Mysql大并发热点行更新的问题

来源：小伙伴的面试题

一 问题

当某个秒杀商品特别吸引人时，比如iPhone2100，1元秒杀 ，100W人抢1W个商品

肯定有一大波人为了少卖一个肾 疯狂去抢申请资格。有甚者利用机器人、外挂去秒杀

iPhone2100的**库存数**，update 操作给表加上行锁，导致后面的请求全部排队等待前面一个update完成,释放行锁后才能处理下一个请求。

一次 200毫秒，1秒能搞5次， 1W次需要2000S，快一个小时了

热点行导致的雪崩的来了

大量后来请求等待，RT飙高，占用了数据库的连接。一旦数据库连接数被占满（默认mysql的最大并发连接数是100个，最大可以达到16384个），数据库连接耗尽，就会导致服务层请求因拿不到连接

服务层RT会异常飙高, 业务请求出现无法及时处理，服务层连接就会被占满

服务层连接耗尽，接入层的请求得不到处理，接入层RT会异常飙高,

数据库系统的RT会异常飙高,服务层RT会异常飙高, 接入层RT会异常飙高, 用户请求RT会异常飙高, app层的连接耗尽，一系列的雪崩效应!

二 解决方案设计

从上面的背景分析,解决热点数据并发更新需要注意核心问题:

> 1 限流：保护接入层不崩溃
>
> 2 然后锁定有效流量： 用高性能的组件，然后锁定有效流量，比如使用nginx+redis+lua实现数据的过滤
>
> 3 进行流量削峰（同步调用降低）： 用高性能的消息队列组件，进行流量削峰，数据库进行异步的慢速消费
>
> 4 分段操作： 空间换时间，分段进行操作，把一行数据分成多个段

三 热点行分段实操

SeckillSegmentStockPO

SeckillSegmentStockDao

服务层:

stock-provider

initSegmentStockAmount

seckill-provider

executeSeckill

配合Redis分段锁来使用

seckill-provider

executeSeckill

doInTransaction

四 其他的解决方案

方案方法，实际上都是八仙过海，各显神通，千千万万，甚至优劣难分。

大家面试的时候，如果遇到这种问题，重要的是：

头头是道的就分析，逐层递进，这个陈述的过程很重要。

总之，把上面的过程介绍清楚，及格，一点问题没有。