04 | 流量削峰这事应该怎么做？

2018-10-04 许令波

如何设计一个秒杀系统 进入课程 >

讲述：秭明
时长 12:34 大小 5.76M

如果你看过秒杀系统的流量监控图的话，你会发现它是一条直线，就在秒杀开始那一秒是一条很直很直的线，这是因为秒杀请求在时间上高度集中于某一特定的时间点。这样一来，就会导致一个特别高的流量峰值，它对资源的消耗是瞬时的。

但是对秒杀这个场景来说，最终能够抢到商品的人数是固定的，也就是说 100 人和 10000 人发起请求的结果都是一样的，并发度越高，无效请求也越多。

但是从业务上来说，秒杀活动是希望更多的人来参与的，也就是开始之前希望有更多的人来刷页面，但是真正开始下单时，秒杀请求并不是越多越好。因此我们可以设计一些规则，让并发的请求更多地延缓，而且我们甚至可以过滤掉一些无效请求。

为什么要削峰

为什么要削峰呢？或者说峰值会带来哪些坏处？

我们知道服务器的处理资源是恒定的，你用或者不用它的处理能力都是一样的，所以出现峰值的话，很容易导致忙到处理不过来，闲的时候却又没有什么要处理。但是由于要保证服务质量，我们的很多处理资源只能按照忙的时候来预估，而这会导致资源的一个浪费。

这就好比因为存在早高峰和晚高峰的问题，所以有了错峰限行的解决方案。削峰的存在，一是可以让服务端处理变得更加平稳，二是可以节省服务器的资源成本。针对秒杀这一场景，削峰从本质上来说就是更多地延缓用户请求的发出，以便减少和过滤掉一些无效请求，它遵从“请求数要尽量少”的原则。

今天，我就来介绍一下流量削峰的一些操作思路：排队、答题、分层过滤。这几种方式都是无损（即不会损失用户的发出请求）的实现方案，当然还有些有损的实现方案，包括我们后面要介绍的关于稳定性的一些办法，比如限流和机器负载保护等一些强制措施也能达到削峰保护的目的，当然这都是不得已的一些措施，因此就不归类到这里了。

排队

要对流量进行削峰，最容易想到的解决方案就是用消息队列来缓冲瞬时流量，把同步的直接调用转换成异步的间接推送，中间通过一个队列在一端承接瞬时的流量洪峰，在另一端平滑地将消息推送出去。在这里，消息队列就像“水库”一样， 拦蓄上游的洪水，削减进入下游河道的洪峰流量，从而达到减免洪水灾害的目的。

