文章编号: 1000-5641(2019)04-0083-14

基于微服务的研究生培养系统的设计与实现

殷佳玲[1], 夏 帆[1], 顾 航[1], 祝 翔[1], 孙 晨[1],
王 晔[1], 董启文[1], 宋树彬[2], 傅云斌[1]

(1. 华东师范大学 数据科学与工程学院, 上海 200062;
2. 华东师范大学 研究生院, 上海 200241)

摘要: 随着信息技术的蓬勃发展, 高校研究生管理模式不断变化. 从服务对象来说, 华东师范大学现有的研究生培养系统面向的主要服务对象是学校管理人员, 但随着"以学生为中心"教育理念的深入推广, 现有系统的功能已无法满足以学生为中心的业务需求. 同时, 随着招生规模的扩大, 系统中数据的规模越来越大, 增长速率也越来越快, 研究生管理需要处理的主体由原先少量、简单的管理数据转变为复杂、大量的各类研究生教学行为数据. 基于以上原因, 采用开源框架 AngularJS 和 Spring Boot, 完成了系统的自主研发. 架构方面, 基于微服务架构以支持系统的自动化持续部署, 实现了一个为研究生、教师和管理者提供个性化服务, 并且可以快速迭代的研究生信息智慧平台.

关键词: 培养管理; 学生为中心; 微服务

中图分类号: TP 391 **文献标志码**: A **DOI:** 10.3969/j.issn.1000-5641.2019.04.009

Design and implementation of graduate student cultivation system based on micro-services

YIN Jia-ling[1], XIA Fan[1], GU Hang[1], ZHU Xiang[1], SUN Chen[1], WANG Ye[1],
DONG Qi-wen[1], SONG Shu-bin[2], FU Yun-bin[1]

(1. *School of Data Science and Engineering, East China Normal University, Shanghai* 200062, *China*;
2. *Graduate School, East China Normal University, Shanghai* 200241, *China*)

Abstract: With the development of information technology, the model of graduate student management has changed. In terms of the service target, the existing graduate student cultivation management system of East China Normal University (ECNU) was primarily targeted at school administrators. With the promotion of a "student-centered" educational concept, however, the existing system proved unable to meet student requirements. With the expansion in enrollment, the scale and growth rate of data in the

收稿日期: 2018-08-06

基金项目: 国家重点研发计划(2016YFB1000905); 国家自然科学基金广东省联合重点项目(U1401256); 国家自然科学基金(61672234, 61402177); 华东师范大学信息化软课题(41600-10201-562940/008)

第一作者: 殷佳玲, 女, 硕士研究生, 研究方向为系统开发. E-mail: 1140969322@qq.com.

通信作者: 傅云斌, 男, 博士后, 研究方向为机器学习与知识工程. E-mail: ybfu@dase.ecnu.edu.cn.

systems are ever increasing. The data used by the graduate student management system has changed from a small set of simple data to a large set of complex data on teaching behavior. Based on the above reasons, ECNU has adopted an open source framework, such as AngularJS and Spring Boot, to complete the system's independent development. In terms of architecture, the system aims to use micro-services architecture to automate continuous deployment for developing an intelligent graduate student information platform that provides graduate students, teachers, and administrators with personalized services and can be quickly iterated by developers.

Keywords: cultivation management; student-centered; micro-services

0 引 言

随着信息技术的蓬勃发展, 研究生招生规模不断扩大. 我国各大高校纷纷响应国家教育部号召, 基于"以学生为中心"的教育理念[1], 推进信息化建设. 在高等教育繁荣发展的形势下, 华东师范大学(简称华东师大)也积极开展了研究生教育平台数据管理系统的研发. 华东师大现有的研究生系统于 2005 年开始筹建, 经软件公司开发, 实现了包括招生、学籍、培养和学位等 4 个子系统的研究生培养系统; 然而, 由于目前研究生培养和管理的很多业务逻辑及流程已经发生了较大的变化, 现有的研究生系统使用至今, 已出现明显的不足. 经深入调研, 现有研究生系统主要存在以下问题.

(1) 系统代码耦合较大, 维护成本高. 现有系统集成了学籍、培养等诸多服务, 导致代码中不同模块的耦合度较高. 如, 现有的学籍表包含了 200 多个字段, 已经难以对其表结构进行修改.

(2) 系统错误隔离性低. 从开发角度看, 一个服务的 bug 可能导致其他服务出现异常. 从运维角度来看, 不同服务运行在同一个系统实例中, 某个服务的故障可能会导致整个系统瘫痪.

(3) 系统迭代代价高, 其功能远滞后于实际培养制度. 培养系统的代码库极其庞大和复杂, 开发人员难以快速理解并掌握系统代码, 既使微小的改动也无法由华东师大自己的开发团队实施; 若是通过软件开发公司进行升级, 除了涉及环节多, 系统的复杂性也延长了开发周期.

(4) 现有系统的可扩展性低, 性能上已无法满足用户的正常使用. 在设计之初, 开发人员对系统的扩展性考虑不周; 而随着系统中数据规模的越来越大, 数据增长速率越来越快, 系统的性能已开始无法满足用户的正常使用.

(5) 从功能角度来看, 现有的系统在实际使用中, 存在着浏览器兼容性差、不支持移动端访问等问题.

(6) 从业务角度来看, 现有的系统已经难以快速响应国家教育部提出的新数据需求, 导致大量信息缺失和数据的不一致, 相关职能部门不得不花费大量的时间与精力, 来维护信息数据.

针对以上问题, 无论从业务角度还是功能角度, 都必须从提高研究生数据质量和信息管理质量出发, 以灵活收集、组织、存储、利用教学管理数据为目标, 基于微服务[2]模式, 完成新一代研究生管理系统[3]的设计与实现. 为此, 华东师大研究生院于 2016 年 8 月组建了研发团队, 基于敏捷开发的模式开发新一代研究生培养系统, 迄今已完成了学籍和培养两个子

系统. 本文旨在以培养业务为切入点, 研究如何利用成熟的主流开源软件框架来打造一款以学生为中心, 同时又具备高效、高可扩展、高弹性的新型研究生培养系统. 具体来说, 本文的主要工作如下.

(1) 与华东师大研究生院相关业务人员沟通与交流, 定义培养系统中的领域模型, 梳理培养业务的详细需求, 提供方便管理人员进行交互的数据接口.

