微服务平台设计在轨道交通智能运维产品中的应用

贺嘉贝，黄　铖，王　朵
（株洲中车时代电气股份有限公司，湖南 株洲　412001）

摘　要：为解决传统运维模式下烟囱式开发重复建设、单节点服务可扩展性低等问题，文章通过引入互联网行业内微服务、容器化等先进技术，搭建了适用于轨道交通的智能运维领域的微服务平台；并以一套 FORESEE 轨道交通智能运维产品为例，详细介绍了微服务平台总体架构设计及微服务集群设计。该微服务平台采用高效的工程化开发模式，以平台复用、量产及分布式为目标。通过微服务平台实际应用情况可以看出，其应用可使平均人力投入削减 70%，并可降低部署运维的成本和门槛，总体实现了降本增效的目的。

关键词：智能运维；微服务；复用性；集群设计
中图分类号：TP311；U31^{+}.92　　**文献标识码：**A　　**文章编号：**2096–5427(2021)05–0052–08
doi:10.13889/j.issn.2096–5427.2021.05.009

Application of Microservice Platform Design in Rail Transit Intelligent Operation and Maintenance Products

HE Jiabei, HUANG Cheng, WANG Duo
(Zhuzhou CRRC Times Electric Co., Ltd., Zhuzhou, Hunan 412001, China)

Abstract: In order to solve the problems of repeated construction of chimney development and low scalability of single node service under the traditional mode, this paper introduces advanced technologies such as microservice and docker in the internet industry to build a microservice platform suitable for the field of rail transit intelligent operation and maintenance, and takes a set of FORESEE rail transit intelligent operation and maintenance products as an example, the overall architecture design of microservice platform and the design of microservice cluster are introduced in detail. The microservice platform adopts an efficient engineering development mode and aims at platform reuse, mass production and distribution. Through the actual application of the microservice platform, it can be seen that its application can reduce the average human investment by 70%, reduce the cost and threshold of deployment and operation and maintenance, and achieve the purpose of cost reduction and efficiency improvement.

Keywords: intelligent operation and maintenance; microservice; reusability; cluster design

0　引言

“十三五”规划实施以来，我国轨道交通行业发展迅猛，随着该领域数据量的不断增长，运维人员数量缺口越来越大，日常运维质量保持面临极大的挑战[1]。智能运维是近几年来在轨道交通行业内兴起的一种新的运维模式，其利用传感装置获取车辆运行时各设备的实时状态和故障数据，并借助大数据、云计算和人工智能等技术对设备系统进行故障诊断和状态管理[2]。

FORESEE 是中车株洲电力机车研究所有限公司研发的一套轨道交通智能运维产品，本文以此为例介绍微服务平台设计在轨道交通智能运维产品中的应用。FORESEE 以 WEB（网页）端软件为主，涉及城轨及干线铁路领域项目众多，涵盖业务场景包括远程监视、健康管理、专家知识库、列车履历、应急处置、车载软件管理、BI（商务智能）分析、能耗分析、移动终端及系统管理等各类型需求；其特点在于业务

收稿日期：2021–05–20
作者简介：贺嘉贝（1995—），男，硕士，工程师，主要研究信息化微服务、分布式技术在轨道交通中的应用。

场景多、技术跨度广、需求变化快，这就使得系统结构庞大且复杂，对可维护性、可扩展性要求日益增加。在以往传统的烟囱式开发模式下，存在大量重复建设和可扩展性问题，这无疑给运维系统研发效率和成本带来了持续性的挑战[3]。因此，系统需要优化技术架构，提高可扩展性，使得项目开发及时响应外部市场的快速变化；同时，需要保证在大数据场景下运维服务变得更加可靠，而微服务技术为此提供了帮助。

早在 2010 年前，国外互联网企业如 Amazon, Uber, Netflix 等就已经完成了由单体应用向微服务框架的全面转型，从而解决了日益庞大的系统中所存在的上述问题。约 2012 年之后，国内互联网企业（如百度，阿里，腾讯等）逐步开展微服务化。到 2018 年，微服务被写入了工业互联网平台标准化需求[4]，白皮书首次提出了应用系统应该建立“微服务框架”的要求。2020 年 8 月，国家标准计划 20203865-T-469《工业互联网平台 微服务参考框架》[5] 正式立项。可见用微服务框架思想进行大型软件工程化已经成为大势所趋。

微服务平台设计核心是分离解耦，在此基础上做到共性能力的抽象沉淀和共享复用[6]，从而解决传统架构下的交叉依赖、重复建设等问题，以开放服务的形式被不同业务调用。

本文阐述的微服务平台设计，一方面，通过运用目前主流的 SpringCloud（开源微服务编程框架）、Docker（应用容器引擎）及中间件技术等进行了基础架构设计；另一方面，通过结合实际轨道交通业务场景，设计了一套可复用的微服务集群，以此构建一个扩展性强、可靠性高的系统层来适配业务需求差异。