用消息队列来缓冲瞬时流量的方案，如下图所示：

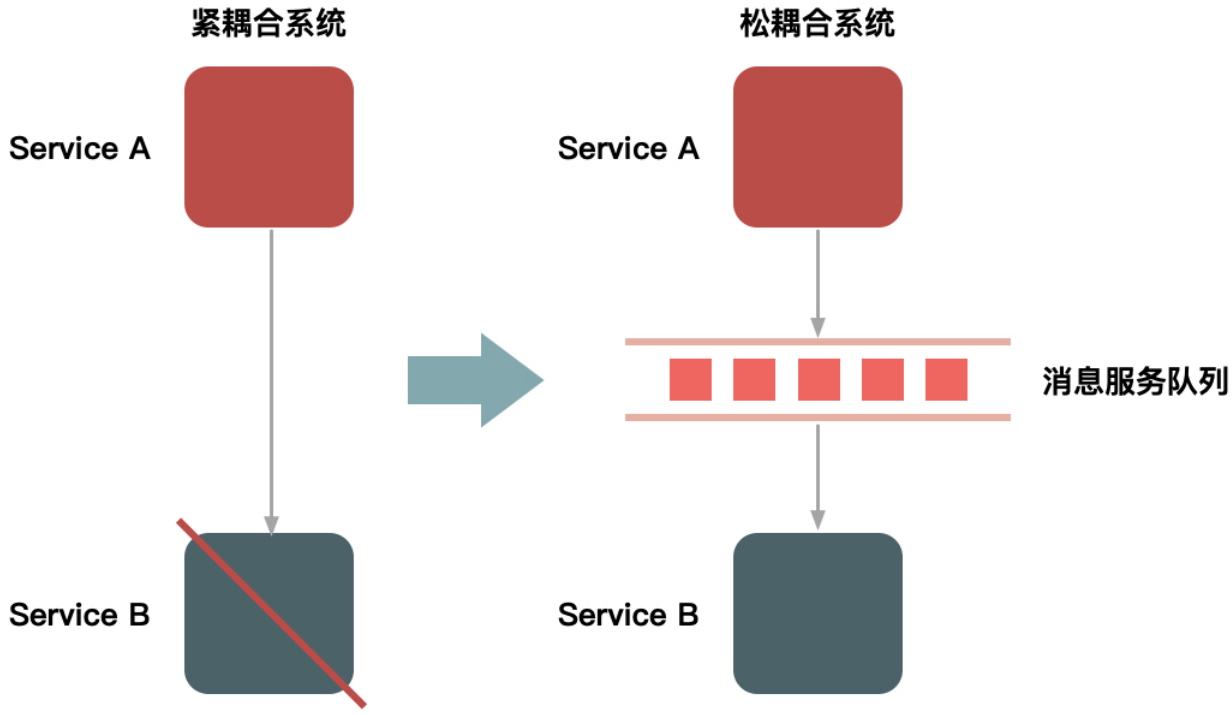

图 1 用消息队列来缓冲瞬时流量

但是，如果流量峰值持续一段时间达到了消息队列的处理上限，例如本机的消息积压达到了存储空间的上限，消息队列同样也会被压垮，这样虽然保护了下游的系统，但是和直接把请求丢弃也没多大的区别。就像遇到洪水爆发时，即使是有水库恐怕也无济于事。

除了消息队列，类似的排队方式还有很多，例如：

1. 利用线程池加锁等待也是一种常用的排队方式；
2. 先进先出、先进后出等常用的内存排队算法的实现方式；
3. 把请求序列化到文件中，然后再顺序地读文件（例如基于 MySQL binlog 的同步机制）来恢复请求等方式。

可以看到，这些方式都有一个共同特征，就是把“一步的操作”变成“两步的操作”，其中增加的一步操作用来起到缓冲的作用。

说到这里你可能会说，这样一来增加了访问请求的路径啊，并不符合我们介绍的“4 要 1 不要”原则。没错，的确看起来不太合理，但是如果不增加一个缓冲步骤，那么在一些场景下系统很可能会直接崩溃，所以最终还是需要你做出妥协和平衡。

答题

你是否还记得，最早期的秒杀只是纯粹地刷新页面和点击购买按钮，它是后来才增加了答题功能的。那么，为什么要增加答题功能呢？

这主要是为了增加购买的复杂度，从而达到两个目的。

第一个目的是防止部分买家使用秒杀器在参加秒杀时作弊。2011 年秒杀非常火的时候，秒杀器也比较猖獗，因而没有达到全民参与和营销的目的，所以系统增加了答题来限制秒杀器。增加答题后，下单的时间基本控制在 2s 后，秒杀器的下单比例也大大下降。答题页面如下图所示。

图 2 答题页面

第二个目的其实就是延缓请求，起到对请求流量进行削峰的作用，从而让系统能够更好地支持瞬时的流量高峰。这个重要的功能就是把峰值的下单请求拉长，从以前的 1s 之内延长到 2s~10s。这样一来，请求峰值基于时间分片了。这个时间的分片对服务端处理并发非常重要，会大大减轻压力。而且，由于请求具有先后顺序，靠后的请求到来时自然也就没有库存了，因此根本到不了最后的下单步骤，所以真正的并发写就非常有限了。这种设计思路目前用得非常普遍，如当年支付宝的“咻一咻”、微信的“摇一摇”都是类似的方式。