(2) 依据前后端分离开发[4]的方式以及 REST[5]风格的服务接口设计原则, 确定后台暴露的接口以及使用方式.

(3) 在微服务的技术架构下, 将系统拆分为培养、通知等多个相对独立的微服务, 确保每个服务专注于各自的职责领域, 以降低服务之间的关联耦合程度; 使用网关模式以支持不同客户端; 实现自动化部署和持续集成, 从而保障系统可以持续、快速、便捷地融入新功能.

本文后续内容组织如下: 第 1 节讨论相关工作; 第 2 节给出研究生培养系统的业务设计; 第 3 节叙述系统架构; 第 4 节展示系统的实现; 第 5 节展示部分模块的性能实验及测试结果; 第 6 节总结全文.

1　相关工作

近年来, 随着移动终端用户的迅猛增长, 手机逐渐成为继电视、广播、报刊、互联网之后的全新媒介形式. 传统的信息系统正在积极向移动端方向发展, 除了原生的 Android 和 iOS 的应用, 新一代研究生培养系统基于 Web 3.0 的软件设计理念, 充分考虑了本系统在移动端的展示, 从而打造一款适应 PC 端与移动端的高可用的研究生信息平台. 架构方面, 现有的大部分研究生系统, 包括华东师大现有的研究生系统, 主要表现为如图 1 所示的传统 MVC(Model View Controller)3 层架构的单体式架构[6]. 这种单体式应用通常将所有服务集成于一个系统[7]中. 但由于系统代码库的规模过于庞大, 并且耦合度高, 因而难以维护和更新. 从运维角度出发, 这类系统虽然部署简单, 但是错误隔离性差, 可扩展性低.

随着微服务架构的持续火热, 越来越多的人开始关注与思考如何从单体架构迁移到微服务上. “微服务”一词最早出现在 2011 年 5 月在威尼斯召开的软件架构研讨会上, 它克服了单体式应用架构的诸多缺陷, 已经在 Netflix、Amazon 等知名互联网公司中得到了广泛应用. 基于微服务架构的系统拥有一些显著的特征: 常常基于领域驱动开发的原则, 将系统拆分成多个低耦合的微服务; 系统开发采用分布式治理的原则, 不同服务可以采用不同技术开发, 可以自主地管理自己的数据; 都有一套完整的自动化基础设施以支持持续部署, 从而保证新功能能够快速正确地加入生产环境.

具体来说, 新一代研究生系统前端样式采用 Bootstrap 框架提供的样式风格, 实现 PC 端与移动端的自适应布局; Java Script 采用开源的 AngularJS, 弥补 HTML 本身在 Web 构建应用方面的缺陷; 后端采用 Spring Boot, 用于简化新型 Web 应用的开发过程; 数据库方面, 采用关系型数据库 MySQL 作为存储, 并使用 Spring 提供的 Spring Data JPA 来简化数据库访问. 而本系统的构建, 采用微服务架构, 实现自动化持续部署, 可根据研究生院的实际业务情况, 进行快速迭代, 方便开发人员独立、快速、便捷地完成系统升级发布流程, 利于系统的更新迭代.

2　业务设计

从业务角度来看, 新一代的华东师大研究生培养系统根据学校业务需求, 在学籍系统的基础之上建立整体业务. 整体业务如图 2 所示, 主要分为以下 5 大模块: 培养过程模块、课

程管理模块、考试–成绩管理模块、课程质量管理模块和通知模块, 其中, 学籍系统提供学籍库, 后勤系统提供排课管理的教室库.

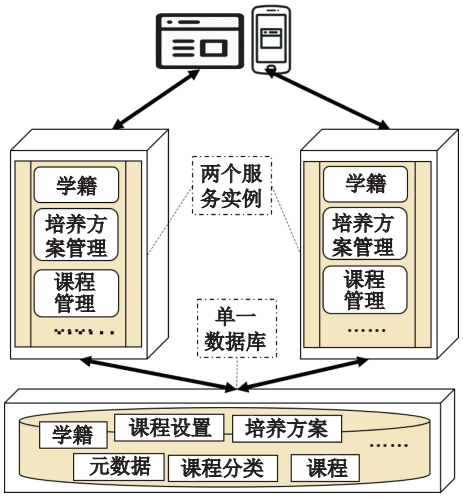

图 1 传统 MVC 3 层架构的单体式架构

Fig. 1 Traditional architecture with three layers of MVC

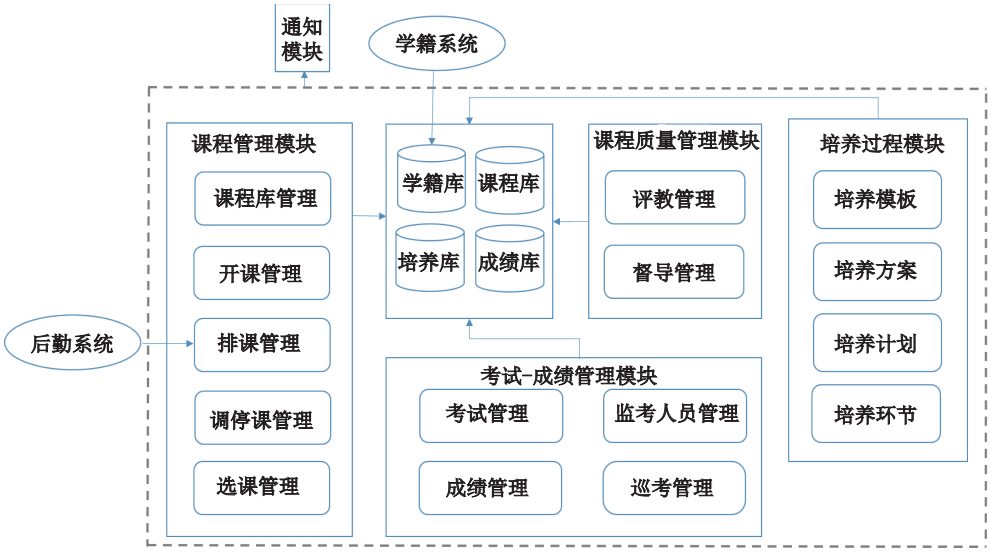

图 2 培养系统的整体业务图

Fig. 2 The overall service chart of the cultivation system

2.1 培养过程模块

培养过程主要包括培养模板、培养方案、培养计划以及培养环节: ① 培养模板是由研究生院制定维护, 整个培养过程都是基于该模板来维护的. ② 培养方案是以培养模板为依据和指导, 由院系按需细化不同维度(一般按照二级学科)制定培养方案, 其内容包括具体描述不同课程类别中的课程(如学位公共课、学位基础课、学位专业课), 以及培养过程中各培养环节(如学术报告、资格考试、中期考核). 培养方案的制定或修改, 是在研究生院组织下, 由院系具体实施, 通过研究生院的审核, 由院系研究生秘书录入培养系统. ③ 培养计划中的课

