DOI：10. 16516/j. gedi. issn2095-8676. 2020. S2. 003

基于微服务的电力信息系统架构研究与设计

陈酌灼[1,✉]，张梦清[1]，茹春平[2]
（1. 广东电网有限公司广州供电局，广州 510620；2. 中山大学 数据科学与计算机学院，广州 510006）

摘要：［**目的**］为解决电网信息化建设过程重，单体应用架构系统性能低、开发效率低、难于扩展、容错性差等局限性，［**方法**］提出电网信息化系统建设采用微服务化改造方式。［**结果**］总结出电网信息化建设微服务架构方案、服务拆分方案、容器化部署、开发运维一体化方案，并以 Kubernetes+Docker+Jenkins 的微服务最佳实践。［**结论**］微服务化改造的电力信息化系统，为系统粒度、软件质量、系统运维上都带来了显著的提升，为实现电网信息化更丰富的业务场景、更短的系统交付周期、更高质量要求的系统交付要求提供微服务的技术路线。

关键词：电网信息系统微服务化；信息化；微服务；快速交付
中图分类号：TM7；R857. 3 **文献标志码：**A **文章编号：**2095-8676(2020)S2-0018-07
开放科学(资源服务)二维码：

Research and Design of Power Information System Architecture Based on Microservice

CHEN Zhuozhuo[1,✉], ZHANG Mengqing[1], RU Chunping[2]
（1. Guangzhou Power Supply Bureau of Guangdong Grid Co. , Ltd. , Guangzhou 510620, China;
2. School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China）

Abstract:［**Introduction**］In order to solve the limitations of single application architecture system of power grid information system, such as low performance, low development efficiency, difficult to expand, poor fault tolerance etc.［**Method**］This paper proposed that the construction of power grid information system adopts micro service architecture.［**Result**］We summarizes the microservice architecture scheme, service splitting scheme, containerization , development of operation and maintenance scheme of power grid informatization.［**Conclusion**］The electric power information system transformed by microservice has brought significant improvement to system granularity, software quality and system operation and maintenance. It provides microservice technical route for realizing richer business scenarios, shorter system delivery cycle and higher quality system delivery requirements of power grid informatization.

Key words: power grid informatization; mirco-service; method of mirco-service splic, solution of micro-service

随着供给侧结构性改革、电力体制改革、国资国企改革、“一带一路”、“互联网+”、国家网络安全法、新一代人工智能发展规划等新政策不断推进，电网信息化系统建设工作面临更多的系统需求、更丰富的业务场景、更短的系统交付周期、更高质量要求的系统交付等问题。

然而在电网信息化建设历程中，大多数系统技术架构采用的是传统单体应用程序框架，单体应用架构系统存在性能低、开发效率低、难于扩展、容错性差等局限性[1]。在追求客户极致体验的系统服务时代背景下，需要根据市场的变化需求对系统进行频繁升级完善，单体应用架构系统应对这些升级改造时，牵一发而动全身，添加新功能的同时，往往会带来新的问题，并且随着版本的迭代，系统代码会非常臃肿，维护升级越来越困难。

微服务架构（Micro-service Architecture）和容器（Container）技术的出现，可以通过对传统单体架构应用的改造，实现自动弹性资源扩容、流量负载均衡和热点微服务的实时水平拓展[2]，借助开发运维一体化（DevOps）持续集成方案，实时快速

收稿日期：2020-04-01 **修回日期：**2020-04-23

响应业务域运营问题及需求。

本文规整微服务在落地过程中的思路、问题、解决方案以及最终建设后的实施成果，全面阐述微服务架构系统。为后续微服务系统建设提供参考方，并提供微服务架构系统的落地案例，实现提升信息系统服务、提高交付软件质量，缩短系统交付周期的目标。

1　微服务系统架构

1.1　单体应用架构

要理解微服务，首先要理解单体应用。单体应用是传统的Web开发方式，和微服务架构相对应的，这种方式将所有的功能打包在一个软件包，并部署在一个Web容器（如Tomcat，JBoss，WebLogic）里，包含了数据层，业务层，界面层等所有逻辑。常见的单体应用架构图如图1所示：

