模型管理平台设计文档

1. 概述

模型管理平台对现有的推理模型实施统一管理，并提供接口对用户图像进行模型检测。

1.1. 目标

在有限的资源下对不同的推理需求进行高效管理。

1.2. 背景

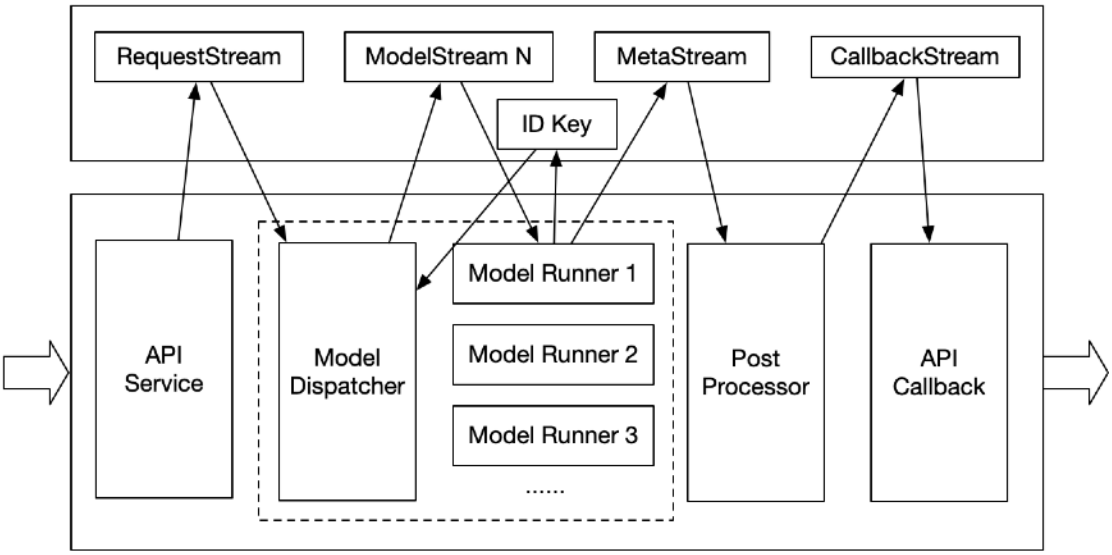
目前各种检测、分类、分割模型较多，但是推理服务器上的硬件资源却非常有限，无法同时加载并推理所有模型，所以我们需要一个管理平台高效的管理所有的模型，根据硬件资源来动态加载、销毁推理模型，做到推理需求与硬件资源的解偶合。

2. 系统架构

我们采用微服务的形式将系统的各个模块组织起来，每个模块独立运行在各自的Docker容器内，模块之间用REDIS请求来进行数据交互。

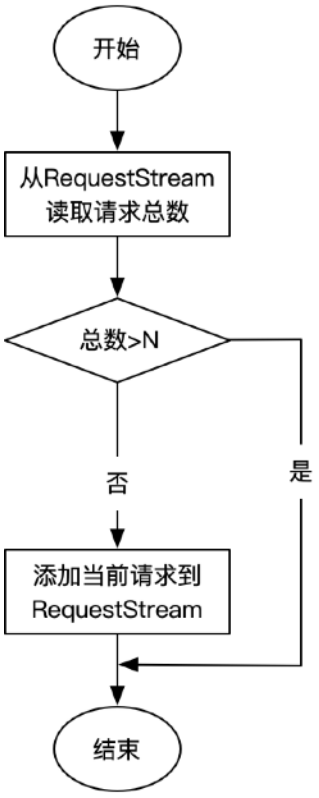
2.1. 模块划分

该管理平台一共包括5个模块：API Service、Model Dispatcher、Model Runner、Post Processor和API Callback。



