## Redis分布式锁

### 单节点

获取锁：

SET resource\_name my\_random\_value NX PX 30000

释放锁：

if redis.call("get",KEYS[1]) == ARGV[1] then

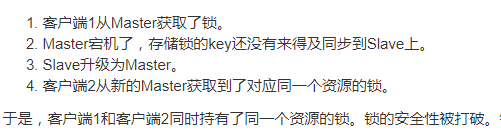
return redis.call("del",KEYS[1])

else

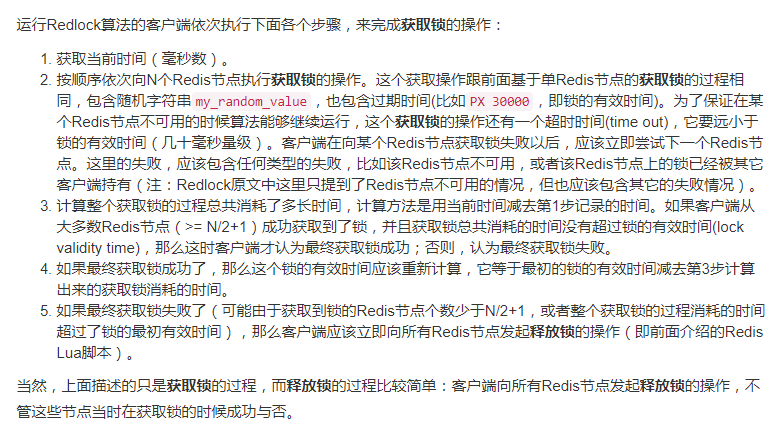
return 0

end

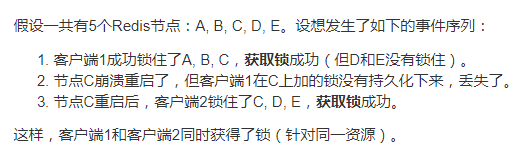
存在的问题：

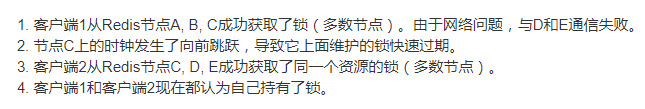
1. A持有了锁 然后发生了长时间的GC 已经超过了有效期， GC完成以后 继续执行 实际上此时锁已经被B持有了
2. 

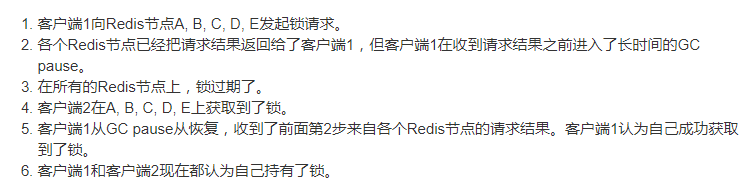
### 多节点



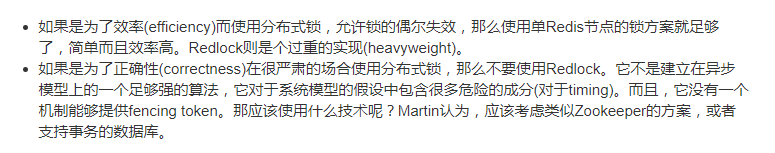
存在的问题：







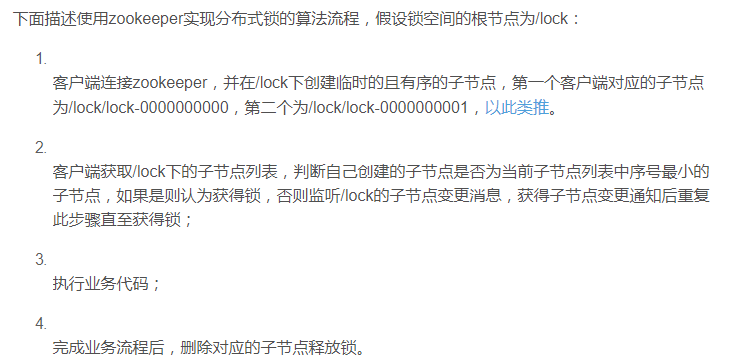
### 结论



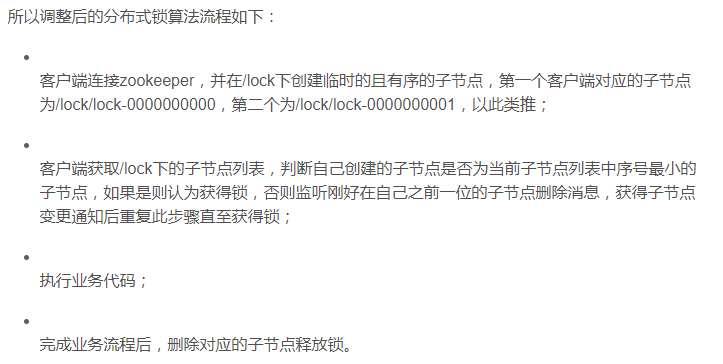
### 参考

http://zhangtielei.com/posts/blog-redlock-reasoning.html

## Zookeeper分布式锁



对于这个算法有个极大的优化点：假如当前有1000个节点在等待锁，如果获得锁的客户端释放锁时，这1000个客户端都会被唤醒，这种情况称为“羊群效应”；在这种羊群效应中，zookeeper需要通知1000个客户端，这会阻塞其他的操作，最好的情况应该只唤醒新的最小节点对应的客户端。应该怎么做呢？在设置事件监听时，每个客户端应该对刚好在它之前的子节点设置事件监听，例如子节点列表为/lock/lock-0000000000、/lock/lock-0000000001、/lock/lock-0000000002，序号为1的客户端监听序号为0的子节点删除消息，序号为2的监听序号为1的子节点删除消息。



### 参考

https://blog.csdn.net/qiangcuo6087/article/details/79067136