文章编号:1671-4598(2026)07-0284-10 DOI:10.16526/j.cnki.11-4762/tp.2026.07.035 中图分类号:TP31 文献标识码:A

基于微服务的可重复使用运载火箭试验数据交互平台

彭炳锋[1], 王诗雯[1], 邓 闯[1], 欧阳李青[1], 兰旭东[1], 徐 昕[2]
(1. 上海航天电子技术研究所, 上海 201109;
2. 中国航天科技集团 商业火箭有限公司, 上海 201109)

摘要: 由于传统运载火箭试验数据交互平台在通用化、可靠性、扩展性等方面存在不足, 开展基于微服务架构的试验数据交互平台设计研究; 采用容器技术 (Docker 引擎) 与容器编排技术 (K8s, Kubernetes), 设计五层分布式平台架构, 实现服务模块解耦和冗余部署; 设计无代码数据判读语句体系, 兼容 MySQL 数据库、达梦数据库、时序数据库 (InfluxDB) 多种数据库, 并完成运载火箭测试数据接入、解析、显示、判读全流程管理; 经测试, 平台单帧数据处理时延、故障自愈速度、容灾能力等指标均达到设计目标, 数据判读准确, 平台实现高可靠、易扩展、自主可控的工程目标, 满足新一代火箭地面测发控系统的发展需要。

关键词: 可重复运载火箭; 通用化; 测发控系统; 监测评估; 数据判读

A Microservice-Based Test Data Interaction Platform on Reusable Launch Vehicles

PENG Bingfeng[1], WANG Shiwen[1], DENG Chuang[1], OUYANG Liqing[1], LAN Xudong[1], XU Xin[2]
(1. Shanghai Aerospace Electronic Technology Research Institute, Shanghai 201109, China;
2. China Aerospace Science and Technology Corporation Commercial Rocket Co., Ltd., Shanghai 201109, China)

Abstract: Due to the deficiencies in generalization, reliability, and scalability of traditional launch vehicle test data exchange platforms, research on the design of microservice-based architecture test data exchange platforms is conducted. Adopt container technology (Docker engine) and container orchestration technology (K8s, Kubernetes), design a five-layer distributed platform architecture, and achieve service module decoupling and redundant deployment. Design a no-code data interpretation statement system compatible with multiple databases including MySQL database, Dameng database, and time-series database (InfluxDB), and complete the full-process management of launch vehicle test data access, parsing, display, and interpretation. After testing, the platform's single-frame data processing latency, fault self-healing speed, and disaster tolerance capabilities have all met the design goals, with accurate data interpretation. The platform exhibits the engineering objectives of high reliability, easy scalability, autonomous controllability, meeting the development needs of the new generation rocket ground test, launch, and control system.

Keywords: reusable launch vehicle; generalization; test and launch control system; monitoring and evaluation; data interpretation

0 引言

航天技术的发展是国家科技实力的重要体现, 而运载火箭作为航天活动的基础工具, 其技术水平直接影响着航天任务的成本和效率。传统一次性运载火箭的高成本限制了航天活动的规模化发展, 因此可重复使用运载火箭 (RLV, reusable launch vehicle) 成为了当前航天领域的研究热点。运载火箭试验数据交互平台运行在新一代测发控系统中, 测发控系统通过各类软硬件接口接收运载火箭箭上、地面及发射场等数据, 并将数据实时显示及持久化存储。上一代火箭信息交互平台[1-3]相比于传统的显示判读软件[4-6]使用 B/S 架构并基于 Spring-Boot 框架开发, 虽然上一代平台解决了敏捷开发与移植性的问题, 但仍旧存在以下问题: (1) 平台通用化设

收稿日期:2026-03-27; **修回日期**:2026-04-29。
作者简介:彭炳锋(1998-),男,硕士,助理工程师。
引用格式:彭炳锋,王诗雯,邓 闯,等.基于微服务的可重复使用运载火箭试验数据交互平台[J].计算机测量与控制,2026,34(7):284-293.

计不足，无法适应多种数据库与多个型号的显示判读需求；(2) 服务在单模块或单节点出错后影响整个系统运行，服务高可用性有限；(3) 单体软件在新需求下扩展性有限，无法对单模块独立开发、测试和部署，难以实现快速迭代。为了满足新一代测发控系统智能化的发展需要以及解决上述不足，本文调研了微服务在其他领域的一些成功实践和本领域中的现有方法。文献 [7 - 8] 通过对服务进行划分，并依托 K8s 平台进行管理，实现了模块的解耦与服务的高可用性。文献 [9 - 10] 基于云原生技术，在航空空域管理自动化领域提出了一种面向服务的架构，该架构不但提供可验证数据服务，而且有效地降低了系统研发成本、缩短了新需求的交付与部署周期，实现了整体效率与灵活性的提升。文献 [11 -13] 为解决运载火箭型号高密度发射，提出了基于模型驱动的海量数据判读系统并设计了面向特定领域的数据判读建模语言，但该系统在扩展性、敏捷开发等方面存在不足。本文针对此背景，系统性地设计了一种基于微服务的运载火箭试验数据交互平台和数据判读语言，旨在实现以下几个目标：(1) 设计一种通用化程度更高的运载火箭试验数据交互平台，支持对数据显示内容及方式、数据判读判据及判读报告内容的可配置化；(2) 将软件中各模块进行解耦设计和冗余设计，保证平台服务的高可用性及高可靠性；(3) 平台设计对接多种数据库，保证平台能支持国产化数据库，满足自主可控要求。(4) 平台需求扩展性高，各模块可单独进行更改、测试及部署。

1　平台总体设计

与传统一次性运载火箭相比，可重复使用运载火箭在试验数据管理方面存在显著的核心差异：传统火箭仅需关注单次试验任务内的数据状态，数据管理聚焦于单批次试验的正确性验证；而可重复使用火箭需要支撑多次飞行试验、快速周转复用的业务场景，试验数据管理不仅要覆盖单次任务的状态监测，也需要实现跨任务的历史状态比对、部件疲劳损伤累积分析、寿命预测与复用决策支持。针对这一差异，平台在设计中针对性地强化了多批次试验数据的关联管理、历史数据对比分析、部件健康趋势跟踪等能力，以满足可重复使用火箭的独特业务需求。

1.1　微服务架构

微服务架构是把单体应用合理、自然地拆分成松耦合、可独立部署、以业务能力为划分依据的若干小型服务的分布式架构风格。目前广泛采用的微服务定义为[14-15]：将单一应用开发为一组小服务，各服务运行于独立进程，以轻量级机制通信，可独立自动化部署，支持异构技术栈和最小化集中管控。其核心特征包括：单一职责与限界上下文，独立开发、测试、部署，轻量级通信，技术异构与数据自治，容错与可观测性。与单体架构相比，微服务有利于提高迭代速度、加快部署频率、增强故障隔离与恢复能力。

