

文件编号：
版 本：
修订状态：

产品/项目 总体设计说明书

编制：

审核：

批准：

目录

总体设计 3

 系统概述..... 3

 系统拓扑图..... 3

 基本设计思想及处理流程..... 4

 技术方案选择..... 9

 系统架构..... 18

 项目需求概述..... 18

接口设计 20

系统数据结构和算法设计 23

系统故障处理设计 28

 故障信息和处理措施..... 28

安全保密设计..... 29

系统维护设计..... 29

系统可靠性设计 29

系统可扩展性设计..... 29

系统性能设计..... 29

系统概述

集中管理包括图像管理、数据统计、日志管理、设备管理、黑名单管理、用户管理、维保管理等子模块;

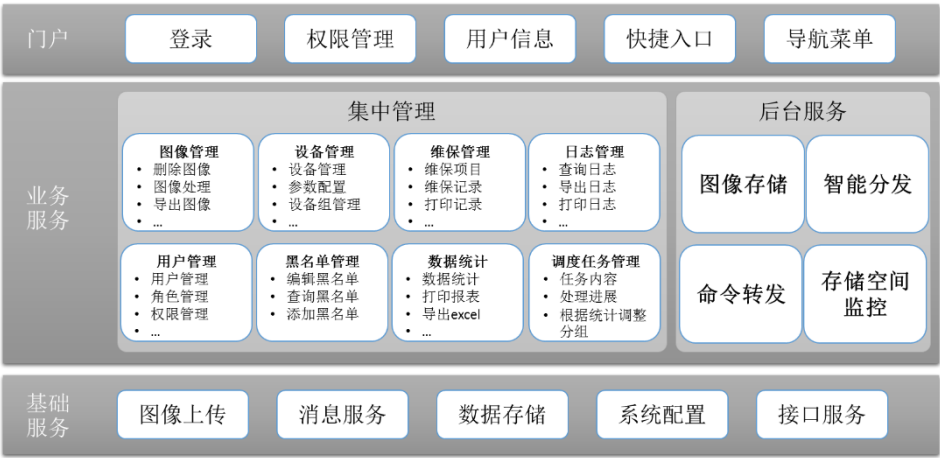
系统拓扑图



- 1) 本部署方案的设计思想：核心服务冗余设计，避免单点故障，提高系统的可用性；整个系统采用 Nginx 负载均衡设计，消息服务采用 HAProxy 作负载均衡；
- 2) 服务支持横向扩展，根据实际需要扩展服务器集群；重要服务如安全服务、接口服务等可根据需要进一步拆分，单独部署服务器；
- 3) 后台数据服务与设备端、判图站、手检站之间的接口：消息通过 MQ 通信，文件通过 FTP 通信；
- 4) 大规模数据采用磁盘阵列存储。
- 5) 监控采用 Zabbix 服务
- 6) 大容量数据，例如日志，考虑采用非结构化方式存储。

基本设计思想及处理流程

一、 整体框架



系统从业务上分为三层，第一层为系统门户，处理用户权限等功能；第二层为业务服务层，处理集中管理和后台服务两部分主要业务功能。基础服务负责为上层服务提供数据和参数配置等功能。

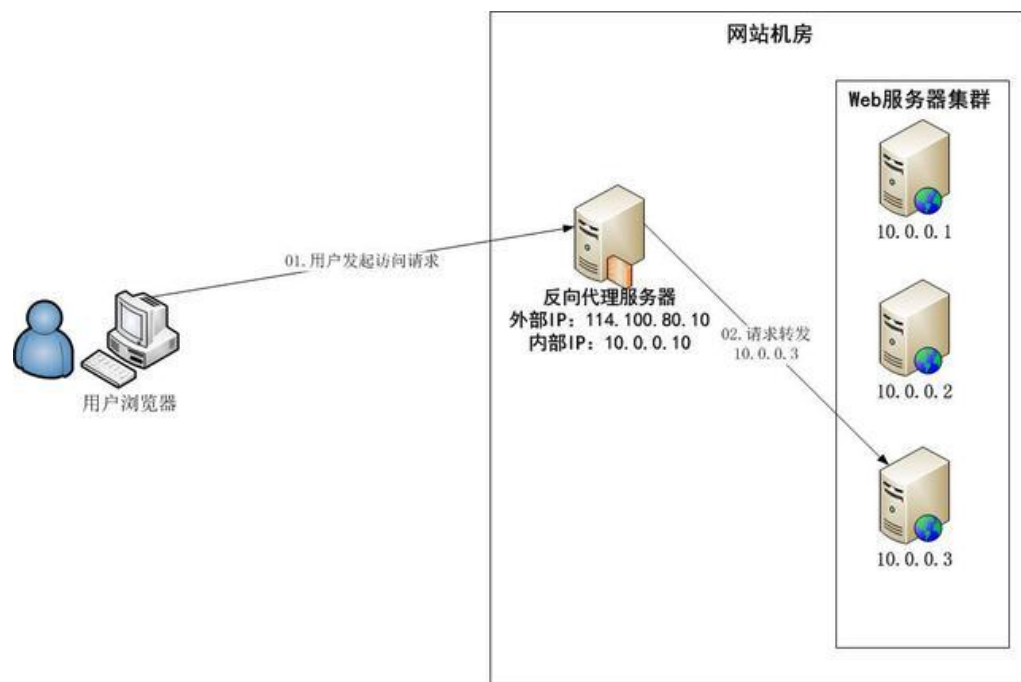
二、 模块化设计

- 1. 本系统五大模块（设备端、远程端、手检端、后台服务、集中管理）之间采用消息队列进行信息交互；
- 2. 后台服务和集中管理之间采用数据库进行数据交换；
- 3. 后台服务和集中管理内部，采用 Restful 接口