这里，我重点说一下秒杀答题的设计思路。

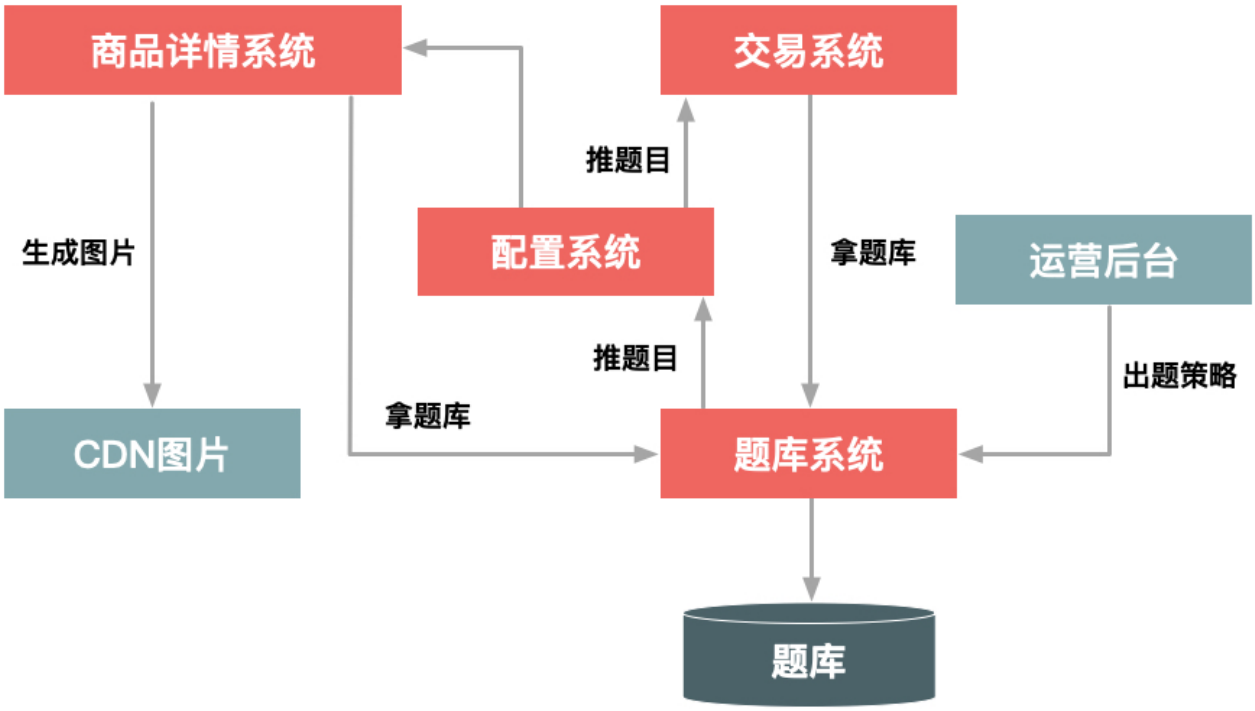

图 3 秒杀答题

如上图所示，整个秒杀答题的逻辑主要分为 3 部分。

1. **题库生成模块**，这个部分主要就是生成一个个问题和答案，其实题目和答案本身并不需要很复杂，重要的是能够防止由机器来算出结果，即防止秒杀器来答题。
2. **题库的推送模块**，用于在秒杀答题前，把题目提前推送给详情系统和交易系统。题库的推送主要是为了保证每次用户请求的题目是唯一的，目的也是防止答题作弊。
3. **题目的图片生成模块**，用于把题目生成为图片格式，并且在图片里增加一些干扰因素。这也同样是为防止机器直接来答题，它要求只有人才能理解题目本身的含义。这里还要注意一点，由于答题时网络比较拥挤，我们应该把题目的图片提前推送到 CDN 上并且要进行预热，不然的话当用户真正请求题目时，图片可能加载比较慢，从而影响答题的体验。