运载火箭试验数据交互平台使用微服务架构后，允许多个团队根据需求同时快速独立开发功能服务，并保证每个服务在资源层面运行在独立的实例中，服务之间采用 RESTful API 实现轻量级调用，并使用消息中间件集群保证服务之间的解耦。

1.2　容器化技术

容器化是面向应用交付与运行环境标准化而设计的轻量化虚拟化方案，能够将应用程序、运行所需环境和配置文件等进行统一打包，从而实现在不同平台、不同环境的统一部署。容器以系统层面的隔离机制实现了资源及运行环境的解耦，在保证应用独立性的同时又极大降低了资源开销，因而部署密度和启动效率大幅提高[16]。和传统虚拟机不同的是，容器不需要模拟完整的操作系统、驱动等，而是可以直接使用宿主操作系统的内核，因此相比传统虚拟机具有轻量、高效、可移植等优点，已成为目前微服务架构的基础支撑技术之一。容器技术的发展有力地促进了云原生架构的落地，保障了服务的弹性伸缩、持续集成与部署[17-18]。

本文设计的运载火箭试验数据交互平台采用 Docker 技术搭建平台所需的容器化运行环境；同时基于 K8s 实现对容器进行统一编排与调度，将容器部署在多个服务器中，并根据需要进行动态扩缩容，当出现故障时自愈，进行不停机地滚动升级和服务集成。

1.3　系统架构

为适应可重复运载火箭多源试验数据实时交互、高可靠判读、通用化适配等多种需求，结合微服务架构与容器化技术特性，本文设计五层分布式系统架构，整体架构遵循“数据接入解耦、服务独立自治、存储分层适配、运维集中管控”的设计原则，实现箭上系统、地面测发控系统与平台服务间的高效数据流转，同时保障系统高可用、易扩展与自主可控特性。平台总体架构如图 1 所示，各组件分工明确、协同联动，覆盖地面测发控系统试验数据从采集到处理到显示与判读以及存储的全过程。

1.3.1　数据接入层

数据接入层是平台与运载火箭测试数据的交互接口，也是试验数据预处理与标准化处理的重要环节，主要包括“外部接口适配”与“数据帧解析处理”两大核心功能，通过屏蔽底层硬件不同、通信协议与数据格式的差异，实现多源试验数据的统一接收、校验以及标准化处理。该层对接运载火箭测发控系统、电气系统、测量系统、动力系统等箭上及地面终端设备，通过接收组播实现试验数据的实时传输，其中测发控系统的通讯管

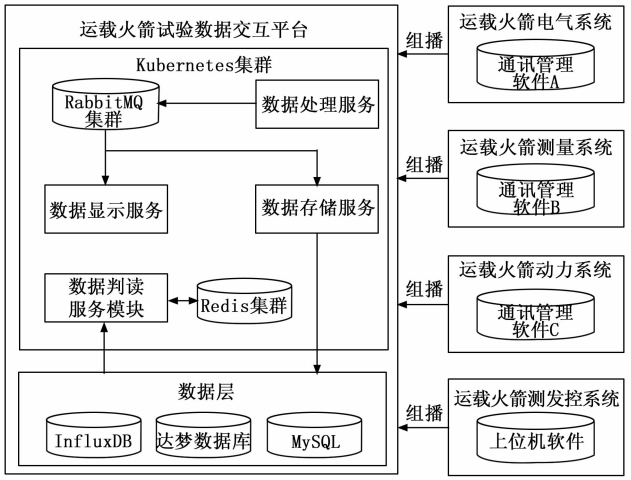

图 1 平台总体架构图

理软件 A、B、C 仅作为外系统对接的轻量化接口模块，不做复杂解析处理，仅负责完成外部数据的接收、初步校验以及转发，保障数据不丢失、无误码。

数据接入层使用数据处理服务完成数据的标准化处理工作，该服务先从配置数据库中读取各类数据帧的解析协议，包括帧头、帧尾、内容等核心配置信息；一个帧协议中包含一个唯一帧头和帧尾参数以及大量参数的信息，配置数据库使用内容表对这些参数的类型、存储位置、字节长度等信息进行存储。数据处理服务通过对帧头、帧尾双重校验，判断接收数据帧是否合规，剔除残缺、存在误码的异常帧；针对正常帧数据，使用对应的解析协议完成参数解析与参数输出格式标准化，将异构数据转换为平台设计的标准格式，并且发送至消息中间件集群，由中间件异步转发至核心业务层，实现服务之间的数据解耦。该层通过对数据处理服务设置多实例冗余，单一接口模块或解析节点故障不影响平台数据解析业务，同时支持帧协议的动态修改与新增，能快速适配多型号火箭试验数据帧协议。

1.3.2 核心业务层

核心业务层是平台的功能中枢，该层基于微服务的架构开发多个服务模块，从消息中间件中订阅数据。微服务开发以“单一职责、独立部署”的拆分原则，聚焦运载火箭试验数据的业务处理，拆分为数据显示服务、数据判读服务两大核心模块，各服务独立进行开发、测试与部署，在提高开发效率的同时也实现了服务之间的解耦。与单体架构相比，微服务的设计有利于对单一服务做迭代优化，功能升级周期大大缩短，也更契合航天试验场景中状态变更频繁、灵活扩展的实际需要。

其中，数据显示服务实现可配置化的显示界面，实现试验数据的流程状态显示、实时曲线绘制、设备状态监控等，通过结合 Grafana 开源组件的二次开发，支持用户自定义显示面板和显示控件，能适配不同型号火箭的试验显示需求；数据判读服务实现了一套新设计的数据判读引擎，用户通过编写判读脚本上传后，平台使用该脚本作为判据生成判读报告，实现试验数据的自动化判读。两大服务之间互不干扰，单个服务故障仅影响对应的平台功能，对其他功能没有影响，保障平台核心业务的稳定运行以及平台的可靠性。

此外，核心业务层新增历史试验数据对比判读功能，支持用户快速调取同型号、同部件的历史试验数据，实现当前试验状态与历史状态的自动化比对，为识别部件性能退化、状态异常提供支撑。并提供箭上电磁阀设备的性能退化评估。

1.3.3 消息中间件层

消息中间件层是连接数据接入层、核心业务层、数据层的关键，由 RabbitMQ 集群作为消息队列与 Redis 集群作为缓存组成，目的在于解决各层之间的异步通信、流量削峰填谷、数据共享等问题，同时能进一步提高微服务之间的解耦。如使用 RabbitMQ 集群完成异步消息队列机制，避免微服务之间调用导致的阻塞问题，确保试验环境下数据传输的流畅和稳定。

