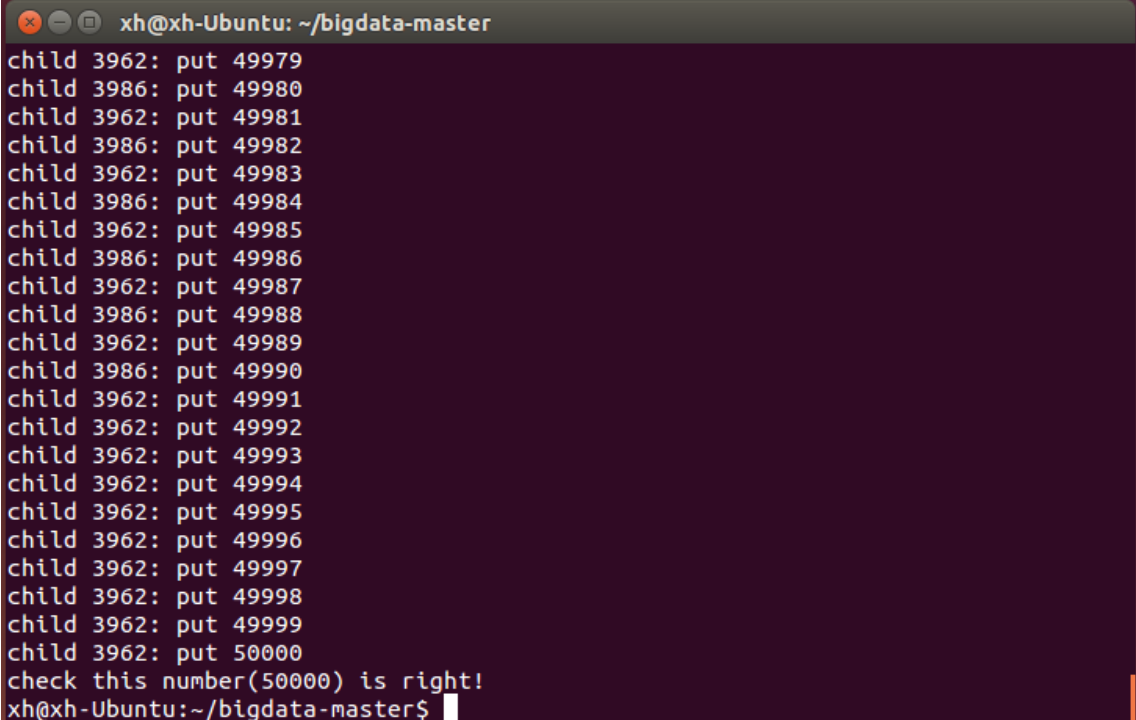


# 实践项目 1：一个支持分布式锁的简单 Consensus 系统设计

---

## 测试 1

- 运行：`./do.sh -c 50 -n 1000`
- 结果：如下图所示



```
xh@xh-Ubuntu: ~/bigdata-master
child 3962: put 49979
child 3986: put 49980
child 3962: put 49981
child 3986: put 49982
child 3962: put 49983
child 3986: put 49984
child 3962: put 49985
child 3986: put 49986
child 3962: put 49987
child 3986: put 49988
child 3962: put 49989
child 3986: put 49990
child 3962: put 49991
child 3962: put 49992
child 3962: put 49993
child 3962: put 49994
child 3962: put 49995
child 3962: put 49996
child 3962: put 49997
child 3962: put 49998
child 3962: put 49999
child 3962: put 50000
check this number(50000) is right!
xh@xh-Ubuntu:~/bigdata-master$
```

- 解释：
  - `-c 50` 代表开启 50 个客户端。由于系统消息队列一般为 16k, 并且当客户端数量过多时, 服务端消息队列回满。这时, 当 Master 向 Follower 同步数据时, Master 节点会阻塞; 当 Follower 先 Master 转发消息时, Follower 也会阻塞。所以, 客户端数目有一定限制。
  - `test.txt` 存储一个数字 (初始为 0), 每个客户端对这个数字+1, 通过这个操作模拟客户端对同一资源的访问。`-n 1000` 代表每个客户端要完成的操作次数。即整个过程中数字被自加  $c*n$  次 ( $c$  个客户端, 每个客户端完成  $n$  次操作), 通过分布式锁来保证互斥访问, 如果分布式锁有效, 最后的数字应该是  $c*n$ 。
  - 控制台输出每次操作由哪个客户端完成。`server.log` 记录 Master 对客户端请求锁是否授予。
  - 最后如果完成时文件中的数字满足要求将会打印一个验证正确的结果。

## 测试 2

### 1. 启动服务端

```
chmod +x ipcrm
```

```
./ipcrm --all=msg
```

```
./server 0 > server.log &
```

```
./server 1&
```

```
./server 2&
```

### 2. 在不同终端启动多个客户端

```
./clent
```

### 3. 在客户端输入命令

```
req <lock name>
```

如果请求锁成功，客户端输出 own this lock

如果请求锁失败，客户端输出 lock is busy.

```
release <lock name>
```

如果成功，客户端输出 lock is idle

如果失败，客户端输出 lock is busy.