其实真正答题的逻辑比较简单，很好理解：当用户提交的答案和题目对应的答案做比较，如果通过了就继续进行下一步的下单逻辑，否则就失败。我们可以把问题和答案用下面这样的 key 来进行 MD5 加密：

问题 key：userId+itemId+question_Id+time+PK

答案 key：userId+itemId+answer+PK

验证的逻辑如下图所示：

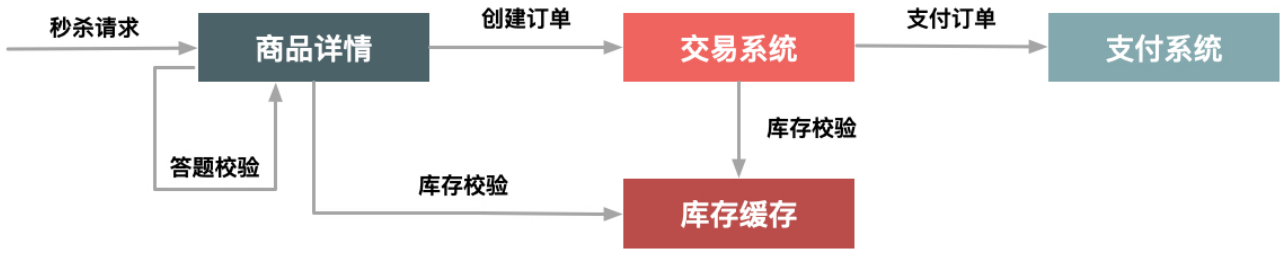

图 4 答题的验证逻辑

注意，这里面的验证逻辑，除了验证问题的答案以外，还包括用户本身身份的验证，例如是否已经登录、用户的 Cookie 是否完整、用户是否重复频繁提交等。

除了做正确性验证，我们还可以对提交答案的时间做些限制，例如从开始答题到接受答案要超过 1s，因为小于 1s 是人为操作的可能性很小，这样也能防止机器答题的情况。

分层过滤

前面介绍的排队和答题要么是少发请求，要么对发出来的请求进行缓冲，而针对秒杀场景还有一种方法，就是对请求进行分层过滤，从而过滤掉一些无效的请求。分层过滤其实就是采用“漏斗”式设计来处理请求的，如下图所示。

