链滴

售票系统卖票问题总结

作者：xjlnjut730
原文链接：https://ld246.com/article/1548563743559
来源网站：链滴

这篇文章总结一下最近朋友公司遇到的售票系统上线以及卖票的问题。

这是个售票系统，卖明星的演唱会门票。这种门票的售卖场景其实跟秒杀是类似的，都是集中在某一时间段内并发订单量会非常很高。年前卖票一共4场，每场订单量在1w以内。接下来还原每一场的问。

一、第1场

1.1 订单保存接口并发慢

在上线之前，对售票系统进行压测，发现订单保存接口很慢，差不多200并发就会存在异常响应。扣库存过程用的是redis分布式锁，分布锁本身存在羊群效应，每次只有一个人能过，并发肯定上不去，以在高并发下场景下，效果不好，某些订单会超时，超时也带来另一个问题，用户可能已经退出来再尝试发起订单了，会增加系统负载。

由于时间紧急，来不及优化，考虑到订单不多，硬着头皮上线了。这也考虑到了即便部分用户失败也可以接受的，只要票成功卖掉就行了。好在上线的时候，数据库的问题盖过了该问题，订单保存的接没出问题。

1.2 数据库满负荷

出问题的是数据库，由于监控没有跟上，1个小时之后才发现数据库CPU一直100%。由于有测试环境压测，压根没有想到数据库会存在瓶颈。好在用的是云服务，发现后直接扩容解决了该问题，数据库负载下来了。回顾了下原因，主要是线上库的数据量比测试环境大多了，压测时没有考虑这一点。当还有一些其它问题，比如索引也没有用好，字段设计存在很多不合理的地方，字段太长等等。

1.3 超卖

这一场存在超卖的情况，大致超了几十张票的样子，分散到每个票档，不多。上线时对这个问题比较视，预留了库存，业务上解决了这个问题。技术并没有找到问题在哪里，因为测试压测过，没有问题并且用了分布式锁，理论是解决了库存问题的。不过，由于订单保存接口要优化，就没有继续调查下。

二、第2场

第2场，我们对订单保存接口做了调整。主要测试了两个版本：

1. 对redis锁做优化，尝试活锁检测

结果测试下来，并发上并没有提升，因为最终还是一群“羊”一个个过“栅栏门”。

2. 使用redis队列来实现库存扣减

测试环境测试了下，并发数上去了很多，将近1000的并发，没有报错，性能提升了将近10倍，在测环境没有超卖的情况。目前是最优的方案了。

2.1 答题列表缓存不对

在下单之前，有一个答题模块，为阻止黄牛的模块。在上线前，为了能够进行压测，把题库改成了测题库。结果上线之后，题库还是测试题库。影响了正常购票。发现后，我们就去redis里删除题库，但一直删除不了，还是旧题库。这里要吐槽下，Redis Desktop Manager不知道为什么删除不了，去

令行下redis-cli执行del就成功了，关键时刻还是官方命令行靠谱。

2.2 超卖

这场还是超卖了，而且比较严重。这并不是redis队列的错，而是上线时失误了，一个订单可以卖两张，结果上线的时候，代码的逻辑还是一张的。

这场实在协调不了票，只有退款了。公司的客服面临了很大的压力，很多明星的粉丝是哭着打电话过的。真的心酸无助，做技术的切记：仔细仔细再仔细，否则一个人几天的心情说不定就被你毁了，跟多多优惠券毁了一个部门的年终奖是一个道理。

二、第3场

第3场主要是把程序完善了。但是还是遇到了超卖问题，不过找到了前几场一直存在的问题。

3.1 超卖

这次分批次手工放票卖了，最终只超卖了一张，但是在放票过程中，其实一直有超卖的情况，而且不。但是不能理解的是，已经用了redis队列，仔细核对订单保存代码，也没有发现问题。测试压测也没问题。到底问题出现在哪儿呢?

通过分析数据，最终发现，是另一个支付环节出了问题：

系统配置的订单超时是5分钟，但支付超时却设置成了30分钟。

有将近50笔订单在超时5分钟的时候被标记为过期并回滚库存了。而在支付成功后，系统没有做判断接将订单状态又改成了支付成功。这些订单不在库存限制之中，所以出现了超卖。

需要做以下优化：

1. 支付超时与订单超时设成一致。

2. 超时时间设置得更长一些，减少因为支付过长导致的问题，比如10分钟。

3. 支付成功时，判断当前订单状态是否已经变成过期，如果过期，则标记为需退款状态，由人工进行款处理。

其中第3点在业务上不被允许的，因为付了钱就是合同成立了，单方面退款算违约。但是我觉得还是要有个日志或是通知机制来保底，不信任第三方的判断是需要的。

四、第4场

// TODO

但愿没问题

总结

1. 分布式锁尽量在低并发的情况下用，比如更新过期订单这种。高并发下分布式锁并不是一个好的解方案，会存在性能瓶颈。

2. 数据库设计要慎重，字段的长度一定要限定好，用索引的时候，字段类型一定要匹配。

3. 代码里的日志对各种异常情况一定要详细记录，方便回顾定位问题。日志会对性能有影响，但几乎

会产生决定性的影响。

4. 理解业务远比理解技术重要，技术架构也是在明确了业务之后才能确定的。不清楚，就动手画画图对理解特别有帮助。