智汇盒软件架构设计方案			
文件编号：		当前版本：	V1.0
页数：	页	发布日期：	2019-08-10
文件状态：初版■　　评审通过□　　正式发布□			

作者	凡超		
评审人员			
审核人员			
批准			

成都臻识科技发展有限公司 Vision-Zenith Tech Co., Ltd.
地址：中国四川省成都市高新区交子大道 300 号誉峰国际中心 M3 栋 22 楼 5-8 号，610095
电话：+86 028 87931722　　传真：+86 028 87931722　　邮箱：service@vzenith.com
网址：www.vzenith.com　　全国统一服务热线：400 885 2262

修 订 详 细 记 录				
版本	修订内容详述	编制/修订人	批准人	修订日期

目录

第1章 概述

1.1 目的

针对智汇盒提供软件技术可行性方案和产品开发人员提供技术支撑，本文档在满足产品需要的基础上给出了应用层整体框架设计，及主要注意技术点

1.2 适用范围

1. 产品、研发及测试人员适用

第2章 主要职责

2.1 执行职责

本程序实现由研发部相关人实现并自测。

2.2 编制、修改、评审、审批、更新职责

本程序文件由研发部负责人编制、修改，组织由相关部门共同评审，并经部门经理和质量保证工程师审核，研发部经理签字审批认可方能生效。

第3章 任务概述

3.1 任务目标概述(详见《智汇盒 RD 规格书》)

第一阶段(本地私有云)：

系统管理：用户管理/日志管理/系统配置/系统升级

系统升级：当前版本/插件式升级

智能配置：支持图片流的车牌/人脸识别，区域计数

实时监控：视频可视化

第二阶段(集群，普通相机智能化)

支持 IPC 与抓拍机的连接，行业通用协议支持

支持视频流识别（车牌，人脸，区域计数）

支持历史记录查询

支持 docker，便于客户二次开发

支持 10w 底库

支持集群化管理

第三阶段(服务各行业 saas 客户)

支持算法按应用场景选配，算法插件化应用，应用商店统一管理

支持用户私有算法开发

可扩展管理 NVR

免插件视频转码，支持 PC、手机多终端的访问

3.2 运行环境

Hi3559AV100/Hi3519AV100　硬件平台

3.3 需求概述

考虑到智汇盒产品的功能较多，采用分阶段迭代开发。需求概述见《业务计划书》，此处略

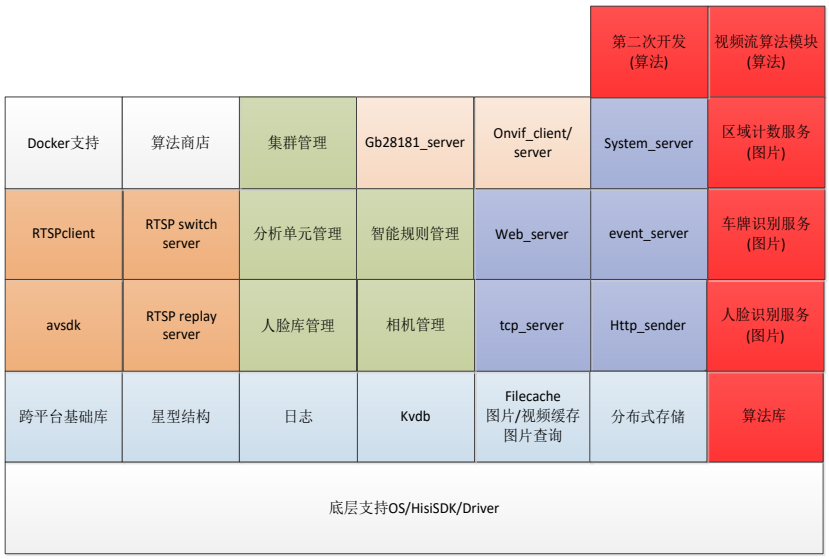
3.4 条件与限制

第4章 总体设计

本章主要介绍系统的模块划分与系统主要时序

4.1 模块介绍

描述整个系统将涉及的模块



4.2 功能需求与对应模块

4.2.1 系统管理：主要由 system_server 负责，主要完成以下功能：

系统管理	二级	三级	描述	阶段
	用户管理	添加删除用户		1
		用户密码修改		
		用户权限修改	管理员、操作员、观察员	
	平台接入管理(做到接口兼容，不用实现该功能)	asccessid 和 asccesskey 的生成和管理	分配 accessid 和 asccesskey，客户端通过它们来接入服务，公有云接入的简化版本	1
	日志管理	日志分类查询、下载日志	查询条件：时间段、日志类型（业务、系统、异常、操作） 日志列表：类型、记录时间、详细信息	1
		时间段选择		
		打包导出日志		

	系统配置	时间配置(ntp/手动)	支持 ntp 同步、手动设置时间（同步本地时间，手动输入）、支持时区设置。 集群模式下，子节点的时间跟 mater 同步	1
		手动抓拍路径设置	本地 pc 端存放目录	1
		存储设置	存储状态查询、格式化、 相机存储策略配置	2
		设备维护	立即重启、 按计划配置重启、 恢复出厂设置(完全恢复、部分恢复)	1
	网络配置	网口选择	eth0、eth1 双网口配置，内外网场景	1
		IP 地址，掩码，网关，DNS 配置，DHCP	开启 dhcp 的情况，需求外部工具搜索，2 阶段	1
		Http 推送配置		2
		FTP 上传配置		
		上云 PDNS		
		安全设置		
		VPN		
		断网重传		
	系统升级	版本列表显示		1
		升级文件上传		
		选择升级的分析单元（全选、单选）	集群升级/单点升级。第 2 阶段	
		支持插件式升级	人脸识别插件升级、 车牌识别插件升级、 区域计数插件升级、 软件系统升级	
		升级状态	升级中，升级成功、 升级失败	