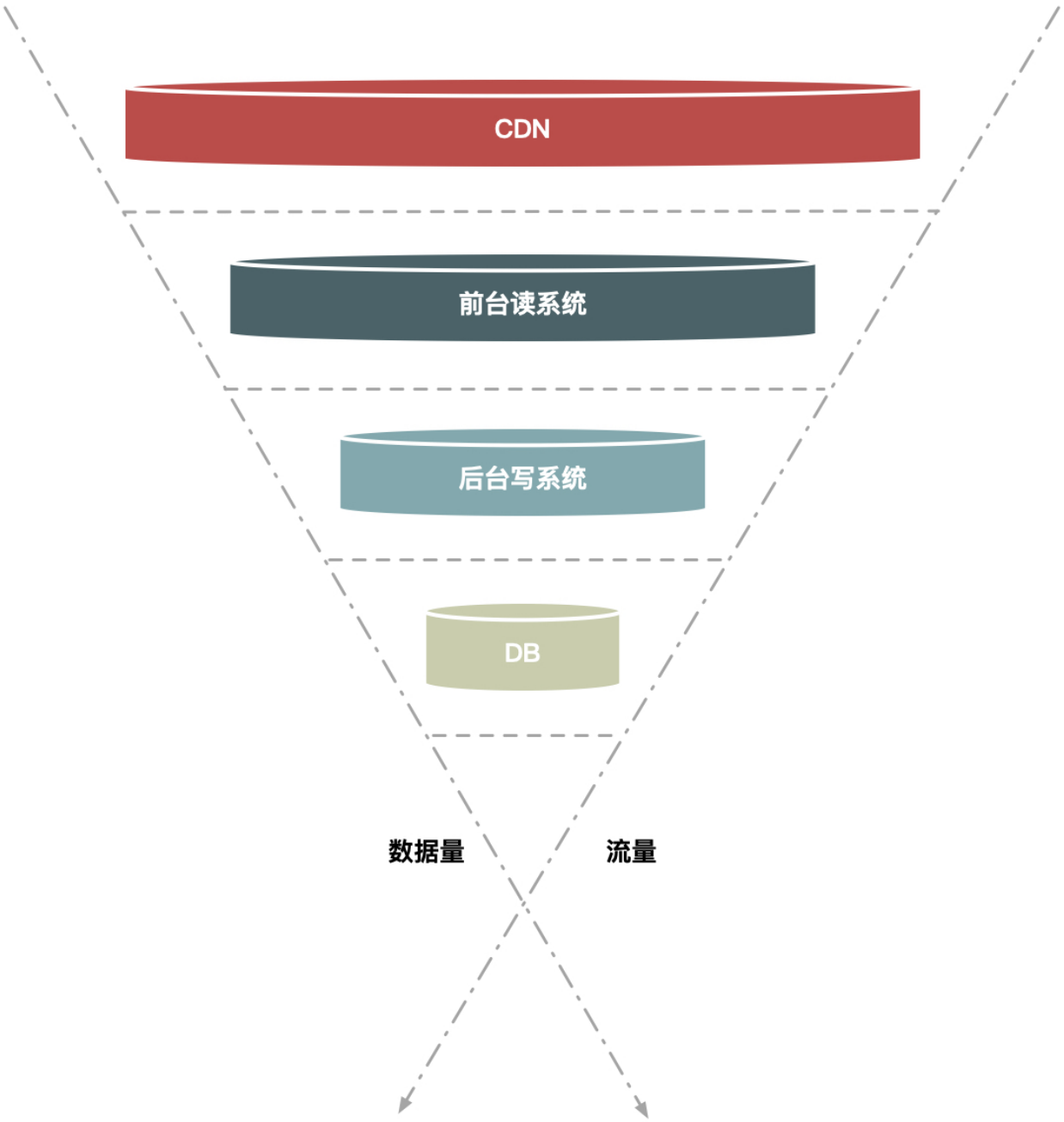

图 5 分层过滤

假如请求分别经过 CDN、前台读系统（如商品详情系统）、后台系统（如交易系统）和数据库这几层，那么：

大部分数据和流量在用户浏览器或者 CDN 上获取，这一层可以拦截大部分数据的读取；

经过第二层（即前台系统）时数据（包括强一致性的数据）尽量得走 Cache，过滤一些无效的请求；

再到第三层后台系统，主要做数据的二次检验，对系统做好保护和限流，这样数据量和请求就进一步减少；

最后在数据层完成数据的强一致性校验。

这样就像漏斗一样，尽量把数据量和请求量一层一层地过滤和减少了。

分层过滤的核心思想是：在不同的层次尽可能地过滤掉无效请求，让“漏斗”最末端的才是有效请求。而要达到这种效果，我们就必须对数据做分层的校验。

分层校验的基本原则是：

1. 将动态请求的读数据缓存（Cache）在 Web 端，过滤掉无效的数据读；
2. 对读数据不做强一致性校验，减少因为一致性校验产生瓶颈的问题；
3. 对写数据进行基于时间的合理分片，过滤掉过期的失效请求；
4. 对写请求做限流保护，将超出系统承载能力的请求过滤掉；
5. 对写数据进行强一致性校验，只保留最后有效的数据。

分层校验的目的是：在读系统中，尽量减少由于一致性校验带来的系统瓶颈，但是尽量将不影响性能的检查条件提前，如用户是否具有秒杀资格、商品状态是否正常、用户答题是否正确、秒杀是否已经结束、是否非法请求、营销等价物是否充足等；在写数据系统中，主要对写的数据（如“库存”）做一致性检查，最后在数据库层保证数据的最终准确性（如“库存”不能减为负数）。