2.1.1.API Service

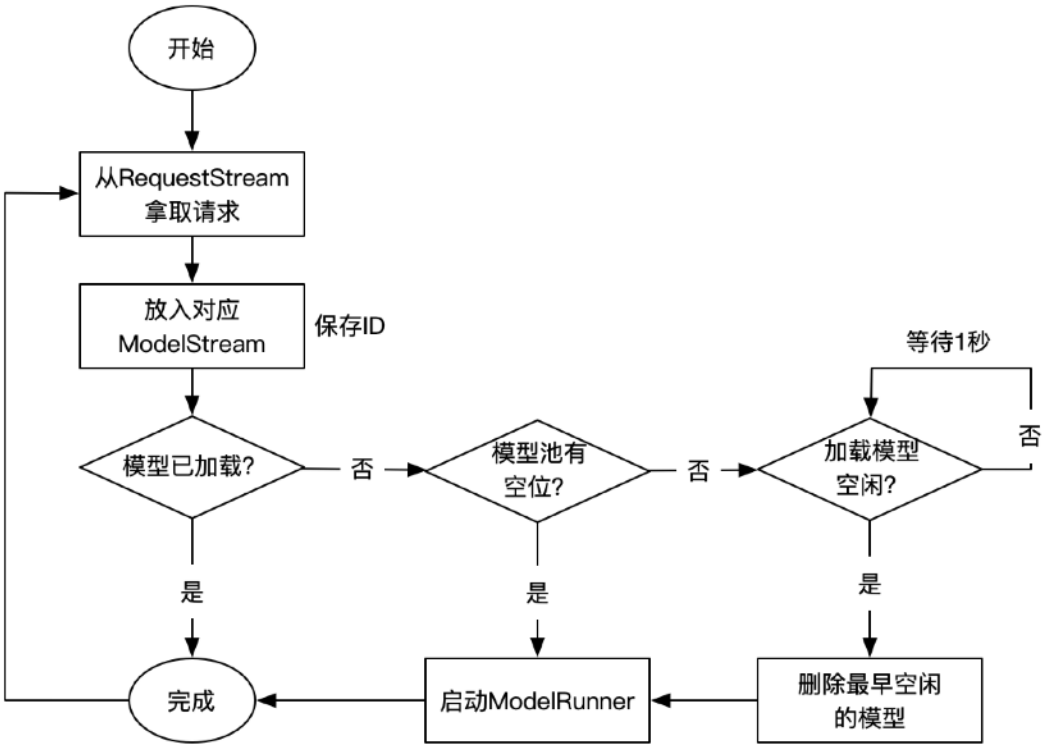
API Service负责监听HTTP请求，并将请求放入REDIS RequestStream消息队列中。为了防止RequestStream消息队列堆积太多的推理请求，当此消息队列达到最大允许的请求个数后，后续的推理请求将会返回失败。



2.1.2.Model Dispatcher

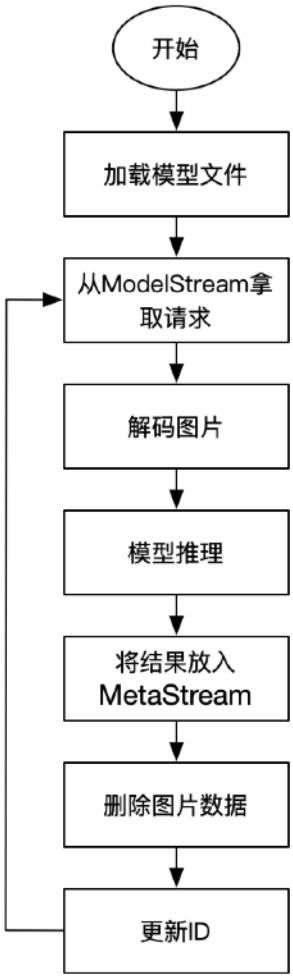
Model Dispatcher负责调度模型。此模块会根据推理请求启动相应的Model Runner，模型池