4.2.2 智能应用：主要涉及模块：智能规则管理、人脸底库管理、区域计数服务、人脸识别服务、车牌识别服务、算法模块、Webserver(提供统一的 api 接入)

智能分析	二级	三级	描述	阶段
	人脸库管理(接口兼容公有云)	人脸库管理：增删查改	创建库、修改、查询、删除	1
		特征库存在重建可能	算法特征向量变化时，支持特征库的重新生成	
		单张/批量导入		
		单张删除		
		导入失败列表		
	人脸抓拍识别视频流	配置相机抓拍参数	算法参数、区域等	2
		配置相机的识别参数	关联人脸库、阈值	
	人脸识别图片流(接口兼容公有云)			1
	区域计数视频流	配置相机计数区域		2
	区域计数图片流(api)	参数中提供检测位置		1
	车牌车型识别视频流	配置相机的车牌识别参数	算法参数、区域、虚拟线圈	2
	车牌车型识别图片流(接口兼容公有云)	接收图片，返回结果		1

4.2.3 数据查询功能：主要涉及模块：Rtsp 录像回放模块、filecache

数据查询	二级	三级	描述	阶段
	历史告警查询	条件： 1.相机 ip 筛选 2.时间段 3.车牌结构化 4.人脸结构化		2
	图片查询	同上		
	智能查询	1：1 比对		
		1：N 比对	选择底库图片对比，把比对结果呈现出来	
		以图搜图	选择本地图片/选择库图片搜索对比	
	视频回放	选择相机、设定时间段查询回放		

4.2.4　实时监控：主要涉及相机管理模块、rtspclient、rtsp 转发服务、tcp_server、http_sender、ftpclient

实时监控	二级	三级	描述	阶段
	相机管理树	摄像机分组管理显示	登录主节点时，可现实所有的分组 登录单节点时，只呈现节点上的相机	1
		摄像机名称		
		相机状态显示	(在线/不在线)	
		相机预览图像	在预览区域选择空闲窗口预览图像	
	实时预览	1.2.4.6.9.16 多风格预览		1
		可全屏显示切换		
		视频轮训设置		
		单窗口设置	可设置预览子码流\主码流	
		手动抓拍		
	人脸抓拍结构化	性别、年龄、帽子、口罩、评分等		2
		人头		
	报警信息	抓拍时间		
		报警图片显示		
		报警时间		
		相似度		
		查看详情		
	车辆抓拍结构化	车牌号、车型、车牌宽度、车辆信息、车牌颜色、车辆颜色		

4.2.5　资源管理：主要涉及集群管理模块、分析单元管理模块、相机管理模块

资源管理	二级	三级	描述	阶段
	集群管理	创建/删除/修改集群		2
		集群名称		
		添加/删除分析单元至集群	通过 ip 地址添加分析单元到集群	
		手动设置集群的主备单元	通过指定 ip 地址配置	
	分析单元管理	分析单元自动发现		1
		分析单元参数配置	名称、IP、端口、DHCP 开关、子网掩码、网关、首选/备选 DNS 服务器	

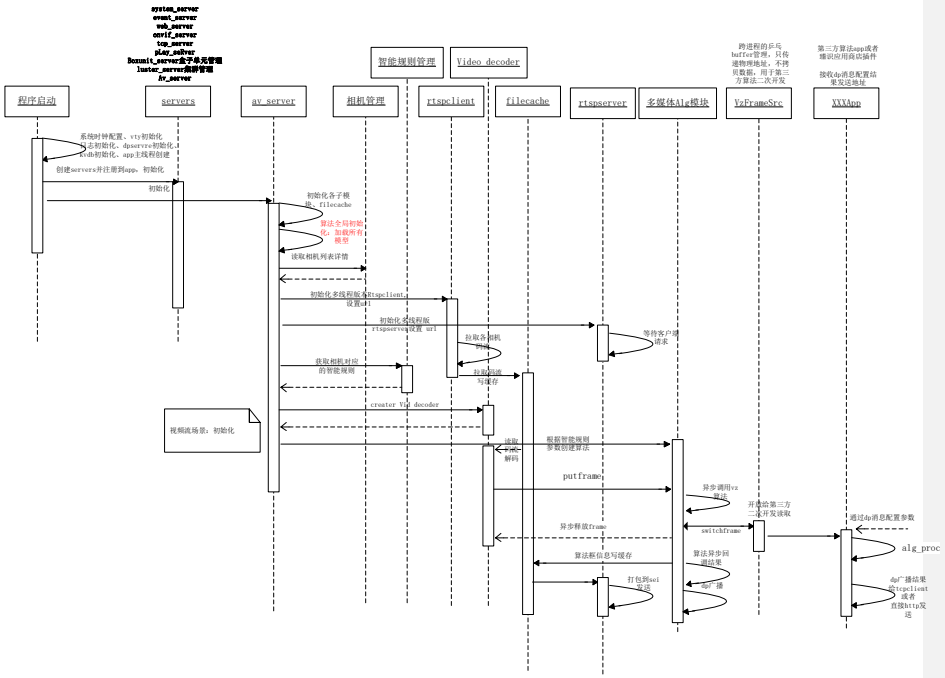
		分析单元性能查看	算力评估、存储、CPU 占用率、状态、温度	
	相机管理	相机分组管理	新建分组，修改删除分组	1
		添加相机至相机列表	添加时配置相机参数，接入协议(rtsp,onvif,gb28181)	
		相机添加(移除)至分组	一个相机只在一个组	
		删除相机	自动从组中删除	
		修改相机参数		