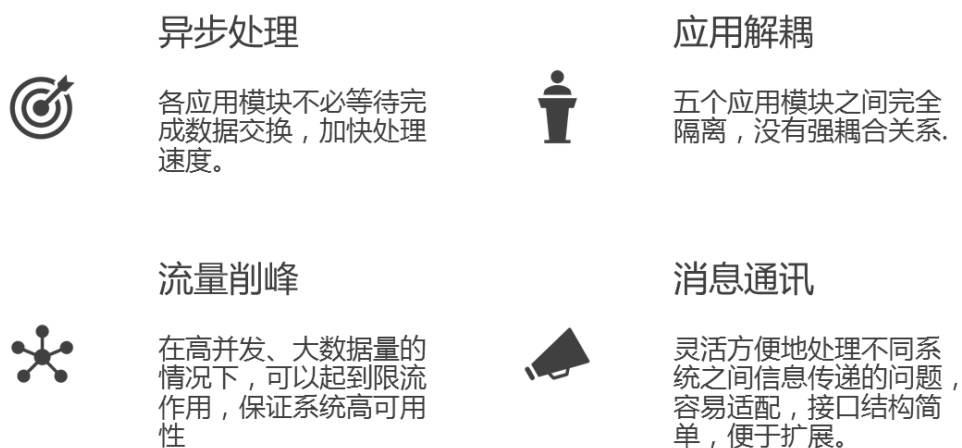
五大模块之间 建议采用消息队列	内部接口 建议采用RESTful API	集中管理和后台服务 建议读写同一个数据库
<ul style="list-style-type: none">• 每个模块架构、数据结构、协议不同，采用消息队列容易适配• 消息队列建议采用Rabbit MQ• 文件传输采用FTPS和FASTDFS	<ul style="list-style-type: none">• 利用 HTTP 协议本身语义，无状态• 轻量，不需要任何别的诸如消息协议。• 一目了然，具有自解释性• 一般以json做数据交换	<ul style="list-style-type: none">• 两个模块之间关系密切数据结构可以统一设计可直接读写同一个数据库，加快处理速度

三、 高性能设计

本项目采用负载均衡、消息队列、缓存等技术提高访问和处理性能。通过 Nginx 负载均衡技术可提高用户访问速度，加快业务处理。



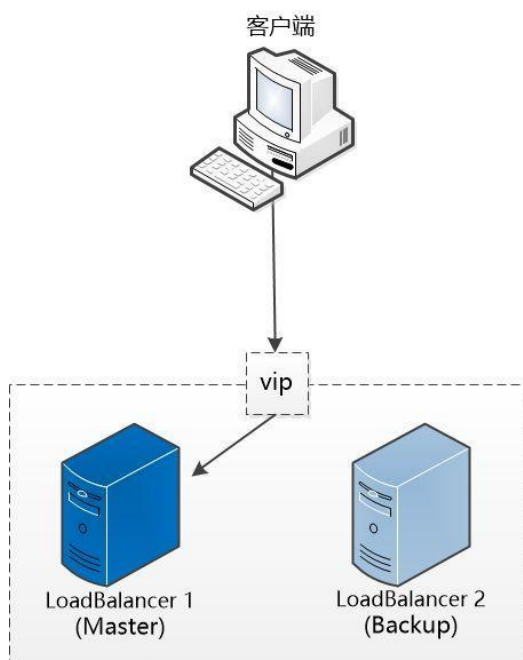
采用消息队列设计，既达到模块解耦的目的，也起到平衡流量，提高性能的目的。



缓存技术既能提高查询的速度，又能辅助 tomcat 解决分布式 Session 的问题。借助缓存技术，还可以在服务端方便快捷地校验用户身份、权限等信息。

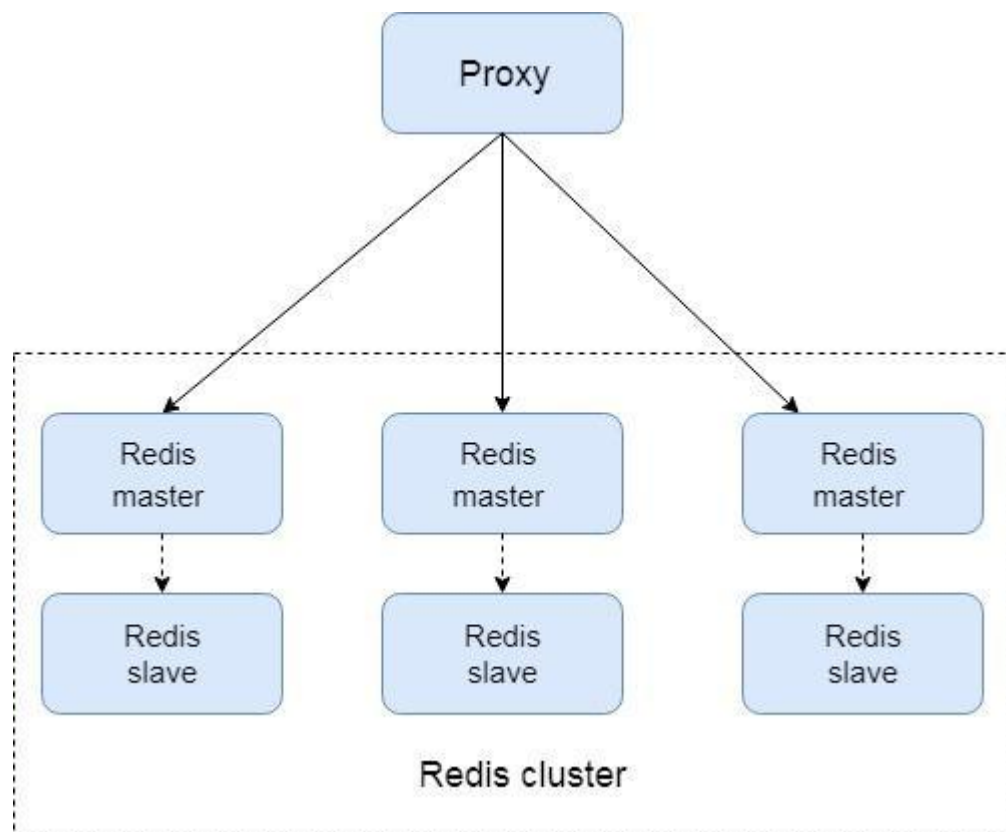
四、 高可靠性设计

本项目在重要服务中，普遍采用了双机或集群的架构，可以明显提高系统的可靠性。主要包括：负载均衡服务、应用服务、数据库服务、消息服务、文件服务等。下图以负载均衡为例，说明双机的原理和作用。



