文章类型：论文|刊号（ISSN）：2972-4236(P) / 2972-4244(O)

微服务架构在现代软件开发中的应用与挑战

曹严清
北京云科伟业信息科技有限公司
DOI:10.12238/acair.v2i3.8647

[摘　要] 微服务架构运用独立模块式拆分业务逻辑以及轻量级通信协议分散运行方式,成功克服传统单体架构的扩展难及紧耦合问题,具有粒度细小、专注单一功能及优异的扩容潜力。在电商平台、大数据分析、金融业机构、医疗保健与工业企业领域受到热捧,显著提升系统稳定性与易维护性。
[关键词] 微服务架构；软件开发；实践应用；挑战
中图分类号：TP31　文献标识码：A

Application and challenge of micro-service architecture in modern software development

Yanqing Cao
Beijing Yunke Weiye Information Technology Co., Ltd

[Abstract] Micro-service architecture successfully overcomes the problems of difficult expansion and tight coupling of traditional single architecture by using independent modular split business logic and lightweight communication protocol decentralized operation mode, and has fine granularity, single function and excellent expansion potential. It is popular in the fields of e-commerce platform, big data analysis, financial institutions, health care and industrial enterprises, which significantly improves the stability and maintainability of the system.
[Key words] Micro-service architecture; Software development; Practical application; challenge

前言

作为现代软件设计方式的微服务架构正逐渐成为大家的首选,应用程序分解为一系列小而独立的服务,在开发部署和维护上都变得更加灵活高效,既提高了开发团队的工作效率,又增强了系统的可扩展性和容错能力。所以微服务架构在软件工程中扮演着举足轻重的角色。同时,为了更好地发挥微服务架构的优势,对微服务的设计和实现也提出了更高的要求。

1 微服务架构简述

微服务的核心概念在于将系统业务逻辑细分为若干模块,具备粒度适中、聚焦单一任务、采用分布式进程、轻量化通信协议、松散耦合、独立运行以及支持多版本、多实例部署等特性。例如,在企业大数据平台构建过程中,可将前期数据获取、数据分析、用户及员工数据可视化报表展示等功能单独拆分,各模块使用独立代码进行开发,通过接口方式实现通信。若某一模块出现故障,可利用服务降级策略在系统内进行提示,提高用户体验。通过微服务架构,系统中的各个模块被划分为独立的服务,每个服务只需关注自身功能即可,无需担忧对其他服务产生影响,符合现代开发中持续集成、持续迭代的理念[1]。

2 微服务架构在软件开发领域的应用

2.1微服务架构在电子商务平台的实践

随着互联网购物热潮兴起,电商平台规模不断扩大。面对日渐增长的业务需求,各电商平台开始采用微服务架构。其特点在于,将原有单体应用进行细化拆分,例如,将订单模块交由专门团队开发,而购物车则由另一个团队负责。各团队独立于其他功能,利用RPC实现通信,以保证系统稳定性及高效运营。

2.2微服务架构在大数据分析领域的应用

大数据具有多种多样的数据形式、属性以及频繁变化的特性。传统上,仅需运用一种如MYSQl的数据库,但在微服务模式下,各服务模块可拥有各自的数据库,包括SQL类型的Oracle、MYSQL、PostgreSQL等关系型存储数据库,或NoSQL类型的MongoDB、Cassandra、Amazon DynamoDB等。企业可根据自身业务需求,选择适合的NoSQL数据库。针对不同模块,选用相应的开发语言,并由不同团队负责不同维度的应用服务开发。这种模式将庞大的应用系统细分为多个子系统,分配给不同的数据支持团队进行开发,如订单数据处理分析,即可交由订单大数据处理团队完成。该团队可运用擅长的语言和方法进行大数据分析,再将结果与数据中台对接,从而获取订单模块的大数据分析结果。

2.3金融机构(如银行)

微服务架构已经成为金融机构如银行中一种举足轻重的系

统设计模式。将传统的单体应用程序拆分成多个独立的微服务模块,使银行能够更加灵活高效地管理和经营它的业务。作为微服务模块之一的账户管理微服务负责处理用户账户的各种创建更新查询和关闭等操作,并能够独立于其他系统进行开发和部署,保证账户资料的安全与一致性。同时,账户管理微服务还能方便地进行功能扩展,例如增加新的账户类型或集成新的验证方式,从而在系统灵活性和可扩展性方面得到提升。