1　总体架构设计

微服务架构与面向服务的架构类似，但有别于传统的单体式方案，其将应用拆分成多个核心功能，每个功能为独立服务，可单独构建和部署[7]。微服务间可相互通信，且遵循无状态通信，其优势具体突显在以下几点：

（1）敏捷开发性。开发周期缩短，有助于实现敏捷部署和更新。

（2）高度可扩展。服务间耦合度低，可以任意组合，且支持多服务器分布式部署。

（3）出色弹性。微服务彼此互不影响，一个服务出现故障不会导致整个应用下线。

（4）易于运维。微服务模块化且小巧，易定位问题。

本文介绍的 FORESEE 微服务平台基于分布式架构构建了一系列微服务，除了共用的业务功能之外，还包括通用技术能力、服务运维能力和安全可靠能力等，且在众多业务场景中被集成。总体而言，微服务平台架构包括以下 4 个组成部分：微前端应用层、微服务平台层、技术中间件层和运维支撑层（图 1）。

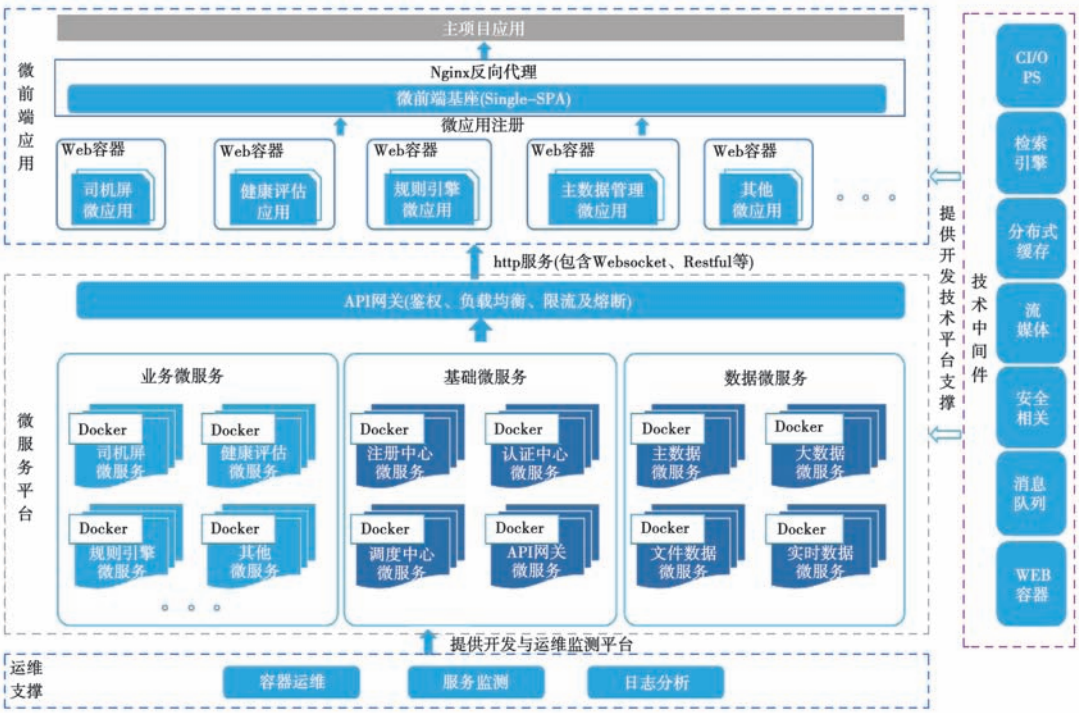

图 1　FORESEE 微服务平台整体架构设计

Fig. 1　Architectural design of FORESEE microservice platform

1.1 微前端应用层

如图 1 所示，微服务平台架构设计是前后端分离的，最上层基于 Angular（前端编程框架）并遵循 Single-SPA（微前端技术规范）开发设计了一系列具有独立业务的 UI（用户界面）子应用，其目的是便于和后台微服务一起进行灵活组合部署，形成可扩展的前端服务框架和 UI 组件库。

微前端应用层用来支撑各类轨交业务展示需求，如大屏展示、司机屏显示、地图实时监控车辆状态、各类轨交 BI 分析图表生成、数据回放等。设计采用了主从模式，将上述定制化展示需求封装为子应用，由主工程基座应用，动态加载和管理若干子应用，使其具备高度灵活性。

1.2 微服务平台层

微服务平台层指的是后端数据服务，概念与前端对应，基于 Spring Cloud 微服务技术架构和 Docker 容器化部署，形成一系列通用业务微服务集群和一系列具有领域特征的业务微服务集群，以此作为整个系统的数据后台。前后端分离设计的意义不仅在于工程化上，同时，本系统的后台可以单独提供数据服务，借助 API（application programming interface）网关供第三方系统调用。其数据范围除了系统和用户基本信息外，还覆盖城市轨道交通和干线铁路轨道交通领域的各类数据，如线路、站点、轨旁、车辆、故障、预警、应急事件、故障及信号量等，宏观上实现了各类历史数据的存储和统计以及实时数据的推送，具体设计详见第 2 节。