两个 Load Balancer 中，一个是主服务（Master）， 另外一个辅助服务（Backup）， Master 主要提供服务， Backup 待命，一旦 Master 宕机， Backup 服务器立刻接管，不会使业务中断。

下图以缓存服务为例，说明集群架构设计：



该架构可线性扩展到 1000 个节点，采用无中心架构、一致性哈希思想，客户端直连 redis 服务，免去了 proxy 代理的损耗。并且能保证在一台服务宕机后，整个集群的服务不会中断，能更大的提高系统可靠性。

另外，对重要数据，本项目还采用定期备份、及时同步等措施，来保证高可靠性。

五、 安全性设计

本系统通过 RSA 和 AES 加密混合加密流程算法，提高系统信息安全性。

(1)混合加密体制的数据加密

①使用密钥为 K 的 AES 加密算法，假设其为 $E1$ ，那么明文(data)经加密后为 $E1(data,k)$ ，即密文数据。

②假设 RSA 算法为 E2，使用 RSA 算法的公钥 PK 对密钥 K 进行加密，则加密后的 K 为 E2(K,PK)。

(2)混合加密体制的数据解密

①利用 RSA 算法的私钥 SK 对 E2(K,PK)进行解密，得到 AES 算法的密钥 K。

②用密钥 K 进行数据解密，从而得到数据的明文(data)形式。

此外，系统还采用防 SQL 注入、防 XSS 跨站攻击等安全措施，有效提高系统安全性。

技术方案选择

2.4.1 技术选型说明

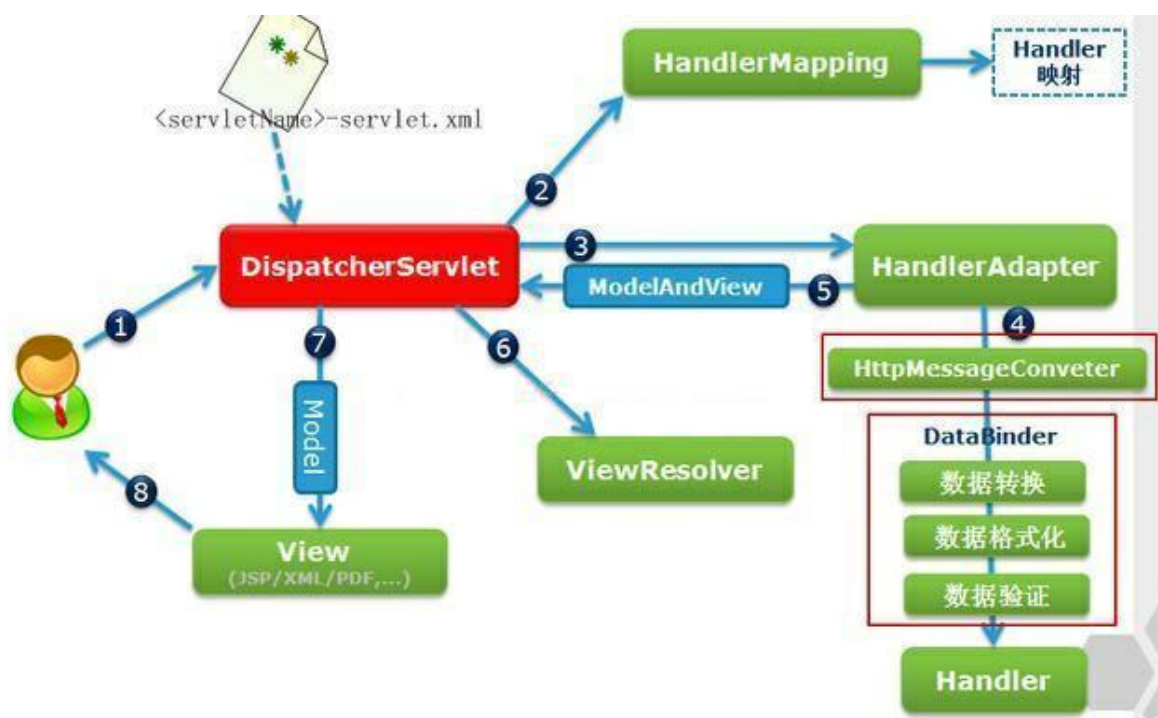
下图是本项目主要的技术选型。



一、 SpringMVC+Vue.js

SpringMVC 是比较流行的 MVC 框架，其设计原理是基于方法的设计，性能比其他框架要快，具有方便地实现 restful、数据共享、处理 ajax 的请求等特性，目前是 WEB 应用中主要备选的 MVC 框架。

