VE 商城 PC 端重构
架构设计文档

日期	版本	作者	描述
2014-12-25	V0.0.1	Coco	初始化文档

内容目录

1. 前言

1.1. 编写目的

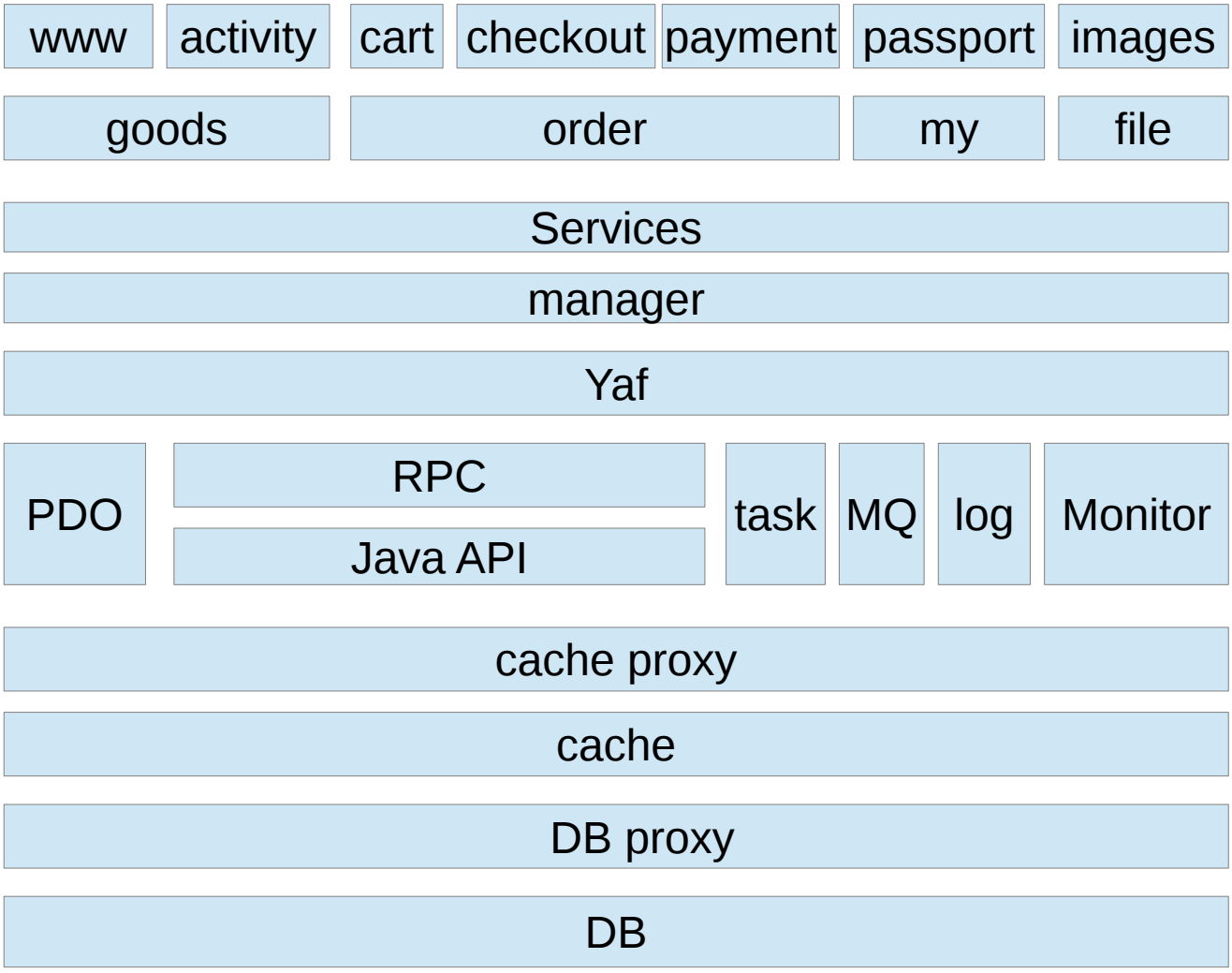
为了清晰明了的对当前 VE 商城的重构进行说明，并从基础架构，代码分层，开发规范等方面详细说明，特拟此文档，以供参考。