程是由学生根据个人兴趣, 从各自培养方案规定的课程范围中, 自主选择, 从而定制个性化的培养计划. 其中, 公共课由研究生院设置; 学位基础课、专业必修课由院系设置; 专业选修课、跨专业课程、公共选修课由学生自主选择. 需要注意的是, 培养计划由院系统一设置框架, 学生在导师的指导下进行课程选择, 培养计划的确定必须经导师审核. ④ 培养环节是为了强化研究生培养过程的规范和考核, 一般包括论文开题、中期考核等相关量化的学习过程和成果.

相比于现有的培养系统, 新系统主要完成了以下工作: ① 提供按院系、学科等多种维度进行培养方案的维护、审核, 以加强课程修读、培养方案的规范管理; ② 根据院系研究生秘书指定的维度, 自动将每个学生分配到相应的培养方案中去, 然后学生选择自己感兴趣的课程, 完善各自的培养计划(本功能在线上实现了原来人工完成的繁琐业务逻辑, 通过系统完成了大部分院系研究生秘书的工作, 大大减少了他们的工作量); ③ 基于实际业务需求, 实现了培养办、院系、导师、学生分权限管理的功能, 利于各职能部门的协作配合.

2.2　课程管理模块

课程管理主要包括课程库管理、开课管理、排课管理、调停课管理和选课管理. 系统中涉及的所有课程都保存在课程库中, 课程资料由院系研究生秘书或教师填写, 经研究生院审核后, 系统自动生成课程编码, 并加入课程库; 若某课程被禁用, 则该课程的编码将会失效. 通过对课程库的中心化管理可以方便地维护系统中所有课程. 开课是由研究生院或院系从课程库中选择相应的课程, 进行开课设置. 研究生院或院系需在设置的排课时间段内, 设置相应的课程、时间、任课教师、上课人数, 并依据后勤系统提供的教室库, 合理规划上课地点完成排课. 随后, 学生在一定时间段内, 依据经过导师审批的培养计划, 对已经开放的选课课程进行选课、退课等操作, 同时, 各职能部门可以收集、分析学生操作的记录数据. 当遇到一些突发事件, 任课教师可以使用调停课功能, 在研究生院设置的调停课提前天数内, 进入系统申请调停课, 经院系、研究生院审批后, 将调停课相关信息推送给院系、上课学生、申请教师、课程相关督导.

新系统的设计实现了如下几个要点: ① 建立了安全可靠的课程库, 并使用和本科生信息管理系统一致编码规则的课程编码, 方便了两个系统之间的数据交互; ② 根据具体需求, 个性化地设计开课, 并且在系统的开发和部署上, 使用不同的策略来避免选课高峰期出现服务响应缓慢的情况, 保证了选课数据的安全可靠; ③ 将课程管理与通知模块关联, 通过自动发邮件的方式将调停课申请、审批结果及时通知用户, 让用户和系统随时、紧密地交流.

2.3　考试–成绩管理模块

考试–成绩模块主要包括考试管理、成绩管理、督导管理、巡考管理. 考试是由开课单位在研究生院设置的考试日期内, 完成该开课单位下本学期开课课程的考试安排; 考试管理还包括考场情况管理, 即开课单位根据监考人员提交的考场情况, 将考场情况录入系统进行管理; 开课单位可以查询、统计本单位所开课程的考试信息, 研究生院则会保留所有考试的查询、统计等功能. 考试成绩的录入是授课教师在规定的录入时间段内, 录入该教师本学期所开课程的学生考试成绩, 相关的院系研究生秘书需在授课教师录入完后审批所有成绩, 审批完成, 任何人不能修改成绩; 院系研究生秘书也可代授课教师录入成绩.

新系统的设计有如下要点: ① 为方便考试管理, 公共课考试由研究生院、公共课开课单位、院系组织实施, 而专业课只由院系管理; ② 系统根据考试安排向监考老师和学生推送考试安排; ③ 建立常用监考人员库、巡考人员库; ④ 采用数据加密、日志收集、专人管理

等多种措施确保成绩数据的安全可靠; ⑤ 提供中英文成绩单打印的功能, 如实记录考试成绩.

2.4 课程质量管理模块

课程质量管理包括课程评教和督导管理这两个部分. 课程评教是保障学生课程质量的重要手段, 其实施过程是, 研究生院根据指定维度(如课程类型、院系等)选择需评教的课程, 设置评教时间; 参与评教课程的学生按要求完成评教, 院系和教师无法查看学生具体的评教分数. 督导管理主要是对研究生院安排督导(学校或院系领导)听课的管理, 以及管理、统计督导对课程质量的评价.

新系统根据上述需求, 基于现有系统, 做了如下改进: ① 建立督导库, 由研究生院管理该库; ② 将每位评教的学生匿名处理, 确保评教的透明实施; ③ 根据权限按院系、教师、学生等不同角色来区别显示评教、督导评价的数据; ④ 按分数、学期等维度, 生成学生、督导评价的相关统计数据, 并按分数段展示统计图表等等.

2.5 通知模块

本文团队调研整个系统发现, 一旦有学生或其他角色提交了需审核的信息, 管理人员根据业务流程进行层层审批方能完成整个流程. 然而, 这一过程通常会由于审批人员不能及时得知系统中有亟待审核的信息, 而造成审核不能尽快完成, 导致学校职能部门办事效率降低, 申请人需等待很长时间才能得知申请结果; 如果申请人有较紧急的申请需要审批, 还得采用发邮件、打电话等方式来催促职能部门完成审批.

基于上述情况, 新系统设计并实现了一个通知模块, 以灵活支持系统中的消息定制发送. 由于今后其他子系统的开发也需以此来通知相关人员, 新系统则将通知服务与培养其他服务基于微服务架构拆分, 独立部署, 并使用单独的数据库系统, 实现了模块间的低耦合, 降低了维护成本.