1.3 技术中间件层

为进一步加强基础技术的支撑，面向数据后台设计了一套通用技术中间件库，以集成业内多类主流中间件，如 Nginx（反向代理软件）、RabbitMQ（消息代理软件）、Redis（高性能数据库软件）和 ElasticSearch（全文检索引擎）等，从内部提供底层技术组件支撑，涵盖数据查询、数据推送、性能负载、任务管理及应用安全等功能。

1.4 运维支撑层

如图 2 所示，在 FORESEE 产品架构中，本文面向编码开发到部署的全生命周期，设计了一套工程化运维管理模式，以达到流程化管控和少人化目的。基于“Gerrit（代码仓库）+Jerkins CI（持续集成平台）+Kubernetes（容器管理平台）”的整体架构，运维支撑层具体设计包含以下 4 个方面：

（1）源码管理。为保证源码的质量和安全，由内网 Gerrit 仓库进行统一管理，面向编码开发人员，仓库记录每一次代码提交历史，可回溯和回滚，具备审核流程控制，通过审阅后才能合入项目。

（2）编译打包管理。为实现自动编译和打包，代码合入后，将触发 Jerkins CI 拉取代码，并执行编译及预检查，内容包括语法错误、单元测试等。测试通过后，将编译后的程序打包成容器化 Docker 镜

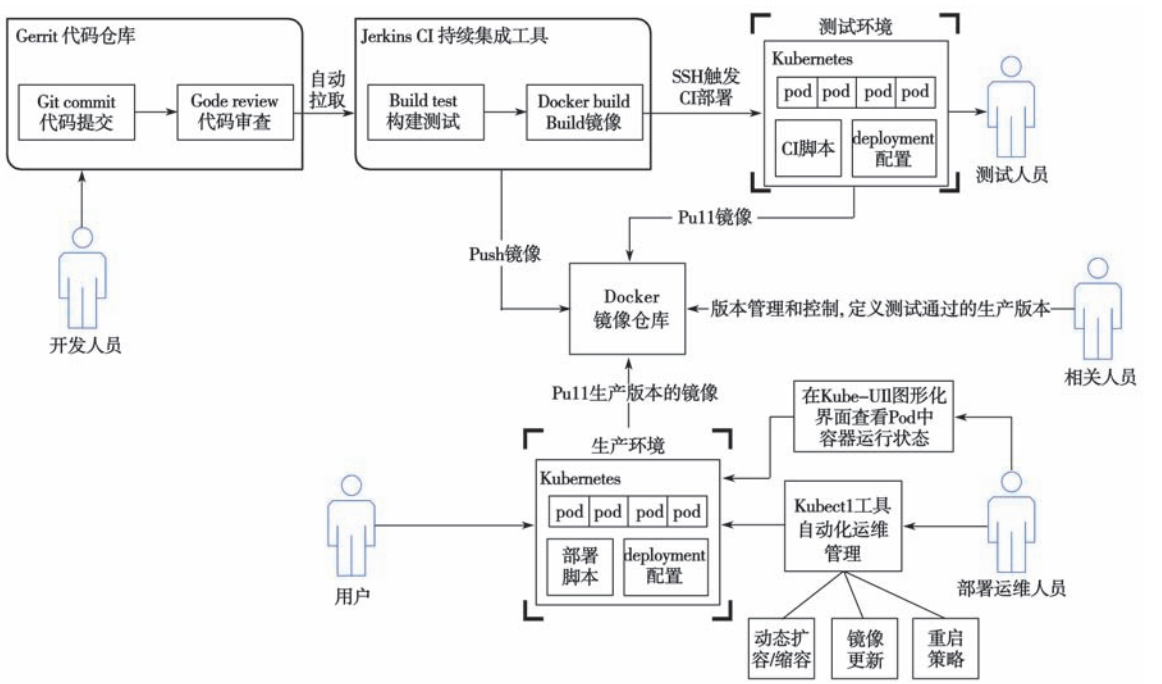

图 2 工程化运维支撑

Fig. 2 Operation support in engineering development

像，推送（push）到 Docker 仓库存储。在此过程中，将完成一次测试环境的系统集成部署，以代替人工操作，减少人力。

（3）软件包管理。为了集中管理软件包并实现版本控制，编译后的每个微服务，通过 Docker 容器化技术进行镜像封装，并通过内网的中央镜像仓库进行集中式存储。所有软件包都是高度封装的 Docker 镜像，以确保运行隔离性和源码安全[8]。软件版本管理人员通过标识标签（tag）描述其版本信息。