总结一下

今天，我介绍了如何在网站面临大流量冲击时进行请求的削峰，并主要介绍了削峰的 3 种处理方式：一个是通过队列来缓冲请求，即控制请求的发出；一个是通过答题来延长请求发出的时间，在请求发出后承接请求时进行控制，最后再对不符合条件的请求进行过滤；最后一种是对请求进行分层过滤。

其中，队列缓冲方式更加通用，它适用于内部上下游系统之间调用请求不平缓的场景，由于内部系统的服务质量要求不能随意丢弃请求，所以使用消息队列能起到很好的削峰和缓冲作用。

而答题更适用于秒杀或者营销活动等应用场景，在请求发起端就控制发起请求的速度，因为越到后面无效请求也会越多，所以配合后面介绍的分层拦截的方式，可以更进一步减少无效请求对系统资源的消耗。

分层过滤非常适合交易性的写请求，比如减库存或者拼车这种场景，在读的时候需要知道还有没有库存或者是否还有剩余空座位。但是由于库存和座位又是不停变化的，所以读的数据是否一定要非常准确呢？其实不一定，你可以放一些请求过去，然后在真正减的时候再做强一致性保证，这样既过滤一些请求又解决了强一致性读的瓶颈。

不过，在削峰的处理方式上除了采用技术手段，其实还可以采用业务手段来达到一定效果，例如在零点开启大促的时候由于流量太大导致支付系统阻塞，这个时候可以采用发放优惠券、发起抽奖活动等方式，将一部分流量分散到其他地方，这样也能起到缓冲流量的作用。

最后，欢迎你在留言区和我交流，你也可以说说在实际工作中，还有哪些对流量进行削峰的不同思路或方案，非常期待。

上一篇 03 | 二八原则：有针对性地处理好系统的“热点数据”

下一篇

精选留言 (41)

写留言

胡镇华
2018-10-04

30

用消息队列实现的话，处理结果无法立即知晓，用户体验不真实，有没有更实时的方案？

作者回复: 就是实时处理，每个请求过来实时处理，先过来先处理

烛火下的乌...
2018-12-26

18

MD5说成加密又怎么样？本来MD5的存在不就是为了加密吗？至于原理大家明白就可以了，真的是。。。

wj
2018-10-06

17

纠正一下，作为P8把Md5理解为加密算法，太不应该了，这个仅仅是单向散列，不可逆..

作者回复: 呵呵，感谢指正

Lost In ...
2018-10-04

9

请求入队列后怎么给用户“交代”？

展开

作者回复: 参考一下nero的回答哈

Ballontt
2018-10-04

7

可不可以前端直接按照1%的概率去请求后台接口。请数量一下就下降了100倍

作者回复: 光从减少请求的角度可以，但是体验会很差😀

Nero
2018-10-08

6

请问徐老师，当请求被丢进消息队列以后，是就直接返回给用户吗？ 那用户怎么知道请求是否成功了呢？

作者回复: 如果是同步的就要等待消息被正确投递后才返回结果，但大部分就是异步的，寄发送后即返回，然后由消息队列保证最后最终被投递，这个要由消息队列自己来承诺sla

看不到de颜...
2018-10-15

5

看完本篇文章，有几点疑问还想请老师解答一下。
1.使用消息队列削峰的话，可以知道消息是否被消费，但是是否真的秒杀成功该如何向用户返回。
2.看到老师提到削峰的方式，之前有看过到诸如令牌桶，漏桶之类的算法。是否也可以在此引入呢？...
展开

作者回复: 大家对请求队列这块问的比较多，疑问也多，后面相关的问题我找个时间统一回答一下吧。

GrubbyLu
2018-10-14

3

徐老师看了其他同学的留言以及您的回复之后，还是有一点不能理解。就是用消息队列进行解耦之后，如何把消息队列处理秒杀请求的结果反馈给用户，有什么好的通知方式嘛？（看到有一个同学留言说用长链接异步推送结果，您说用户体验偏差，麻烦能介绍一些好的方式嘛）
展开

作者回复: 大家对请求队列这块问的比较多，后面相关的问题我找个时间统一、详细回答一下吧

大老杨

2018-10-04

3

这种答题的是不是对于秒杀场景用户体验不是很好

展开

作者回复: 答题是有两种效果
一是可以防止一些秒杀器
二是可以延长一部分答题时间
是不是影响体验，我觉得体验和上面两条相比应该要做些妥协

Geek_c991e...