下图是 SpringMVC 的工作流程图。可以看到其更为合理和优雅的设计。



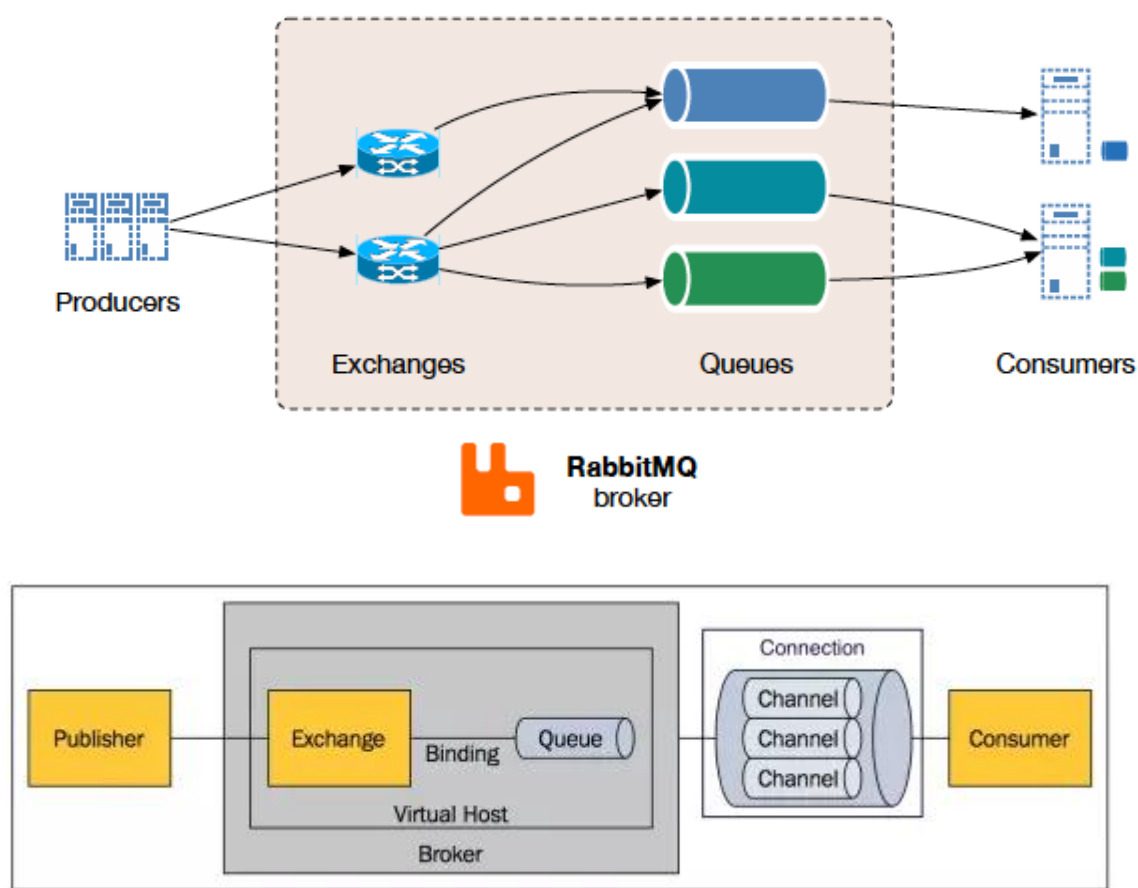
Vue.js 是一个构建数据驱动的 web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。它不仅易于上手，还便于与第三方库或既有项目整合。目前 vue.js 已经成为了前端开发必用的框架之一。

Vue 的核心特点是数据驱动和视图组件化，可以实现前后端分离开发，是前端开发中优选框架。其平缓的学习曲线、模块化、灵活的开发环境、丰

富的生态系统、特有的响应机制和高性能，是本项目选择 Vue.js 的主要考虑因素。

二、 RabbitMQ

RabbitMQ 是一种消息中间件，用于处理来自客户端的异步消息。服务端将要发送的消息放入到队列池中。接收端可以根据 RabbitMQ 配置的转发机制接收服务端发来的消息。RabbitMQ 依据指定的转发规则进行消息的转发、缓冲和持久化操作，主要用在多服务器间或单服务器的子系统间进行通信，是分布式系统标准的配置。



上图是 RabbitMQ 的工作原理图。其优点包括：

1. 由于 erlang 语言的特性，mq 性能较好，高并发；

2. 吞吐量到万级，MQ 功能比较完备
3. 健壮、稳定、易用、跨平台、支持多种语言、文档齐全；
4. 开源提供的管理界面非常完善；
5. 社区活跃度高；

本项目中各设备及后台服务之间传递大量数据和消息，采用消息队列可以起到模块解耦、对接方便、处理速度快等优点，对整个系统正常运行起到重要作用。

三、 Redis

Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。

Redis 是一个高性能的 key-value 数据库。Redis 的出现，很大程度补偿了 memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang 等客户端，使用很方便。

其优点主要包括：

1 Redis 读写性能优异，从内存当中进行 IO 读写速度快，支持超过 100K+ 每秒的读写频率。

2 Redis 支持 Strings, Lists, Hashes, Sets, Ordered Sets 等数据类型操作。

3 Redis 支持数据持久化，支持 AOF 和 RDB 两种持久化方式

4 Redis 支持主从复制，主机自动将数据同步到从机，可以进行读写分离。

5 Redis 的所有操作都是原子性的，同时 Redis 还支持对几个操作全并后的原子性执行。

6 Redis 是单线程多 CPU，这样速度更快。因为单线程，没有线程切换的开销，不需要考虑加锁释放锁，也就没有死锁的问题。单线程-多路复用 IO 模型。效率高。

Redis 优秀的设计思想，使其成为目前最为流行的缓存框架。本项目的特点之一，是存在大量的静态数据，包括图像、基本信息等，在进行查询时，如果使用传统的数据库技术，性能很难有显著提升，借助 redis 缓存技术，将会对系统处理效率大幅度提升。

四、 Nginx

Nginx 是一款自由的、开源的、高性能的 HTTP 服务器和反向代理服务器；同时也是一个 IMAP、POP3、SMTP 代理服务器；Nginx 可以作为一个 HTTP 服务器进行网站的发布处理，另外 Nginx 可以作为反向代理进行负载均衡的实现。

