慕课网秒杀课程有感——该补充一下数据高并发下的安全和一致性的知识原创

突然发现慕课网有秒杀课程，简单的看了一下DAO层，基本都在整合ssm框架，教教新人是可以的，但是这里需要补充的是使用秒杀需要考虑性能和数据安全性一致性的问题，所以这里我会分析三种在保证数据安全一致性前提下的算法。

秒杀最困难的除了是性能问题，还有另外一个就是高并发下的数据安全和一致性的问题。比如抢红包，线程1,2,3，极有可能刚开始都可以读到余额为100元，但是扣减的时候也是有先后顺序的，这些会引起数据并发安全和一致性的问题，这里互联网高并发的核心问题，也是秒杀的核心问题，所以这里我有必要对这些知识进行一定的补充和描述。

1、悲观锁

讲课老师谈到的用数据库行锁：

select * from t_table where id =xx for update;

直至其提交，解锁for update。对的这样可以解决线程数据安全性的问题，然而这种在企业中基本不会用得到？为什么？因为这个是一个独占锁，当加入行锁后，所有的线程都需要等待持锁的线程。而秒杀是什么概念？一个时刻几千个请求，也就是说如果用这样的一个锁，那么就是几千个线程等待你一个线程，需要挂起，切换线程环境。这是一个多大的代价？？所以很快这样的方法被企业排除了。

2、乐观锁

对于我们而言，使用数据库，我们更喜欢这么几个步骤：

1、我们使用无锁机制读取数据：

select * from t_table where id =xx;

现在我们表里面加入一个version字段，一开始我们读入这个version，假设我们读取到3，

2、操作逻辑完成，更新的时候使用：

update t_table somefiled = xxx, version = version + 1

where id= xx and version = 3;

这里如果version还是等于3则代表没有更其他线程更新过。

3、如果为上面update语句影响记录为1，则更新成功，继续插入秒杀业务数据；如果更新数为0，那么我们循环在从1到3步骤，直至这步返回为1。
这样的好处在于我们不需要使用独占锁，不需要一个时刻多个线程去等待一个线程，而到最后的更新，我会去判断新旧值是否被其他线程更新过（这里主要判断version字段），如果被其他线程更新过就不做更新，继续循环步骤直至成功；如果没有则可以成功处理，这就是乐观锁。乐观锁可能会引起ABA问题，这个问题读者可以自行度娘。

3、使用Redis

使用redis秒杀是我们目前最流行的方式，首先redis是一种nosql工具，数据和操作在内存上，速度快；而数据库是磁盘，速度慢，所以redis的运行速度是数据库的几十到上百倍。
1、当请求达到，开启Redis事务先watch Redis的某一个key，比如产品的总数量。
2、开始根用户扣购买减产品数量。
3、保存用户购买产品信息到redis的列表
4、提交Redis事务，如果成功修改数据，则继续，如果失败则重复第1步，直至成功。
5、检测产品是否卖完？如果卖完，则将用户购买信息，批量存入数据库。完成持久化，否则什么都不做，等待下相应下一次的请求。
其实这也是一种变相的乐观锁。
这里我们在和用户交互的过程中主要用到了redis工具，它是内存上的，读写的速度是数据库的几十倍到上百倍，所以企业更喜欢使用它进行秒杀业务。等产品卖光，或者我们抢红包的红包金额为0，我们就会触发将业务数据批量刷入数据库完成持久化的工作，而可以看到整个秒杀的过程基本在Redis完成，而数据库的持久化只是一次性的批量问题。

这就是目前处理高并发数据一致性的三种，最常用的方式，目前最快，最流行的当然是Redis秒杀，它在保证数据安全和一致性的情况下，大大提高的互联网秒杀的性能。