Redis 集群作为分布式缓存组件，缓存高频访问的试验流程信息、需要共享的状态量、判读规则脚本等数据。将关键信息保存在缓存中，保证了单服务实例宕机后其他实例能无感切换并接替该实例的工作，同时，使用缓存能大幅降低底层数据库的访问压力，提升数据读取的响应速度。此外，缓存层支持数据过期策略及清除策略与主从备份，保证缓存集群运行高效的同时提高数据可靠性，为核心业务层提供高效的缓存服务支持。

1.3.4 数据层

数据层聚焦于试验数据的可靠存储与管理，以多数据库兼容为设计目标，实现对 MySQL 数据库、InfluxDB、达梦数据库的适配与在线切换，通过针对各数据库编写对应的驱动包，完成对多种数据库的支持，满足对现有平台版本的兼容和自主可控替代需要。该层设计统一数据访问接口，屏蔽各类数据库的查询语法、存储结构等差异，核心业务层不再针对单一数据库定制开发，而是调用统一数据访问接口，即可完成对多种数据库的支持。

其中，关系型数据库（MySQL、达梦数据库）主要负责存储平台配置信息、用户权限、数据帧协议、判读脚本等结构化数据。MySQL 数据库用于满足对旧版本测发控系统软件的适配，达梦数据库则是面向未来运载火箭愈发急迫的自主可控需求；InfluxDB 数据库专门处理火箭试验中高频率、高吞吐量的各类传感器参数的存储及查询，可以大幅提高实时数据写入和历史数据查询效率。数据层通过缓存中间件或统一的查询接口对接核心业务，既兼容了现有版本数据交互平台的数据库，也支

持未来国产化数据库的替换，使得平台可以快速部署到现有运载火箭型号上，也能快速适配新研型号。

针对可重复使用运载火箭的多次试验场景，平台将不同试验任务、不同飞行次数的试验数据通过 InfluxDB 的标签（Tag）机制进行区分存储，实现多批次试验数据的分类管理与快速关联查询。

1.3.5　容器编排与运维层

容器编排与运维层是平台的平稳可靠运行的基础。该层基于 K8s 技术与 Docker 容器技术构建，实现微服务模块的标准化部署、自动扩容以及低成本运维。所有微服务开发完成后均打包为独立 Docker 镜像，该镜像包含服务运行所需的所有环境，可以实现一处打包，处处运行，解决了跨平台兼容性的问题。K8s 集群作为容器编排工具，为所有容器提供的统一调度、故障恢复、滚动更新及负载均衡等功能，并具备实现服务的自启动、批量部署和动态扩缩容的能力。针对运载火箭试验场景的高可靠性要求，平台单个服务实例故障时，K8s 不但有多个服务实例作为备份，而且会迅速启动新的实例以满足实例数量达到预设目标。同时可以在 K8s 集群中添加组件以实现服务监控、日志收集、节点性能监控等能力，大大降低运维成本。

2　平台详细设计

基于总体设计中的五层整体架构，平台针对核心功能模块进行详细设计说明，明确各模块的功能需求、实现逻辑、处理流程，详细介绍平台对运载火箭试验数据从解析、展示、判读到存储的全流程实现。结合地面测发控系统研制经验，详细设计完成了云容器化部署、无代码判读设计、智能算法解耦、强适配性等创新性工作。

为保障微服务拆分的合理性，平台采用领域驱动设计（DDD，domain-driven design）方法论进行服务划分，通过事件风暴梳理领域内的业务对象、业务行为与领域事件，明确各微服务的限界上下文，避免服务边界模糊导致的耦合问题。

在领域建模过程中，我们将试验数据交互平台的核心领域划分为 4 个独立的限界上下文，每个限界上下文对应一个独立的微服务，各自拥有独立的领域模型与数据自治能力，具体如下。

1）试验数据接入上下文：对应数据处理服务，该上下文的核心聚合根为标准化试验数据帧，聚焦于原始试验数据的接入、校验、解析与标准化转换，负责屏蔽底层硬件与协议的差异，该上下文内的业务逻辑变更仅局限于数据接入环节，不会对上层业务产生影响。

2）试验数据可视化上下文：对应数据显示服务，该上下文的核心聚合根为可视化面板配置，聚焦于试验数据的可视化展示、监控面板的配置与管理，仅依赖标准化后的试验数据，无需关心数据的来源与解析过程，支持独立的可视化功能迭代。

3）自动化判读上下文：对应数据判读服务，该上下文的核心聚合根为判读规则与判读报告，聚焦于判读规则的管理、试验数据的自动化判读、判读报告的生成，独立封装判读逻辑，可独立进行判读算法的迭代与优化。

4）算法分析上下文：对应电磁阀等专项分析服务，该上下文的核心聚合根为算法分析模型，聚焦于电磁阀部件的专项数据分析、健康度评估，支持独立的算法模型迭代，无需改动核心业务逻辑。

针对服务间的数据一致性问题，平台采用异步事件驱动的最终一致性策略，通过消息中间件实现服务间的解耦，避免同步调用带来的事务耦合问题，同时通过消息持久化、消息重试机制保障数据的可靠流转，满足试验数据的高可靠传输需求。

平台跨服务数据流转如图 2 所示，整个业务流程基于异步事件驱动的架构模式构建，实现了多服务间的解耦协作。原始试验数据由测发控系统完成采集后，首先传输至数据处理服务，由该服务完成原始数据的帧校验、协议解析与标准化转换；处理完成的标准化试验数据，一方面持久化至数据库中，完成原始试验数据的持久化存储，为后续的历史数据比对、部件健康趋势分析提供数据支撑；另一方面，标准化数据被封装为领域事件，发布至 RabbitMQ 消息队列，完成数据的分发。平台基于发布－订阅模式，消息队列将标准化数据事件异步推送至订阅了该主题的微服务。数据判读服务调用数据存储服务的接口，获取特定型号、特定试验的数据，并从 Redis 缓存中获取数据判读规则，完成数据判读功能，并将判读结果存储到关系型数据库中。该数据流机制实现了服务间的完全解耦，避免了同步调用带来的阻塞与耦合问题，同时支撑了各服务的独立弹性伸缩，有效适配了可重复使用运载火箭高密度试验、高吞吐数据处理的业务需求。

