

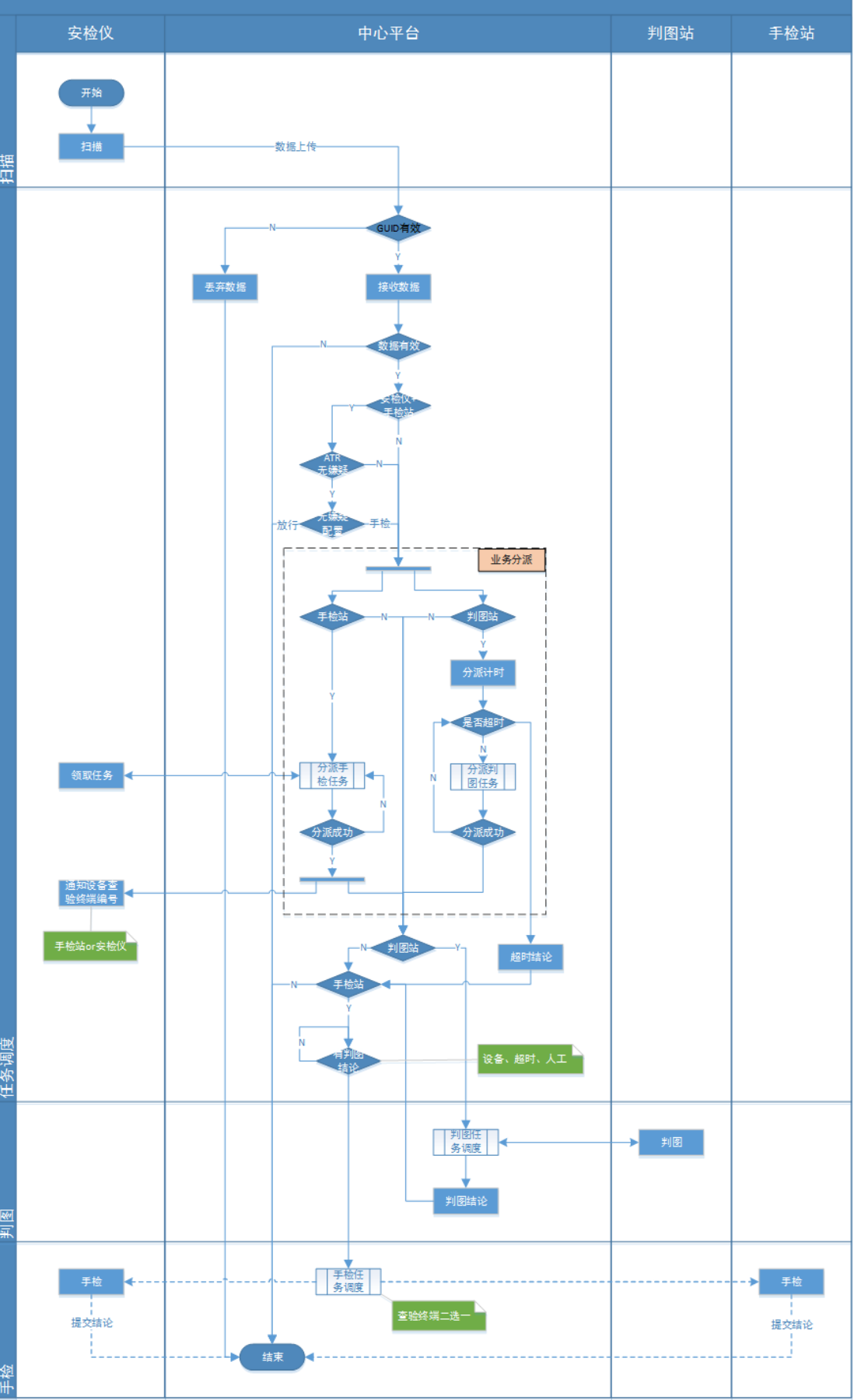
# 需求说明

## 一、原始需求

见《毫米波集中查验需求模型 929-New》

其中设备控制命令转发的流程图如下所示。可以参考附件 ppt 的说明。

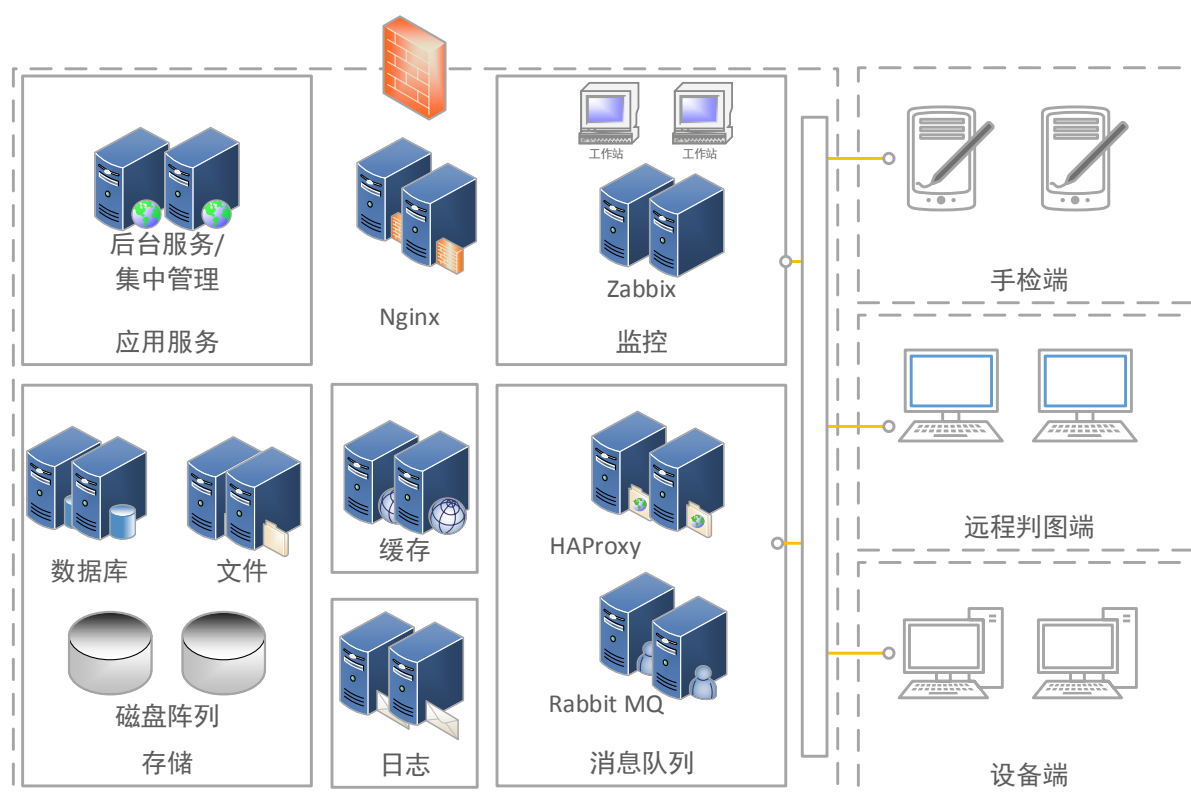
毫米波安检仪集中查验业务流程图



## 二、需求分析

后台服务包括图像存储、智能分发、存储空间监控、命令转发、定时任务等子模块。

### 系统拓扑图



- 1) 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；
- 2) 大容量数据，例如日志，考虑采用非结构化方式存储。
- 3) 缓存内容：

(1) 登录用户的信息、用户和设备的关联信息

设备的 id，设备类型，用户 id，用户账号，权限等

(2) 设备调用运行配置信息（设备绑定的调度策略信息）

安检仪绑定的手检端、审图端、男查男，女查女的信息

(3) 扫描任务信息（安检仪扫描环节数据上传审图端环节，同时被扫描人安排到对应的手检端进行手检，审图端结果出来后，后台把信息发送到指定的手检端）