本文档面向 VE 商场重构项目负责人、开发成员、测试人员及其它具有查阅权限的人员。

1.2. 项目背景

VE 商城一期基于方维团购系统进行二次开发，随着公司业务的扩展与规模递增，原有系统已经不能满足当前生产需要。因此，对原有系统进行全新的架构设计与代码重构显得尤为迫切。

2. 总体设计



3. 架构分析

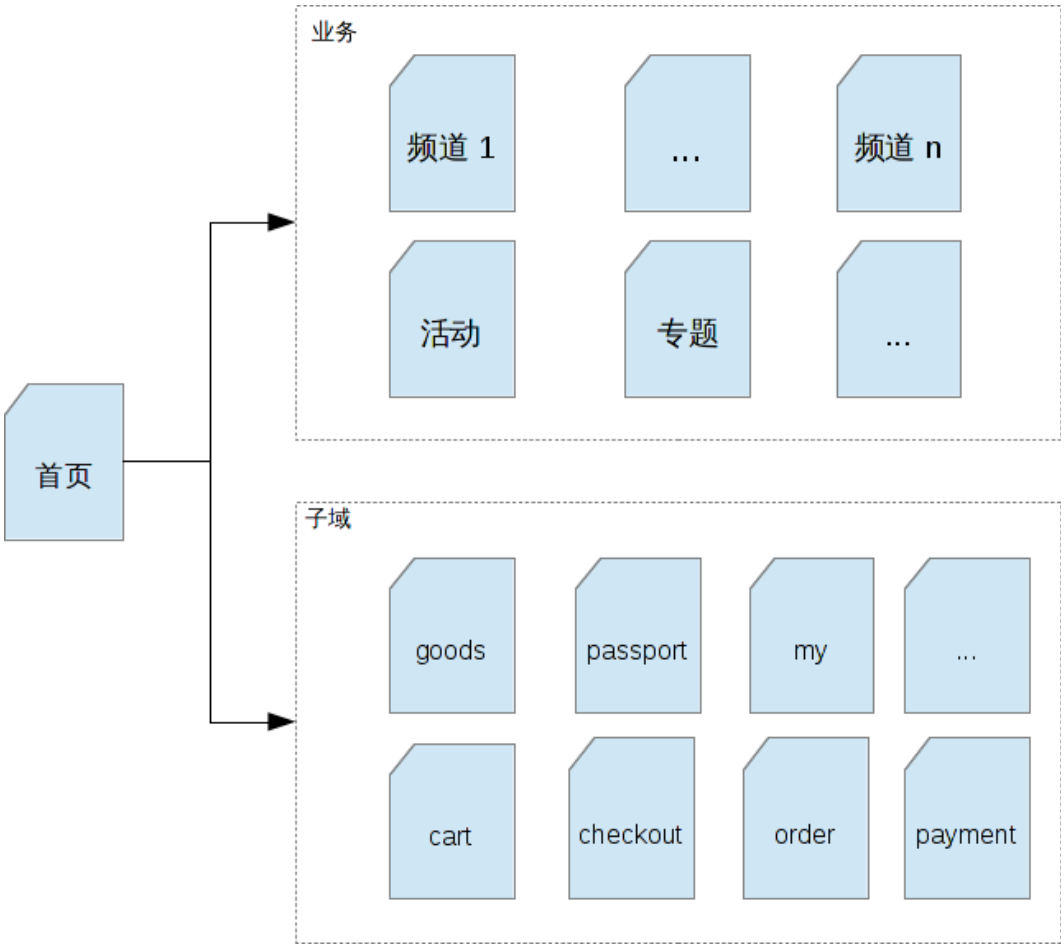
3.1. 业务分域

3.1.1. 商城展示

3.1.1.1.主域

主域即商城的 www 域。这是全站最主要的流量入口。同时也是其它各子域的引导入口。首页采用静态化，提高页面访问速度。

主域：www.ve.cn



3.1.1.2.活动专题