3 系统架构

系统的建设旨在规范收集、组织、存储、利用、管理相关职能部门、学生、教师的教育数据, 并起到提高工作效率的重要作用. 如果系统无法根据实际业务需求, 进行高效的演变, 这势必会降低用户体验, 并给用户带来额外的工作量, 这不仅不能达到建设的初衷, 甚至会阻碍业务流程的规范与提升.

因此, 搭建一套高效、高可扩展、高弹性的系统来支撑研究生教育的高速发展就显得极为重要. 经过技术调研与深入探究, 完成了新系统的总体架构, 如图 3 所示. 下面从 5 个方面来介绍新系统的架构.

3.1 基于微服务的业务架构

在微服务的架构下, 采用用例拆分的模式, 以学生为中心, 从学生视角组织各类用例, 确定恰当的服务边界, 将整体系统拆分成培养管理、通知管理等多个微服务. 这种拆分方式确保了每个服务专注于各自的职责领域, 降低了服务之间的关联耦合. 由于每个微服务的代码规模相对较小, 并且服务之间的耦合程度较低, 则对微服务进行升级相对容易. 并且, 在系统进行部署运维时, 每个微服务均可独立部署, 系统根据服务的实际工作负载, 可以动态调整服务的实例数量. 因而采用微服务架构的培养系统能够适应目前华东师大的高速发展, 并且利于系统的持续集成.

由于微服务架构下划分出的多个微服务实际是由多个不同的小应用来提供, 且不同的小应用具备完全独立的前端 Web ui、控制层、逻辑层和数据库访问层, 因而可以自主管理

独立的数据, 也可以采用不同技术开发, 这就提高了面对多种遗留系统时开发的兼容性. 新系统根据网关模式设计了针对不同使用场景的网关,如图 4 所示. 采用分布式治理和应用网关, 可以自主地管理自己的数据并灵活通信, 提高了各个小应用之间数据通信的灵活度, 也提高了系统对多种访问客户端的支持. 具体来说, 通过 Nginx 接收每个用户的访问请求, 并把路由转发给当前工作负载最低的 Web 前端服务器; 当用户发送如前缀为/api/notification 的请求时, 则是表明调用通知子服务相关的业务模块, 接着通过网关转发, 调用通知子服务的业务逻辑, 并以 JSON(JavaScript Object Notation) 格式返回给 Web 前端, 用户从 Web 前端可以看到请求的结果.

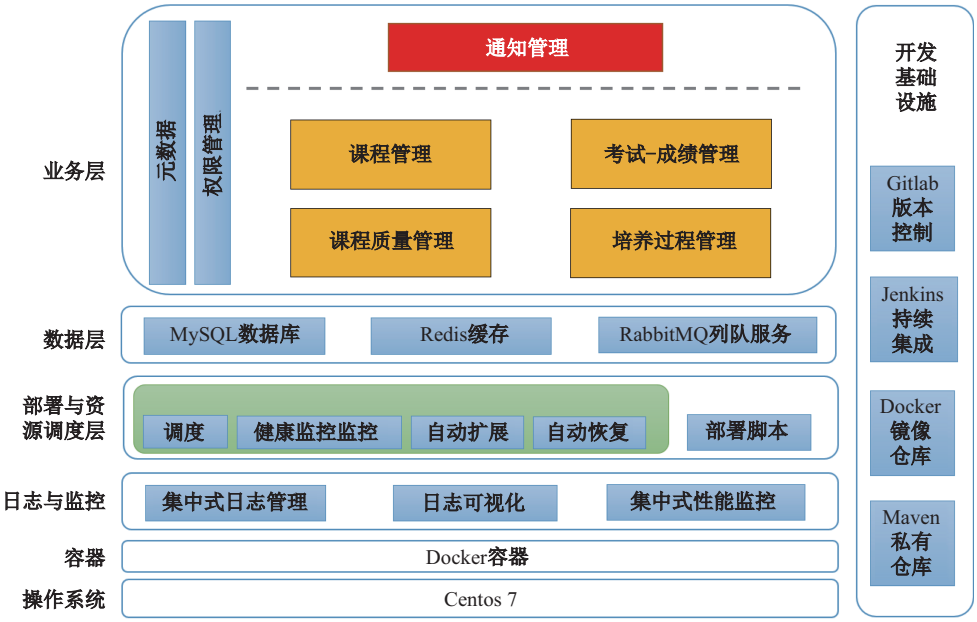

图 3　新研究生培养系统的总体架构

Fig. 3　The overall structure of the new graduate student cultivation system

新系统还整合了 REST(Representational State Transfer) 风格和消息订阅发布机制[8]. REST风格强调资源的概念, 设计出的 URI 结构清晰, 易于理解, 如/students/xxx/study-plan 可用于标识学号为 xxx 的学生的培养计划. 因而, 采用REST风格进行系统整合, 能实现较低的请求延迟, 但其容错性低. 信息订阅发布机制虽然增加了请求的延迟, 但将不同服务之间解耦, 使得服务能够自主控制其服务速率. 具体而言, 以调停课管理为例, 通知服务和培养服务通过以下两种方式进行: ① 消息订阅发布从培养服务那里获得调停课申请, 通知服务通过 RabbitMQ(消息中间件)获取培养服务推送的调停课申请; ② 通知模块调用其他服务提供的 REST 接口, 获取相关的数据.

3.2　数据层

新系统选用 MySQL 作为存储数据库, 但传统 MySQL 的主从模式不能保证数据一致, 会导致数据紊乱等问题. 为了避免这种情况, 数据库集群集成了 multi-cluster 的 Galera[9]插件. 这是因为, 在新系统中, 某些业务对数据有高一致性的要求, 因而使用新型、高冗余的 Galera 架构方式, 即集群上的节点都可以作为主节点. 因此在多个节点写入时, 能够保证整个系统的数据一致性、完整性与正确性. 系统还使用了 Redis 缓存 Session、菜单等数据, 以

此加快读写速度, 并支持数据的持久化.

