速度不一致的问题，小袁在不知不觉中实现了一个巧克力工厂版的消息队列。

哪些问题适合使用消息队列来解决?

接下来我们说一下日常开发中，哪些问题适合使用消息队列解决。

1. 异步处理

大多数程序员在面试中，应该都问过或被问过一个经典却没有标准答案的问题：如何设计一个秒杀系统？这个问题可以有一百个版本的合理答案，但大多数答案中都离不开消息队列。

秒杀系统需要解决的核心问题是，如何利用有限的服务器资源，尽可能多地处理短时间内的海量请求。我们知道，处理一个秒杀请求包含了很多步骤，例如：

风险控制；

库存锁定；

生成订单；

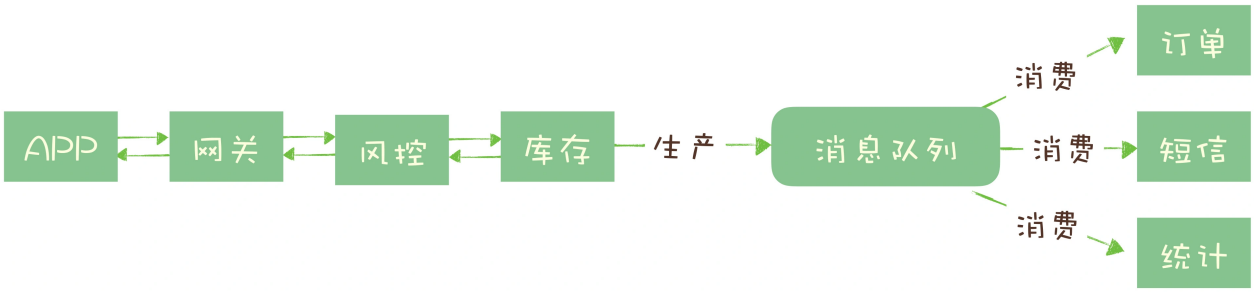
短信通知；

更新统计数据。

如果没有任何优化，正常的处理流程是：App 将请求发送给网关，依次调用上述 5 个流程，然后将结果返回给 APP。

对于这 5 个步骤来说，能否决定秒杀成功，实际上只有风险控制和库存锁定这 2 个步骤。只要用户的秒杀请求通过风险控制，并在服务端完成库存锁定，就可以给用户返回秒杀结果了，对于后续的生成订单、短信通知和更新统计数据等步骤，并不一定要在秒杀请求中处理完成。

所以当服务端完成前面 2 个步骤，确定本次请求的秒杀结果后，就可以马上给用户返回响应，然后把请求的数据放入消息队列中，由消息队列异步地进行后续的操作。



处理一个秒杀请求，从 5 个步骤减少为 2 个步骤，这样不仅响应速度更快，并且在秒杀期间，我们可以把大量的服务器资源用来处理秒杀请求。秒杀结束后再把资源用于处理后面的步骤，充分利用有限的服务器资源处理更多的秒杀请求。

可以看到，在这个场景中，消息队列被用于实现服务的异步处理。这样做的好处是：

可以更快地返回结果；

减少等待，自然实现了步骤之间的并发，提升系统总体的性能。

2. 流量控制

继续说我们的秒杀系统，我们已经使用消息队列实现了部分工作的异步处理，但我们还面临一个问题：如何避免过多的请求压垮我们的秒杀系统?

一个设计健壮的程序有自我保护的能力，也就是说，它应该可以在海量的请求下，还能在自身能力范围内尽可能多地处理请求，拒绝处理不了的请求并且保证自身运行正常。不幸的是，现实中很多程序并没有那么“健壮”，而直接拒绝请求返回错误对于用户来说也是不怎么好的体验。

因此，我们需要设计一套足够健壮的架构来将后端的服务保护起来。**我们的设计思路是，使用消息队列隔离网关和后端服务，以达到流量控制和保护后端服务的目的。**

加入消息队列后，整个秒杀流程变为：

1. 网关在收到请求后，将请求放入请求消息队列；
2. 后端服务从请求消息队列中获取 APP 请求，完成后续秒杀处理过程，然后返回结果。

秒杀开始后，当短时间内大量的秒杀请求到达网关时，不会直接冲击到后端的秒杀服务，而是先堆积在消息队列中，后端服务按照自己的最大处理能力，从消息队列中消费请求进行处理。