（4）部署运维管理。为达到部署后系统的高效持续运维，通过 Kubernetes 技术，运维人员可基于图形化页面或命令行实现远程运维监控，提供系统日志收集、存储、分析以及系统实时状态监测、升级部署、更新 / 回滚、自动重启、自动伸缩 / 扩展等。这样的技术设计，替代了传统运维中面向底层的 Linux 服务器和文件系统，取代大量使用命令行进行人工操作的场景，降低了时间成本和技术门槛。

2　微服务集群设计

在实现平台复用性上，本节介绍了通过微服务划分建立的一系列通用型微服务及框架中的 SDK（软件开发工具包）。

2.1　微服务划分

业内主流的后端微服务框架有 SpringCloud 及 Dubbon（阿里巴巴的国产微服务编程框架）两种。本方案选用了 SpringCloud，因其具有更加开放、良好的发展生态。其设计思路是对后台数据服务进行拆分，抽离其通用能力。根据职责的不同，微服务被划分为基础微服务、数据微服务和业务微服务 3 类，后两类被统称为应用微服务（图 3）。由于业务微服务是面向特定开发项目的特定业务表现形式，而本文聚焦于平台设计本身，故不会对此展开介绍。

基础微服务为应用微服务提供了通用性技术功能。作为基础建设部分，其包括注册中心、网关及认证中心等。

应用微服务区分了业务微服务和数据微服务两类，并存在前者依赖于后者的关系。例如，司机屏和机车健康评估这些项目中的具体业务板块都依赖于底层数据接口，这些接口以 Http（超文本传输）或 WebSocket(全双工通信) 协议形式被封装在数据微服务集群中，集群包括

（1）用于提供各类静态配置数据的主数据微服务，例如线网级基础码表。

（2）用于统计信息和实时信息的大数据服务，例如车辆实时信号量。

（3）用于存储文件的文件数据微服务，例如车辆日检的离线文件。

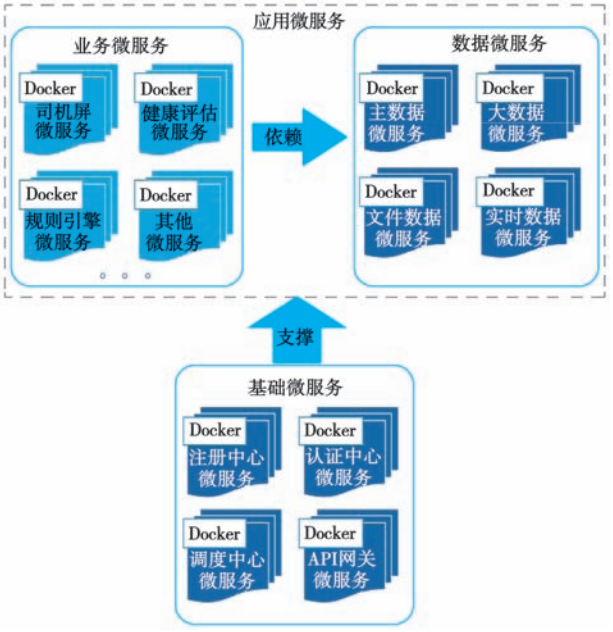

图 3　微服务划分
Fig. 3　Microservice division

微服务架构下平台具备很强的扩展性，可根据不同项目的特殊需求来定制增加新的内容，可能是一个，也可能是多个。新的部分仍然以微服务形式加入，保证了服务之间的隔离和解耦[9]。因此，在此分布式架构下，只要地铁公司或机务段内的硬件条件允许，便可提供高效、高性能的微服务扩展以及负载均衡，在多节点 / 单节点多种方式下灵活运行。下面对基础微服务和数据微服务中的关键设计进行详细介绍。

2.1.1　基础微服务的 API 网关

各机务段和地铁公司对 WEB 应用安全存在基本要求，即 GB/T 22239-2019《信息安全技术 网络安全等级保护基本要求》[10] 规范，以达到防范各类外部攻击的能力。本架构下主要以网关微服务配合 Nginx 反向代理进行防治。

网关是微服务集群的入口，本身也是一种特殊的微服务；但作为系统流量的瓶颈，网关的底层设计是实现异步非阻塞的，用于实现后台程序对外入口的路由转发以及安全防线。图 4 展示了认证过程中网关微服务、认证中心微服务以及其他微服务之间的协作关系。

