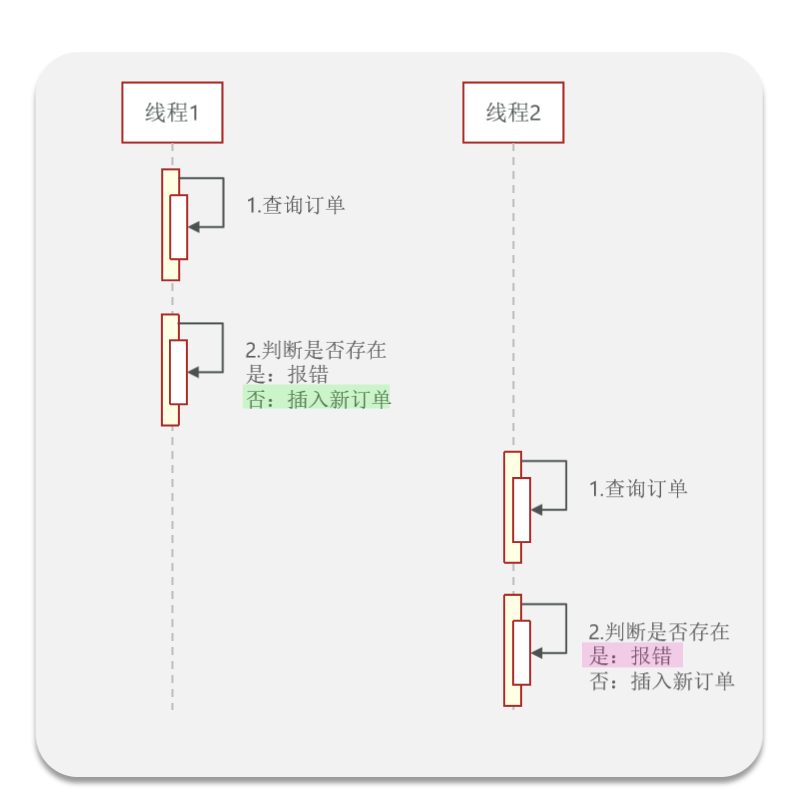
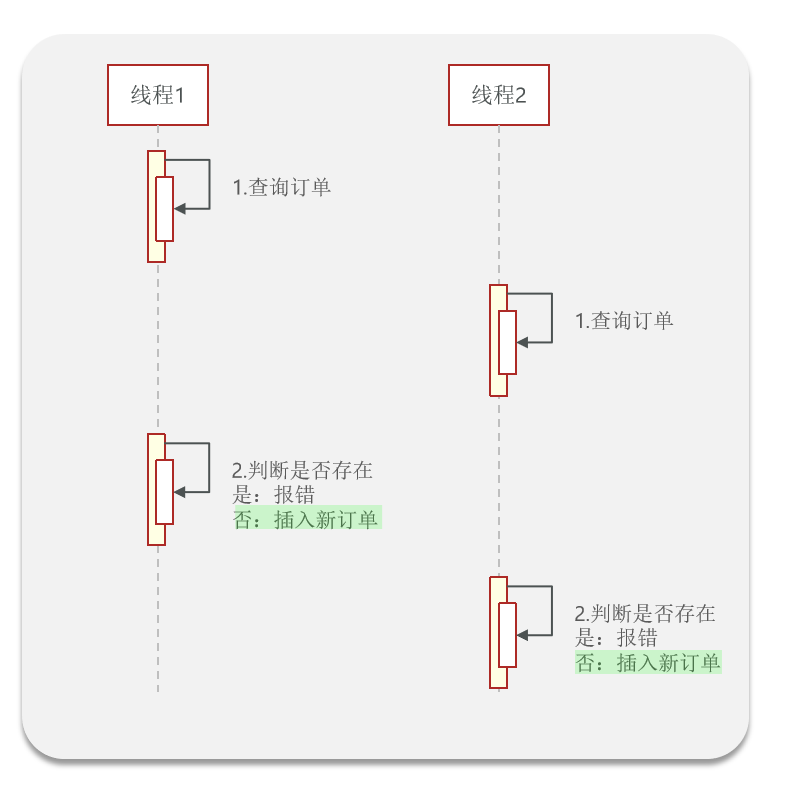
# 分布式锁

