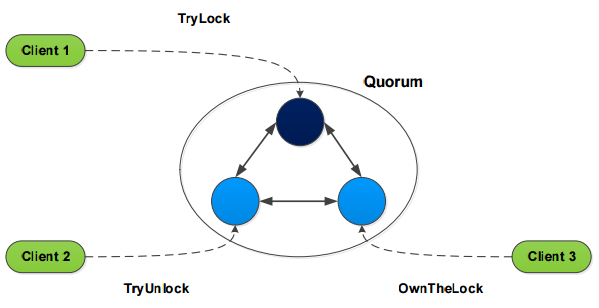
基于Python Socket的分布式锁实现-实验报告

# 实验内容



本实验意图实现如图所示的系统，系统分为三个部分。首先是主服务器(Leader)，在图中用深蓝色表示。其次是从服务器(Follower)，用浅蓝色表示。最后是客户端(Client)，图中用浅绿色表示。

主服务器只有一个，可以应答从服务器的请求，也可以应答客户端请求。主服务器存储并维护一个包含锁名称及状态的字典，锁的状态改变都要通过主服务器的认证和传播才能最终得到修改。当主服务器收到从服务器发来的锁状态修改请求时，主服务器首先根据收到的请求修改本地存储的锁状态，之后将锁状态的改变发送给所有的从服务器，以保证数据一致。

从服务器可以有多个，其在本地存储一个锁的状态表，并且时刻与主服务器上的锁的状态表保持一致。当收到客户端发来的查询请求时，查询本地的表并返回值。当收到客户端的锁状态修改请求时，从服务器将请求直接发送给主服务器处理，并接受处理后的结果返回给客户端。当收到主服务器发来的锁状态修改请求时，则按照请求的内容修改自身的锁状态表。

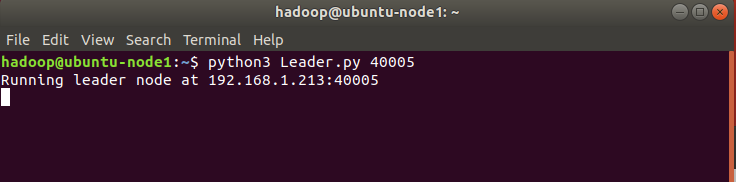
在本次实验中，由于在同一窗口中同时运行对从服务器信息的处理和对客户端信息的处理会导致输出信息的混乱，同时为避免不必要的代码量，我们在运行主服务器的虚拟机再运行一个从服务器端口，以模拟主服务器与客户端的交互。