可以限制最多同时运行的模型数量，当模型池没有空闲但是需要启动新模型的时候，调度器会查找并销毁最早空闲的Model Runner。



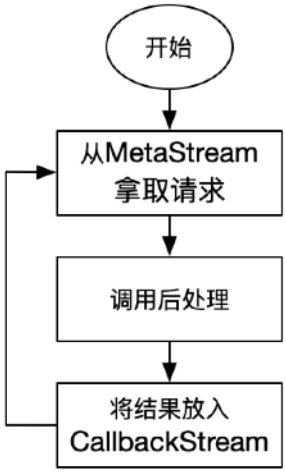
2.1.3.Model Runner

Model Runner负责加载模型和推理图片。每个模型都有一个单独的请求Stream和ID Key，当该模块启动后，它会循环到对应的的ModelStream拿取请求，并对图片进行推理，完成后会将结果发送到MetaStream中，然后更新ID Key。



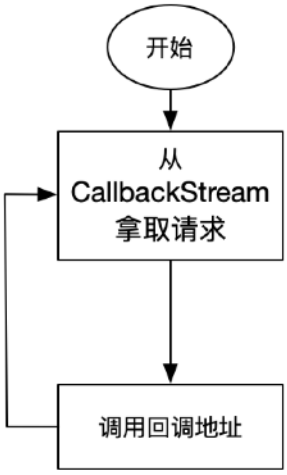
2.1.4.Post Processor

Post Processor负责对推理结果进行后处理，为了防止回调阻塞后处理，该模块会将后处理结果发送到CallbackStream中，又单独的模块进行回调。



2.1.5.API Callback

API Callback负责将后处理的结果进行回调。



2.2. 技术选型

该方案采用REDIS STREAM来作为消息队列，主要基于以下原因：

1. 请求添加后可以获取到唯一ID
2. 可以避免请求丢失

3. 详细设计

3.1. 接口设计

见附录A

4. 部署方案

所有模块都以Docker的形式部署在服务器上，以下是docker-compose.yml的示例。

```
services:
  redis:
    image: redis:7.2.4
    network_mode: host
    restart: unless-stopped
    volumes:
      - /etc/localtime:/etc/localtime

  api-service:
    image: jinfeng/model-repository:0.0.1
    network_mode: host
    restart: unless-stopped
    command: python api-service.py
    depends_on:
      - redis

  model-dispatcher:
    image: jinfeng/model-repository:0.0.1
    network_mode: host
    restart: unless-stopped
    command: python model-dispatcher.py
    depends_on:
      - redis

  post-processer:
    image: jinfeng/model-repository:0.0.1
    network_mode: host
    restart: unless-stopped
    command: python post-processer.py
    depends_on:
      - redis

  api-callback:
    image: jinfeng/model-repository:0.0.1
    network_mode: host
    restart: unless-stopped
    command: python api-callback.py
    depends_on:
      - redis
```