2018-11-05

2

大神问下，如果秒杀库存总数是10，那削峰队列大小就是10吗，如果有后面不买了，但是已经入了队列了，怎么办。还是说队列放所有请求，这样的话是不是浪费啊

展开

作者回复: 入了队列不处理就超时了，队列的大小不应该和秒杀商品数关联

食指可爱多

2018-10-12

2

下单请求进消息队列，RocketMQ的topic个数或kafka的partition个数为消费端并发度，那么topic的个数怎么设置。看到一种说法是每个商品id对应一个topic。嗯，理论上完美了，每个商品一个队列，但是这些topic工程实践里又是如何管理的呢？

展开

作者回复: 你说的topic工程实践是指太多不好管理吗？

每个商品一个topic的确太多，而且topic太多下游也不好订阅，topic不应该太多，不应该通过人为分散topic来提升性能，这样会增加维护成本，增加的成本可能比省的几台硬件成本更高，所以应该优化MQ软件本身入手

Lee
2018-10-10

2

有两个小问题请教一下：
生产者将用户请求放入队列后，用户的请求就结束了，但是消费者还未处理，这时生产者给用户返回什么呢？
队列的消费者处理完用户的请求后，怎么返回结果给用户呢？

展开

作者回复: 参考下我给nero的回答哈

五年
2018-10-11

1

零点大促开始了...发放优惠券怎么操作呢....我这个时候已经在指定商品这里等着了....

作者回复: 这个地方发放优惠券这个是一个营销策略，主要是为了分散流量，例如在活动页面可以通过弹窗的方式，把一部分用户吸引到一个新的业务，让用户玩个游戏，通关了就发放一个优惠券。这个方式当然是吸引那种还没有明确下单目标的用户，如果你已经有了目标商品了，就等着时间一到来下单了，那么优惠券对你也没有吸引力，其实优惠券也不是要吸引所有的用户，那样也起不到分流的目的了

func
2018-10-08

1

许大神你好，请求入队列后怎么给用户 反馈结果？

展开

作者回复: 你可以了解一下rocketmq是怎么做的😃

小喵喵
2018-10-06

1

1.分层过滤，既然是请求，为什么有些是无效呢？
2.将动态请求的读数据缓存（Cache）在 Web 端，过滤无效的数据读。这个cache是动态的呀，是不是经常需要更新cache呢？能举例一下，到底什么样子的动态数据缓存（Cache）在 Web 端？？
3.限流保护具体怎么做？谢谢。

展开

作者回复: 1.无效请求是针对没发再抢到商品的人来说的
2.例如库存数据
3.限流保护回来最后一章有介绍

潘政宇
2018-10-04

1

队列被打满了，直接丢包吗？

展开

作者回复: 有多种处理方式，一种是丢弃
还有可以把队列序列化到文件，然后再慢慢消化

Ben
2018-10-04

1

第一第一

展开

作者回复: 😉

無忘
2019-05-19

很无奈，关键的下单通过消息队列接收，但是如何及时将用户是否抢成功的结果反馈给前端用户，这个没有说明！

用户是下单成功页面停留？间隔刷新去获取最后的抢购结果吗？还是说单独给个页面让用户查询自己是否抢购成功！...

展开

LionHeart
2019-05-17

有个奇怪的想法，比如10万个人12:00秒杀10个商品，系统在12:00前在10万个人中挑选10个用户id放缓存，真正秒杀时直接判断用户id在不在缓存中，在缓存就进入后面的逻辑，最差的情况就是缓存中的10个用户最终没有来秒杀，那再走正常的秒杀逻辑

展开

作者回复:

公众号「后...
2019-04-23

来晚了，老师讲的很好，顺便买了老师的书来学习，哈哈