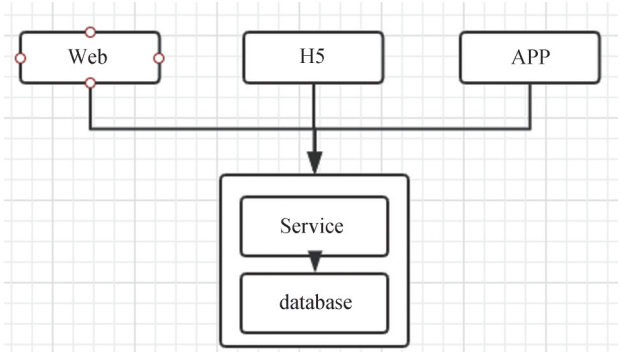

图1　单体应用架构

Fig. 1　Single application architecture

单体应用架构系统，有如下几个优点：

1）开发简单，代码集中式管理。

2）部署单一，没有分布式的管理和调用消耗。

3）日志单一，问题定位相对方便。

单体应用架构系统在历史IT信息系统开发和运营过程中，也暴露出了如下问题：

1）效率低：开发都在同一个项目改代码，模块开发需要相互等待，冲突不断[3]。

2）维护难：功能代码高度耦合[3]，维护代码修改难度大，团队新成员尤甚。

3）不灵活：构建时间长，任何小修改都要重构整个项目，过程耗时[3]。

4）稳定性差：一个微小的问题，都可能导致整个应用系统崩溃[3]。

5）扩展性差：高并发的业务，无法通过水平、垂直的方式快速扩容，导致用户一直在等待响应，体验极差。

1.2　微服务

所谓微服务指的是一些由API驱动的小型应用程序，它们在追求一个共同目标的前提下负责把一件事情做好[4]。微服务的概念是相对于单体架构应用而言，单个微服务其本身也是单体的应用，是从单体应用系统中独立出来的一个子系统，多个子系统联合组成完整的服务系统。微服务相对于单体架构应用有如下特点：

1）是从单体应用架构系统中剥离出来的一个相对功能独立的模块，独立部署形成服务。

2）服务间代码、部署、数据独立，隔离代码和运行环境，避免耦合带来的资源竞争问题，更专注做好自己的任务。

3）服务间代码、部署、数据独立，但业务不独立，通过服务间相互调用的方式来完成业务流程的处理。

4）更小粒度的功能迭代，更快速的交付周期，更可控的系统升级风险。

5）开发自由度更高，不限开发语言，开放的技术选型模式，不再受限于单体应用架构的既定技术框架。

6）更灵活的资源调度，可配合工具实现自动扩容和流量负载均衡，提应对更高的访问并发，提供更强的服务能力。

7）更高的容灾能力，更小容灭影响。

微服务为我们提供许多便利，但任何软件架构都是诸多trade-off的结果[5]，要获得微服务架构所带来的好处也就意味着要接受它所带来的挑战。

在微服务项目落地过程中，我们总结了以下几个重点问题和应对策略：

1.2.1　服务入口管理困难

由于微服务部署在不同地址和不同端口上，访问服务时涉及到多地址、多端口的管理，微服务系统规模较大的情况下，管理困难。解决的方式应根据系统规模进行选择：（1）系统较小，服务数量较少的情况下，可采用内置配置文件的方式，直接访问配置来定位微服务；（2）系统较大，服务划分较多的情况下，采用服务注册、服务发现、统一网关

的形式解决服务入口管理的问题。

1.2.2 问题跟踪定位困难

当出现系统BUG异常问题时，因为日志记录在不同的服务内，导致不好定位域跟踪解决。

如图2所示，一个用户的请求会通过5个服务，前后台的业务流会经过很多个微服务的处理和传递，调用过程复杂、数据流向节点多，导致了事务的跟踪相对困难。

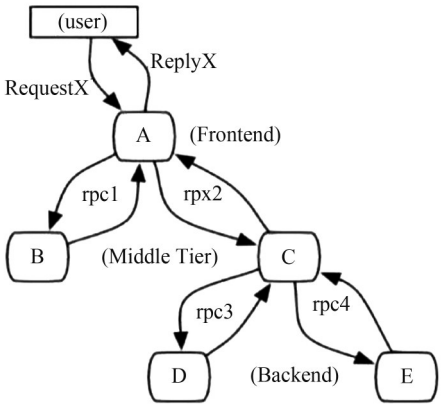

图 2 请求服务流转图

Fig. 2 Request data follow

针对这种问题在实施过程中，我们采用如下方案解决：（1）建立日志中心，统一日志存储及查询管理；（2）规定接口请求带上必须一个ReqID，处理请求的过程日志和ReqID建立对应关系，通过ReqId可全流程跟踪该请求的运行日志。