## 单体模式下一人一单并发安全问题：

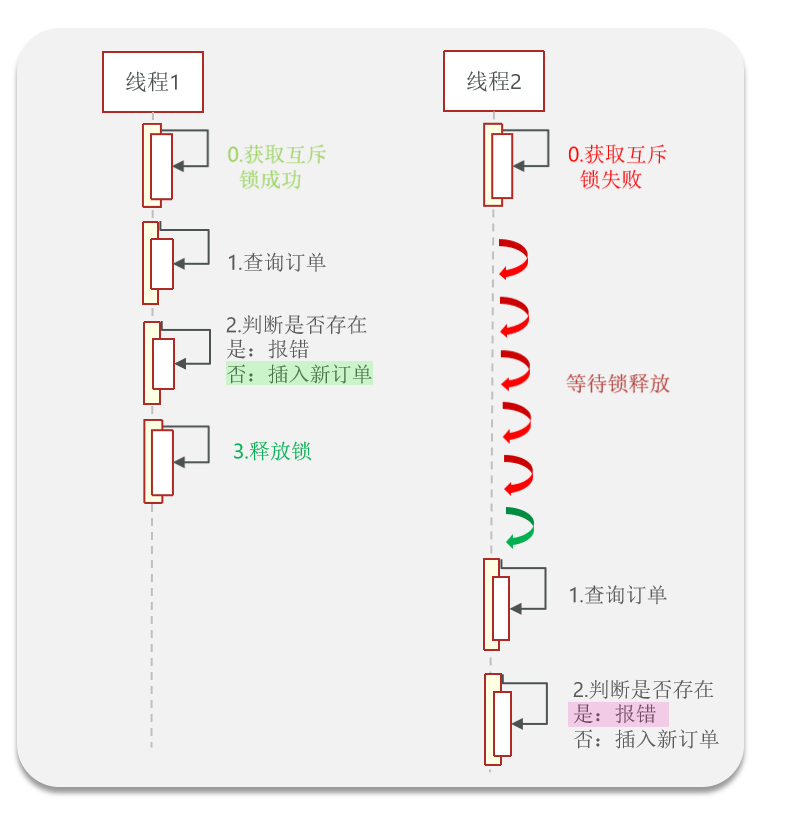
### 正常情况：



### 异常情况：



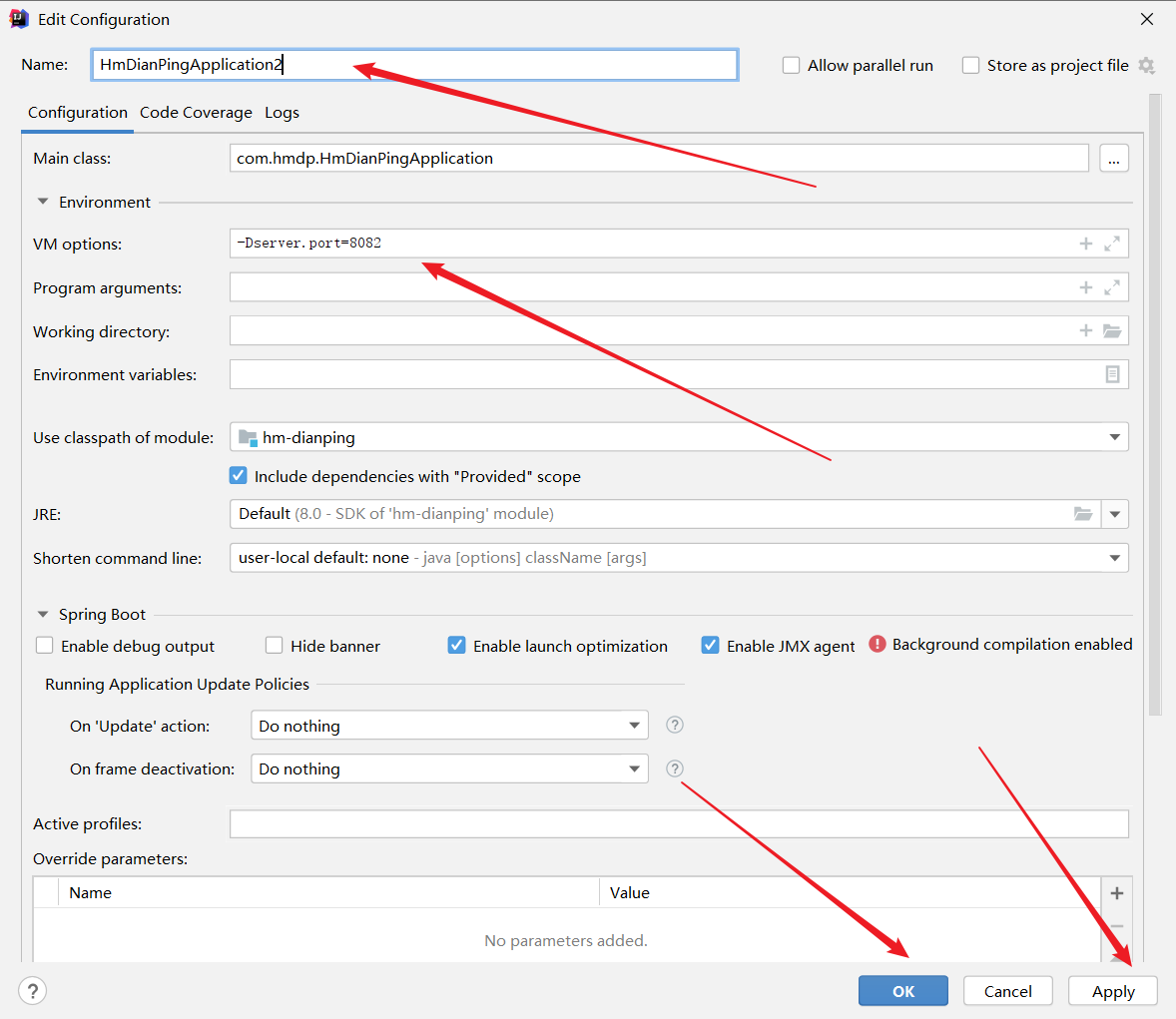
### 解决单体模式一人一单问题：



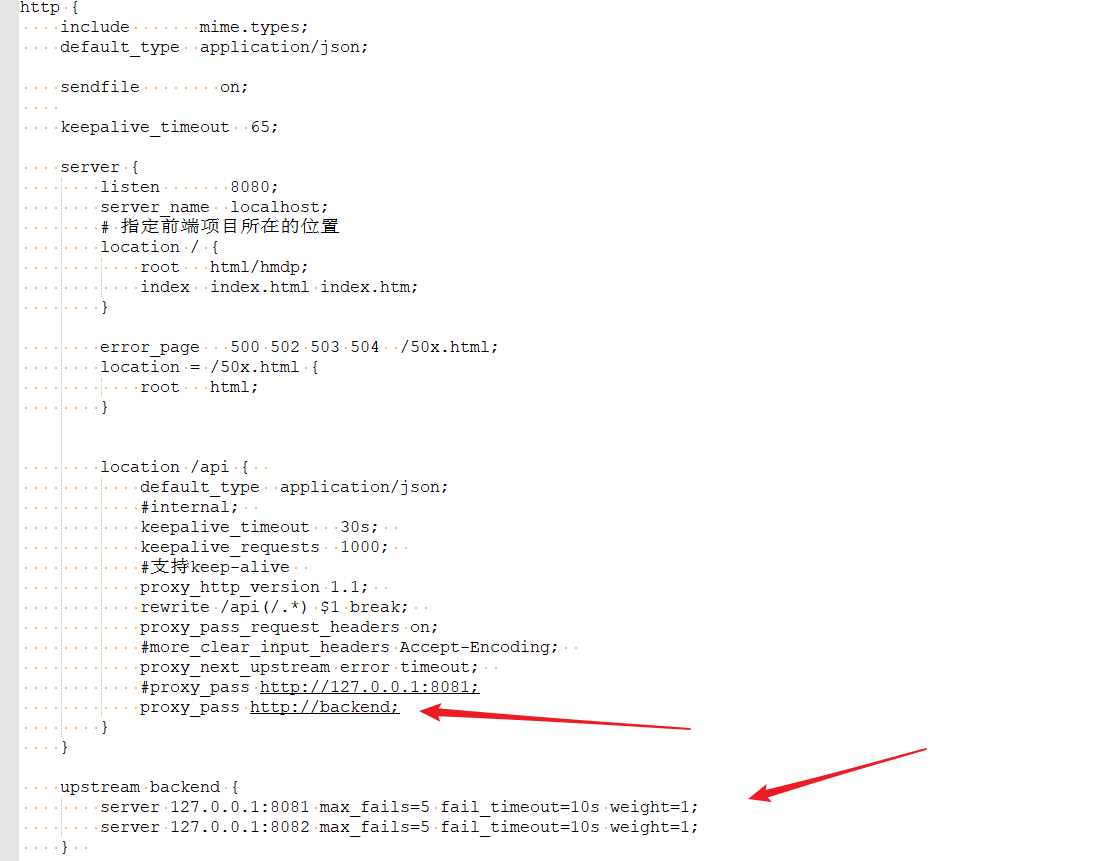
以上已经演示了秒杀下单的一人一单功能，并且基于悲观锁synchronized解决了一人一单的并发安全问题。不过这种解决方案仅仅适合在单机情况下使用，我们的项目虽然是单体项目，但是将来如果有高并发场景，我们可以做集群部署的，就是把一个项目部署到多个机器，形成负载均衡的集群。而在集群模式下，我们使用悲观锁synchronized就不能解决一人一单的并发安全问题了。

