## 一、实验要求

设计一个分布式处理系统。一个节点（进程）作为Master,负责节点注册，任务分配；其它多个节点（进程）作为worker，执行客户端发送的请求。系统执行流程为：

* Master首先启动
* Worker节点启动后向Master进行“注册”；Master维护所有已注册节点的信息（比如IP地址和TCP/UDP端口，服务类型等）；
* 客户端提交任务前首先向Master发送“查询”消息，询问将任务提交给哪个worker
* Master根据一定策略（如轮询）从所有Worker中选择一个worker，比如A，将其信息发送给Client
* Worker A完成任务后将结果返回给客户端
* 计算任务为：两个数的+，-，\*，/。数据类型支持32位整数、64位整数、浮点数
* 要求消息编码格式：google protocol buffer
* Worker端要支持多个客户端请求的并发处理
* 测试要求：

Worker数量不少于3个

客户端数量不少于3个

单个客户端发送请求数要大于100万个，或者测试时间不少于10分钟

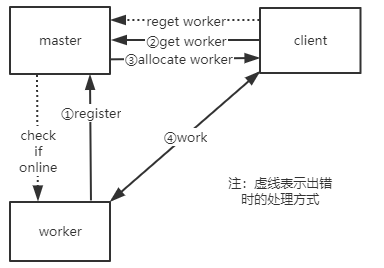
## 二、实现功能

1、基础功能：master对worker信息的管理和分配，client申请worker和交互，worker的注册和并发处理

2、扩展功能：master对worker可用性的复核及注销，client对worker不可用的纠错机制，在用户层隐藏worker的更换操作。

## 三、具体实现

### 总体架构



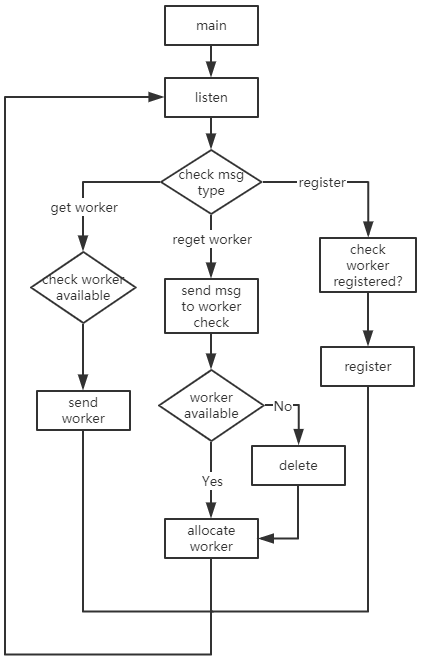
在正常情况下，master先启动，之后worker再启动，并向master发布注册请求。注册成功后两者连接断开，即步骤①。

每次client申请都需要向master申请worker，由master分配worker给client，即步骤②和③。

之后client向worker发送请求，进行交互计算操作，即步骤④

其中master功能尽量为轻负载模式，其并不一直维持worker可用，而是在分配worker出现错误后由client报告，并向master申请分配新的worker，之后由worker复核，最终确定是否原worker不可用，并决定是否注销。即图中虚线所示。

### master（sever）



**主要数据结构**：

IpNode：负责存储worker信息，并以链表的形式连接，其中分配为O(1)复杂度，增删worker为O（n）复杂度。（由于worker数量在个位数级，所以未使用hashmap结构。如果之后想增删在O（1）复杂度下，可以再维护一个hashmap，存储IP节点及对应IpNode地址即可）。

**主要流程：**

master会接收到从worker的注册信息和client的申请worker信息，其中client分为第一次申请及worker不可用后的再申请。

其中需要通过flag位判断信息类型：

如果是注册，则需要对目前维护的worker检查，以防多次添加。

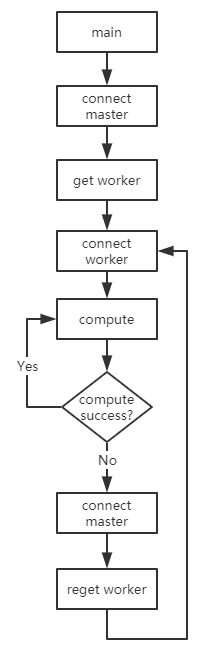
如果是再申请worker，则需要对client发送的worker进行复核，并决定是否将其删除。之后与第一次申请处理机制相同。

如果是第一次申请，则通过轮询，分配节点，此时仅需要指针地址改变即可。

在此过程中，由于维护的worker读写一致性，故不适用并发机制。因此要保证master的轻负载，保证master不成为工作瓶颈。所以在此过程中master只维护worker序列，由client充当实验者，判断worker实时可用性，再由master复核。

而且循环链表很适合增删频繁的结构，虽然增删需要O（n）的复杂度，但在此题worker数目小下不需要额外结构即能保证效率。否则可使用hashmap，通过维护worker的ip与worker指针的对应关系，保证增删在O（1）的效率下完成。