1.2.3 数据一致性问题

当一个业务流程下来，涉及到多个数据分布在不同微服务中的情况下，数据一致性比较难保证。在实施过程中，我们采用如下方案解决：（1）合理划分服务，规避这种情况的出现，把有状态的服务放在一个聚合服务中；（2）使用基于消息中间件的分布式事务处理方案，保证数据的最终一致性[6]。

1.2.4 运维管理问题

由于每个微服务都独立部署，因此运维的工作量从原来的1个单体服务变成了*n*个服务的部署与运维，倍数级的增加了运维的工作量。而微服务架构属于分布式架构，分布式架构下，我们面临如下运维管理问题：（1）容量管理问题，即如何在细粒度的状态下，更有效地管理数量庞大的微服务；（2）容器编排与配置管理，如何合理地实现容器编排和配置管理；（3）服务监控，如何有效地监控数量庞大的微服务；（4）故障恢复与业务调度，出现故障时如何进行业务调度；（5）快速扩展，由节日、活动等原因所导致的业务爆炸式增长或触发时，如何快速扩容；（6）资源的利用效率，运维人员需要思考如何在保障业务稳定发展的同时，控制好成本不会大幅增长。

按传统IT运维模式无法适应于微服务运维中，我们的解决方案是：

1）引入Devops，实现开发、测试、集成部署自动化等一体化管理流程[7]，通过自动化部署方式，减少部署的工作量。

2）使用Kubernetes容器编排组件功能，管理容器的编排与配置。通过Kubernetes可以及时部署应用程序，轻松扩展相同的应用程序并无缝地推出新功能，同时限制硬件资源的消耗。它还提供额外的功能，例如自我修复服务，包括自动放置（auto-placement），自动复制（auto-replication）和容器的自动重启（auto-restart）。