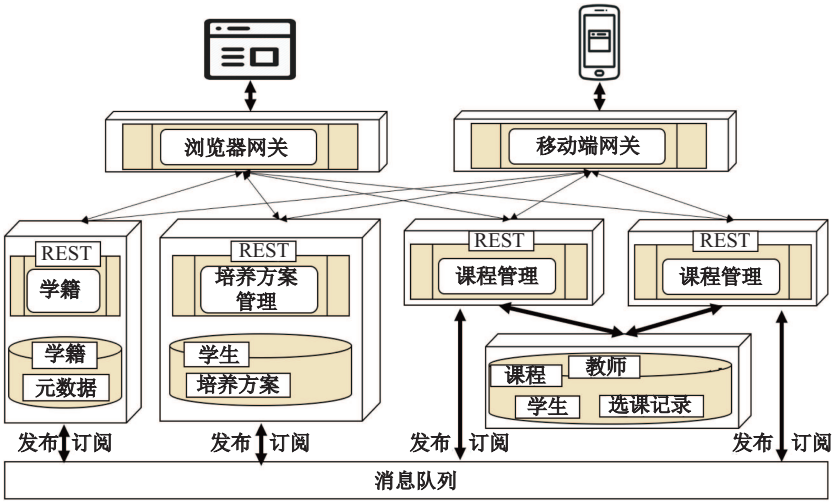

图 4 基于微服务研究生培养系统的架构

Fig. 4 The architecture of the graduate student cultivation system based on micro-services

3.3 部署与资源调度层

新系统中, 采用 Kubernates[10]来管理 Docker 容器组, 有效地实现了自动化部署, 从而保障了系统能够快速迭代并且平滑部署上线. 但在实际使用过程中, 经日志分析发现, 系统在用户使用高峰期, 服务响应缓慢甚至崩溃, 如选课时段. 针对以上问题, 根据 Kubernetes1.3 后新特性——Pod(一个或多个 Docker 容器组)水平自动伸缩(Horizontal Pod Autoscaling, HPA), 新系统的部署过程改进了配置, 即当业务需求增加时, 系统将无缝地自动增加适量容器, 并在不需要的时候缩容, 以便节省成本. 具体来说, 通过创建 HPA 用于控制 Pod 副本在 1 到 10 之间, HPA会通过自动增加或减少副本的数量, 来保持所有 Pod 的平均资源利用率在 50% 以内. 通过这一配置, 动态调整了负载, 新系统在服务繁忙时也能迅速响应用户的请求, 提高系统稳定性, 极大避免了峰值情况下服务崩溃的问题.

3.4 日志与监控层

系统还部署构建了一套完整的日志系统——ELK, 用于集中管理日志. ELK 是 3 个开源软件的缩写, 分别表示Elasticsearch、Logstash、Kibana[11]. 具体过程: 先使用 Filebeat、MetricBeat 收集日志, 并传送给 Logstash, Logstash 将日志过滤解析转换, 再一并发给 Elasticsearch; Elasticsearch 分析存储日志, 最后使用 Kibana 为 Logstash 和 Elasticsearch 分析的日志结果提供友好的图形化界面, 将数据展示汇总, 起到实时检测、保证安全、发现 bug 的作用.

3.5 开发基础设施

自动化基础设施包括如 Git 等代码版本控制技术、Jenkins[12]等持续集成工具以及 Docker 等容器技术. 具体过程: 当发布一个新的版本时, 开发人员首先使用 Git 创建标签推送至 GitLab, 如 git tag 版本号; GitLab 会触发 Jenkins, Jenkins 则会通知执行服务器有需要发布的任务; 接着执行服务器从 GitLab 获得最新分支,前端使用 npm, 后端使用 Maven, 执行编译、测试、构建, 最后打包成 Docker 镜像推送给 Kubernetes.

4　系统实现

由于篇幅有限, 本节以通知模型、培养过程模型、开课选课模型为例, 对培养系统的设计与实现进行详细说明.

4.1　通知模型

培养系统中的某些功能, 例如各类时间设置、调停课审核等, 需要根据业务的流程进展对相关人员发送通知. 以开课时间设置和调停课审核为例来说明具体的业务流程.

第一, 在研究生院管理人员设置开课时间之后, 向相关开课的管理人员发送邮件告知开课时间节点.

第二, 在调停课审核进入“审批中”的状态之后, 通知负责审批的人员前往审批, 并告知提交申请的教师审批进展, 如院系秘书审核通过、等待研究生院审核相关的信息.

第三, 根据调停课审批流程, 申请被批准或者被否决之后, 通知提交申请的教师审批结果, 同时也向审批所涉及的相关人员发送消息, 如已经安排督导的课程申请调停成功、需向负责的督导发送通知消息.

为了能够支撑以上需求, 通知服务实现了两个关键功能, 以支持灵活的消息定制功能: ① 设计灵活的消息模板, 即在消息模板中设置当前角色下能够访问的变量, 例如学生姓名、学号、转入院系的信息等等; ② 定义灵活的通知人群, 设计并实现一个能够从当前系统中获取学号、工号、院系等信息的机制, 结合消息模板配置的通知角色, 定位需要发送通知的用户.

针对不同业务需求, 通知模块的邮件发送策略是: 第一, 对于紧急消息(消息给定的截止时间在某个范围内), 直接通过邮件推送给用户; 第二, 对于非紧急类消息, 先保存在数据库中; 第三, 用户通过 quartz[13] (作业调度框架), 可以自己定义消息推送的时间间隔, 每当超过这个时间间隔时, 通知模块判断是否有邮件需要发送给用户.

通过上述设计, 可配置的消息模板管理可以满足潜在的业务变化, 对于其他需通知的业务, 如培养环节时间预警, 可以复用该模块, 降低代码量, 实现高内聚低耦合的目标, 提高系统的可维护性.

4.2　培养过程模型

培养过程要求新系统将以往人工维护培养方案、培养计划的业务转移到线上, 具体来说, 就是, 在系统中提供按二级学科(或更细维度)维护培养方案, 并完成培养方案到学生培养计划的映射.

新系统的实现步骤: 先完成培养模板的维护; 然后培养方案会完全复制培养模板; 最后各院系从培养方案到培养计划的分配策略, 细化培养方案的维度(如二级学科、专项计划等), 从而实现培养计划的自动化分配. 具体来说, 如图 5 所示, 实现了一个抽象类 (Abstract) Cultivation, 它关联 Discipline(学科)、EducationMode(学习形式)、SpecialPlan(专项计划)、Unit(院系)、CultivationProcedure (培养环节)、CourseSpecfication (课程详情)、TextSpecification (描述小结详情); 主体业务 CultivationTemplate (培养模板)、CultivationScheme (培养方案)、CultivationPlan (培养计划) 分别继承 Cultivation, 其中, 一个 CultivationTemplate 可以被多个 CultivationScheme 关联, 一个 CultivationScheme 可以被多个 CultivationPlan 关联.

