一说起秒杀，大家都觉得这事很有技术含量。实际上，并不是这个样子的，秒杀这种互联网的交易方式其实并没有我们想像中的那么复杂。
下面先让我们来简单地看一下，秒杀是怎么做的。

秒杀的流程

“秒杀”其实是商家为了促销，使用非常低的价格销售商品，比如，1 元卖 iPhone，100 台，于是来了一百万人抢购。

我们把技术挑战放在一边，先从用户或是产品的角度来看一下，秒杀的流程是什么样的。

- 首先，你需要一个秒杀的 landing page，在这个秒杀页上有一个倒计时的按钮。
- 一旦这个倒计时的时间到了，按钮就被点亮，让你可以点击按钮下单。
- 一般来说下单时需要你填一个校验码，以防止是机器来抢。

从技术上来说，这个倒计时按钮上的时间和按钮可以被点击的时间是需要后台服务器来校准的，这意味着：

- 前端页面要不断地向后端来请求，开没开始，开没开始……
- 每次询问的时候，后端都会给前端一个时间，以校准前端的时间。
- 一旦后端服务器表示 OK 可以开始，后端服务会返回一个 URL。
- 这个 URL 会被安置在那个按钮上，就可以点击了。
- 点击后，如果抢到了库存，就进入支付页面，如果没有则返回秒杀已结束。

这个不断轮询的过程，就好像大家等着抢。你想想，有 100 万人来不停地询问有没有开始了这个事，估计后端也扛不住。

秒杀的技术挑战

接下来，我们需要来看一下“秒杀”的技术挑战。

面对上面我们要解决的技术问题，我们的技术上的挑战就是怎么应对这 100 万人同时下单请求？100 万的同时并发会导致我们的网站瞬间就崩溃了，一方面是 100 万人同时请求，我们的网络带宽不够，另一方面是理论上来说要扛 100 万的 TPS，需要非常多的机器。

但是最恐怖的是，所有的请求都会集中在同一条数据库记录上，无论是怎么分库分表，还是使用了分布式数据库都无济于事，因为你面对的是单条的热点数据。

这几乎是一件无法解决的技术问题。

秒杀的解决方案

很明显，要让 100 万用户能够在同一时间打开一个页面，这个时候，我们就需要用到 CDN 了。数据中心肯定是扛不住的，所以，我们要引入 CDN。

在 CDN 上，这 100 万个用户就会被几十个甚至上百个 CDN 的边缘结点给分担了，于是就能够扛得住。然后，我们还需要在这些 CDN 结点上做点小文章。

一方面，我们需要把小服务部署到 CDN 结点上去，这样，当前端页面来问开没开始时，这个小服务除了告诉前端开没开始外，其还会做一下有多少人在线的一个统计。每个小服务会把这当前在线等待秒杀的人数每隔一段时间就回传给我们的数据中心，于是我们就知道全网总共在线的人数有多少。

假设，我们知道有大约 100 万的人在线等着抢，那么，在我们快要开始的时候，由数据中心向各个部署在 CDN 结点上的小服务上传递一个概率值，比如说是 0.02%。于是，当秒杀开始的时候，这 100 万用户都在点下单按钮时，首先请求到的是 CDN 上的这个小服务，这个小服务按照 0.02% 的量把用户放到后面的数据中心，也就是 1 万个人放过去两个，剩下的 9998 个都直接返回秒杀已结束。

于是，100 万用户被放过了 0.02% 的用户，也就是 200 个左右，而这 200 个人在数据中心抢那 100 个 iPhone，也就是 200 TPS，这个并发量怎么都应该能扛住了。

这就是整个“秒杀”的技术细节，是不是有点不敢相信？

说到这里，我相信你一定会问我 12306 和奥运会抢票的问题。我觉得 2008 年奥运会抢票把服务器抢挂了是可以使用秒杀这个解决方案的。而 12306 则不行，因为他们完全不知道用户来是要买哪张火车票的。不知道这个信息，很不好过滤用户，而且用户在买票前需要有很多查询操作，然后在查询中选择自己的车票。

对此，12306 最好的应对方式，除了不要一次把所有的票放出来，而是分批在不同的时间段把票放出来，这样可以让人们不要集中在一个时间点来抢票，做到人肉分流，可以降低一些并发度。

另外，我一直觉得，12306 最好是用预售的方式，让大家把自己的购票先输入到系统中。系统并不真正放票，而是把大家的需求都收集好，然后做整体统筹安排，该增加车次的增加车次，该加车厢的加车厢，这样可以确保大家都能走。实在不行，那就抽签了。

更多的思考

我们可以看到，解决秒杀这种特定业务场景，可以使用 CDN 的边缘结点来扛流量，然后过滤用户请求（限流用户请求），来保护数据中心的系统，这样才让整个秒杀得以顺利进行。

那么，如果我们像双 11 那样，想尽可能多地卖出商品，那么就不像秒杀了。这是要尽可能多地收订单，但又不能超过库存，其中还有大量的银行支付，各大仓库的库存查询和分配，这些都是非常慢的操作。为了保证一致性，还要能够扛得住像双 11 这样的大规模并发访问，那么，应该怎么做呢？

使用秒杀这样的解决方案基本上不太科学了。这个时候就需要认认真真地做高并发的架构和测试了，需要各个系统把自己的性能调整上去，还要小心地做性能规划，更要把分布式的弹力设计做好，最后是要不停地做性能测试，找到整个架构的系统瓶颈，然后不断地做水平扩展，以解决大规模的并发。

但是，从另一方面来说，像我们用边缘结点来解决秒杀这样的场景的玩法，是否也有一定的普适性？这里，我想说，一定是有的。

有些时候，我们总是在想数据中心的解决方案。其实，我们有时候也需要换一换思路，也许，在数据中心解决并不一定是最好的方式，放在边缘来解决可能会更好一些。尤其是针对一些有地域特征的业务，比如像外卖、共享单车、打车这样的业务。其实，把一些简单的业务逻辑放在边缘，比放在数据中心不但能够有更好的性能，还有更便宜的成本。

我觉得，随着请求量越来越大，数据也越来越多，数据中心是有点到瓶颈了，而需要边缘结点来帮忙了。而且，这个边缘化解决方案的趋势也会越来越有优势。

在这里，我先按住不表，因为这是我的创业方向，我会在下一篇文章，也是本系列的最后一篇文章，向你介绍边缘计算以及我想用边缘计算干些什么事。

小结

好了，我们来总结一下今天分享的主要内容。首先，我介绍了秒杀。先是分析了其业务流程，并列举了其所面临的技术挑战，随后介绍了其解决方案。接着，分析了相关的奥运会和 12306 抢票问题，以及双十一购物节问题。

它们各自有不同的解决思路，其中双十一则要求我们必须认认真真地用高并发架构来应对。最后，从秒杀解决方案中的 CDN 边缘节点计算，我引出了普适的边缘节点计算。下篇文章中，我们详细讲述边缘计算。希望对你有帮助。

也欢迎你分享一下你参与过秒杀系统的构建吗？双十一呢？解决方案是怎样的呢？

文末给出了《分布式系统设计模式》系列文章的目录，希望你能在这个列表里找到自己感兴趣的内容。