对于超时的请求可以直接丢弃，APP 将超时无响应的请求处理为秒杀失败即可。运维人员还可以随时增加秒杀服务的实例数量进行水平扩容，而不用对系统的其他部分做任何更改。

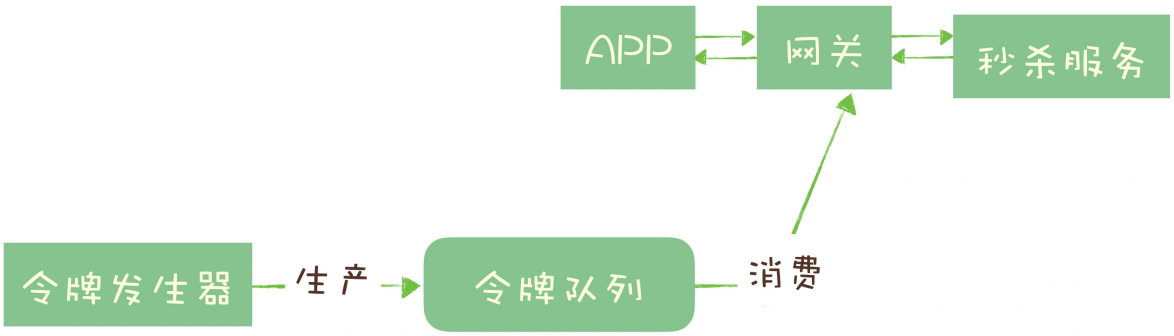
这种设计的优点是：能根据下游的处理能力自动调节流量，达到“削峰填谷”的作用。但这样做同样是有代价的：

增加了系统调用链环节，导致总体的响应时延变长。

上下游系统都要将同步调用改为异步消息，增加了系统的复杂度。

那还有没有更简单一点儿的流量控制方法呢？如果我们能预估出秒杀服务的处理能力，就可以用消息队列实现一个令牌桶，更简单地进行流量控制。

令牌桶控制流量的原理是：单位时间内只发放固定数量的令牌到令牌桶中，规定服务在处理请求之前必须先从令牌桶中拿出一个令牌，如果令牌桶中没有令牌，则拒绝请求。这样就保证单位时间内，能处理的请求不超过发放令牌的数量，起到了流量控制的作用。



实现的方式也很简单，不需要破坏原有的调用链，只要网关在处理 APP 请求时增加一个获取令牌的逻辑。

令牌桶可以简单地用一个有固定容量的消息队列加一个“令牌发生器”来实现：令牌发生器按照预估的处理能力，匀速生产令牌并放入令牌队列（如果队列满了则丢弃令牌），网关在收到请求时去令牌队列消费一个令牌，获取到令牌则继续调用后端秒杀服务，如果获取不到令牌则直接返回秒杀失败。

以上是常用的使用消息队列两种进行流量控制的设计方法，你可以根据各自的优缺点和不同的适用场景进行合理选择。

3. 服务解耦

消息队列的另外一个作用，就是实现系统应用之间的解耦。再举一个电商的例子来说明解耦的作用和必要性。

我们知道订单是电商系统中比较核心的数据，当一个新订单创建时：

1. 支付系统需要发起支付流程；
2. 风控系统需要审核订单的合法性；
3. 客服系统需要给用户发短信告知用户；
4. 经营分析系统需要更新统计数据；
5. ……

这些订单下游的系统都需要实时获得订单数据。随着业务不断发展，这些订单下游系统不断的增加，不断变化，并且每个系统可能只需要订单数据的一个子集，负责订单服务的开发团队不得不花费很大的精力，应对不断增加变化的下游系统，不停地修改调试订单系统与这些下游系统的接口。任何一个下游系统接口变更，都需要订单模块重新进行一次上线，对于一个电商的核心服务来说，这几乎是不可接受的。

所有的电商都选择用消息队列来解决类似的系统耦合过于紧密的问题。引入消息队列后，订单服务在订单变化时发送一条消息到消息队列的一个主题 Order 中，所有下游系统都订阅主题 Order，这样每个下游系统都可以获得一份实时完整的订单数据。