### client

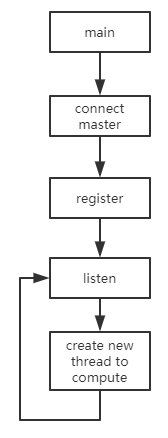


client主要完成与worker的交互计算操作。其中发送数字均先转化为string后再转发，保证并用性。

在启动阶段client首先连接worker，申请worker，之后创建新socket与worker连接，接受到的数据首先根据flag位进行string到指定数字类型的转换，并存储到本地文件中。

如果worker不可用，则client后台自动发送消息给master，申请更换worker，并自行连接。此操作对用户透明，即用户无需感觉到worker的更换，其只要确保运算完成即可。

### worker

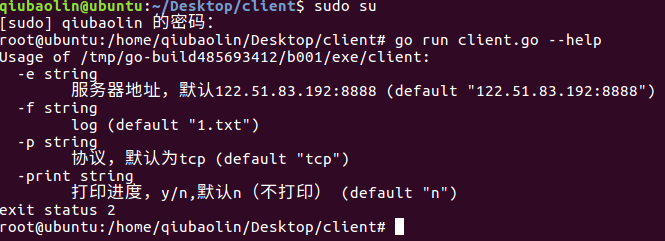


worker端负责与client的计算，其中同一worker可处理多个client的运算请求。即在监听后，每收到一个client的连接请求即创建协程（goroutine），新协程负责处理并发计算操作。

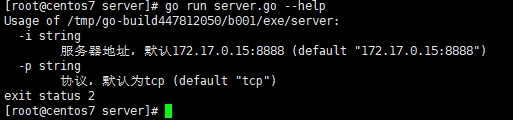
## 四、程序使用及演示

本实验已添加脚本文件，只需要在shell中运行即可。其中worker和master均在服务器后台运行，用户只需要运行client即可。

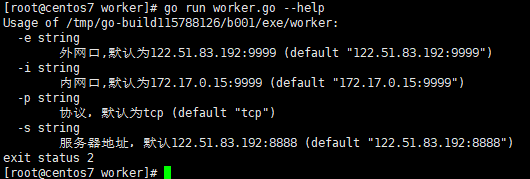
否则，可使用go run client.go –-help查询具体参数要求。（建议在root权限下运行）



client使用示例



server使用示例

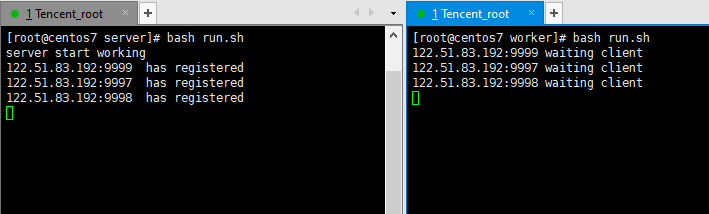


worker使用示例

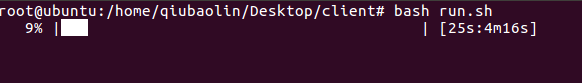
服务器使用的腾讯云，所以client连接需要连接外网IP，server和worker需要监听内网IP。如果是自己主机则不需要区分内外网口。

### 具体实例：

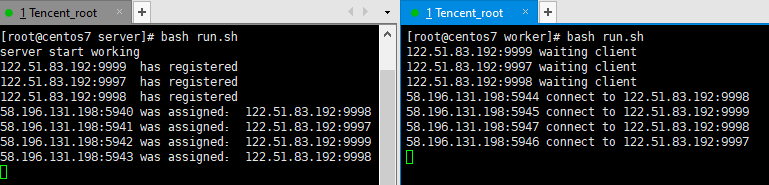
**正常流程**



server启动和worker注册

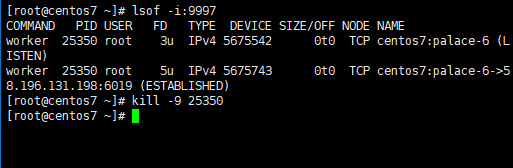


client申请计算及运行初期

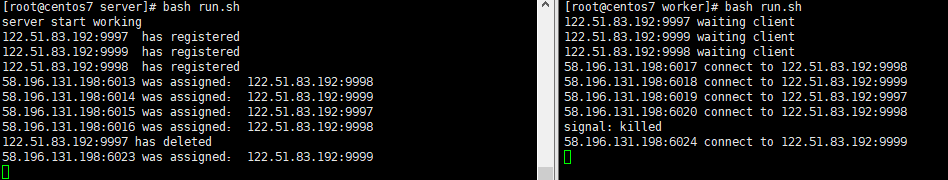


server分配worker及client与worker连接

**worker掉线异常处理**



断开9997端口worker



分配给9997端口的client重新分配到9999端口