3）通过cAdvisor代理组件，监视和收集每个节点上每个容器的系统资源利用率和性能指标（例如CPU，内存，文件和网络）。

4）通过kubelet组件，管理每个节点的运行状态，并确保节点上的所有容器都是健康的。控制平面中管理器会指示kubelet处理容器内应用程序容器的启动和停止。

5）通过调度程序，调度逻辑基于资源可用性。调度程序会记录每个节点的资源利用率，确保分配的工作负载不会超过物理机或虚拟机上可用的资源。

1.3 微服务架构

微服务系统架构应主要由服务网关、注册中心、服务监控、配置中心等核心组件和微服务组成。各组件的作用及功能要求如下：

1）服务网关：为前端应用提供统一的访问入口。

2）注册中心：提供服务相关信息的存储，通过服务注册与服务发现实现微服务之间的调用通道。

3）服务监控：对服务的运行状态和调用链路进行监控，服务日志的统一管理，保障微服务以健

康状态运行。

4）配置中心：为微服务提供多个环境下的统一动态配置信息的存储，在各个运行环境下通过推拉的方式将更新的信息同步到微服务中。

微服务的整体架构如图3所示。

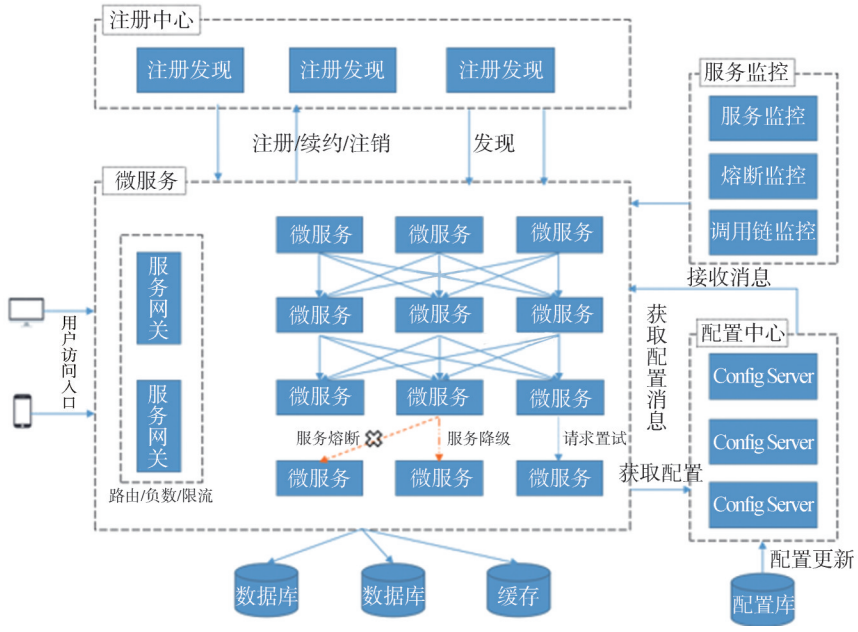

图3　微服务架构图

Fig. 3　Request data follow

2　微服务架构实践

2.1　业务系统选择

停电抢修应用属于营销域业务系统范畴，订单服务的对象是C端用电用户，有用户基数大、系统体验要求高、服务快速响应的要求特点，当区域性停电等情况下出现时，系统存在短时请求突然爆增的情况，这些特点符合微服务化改造的客观条件。

2.2　停电抢修业务简述

如图4业务流程图所示，低压停电抢修复电的业务始于营销系统产生的报障工单，经监控中心签收后派发到各区局，再按照报障位置所处网格派发到各驻点班组，由驻点进行抢单并生成抢修单，抢修单内的工作票、工作许可等并根据工作情况填写相关单据。抢修出车过程中也结合移动应用的定位功能实现路径导航、活动轨迹收集、到点签到等功能。

2.3　微服务的边界划分

微服务边界的划分，影响着整个系统微服务化的实施过程，合理划分服务是微服务落地中关键的一个步骤之一，合理的划分方案会减少后续开发、运维过程中的不少的问题，如：分布式事务处理困难、数据一致性无法强一致性保证、运维工作量快速膨胀等等。

停电抢修应用项目中的服务划分，我们遵循以下原则：

1）领域驱动原则。

2）职责单一原则。

3）数据共享原则。

经过系统的分析，根据领域驱动设计的思想，结合生产系统的实际情况，规整出如表1所示领域服务。

将各领域做进一步的逻辑解耦，我们将上个阶段分解的领域，以及当前系统在用的功能分解成四个层次：用户界面层、平台网关层、业务应用层、领域服务层、基础服务层（如图5所示）。

2.4　微服务最佳实践

停电抢修项目在微服务化改造过程中，我们采用Kubernetes + Docker + Jenkins的实践组合完成微服务的整体架构。

研发阶段：代码编写完成后，通过Jenkins组

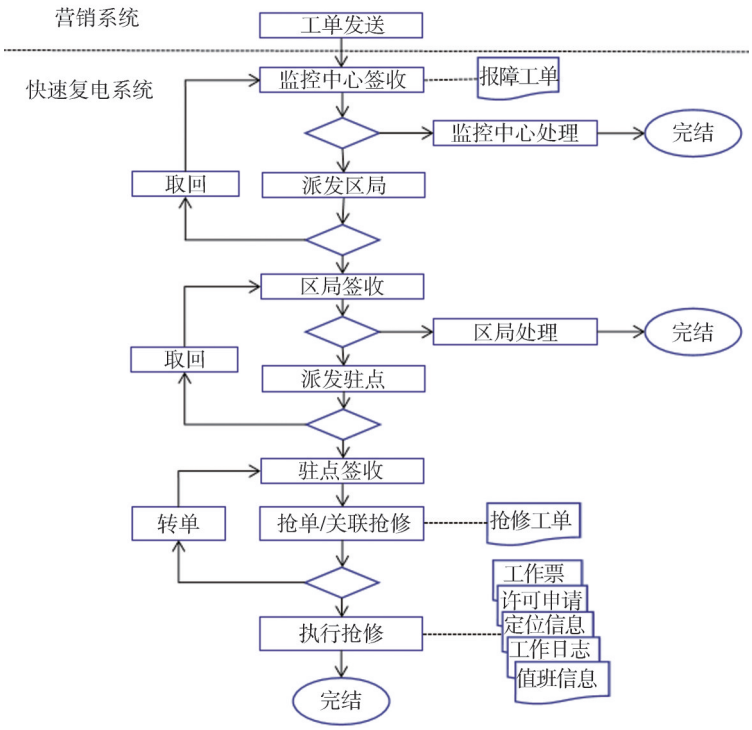

图 4　停电抢修业务流程图

Fig. 4　Flow chart of power outage and emergency repair

表 1　停电抢修系统服务划分

Tab. 1　Service split of power failure emergency repair system

领域	领域类型	作用
抢修服务	核心领域	查询处理停电抢修单
工单服务	关联领域	营销停电报障单同步
两票服务	关联领域	工作票、操作票同步
认证服务	支持领域	用户认证授权
台账服务	支持领域	设备台账查询
位置轨迹	支持领域	移动轨迹跟踪
文件服务	支持领域	过程文件存储
推送服务	支持领域	应用信息推送
短信服务	支持领域	应用短信发送
日志服务	支持领域	应用日志记录
通道服务	支持领域	内网系统数据通道

