1 引言.....2

1.1 编写目的.....2
1.2 背景.....2
1.3 定义.....2
1.4 参考资料.....2

2 程序系统的结构.....3

3 程序 1（标识符）设计说明.....3

3.1 程序描述.....4
3.2 功能.....5
3.3 性能.....6
3.4 输人项.....7
3.5 输出项.....8
3.6 算法.....8
3.7 流程逻辑.....9
3.8 接口.....10
3.9 存储分配.....12
3.10 注释设计.....12
3.11 限制条件.....12
3.12 测试计划.....12
3.13 尚未解决的问题.....13

详细设计说明书

1 引言

1.1 编写目的

本文档对车牌识别系统的概要设计，说明整个软件系统的结构，明确系统各个模块的处理流程和模块交互之间采用的数据结构及协议。文档面向软件的开发和测试人员，软件开发者根据本文档对软件进行进一步的细化设计，然后进行开发工作，软件测试人员依照本文档对系统的功能模块进行测试工作，保障系统能够按照设计要求正确工作。

1.2 背景

1. 软件名称：车牌识别系统
2. 开发者：
3. 用户：交通执法人员、交通保险现场处置人员以及希望在其开发的软件中集成车牌识
4. 运行环境：
   服务端：计算机集群
   客户端：具备图像采集和网络访问能力的任意终端设备

1.3 定义

无

1.4 参考资料

[1]张海藩.软件工程导论[M].2008

[2]JSON-RPC Working Group.JSON-RPC 2.0 Specification[S].2013

[3]（英）SAM NEWMAN 著；崔力强，张骏译.微服务设计[M].2016

2 程序系统的结构

2.1 客户端

序号	功能名称	功能描述
F-1	文件加载	加载链接图片数据库或者路径
F-2	凭证请求	向服务器请求图片下载权限
F-3	文件上传	向服务器上传图片
F-4	任务提交	向服务器提交节点识别任务
F-5	结果展示	向用户展示识别结果

2.2 服务端

序号	功能名称	功能描述
F-1	消息队列	用于完成处理任务的各种调度
F-2	凭证管理	管理凭证的请求
F-3	Docker 管理	用于管理计算节点
F-4	车牌识别	用于进行车牌识别的核心算法