网站的线上运营活动是通过活动专题来进行引导的。通过各个不同的定制专题页面，进行相应的营销活动。

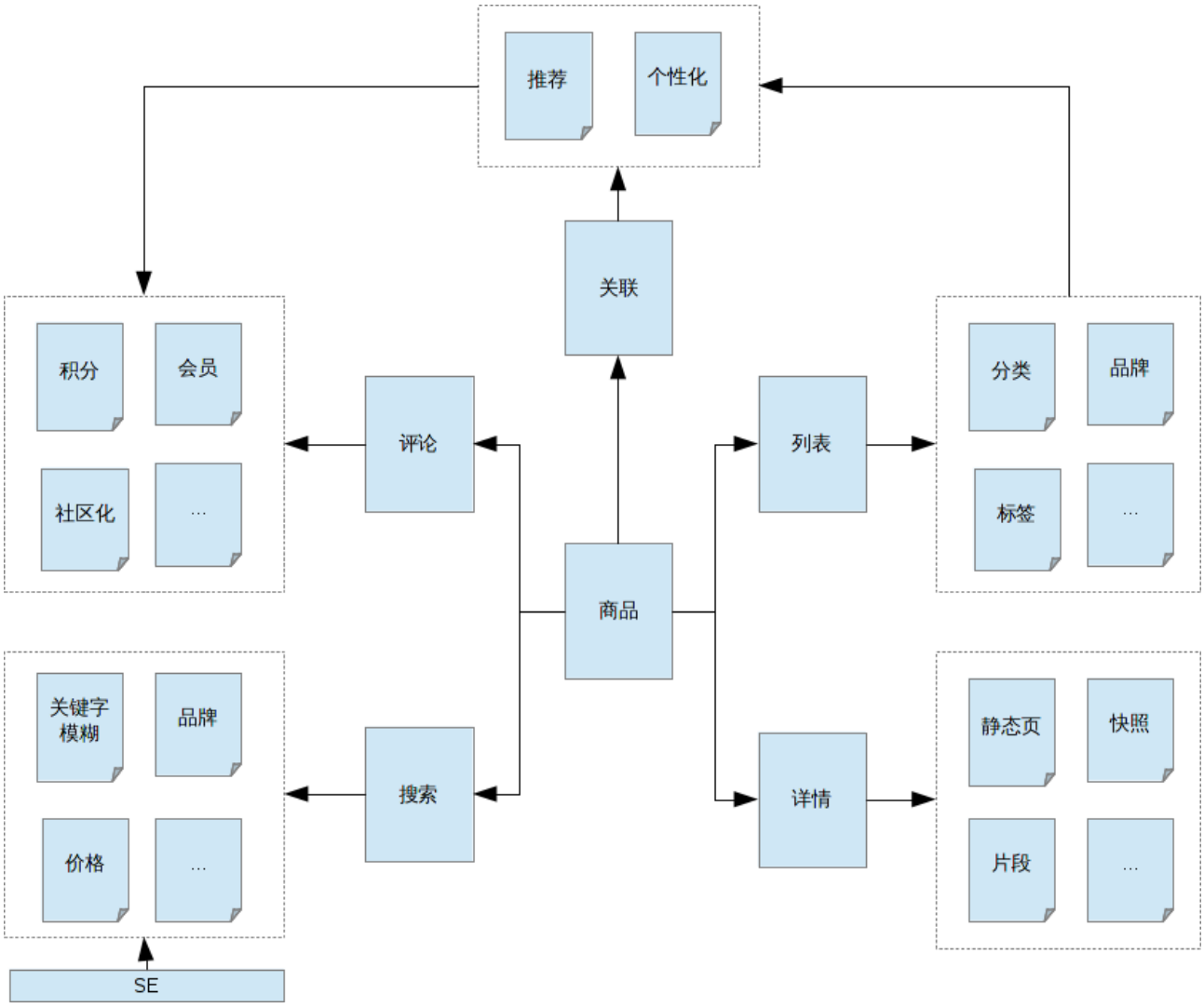
一般地，活动专题归纳在主域之下，不分配子域名。

3.1.1.3.商品展示

商品是商城中最主要的展示信息。结合商品分类、品牌、标签等属性，对商品展示进行有机的组合，

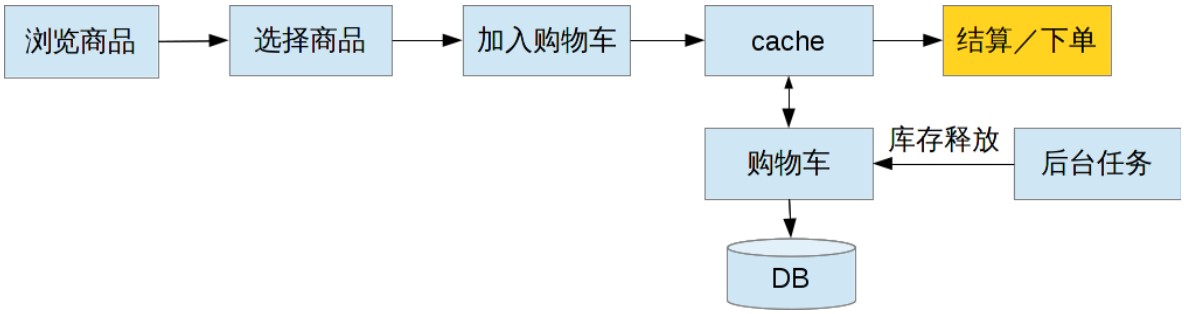
配合相应的营销策略，同时建立商品之间的内部关联与推荐算法，进一步提高商品的购买成交率。商品搜索是一个十分得要的功能，对提高用户体验有显著作用。在这方面，有大量的开源 SE 可用，如 sphinx，solr 等。特别地要注意对中文化的分词处理，选择一款合适的 SE 作为开发基础。