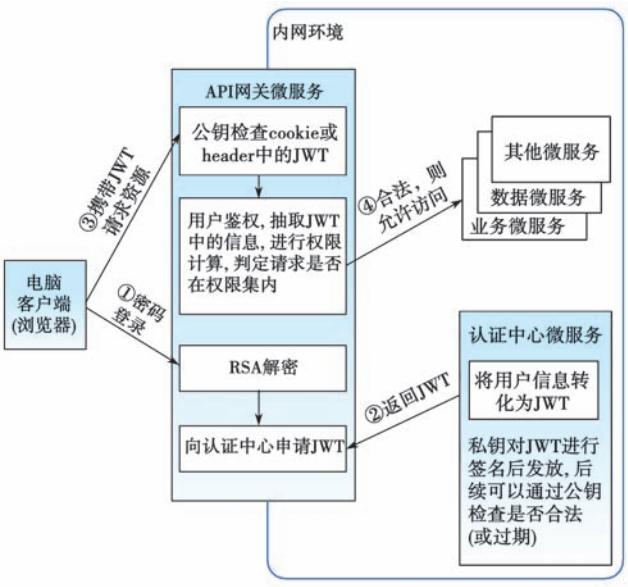

图 4　网关中的认证流程

Fig. 4　Authentication process in gateway

网关跨一个或多个内部 API 提供统一、规范的多种 API 入口点，并抽离大多数通信层逻辑，其目的是降低内部微服务自身的复杂性。以下功能被设计用以支撑内部集群的使用：

（1）路由功能。其将外部公共接口与内部微服务接口分开，支持 Http 和 Websocket 多协议代理转发规则，允许添加微服务和更改边界；为所有微服务提供单一入口点，对客户端隐藏服务发现和版本控制详细信息；实现前后端分离架构下的前端获取数据标准入口，同时亦支持对外单独提供第三方 API 的服务。

（2）额外的安全层。其提供统一的 JWT（JSON web token）鉴权认证拦截。此外，API 网关通过提供一个额外的保护层来防止恶意攻击，包括 SQL（结构化查询语言）注入、DoS（拒绝服务）攻击、CSRF（跨站请求伪造）攻击及 XSS（跨站脚本）攻击。

（3）负载均衡功能。其对接口请求进行负载均衡，包括 7 种类型策略，即随机、轮询、重试、最低并发、可用过滤、响应时间加权和区域权衡。

2.1.2　基础微服务的认证中心

根据各机务段和地铁公司对权限管控的需求，针对不同种类用户（如指挥调度，检修作业人员等）设置不同的操作权限，根据角色决定具体资源是否可见。

因此，在鉴权方面，通过认证中心服务管理进行集群内所有服务的各类鉴权操作，采用用户 - 角色 - 权限 3 层设计逻辑，即 RBAC(role-based access control)。底层权限又设计了两类，包括前端资源（模块、路由、页面、按钮）和后端资源（API 及其参数约束）。这使得无论是页面访问还是接口数据访问，都是根据不同角色、不同用户进行制约的。

在底层认证实现上，考虑到分布式架构特点，选用了 Token（令牌）模式认证，而非传统 Session（会话）模式；基于“Oauth2（授权开放标准）+ JWT + SpringSecurity（安全管理编程框架）”进行底层认证管理，具备 4 类认证模式，即授权码模式、简化模式、密码模式和客户端模式。此外，设计时对认证相关的元数据进行了 Bcrypt（不可逆的加密算法）加密，以保证持久层的安全。

如图 4 所示，认证中心具备对操作用户的全局认证和令牌发放功能，并支持供第三方的认证登录 API 以接入系统。通过这种认证中心的抽离解耦，可以灵活接入业主的人力资源（human resource, HR）原始数据，以面向不同组织架构的机务段和地铁公司需求。

除了前文提到的鉴权和认证基本功能之外，认证中心还设计了用户行为日志的功能。对于用户的页面操作，通过前端埋点和后端微服务埋点两种方式向认证中心服务进行数据传递，并记录到日志表中。其信息包含源 IP、用户名、功能板块、事件类型、操作内容、操作事件及操作结果，为后期的分析统计及记录追溯提供基础。

2.1.3　基础微服务的注册中心

注册中心主要用于微服务集群管理，管理平台内各微服务的注册上线和离线。目前业内主流技术是 ZooKeeper（Apache 分布式协调工具）或 Eureka（Netflix 服务发现框架）。在分布式架构的 CAP（consistency，availability，partition tolerance）理论下，ZooKeeper 偏向于 CP（consistency，availability），Eureka 定位则是为了保证服务高可用，用来保证 AP（availability，partition tolerance）。本文选用了 Eureka，因为作为单纯的服务注册中心，Eureka 更加专一，注册服务更重要的是可用性，可以接受短期内达不到一致性的状况。

本设计基于 Eureka 的服务发现、服务注册和服务监视能力，通过 Spring Actuator（服务健康审计工具）接口，对集群中所有微服务进行数据交互和通信；并将数据发送至注册中心，实现实时监控及辅助运维，使得注册中心服务具备面向微服务的观察工具平台以及可视化 UI 界面。注册中心功能包括微服务列表、微服务启停日志、环境参数查看、性能监控、微服务运行日志、Http 请求跟踪及 JVM（Java 虚拟机）监视等。