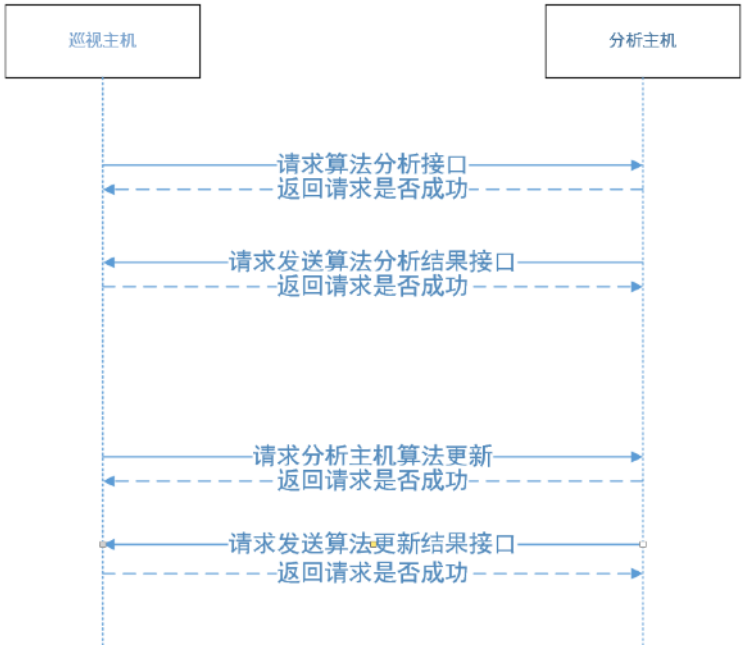
A.2.1.巡视主机与分析主机通信过程

巡视主机与分析主机通信过程满足以下要求：

a) 巡视主机和分析主机之间交互采用HTTP传输协议，参数采用JSON格式进行封装；
b) 分析主机默认端口定义为20011，更新模型端口默认定义为20013。

A.2.2.巡视主机与分析主机交互过程

巡视主机与分析主机交互流程如下：



图A.1 巡视主机与分析主机交互流程

a) 巡视主机向分析主机发送图像分析请求；
b) 分析主机在收到分析请求后，应先应答，待分析图像分析完成，应向巡视主机请求发送分析结果。
c) 巡视主机向分析主机发送算法更新请求；
d) 分析主机在收到算法更新请求后，应先应答，待算法更新完成后，应向巡视主机请求发送算法更新结果。

A.3.应用协议定义

A.3.1.请求图像分析

请求图像分析接口详见表A.1：

1. 请求图像分析接口

请求路径	http://ip:port/picAnalyse
请求类型	POST
调用方	巡视主机
服务方	分析主机
接口描述	请求图像分析
方式	POST
请求头	Content- - Type: application/json;charset=UTF-8

A.3.1.1.请求参数说明

巡视主机向分析主机请求图像分析支持批量上传和单张上传两种方式。taskList参数中包含本次上传图像的URL、相关配置参数及包含的类型。

请求图像分析接口参数说明详见表A.2：

2. 请求图像分析参数说明

参数名称		类型	必选/可选	参数含义	备注
requestHostIp		string	必选	分析结果反馈ip地址	
requestHostPort		string	必选	分析结果返回端口	
requestId		string	必选	请求分析数据唯一标识，UUID	
objectList		string	必选	分析列表，JSON格式的数组	支持多个巡视点位分析请求
	objectId	string	必选	分析点位标识	巡视点位ID
	typeList	string	必选	图像分析类型	定义见A.3.3
	imageNormalUrlPath	string	可选	判别基准图	用于判别模板
	imageUrlList	string	必选	待分析图像的URL	支持多张图像

A.3.1.2.请求参数示例

3. 请求图像分析示例

```
{
"requestHostIp":"XXXXXXXXXXXXXXXXXXX",
"requestHostPort":"XXXXXXXXXXXXXXXXXXX",
"requestId":"XXXXXXXXXXXXXXXXXXX",
"objectList":[
        {"objectId ":"121",
        "typeList":["bj_bpmh","bj_bpps"], //缺陷类识别可以选多种类型，状态类识别只选一种类型
        "imageUrlList":["jpg图像"]} , //单张图片识别表盘破损和表盘模糊
        {"objectId ":"122",
        "typeList":["tx_pb"], //判别类型
        "imageNormalUrlPath ":"jpg图像", //判别基准图，选填
        "imageUrlList":["jpg图像", "jpg图像"]} //多张图片只用于判别
]
}
```

A.3.1.3.接收响应

接收响应返回值见表A.4：

4. 接收响应返回值

序号	字段	取值范围	说明
1	code	int	200：成功; 400：表示客户端请求有语法错误，不能被服务器所理解;

示例：

{
"code”:200,
}

A.3.2.分析结果反馈

分析主机返回分析结果采取分析一张返回一份分析结果的方式。分析结果反馈接口详见表A.5：

5. 分析结果反馈接口

请求路径	http://ip:port/picAnalyseRetNotify
请求类型	POST
调用方	分析主机
服务方	巡视主机
接口描述	分析结果反馈
方式	POST
请求头	Content- - Type: application/json;charset=UTF-8