商品域：goods.ve.cn



3.1.2. 订单流程

3.1.2.1.购物车



购物车域：cart.ve.cn

购物车的主要功能即：

- 库存控制
- 用户购物车记录

库存扣减操作。通常有两种策略：

- 加入购物车时扣减库存。

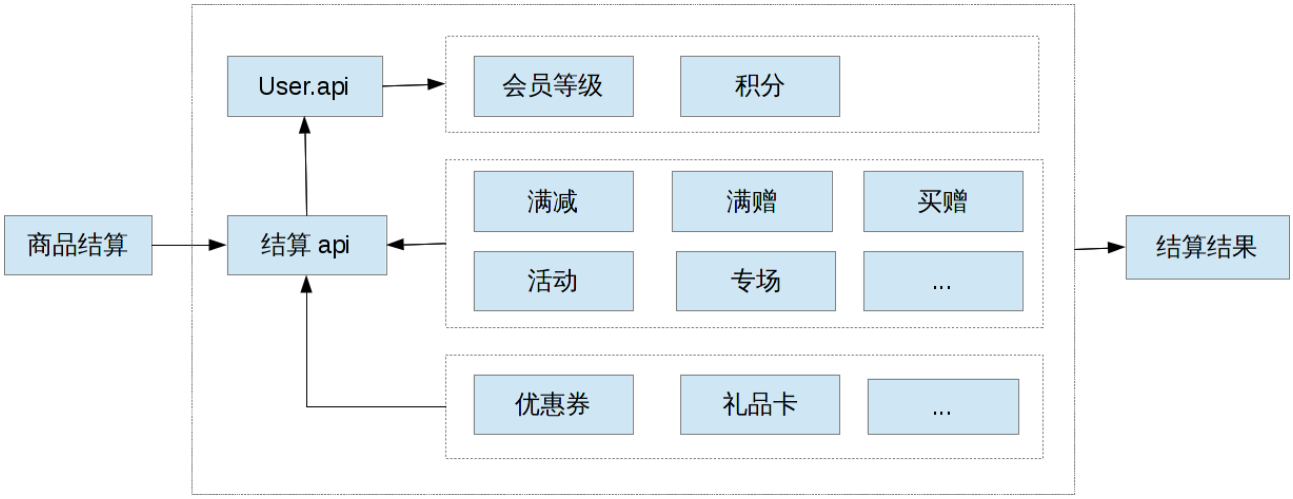
  这种方式需要通过库存缓存的严格控制来防止超卖产生。如果用 db 作为库存事务控制，无法承受高并发操作。通常选用具有事务控制的高速缓存来实现，如 redis。为了防止超时占用库存，此种方式需要在后端使用定时机制清理过期占用的库存，以释放库存数，提高成单率。