4.2.6 算法商店：主要涉及算法商店模块

算法商店	二级	三级	描述	阶段
	算法 App 管理	通过 web 查询商店算法列表	算法使用 api 可在线提供	3
		通过授权 license 下载算法安装		
		通过 license 管理授权(试用/无时限)		
		可查看/卸载已安装 app		

4.3 系统关键时序图

4.3.1 系统启动主要时序图及视频流时序

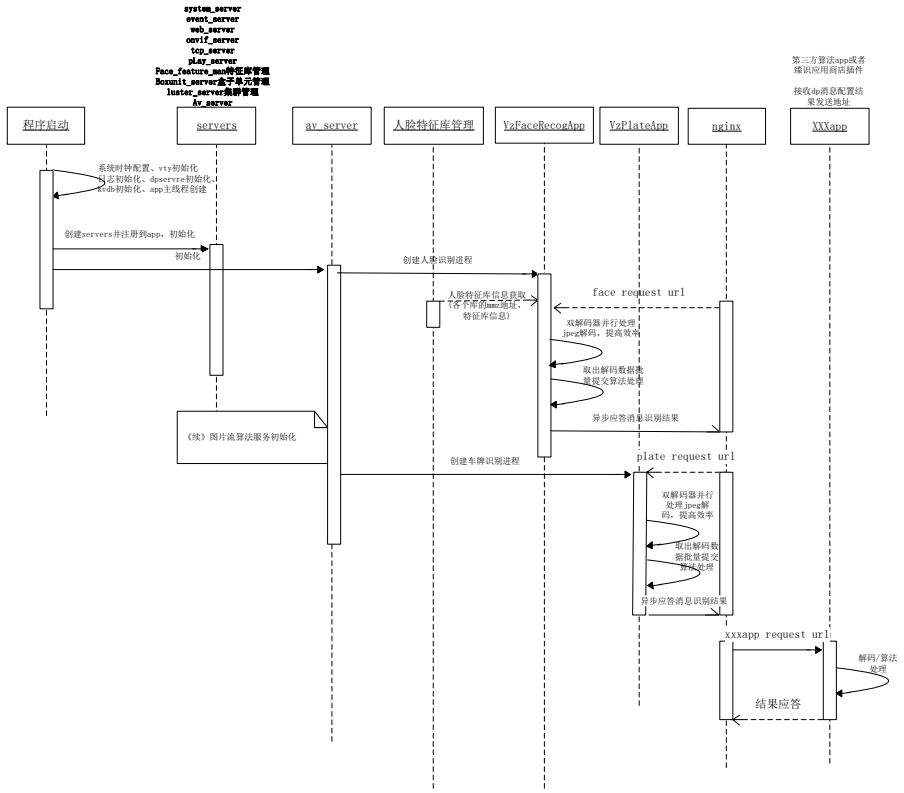


1. 算法全局初始化：加载所有模型，提高程序运行的整体并行性，加快启动速度。
2. 实现时，要考虑耗时模块的并行度，减少系统整体启动时间。如模型加载，底库加载等
3. avserver 读取相机列表信息及相机对应的参数配置信息
4. 创建多线程版本的 rtspclient 和 rtsp 转发 server，filecache 将支持码流多通道跨进程通信，支持多个读者。(如果外部没有其他进程需要码流，可暂用 mediacache 来代替 filecache)
5. 创建解码器，每个相机都对应通道号，通道号同解码器绑定。
6. 根据相机智能规则创建对应算法。
7. 解码后码流地址给算法和进程间通信的(第三方/臻识)算法插件(采用零拷贝方式)。
8. 算法处理结束，回调结果广播。算法插件/第三方根据实际场景自定义结果的处理方式。
9. 算法插件的参数配置，支持两种方式：WebAPI 和 web 页面。如果采用页面配置，那么页面呈现上需要实现一个合适框架来做插件的配置页。原理上是可实现的，放到后面阶段再设计。

批注 [f1]: 耗时操作，提高系统并行度

批注 [f2]: 算法插件的参数配置可在 web 上呈现出来

4.3.2 图片流时序图



1、图片流的处理采用三个独立的 app 来实现(facerecogapp/platerecogapp/areacountapp)

2、每个进程都采用双 decoder 和批量提交算法的策略，提高并行度和处理效率

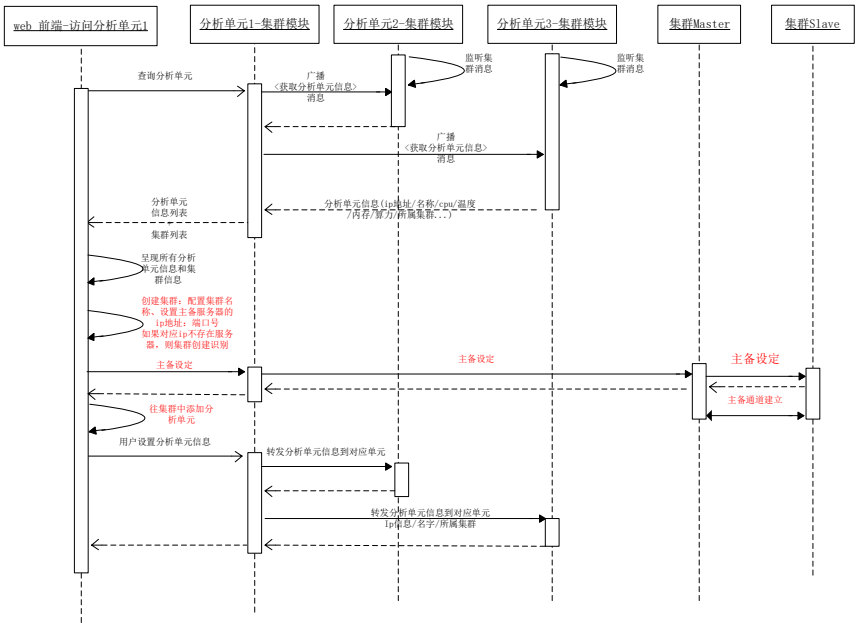
3、使用 Nginx 做 web 服务器，Nginx 的高并发、http 代理、和海量的插件也将给后续的开发带来更多的便捷。