## 模拟集群模式下一人一单并发安全问题：

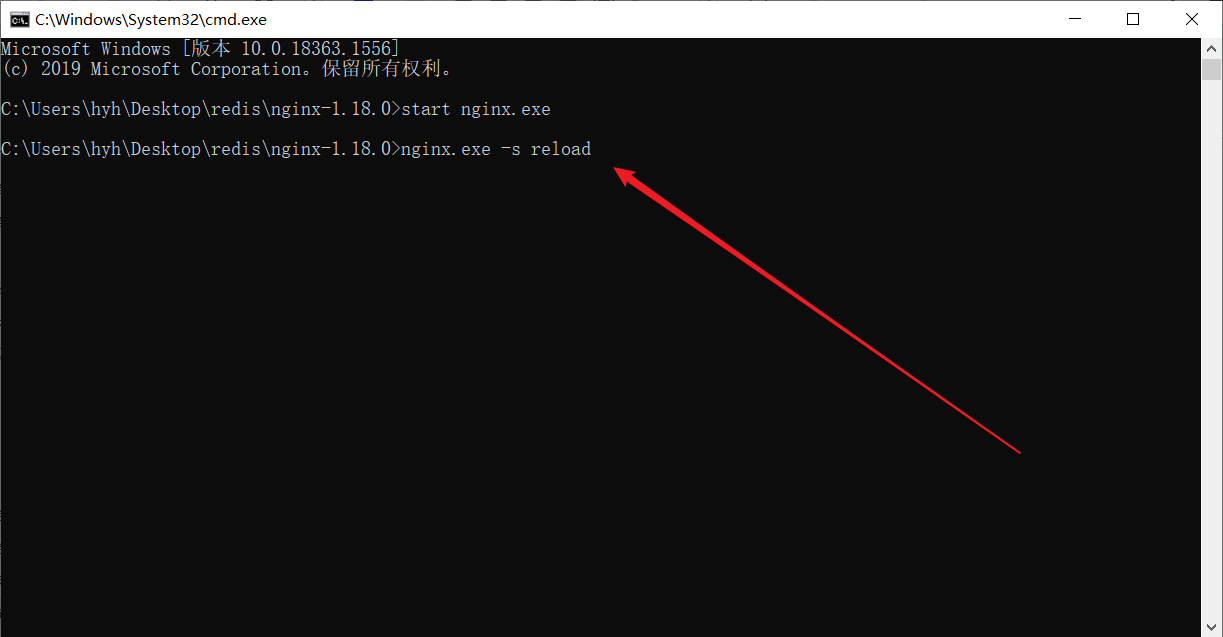
### 1：我们将服务启动两份，端口分别为8081和8082



### 2：然后修改nginx的conf目录下的nginx.conf文件，配置反向代理和负载均衡



### 3：重启nginx



### 4:：浏览器输入http://localhost:8080/api/voucher/list/1

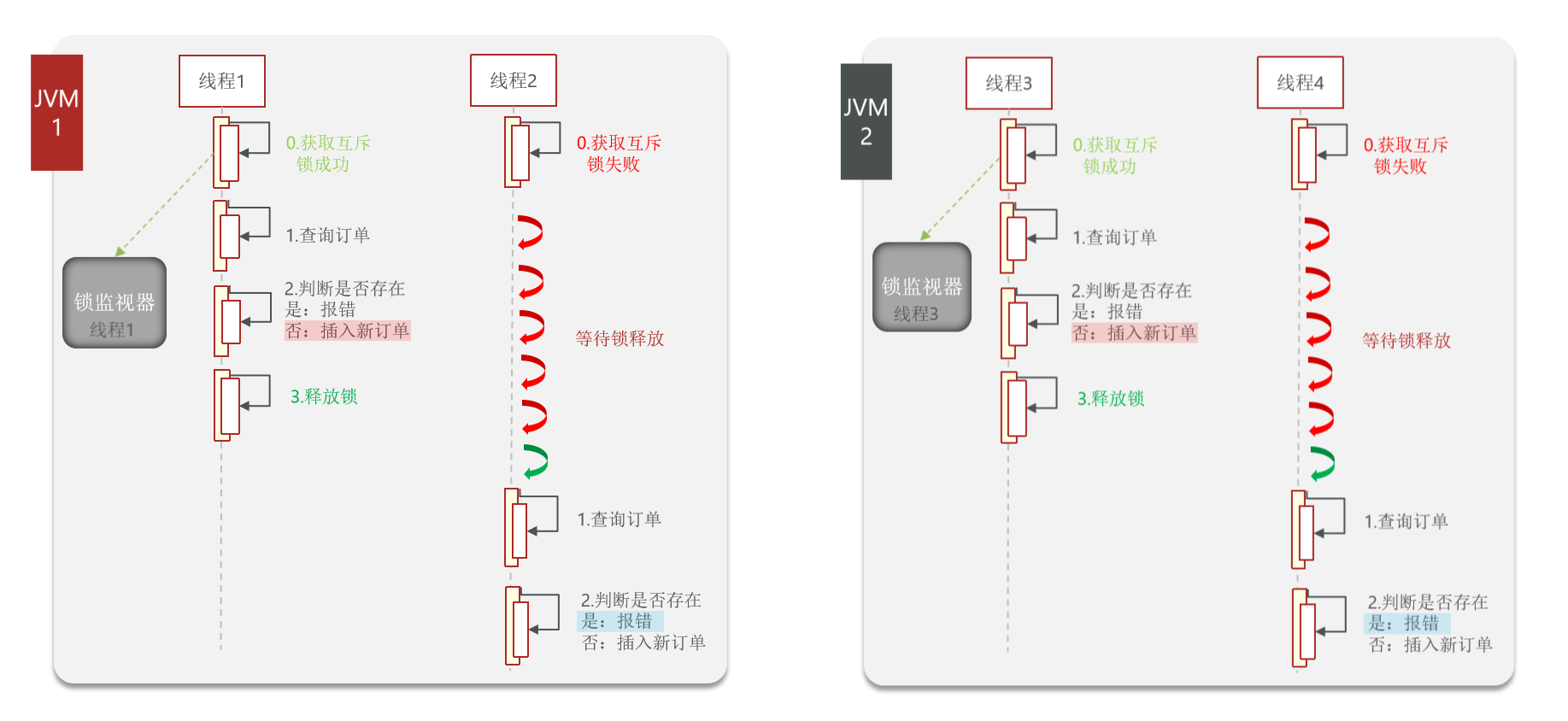
## 集群模式下的线程安全问题

### 产生原因

一人一单问题集群模式下，synchronized锁会失效，因为synchronized只能保证在单个jvm内部的多个线程互斥！没有办法让集群下的多个jvm进程互斥！！！

在集群模式下还是并发安全问题，因为集群时有多个jvm的存在，每个jvm都有自己的锁监视器（synchronized就是利用jvm的锁监视器来控制线程的），导致同一个JVM中的每个锁有可以有一个线程获取。于是出现并行运行，那么就可能出现安全问题。

也就是说锁监视器在当前JVM的内部可以监视到线程实现互斥，但是如果有多个JVM，就会有多个锁监视器，因为集群模式先一个服务一个JVM。一个JVM内部维护一个锁监视器对象，即不同JVM有各自的锁监视器，集群下就会出现锁失效的情况。