- 下单时扣减库存。

  此种方式最大的问题在于，当用户点击下单时，发现库存不足，导致流单，对用户体验有一定影响。采用此种方式购物车无须对库存进行控制。

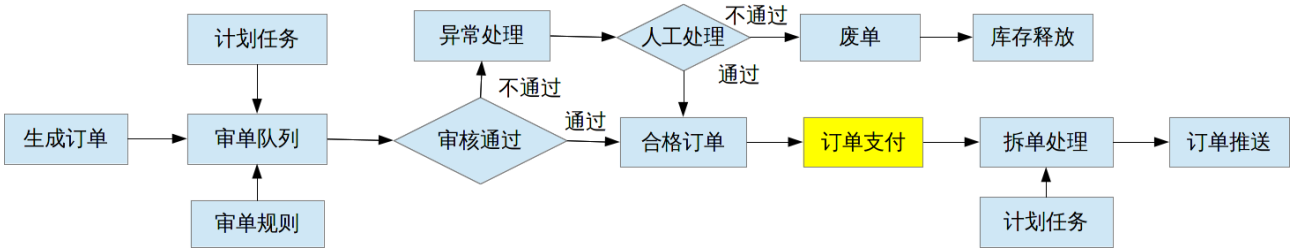
3.1.2.2.结算

结算域：checkout.ve.cn



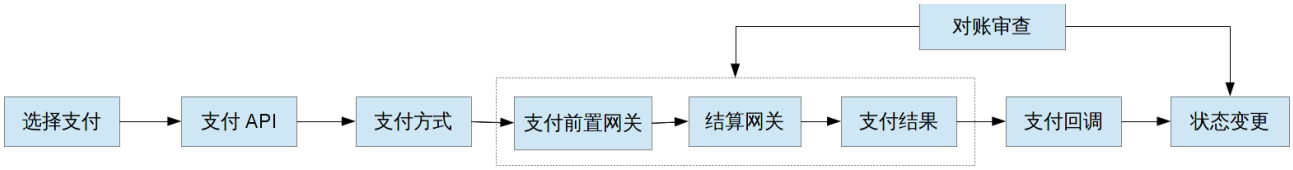
3.1.2.3.订单入库

订单域：order.ve.cn



3.1.2.4.支付

域：pay.ve.cn



3.1.3. 会员

3.1.3.1.登录注销

域:passport.ve.cn

- 登录
- 注销
- 密码找回

3.1.3.2.个人中心

- 帐号信息
- 收货地址
- 邮箱 / 手机验证
- 订单查询
- 站内消息
- 密码服务
- 优惠券
- 礼品卡
- 其它

3.1.4. 静态资源

3.1.4.1.图片 / 文件

3.1.4.2.样式

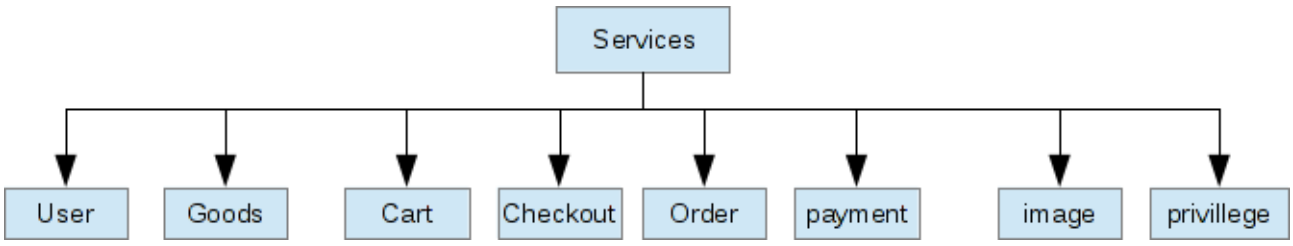
样式(CSS)与图片一样，需要放入 cdn，减少对源服务器的访问次数，以提高服务器的负载，同时加快页面加载速度。

3.1.4.3.交互脚本

交互脚本是指为页面功能服务的大量 js 脚本。基与具体的页面样式与结构无关。通常以大量的 ajax 操作为主。在单页面应用中，交互脚本的代码编写量更大，所以页面逻辑，全部由 js 进行控制。同样地，所有 js 文件，都需要放入 cdn。

3.2. 服务层

服务层是一个独立的分层，需要根据业务进行拆分。划分出不同的服务域。



3.3. 服务管理层

服务管理层对 Model 层进行管理。负责封装 model 的访问操作。按不同业务划分为 UserManger，OrderManger，GoodsManger 等等。

3.4. 基础框架 YAF

3.4.1. 目录结构

/--根

|--public 对外开放目录

| |--index.php 单入口文件

| |--skins 皮肤

