什么是流量削峰？如何解决秒杀业务的削峰场景？

题目标签

学习时长：20分钟

题目难度：中等

知识点标签：流量削峰、消息队列、流量削峰漏斗、分层过滤、MQ

题目描述

什么是流量削峰？如何解决秒杀业务的削峰场景？

1.流量削峰的由来

主要是还是来自于互联网的业务场景，例如，马上即将开始的春节火车票抢购，大量的用户需要同一时间去抢购；以及大家熟知的阿里双11秒杀，
短时间上亿的用户涌入，瞬间流量巨大（高并发），比如：200万人准备在凌晨12:00准备抢购一件商品，但是商品的数量是有限的100-500件左右。

这样真实能购买到该件商品的用户也只有几百人左右， 但是从业务上来说，秒杀活动是希望更多的人来参与，也就是抢购之前希望有越来越多的人来看购买商品。

但是，在抢购时间达到后，用户开始真正下单时，秒杀的服务器后端却不希望同时有几百万人同时发起抢购请求。

我们都知道服务器的处理资源是有限的，所以出现峰值的时候，很容易导致服务器宕机，用户无法访问的情况出现。

这就好比出行的时候存在早高峰和晚高峰的问题，为了解决这个问题，就有了错峰限行的解决方案。

同理，在线上的秒杀等业务场景，也需要类似的解决方案，需要平安度过同时抢购带来的流量峰值的问题，这就是流量削峰的由来。

2.怎样来实现流量削峰方案

削峰从本质上来说就是更多地延缓用户请求，以及层层过滤用户的访问需求，遵从“最后落地到数据库的请求数要尽量少”的原则。

1.消息队列解决削峰

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。

结果没想到我们运气居然这么好，碰上个 CEO 带着我们走上了康庄大道，业务发展迅猛，过了几个月，注册用户数达到了 2000 万！每天活跃用户数 100 万！每天单表数据量 10 万条！高峰期每秒最大请求达到 1000！同时公司还顺带着融资了两轮，进账了几个亿人民币啊！公司估值达到了惊人的几亿美金！这是小独角兽的节奏！

好吧，没事，现在大家感觉压力已经有点大了，为啥呢？因为每天多 10 万条数据，一个月就多 300 万条数据，现在咱们单表已经几百万数据了，马上就破千万了。但是勉强还能撑着。高峰期请求现在是 1000，咱们线上部署了几台机器，负载均衡搞了一下，数据库撑 1000QPS 也还凑合。但是大家现在开始感觉有点担心了，接下来咋整呢……

再接下来几个月，我的天，CEO 太牛逼了，公司用户数已经达到 1 亿，公司继续融资几十亿人民币啊！公司估值达到了惊人的几十亿美金，成为了国内今年最牛逼的明星创业公司！

但是我们同时也是不幸的，因为此时每天活跃用户数上千万，每天单表新增数据多达 50 万，目前一个表总数据量都已经达到了两三千万了！扛不住啊！数据库磁盘容量不断消耗掉！高峰期并发达到惊人的 5000~8000 ！你的系统肯定支撑不到现在，已经挂掉了！

好吧，所以你看到这里差不多就理解分库分表是怎么回事儿了，实际上这是跟着你的公司业务发展走的，你公司业务发展越好，用户就越多，数据量越大，请求量越大，那你单个数据库一定扛不住。

消息队列中间件主要解决应用耦合，异步消息， 流量削锋等问题。常用消息队列系统：目前在生产环境，使用较多的消息队列有 ActiveMQ、RabbitMQ、 ZeroMQ、Kafka、MetaMQ、RocketMQ 等。

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。

具体的消息队列MQ选型和应用场景可以参考分布式之消息队列的特点、选型、及应用场景详解。

2.流量削峰漏斗：层层削峰

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。 尽量把数据量和请求量一层一层地过滤和减少了。

1）分层过滤的核心思想

- 通过在不同的层次尽可能地过滤掉无效请求。
- 通过CDN过滤掉大量的图片，静态资源的请求。
- 再通过类似Redis这样的分布式缓存，过滤请求等就是典型的在上游拦截读请求。

2）分层过滤的基本原则

- 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。
- 对写请求做限流保护，将超出系统承载能力的请求过滤掉。
- 涉及到的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。
- 对写数据进行强一致性校验，只保留最后有效的数据。

最终，让“漏斗”最末端(数据库)的才是有效请求。例如：当用户真实达到订单和支付的流程，这个是需要数据强一致性的。

3.流量削峰总结

1.对于秒杀这样的高并发场景业务，最基本的原则就是将请求拦截在系统上游，降低下游压力。如果不在前端拦截很可能造成数据库(mysql、oracle等)读写锁冲突，甚至导致死锁，最终还有可能出现雪崩等场景。

2.划分好动静资源，静态资源使用CDN进行服务分发。

3.充分利用缓存(redis等)：增加QPS，从而加大整个集群的吞吐量。

4.高峰值流量是压垮系统很重要的原因，所以需要Kafka等消息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。