4、每个 app 采用异步应答方式，以支持解码和算法的高并发处理

4.3.3 创建集群时序图

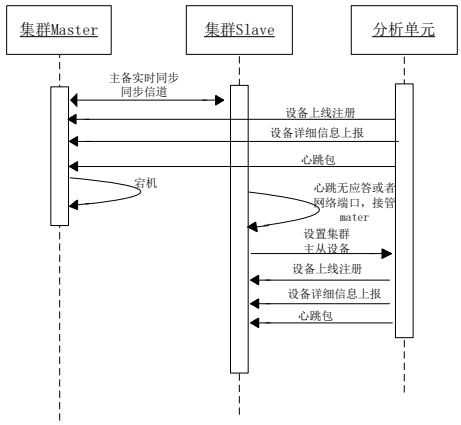
集群管理采用主-备模式管理，分析单元就是一个计算单元。设计上主设备和备份设备可以运行在分析盒上也可以运行在普通 pc 上。创建集群时，用户需要设置主备设备的 ip 地址和端口号。

批注 [f3]: 主备模式

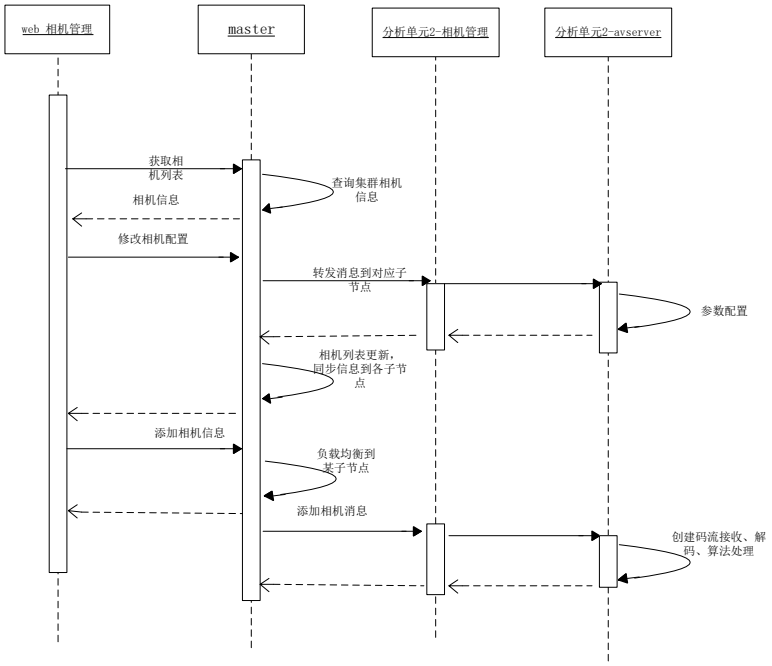


4.3.4 主-备、分析单元上线时序图.

