# 开放平台架构设计说明书

编制部门：公共技术平台

编 制 人：梁晓发

审 核：

会 签：

批 准：

|  |  |  |  |
| --- | --- | --- | --- |
| **文件修订履行** | | | |
| 版本 | 修订内容简述 | 生效日期 | 修订部门/修订者 |
| 1 | 新制订 | 2017-8-30 | 公共技术平台/梁晓发 |
| 2 | 修改 | 2017-9-8 | 公共技术平台/梁晓发 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目 录

[1. 引言 4](#_Toc8145)

[1.1背景与目的 4](#_Toc25789)

[1.2产品信息 4](#_Toc3517)

[1.3软件名称 4](#_Toc28578)

[1.4术语与略缩语 4](#_Toc9905)

[1.5参考资料 5](#_Toc20731)

[2. 系统概述 5](#_Toc29353)

[2.1平台吞吐量理论值预估 5](#_Toc16174)

[2.2技术指标 5](#_Toc19299)

[2.3前端与后台访问序列图 6](#_Toc32298)

[2.3.1前端与人脸识别后台注册交互流程图如下： 6](#_Toc6547)

[2.3.2前端与人脸识别后台识别交互流程图如下： 7](#_Toc15949)

[2.4服务器配置分析 8](#_Toc29876)

[2.4.1应用服务器的配置方案 8](#_Toc1902)

[2.4.2文件系统服务器配置方案 8](#_Toc606)

[2.4技术选型 9](#_Toc14691)

[2.5开发环境 9](#_Toc6471)

[3.部署架构图 9](#_Toc13682)

[4. 系统设计方案 10](#_Toc3420)

[4.1 MYSQL与mongodb数据同步设计 10](#_Toc8334)

[4.1.1双写机制 10](#_Toc30101)

[4.1.2定时任务补充检测机制 11](#_Toc27737)

[4.2系统日志设计 11](#_Toc367)

[4.3核心数据表设计 12](#_Toc1335)

[4.4用户界面接口 12](#_Toc25395)

[5. 重用设计 13](#_Toc13816)

[5.1代码的可重用设计 13](#_Toc15727)

[6. 风险 13](#_Toc5607)

# 引言

## 1.1背景与目的

要设计AI开放平台，先要清楚基于该平台这个层面上，需要依赖的是什么？被调用的客户端又是什么？根据目前得知的业务需求，前面由安卓使用HTTP协议调用，需要依赖的是C++编写的SDK，用于分析人脸识别结果。由于目前的用户基数是数千级别的，所以一开始在服务器配置选型上，还没涉及到超高性能的要求；但是，围绕着后期的持续发展的用户数以及并发请求数的增多，平台的架构设计必须一开始就定位为可伸缩，可扩展，高安全，可维护性强的架构系统。

可伸缩，表示可以线性添加服务器构成分布式集群部署的方式处理业务请求。

可扩展，意味着平台层面上的扩展力强，无论后期添加多个服务中间件或者并且后续有新需求，都能够在极少改动原有代码或基本不需要改动源代码的基础上实现。

高安全，因为客户端调用的是安卓系统，传输的人脸识别数据以及人脸图片数据属于机密属性，因此，需要考虑防窃听防篡改措施，因此需要使用SSL/TLS证书。

可维护性强，上线后需要维护，维护成本高低取决于代码质量以及逻辑分层设计，这可以通过编码规范，数据库设计规范等等手段去约束。

## 1.2产品信息

AI开放平台

## 1.3软件名称

UBT\_AI\_PLATFORM V1.0.0

## 1.4术语与略缩语

| 术语/缩略语 | 说明 |
| --- | --- |
| QPS | Query Per Seconds 服务器每秒能处理的请求数. |
| PV | Page View 页面浏览量,也即后台接口每天请求总数 |
| ART | Average Response Time 平均响应时间 |
| AC | Agency Center代理中心 |

## 1.5参考资料

【无】

# 系统概述

## 2.1平台吞吐量理论值预估【单机】

仅对单机进行吞吐量的理论值预估，后续究竟需要多少台服务器，主要根据实际的整体接口调用访问量进行动态线性伸缩。理论上整体吞吐量=单机吞吐量\*服务器数量。

要预估吞吐量，重要的参数是QPS，开放平台的每个API接口每秒的并发处理能力不一样，因此，QPS指所有API接口的QPS平均值。

QPS指代最佳线程数。超出此线程数的并发请求会影响吞吐量，此时需要横向扩展服务器。

### 2.1.1 最高理论值预估

QPS：

根据每秒并发请求数60个来评估。(具体需要经过压测，QPS=30是折中参考值)

公式：(一天总秒数\*QPS)=86400\*60=5184000(PV)

### 2.1.2 参考二八原则理论值预估

参考理论：二八原则，即每天80%的请求落在20%的时间上.