2.1　数据接入层详细设计

数据处理服务是数据接入层的核心，承担着各系统二进制数据流到平台标准化数据的转换功能，是连接试验数据入口、保障数据质量、支撑核心业务的关键服务。该服务采用无状态微服务＋实例集群的架构，基于 K8s Deployment 实现多实例冗余部署，单个实例故障可通过 K8s 自愈机制在 30 s 内重启并恢复业务，其他冗余实例无缝接替该实例工作，全程无数据丢失、无处理中断。服务主要流程包括协议库预加载与热更新、组播数据接收与格式转换、双重数据帧合规验证、解析解码、数据标准化封装、数据推送至消息队列等，如图 3 所示。

为实现帧解析规则的精细化、持久化管理，设计帧

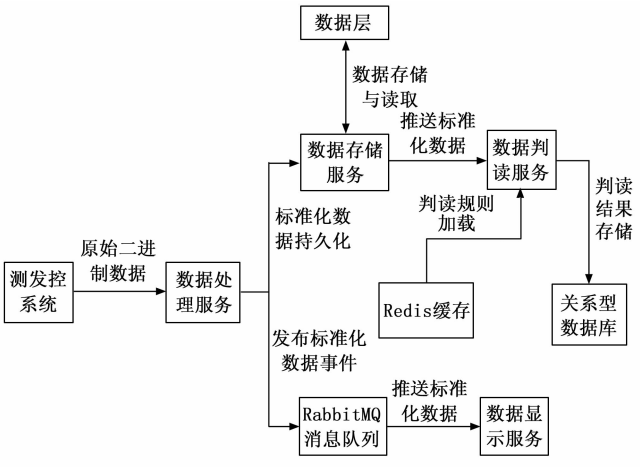

图 2 平台数据流图

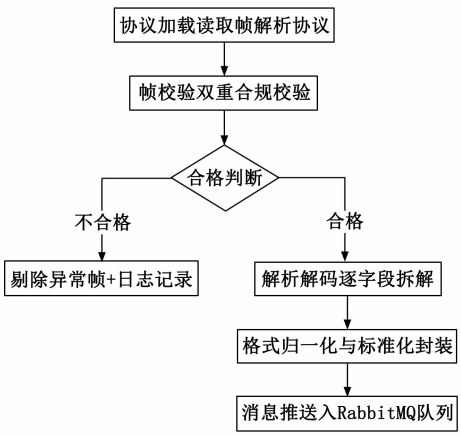

图 3 数据接入层处理流程图

协议核心数据表体系，包含帧协议表、帧头表、帧尾表、参数内容表。针对帧头、帧尾的工程化解析需求做字段细化，帧头新增前导码、帧长、校验和、组播编号等属性，帧尾补充校验算法、尾校验码等字段；表间关联关系遵循：帧协议与帧头、帧尾为一对一关联，帧协议与参数内容为一对多关联，各子表通过外键绑定所属帧协议主键。

2.1.1 协议库加载与热更新机制

数据处理服务启动时建立与关系型数据库（达梦/MySQL）的连接，采用启动时预加载＋增量更新模式读取帧解析协议数据库，避免频繁数据库访问导致的性能损耗。协议库采用四级结构化存储：帧协议主表存储协议基础信息，帧头表存储前导码、帧长信息、组播编号等内容，帧尾表存储尾校验码等尾部配置，内容表存储数据帧内所有参数的解析元数据（参数名，参数类型，参数长度，偏移量等）。为适配地面测发控系统数据传输协议迭代，服务运行协议监听线程，实时监听数据库协议变更日志，检测到新增、编辑、停用协议时，无需重启服务即可动态加载最新数据传输协议，更新延迟控制在一分钟内，不仅不影响运载火箭试验，而且不影响当前数据处理流程。

2.1.2 数据帧双重校验逻辑

为从源头拦截异常数据，服务采用帧头帧尾的双层校验逻辑，双重校验均通过后方可进入解析环节。第一重帧头校验：根据对应协议配置的前导码、帧长、校验码等信息，实时比对接收帧头的实际信息，剔除帧头异常的错误帧；第二重帧尾校验：基于协议配置的帧尾十六进制码，从帧尾匹配唯一标识，锁定完整数据帧边界，同时进一步获取数据帧真实帧长信息。对于校验失败的异常帧，服务在不进行解析的同时将异常数据帧源码、异常情况、时间戳、数据帧类型等写入异常日志，供用户后续查询与分析。

2.1.3 解析解码流程

正常数据帧进入解析环节后，服务根据通讯协议配置进行解码操作，全程采用位（bit）级精准定位，适配大小端字节序、十六进制转码、各数据类型解析等多种场景。首先根据帧头表和帧尾表中配置内容，将正确数据帧帧头中组播编号等有效内容作为该数据帧所有参数的通用信息；再按照内容表配置的偏移量、长度、数据类型、大小端等，逐个参数地拆解数据帧，转换为整型、浮点型、布尔型、枚举型等不同类型的数据；针对浮点型参数统一转换为 Double 类型防止精度丢失，针对枚举类型参数完成 16 进制数值到状态文字的转义翻译（比如将 0x01 映射为“起飞”）。

2.1.4 标准化数据封装与消息推送

数据帧解析完成后，服务将不同帧的数据统一封装为平台设计的标准 JSON 格式数据结构，核心字段包含：参数唯一标识（命名空间. 参数名）、参数名称、实测值、单位、时间戳（数据帧自带和北京时间两种）、数据源编号、帧编号、协议编号、设备编号、解析状态，确保核心业务层的服务可直接消费使用。封装完成的数据通过异步模式推送至消息队列集群，按照数据类型、火箭子系统、试验角色等划分队列，实现数据分流转发，同时常规参数、关键参数、时序数据分队列区分，避免高频参数阻塞队列影响重要参数队列运行。

2.2 核心业务层

2.2.1 数据显示服务

数据显示服务依托 Grafana 开源可视化平台搭建，深度整合平台消息队列与数据库数据源，摒弃传统定制化界面开发周期长、扩展性差、复用率低的局限性，实现试验数据的可配置、可拖拽、可共享、多形态展示，满足不同岗位、不同型号火箭的实时监测、状态监控需求，兼容定制化流程屏与通用化仪表盘双模式展示。

Grafana 使用数据源插件对接 RabbitMQ 集群，不需要重新开发数据接口，使用开源插件即可显示数据处理服务封装后的数据。Grafana 面板采用模块化拖拽设

计，支持用户自定义创建专属监测页面，支持折线图、柱状图、表格、仪表盘、热力图等可视化组件，可灵活配置数据刷新周期、时间范围、显示精度等内容，满足用户对实时显示性能和全程显示数据的需求。针对运载火箭试验场景，各系统用户可以在试验前根据重点关注的参数和内容，自行设定显示的数据；此外用户可根据试验需求拖拽组件、调整布局，快速搭建适配当前任务的个性化数据显示界面，且支持面板共享以及多工位同步查看。