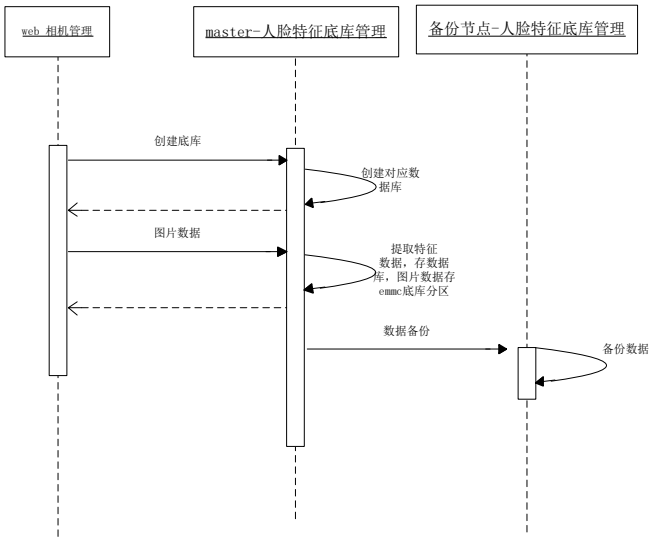
当主设备宕机，备份设备成为主设备。



4.3.5 相机管理时序

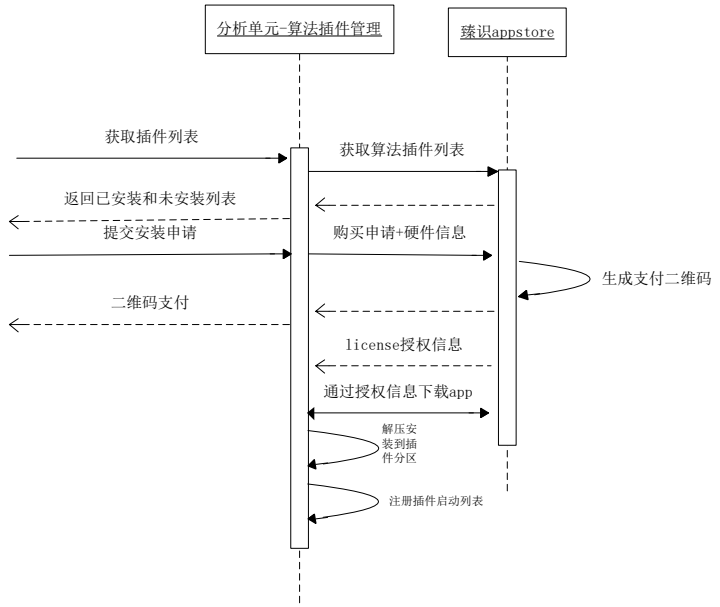

web 相机管理
master
分析单元2-相机管理
分析单元2-avserver
获取相机列表
查询集群相机信息
相机信息
修改相机配置
转发消息到对应子节点
参数配置
相机列表更新，同步信息到各子节点
添加相机信息
负载均衡到某子节点
添加相机消息
创建码流接收、解码、算法处理

4.3.6 人脸底库创建时序

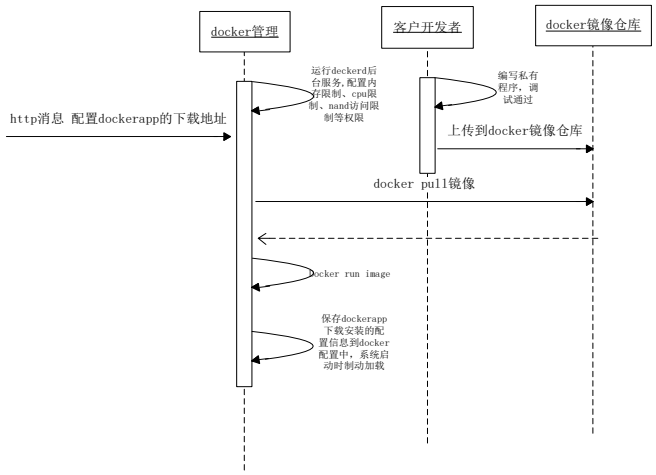


单机方案的人脸底库的数据限制在 10w，每增加一个节点底库可增加 2w,总数据不超过 20w。这主要是考虑到集群情况下，数据的备份和同步带来较大的性能开销。

4.3.7 算法插件安装启动时序

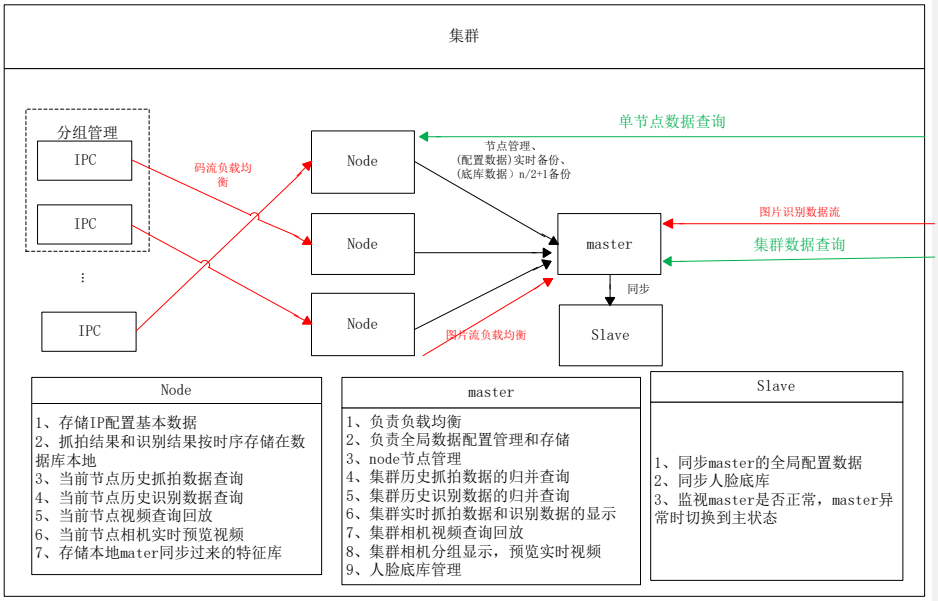


4.3.8 第三方 docker 用户的使用流程



4.3.9 集群框图

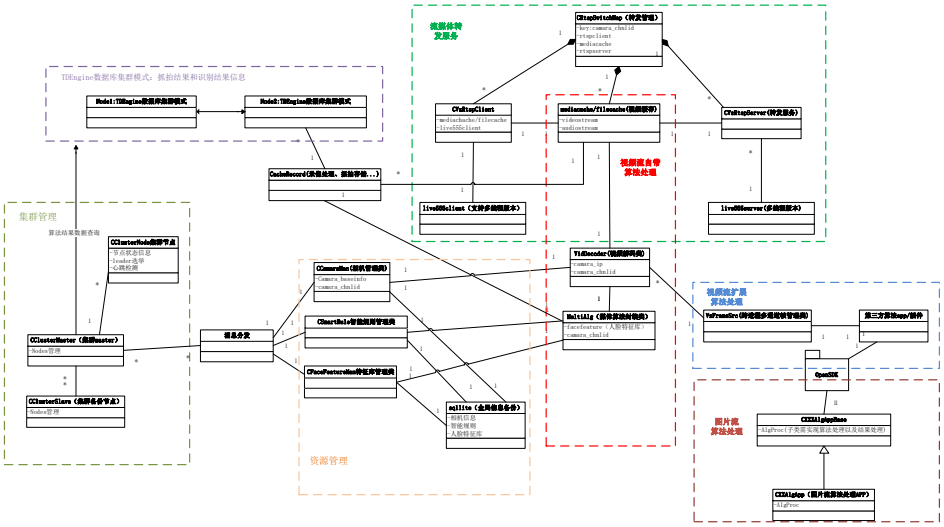
特点：自动负载均衡、支持统一入口数据查询与预览



第5章 关键类图

5.1 新增关键类图

主要涉及：集群管理、全局数据备份、算法输出结构化数据分布式存储和查询、相机管理、智能规则管理、特征数据管理、rtspclient、rtsp 转发服务、视频流解码及算法、图片流算法 app、第三方开发/算法插件：