无论增加、减少下游系统或是下游系统需求如何变化，订单服务都无需做任何更改，实现了订单服务与下游服务的解耦。

小结

以上就是消息队列最常被使用的三种场景：异步处理、流量控制和服务解耦。当然，消息队列的适用范围不仅仅局限于这些场景，还有包括：

作为发布 / 订阅系统实现一个微服务级系统间的观察者模式；

连接流计算任务和数据；

用于将消息广播给大量接收者。

简单的说，我们在单体应用里面需要用队列解决的问题，在分布式系统中大多都可以用消息队列来解决。

同时我们也要认识到，消息队列也有它自身的一些问题和局限性，包括：

引入消息队列带来的延迟问题；

增加了系统的复杂度；

可能产生数据不一致的问题。

所以我们说没有最好的架构，只有最适合的架构，根据目标业务的特点和自身条件选择合适的架构，才是体现一个架构师功力的地方。

思考题

在你工作或学习涉及到的系统中，哪些问题可以通过引入消息队列来解决？对于系统中已经使用消息队列，可以对应到这一讲中提到的哪个场景？如果没有可以对应的场景，那这个消息队列在系统中起到的是什么作用？解决了什么问题？是否又带来了什么新的问题？欢迎在留言区写下你的想法。

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 (180)

小伟

2019-07-24

个人的体会，消息队列的本质是将同步处理转成异步处理，异步会带来相应的好处，但也有弊端。
Pros:
1.可在模块、服务、接口等不同粒度上实现解耦
2.订阅/消费模式也可在数据粒度上解耦
3.可提高系统的并发能力，集中力量办大事(同步部分)，碎片时间做小事(异步部分)
4.可提高系统可用性，因为缓冲了系统负载

Cons:
1.降低了数据一致性，如要保持强一致性，需要高代价的补偿(如分布式事务、对账)
2.有数据丢失风险，如宕机重启，如要保证队列数据可用，需要额外机制保证(如双活容灾)

总体来说，消息队列的适用场景还是很多的，如秒杀、发邮件、发短信、高并发订单等，不适合的场景如银行转账、电信开户、第三方支付等。关键还是要意识到消息队列的优劣点，然后分析场景是否适用则会水到渠成。

作者回复: 总结到位，赞👍 。

共 19 条评论> 208

beiler
2019-07-25

还有个问题，如果消息量特别大的时候，消息是适合存在到redis中还是适合存到rabbitmq中？必定您在文中提到一个词，小仓库，如果货量大了怎么办？

作者回复: 首先redis肯定是不适合存消息的，虽然redis性能很好，但那是和主流的数据库比，一般大概能到几万tps左右；而现代的消息队列都能很轻松的做到几十万tps级别的性能。

消息量特别大的时候，需要考虑使用有消息堆积能力的MQ，因为一旦消费慢，大量消息就会堆积到MQ中，这种情况不太适合用RabbitMQ，可以考虑RocketMQ、Kafka和Pulsar。

共 8 条评论> 88

游弋云端
2019-07-24

是否可以利用共享内存、RDMA加速消息队列的性能，老师在这块有没有实践经验？

作者回复: 如果你说的共享内存指的是PageCache，很多消息队列都会用到，RDMA据我所知常见的几种消息队列应该都还没有使用，像Kafka它在消费的时候，直接使用Zero Copy，数据直接从Page Cache写到NIC的缓冲区中，都不需要进入应用内存空间。

另外，现代的消息队列瓶颈并不在本机内存数据交换这块，主要还是受限于网卡带宽或者磁盘的IO，像JMQ、Kafka这些消息队列，都可以打满万兆网卡或者把磁盘的读写速度拉满。

共 8 条评论>

58

beiler

2019-07-25

令牌桶给了我很大的启发，我们可以在策略中心设置令牌桶，然后通过令牌桶控制整个job的产出和数量。这样就不会经常有几百万个job了，缓存的压力也会大幅度减小。但是有一个很诡异的问题，就以秒杀系统为例（我们的系统要比秒杀复杂点），我发现这种异步系统如果需要统计任务数量的时候经常会计数不准，尽管在计数的时候我选择了原子操作，但是计数还是会出现不准的现象。这个让我很苦恼，而且往往是运行很久的任务会出现不准，往往只有在任务结束的时候发现任务不准，这个问题很难查，请问老师有什么好建议吗