2.2.2　数据判读服务模块

2.2.2.1　无代码数据判读服务

针对运载火箭数据判读需求变动频繁、定制化需求高的特点，该服务模块设计了一套扩展性强、灵活度高的判读语句体系，以实现无代码化判读流程的搭建。其核心语法规范如下。

1）参数唯一标识符规范：构建“命名空间. 参数名称”的唯一参数映射规则，将箭上和地面的各类试验参数（如压力、温度、电流、时序）与唯一标识符绑定，该标识符通常与数据处理模块的标准数据结构中定义的标识符保持一致，平台同时维护一张数据表用于存储所有参数的标识符以及该参数存储位置的映射。例如标识符“地面电源 . D1 电流”、“综控时序 . Tk1y”等，标识符全局唯一，并映射数据存储位置，如关系型数据库的哪个表或者时序数据库的哪个标签中。

2）基础运算语法：兼容算术运算（＋、－、×、÷）、比较运算（＜、＞、＝、≥、≤、≠）、逻辑运算（&&、‖、!）3 大类基础运算，支持单参数阈值判定、多参数联合逻辑判定，例如：

namespace. n1＞0. 5&&namespace. n2＞25

3）高级判读语法：定制运载火箭试验判读所需的专属高级语法，使用关键字的形式进行调用，如“＃FREQUENCY”关键字用于计算发动机点火阀的频率，“＃SALTATIONTIME”关键字用于计算参数发生跳变的时间等。关键字部分支持快速扩展，扩展时仅需实现关键字接口而不需要改动其他代码，保证了判读服务高效的敏捷开发。新增“＃HISTORY _ COMPARE”关键字用于实现历史试验数据对比判读功能，用户可通过该关键字指定目标参数与历史试验任务标识，平台自动调用平台的历史数据查询函数，拉取对应历史试验的同参数数据，自动完成当前试验数据与历史数据的差值比对、趋势比对，实现跨试验任务的状态一致性判读，支撑可重复使用火箭的状态复现与异常识别。

4）语句合法性校验规则：内置语法解析引擎，对编写的判读语句进行实时校验，过滤关键字错误、运算符号缺失、逻辑冲突、参数不匹配等无效语句，校验通过后方可入库生效，避免执行阶段出现语法崩溃。

平台支持两种判读语句的修改方式，分别是页面在线修改和文件导入。用户可以点击进入判读语句中进行修改，根据需要的判读逻辑自定义判读语句，修改完成后平台会自动根据修改后的语句生成用户所需要的判读报告。文件导入的方式则是使用 Excel 进行编辑，完成判读报告脚本的编写，将文件导入后平台可以自动使用该脚本生成对应的运载火箭数据判读报告。平台判读脚本修改流程图如图 4 所示，用户通过支持的两种修改方式完成判读脚本的修改后，平台自动验证脚本的语法合规性，并完成规则入库及热更新，不再和现有数据平台一样存在需要改动代码的情况。

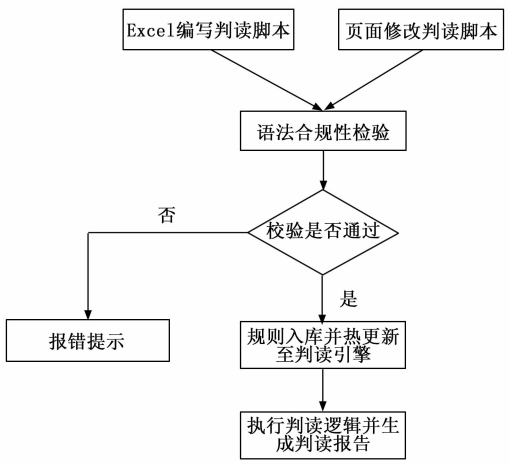

图 4　判读脚本修改流程图

2.2.2.2　电磁阀数据分析服务

电磁阀作为运载火箭上的关键执行机构，其运行状态直接影响任务的成败，因此平台搭建电磁阀数据判读专项服务，采用电磁阀数据解析服务与电磁阀标点算法服务解耦的设计，将智能特征点标注算法封装为独立微服务，通过 RESTful API 与电磁阀解析服务交互。该模块聚焦电磁阀电流、电压、开启时间、关闭时间、稳态电流、动作时序等核心参数。除常规无代码判读语法外，该模块默认新增基于极值点和导数的特征点寻找算法，可快速识别电流曲线起始点、开启点、稳态点、关闭点等关键特征。在默认特征点标点算法外，平台使用 Python 独立开发了基于可变大小滑动窗口的特征点标注算法，封装为 Docker 容器部署为独立的微服务，支持独立迭代升级，不影响核心判读业务稳定性，用户通过配置选择不同的标注算法，算法开发方面可扩展性强，支持的语言不限，只需提供统一的接口即可。电磁阀数据分析服务整体框架如图 5 所示。

在此基础上，该服务针对可重复使用火箭的部件寿命监测需求，基于运载火箭故障诊断数据仓库[19]，开发了电磁阀健康度趋势分析算法模型。模型通过分析该电磁阀历次飞行试验的特征数据，包括开启时间、稳态电

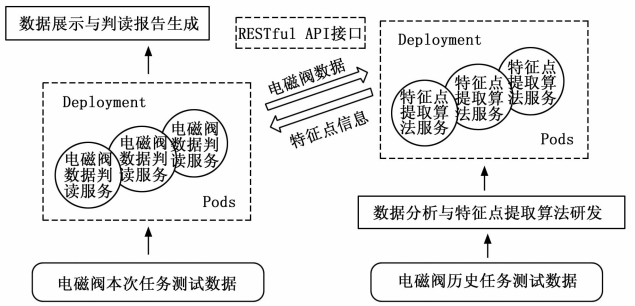

图 5 电磁阀数据分析服务架构与数据流程

流偏差、关闭时间等多维度特征，构建部件健康状态的时序评估模型，实现对电磁阀健康度的趋势跟踪与剩余寿命预测，为火箭部件的快速复用决策提供数据支撑。

2.3 消息中间件层

消息队列集群与缓存集群是本试验数据交互平台实现微服务解耦、数据高吞吐并发流转、低延迟显示的核心，消息中间件层承接数据处理层与核心业务层的数据交互，针对运载火箭试验数据高密度采样、强实时性、高可靠性、全程数据无误的要求，采用 RabbitMQ 高可用集群与 Redis 3 主 3 从哨兵集群的双集群冗余架构。该设计严格遵循分布式系统高可用、低耦合、可扩展的设计准则，用于解决平台内多微服务接口串行调用造成的阻塞、流量峰值冲击导致单节点宕机等核心问题。