| | |--basic 皮服 basic

| | | |--css 样式

| | | |--js 样式所用 js

| |--images 图片

| |--js 交互 js

|--conf 配置目录

| |--application.ini 程序主配置

|-application 应用目录

| |--components 基础组件

| | |--Controller.php

```
|  |  |--Model.php
|  |--controllers  控制器
|  |--views  视图
|  |  |--layouts  布局
|  |  |--index 默认控制器
|  |  |  |--index.php  默认方法
|  |--themes  主题
|  |  |--basic  基本主题
|  |  |  |--layouts  布局
|  |  |  |--index 默认控制器
|  |  |  |  |--index.php  默认方法
|  |  |--sunny  自写义主题 suny
|  |--modules  模块
|  |--services 服务层目录
|  |  |--Service.php
|  |  |--user  用户服务
|  |  |--goods  商品服务
|  |  |--order  订单服务
|  |--models  模型
|  |  |--manager
|  |  |  |--Manager.php
|  |  |  |--UserManager.php
|  |  |  |--GoodsManager.php
|  |  |  |--OrderManager.php
|  |  |--dao
|  |  |  |--DAO.php
|  |  |  |--user  用户 DAO
|  |  |  |--goods  商品 DAO
|  |  |  |--order  订单 DAO
|  |--library  第三方库
|  |--plugins  插件
|--logs 日志
```

