

进程3

进程2

进程1

线程1

线程2

。。。

线程N

线程1

线程2

。。。

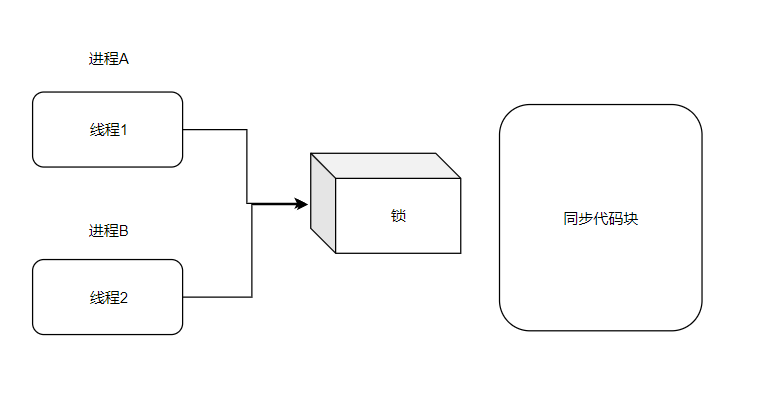
线程N

线程1

线程2

。。。

线程N



**为什么加锁可以解决问题？**

当我们创建一个对象的时候，其实不仅仅只有字段，还另外开辟两个字段，一个是类型指令，一个是同步索引块 CLR Via C# 这本书有这块内容。电子书第四版需要的找我。

**阻塞锁的事务：**

若已超时，则尝试使用提交事务的方式覆盖新的时间戳，事务提交成功（即抢到锁），

如果两个线程同时判断到了锁的过期了。。。。redis给他返回了一个当前的key的版本号

两个线程同时判断value里面的值过期了，然后拿到版本号，是1

开启事务-- 保证两个线程只能有一个线程去写key的值，只会有一个线程拿到锁

版本号是1

提交事务的时候，再去验证版本号是不是1，如果版本号是1，你可以提交，如果不是，抛异常，返回如果我们A执行的比较快，则立刻马上把值写进去，然后版本号变成2

然后同时去往redis里面写值

key的值是当前时间+timeout,当其他线程来获取锁发现key的值<当前时间,就说明锁过期了,直接设

置新的value就算拿到锁了

**非阻塞：**

尝试在redis中创建一个字符串结构的缓存项，方法传入的key，value无意思。

过期时间为传入的timeout。如果redis没有这个key，创建成功，抢到锁。，然后立即返回true，如果已经有key，立即返回false。以上过程都是全局单线程原子操作，整个过程独占式。

Timeout即成功加锁后的过期时间

利用redis在key上设置的expire time 来通过key的过期实现

**超时解决办法** ：（特殊情况才需要这样做）

比如某个业务复杂，执行的时间比较长，那就需要延长超时时间。

守护进程，延时续命。通过某种规则，比如心跳检查进某个进程，请求3次如果该进程没挂，但是没有返回值就杀掉该进程。

文件夹DistributedTransactionLock\redis包\Redis-x64-3.2.100下面管理员权限运行redis-server.exe

程序启动命令:dotnet LockService.dll minute=当前时间分钟数,测试需要多开，3个以上最好。

Lock锁运行结果: 3进程

图形用户界面

描述已自动生成

阻塞锁：3进程

图形用户界面

描述已自动生成

非阻塞锁： 并发量越大效果越好,这里开5进程，不一定卖的完

文本

描述已自动生成

显示结果有个小问题,库存是减之前的，代码已更正