件来自动化处理代码编译、打包 Docker 镜像工作，并自动部署到 Kubernetes 平台。

运行阶段：利用 Kubernetes 的编排、配置、监控管理等工具，管理应用的整个生命周期，保障日常运营需要的同时，在流量爆发情况下实现基于访问流量的灵活扩容或缩容。

2.5　数据一致性问题处理

停电抢修应用主要是对工单数据的处理，工单

图 5　服务分层

Fig. 5　Service lamination

数据包含：营销系统报障工单、故障抢修工单 2 个部分的数据。在微服务化系统改造过程中，数据一致性问题并不突出，主要归功与项目前期对服务边界进行了清晰的划分，规避了大部分一致性的问题。但问题还是存在，主要体现在：报障工单状态与营销系统工单状态一致性。

针对此问题，因为数据率属于不同的系统，为减免双方系统改造成本，采用 OGG（Oracle GoldenGate，是基于日志的结构化数据复制工具，通过交易数据的实时捕捉、变换和投递，实现源数据库

与目标数据库的数据同步，保持亚秒级的数据延迟）的方式保证数据一致性同步。

2.6　流量爆发问题应对

针对区域性停电等突发情况产生的流量爆发问题，我们采用Kubernetes的HPA水平Pod自动伸缩方案进行应对。HPA（Horizontal Pod Autoscaler）可根据平台资源的利用率，自动扩展与收缩部署应用Pod数量，让应用的部署规模接近于实际访问负载。

在停电抢修应用中，我们利用K8s的cadvisor监控组件，采集Pod运行指标数据，提交Metrics Server。HPA控制器获取Metrics Server的指标数据，结合系统伸缩配置信息，向副本控制器CI发起Sacle操作指令，实现自动扩容伸缩部署。实现机制如图6示：

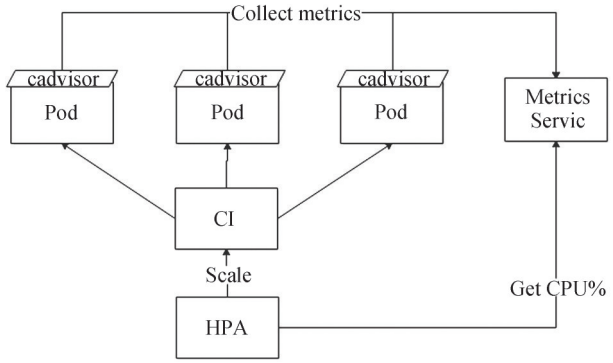

图6　HPA自动伸缩

Fig. 6　Horizontal pod auto scaler

2.7　项目管理的调整

服务划分并确定架构之后，即进入开发阶段。在开发阶段，不同服务的由不同的开发团队负责，实现了任务的“分工”，但分工之后也给“合作”的管理带来了新的问题。“分工”最大的一个问题之一就是信息同步成本的巨大，根据经验和统计发现，项目成本的80%体现在沟通上一点都不为过，其中信息不对称所产生的成本浪费约占沟通成本的50%以上。

因此，我们在微服务项目的管理过程中重点把控需求沟通与管理，加强服务团队间的信息共享，避免需求理解、沟通问题导致的资源浪费。如图7所示，采用敏捷迭代的需求管理方式，需求确认、开发计划、开发、测试、功能确认5个节点不断循环迭代。