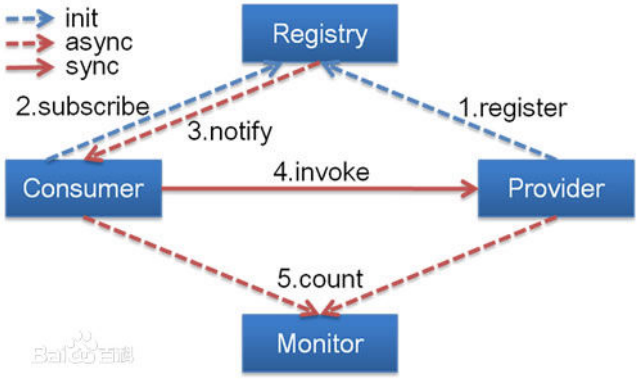
3.5. 数据访问层 PDO

PHP 数据对象 （PDO） 扩展为 PHP 访问数据库定义了一个轻量级的一致接口。PDO 提供了一个 数据访问 抽象层，这意味着，不管使用哪种数据库，都可以用相同的函数（方法）来查询和获取数据。 PDO 不不提供 数据库 抽象层；它不会重写 SQL，也不会模拟缺失的特性。如果需要的话，应该使用一个成熟的抽象层。

更多可参考：http://php.net/manual/zh/book.pdo.php

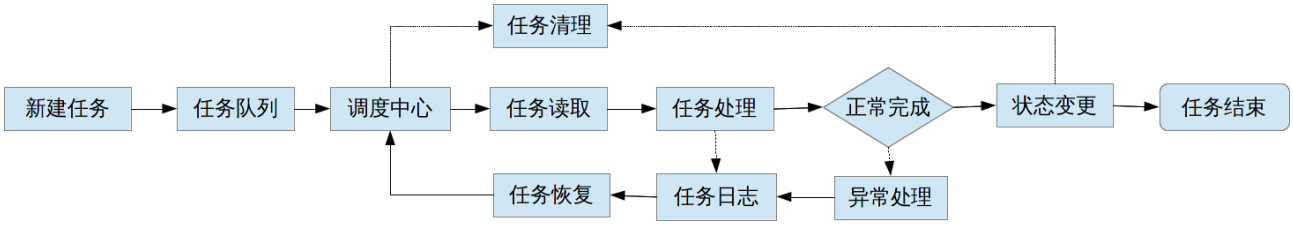
3.6. 服务 API 层

服务层 API 采用 Java 方式实现。基于 Dubbo。Dubbo 是阿里巴巴公司开源的一个高性能优秀的服务框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，可以和 Java Spring 框架无缝集成。



更多参考：https://github.com/alibaba/dubbo

3.7. 计划任务



计划任务使用 linux 上的 crontab 进行任务管理。对定时少于 1 分钟的任务，需要在程序中实现。对 PHP 来说，可以使用“死循环”方式结合 sleep 得到对秒级的任务控制。即：

```
<?php
while (TRUE) {
    // do something here
    …

    // 执行完成睡眠 5s 进行下一步处理。
```

sleep(5);