2.3.1 RabbitMQ 高可用消息集群设计

本平台选用基于 AMQP 消息队列协议的 RabbitMQ 作为异步消息队列中间件，同时开启 RabbitMQ 的 Stream 类型队列，用于对接 Grafana 开源显示组件进行显示。针对运载火箭试验多源数据高频次并发接入、微服务异步解耦的需求，采用 3 节点集群模式部署，规避单节点单点故障风险，满足运载火箭试验场景的可用性指标。集群通过队列镜像冗余复制、节点状态实时同步、故障自动主从切换 3 大核心机制实现高可用，集群核心数据采用磁盘持久化存储，并使用 NFS 提供网络共享存储，保证数据不丢失，进而保障集群重启、节点宕机场景下配置信息与消息队列不丢失。相较于单节点消息队列部署方案，集群架构在同时处理数据量上明显提升，可支撑每秒 10 万级标准化封装数据包的处理，消息传输时延显著低于单体架构上的消息队列，适配运载火箭试验 20ms/帧高密度采样数据的实时流转需求，保障平台数据链路的通畅与稳定。

2.3.2 Redis 3 主 3 从集群设计

针对平台缓存高频访问的试验流程信息、需要共享的状态量、判读规则脚本需求，选用 Redis 内存数据库构建 3 主 3 从高可用缓存集群，严格遵循分布式缓存高可用与负载均衡设计规范。集群采用 3 主 3 从数据节点架构，主节点承担数据读写任务，从节点通过异步增量复制同步主节点数据，实现读写分离与数据冗余备份。

2.4 数据层

数据持久化层通过统一数据访问抽象层＋数据库驱动封装＋多数据库存储适配的设计模式，实现对 MySQL 关系型数据库、InfluxDB 时序数据库、达梦国产化数据库的兼容适配，解决多版本测发控系统软件兼容、国产化替代、跨数据库无缝切换，实现核心业务与底层数据库之间的隔离，大大降低新需求的开发难度，整体架构如图 6 所示。

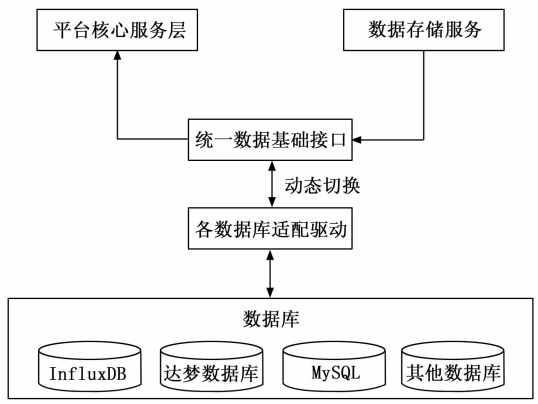

图 6 数据层总体架构图

2.4.1 统一数据访问层核心架构设计

为屏蔽不同数据库的语法与模型差异，平台设计数据库适配驱动，核心包含 SQL 方言解析器、动态连接池管理两大核心组件，具体如下。

1）SQL 方言解析器：该组件负责将上层业务传入的标准 SQL 语句进行语法解析，生成通用的抽象语法树（AST，abstract syntax tree），随后根据当前目标数据库的类型，将通用语法树转换为对应数据库的原生查询语法：针对 MySQL 与达梦等关系型数据库，完成分页语法（LIMIT）、函数语法的方言转换；针对 InfluxDB 时序数据库，则完成关系模型到时序模型的映射转换，将标准 SQL 的表、字段映射为 InfluxDB 的 Measurement、Tag 与 Field，同时将 SQL 查询语句转换为 InfluxDB 原生的 Flux 查询语法，彻底屏蔽时序数据库与关系型数据库的查询语法差异。

2）动态连接池管理：平台为每种数据库维护独立的动态连接池，连接池支持基于负载的自动扩缩容：试验高峰期，针对高吞吐的时序数据写入，自动扩容 InfluxDB 的连接池数量；空闲时段则自动收缩连接数，避免资源浪费。同时连接池内置了健康检查机制，定期对连接进行有效性校验，自动剔除失效的连接，保障数据库连接的稳定性。

2.4.2 跨数据库一致性保障机制

针对多数据库的协作场景，平台采用最终一致性的事务保障策略，避免了分布式事务的复杂度：写入操作

首先写入本地的预写日志，随后异步写入目标数据库，同时内置了重试机制与幂等性保障，即使数据库临时故障，也不会导致数据丢失，待数据库恢复后，可自动重试未完成的写入操作，保障数据的最终一致性。针对跨库的长事务场景，平台采用 Saga 模式（一种用于管理分布式系统中长事务的设计模式），将分布式事务拆分为一系列本地事务，通过事件驱动的方式完成事务的流转，同时提供了事务补偿机制，在异常场景下可完成事务的回滚，保障跨库操作的一致性。

2.4.3　数据库动态切换与数据迁移机制

为支持数据库的无缝切换，平台设计了全流程的自动化数据迁移与一致性校验机制：当用户触发数据库切换操作时，平台首先开启双写模式，新的写入请求会同时写入旧数据库与新数据库，保障新数据的同步；随后平台启动异步的历史数据迁移任务，将旧库中的历史数据全量迁移到新库中；迁移完成后，平台会自动进行一致性校验：通过行级校验，验证新旧库的数据一致性，校验通过后，平台自动将读写请求切换到新数据库，整个过程无需停止服务。若切换后出现异常，平台还支持一键回滚，快速切回旧数据库，保障业务的连续性。

2.4.4　时序数据专属存储设计

针对 InfluxDB 时序数据库，平台针对可重复使用火箭的多批次试验数据设计了专属的存储 Schema：将箭体编号、试验任务 ID、飞行次数作为数据的标签（Tag）维度，将传感器采集的各类参数作为字段（Field）维度。这种存储方式下，同一部件的多次试验数据可通过标签进行快速区分与关联，标签自带索引特性，可支持千万级时序数据下的秒级多试验数据关联查询，为历史数据比对、寿命趋势分析提供高效的存储支撑。

2.5　容器编排与运维层

本平台采用 K8s 作为云服务容器编排器，将平台所有微服务、消息中间件、缓存集群均封装为标准 Docker 容器，实现全组件和服务的统一部署、低成本运维与自动扩缩容。该层设计彻底解决传统部署模式下环境适配性、运维困难的痛点，同时针对运载火箭试验期间长时间不间断运行、新火箭研发周期大大缩短的需求，构建具备故障恢复、服务冗余、集中管理的容器化运维模式，保障平台的稳定运行与高效的敏捷开发。