第6章 接口设计

6.1 外部接口

【与本文档范围之外的接口，如用户界面、软件接口与硬件接口】

6.1.1 网页接口

前端接口主要采用 appliction/json 方式进行交互，主要交互接口 URL 定义如下

http://www.domain.com/recognition/snapshot.php

http://www.domain.com/recognition/bbjson.php

http://www.domain.com/recognition/avsjon.php

http://www.domain.com/recognition/sysjon.php

http://www.domain.com/recognition/evtjon.php

http://www.domain.com/recognition/stpjon.php

http://www.domain.com/recognition/httpjon.php

http://www.domain.com/recognition/tcpjon.php

http://www.domain.com/recognition/vb.htm?xxx (get)

6.1.2 HTTP 接口

HTTP 支持用户自定义配置，不同的 URL 资源对应不同的数据推送功能

6.1.2 TCP 接口

TCP 主要采用 JSON 消息格式进行通信，主要数据头和数据格式如下：

0	1	2	3	4	5	6	7	标尺
'V'	'Z'	包类型	包序号	包序号	包序号	包序号	包数据长度	数据头
Data	Data	Data	Data	Data	Data	Data	Data	数据

- 字节位 0、 ASCII 字符’V’，整数 86。
- 字节位 1、 ASCII 字符’Z’，整数 90.
- 字节位 2、 Data 数据类型
- 0X00 数据包。
- 0X01 心跳包，数据长度为 0， DATA 为空
- 字节位 3、 包序号，包序列号为递增的数值，用于对应请求命令与返回命令；例如发送一个请求命令到服务器端，服务器端在返回结果时，会将请求命令中的包序列号填充到返回数据包的包序列号中，便于客户端这边将返回结果与请求命令进行对应。如果不存在对应问题，则设置为 0 即可。如果编号到 255，则从 0 开始重新编号。
- 字节位 4 - 7、一共四位，代表接下来的数据长度。这个数据是网络字节序，在接收的时候务必调用 htol 这样的函数将网络字节序转换成主机字节序。数据的长度不要大于 1024 * 1024 = 1MB。服务器将不会接收超过 1MB 大小的数据，同样服务器也不会发送超过 1MB 大小的数据包。理论上，除了心跳包之外，其它所有数据的长度都不会是 0。
- DATA 数据位、根据前面的包头，有不同的数据。

6.2 内部通信接口

内部通信主要定义了系统内部各个子模块之间相互通信的主要数据结构，用来合理的在各个模块之间进行数据交互。

6.2.2

通信消息接口（系统内部采用星型结构消息进行交互，在单进程中可以将各个模块之间的交互进行解耦）：

```
// 发送一个广播消息，消息数据和内容由自己指定
// method: 消息描述
// session_id: 会话 id，用户自定义
// data: 消息内容
// data_size: 消息内容长度（Byte）
```

```
// return 成功，true；失败，false
virtual bool SendDpMessage(const std::string method, uint32 session_id,
                           const char *data, uint32 data_size) = 0;

// 发送一个星形结构请求，请求的结果通过指针 res_buffer 返回，
// 该接口为阻塞调用，直到收到回复消息或者超时时返回
// method：消息描述
// session_id：会话 id，用户自定义
// data：请求消息内容
// data_size：消息内容长度（Byte）
// res_buffer：输出参数，由 Request 端提供内存，Replay 端填写回复消息内容
// timeout_millisecond：超时时常（ms）
// return 成功，true；失败，false
virtual bool SendDpRequest(const std::string method, uint32 session_id,
                           const char *data, uint32 data_size,
                           DpBuffer *res_buffer, uint32 timeout) = 0;

// 回复一个星形结构消息，收到 DP 请求后，直接填写回复内容到请求
// DpMessage 中的 res_buffer，res_buffer 内存由 request 端提供
// dp_msg：dp 消息指针，即 DP Request 收到的 dp 消息指针
// return 成功，true；失败，false
virtual bool SendDpReply(DpMessage::Ptr dp_msg) = 0;
```

第7章 数据结构设计

必填

给出主要的数据结构设计，在智能相机领域，特别要给出关于识别结果的结构体定义以及每一个相关字段的描述。

7.1 逻辑结构设计

【给出本系统内主要的数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系】

7.2 物理结构设计

【给出本系统内主要的数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系（索引、设备、存储区域）、设计考虑和保密条件】

7.3 数据结构与程序的关系

【说明主要数据结构与访问这些数据结构的形式】

第8章 运行设计

（必填）

重点描述模块的启动依赖关系，初始数据架构、初始环境准备的功能

8.1 运行条件

*【此节主要描述保证本模块正常运行所需要的条件，以及如何设计，保证模块正常运转】*运行控制

【其它功能模块对本功能模块的调用方法】

调用实例

8.2 运行资源

内存使用预分配

名称	内存占用(MB)	所属类别	大小
Linux system+vzapp	1.5GB	系统及应用程序	2G
第三方 app 和插件预留	512MB		
MMZ	1984MB	VDEC ALG	2G
DSP	64MB		
总计			4G