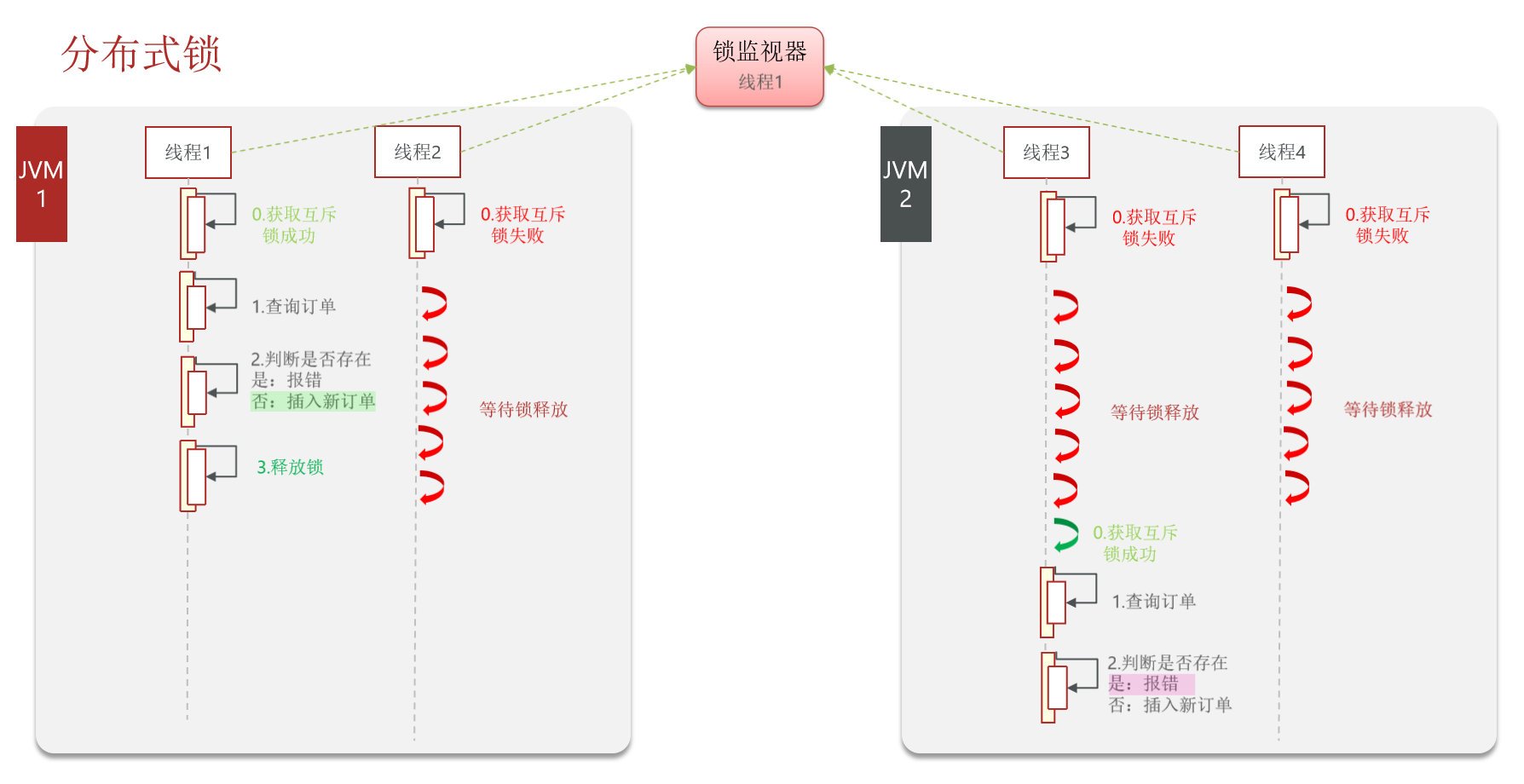
### 解决方法：分布式锁

让多个JVM只能使用同一把锁！即需要跨JVM（跨进程）的锁！！也就是分布式锁！！！这样的锁啊可不是JDK里面有提供的，需要我们自己去实现。分布式锁就是跨JVM、跨进程的锁。