作者回复: 如果计数只是为了控制流量，没必要那么精确。

如果计数是业务需求必须要求准确，简单一点的话，可以使用Redis的INCR命令来计数，这个是可以保证原子性的。Redis性能要是不能满足要求，也可以用Kafka+flink集群来解决。这两种方案都是可以保证完全准确计数的。

另外，计数不准的问题，并不一定是计数模块本身的问题，还要查一下是不是系统的其它部分有bug，导致重复计数或者漏计。

共 4 条评论>

50

撒旦的堕落

2019-07-31

我懵的地方就是用队列 将同步改成了异步 那么原来同步的request 和response是一对 那么改成异步后 怎么通知用户 难道还用原来的那个response？ 还是当秒杀成功后 根据用户的id 查询到信息 比如手机号码 然后发短信给他 或者是向用户推消息什么的

作者回复: 对于网关某一个处理前端请求线程来说，大致的流程是：

0.收到Request

1.发消息

2.阻塞等待，直到超时或者收到后端的秒杀结果；

3.返回Response

共 15 条评论>

linqw

2019-07-30

APP⇆网关--生产-->消息队列--消费-->秒杀服务，有几点疑惑，老师有空帮忙解答下哦

1、海量的请求都放在消息队列中，消息队列的整体容量如何衡量了？消息队列不可能能存放无限的消息，消息队列满应该也会有拒绝策略，比如线程池的任务队列，任务队列满，并且超过最大的线程池数，四种的拒绝策略。

2、APP响应超时，即网关超过一定的时间没有返回，消息还在任务队列中，还是会被秒杀服务处理，这样的话，返回给APP秒杀失败，但是秒杀服务已经消费了消息？难道是在网关做补偿么？如果连接已经断开，将秒杀服务对此消息的处理做回滚操作么？

3、网关和秒杀服务是通过消息队列进行通信，那响应消息也通过队列进行返回么？队列中会有APP对应的地址比如IP之类的？那这样的话，APP的海量连接都同时连接着网关，不是会有问题么？

4、消息队列应该也会做多备的策略？比如队列消息的服务挂了，那些消息全部不见，这样不是也会存在问题么？

作者回复: A1：实际上，只要有足够的磁盘容量，消息队列确实可以存放无限的消息。像秒杀请求这种数据，峰值并发高，但总数据量并不是很大，所以，堆积在消息队列中完全没问题。

A2：都按照秒杀失败处理即可。

A3：响应一般采用RPC来实现。超时或者返回秒杀结果之前，网关和APP确实要保持连接，这是HTTP协议决定的。至于网关能不能承受海量的APP连接，这个应该不用担心，网关的作用就是用来抗海量连接的，它也会有各种方法来解决这个问题。

4、是的，大部分生产系统中的消息队列要配置成集群，确保可用性和数据可靠性，这个后面的课程我们会讲。

共 14 条评论>

白小白
2019-07-23

现在用的消息队列主要是做数据异步传输用的，之前也做过多个系统之间的解耦。看到用消息队列做秒杀系统，忽然想到之前只想过用redis去做，利用redis去做了流量的把控。不过细想想，这种情况下的redis和文章中的令牌桶很像……

作者回复: 是的，令牌桶可以用消息队列实现，也可以用Redis实现，你也可以写一个简单的令牌桶服务，原理是一样的。

共 3 条评论>

30

小趴菜～
2019-07-23

生产项目中用到了kafka，
1 异部的处理交易：提高用户请求的响应速度，同时也提升了用户的体验感。
2 削峰 ： 保护服务器的一种方式，用户的请求放到kafka中，交易服务根据自己服务器的消费能力来消费交易数据。
3 项目的解耦：交易服务和后续的服务之间是通过Kafka进行交付，当一个服务为多个服务提供数据的时候，可以通过MQ进行交换来解耦服务间的耦合。