# 环境配置

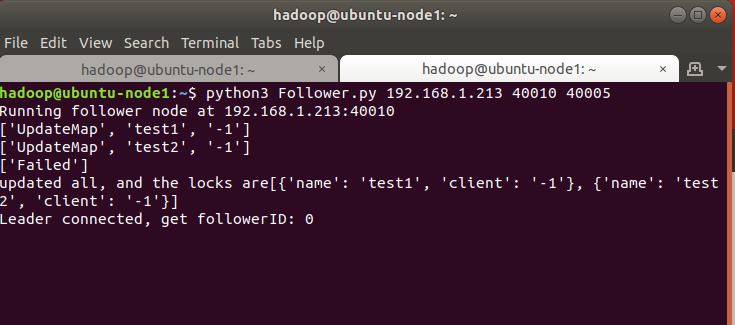
Ubuntu 18.04, Python 3.6.5

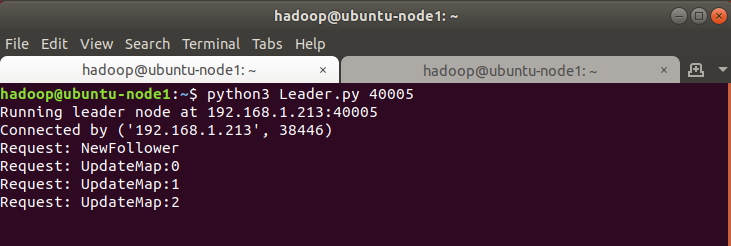
# 实验结果及截图

1. 在虚拟机1上启动主服务

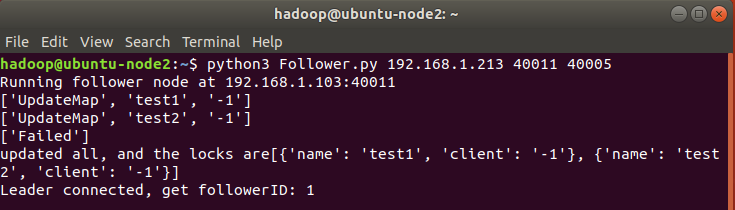


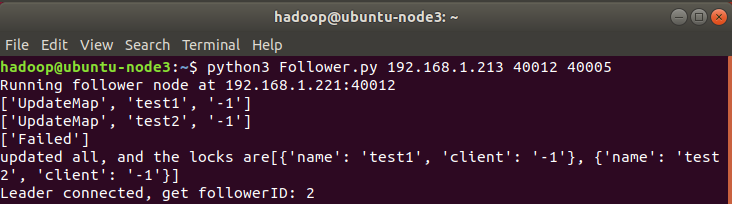
1. 在虚拟机1上启动从服务，同时主服务接受到请求



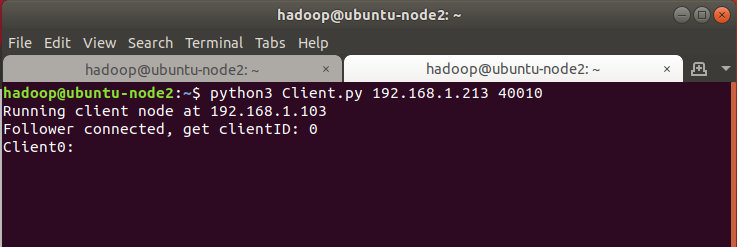


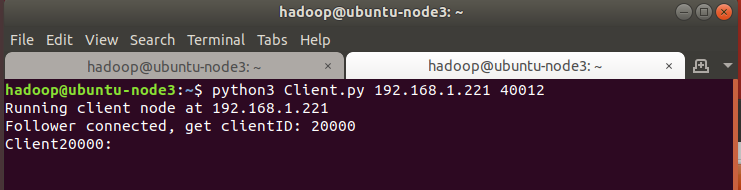
1. 在虚拟机2和3分别启动从服务

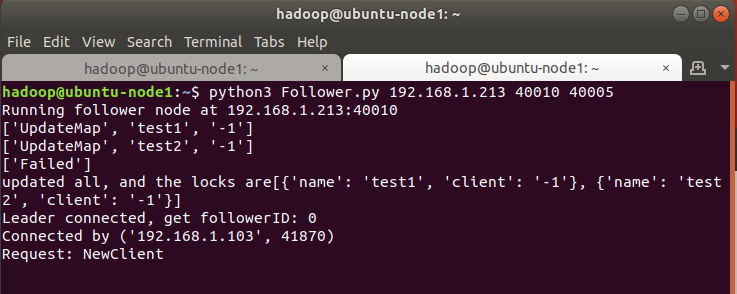




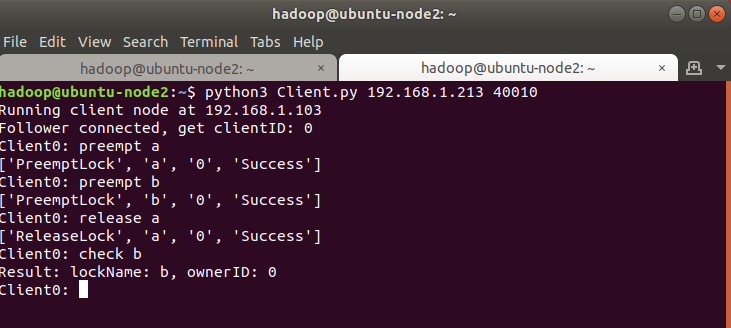