2.1.4　数据微服务的实时数据

按照目前城市轨道交通和机车的智能运维系统具体业务功能划分，其主要实现与故障、应急事件及状态预警等相关的常用业务的实时数据服务。故设计了一种统一提供实时数据的微服务，其底层数据均由此服务提供，而依赖的具体微服务只需实现具体业务功能即可，如故障业务中故障新增、故障消除、故障过滤、原生及次生故障分类等相关业务。

根据业务实现模块化开发，基于配置实现业务模块弹性部署。采用业务实现接口模式开发、定义标准接口参数，保证前端业务方法调用统一。实时数据微服务设计如图 5 所示。

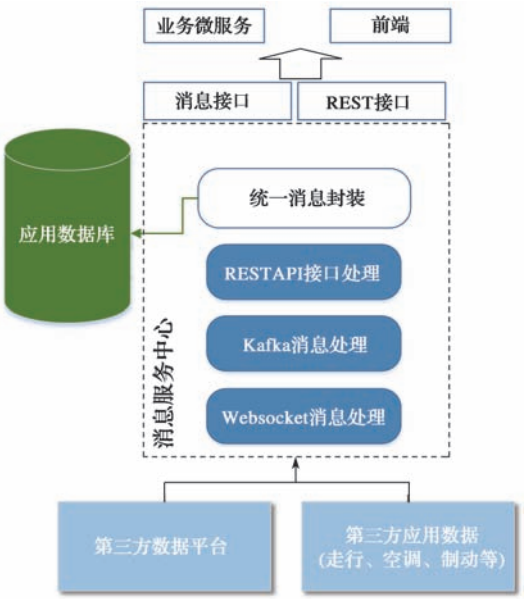

图 5　实时数据微服务

Fig. 5　Microservice of real-time data

通过建立实时消息服务中心，对接外部第三方平台的数据，支持消息队列，实时数据及 REST（表述性状态转移原则）数据，提供统一标准的实时接口和 REST 接口，保障提供到后端和前端应用数据结构一致。这样设计的好处是，如有定制内容，只需变更消息封装模块即可实现中台处理和外部数据的解耦。其具体实现包括

（1）提供第三方应用或第三方消息队列数据接入和缓存能力，消息数据可作为计算引擎或微服务的数据源，用来面向异步的数据（如下发的轨道交通检修计划）。

（2）提供平台或第三方实时数据 Websocket 接入和转发能力，面向轨道交通实时故障信号、信号量等数据。

（3）提供第三方 RESTAPI 数据接入和缓存能力，主要面向历史数据查询。

2.1.5　数据微服务的大数据

大数据微服务面向的数据来源是大数据平台的场景，其可实现数据转化、数据过滤及应用封装，并提供统一的大数据接口供其他业务微服务使用，包括大数据相关的数据查询、数据转化、数据过滤及数据封装，如图 6 所示。

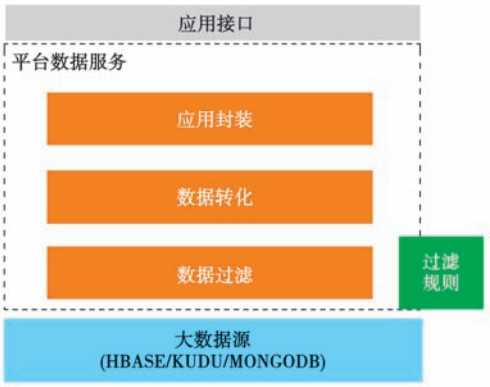

图 6　大数据微服务

Fig. 6　Microservice of big data

设计实现包括

（1）支持 HBASE(分布式数据库)、KUDU（新型列式存储系统）等大数据存储系统；

（2）基于时间、信号量、车辆号及故障码等关键信息查询；

（3）关键信号数据转化，包括数据格式转换及数据单位转换等；

（4）实现数据过滤，包括时间数据、信号量大小、故障码及自定义规则等；

（5）实现数据按具体格式封装，包括 JSON, XML 等数据格式。

2.1.6　数据微服务的主数据

面向静态的基础数据，形成通用化主数据管理服务，提供统一对外的服务接口；主数据服务设计范围包括车辆及相关设备的基础数据。提供平台级的基础服务集成服务，对线网级、单车级的基础数据进行数据集成，提供统一的对外主数据服务接口。内容如下：

（1）线网级主数据，包含线路、站点、轨旁、车辆等的基础数据。

（2）单车级主数据，包含基础故障码表数据、预警码表数据、应急事件码表、自定义故障规则数据及版本数据等。

（3）客户主数据，包含用户数据、权限数据及组织部门数据等。

2.2 面向微服务的 SDK 类库

虽然利用通用型微服务的可复用特征，新项目设计时可以直接复用上述基础服务和部分数据服务，减少了大量的重复开发工作，提高了开发效率，但目前仍面临着其他方面挑战。例如，在协作开发过程中，易出现组件兼容性问题以及因开发人员技术水平的差异性而导致的难以统一把控问题。为进一步提高开发效率，使开发趋于规范化，从编码开发角度入手，采用模板工程和依赖类库软件开发工具包（software development kit，SDK）的模式，面向开发人员提供充分的二次开发支持。如图 7 所示，开发人员将按照模板工程，引入 SDK 中大量 JAVA 类，将更多精力聚焦于纯粹的业务开发内容。