Nginx 的主要优点：

1. 高性能，高并发。
2. 可扩展性。
3. 稳定性。

4. 可热部署。

下图是几种常见 web 服务器的对比，可以看到 nginx 的优越性。本项目中，对后台系统访问量较大，设备之间关系复杂，采用 Nginx 可以起到负载均衡、提高性能的目的。

对比项\服务器	Apache	Nginx	Lighttpd
Proxy 代理	非常好	非常好	一般
Rewriter	好	非常好	一般
Fcgi	不好	好	非常好
热部署	不支持	支持	不支持
系统压力	很大	很小	比较小
稳定性	好	非常好	不好
安全性	好	一般	一般
静态文件处理	一般	非常好	好
反向代理	一般	非常好	一般

五、 FTPS

FTPS 是使用安全套接层（SSL）证书的 FTP 安全技术。整个安全 FTP 连接使用用户 ID、密码和 SSL 证书进行身份验证。一旦建立 FTPS 连接，FTP 客户端软件将检查目标 FTP 服务器证书是否可信的。

优点：

通信可以被人们读取和理解

提供服务器到服务器文件传输的服务

SSL/TLS 具有良好的身份验证机制（X.509 证书功能）

FTP 和 SSL 支持内置于许多互联网通信框架中

本项目采用 FTPS，提高数据传输的安全性和保密性。

2.4.2 技术预研

为了加快研发进度，降低项目风险，本项目开发团队在前期对重要和关键技术进行了预研。目前这些预研工作已经取得阶段性成果，达到了预研目标。在后续的开发工作中，将采用这些预研的结果深入开展研发工作。

技术预研的主要工作包括：

- 1、项目应用开发服务器部署
- 2、RabbitMQ 集群搭建部署架构
- 3、RSA 和 AES 加密混合加密流程算法和应用场景
- 4、zxbbix 部署