第9章 核心业务流程详细设计

（必填）

9.1 核心业务流程概述

主要描述核心业务的流程图，整个处理流程过程中的关键流程。整个流程存在的必要性，以及如何保证流程

9.2 核心业务分模块分析

核心业务流程所经过的各个模块对核心业务处理的细节性描述，重点描述如何保证核心业务的细节性性能

第10章 出错处理设计

这个层面需要从大的输入层面来进行讨论，需要尽量将所有的错误全部都列出来完。核心的设计都处于这个部分。一切模块内部细节性的协议相关的错误，不需要写到这里。

10.1 出错输出信息

【请说明每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法】

10.2 出错处理对策

【如设置后备、性能降级、恢复及再启动等】

10.3 冲突处理对策

【请给出各资源与功能的冲突矩阵表】

第11章 安全设计

【系统安全与潜在危险的预防措施，包含如何保证数据的正确性及有效性；系统中存在哪些不安全因素（数据/系统的权限控制、异常处理、操作系统漏洞）；安全设计的方案及解决】

第12章 文件结构

【在此处给出该模块的目录与文件结构设计，以工程结构为主】

第13章 维护设计

【说明为方便维护工作的设施，如维护模块等。说明为了系统维护的方便而在程序内部设计中增加的一些专门用于系统的检查与维护的检测点和专用模块。如 debug 信息，错误记录，关机记录等模块。

可从如下方面介绍：a，内容描述； b，资源需求；c，设计流程】

第14章 测试设计

必填

【总体设计的主要测试要点和测试要求】

第15章 子模块设计分述

与驱动这边，需要将 DeviceSDK 所有的接口列举出来

多媒体需要将主要的流程和算法的接口列举出来

15.1 AVS 语言转换模块

15.1.1功能与性能

该模块用于从多语言数据库中预加载所需要的内容到内存中，如车型，车标，车款以及车牌颜色等。预加载的原因是以空间换时间，多语言文件使用的配置方式，查找效率低而 OSD 叠加对速度要求很高。

15.1.2输入与输出

输入：语言配置文件名，默认 en-us.language 路径/tmp/app/html/language/

15.1.3数据结构

【模块所涉及的数据结构，消息类型和枚举类型等】

15.1.4算法与流程

【模块所选用的算法，

详细描述模块实现的算法，可采用：

1. *标准流程图；*
2. *PDL 语言；*
3. *N—S 图；*
4. *PAD；*
5. *判定表等描述算法的图表】*

15.1.5对外接口

15.1.5.1 int LangTool::Init(std::string filename=" en-us.language");

【描述】

初始化语言转换模块，预加载所需内容到内存中，使用 map，加快查找速度。

【语法】

int LangTool::Init(std::string filename=" en-us.language");

【参数】

参数名称	描述	输入/输出

Filename	输入当前选择语言文件	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述
-1	文件不存在或打开失败
-2	文件读取失败
-3	预加载失败

15.1.5.2 int LangTool::Exit();

【描述】

清理预分配所占资源。

【语法】

Int LangTool::Exit();

【参数】

参数名称	描述	输入/输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述

15.1.5.3 int LangTool::Update();

【描述】

语言环境改变或其他方式改变，更新对应的资源。

【语法】

Int LangTool::Update ();

【参数】

参数名称	描述	输入/输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述

15.1.5.4 std::string LangTool::getTranslatedPalate(const uint8_t * license=NULL);

【描述】

转换车牌识别结果，主要用于无牌车的转换，如果需要其他类型的转换，以重载实现。

例如 无车牌的情况，简体中文显示 “无”，繁体中文显示“無”，英语显示“NONE”

【语法】

std::string LangTool::getTranslatedPalate(const uint8_t * license=NULL);

【参数】

参数名称	描述	输入/输出
license	默认空值即可，根据当前语言转换无牌车	输入

【返回值】

返回值	描述
Std::string	返回转换的结果，当前语言无效，返回 NONE

【错误码】

错误码	描述

15.1.5.5 int LangTool::SetDefaultEmptyPalate(std::string str=”NONE”);

【描述】

设置语言文件无效时默认返回的无车牌转换结果。

【语法】

int LangTool::SetDefaultEmptyPalate(std::string str=”NONE”);

【参数】

参数名称	描述	输入/输出
Str	语言文件无效时，返回无车牌的默认值	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述

15.1.5.6 std::string LangTool::GetTranslatedColor(std::string color);

【描述】

获取根据当前语言转换后的颜色值，用于算法输出结果。

【语法】

std::string LangTool::GetTranslatedColor(std::string color);

【参数】

参数名称	描述	输入/输出
Color	英文颜色单词，无需关心大小写，处理时均转换为小写。 如 Black black bLack	输入

【返回值】

返回值	描述
Std::string	颜色字符串，不支持的颜色时返回默认值

【错误码】

错误码	描述

15.1.5.7 std::string LangTool::GetTranslatedColor(int color);

【描述】

获取根据当前语言转换后的颜色值，用于算法输出结果。重载

【语法】

std::string LangTool::GetTranslatedColor(int color);

【参数】

参数名称	描述	输入/输出
Color	输入算法支持的颜色索引值见下方 VZ_PLATE_COLOR	输入

【返回值】

返回值	描述
Std::string	颜色字符串，不支持的颜色时返回默认值

【错误码】

错误码	描述

算法支持的车牌颜色

```
//车牌颜色
typedef enum
{
    LC_UNKNOWN = 0,        //未知
    LC_BLUE,               //蓝色
    LC_YELLOW,             //黄色
    LC_WHITE,              //白色
    LC_BLACK,              //黑色
    LC_GREEN,              //绿色
    }VZ_PLATE_COLOR;
```

15.1.5.8 int LangTool::SetDefaultColor(std::string color =“unknown”);

