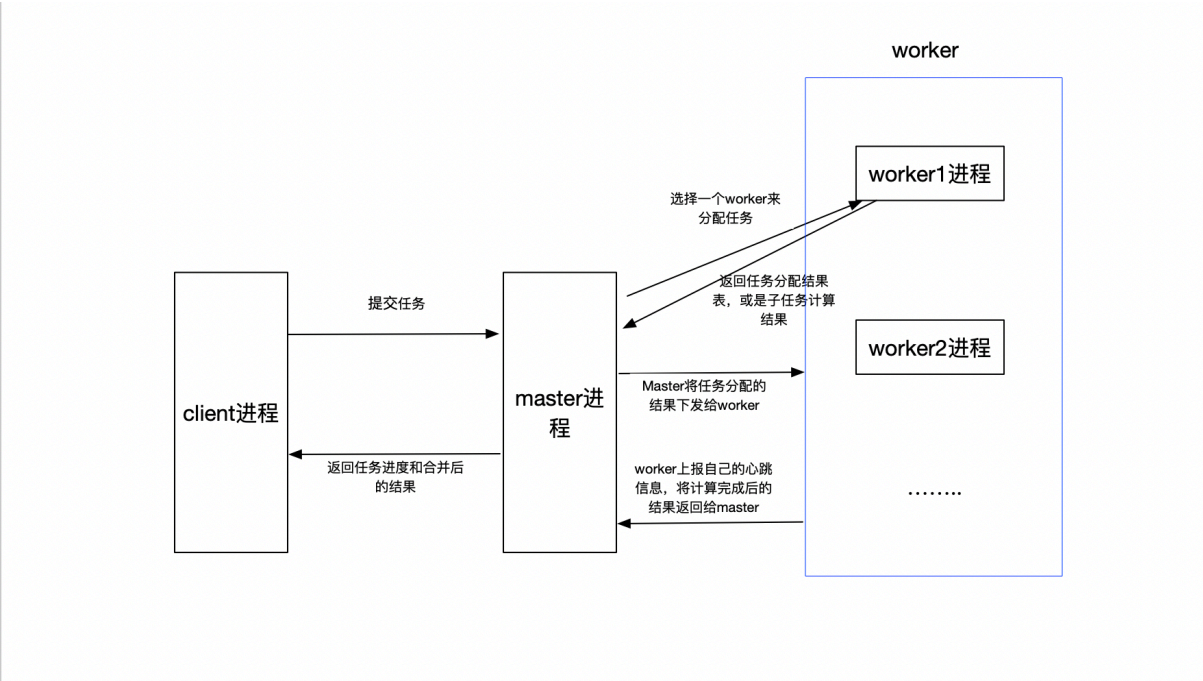
一 总体架构图



说明：

1、worker 上报给 master 的心跳信息内容包括

（1）唯一标识 workerId(默认为本机 ip+一个随机数)，
（2）队列大小 queueSize(队列用来存放新收到的任务);
（3）线程池大小 coreSize(线程池模拟多核机器);
（4）空闲队列大小 freeQueue;
（5）空闲线程的大小 freeCore;
（6）上报时间 time;

2、master 委派哪个 worker 来分配任务，委派算法为

（1）遍历存放 worker 状态信息的 workerStatusMap;
（2）如果某个 worker 有空闲线程，则委派该 worker 来分配任务，选择好之后结束遍历 workerStatusMap;
（3）如果所有的 worker 都没有空闲线程，则将分配任务委派给空闲队列最大，即排对最快的 worker;
（4）如果所有的 worker 都没有空闲线程且队列都被打满，则返回给客户端”资源不足，需要等待”的信息;等到有空闲线程时再进行分配;
（5）master 收到 worker 的子任务结果后，需要检查结果是否正常以及返回 client 任务进度信息，如果结果正常且所有子任务结构都返回，则合并子任务结果为最终结果给 client

3、worker 执行任务

（1）当收到任务，需要进行分配时，分配算法与上面的委派算法相同，计算完成后返回分配列表给 master;

（2）当收到计算子任务列表，依次将子任务列表中的任务加入线程池中运行，如线程池满的需要等待，运行成功后返回结果，若有异常则返回任务的错误标识（错误标识在 task 中自定义，为不再期望区间的值）；
（3）client 上报新的 Task 并等待 master 返回进度信息和最终结果。
当收到最终结果信息后，需要验证最终结果是否正常，如果与错误标识相同则标识任务执行失败，否者打印正常结果

4、通信协议

（1）通信数据的 byte 数组由中 40byte 的 head 部分和内容对象字节数组组成；
（2）head 格式为：flag+进程编号+内容对象数组长度，以上 3 部分下划线字符分隔开，其不满 40byte 部分在后面填充下划线字符，flag 为固定的 5 位字符串，有以下几种分类：

state：worker 状态信息；
task_:client 提交的新任务或 master 下发给 worker 子任务列表；
talbe：master 下发给 worker 新任务以计算子任务分配列表或 worker 返回子任务分配列表；
value：woeker 返回子任务结果或 mater 返回 client 最终结果。

（3）进程编号：默认由机器 Ip+一个四位数的随机数组成；

（4）发送消息时需要先按照上面格式组装 byte 数组并发送，当进程收到信息时，需要先取出前 50byte 的消息头，并解析出 flag、进程编号和内容长度，然后读取对应长度的 byte 并反序列化为对应对象再进行对应的处理步骤。

5、Task 接口定义

接口定义如下，所有新任务需要实现 Task 接口才能提交到集群运行

```
public interface Task<M, R> {
    /**
     * 在worker 中运行的方法
     *
     * @return 返回子任务的结果
     */
    M map();

    /**
     * 在master 中汇集子任务结果的方法
     *
     * @param list
     * @return 返回最终结果
     */
    R reduce(List<M> list);
```

```
    /**
     * @return 获取子任务个数
     */
    int getSubTaskCount();

    /**
     * @return 获取程序出错时的错误标志代码，其值不在子任务和任务的结果当中
     */
    M getErrorFlag();

    /**
     * @return 返回分解后的任务列表
     */
    List<Task> decompose();


    /**
     * @return 返回提交任务客户端的 id
     */
    String getClientId();

}
```

二 运行时序图