作者回复: 总结的很赞！

24

落尘kira
2019-07-30

看了下评论，我就简单补充一下实际用过的场景：
1.数据同步：包括业务服务之间的业务数据同步（主要是状态）、DB间的数据同步等等
2.异步通知：包括发送IM消息、异步日志、异步短信/邮件（尤其是批量数据）或注册/开启任务等等
3.信息收集：主要用于数据统计、监控、搜索引擎等等
4.服务解耦：主要用于重构和新设计时，对频繁变动的接口服务进行解耦（通常是被需求给逼的）
5.分布式事务消息：尤其是对数据一致性有要求的异步处理场景
6.主动性防御：秒杀、限流

作者回复: 总结的非常到位！

共 4 条评论>

23

ly

2019-07-25

老师，关于第二点的流控有点疑问：网关将request信息放入mq中，然后后端服务去mq中消费这个请求，我通常晓得的mq储存文本消息，那这样的场景下，后端处理完秒杀以后，是如何得到response响应客户端的请求呢?

作者回复: 这个取决于网关是如何实现的。大致的思路是，网关会把用户的request缓存起来，然后发消息，至于发的消息内容不一定就是这个原封不动的request对象，只要把Request中必要的信息发给后端就可以了。

后端服务可以用RPC通知网关秒杀结果，网关收到结果后找到对应的Request来构建Response返回即可。

共 9 条评论>

22

风中花

2019-07-23

要不要继续买，继续买要不要！老师讲得这么好！纠结

作者回复: 你买不了吃亏，买不了上当，买到的只有知识。

共 3 条评论>

18

大白先生

2019-07-30

那秒杀时后端请求没处理完，app返回超时后，后续服务处理之前请求时会不会进行库存扣减，还是说，后端能识别出哪些请求超时，不进行处理

作者回复: 这个就是比较细节的问题了，实现的方式也可以有多种，比如：在消息中带一个请求时间戳，后续服务在处理前先检查一下是否已经超时，超时就直接丢掉不处理。

共 2 条评论>

15

微微一笑

2019-07-23

看到消息队列的专栏很兴奋，能学到底层源码、设计思想一直是我的梦，哈哈哈。目前在一家互金公司负责一个资金平台的项目，负责对接车贷、消费金融两个系统，同时与第三方资金渠道进行对接。在于车贷、消费金融这俩系统对接中，使用了rocketMQ进行系统间的解耦，系统间升级优化上线互不影响。由于对接的第三方渠道越来越多渠道间耦合较严重，下一步准备进行系统拆分，系统与系统间经过消息队列进行解耦。

作者回复: 涉及到钱的系统，数据可靠性是最先需要考虑的问题。

共 6 条评论>

13

x.l
2019-09-21

老师，你好！工作中有按业务优先处理的需求，想实现个优先队列，问下老师有没有常规的解决方案？

作者回复: 一个主题设置多个分区，每个分区代表一个优先级。
发送的时候，根据优先级指定分区发送到对应的分区上。
消费的时候，按照优先级从高到低，指定分区消费。

共 2 条评论>

11

Jxin
2019-07-23

1.拆单失败的延时重拆，死信告警。2.消峰和解耦也用到。

问题：控制topic消费线程也能限流，不一定要引入令牌桶，要弄令牌桶，其实走redis更好一点。

作者回复: 限流的方法有很多，当然不止令牌桶。令牌桶的优势是实现简单，易于控制。

11

Geek_bbe9ea
2019-07-23

修改数据库做数据同步也可以用

作者回复: 是的，很多公司会用消息队列来做异构数据库之间的数据同步，但是一定要注意顺序问

题。像MySQL Binlog这种，是要求严格有序的，否则会出现问题。

共 6 条评论>

9

Geek_e7834d
2019-07-27

使用消息队列怎么保证实现节点时效时候能够切换到异地节点 然后还要保证不丢失消息呢或者尽量少丢失消息？异地冗余都有什么好方案呢？

作者回复: 异地容灾是个比较难解决的问题。

我的经验是：绝大部分主题是不需要异地容灾的，因为消息队列不会直接堆外提供服务，它直接服务都是机房内部的应用，当出现整个机房大面积断电或者机房外网中断的时候，消息的生产者本身已经不能提供服务了，这时候消息队列的容灾是没有意义的。如果生产者它本身支持异地容灾能自动把服务迁移到其它机房，那这个应用在其它机房的实例使用本机房内的消息队列就行了，也不需要消息队列做异地复制和容灾。