K8s 采用声明式 API＋控制器闭环的核心架构，其核心组件主要包括：Etcd、Apiserver、Scheduler，controller-manager 等。Etcd 作为一致且高度可用的键值存储，用作 K8s 的所有集群数据的后台数据库存储平台服务配置、节点状态、调度规则等核心信息，基于 Raft[20] 一致性算法保障集群元数据强一致性；ApiServer 作为集群唯一交互入口，完成身份认证、权限校验、请求路由等操作，主要负责处理接收请求的工作；Scheduler 是控制平面的组件，负责监视新创建的、未指定运行节点的 Pods，并选择节点来让 Pod 在上面运行；Controller Manager 是控制平面的组件，负责运行控制器进程。

本平台结合各服务特性，采用不同的部署策略：针对数据处理服务、数据显示服务、数据判读服务等无状态的服务，采用 Deployment 进行部署，保证多实例冗余、滚动更新、自动扩缩容等功能，确保这些核心功能试验期间运行稳定；针对 RabbitMQ 集群、Redis 集群等有状态组件或服务，采用 StatefulSet 部署，保障集群有序部署的需求；通过 Service 服务屏蔽 Pod 动态 IP 变动，实现平台微服务间的服务发现、服务通讯以及服务的负载均衡。

2.6　安全架构设计

可重复使用运载火箭的试验数据属于敏感数据，随着多次试验的数据积累，数据的价值与泄露风险同步提升，因此平台设计了基于零信任模型的安全架构，覆盖身份认证、访问控制、审计日志等流程，保障试验数据的安全性。

2.6.1　微服务间零信任身份认证

平台采用零信任的安全模型，默认不信任任何内部或外部的服务与用户，所有的服务间调用、用户访问都需要进行身份认证。针对微服务间的通信，平台采用双向传输层安全（mTLS）＋JWT 令牌的双认证机制：每个微服务都拥有独立的身份证书，服务间建立连接时，首先通过 mTLS 完成双向的身份认证，验证对方的服务身份；随后，服务调用的请求中携带 JWT 令牌，令牌中包含了服务的身份与权限信息，接口层会对令牌进行校验，验证服务的访问权限，避免非法服务的仿冒与未授权访问。

2.6.2　细粒度访问控制与审计

平台实现了基于角色基础访问控制（RBAC）的细粒度访问控制机制：针对用户，根据岗位与职责划分不同的角色，不同角色拥有不同的数据访问权限，例如普通测试人员仅可访问自身负责子系统的试验数据，管理员可访问全量数据；针对微服务，同样基于角色划分了服务的访问权限，每个服务仅可访问自身业务所需的接口与数据，避免了服务越权访问的风险。

同时，平台实现了全链路的审计日志机制：所有的用户操作、服务访问、数据读写操作，都会生成不可篡改的审计日志，日志中记录了操作主体、操作时间、操作内容、访问的数据范围等信息，审计日志独立存储在专用的审计存储系统中，无法被平台的常规操作修改，支持事后的安全审计与异常行为追溯，保障操作的可追溯性。

3 测试结果与分析

为验证本平台是否满足设计目标，本章节搭建贴合真实使用场景的测试环境、制定测试方案、模拟采集测试数据、对比设计指标达标情况，客观评估平台核心能力。

3.1 测试环境搭建

本次测试严格模拟可重复使用运载火箭地面测发控真实工况，采用“仿真数据源＋容器化集群＋终端设备”的组合环境，兼顾测试可控性与工程真实性，软硬件配置与网络环境如下。

硬件环境：服务器采用 4 台计算机（8 核处理器、16 GB 内存）搭建 K8s 高可用集群（1 台控制节点，3 台工作节点），1 台数据仿真服务器模拟箭上、地面测发控系统多源数据输出，2 台客户端终端用于功能操作与结果查看；

软件环境：操作系统为银河麒麟国防版 V10 SP2，容器引擎为 Docker 18.06，编排工具为 Kubernetes 1.24，数据库采用 MySQL 8.0、InfluxDB 2.6、达梦 8 国产化数据库，中间件为 RabbitMQ 3.9、Redis 6.2，可视化组件为 Grafana 9.0；

网络环境：千兆以太网组网，开启组播通信模式，模拟试验现场数据传输链路，设置网络抖动、瞬时丢包等异常场景，验证平台容错能力。

3.2 判读脚本热更新功能测试

基于 Excel 编写的上下限判读、时序判读、多参数联合判读等判读语句，覆盖发动机遥测参数、时序等判读场景，测试判读规则导入、语法校验、自动判读、报告生成等，测试结果如下：判读语句语法校验正确率 100％，规则热更新延迟≤1 s，异常参数识别准确，无漏判、误判情况；无代码配置流程简单易用，非研发人员可独立完成规则编辑，相较于传统硬编码或硬编码加配置判读，配置效率大幅提高。充分验证本文设计的判读语句体系的实用性与通用性，实现无代码化判读逻辑搭建。

3.3 系统性评估对比测试

模拟 20 ms 一帧、帧长 30 720 字节，高密度采样数据注入，实测单帧数据平均处理时间为 11.6 ms，满足小于 20 ms 设计目标，满足处理大规模试验数据的要求；RabbitMQ 消息集群平均传输时延 1.5 ms，消息无拥堵、无延迟，配合消息持久化机制，保障数据传输高效可靠。平台采用微服务并行处理、容器化弹性扩容、消息队列异步解耦多种设计，数据流转效率比传统单体系统提高了 60％以上。

表 1 部分性能测试展示

性能指标项	设计目标	实测结果	达标情况
单帧平均处理时延	＜20 ms	11.6 ms	达标
中间件平均传输时延	＜5 ms	1.5 ms	达标
最大并发用户访问量	＞100	110	达标

模拟工作节点故障宕机，将一台工作节点网线拔出模拟实际工况中计算机宕机、网络故障等情况。设计要求平台工作不中断，对试验流程无影响，并对比单体软件版本的运载火箭数据交互平台，如表 2 所示。经验证，平台相比现有版本大幅提高可靠性，满足设计要求。

表 2 可靠性及故障恢复测试

性能指标项	设计目标	实测结果	现有平台
Pod 恢复时间	＜30 s	25 s	无
单节点故障后单帧处理时延	＜20 ms	16.3 ms	无
法处理单节点故障后平台功能	功能正常	功能正常	功能异常

此外，为系统性评估本平台的优势，本平台从开发效率、可靠性、功能扩展性等多个维度与传统的单体架构平台进行量化对比，同时验证无代码化设计带来的效率提升，对比结果如表 3 所示。