}

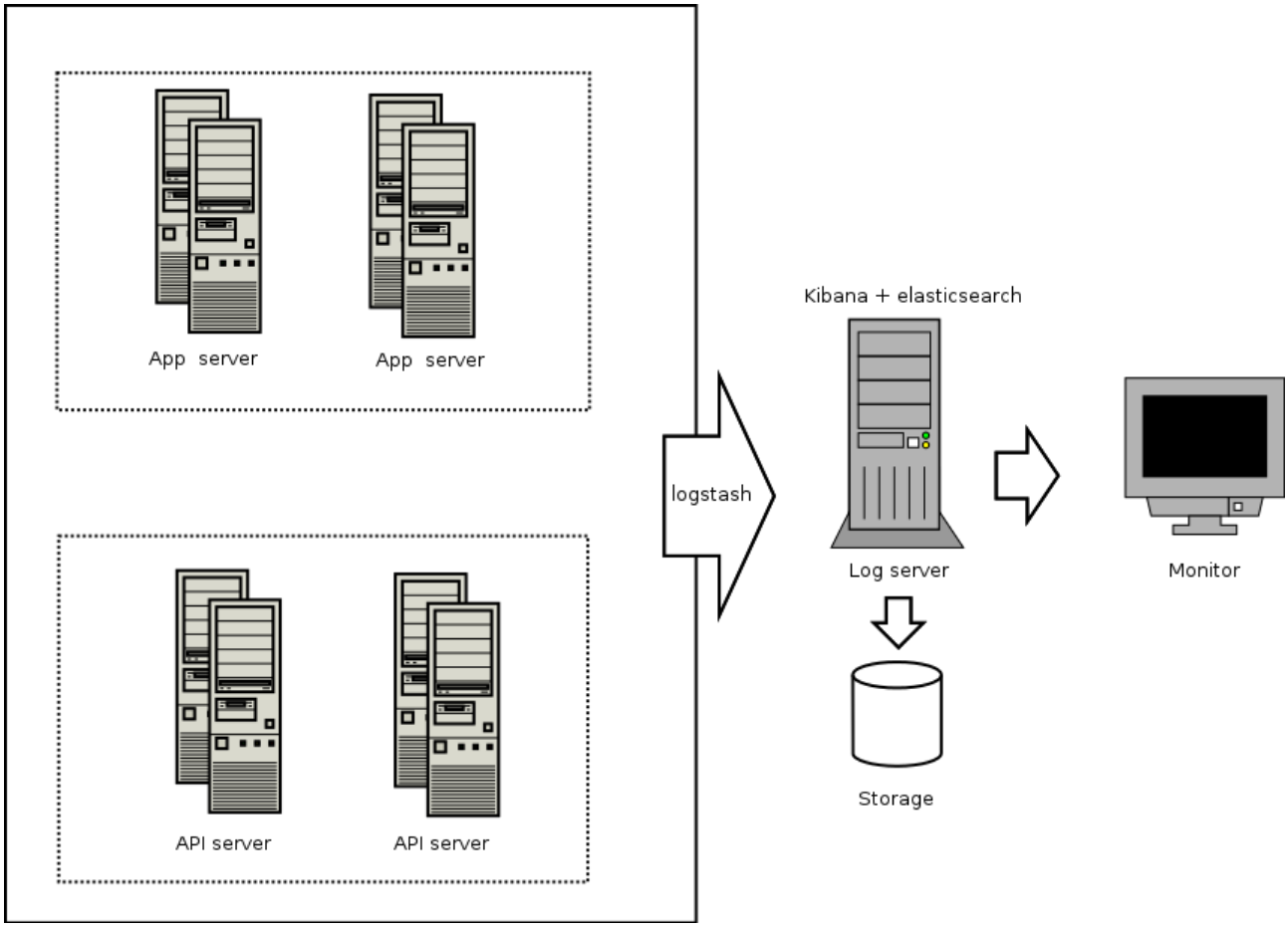
上面实现中，并不能确保真正的“每隔 n 秒“执行。因为每次处理的耗时都不一样。因此，如果要确保精确，需要换用别的方式实现。**同时，更重要的是要防止多个不同的进程之间的数据安全，防止重复处理。**

特别地，当需要处理的数据量较大时，此种方式已经不能满足性能要求。必须使用多线程进行处理。如 php pnctl，python,java 多线程方式进行。

3.8. 消息队列

消息队列（Message Queue)采用 RabbitMq 作为消息中间件，进行队列处理。通常如邮件、SMS、短消息。

3.9. 日志记录



日志平台的搭建：http://www.cnblogs.com/buzzlight/p/logstash_elasticsearch_kibana_log.html

3.10. 实时监控

服务器实时监控使用 zabbix。监控的指标有 cpu, 内存，IO，网络负载等。

更多了解：http://www.zabbix.com/

4. 功能分析

5. 数据结构

6. 附录

6.1. Yaf

官网：*http://yafdev.com/*

中文手册：http://yaf.laruence.com/manual/

6.2. Phalcon

官网：http://www.phalconphp.com/zh/