但是，确实有极少数应用比较特殊，它是有异地容灾的需求的，我们目前的方案是多副本分布在多个机房中，配合就近消费来实现。

8

TANMIYOO
2019-07-23

收获总结：
1. 消息队列可以理解为一个暂存消息（可以是一条数据或者一个请求等等）的地方，有生产者有消费者
2.消息队列的主要三个用处：
a. 实现异步处理，利用消息队列可以将串行化的功能，在非必要串行的地方实现并行化，从而提升系统性能，缩短响应时间
b. 实现流量控制 在高并发的情况下，为了避免大量的请求冲击后台服务，可以使用消息队列暂存请求，后台服务以最大处理能力消费请求，保证后台的安全性，其缺点拉长系统调用链，响应时间变长，增加系统复杂度；另外一种不改变系统调用链的实现方式，引入令牌桶的概念，单位时间内生成一定量的令牌放到令牌桶（即消息队列）中，令牌的数量要依据后台系统的处理能力，网关接受到请求后取到令牌才能调用后台服务，取不到则请求失败
c. 系统间解耦 多个下游系统会频繁调用上游系统的接口获取数据的情况下，若上游系统将消息放到指定queue中，多个下游系统订阅消息，就可以避免上游为对接多个下游时频繁地修改接口，降低系统间的耦合度

思考题：
目前erp项目中，订单数据需要同时发给工程去评估以及企划去进行物料核算，现在的实现方式则是系统之间通过接口进行拉去或者推送，这就可以使用消息队列，将订单放到消息队列中，供下游订阅使用，降低系统间解耦

作者回复: 总结的很到位，加油！

共 4 条评论>

8

Fortune
2019-07-23

看完了，也看完了评论，可能只有我一个没有实际项目中接触消息队列了，慢慢学吧，加油！目前做的是支付系统，只知道用redis用来存储用户token和进行验证这样子，当然中间用户请求过来的过程中，是可以加队列来进行削峰的，应该是系统的并发并不高哈，就做了个集群这样子，谢谢老师分享！

共 1 条评论>

8

钱
2019-08-19

老师的比喻很棒，通俗易懂。

1：消息队列的核心作用?
1-1：服务解藕
1-2：削峰填谷
1-3：异步处理

2：消息队列带来的问题?
2-1：增加系统的复杂度
2-2：延迟了请求的响应时间
2-3：可能会丢消息、可能会重复消费消息、可能导致数据不一致、消息可能是无序的

3：消息队列的适用场景?
3-1：秒杀
3-2：发送邮件
3-3：日志收集
3-4：离线数据持久化
3-5：其他不担心数据不是强一致性的大流量高并发场景

4：消息队列不适合的场景?
4–1：要求实时响应
4–2：要求数据强一致性
4–3：不能容忍消息丢失
4–5：直接和钱相关的系统，比如：支付系统

MQ能干啥？啥时候适合使用？这些还是比较容易理解的，痛点在于自己设计并实现一个MQ能否扛住亿级流量，并且具有各种其他优良特性。

李老师，请教几个：
1：消息的全生命轨迹，会描述嘛?
2：producer发送消息是通过TCP连接的，broker什么时候响应客户端?
3：使用JMQ几乎天天报超时异常？这个超时异常有哪些场景会发生？为什么总也解决不了?
4：consumer拉取消息的时机是什么时候?
5：broker怎么实现高性能的消息落盘的?

作者回复: A1：我们会在几节课里面分阶段讲解；
A2：一般都是采用长连接机制，收到请求后响应；
A3：首先需要检查一下本机的是不是负载太高，是不是有fullGC，这些都会拖慢程序整体运行。如果是发消息超时，需要联系运维人员，如果是收消息超时，优先看一下消费的业务逻辑是不是执行的太慢了。
A4：一般如果一直能拉到消息，就会一直不停的拉取，如果没有消息，会定时拉取，直到拉到新的消息。
A5：关于“broker怎么实现高性能的消息落盘的”这个问题，我在进阶篇中会讲到，请关注。