1. 在虚拟机2，3分别启动客户端，连接至从服务0，2，从服务器收到信息

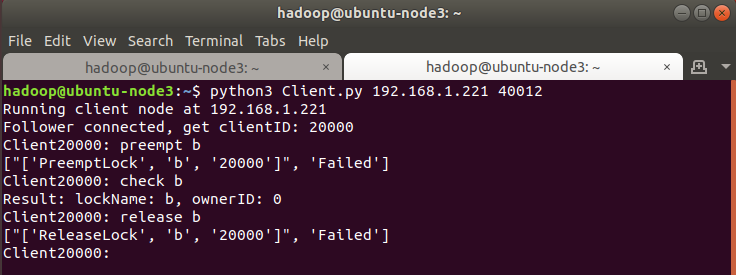




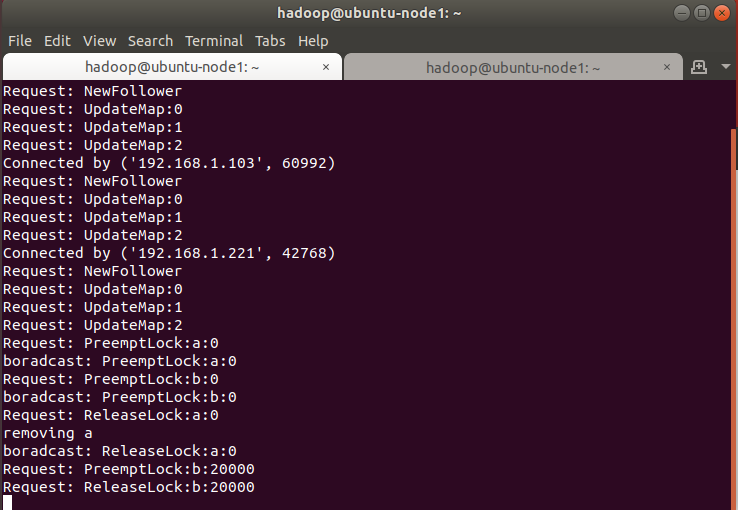


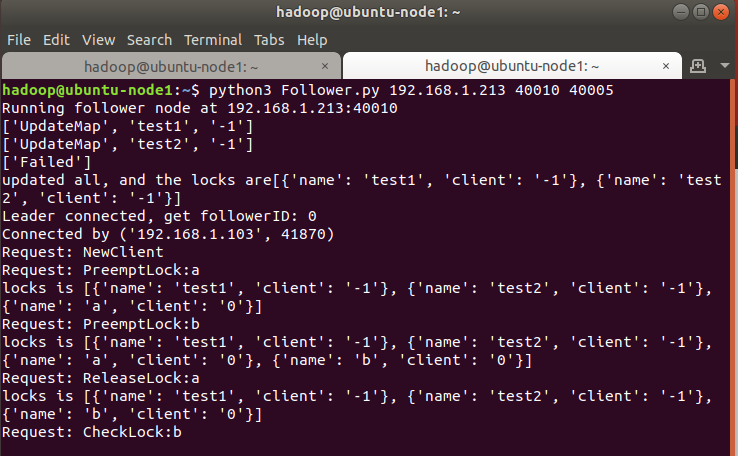
1. 在客户端0和1实验基本的操作

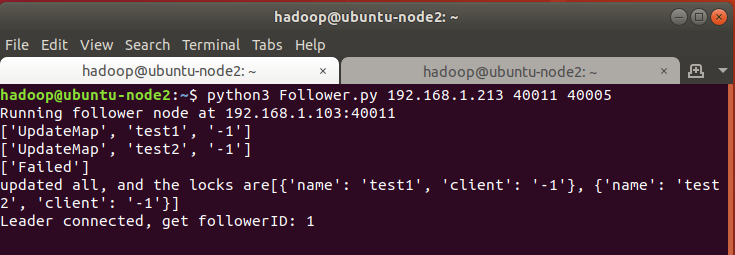


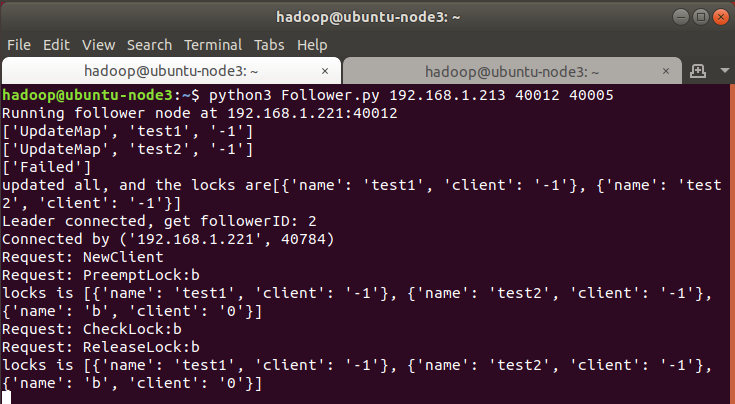


1. 此时，主服务器与从服务器上的信息为

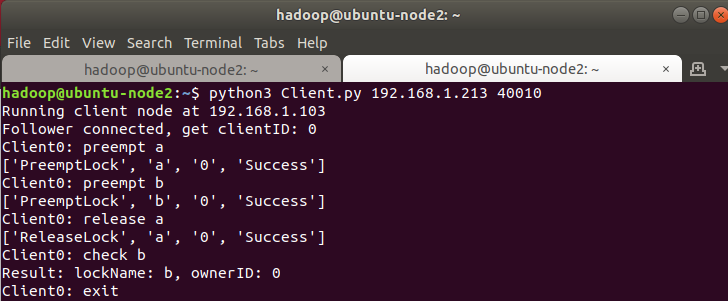








1. 退出客户端



# 实验总结

此实验主要是让我们理解集群中的数据共享与交互，总体代码量不大，主要难点在于进程间的通信与端口的监听。设计出的分布式锁在吞吐量增大，或进程突然终止时可能不一定能保证数据的一致性，此点可能有待改善。