【描述】

设置当前语言无效或不支持的类型时返回的颜色值。

【语法】

int LangTool::SetDefaultColor(std::string color =“unknown”);

【参数】

参数名称	描述	输入/输出
Color	设置默认的颜色名称	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述

15.1.5.9 std::string LangTool::GetTranslatedPixel();

【描述】

根据当前的语言设置，将文字”像素”转换为对应的文字，返回”像素” 2 字的转换结果。

【语法】

std::string LangTool::GetTranslatedPixel();

【参数】

参数名称	描述	输入/输出

【返回值】

返回值	描述
Std::string	像素 2 字的字符串

【错误码】

错误码	描述

15.1.5.10 std::string LangTool::GetTranslatedPixel(int x0, int y0, int x1, int y1);

【描述】

重载，根据当前的语言设置，将文字”像素”转换为对应的文字，并将点的坐标转换为像素值

【语法】

std::string LangTool::GetTranslatedPixel(int x0, int y0, int x1, int y1);

【参数】

参数名称	描述	输入/输出
x0	起点坐标 x	输入
y0	起点坐标 y	输入
x1	终点坐标 x	输入
y1	终点坐标 y	输入

【返回值】

返回值	描述
Std::string	像素值+”像素” 如”220 像素” 或“Pixel 220”

【错误码】

错误码	描述

15.1.5.11 std::string LangTool::GetTranslatedPixel(const TH_RECT & rect);

【描述】

重载，根据当前的语言设置，将文字”像素”转换为对应的文字，并将矩形转换为像素值

【语法】

std::string LangTool::GetTranslatedPixel(const TH_RECT & rect);

【参数】

参数名称	描述	输入/输出
Rect	算法定义的矩形区域参见 TH_REC 定义	输入

【返回值】

返回值	描述
Std::string	像素值+”像素” 如”220 像素” 或“Pixel 220”

【错误码】

错误码	描述

```
typedef struct
{
    int left;
    int top;
    int right;
    int bottom;
    } TH_RECT;
```

15.1.5.12 std::string LangTool::GetTranslatedAuthenticity(int isFake);

【描述】

根据当前语言设置，将“真”“伪”转换为对应的文字显示。

【语法】

std::string LangTool::GetTranslatedAuthenticity(int isFake);

【参数】

参数名称	描述	输入/输出
isFake	是否为假车牌	输入

【返回值】

返回值	描述
Std::string	返回“真”或“伪”对应的文字显示，若当前语言无效，返回空

【错误码】

错误码	描述

15.1.5.13 std::string LangTool::GetTranslatedCarType(std::string type);

【描述】

根据当前语言设置，将车型文字转换为对应的文字显示。

【语法】

std::string LangTool::GetTranslatedCarType(std::string type);

【参数】

参数名称	描述	输入/输出
Type	英文描述车型字符串	输入

【返回值】

返回值	描述
Std::string	车型字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.14 std::string LangTool::GetTranslatedCarType(int index);

【描述】

根据算法给出的车型索引值及当前的语言设置，转换为对应的字符串。

【语法】

std::string LangTool::GetTranslatedCarType(int index);

【参数】

参数名称	描述	输入/输出
Index	算法给出的车型索引值	输入

【返回值】

返回值	描述
Std::string	车型字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.15 std::string LangTool::GetTranslatedCarLogo(std::string type);

【描述】

根据当前语言设置，将车标文字转换为对应的文字显示。

【语法】

std::string LangTool::GetTranslatedCarLogo(std::string logo);

【参数】

参数名称	描述	输入/输出
Logo	英文描述车标字符串	输入

【返回值】

返回值	描述
Std::string	车标字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.16 std::string LangTool::GetTranslatedCarLogo(int index);

【描述】

重载 根据算法给出的车标索引值及当前的语言设置，转换为对应的字符串。

【语法】

std::string LangTool::GetTranslatedCarLogo (int index);

【参数】

参数名称	描述	输入/输出
Index	算法给出的车标索引值	输入

【返回值】

返回值	描述
Std::string	车标字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.17 std::string LangTool::GetTranslatedCarModel(std::string type);

【描述】

根据当前语言设置，将车款文字转换为对应的文字显示。

【语法】

std::string LangTool::GetTranslatedCarModel(std::string model);

【参数】

参数名称	描述	输入/输出
Model	英文描述车款字符串	输入

【返回值】

返回值	描述
Std::string	车款字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.18 std::string LangTool::GetTranslatedCarModel(int index);

【描述】

重载 根据算法给出的车款索引值及当前的语言设置，转换为对应的字符串。

【语法】

std::string LangTool::GetTranslatedCarModel(int index);

【参数】

参数名称	描述	输入/输出
Index	算法给出的车款索引值	输入

【返回值】

返回值	描述
Std::string	车款字符串，若当前语言无效或查找失败返回空

【错误码】

错误码	描述

15.1.5.19 int fxxx_init;

【描述】

详细描述接口的功能及此接口使用时的注意事项，如函数调用时是否阻塞，函数是否可重入，函数返回值是否必须处理等。

【语法】

```
int FXXX_init();
```

【参数】

参数名称	描述	输入/输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参考错误码。

【错误码】

错误码	描述

15.1.6存储分配

【简述本功能涉及到的内存与外存的分配方式】

15.1.7限制条件

15.1.8测试要点

【给出该功能的主要测试要点和测试要求，作为单元测试的依据】

第16章 设计中的遗留问题

1、视频免插件播放功能，暂定为 html5+hls 技术来解决，效果还待验证。

2、100w 级大库的情况下，库采用切片存储，分布式计数来解决识别效率。

附录A 附录

附录B 图、表目录