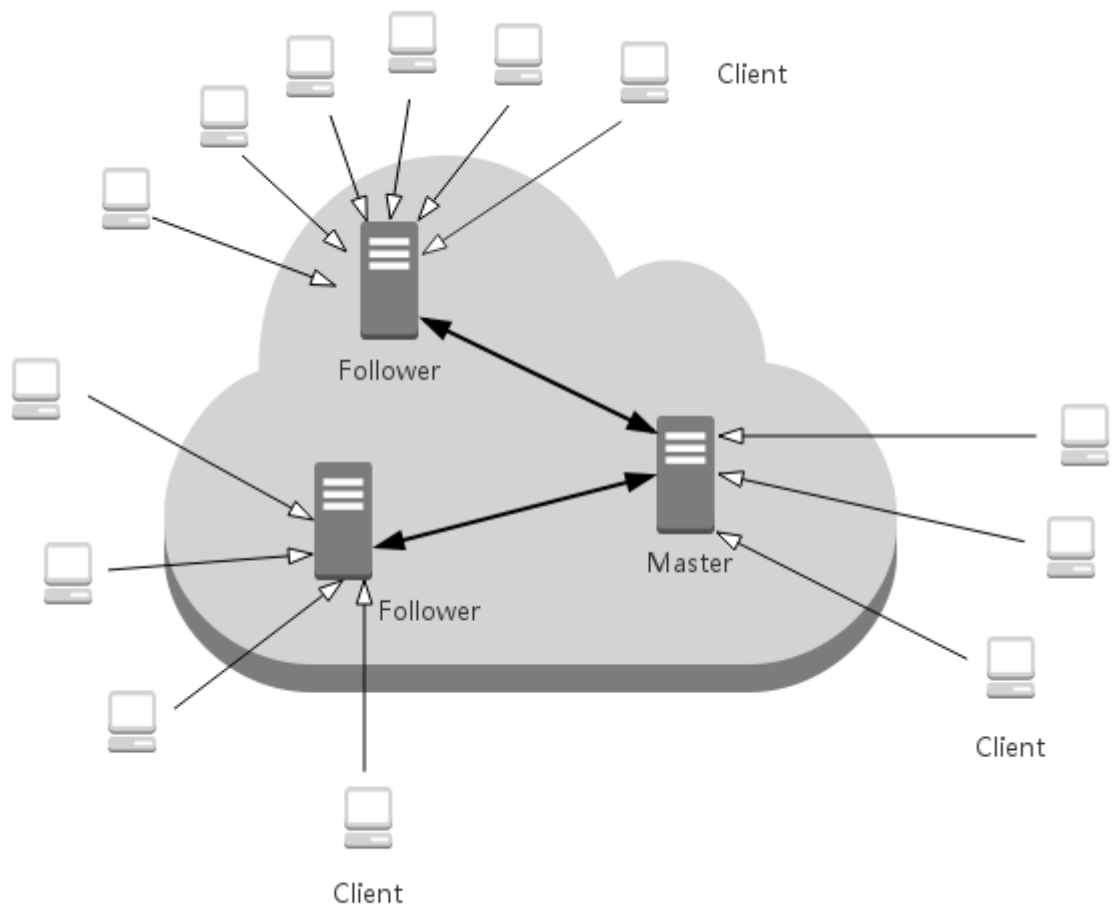
```
inquire <lock name>
```

如果该客户端拥有这个锁，则客户端无输出。

如果客户端不拥有这个锁，客户端输出 lock is busy.

任务说明书里说“任意 Client 均可以查看本 Client 是否是某分布式锁的当前 Owner”，所以 inquire 实现的是查询本 Client 是否拥有该锁，但不说明这个锁是否被其他客户端拥有。

## 系统设计



1. 默认是 3 个服务端进程。server 0 即 Master 服务端。每当有新的客户端加入时采用轮转的方式为其分配服务端。
2. 全部采用 linux 消息队列的方式通信。
3. 客户端可以向对应的服务端发送查询、要求锁，释放锁三种消息类型。
4. 如果是 Fellow 服务端接受要求锁，释放锁的消息，将转发给 Master。相应的有了来自 Master 的回应将转发给相应的客户端。
5. 所有关于锁的记录和分配决策由 Master 负责，每次有更新将向 Follower 推送更新。