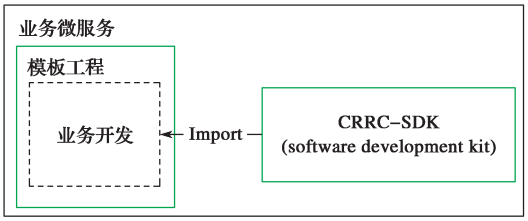

图 7 模板工程和 SDK 类库
Fig. 7 Project template & SDK class library

面向 JAVA 语言，其代码模板工程和依赖类库（CRRC-SDK）以源码形式供二次开发使用，存放于 Girret 代码仓库，具备版本管理。根据实际业务场景和量产需求，以支撑具体应用场景的规范化和快速复用，代码模板包含项目通用目录结构以及各类配套脚本及配置文件支持，作为项目开发的标准参照范例。

依赖类库主要包含 2 个方面，一是统一化的依赖管理、编译及打包脚本；二是公用类库。

前者基于 Gradle（项目自动化构建工具）包管理技术，利用 Groovy（敏捷开发语言）进行定制化脚本开发，形成一套兼容各类型项目的统一依赖管理、统一自动化编译流程、统一 Docker 打包的自动化编译打包程序，将底层依赖管理和编译过程封装在底层，开发人员是无感知的。

后者对当前业务范围内项目中的逻辑进行抽离，供各微服务依赖使用。基于这些 JAVA 类，根据面向对象语言的特性，可进行继承、方法重载等灵活使用。为了覆盖大量的常见编码开发场景，进行如下设计：

（1）统一异常处理类。封装 JAVA 异常对象，面向接口层的监听捕获器，统一常见的业务异常以及前后端通信异常、数据异常和认证异常等。此类会捕获所有常见的程序异常，开发人员无须编写异常捕获的逻辑。

（2）国际化支持类。对于后端数据返回的文本，进行国际化翻译。此类生效范围是全局的，面向所有 JSON 返回值。开发人员面向一个 JSON 配置文件，无须关注其中逻辑。

（3）Websocket 数据通信类。其为基于 Netty（异步非阻塞网络编程框架）的全双工长连接 Websocket 协议通信工具类。此类通过 JAVA 注解实现功能。开发人员在编写时加上此注解，重载相关函数，即可实现 Websocket 推送服务。

（4）数据库类。其为基于 JPA（JAVA 持久层 API）的再封装。对于常见的查询排序逻辑，其进行接口和 Java 泛型的封装；对于常见的增删改查逻辑，开发人员可以全部无须编写，这样减少了低端重复劳动。

（5）工具类。其包含时间操作、进制计算、字符串操作及 JSON 解析等常见业务常见的工具类，便于开发人员在日常开发中直接调用。

（6）安全类。其包含 SQL 注入拦截器、XSS 拦截器、CSRF 拦截器、鉴权拦截器、认证拦截器、Spirng Security 及 Oauth2 安全配置类、JWT 解析器、AES（advanced encryption standard）加密器、RSA(非对称加密) 加密器以及 Jasypt(Java simplified encryption) 加密器。以上功能在使用 SDK 包时将自动注入程序中，为安全性提供支持。

（7）任务调度类。其包含 Quartz(定时任务编程框架) 定时任务调度服务类和 XXL-JOB（轻量级分布式任务调度平台）。

（8）FTP 类（file transfer protocol）。其包含配置类以及服务类（上传、下载等）。

（9）REST 接口类。其对 JSON 请求体以及返回体进行了封装，包含 REST 接口收发器以及字段加密注解。此部分设计的主要目的是在前后端分离场景下对数据结构进行规范化。

（10）Excel（电子报表）解析类。为基于 POI（微软文档解析技术）封装的一系列 Excel 解析工具类。对于文档解析逻辑，其可直接调用相关函数，开发人员无须学习 POI 技术，降低了开发门槛。

3 微服务平台实际应用情况

中车 FORESEE 智能运维微服务系列产品自 2018 年开始推广到实际市场项目开发中，到目前为止，已经被广泛应用在城市轨道交通和干线铁路交通领域，囊括老项目改造和新开项目，涉及广州地铁 2

号线、宁波地铁 3 号线、上海地铁 13 号线及深圳地铁 11 号线等 23 个城市轨道交通领域智能运维项目，以及南宁机务段大数据系统、朔黄机务大数据智能运维系统以及神朔大数据专家诊断系统等 8 个干线铁路交通领域智能运维项目，实现了基础技术架构 100% 复用，功能组件 70% 以上复用。通过微服务的高可用、高可靠、可扩展及可持续演进的特点，能快速响应业主进行持续业务扩展的需求，并在实际应用中集成了第三方的服务。