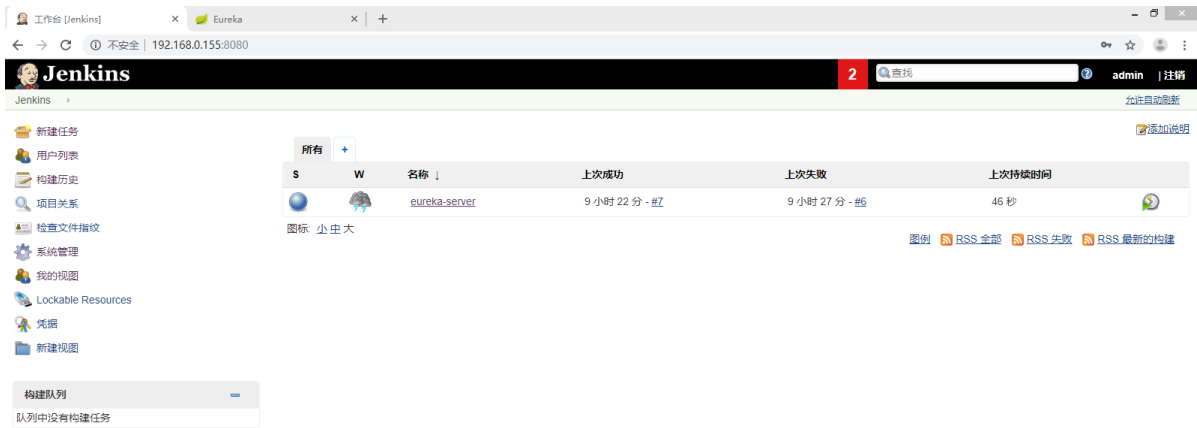
一、项目开发应用服务器部署

环境构建描述

在 CentOS 7.4 安装 maven、eureka 工程服务通过 git 以及推送 gitlab。

jenkins+docker 部署 eureka 服务，实现开发环境自动化部署代码。

部署效果如下图所示：

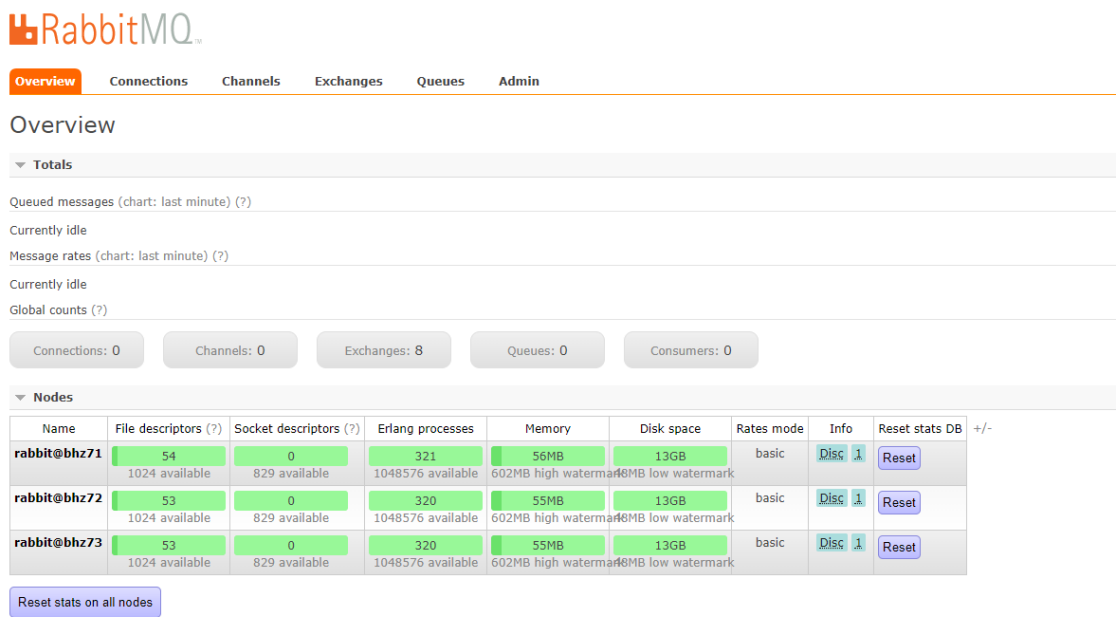


二、rabbitmq 集群搭建部署架构

环境描述：

安装 rabbitmq 3.6.12。

下图为 RabbitMQ 集群队列状态页面截图



三、RSA 和 AES 加密混合加密流程算法

客户端使用 RSA + AES 对重要信息进行加密

客户端加密过程主要分为以下三个步骤：

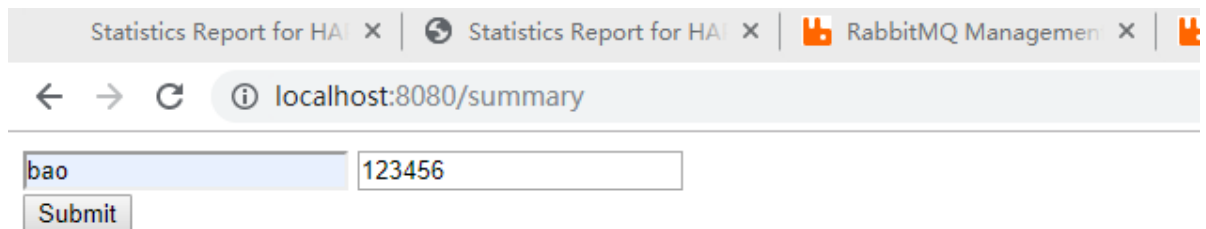
- 1、客户端获取 AES 的密钥；
- 2、对身份证信息（重要信息）进行 AES 加密；
- 3、通过使用 RSA 对 AES 密钥进行公钥加密。

服务端使用 RSA + AES 对重要信息进行解密

服务端解密过程主要分为以下两个步骤：

- 1、获取 AES 密钥
- 2、对加密后的 AES 密钥进行 RSA 私钥解密，拿到密钥原文；
- 3、对加密后的重要信息进行 AES 解密，拿到原始内容。

下图模拟用户信息提交页面 request 信息进行 aes 、rsa 加密



服务端公钥：

```
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCfXfMzg
g4m5RRlg2vcrYBFN4sBhE1VtW1sBkXxC5wtCRa0Zv0kud
k9CIQfU6c+eEaaZKUnygzHWdSqdWURCE0IKgLco1XF+RH
mu/r1977FfjRg9pAkBg5z05PfHDqWqkIsqX0iRaSP31BU
ZOgtwafbiBv2dBvRBMdq03ty4q80QQIDAQAB
```

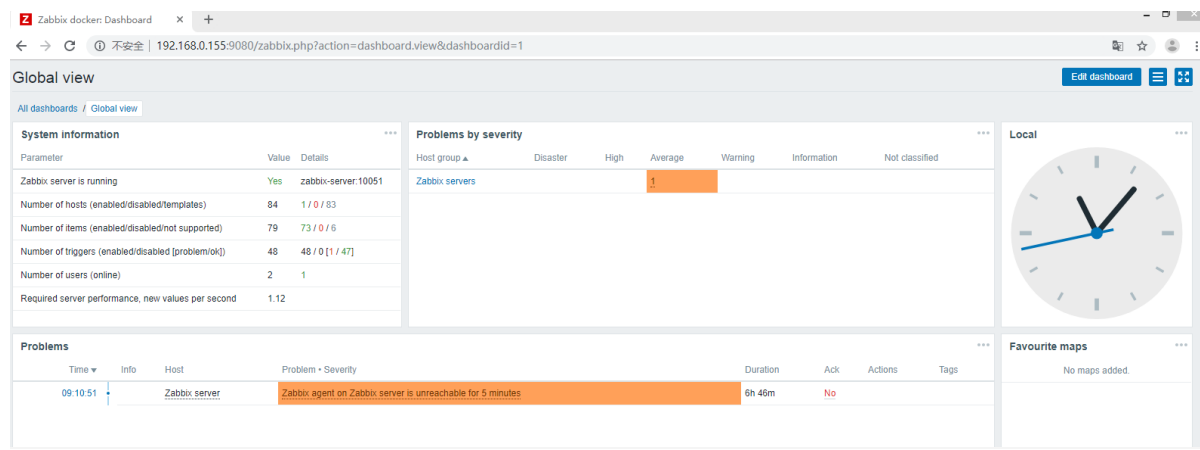
客户端私钥：

```
MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wgJbAgEAA
oGBAITfv1C8+Nr+Vz3DnhuCWw41ax8PG+rCiXt/f4XjRM
lJ9ZC2AuMMbtHLsTMLhCrhgHt1MxdcoYtqvQfxu4AVOh6
pZrxMr2AiyNpw8SecmM3m0YWYNc7tnUB6/vlLyQduikD4
qaxNiB5FcUiRpiRoLpz7rT6UV+/zDh+ibgvZRLDRAgMBA
```