为了防止各服务之间的实现偏离，前置集成、测试工作，集成问题早发现早解决。

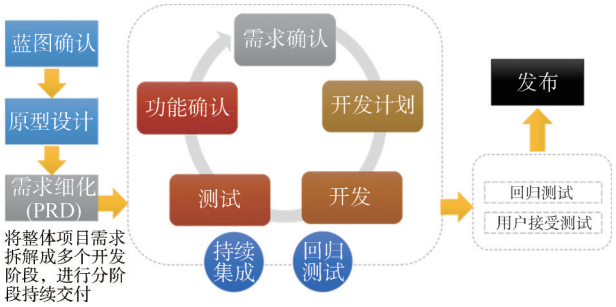

图7　项目管理调整

Fig. 7　Project management adjustment

2.8　实践成果

从停电抢修业务系统微服务化前后的数据对比（图8），我们可以看到：

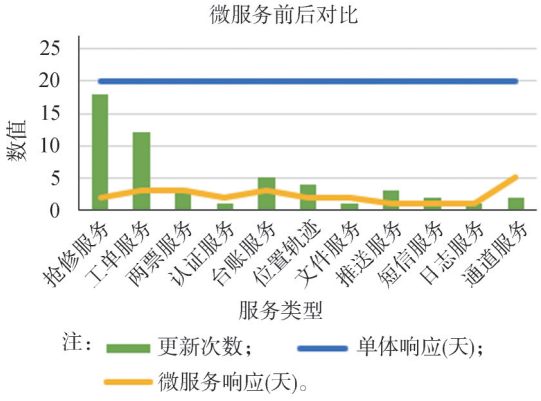

图8　微服务化前后数据对比

Fig. 8　Data comparison before and after

1）业务需求响应从单1的系统周期从原先的20 d，转移到了11个微服务的响应周期上，每个服务器的响应在1~5 d内，响应周期大幅下降，整体效率得到很大提升。

2）抢票服务、工单服务频繁更新迭代，使得研发更专注于核心业务模块，通过快速迭代更新不断提高软件质量。

3）系统运营监控，从原先的单一服务整体监控，下坠到各个子服务分别监控，管理粒度更精细、可靠。

3　结论

本文主要介绍微服务架构的设计理论，并总结落地过程中的思路、问题、解决方式和最终落地后的效果，综合阐述了微服务在电力信息化系统建设中的应用。介绍微服务系统搭建中的架构方案、服

务拆分方案、容器化部署以及开发运维一体化（DevOps）理念，以 Kubernetes + Docker + Jenkins 的微服务最佳实践落地停电抢修系统，支撑变化频繁、弹性伸缩、区域差异的复电业务场景，同时也为电网信息应用系统微服务化改造提供一个成功的案例和具体的解决方案。

参考文献：

[1] 承林，王海宁，高春成. 微服务在电力交易系统中的应用研究[J]. 电网技术，2018，42(2)：441-446.
CHENG L，WANG H N，GAO C C. Research on application of micro service in power transaction system [J]. Power System Technology，2018，42(2)：441-446.

[2] 王备，王刚军. 基于微服务架构实现售电平台的分析和研究[J]. 电力信息与通信技术，2019，17(4)：38-43.
WANG B，WANG G J. Analysis and research of electricity sale platform using micro-service architecture [J]. Electric Power Information and Communication Technology，2019，17(4)：38-43.

[3] LARRY. Microservice architecture [EB/OL]. (2017-06-17) [2019-10-11]. https://blog.csdn.net/u011537073/article/details/73359481.

[4] BASTANI K. Microservices are not the destination [EB/OL]. (2019-02-25)[2019-10-11]. https://tanzu.vmware.com/content/blog/microservices-are-not-your-destination.

[5] FOWLER M. Microservice prerequisites [EB/OL]. (2014-08-28)[2019-10-11]. https://martinfowler. com/bliki/MicroservicePrerequisites. html.

[6] 程伟华，周捷，赵亚. 基于微服务架构的电力系统拆分方法[J]. 信息技术，2018，42(10)：115-119.
CHENG W H，ZHOU J，ZHAO Y. The split method of power system based on microservices architecture [J]. Information Technology，2018，42(10)：115-119.

[7] 耿泉峰，李曦，葛维，等. 基于DevOps的软件开发管理模式[J]. 软件，2019，40(1)：93-96.
GENG Q F，LI X，GE W，et al. Software development and management mode based on DevOps [J]. Computer Engineering&Software，2019，40(1)：93-96.

作者简介：

陈酌灼

陈酌灼（通信作者）
1988-，男，广东广州人，高级工程师，软件工程硕士，主要研究方向电力信息化建设工作（e-mail）chenzhuozhuo@guangzhou. csg. cn。

张梦清
1990-，女，广东广州人，计算机科学与技术学士，工程师，主要从事电力信息化规划与信息技术研究（e-mail）zhangmengqing@guangzhou. csg. cn。

茹春平
1985-，女，广东广州人，中山大学数据科学与计算机学院，教师，主要从事教育工作（e-mail）403286390@qq. com。

（责任编辑　李辉）