## 分布式锁介绍

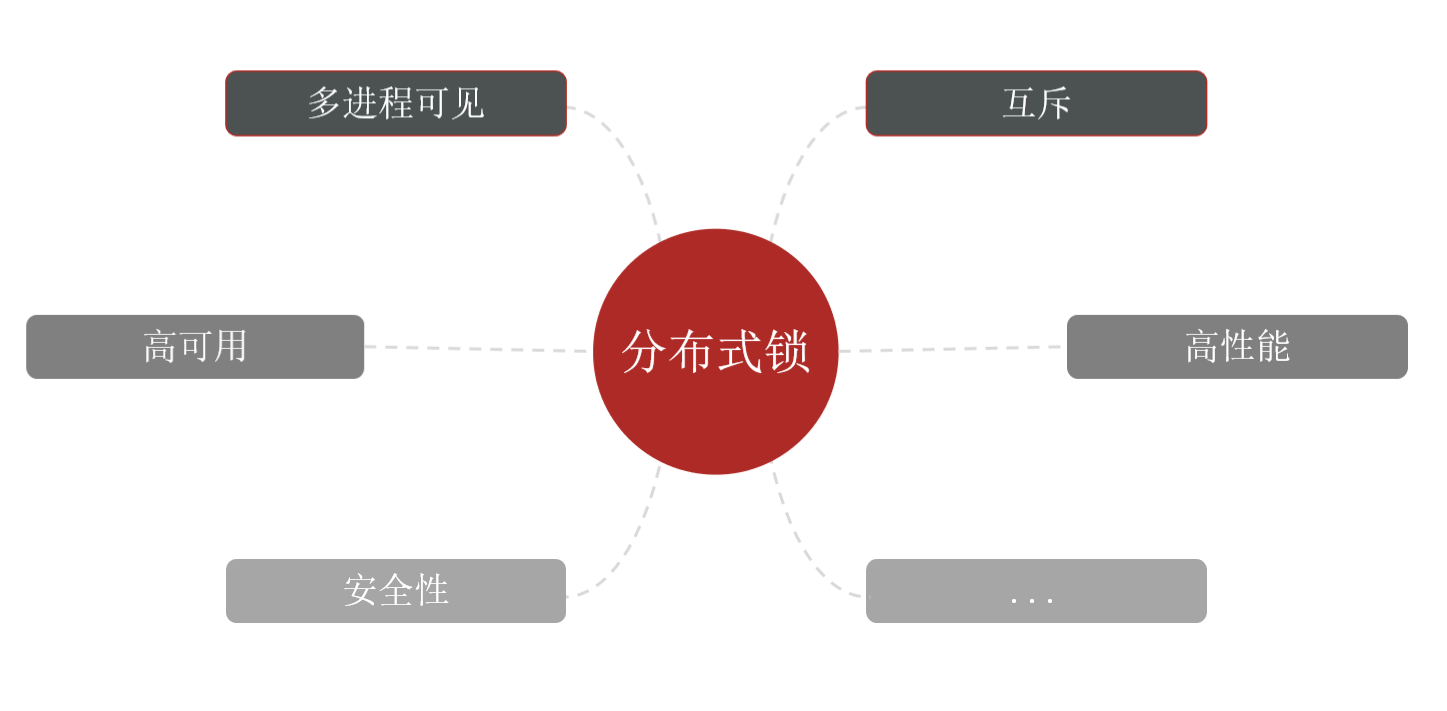
### 什么是分布式锁

我们已经发现，在集群模式下Synchronized失效了，Synchronized只能保证单个JVM内部的多个线程之间的互斥，没有办法让集群下的多个JVM进程之间互斥！这就是集群模式线程安全问题的原因。要解决这个问题，必须使用分布式锁。