表 3 微服务架构平台与传统单体架构多维度对比

评估维度	传统架构	本平台	提升幅度/％
分析功能开发周期	30 天	7 天	缩短 76.7
单节点故障恢复时间	300 s	30 s	缩短 90
判读规则配置周期	3 天	1 天	缩短 66.7
算法分析服务扩展	不支持	支持	—

在开发效率方面，无代码化的判读规则设计，使得测试人员无需研发人员介入，即可自助完成判读规则的配置，将判读规则的配置周期从原来的至少 3 天缩短至 1 天，人力成本降低 66.7％以上；同时微服务的解耦设计，使得新功能的开发周期从 30 天缩短至 7 天，大幅提升了开发效率。

平台通过微服务架构的设计大大提高了平台的可靠性，传统架构节点故障需要人工切换至备份服务器，人工排除故障至少需要五分钟，而基于 K8s 的微服务架构平台则会自动启动 pod，单节点故障恢复时间不超过 30 s，同时故障不会影响服务的运行，大幅提升了平台的可靠性。

此外，平台通过微服务的解耦设计大大提高平台的功能扩展性。而传统架构在设计时并没有设计算法分析服务，因此不支持可重复运载火箭的部件衰减分析等功能的扩展。

3.4 多数据库兼容性测试

通过统一数据访问抽象层，实现 MySQL、达梦 8、InfluxDB 三类数据库的无缝切换，切换过程无需重启服务、无需修改业务代码，数据读写、存储、查询功能正常，结构化数据与时序数据存储精准无误。该结果验证平台底层数据库适配机制的有效性，兼顾老旧系统兼

容与国产化替代需求，满足航天领域自主可控要求，用户还可以在软件界面上进行数据库的切换功能。

3.5 效果分析

本文设计的基于微服务的运载火箭数据交互平台在实验环境下进行了测试，并在某可回收火箭型号中进行应用，相比于现有软件在与运载火箭判读方对接时需要频繁改动代码以适应需求更改不同，基于无代码化信息交互平台可以根据用户的判读要求进行定制化脚本编写，解决了现有平台的痛点。同时，集群状态部署的数据交互平台具有可靠性高，性能好等特点，相比现有平台在设计更复杂，数据规模更大的情况下，仍有不错的性能提升。此外，平台针对不同数据库开发对应专属驱动包，保证平台可以在纯国产化替代场景和现有型号软件上均能正常运行。

4 结束语

本文设计的微服务架构试验数据交互平台，有效解决了现有单体平台的多个痛点，实现了火箭试验数据的通用化、配置化交互与智能化、自动化判读，兼顾多型号通用与自主可控要求。平台虽已通过测试验证，但在复杂参数智能判读等方面仍有优化空间，例如后续可以接入大语言模型，判读人员提供提示词后平台智能化编写判读脚本并生成报告等；同时进一步完成算法开发，利用平台扩展性强的特性，为用户提供更准确、更高效、更丰富的算法，为可重复使用运载火箭的新需求提供分析服务。该平台具备良好的应用价值，在商业航天兴起的今天，能够为各类运载火箭地面试验提供高效可靠的数据交互服务，同时为运载火箭地面测发控技术提质升级。

参考文献：

[1] 陈锴迪，欧阳李青，马玉璘，等. 基于 SpringBoot 的运载火箭信息交互指挥平台 [J]. 计算机测量与控制，2023，31 (3)：247－254.

[2] 任月慧，马小龙，马宗瑞，等. 统型测发控系统一体化发射指挥模式研究 [J]. 导弹与航天运载技术（中英文），2023，(3)：122－126.

[3] 欧阳李青，程宗荣，彭炳锋，等. 长征二号丁运载火箭全箭试验数据决策支持平台建设 [J]. 上海航天（中英文），2025 (z1).

[4] 吴　睫，王　可，卢逸斌，等. 运载火箭测试数据自诊断设计 [J]. 电子设计工程，2016，24 (16)：73－75.

[5] 陈文浩. 以通用化软件模式判读运载火箭全箭信息的系统及方法 [P]. 中国：201310133，2013-04-17.

[6] 刘　伟. 一种运载火箭全箭测发控信息可定制监控的监控系统及其监控方法 [P]. 中国：201610298000，2016-05-06.

[7] 钟伟宏，孙甲琦，朱宏涛，等. 基于 Kubernetes 的航天地面应用软件架构设计与实现 [J]. 遥测遥控，2021，42 (3)：9.

[8] 董宇航，焦冬冬，尹志锋，等. 基于云原生的航天发射一体化指挥显示系统数据引擎设计 [J]. 电子技术应用，2024，50 (7)：118－124.

[9] REN L，CASTILLO-EFFEN M，BENJAMIN，et al. Cloud computing for air traffic management-framework analysis [C] //2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC). New York：IEEE，2013.

[10] SOLOMON A，CRAWFORD Z. Transitioning from legacy air traffic management to airspace management through secure，cloud-native automation solutions [C] //2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC). Piscataway：IEEE，2021.

[11] 王　毅，吴晓蕊，封慧英，等. 基于模型驱动的海量数据判读系统研究与实践 [J]. 计算机测量与控制，2018，26 (12)：5.

[12] 周辉峰，王一雄，曾少龙，等. 运载火箭靶场测试数据自动判读方法 [J]. 四川兵工学报，2013，34 (4)：43－46.

[13] 梁宇坤，王晓君. 基于一体化、通用化、自动化的测量系统地面测发控设计 [J]. 导弹与航天运载技术，2022 (2)：80－83.

[14] BALALAIE A，HEYDARNOORI A，JAMSHIDI P. Microservices architecture enables DevOps：migration to a cloud-native architecture [J]. IEEE Software，2016，33 (3)：42－52.

[15] DRAGONI N，GIALLORENZO S，LAFUENTE A，et al. Microservices：yesterday，today，and tomorrow [J]. Present and Ulterior Software Engineering，2017 (6)：195－216.

[16] MERKEL D. Docker：Lightweight linux containers for consistent development and deployment [J]. Linux Journal，2014，239：2.

[17] BURNS B，GRANT B，OPPENHEIMER D，et al. Borg，omega，and kubernetes [J]. Communications of the ACM，2016，59 (5)：50－57.

[18] LI J，ZHANG H，WANG L. The performance analysis of docker and rkt based on kubernetes [C] //2017 13th International Conference on Natural Computation，Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)，2017：1872－1876.

[19] 陈　卓，汪灏，平佳伟，等. 面向火箭数据分析和故障诊断的数据仓库设计 [J]. 计算机测量与控制，2024，32 (1)：51－56.

[20] ONGARO D，OUSTERHOUT J. In search of an understandable consensus algorithm [C] //2014 USENIX Annual Technical Conference (USENIX ATC 14). Philadelphia：USENIX Association，2014：305－319.