被扫描人的信息，图片地址，嫌疑框等信息，包括流程环节信息，当前任务的 id，流程 id，环节 id。手检结束后，从 redis 中删除，

安检仪扫描环节同时到达扫描环节和手检环节派发大概几秒内，整个处理环节手检端不限制时间，其他环节有时间限制。

(4) 用户配置下发信息

用户列表，数据字典同步（查获物和等级信息，历史收藏便签）

每个设备登录都同步用户列表，字段信息

(5) 设备参数配置信息（可以考虑）

## 系统数据结构和算法设计思路

（以下为设计思路，供参考，实际实现要细致考虑）

### 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。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: 56: 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							

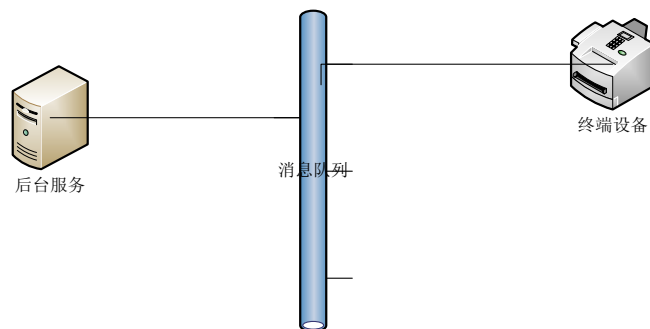
以上图为例，ArrayList 中存储每个流程的各环节，每环节任务完成后，状态变为已完成，前一个环节的状态为已完成以后