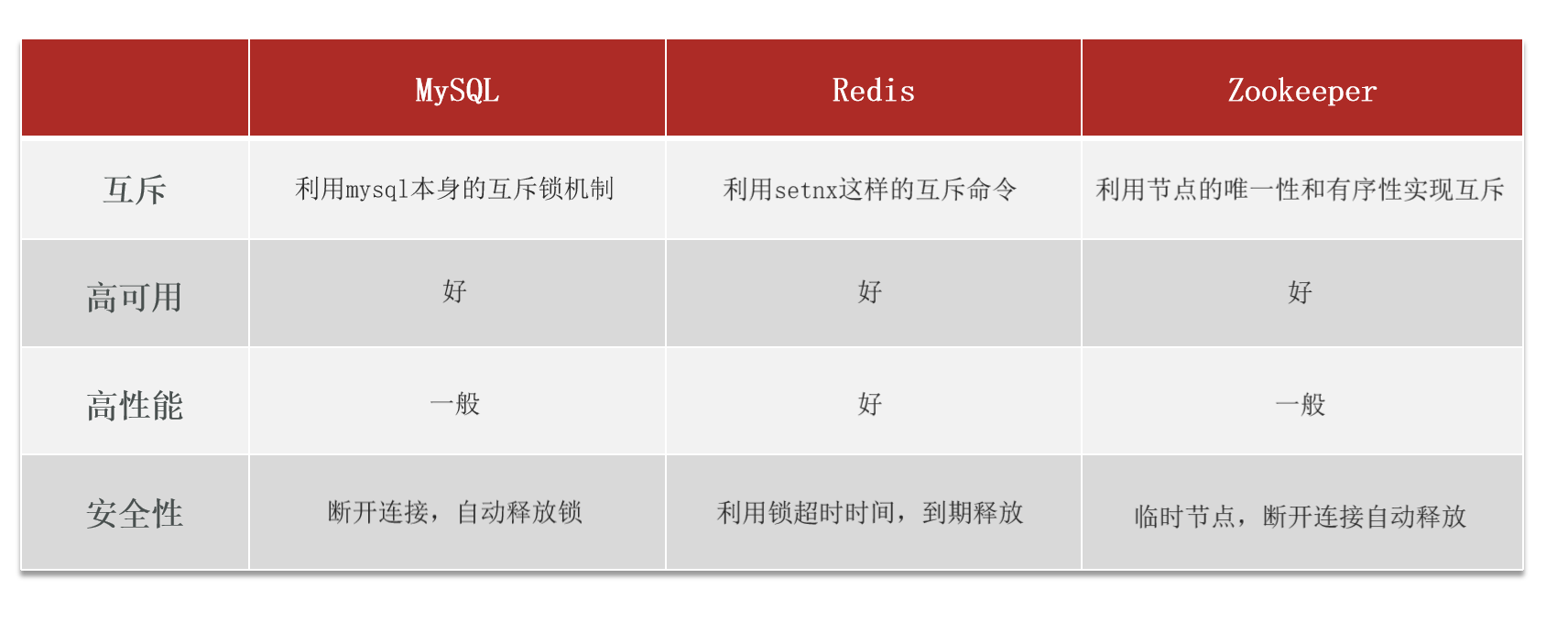


分布式锁，就是在分布式系统或者集群模式下**多进程可见并互斥的锁**，满足：多进程可见、互斥、高可用、高性能/高并发、安全。

### 分布式锁要求：



### 分布式锁实现方式：

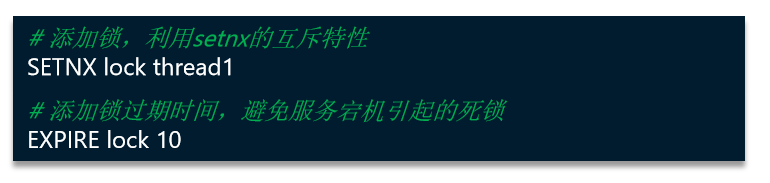


## 基于Redis实现分布式锁原理：

实现分布式锁时需要实现两个基本方法，就是获取锁和释放锁！

### 1：获取锁：

互斥：确保只能有一个线程获取锁，可以**利用redis命令setnx的互斥特性；但是要对锁添加超时时间**，避免服务宕机引起的死锁，即还没来及去释放锁，web服务宕机了,这样一来释放锁的动作永远的得不到执行，也就意味着这个锁永远存在，这样其他线程也就进不来，这样整个业务就处于死锁状态。



如果获取锁分成以上两个命令是有问题的：我们加过期时间的母的就是为了避免因为服务宕机导致锁无法示范，但是存在这种情况：当执行SETNX成功了，还没来得及执行EXPIRE，此时服务宕机了，即服务宕机发生在这两个命令之间，那么锁依然不会释放！所以说必须保证SETNX和EXPIRE要么都成功、要么都失败，即必须具备原子性！于是，上面的两条命令可以写成下面的一条：



到这已经可以去获取一个分布式锁了，不过还有一个小小的问题值得我们讨论：就是在获取锁的时，结果要么成功，要么失败，成功了返回ok，失败了返回nil，但是失败了以后我们怎么做呢？在JDK里提供的锁有两种机制，一种是获取锁失败了会阻塞等待，等到有人释放锁为止，即是阻塞等待式地获取；一种是非阻塞式的获取锁，也就是我来尝试获取锁，如果我获取失败了，我会立即结束，返回一个结果，而不是说一直去尝试，一直等待。

我们实现非阻塞式的分布式锁，因为阻塞式一对CPU有一定的浪费，因为一直等待重试啊，二实现起来会相对麻烦！所以我们会用非阻塞式的方式获取分布式锁，也就是说**set key value nx ex seconds**执操作执行一次，根据返回结果，如果成功返回true，失败返回false！

### 2：释放锁：

手动释放；

超时释放：获取锁时添加一个超时时间；



### 实现分布式锁流程图如下：

