实验配置:

语言:Golang 1.10

RPC 开源库: thrift 0.9.1

实验思路:

1. server,client 通过 RPC 通信，server 端提供 lock，unlock 服务
2. server 端维护线程安全的 acquire set，当有新的 acquire 进入时，检测是否有锁。如果已经存在，则加入 set。否则 random 选择 client 进行加锁
3. 尚未扩充 RPC 库，仅仅是使用 set 保证 at most once 语义(请求重复发送被视为一次请求)

实验流程:

1. server start 监听端口

```
[cui:server]$>go run *.go
Running at: localhost:6790
2018/07/05 14:25:22 add new client:false
```

2. client 按照 ID 不同发送 10 个 acquire

```
[cui:client]$>go run *.go
e.g. acquire,5(不同client请求次数)
e.g. release,1(relase clientID)
e.g. status
acquire,10
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
```

注意: 这里使用的 for 循环进行 RPC 调用，并不是真正意义的并行，所以把 clock 加在的 ID=0 的 client 上

3. 查看 server 端维护的线程安全的 client status

```
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
2018/07/05 14:39:53 ClockID:0 State:acquire
status
2018/07/05 14:39:59 {"0":true,"1":false,"2":false,"3":false,"4":false,"5":false,"6":
false,"7":false,"8":false,"9":false}
```

发现只有 client 0 为 true

4. 释放 ID =0 的 lock

```
release,0
2018/07/05 14:40:04 State:release clientID:0
status
2018/07/05 14:40:08 {"0":false,"1":false,"2":false,"3":false,"4":false,"5":false,"6"
:false,"7":false,"8":false,"9":false}
```

全部都为 false，所有 client 都是未加锁状态

5. 模仿 5 个 client 请求 lock

```
acquire,5
2018/07/05 14:40:21 ClockID:2 State:acquire
2018/07/05 14:40:21 ClockID:2 State:acquire
2018/07/05 14:40:21 ClockID:2 State:acquire
2018/07/05 14:40:21 ClockID:2 State:acquire
2018/07/05 14:40:21 ClockID:2 State:acquire
acquire,5
2018/07/05 14:40:28 ClockID:2 State:acquire
2018/07/05 14:40:28 ClockID:2 State:acquire
2018/07/05 14:40:28 ClockID:2 State:acquire
2018/07/05 14:40:28 ClockID:2 State:acquire
2018/07/05 14:40:28 ClockID:2 State:acquire
status
2018/07/05 14:40:34 {"0":false,"1":false,"2":true,"3":false,"4":false,"5":false,"6":
false,"7":false,"8":false,"9":false}
```

因为这几个 ID 都已经存在于 server 端(注意测试重复 acquire 请求情况),server 随机选择 client 加锁。这里是 clientID=2 被加锁。

通过 status 命令查看符合预期