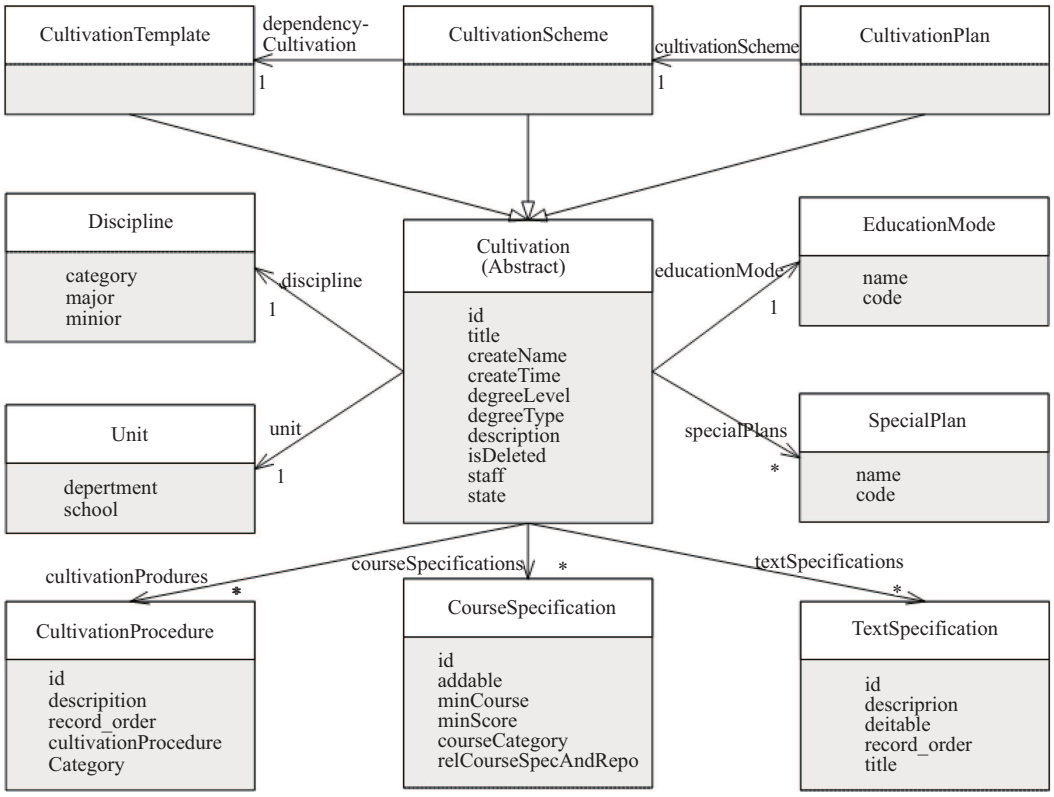

图 5 培养过程模块类图

Fig. 5 Class diagram of the module for the cultivation process

4.3 开课选课模型

开课管理要求系统为相关角色提供功能界面, 功能流程图如图 6 所示, 具体的功能说明及角色说明如表 1 所示. 需要达到的新需求: 第一, 根据研究生院审核通过的培养计划中的课程, 系统自动生成开课的课程、预估人数, 以供研究生院、院系完成开课排课; 第二, 不同类别的课程负责单位不同, 公共课开课由公共课开课单位(如哲学系、对外汉语学院)负责, 专业课开课由院系负责; 第三, 调停课审核的步骤可以根据实际需求增加或减少. 对于第一个需求, 新系统设计了给相关职能部门用户提供开课课程的推荐功能; 对于第二个需求, 新系统设计了完整的权限功能, 管理人员可按需配置各类用户的权限; 对于最后一个需求, 新系统实现了 3 个实体类: AuditRecord(审批记录)、CourseRequest(调停课申请)和用于配置审批步骤的 AuditFlowStep(审批步骤). 实现了可配置的审批流程, 这种设计的好处在于其他功能也能复用 AuditRecord 和 AuditFlowStep 进行审批.

从图 6 可知, 基于培养计划开课的具体实现是, 先根据经导师审批通过后的每个学生的培养计划, 生成向研究生院管理人员或院系研究生秘书推荐的课程以及预计上课的人数; 之后, 研究生院管理人员或院系研究生秘书根据推荐的信息, 编辑开课计划的相关信息, 如上课时间、学生分类配额(本科生、留学生占比)等等, 上课地点则是归入到排课管理中, 教师可以进入开课查询中查询本学期所开课程.