QPS：

根据每秒并发请求数60个来评估。(具体需要经过压测，QPS=60是折中参考值)

理论日PV值：

以QPS为60，根据二八原则可以得出服务器吞吐量每天51.84W次请求。

公式：(一天总秒数\*20%\*QPS)/80% = (86400\*0.2\*30)/0.8 =1296000 (PV)

## 2.2上线时整体技术指标预估

1. 图片数据存储量

(1).使用云存储，理论上无上限

(2).自建文件系统，上限即为磁盘的容量

2.存储吞吐量:256MBps，存储IOPS:20000

3.QPS>=150（服务器集群部署）

4.响应时间 < 1S （不含网络开销）

5.最大日均PV：350W左右

## 2.3 开发者帐号与访问限制

要使用优必选AI开放平台，需要先注册为平台开发者。开发者角色有两种：优必选公司内部帐号以及第三方帐号。

由于平台部署在云端，在网络访问速度上来讲，公司内部帐号以及第三方帐号的访问速度是一致的。但是，第三方帐号的一些约束条件，比如同一帐号的并发请求数，每天访问限次数，超过次数需要付费等等规则，在公司内部帐号中完全没有这种约束体现。

下面是公司内部帐号以及第三方开发者帐号在平台中的一些差异。

|  | 公司内部帐号 | 第三方帐号 |
| --- | --- | --- |
| 每个接口每天限次访问 | 无限 | 1000次/天免费 |
| 并发请求数 | 无限 | 2个并发 |
| 存储空间 | 无限 | 受限 |
| 超过限制付费 | 不需要 | 需要 |
| 服务降级,过载保护 | 不会 | 会 |
| 创建应用数量 | 不受限 | 20个上限 |

说明：

1. 每个接口每天限次访问。从每天0时开始算起，直至23时59分59秒为止，每个接口的调用次数会累计，超出限制次数则返回“访问失败：您的接口免费调用次数已用完”提示。公司内部帐号无限制。
2. 并发请求数。是指同一个开发者帐号对同一个接口的并发请求限制。第三方帐号只有2个并发请求数。公司内部帐号无限制。
3. 存储空间。注册人脸图片的存储空间。第三方帐号受限，比如500G免费空间，公司内部帐号不受限。
4. 超过限制付费。接口每天调用次数以及存储空间如果超出了免费调用次数，如果需要扩展则需要付费。公司内部帐号不需要付费。
5. 服务降级。当服务器在同一个时刻，并发请求数超出一个阀值，公司内部帐号的请求会进入队列排队，而第三方帐号的请求会直接拒绝(服务器繁忙请稍后再试)。

## 2.4 Access\_Token

Access\_Token即为访问令牌，相当于开放API的访问通行证，具备Access\_Token的访问都是合法的访问。要获取Access\_Token访问令牌，必须先成为平台的开发者。获取Access\_Token访问令牌的过程见下图。



上图中的创建应用之后，平台会为该应用分配api\_key和secret\_key密钥对，每创建一个应用，都会产生一对密钥对。然后以此密钥对请求获取Access\_token即可。

第三方开发者帐号创建应用的数量受限制，这里限定20个上限。

Access\_Token是具备超时时间限制的，这里超时时间为30天。Access\_Token在获取之后，一般由客户端缓存，后续多个请求中复用。在超时之后重新获取即可。Access\_Token的获取次数目前没有限制。

## 2.5技术选型

人脸识别后台技术选型：

开发语言：Java

MVC框架：SpringMVC

数据库：MySQL5.6

搜索引擎：ElasticSearch

文件系统：Fastdfs

缓存：Redis

Web容器：tomcat

反向代理服务器：nginx

日志组件：log4j2

消息队列：RocketMQ

## 2.6开发环境

操作系统：Windows7或Ubuntu/centos

Java：Jdk8

IDE：Eclipse

Maven：3.3

版本控制：Git

接口管理系统：RAP

# 3.部署架构图



# 相关设计方案

## 4.1 MYSQL与ElasticSearch数据同步设计

前言：MYSQL作为存储的数据源，ElasticSearch主要作为缓存数据提供检索功能，有效提升查询性能。

### 4.1.1双写机制

重点是MYSQL和ElasticSearch的数据一致性问题。要保证数据一致性，系统架构要求数据的强一致性，即用户注册后立即可以做人脸识别。在设计上使用双写模型，用户在注册时写入的数据将同时写入MYSQL和ElasticSearch，其中一个写入失败，即注册失败，不提供消息队列补时录入方案，以保证数据的强一致性。

ElasticSearch仅作为检索数据的缓存数据库，一些不需要进行检索或检索频率低性能要求不高的数据，可以直接使用MYSQL存储而不需要提供nosql缓存。



图.数据双写机制

### 4.1.2定时任务补充检测机制

除了使用双写模型，提供定时任务每小时(或N分钟)检测一次数据表，数据表提供last\_modify\_time字段，保存该记录最后修改时间。每次定时任务扫描的是该数据表中last\_modify\_time大于上次检测任务执行的时间，扫描出的记录比如为1小时内修改的记录，再同步一次至ElasticSearch.如果没有查询出任何数据，则该次定时任务时间段内没有任何数据新增或被修改，不需要同步至ElasticSearch。

## 4.2系统日志设计

使用log4j2作为系统的日志组件。运行时开启info级别数据用于收集与查看运行日志，日志内容由业务代码产生。

Log4j2开启异步写，业务流程不需等待日志写完；打开缓冲区，减少写日志次数以提高性能。

关于日志级别，开发环境全部级别打开，因为debug级别的日志基本为底层框架的运行日志输出，编译调试；测试环境和生产环境打开INFO或以上级别的，只输出业务日志。

生产环境日志的存储介质为MySQL数据库，便于查看和分析。其他环境写文本。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | DEBUG | INFO | WARN | ERROR | 存储介质 |
| 开发环境 | 开 | 开 | 开 | 开 | 文本 |
| 测试环境 | 关 | 开 | 开 | 开 | 文本 |
| 生产环境 | 关 | 开 | 开 | 开 | MySQL |

## 4.3多线程人脸识别