A.3.2.1.请求参数说明

分析结果反馈接口详见A.6：

6. 分析结果反馈接口参数说明

参数名称	类型	必选/可选	参数含义	备注
requestId	string	必选	请求识别唯一标识，UUID	
resultsList	string	必选	结果集	
objectId	string	必选	分析点位标识	巡视点位ID
results	string	必选	分析结果	
type	string	必选	分析类型	含义详见A.3.3
value	string	必选	值	含义详见A.3.3
code	string	必选	正确 2000 图像数据错误 2001 算法分析失败 2002	图像数据错误是指未能获取到图像数据；算法分析失败是指算法本身分析过程中出错
resImageUrl	string	必选	结果反馈图像url路径	

参数名称			类型	必选/可选	参数含义	备注
		pos	string	必选	图中区域，按照顺时针顺序提供的一系列坐标列表，例如”areas”：[{”x”:”10”,”y”:”15”},{”x”:”100”,”y”:”150”},{”x”:”100”,”y”:”500”} ,{”x”:”10”,”y”:”500”}]	支持多个区域框，支持多边形和矩形框
		conf	float	必选	分析结果置信度	范围0-1，保留4位小数
		desc	string	必选	分析结果描述	

A.3.2.2.返回结果示例

返回示例详见表A.7：

7. 分析结果反馈接口

```
{
“requestId”:”550e8400-e29b-41d4-a716-446655440000”,
"resultList":[
    {"objectId":"1212",
    "results":[
        {“type”:“wcaqm”,
        “Value”: “1”,
        “code”:“2000”,
        “resImageUrl”:””,
        "pos": [{"areas": [{"x": 10,"y": 15}, {"x": 100,"y": 150},{"x": 110,"y": 170},{"x": 200,
         "y": 270}]},{"areas": [{"x": 30,"y": 20}, {"x": 150,"y": 200}]}],//支持多个多边形框区域，只填两个点表示矩形框（左上角坐标+右下角坐标）
        ”conf”:0.9999,
        ”desc”:”未穿安全帽”},
    ]
    }
]
}
```

A.3.2.3.接收响应

接收响应返回值见表A.8：

8. 接收响应返回值

序号	字段	取值范围	说明
1	code	int	200：成功; 400：表示客户端请求有语法错误，不能被服务器所理解; 500：服务端异常
示例：			

{
"code”:200,
}			

A.3.3.图像分析类型定义

A.3.3.1.设备状态识别类型定义

设备状态识别类型定义见表A.9：

9. 设备状态识别类型定义

序号	type属性数值	定义	识别结果值含义
1	isolator	刀闸状态	1 代表分状态，2 代表合状态，3代表分位异常状态，4代表合位异常状态
2	switch	开关/压板状态	一般情况1 代表分状态，2 代表合状态，0代表预留; 可以另行约定业务含义，如1 代表就地状态，2 代表远方状态，0代表停用状态
3	meter	仪表读数	具体仪表值读数（返回多个值以逗号分隔）
4	infrared	红外温度	最高温度，最低温度（以逗号分隔）
5	sound	声音	1 代表正常声音，2 代表异常声音
6	light	指示灯、闪烁灯	1 代表灯灭，2 代表灯亮，3 代表绿灯（常）亮，4 代表红灯（常） 亮，5 代表绿灯闪烁，6 代表红灯闪烁
7	qrcode	实物ID	非空字符串

A.3.3.2.设备缺陷识别类型定义

统一定义缺陷类型值含义： 0表示无缺陷，1 表示有缺陷。

A.3.3.3.设备缺陷判别类型定义

设备缺陷判别类型定义见表N.18：

1. 设备缺陷判别定义

序号	type属性数值	定义	识别结果值含义
1	tx_pb	缺陷判别	0 代表无缺陷，1 代表有缺陷
2	tx_yzwpy	预置位偏移	0 代表无缺陷，1 代表有缺陷