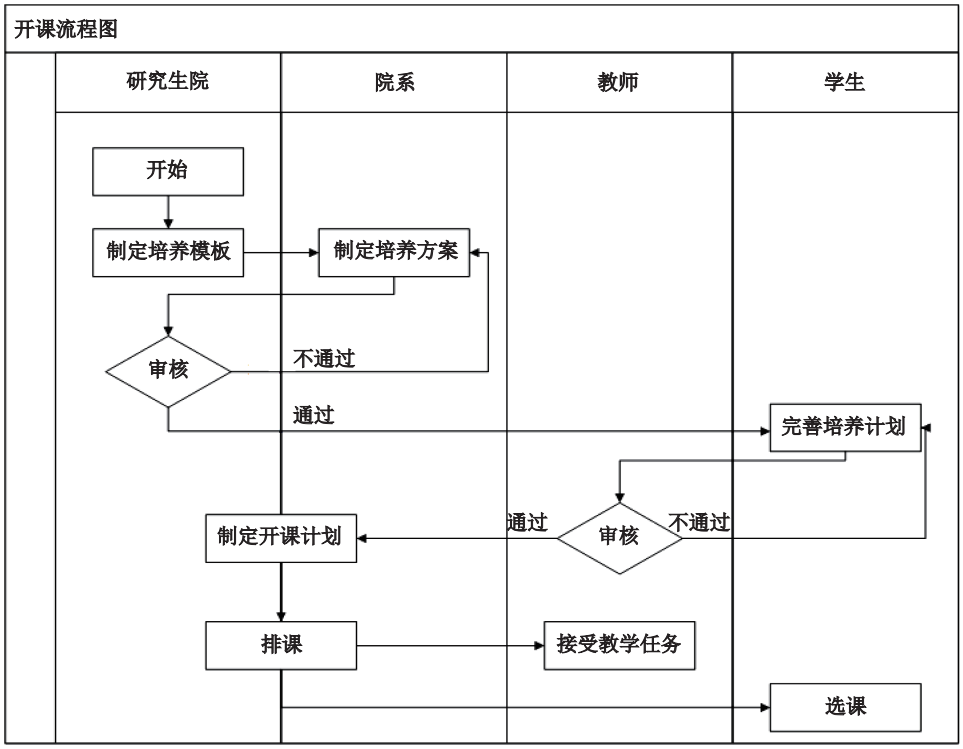

图 6　开课管理流程图

Fig. 6　Flow chart for starting new courses

表 1　开课管理的功能说明与角色说明

Tab. 1　Description of functions and roles for starting new courses

任务	内容	角色	角色所在部门
开课设置	设置开始时间、结束时间	相关老师	研究生院
根据培养计划开课	根据培养计划设定的课程生成开课计划	相关老师	研究生院、院系
根据课程库开课	人工设定开课计划	相关老师	研究生院、院系
开课查询	查询已开课程情况	相关老师	研究生院、院系、教师
排课管理	根据已开课程进行排课	相关老师	研究生院、院系

选课是新系统中学生角色最为重要的功能. 新系统的学生选课、功能说明及角色说明如表 2 所示, 主要表现为: ① 选课的管理, 需确保选课数据的安全可靠; ② 研究生院管理人员设置选课的各类时间节点, 如第一轮选课时间、第二轮选课时间、退课时间等; ③ 为鼓励学生尽早完成选课, 采用“先到先得”的原则; ④ 选课信息使用方便, 具备按照培养计划选课和按照课程库选课等功能.

表 2　选课管理的功能说明与角色说明

Tab. 2　Description of functions and roles for selecting courses

任务	内容	角色	角色所在部门
选课设置	设置开始时间、结束时间、选课对象、选课轮次	相关老师	研究生院
选课退课	根据培养计划选课或根据课程库选课	学生	/

基于以上需求, 新系统个性化地设计了两个入口供学生选课, 分别是根据培养计划选课和根据课程库选课. 根据培养计划选课的具体实现是, 学生进入“按培养计划选课”模块, 系统展示其培养计划, 已开课程在选课时段内呈现“选课”入口, 已选课程在退课时段内呈现“退课”入口, 其他时段及其他课程则提供查看课程详情的入口. 根据课程库选课, 基本与现系统无差, 这里由于篇幅所限, 不做过多说明.

选课时可供选择的课程较多, 但也有部分热门课程具有高并发、课程超选等问题. 因而, 新系统借鉴了淘宝的“秒杀”过程, 并基于实际业务进行了改进, 解决了同一时间段内大量用户同时选课, 系统的访问瞬时激增的难题, 并保障了选课信息安全可靠. 具体实现过程: 首先通过和研究生院交互, 将选课分流, 如系统中公共课和专业课错开开放选课的时间, 但在公共课选课时还会有某时段某些课程超选的问题; 然后, 针对上述问题, 将选课解耦为学生选课与课程匹配学生两个独立并行的部分, 通过使用消息队列的方式实现两部分之间的通信, 两个部分主要涉及学生选课记录表与开课班级表, 具体字段说明见表 3. 表 3 中, 学生选课部分不涉及任何复杂的选课逻辑, 仅仅在学生选课记录表中为该学生加入一条选课记录, 并将选课状态设置为“选课中”, 同时也向消息队列中发送一条学生选课事件. 课程匹配学生部分实现为一个单线程程序, 该部分按照一定的时间周期从消息队列中接收学生选课事件. 对于每个周期内观察到的选课班级, 它为每个开课班级执行如下流程.

(1) 通过 SQL 语句筛选出开课班级表中该班级的可选名额, 其中可选名额为最大人数减去已选人数.

(2) 使用 SQL 语句, 根据“先到先得”原则将学生选课记录表按照学生选课时间排列, 选出选课状态为“选课中”的前可选名额数记录.

(3) 通过 SQL 语句, 批量将返回记录的选课状态设置为“已选”.

(4) 更新开课班级表中该班级的已选人数.

(5) 在时间周期结束之后, 课程匹配学生部分将学生选课记录表中所有选课状态仍为“选课中”的记录设置为“未选中”.

表 3 开课班级表、学生选课记录表字段说明

Tab. 3 Description of the course schedule table and record table for selecting courses

表名	字段
开课班级表	id , 课程 id, 最大人数, 已选人数, 负责老师, 学年, 学期, 人数比例(本科生、留学生), 班级名称
学生选课记录表	id, 学生 id, 开课班级 id, 选课状态(已选、选课中、未选中), 选课时间, 退课时间, 选课操作人, 退课操作人

架构上, 通过业务拆分的方式, 系统将选课操作转变为异步的操作, 课程匹配学生部分可以根据业务需求自主控制接收消息的频率. 未进行拆分前, 针对同一课程班级的选课操作都涉及判断与更新开课班级表的已选人数, 数据库的事务要求可能导致并行发起的选课操作按照串行的方式进行调度执行. 业务拆分之后, 多个选课操作对开课班级表的多次操作被缩减为一次, 从而避免了大量事务导致系统性能降低的可能性.

5 实 验

测试环境为 Centos7 服务器, 配备 8 个主频为 2.40 GHz 的 Intel Xeon CPU E5-2630 芯片, 内存为 16 GB, 磁盘容量为 4 TB, 转速为 7 200 r/s. 主要测试不同并发情况下业务拆分与未拆分这两种方式的性能.

第一组实验是, 热门课程数 c 为 12(华东师大热门课程数在 10 –14 门), 线程数 t 分别为

100、200、500、800、1 000, 用于模拟真实选课下高并发的情况和测试选课接口访问请求的响应时间. 实验中, 在给定课程数 c 情况下, 每个线程执行 $c\times10$ 次选课请求, 用于模拟真实情况下学生不断刷新与选课操作. 实验采用所有访问请求的平均响应时间来衡量不同方式的性能. 实验结果如图 7 所示. 由图 7 可知, 无论线程数 t 设置为多大, 业务拆分方式的性能明显优于未拆分方式, 且随着线程数 t 增长, 两种方式性能之间的差异越来越大. 根据往年选课情况, 华东师大每年研究生新生人数在 6 000 人左右, 并发请求数量一般在 800 以内. 从图 7 看, 在 $t=800$ 时, 未拆分方式的响应时间大约是业务拆分方式响应时间的 4.5 倍.