3 程序 1（标识符）设计说明

无

3.1 程序描述

(1). 文件加载

用于加载链接需要识别的图片

(2).凭证请求

服务器处理请求权限

(3)文件上传

用于图片文件上传

(4)任务提交

向服务器提交任务

(5)结果展示

向用户展示识别结果

(6) 消息队列

用于完成处理任务的各种调度

(7) 凭证管理

管理凭证的请求

(8) Docker 管理

用于管理计算节点

(9) 车牌识别

用于车牌识别的核心算法

3.2 功能

(2).凭证请求

向服务器请求图片处理的权限

(3)文件上传

上传选定要处理的图片

(4)任务提交

向服务器提交任务,进入计算队列

(5)结果展示

向用户展示车牌的识别结果

(6) 消息队列

提交: 提交任务
查询: 查询人物信息
删除: 用于删除任务

(7) 凭证管理

处理来自客户端的凭证请求的确认

(8) Docker 管理

系统状态: 返回状态信息
节点添加: 添加计算节点
节点删除: 删除计算节点

(9) 车牌识别

返回车牌识别结果

3.3 性能

(1). 文件上传

无

(2).凭证请求

无

(3)文件上传

无

(4)任务提交

无

(5)结果展示

无

(6) 消息队列

无

(7) 凭证管理

无

(8) Docker 管理

无

(9) 车牌识别

无

3.4 输人项

(1). 文件加载

详细内容见<<数据结构说明>>

(2).凭证请求

详细内容见<<数据结构说明>>

(3)文件上传

详见<<数据结构说明>>

(4)任务提交

见<<数据结构说明>>

(5)结果展示

见<<数据结构说明>>

(6) 消息队列

见<<数据结构说明>>

(7) 凭证管理

见<<数据结构说明>>

(8) Docker 管理

见<<数据结构说明>>

(9) 车牌识别

见<<数据结构说明>>

3.5 输出项

(1). 文件加载

(2).凭证请求

(3)文件上传

(4)任务提交

(5)结果展示

(6) 消息队列

(7) 凭证管理

(8) Docker 管理

(9) 车牌识别

3.6 算法

(1). 文件加载

(2).凭证请求

HMAC-SHA1 签名算法

(3)文件上传

通俗文件上传算法

(4)任务提交

动态优先实时调度算法

(5)结果展示

(6) 消息队列

Kafka 节点运行 Apache Kafka 消息队列程序，负责存放和分发由 H

TTP 服务节点添加到队列中的任务信息。

(7) 凭证管理

HMAC-SHA1 签名算法

(8) Docker 管理

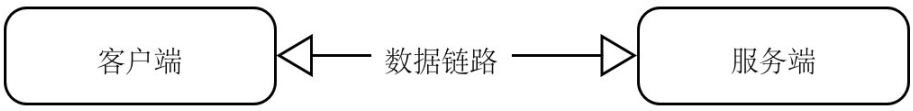
动态规划

(9) 车牌识别

神经网络,sobel,模板匹配

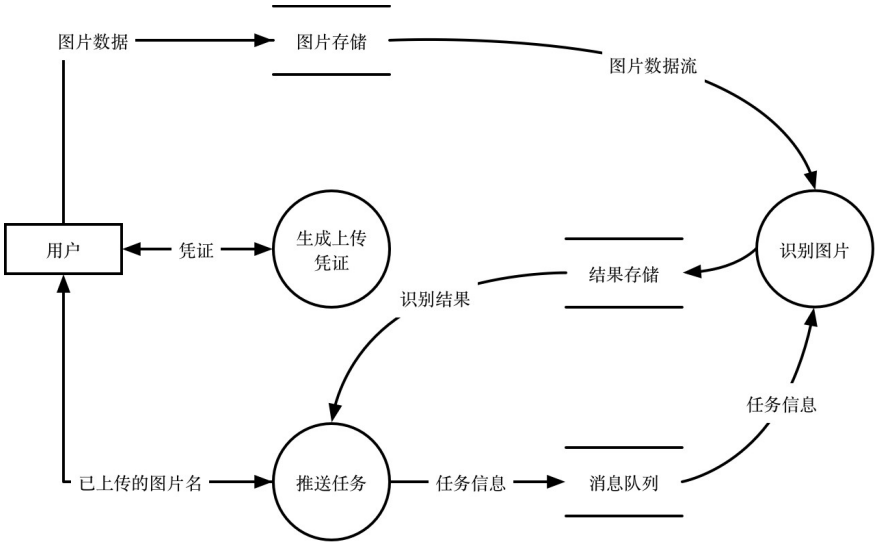
3.7 流程逻辑

3.7.1 基本设计概念和处理流程

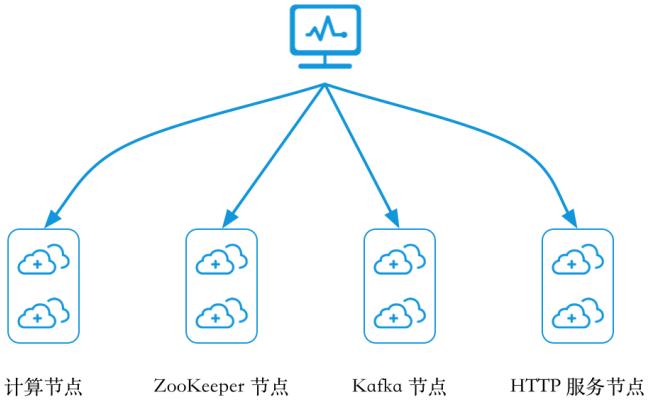


系统对外暴露的结构为 C/S（客户端-服务端）结构，系统的外部结构图如下：
服务端部分为计算机集群，分为以下五类节点：

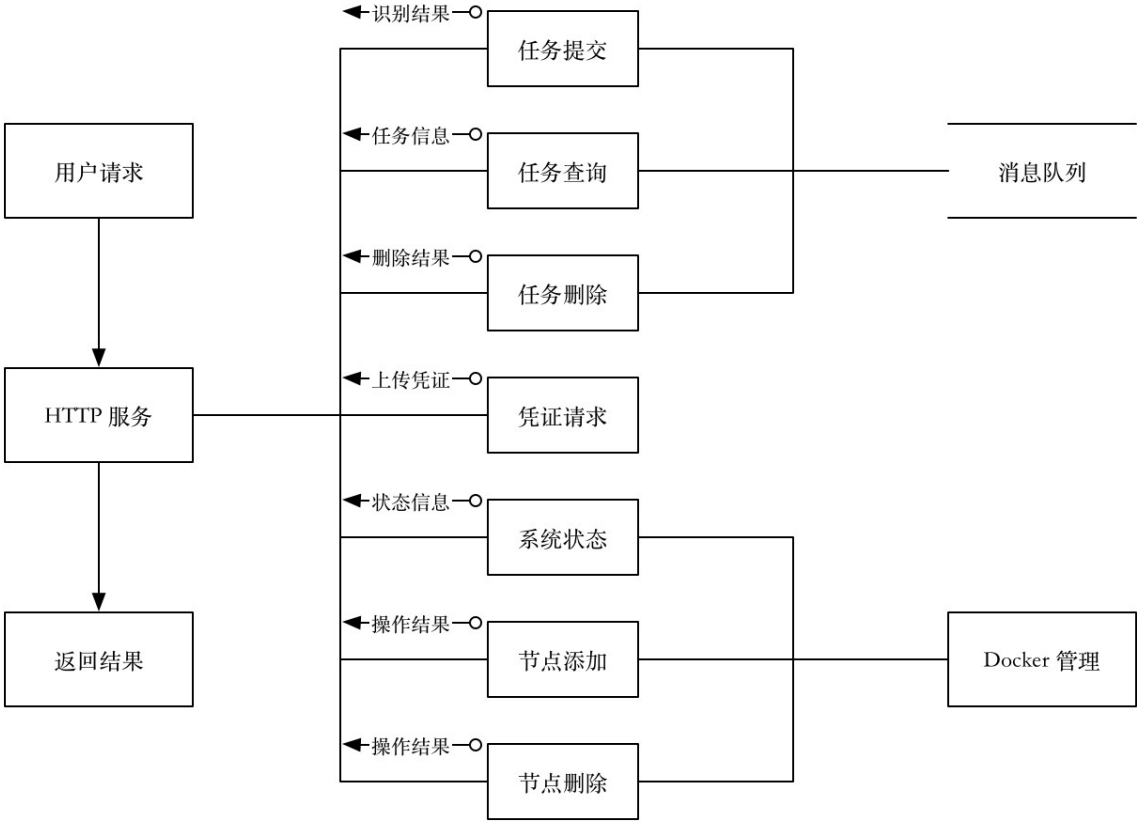
- 管理节点
- 计算节点
- ZooKeeper 服务节点
- Kafka 服务节点
- HTTP 服务节点



