分布式锁的解决方案：

基于mysql的实现方案：

利用数据库自身提供的锁机制实现，要求数据库支持行级锁。

优点：

实现简单，稳定可靠

缺点：性能差，容易死锁，无法优雅的实现阻塞式锁

实现方法：

阻塞式加锁

利用数据库插入数据主键唯一性，在加锁时向数据库插入数据

如果插入数据id的不存在数据库里，则插入数据成功，获得了锁。

如果数据已存在，则插入数据失败，catch抛出的异常，在catch里等待数秒后重新递归调用获取锁的方法。

基于redis的实现方案

利用setnx命令和lua脚本保证对缓存操作的原子性

优点性能好，可以设置过期时间保证不会发生死锁。

缺点实现复杂，无法优雅实现阻塞锁

基于zookeeper的实现方案

JDK1.6之前syn是一个重量级锁，他需要调用操作系统的函数。

操作系统将线程停止，这个过程系统需要从{**用户态}**转换为{**内核态}**。所以执行效率低。

JDK1.7之后，线程同步概念尽量在JVM级别解决了，减少了操作系统的操作。