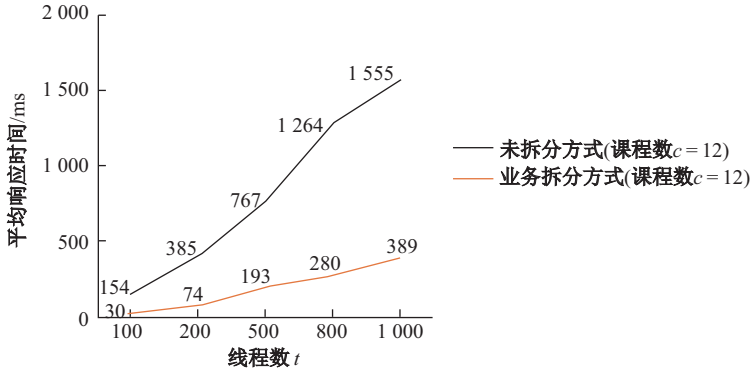

图 7　实验一实验结果

Fig. 7　The results of experiment No.1

第二组实验是, 在线程数 t 取 800 的情况下, 热门课程数 c 分别为 4、8、12、16、20、24, 用于模拟不同热门课程数, 测试这种情况下选课返回用户访问请求的响应时间. 实验中, 两种方式分别设置每种访问请求在课程数 c、线程数 t 的情况下, 每个线程执行 $c\times10$ 次. 实验结果如图 8 所示. 由图 8 可知, 无论课程数 c 设置为多大, 业务拆分方式的性能明显优于未拆分方式, 且未拆分方式的响应时间大约是业务拆分方式响应时间的 4.5 倍.

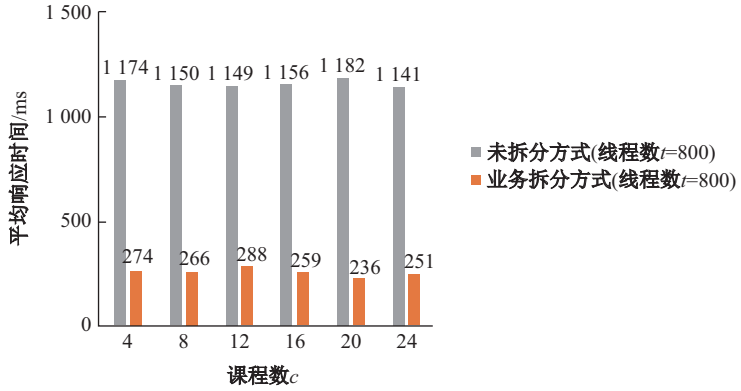

图 8　实验二实验结果

Fig. 8　The results of experiment No.2

综合以上分析, 业务拆分的选课操作无论在何种并发量下, 无论设置多少热门课程数, 都优于未拆分方式的串行操作. 在测试场景下, 未拆分方式的响应时间有 1 s 以上的情况, 用户体验较差; 业务拆分方式的响应时间都在 500 ms 以内, 有着很好的用户体验. 因而, 新一代培养系统的选课业务, 将选课拆分成两个独立并行的部分, 可以有效地降低访问响应时间, 提高系统的性能.

6 总 结

微服务架构将传统的信息系统拆分成多个分散的服务, 以实现这些服务独立进行开发、管理和迭代. 本文利用这种架构方式, 设计实现了新一代华东师大研究生培养系统. 从业务设计角度来看, 基于“以学生为中心”的设计理念, 围绕新业务模型, 从学生和教师的视角自主研发, 充分满足了新的培养业务需求, 方便了收集、组织、存储、利用研究生教育的教学管理数据以及教学行为数据, 打破了各个应用系统之间的数据隔离. 从系统架构和实现的角度来看, 新一代的研究生培养系统基于微服务架构, 支持自动化持续部署, 实现了一个给用户提供个性化服务、能快速迭代的研究生智慧服务平台. 从技术角度来看, 新系统解决了原系统响应缓慢、移动端访问不兼容等问题, 极大地提升了用户体验, 并提高了系统的使用效率.

[参 考 文 献]

[1] 杨彩霞, 邹晓东. 以学生为中心的高校教学质量保障: 理念建构与改进策略[J]. 教育发展研究, 2015(3): 30-36.
[2] HUANG H B, ZHOU B. Research on the construction of the micro service system of library in the era of big data[J]. Journal of Library & Information Science, 2016, 12: 40-43.
[3] 周英, 曾青青, 赵泽慧. 构建灵活的研究生教育一体化管理言息统——中山大学研究生教育管理信息系统的设计与实现[J]. 学位与研究生教育, 2011, 9: 55-60.
[4] 林嘉婷. 试谈前后端分离及基于前端 MVC 框架的开发[J]. 电脑编程技巧与维护, 2016, 23: 5-8.
[5] 丁振凡. Spring REST 风格 Web 服务的 JSON 消息封装及解析研究[J]. 智能计算机与应用, 2012(2): 9-10.
[6] 刘方军. 基于 MVC 三层架构模式的研究与应用[D]. 广州: 广东工业大学, 2011.
[7] 徐玲玲, 冯文超. 基于 MVC 和 EF 的学位与研究生教育信息管理系统设计与实现[J]. 计算机与现代化, 2012(4): 103-106.
[8] 姚思明. 消息中间件元数据管理模块及发布订阅接口的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2016.
[9] 佚名. Galera Cluster——新型的 MySQL 集群架构[J]. 电脑编程技巧与维护, 2017, 12: 4-5.
[10] 陈建娟, 刘行行. 基于 Kubernetes 的分布式ELK日志分析系统[J]. 电子技术与软件工程, 2016, 15: 211-212.
[11] 史兵, 夏帆, 宋树彬, 等. 研究生信息平台中运维系统的设计与实现[J]. 华东师范大学学报(自然科学版), 2017(5): 225-235.
[12] 林新党, 穆加艳. 基于 Jenkins 的持续集成系统研究[J]. 雷达与对抗, 2014(1): 58-61.
[13] 刘光明. Quartz 任务调度框架与 Web 整合的研究[J]. 电脑迷, 2017(1): 103, 105.

(责任编辑: 李 艺)