相较于以前的开发模式，得益于模块复用，FORESEE 产品的数据后台设计周期和前端设计周期大大缩短。以城市轨道交通系列产品编号开发为例，较 2018 年早期项目开发，2020 年度单项目的平均开发人员数量投入削减了 70%。在人员总数没有增加的情况下，由最初的每年二十余人承接 3~4 个城轨项目并行开发，到如今每年十几人支撑 8~9 个城轨项目并行开发。除此外，基础技术的复用和封装设计使得产品的质量大大提升；在安全方面，使用微服务平台构建的所有项目已达到 GB/T 22239-2019《信息安全技术 网络安全等级保护基本要求》标准要求。

另一方面，从产品的部署调试运维角度而言，使用“Docker+Kubernetes”容器化技术之后，现场部署和数据调试的时间大大缩短，以 FORESEE 上海地铁 13 号线智能运维项目为例，相较于 2018 年的老版本，如今进行现场部署和数据调试的时间由 1 个多月缩短至 1 ~ 2 周左右；针对允许互联网接入的业主，通过纯粹的远程操作，现场部署和数据调试时间往往能控制在 1 周以内，并可省去所有的差旅费用。采用微服务平台前后智能运维产品的成本与效率对比如表 1 所示。

表 1　FORESEE 产品中烟囱式工程与微服务工程的成本与效率对比

Tab. 1　Cost and efficiency comparison between stove-piped engineering and microservice engineering in FORESEE products

对比项	烟囱式工程	微服务工程
年度开发人力 /（人数 / 项目）	5	1.5
部署调试时间 /（天 / 项目）	30	10
等保覆盖率 /%	0	100
开发复用率 /%	0	70

4　结语

基于轨道交通智能运维领域开发实际情况，针对目前所面临的项目数量激增、开发周期短、定制化多及人手短缺等诸多挑战，本文设计了一种微服务平台，解决了传统模式下烟囱式开发可扩展性差、重复工作多以及高昂的运维成本等问题。在该微服务平台架构设计下，平台本身即具备可靠运维、符合等保规范、组装式开发、集群可扩展复制及快速发布等特点。这使得实际二次开发的时候，开发人员仅需聚焦于那些给不同种类轨道交通用户定制的业务逻辑，而无须在基础建设上花费大量时间，从而大幅降低开发门槛和成本。

本设计基于分布式场景，故在有限的硬件资源下，存在一些性能上的限制，如何通过信息化技术提升后台数据处理性能，是今后的研究方向。

参考文献：

[1] 交通运输部 . 2018 年交通运输行业发展统计公报 [J]. 驾驶园 , 2019(8): 49-55.

[2] 胡佳琦 . 上海市轨道交通车辆智能运维系统研究与应用 [J]. 现代城市轨道交通 , 2019(7): 5-9.
HU J Q. Research and application of intelligent operation and maintenance system for Shanghai rail transit vehicles [J]. Modern Urban Transit, 2019(7): 5-9.

[3] 周凯歌 , 卢彦 . 工业 4.0: 转型升级路线图 [M]. 北京 : 人民邮电出版社 , 2016.

[4] 佚名 .《工业互联网平台标准化白皮书 (2018)》发布 [J]. 自动化博览 , 2018, 35(2):3.

[5] 中国电子技术标准化研究院 . 工业互联网平台 微服务参考框架 :20203865-T-469 [S/OL]. [2020-11-23]. http://std.samr.gov.cn/gb/search/gbDetailed?id=B4C317FC4F6E7A26E05397BE0A0AB954.

[6] 倪晓熔 , 顾欣 , 刘昭 , 等 . 电信运营商智慧中台架构及建设思路 [J]. 电信工程技术与标准化 , 2020, 33(11): 1-7, 74.
NI X R, GU X, LIU Z, et al. Intelligent middle platform and its construction thoughts for telecom operator[J] Telecom Engineering Technics and Standardization, 2020, 33(11): 1-7, 74.

[7] 周立 . Spring Cloud 与 Docker 微服务架构实战 [M]. 北京 : 电子工业出版社 , 2017:1-8, 199-206.

[8] 孙海洪 . 微服务架构和容器技术应用 [J]. 金融电子化 , 2016(5): 63-64.

[9] 张峰 . 微服务技术构建大规模 web 系统的研究 [J]. 科技创新与应用 , 2017(22): 48-49.

[10] 公安部第三研究所 . 信息安全技术 网络安全等级保护基本要求 :GB/T 22239-2019[S/OL]. [2019-05-10].http://std.samr.gov.cn/gb/search/gbDetailed?id=88F4E6DA63434198E05397BE0A0ADE2D.