3.7.2 系统的总体结构如下：

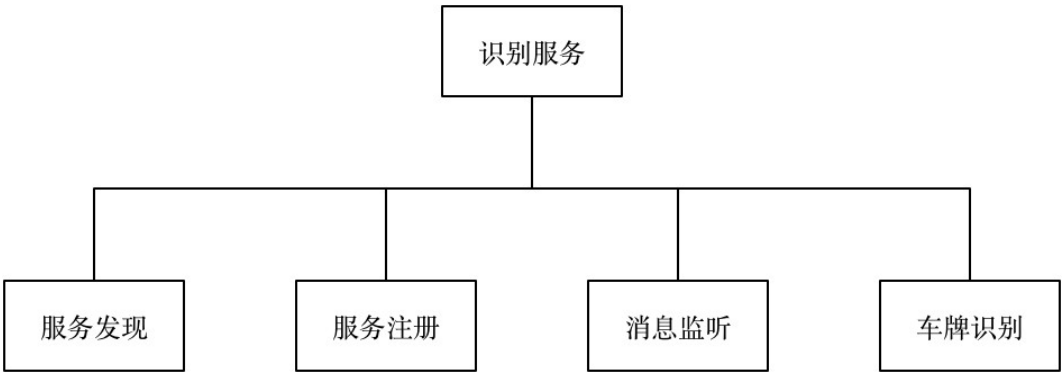


其中 HTTP 负责提供整个系统的对外服务，其结构如下：



ZooKeeper 节点运行 Apache ZooKeeper 分布式协调程序，在本系统中，该节点负责服务的注册与发现功能，其保存着系统中所有运行中的节点信息、识别中的任务信息，以协调整个系统的工作。

Kafka 节点运行 Apache Kafka 消息队列程序，负责存放和分发由 HTTP 服务节点添加到队



列中的任务信息。

识别节点负责从 Kafka 消息队列中取出任务进行识别，其结构如下：

3.8 接口

用图的形式说明本程序所隶属的上一层模块及隶属于本程序的下一层模块、子程序，说明参数赋值和调用方式，说明与本程序相直接关联的数据结构（数据库、数据文卷）。

3.9 存储分配

本程序采用随机存储分配

3.10 注释设计

说明准备在本程序中安排的注释，如：本程序

a. 加在模块首部的注释；
b. 加在各分枝点处的注释；
c. 对各变量的功能、范围、缺省条件等所加的注释；
d. 对使用的逻辑所加的注释等等。

3.11 限制条件

算法识别率只有 80%

3.12 测试计划

测试工件为四个阶段：单元测试、组装测试、确认测试、系统测试。

单元测试：采用白盒法和黑盒法相结合的方法，对于逻辑结构复杂的模块采用白盒法，对于以输入、输出为主的模块采用黑盒法测试，以提高测试的效率。

组装测试：自底向上的增式测试。

确认测试：由用户参与按需求规格说明书验收。

系统测试：采用人工测试方法。

(1) 系统环境模块测本测试是为了检测系统环境模块，数据连接是否正确，数据能否正确，并进行仔细核对。

(2) ZooKeeper 节点运行 Apache ZooKeeper 分布式协调程序，在本系统中，该节点负责服务的注册与发现功能，其保存着系统中所有运行中的节点信息、识别中的任务信息，以协调整个系统的工作。本模块是整个系统线上运行的基础

(3) 识别模块,是系统算法运行的核心功能,要在规定的正确率内运行

3.13 尚未解决的问题

说明在本程序的设计中尚未解决而设计者认为在软件完成之前应解决的问题。

.....