## Online-lock 详细设计

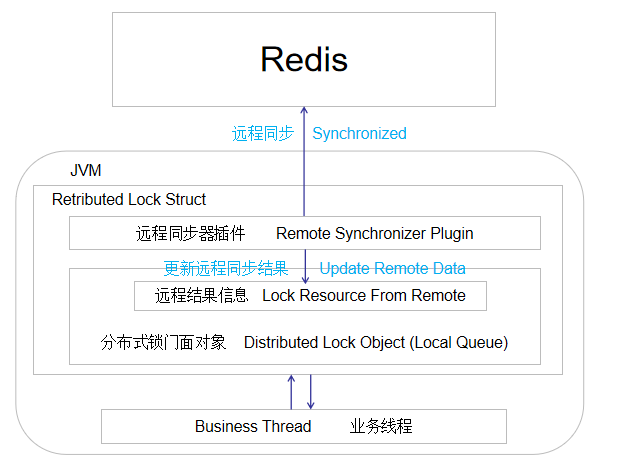
### 一、数据库

无

### 结构设计

该分布式锁，由本地锁管理器（锁门面）和远程同步器两大部件组成。

锁管理器负责将本地请求排队，远程同步器负责与远程服务器同步锁信息（维护锁的申请、保活、释放）。锁管理器和远程同步器之间采用异步执行的方式，他们之间的耦合点主要在于共享部分数据。



### 分部件设计

#### 锁管理器（DistributedLock）

锁管理器维护锁等待队列，由map实现锁键名和等待队列的匹配。

锁等待队列借助JDK的AQS实现，提供本地公平锁和非公平锁（参考ReentrantLock，注意：这里仅能本地公平/非公平）。

###### DistributedLock.Lock lock(String key)方法

阻塞请求锁。

1. 若当前没有其他线程获取到锁，将资源状态标记为等待投食（等待远程同步器），循环检测等待资源状态为已持有。
2. 若有其他线程获取到锁或正在等待投食（等待远程同步器），阻塞当前线程。

正在等待投食的线程，获取到锁后，将资源状态置为进食中，封装已获取到的锁，返回给业务端。

###### DistributedLock.Lock tryLock(String key, long waitMillis)方法

非阻塞请求锁（非阻塞请求锁，天生是非公平的）。循环竞争获取锁，若超过限制时间未获取锁，返回nulll。

###### void initKey(String key, boolean fair)方法

可以通过这个方法事先定义锁键名，后续请求锁时无需再传fair参数

###### DistributedLock.Lock.unlock()方法

释放这把锁，资源状态置为等待餐后收拾。

#### 远程同步器（RemoteSynchronizer）

远程同步器，被设计成一个抽象，实现远程同步器的类需继承该抽象类。抽象类构造函数入参为一个配置类实例，配置类也是一个抽象（RemoteConfigure），该抽象类仅包含一个生成同步器实例的抽象方法。对于单节点的redis，可以使用默认提供的实现类，SimpleRemoteSynchronizer和SimpleRemoteConfigure。

#### 线程

由于远程同步器和锁管理器是异步运行的，所以远程同步器内部，需要有多线程的实现。