四、zabbix 部署

部署环境：采用 zabbix3.4

下图为 zabbix 页面截图：



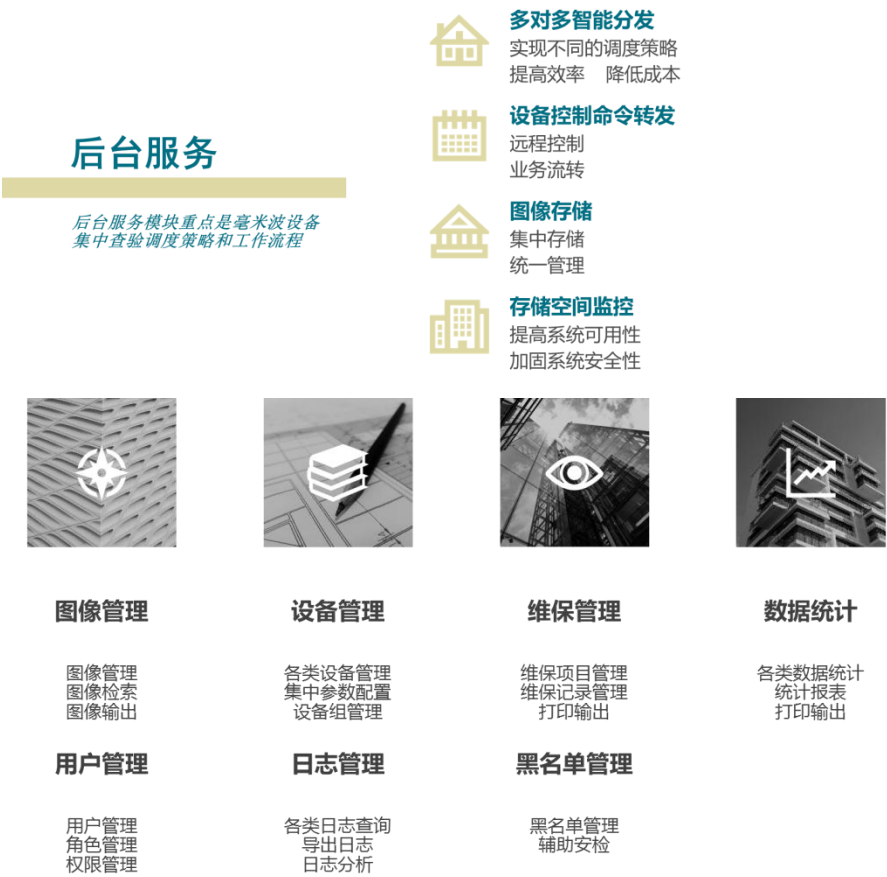
系统架构



从技术上，本系统分为四层，第一层为门户层，处理业务展现，主要应用的技术为 Spring mvc 和 vue.js；第二层为业务应用，主要实现后台服务和集中管理功能；基础服务为业务应用提供技术支撑，主要技术为 Spring 框架、接口、restful、文件服务等。底层服务提供持久化服务，主要技术框架包括 mybatis、非结构化存储、消息。

项目需求概述

本系统需求主要包括：后台服务和集中管理两部分。如下图所示。



后台服务主要包括：多对多智能分发、图像存储、存储空间监控、设备控制命令转发。

集中管理包括：图像管理、数据统计、日志管理、设备管理、维保管
理、用户管理和黑名单管理。

接口设计

系统接口主要包括设备端、远程端和手检端与后台服务之间的接口。（待完善）

一、 设备端

发送到后台服务

- A. 登录（注册）
- B. 登出（注销）
- C. 心跳
- D. flow 信息（即时状态）
- E. 扫描图像信息（包括附加信息）
- F. 请求调度手检站
- G. 提交手检结论（本机手检模式）
- H. 配置更改后发送配置信息
- I. 同步数据

后台服务推送

- A. 用户列表
- B. 判图结论
- C. 调度的手检站信息

- D. 手检结论
- E. 工作超时提醒
- F. 控制信息
- G. 远程端和手检站状态信息

二、 远程端

发送到后台服务

- A. 登录（注册）
- B. 登出（注销）
- C. 心跳
- D. 状态（空闲时发送一次）
- E. 请求图像（包括查验等级、ATR 信息）
- F. 提交判图结论（包括结论、嫌疑框）

后台服务推送

- A. 用户列表
- B. 判图超时提醒（倒计时）
- C. 工作超时提醒
- D. 待判图数量

三、 手检端

发送到后台服务

- A. 登录（注册）
- B. 登出（注销）
- C. 心跳
- D. 状态（空闲时发送一次）
- E. 提交手检结论（OK、误报、漏报、收藏）
- F. 提交备注、拍照记录照片

后台服务推送

- A. 用户列表
- B. 待手检卡通图像（包括查验等级、嫌疑框）
- C. 工作超时提醒

系统数据结构和算法设计

一、工作模式和智能分发

本系统共有五种工作模式，第一种是单机模式，后四种属于联机模式。



单机模式中，与系统没有关联，不需要考虑工作流程。以下着重讨论联机模式的工作流程和调度流程的算法。

1、 工作流程

1) 算法思路

把工作流程的各环节存储到 ArrayList 中，根据环节中定义的任务执行具体操作。

环节任务类型包括：

- 发送命令（包括通知）
- 接收命令
- 存储数据
- 调度审图端

e. 调度手检端

f. 获取图像

g. 保存判图结果

h. 保存手检结果

i. 判断结果

j. 跳转

任务的属性包括：流程 ID，环节 ID，动作发起节点，动作接收节点，时间，动作编码，动作类型，动作内容，当前环节状态等。流程 ID 用以标识当前流程，节点 ID 用以标识当前环节。

