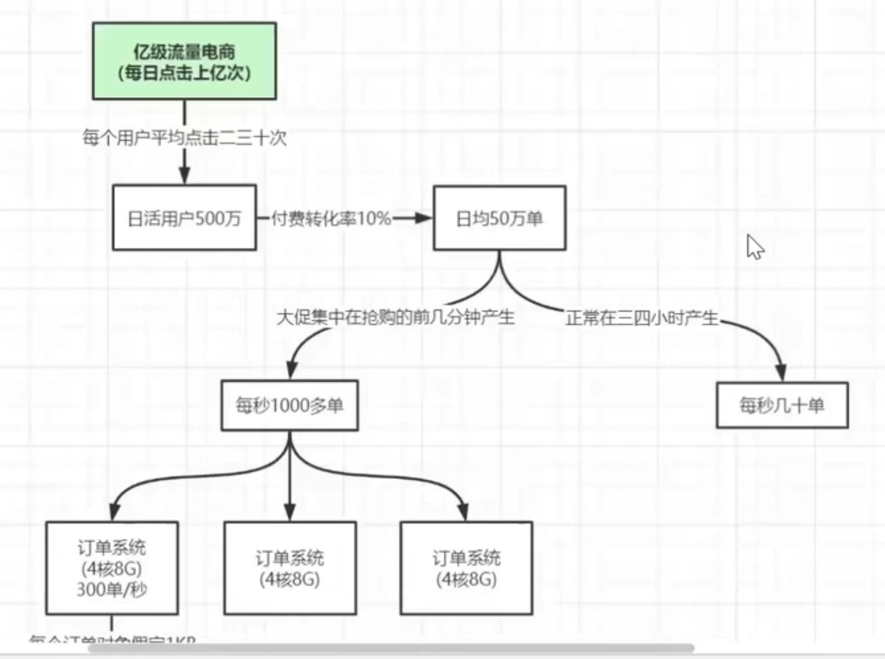
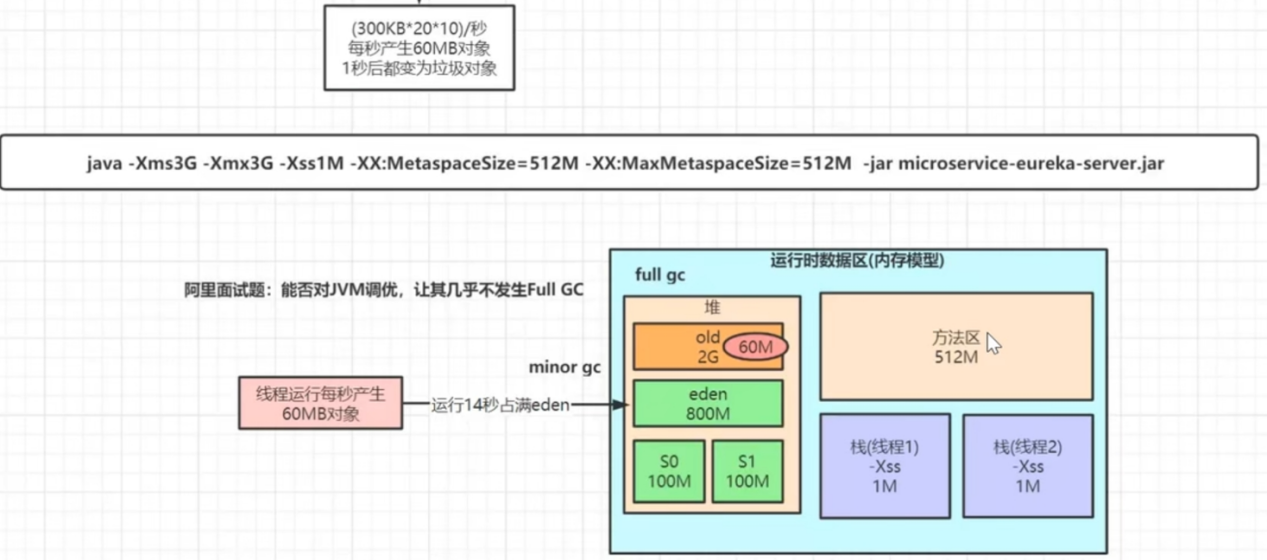
# 亿万流量电商系统JVM优化





# 电商案例

## 并发扣款，如何保证数据的一致性？

方案：悲观锁 -》乐观锁（CAS，无法解决ABA问题）- 》仅对比版本号

1. 悲观锁

悲观锁方案确实可行，但要引入额外的组件(redis/zk)，并且会降低吞吐量。

1. CAS

使用CAS解决高并发时数据一致性问题，只需要在进行set操作时，compare初始值，如果初始值变换，不允许set成功。

*UPDATE t\_yue SET money=$new\_money WHERE uid=$uid AND money=$old\_money;*

根据返回中影响的行数判断是否执行成功

1. 版本号

参考Activiti表版本号字段

其他问题：

1. 能不能使用直接扣减方案？

不可以，并发情况下会将money扣减为负数

# 秒杀系统的设计

参考：

<https://www.cnblogs.com/clarino/p/11932800.html>

https://tech.ctrip.com/articles/a\_architecture/13266/   1分钟售票8万张！门票抢票背后的技术思考

## 分布式锁扣减

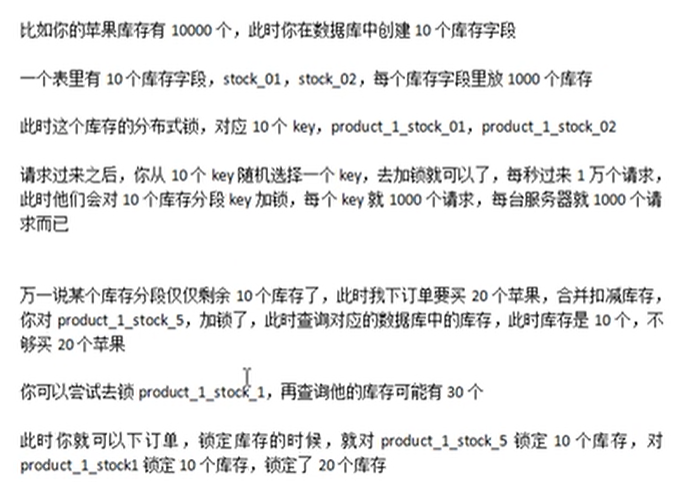
* 方法1：悲观锁：

每个都申请分布式锁，但是造成并发能力弱。

* 优化：

1、分段加锁 + 合并扣减

针对单个段不满足情况，则尝试对多个段同时加锁减库存



2、在缓存中直接扣减，直到扣减后负数（回滚最近一次操作）。通过MQ异步更新数据库。