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。

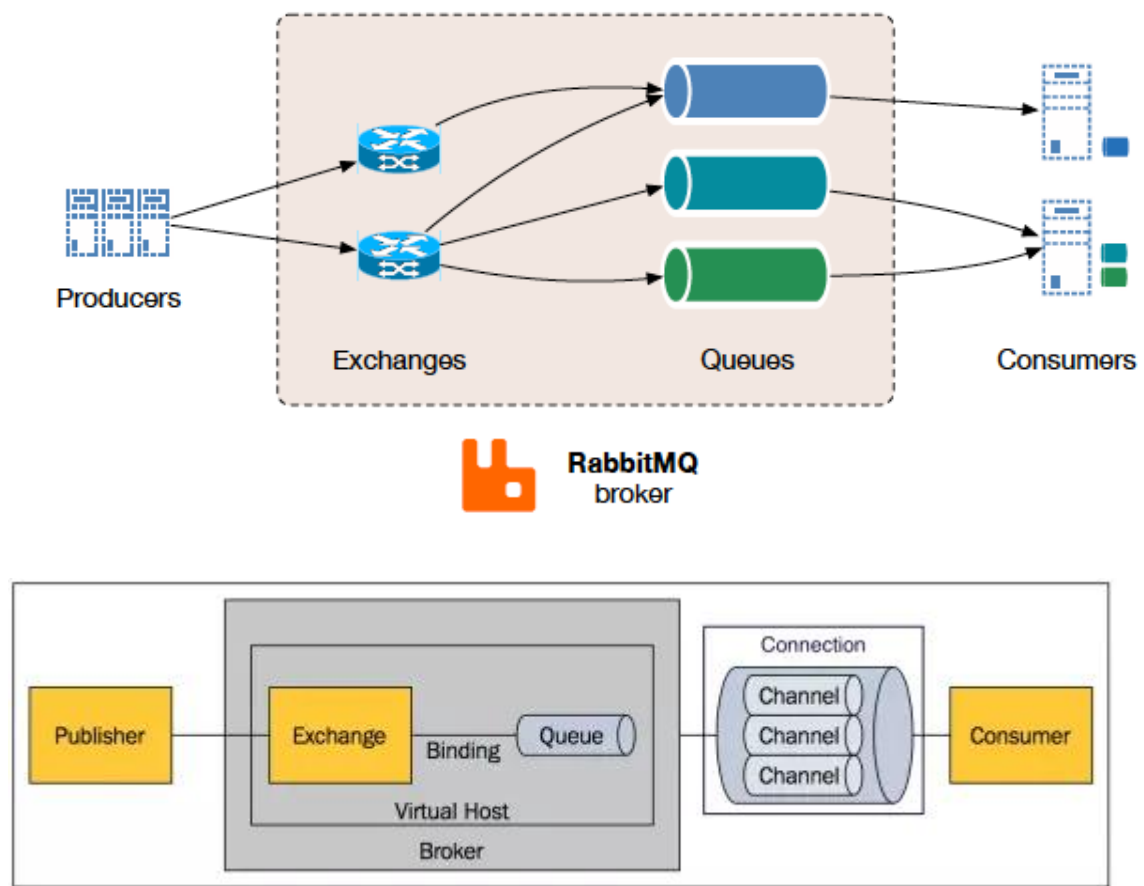
在智能分发时，查找本组（设备 ID 相同）的数据，再根据设备种类、检查员性别、可用状态和在互状态等查找符合条件的记录。在占用节点后，更新该节点为占用状态，在完成任务后，更新该节点为可用状态。

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					

# 所用到的技术

## 一、 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。

## 二、 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，很大程度上补偿了 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，主机会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。

### 三、 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。

#### 四、 Elasticsearch

Elasticsearch 是一个分布式的 RESTful 风格的搜索和数据分析引擎。

查询：Elasticsearch 允许执行和合并多种类型的搜索 — 结构化、非结构化、地理位置、度量指标 — 搜索方式随心而变。

分析：找到与查询最匹配的十个文档是一回事。但是如果面对的是十亿行日志，又该如何解读呢？Elasticsearch 聚合让您能够从大处着眼，探索数据的趋势和模式。

速度：Elasticsearch 很快。真的，真的很快。

可扩展性：可以在笔记本电脑上运行。也可以在承载了 PB 级数据的成百上千台服务器上运行。

弹性：Elasticsearch 运行在一个分布式的环境中，从设计之初就考虑到了这一点。

灵活性：具备多个案例场景。数字、文本、地理位置、结构化、非结构化。所有的数据类型都欢迎。