当前环节状态包括：未启动，已启动，已完成，挂起，中止等。

2) 以“设备端+远程端+手检站”为例说明数据结构

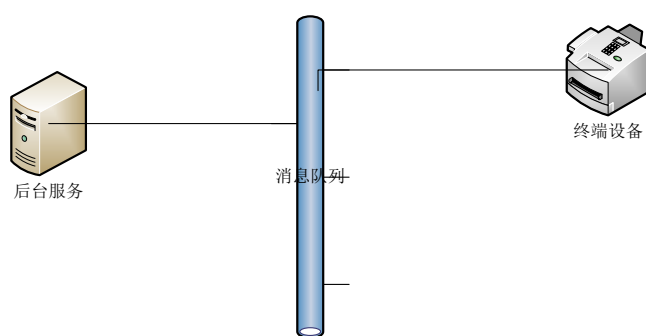
ArrayList								
流程ID:1	环节ID:1	动作发起节点:101	动作接收节点:105	时间: 2019-10-02 16: 55: 01	动作编码: 10001	动作类型: 存储数据	动作内容: 图片url	当前环节状态: 已启动
流程ID:1	环节ID:2	动作发起节点:105	动作接收节点:102	时间: 2019-10-02 16: 55: 50	动作编码: 10002	动作类型: 调度远程端	动作内容:	当前环节状态: 未启动
流程ID:1	环节ID:3							
流程ID:1	环节ID:4							
流程ID:1	环节ID:5							
流程ID:1	环节ID:6							
流程ID:1	环节ID:7							
流程ID:1	环节ID:8							
流程ID:1	环节ID:9							
流程ID:1	环节ID:10							
流程ID:1	环节ID:11							
流程ID:1	环节ID:12							
流程ID:1	环节ID:13							
流程ID:1	环节ID:14							
流程ID:1	环节ID:15							
流程ID:1	环节ID:16							
流程ID:1	环节ID:17							
流程ID:1	环节ID:18							

以上图为例，ArrayList 中存储每个流程的各环节，每环节任务完成后，状态变为已完成，前一个环节的状态为已完成以后才会启动下一环节。最终所有环节任务完成后，本次流程结束，把流程信息持久化到数据库。

3) 典型任务处理模型

a. 发送和接收命令：

如下图所示，后台服务向指定的终端设备发出消息，或者从消息队列中获得终端设备发来的消息。在消息成功发出或接收后，把本环节的状态置为已完成，本环节结束，流程进入到下一环节。在接收命令时，要解析消息内容，根据命令的要求触发下一个环节的任务，直到所有流程结束。



b. 存储数据（包括图像和结果）

解析接收到的消息中的图像地址，把地址和各相关信息存储到数据库中。本环节结束，状态置为已完成。



c. 调度（包括手检端和远程端）

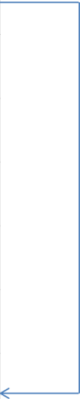
把查验命令通过消息队列发给相应的设备端，与发送命令相同。

d. 判断

使用调度策略中的算法，查找合适的终端设备，如果查不到设备，根据流程的设定跳转到指定的工作环节，如果查到设备，把命令通过消息队列发给终端设备，本环节结束，状态置为已完成。

e. 跳转

有些环节的任务，主要是判断环节，有时不能按环节顺序向下执行，需要根据流程的设定，跳转到某一环节。例如在手检端无可用设备时，流程将跳转到设备端手检环节。

流程ID:1	环节ID:5	判断手检端是否可用	
流程ID:1	环节ID:6	手检端执行手检任务	
流程ID:1	环节ID:7	...	
流程ID:1	环节ID:8		
流程ID:1	环节ID:9		
流程ID:1	环节ID:10		
流程ID:1	环节ID:11	设备端执行手检任务	
流程ID:1	环节ID:12	流程结束	

2、 调度策略

1) 算法思路：把设备分组信息和智能分发要素存入 Hashmap 中，实时更新设备状态等信息，使 Hashmap 保持最新。在智能分发中查询 Hashmap 的元素，找到合适的节点。如果找到多个合适节点，采用随机算法选择一个节点。

2) 算法图示：

如下图所示，最外层 hashmap 存储设备 ID (key) 和设备对应的信息 (value)，设备对应信息中存储一个二维 ArrayList，第一维每个元素对应一个节点的信息（包括设备端、手检端和远程端），第二维存储节点的具体信息，包括 ID、种类、检查员性别、可用状态和在线状态。

在智能分发时，查找本组（设备 ID 相同）的数据，再根据设备种类、检查员性别、可用状态和在线状态等查找符合条件的记录。在占用节点后，更新该节点为占用状态，在完成任务后，更新该节点为可用状态。

Hashmap					
Key (设备ID)	value (设备信息-ArrayList)				
1	ID:101	种类：设备端	检查员性别：女	可用状态：可用	在线状态：在线
1	ID:102	种类：手检端	检查员性别：女	可用状态：占用	在线状态：在线
1	ID:103	种类：手检端	检查员性别：男	可用状态：占用	在线状态：离线
1	ID:104	种类：手检端	检查员性别：男	可用状态：可用	在线状态：在线
1	ID:105	种类：远程端	检查员性别：女	可用状态：可用	在线状态：在线
1	ID:106	种类：远程端	检查员性别：男	可用状态：占用	在线状态：在线
1	ID:107	种类：远程端	检查员性别：女	可用状态：占用	在线状态：在线
2					
2					
2					
...					
99					
99					
99					
100					
100					
100					

系统故障处理设计

故障信息和处理措施

故障分类	故障信息记录方法	解决方案	备注
系统故障	操作系统日志	通过管理客户端远程排除	
软件故障	软件系统日志	通过日志分析故障原因，采取合适的手段排除。	

安全保密设计

本系统对 Web 客户端与服务端通信的数据进行加密处理，加密算法为 RSA 和 AES 加密混合加密流程算法。

系统维护设计

本系统采用 Zabbix 监控硬盘是否有可用空间。采用 rabbitmq 管理客户端监控消息队列的状态。

系统可靠性设计

在排除不可抗力的前提下，系统可靠性设计为 40 年中整个系统故障停机时间不超过 2 小时。

系统可扩展性设计

采用模块化设计，每个功能单独设计为一个模块；功能之间解耦，没有强依赖关系；

采用消息队列设计，各模块之间用消息进行通讯，方便接口的扩展；

模块内部各层之间接口支持 RESTful；

系统支持第三方软件按照约定方式集成；

支持硬件横向扩展。

系统性能设计

系统支持 300 个设备端及 600 个手检端和 600 个远程端同时访问。后台服务支持 2000 位用户同时访问。