交易处理微服务是专门为银行交易事务处理而设计的,如存款/取款/转账等操作。基于微服务架构,交易处理的高并发/高性能特性得到了保证,同时也保证了交易的实时性/准确性,而基于微服务架构的解耦,交易处理与其他服务进行解耦,从而提升了整体系统的响应速度/可靠性。通过微服务架构,交易处理与其他相关服务进行解耦,如账户管理/风险控制等,使得系统整体更加灵活/可扩展。主要目的之一是以微服务为基础,为银行提供高效且富有个性化的客户服务[2]。

微服务可包含客户查询投诉处理以及反馈收集等功能,通过微服务架构的银行能够快速响应客户需求,提供更好地服务体验。另外,客户支持微服务还能整合客户关系管理系统以及聊天机器人等其他服务,在进一步提高客户服务效率的同时提高服务智能化水平。通过整合微服务,银行可实现对客户需求的快速响应,提供更具个性化服务体验。采用微服务架构能使银行在客户响应速度上得到很大地提升。每个微服务模块都能进行独立的部署和扩展,这样就避免了由于单个模块出现问题而导致的整个系统响应缓慢的情况发生。微服务架构还能使功能扩展变得更为容易和快捷,由于新功能能作为独立的微服务进行开发和部署,而无需对现有系统进行大范围的改动,因此在银行系统中进行功能扩展会变得更加容易一些。

2.4医疗保健行业

医疗机构通常使用各种不同的医疗设备和系统,如电子健康记录系统,实验室信息管理系统及正射影像系统等,产生大量的资料数据要相互融合与共享,微服务架构通过将数据整合功能拆分成独立的微服务模块,使不同系统间数据无缝对接与共享,提高了数据的可用性和一致性。通过微服务架构,数据整合功能得以独立开发与维护。以微服务架构为基础的每个资料集成服务都能独立进行开发试验部署工作,简化资料集成服务的开发维护过程,使之在应对变化的需求上具有更大的灵活度与应变能力,举例说明就是当引入新的医疗设备的时候能够以对应的微服务为基础进行资料的集成开发工作而不需要进行大范围的系统变动。另外微服务架构也具有更大的弹性和应变能力所以能够以较快速度适应新的业务需求与技术变化。因此基于微服务架构的医疗机构,能够根据不断变化的业务需求与技术条件而进行快速调整与适应。

3 微服务架构的挑战

3.1复杂性管理

在Microservice架构中,系统被拆分为若干个独立的服务,每个独立的服务负责具体的业务功能。尽管这种拆分使得系统的模块化和灵活性得到了提升,但同时也带来了管理上的复杂性。所有的功能都在传统单一架构中的代码库中,相对来说开发管理起来也比较简单。并且在Microservice架构中,开发团队需要管理大量的代码库,配置文件,部署脚本。每一项服务都需要与其他多项服务进行交互,这就要求复杂的API接口和数据交换格式的设计和实现。服务之间的依赖关系也会增加系统整体的复杂性,从而使得追踪和管理变得更加困难。

服务发现和监控在一个包含众多微服务的系统中显得格外重要。ServiceDiscovery是指如何让一个服务找到并调用另一个服务。这一般需要借助服务注册中心,比如Netflix的欧瑞卡或者Consul。但是,维护服务注册中心的稳定性和可靠性本身就是个挑战,特别是在服务上线、下线频繁的情况下,更是如此[3]。

在监控上,微服务架构的需求很难被传统的监控工具所适应。包括其性能指标、健康状况以及日志信息在内的每一项微服务都需要单独监测。为了对问题进行有效地管理和排查,必须引入分布式追踪系统,如Zipkin或Jaeger,对跨多个服务的请求链接进行追踪。虽然这些工具功能强大,但其复杂的配置和维护对运行维护团队的要求也更高。此外,数据同步同样面临挑战。

3.2数据一致性问题

微服务架构中,每个服务通常都有自己的数据库,这种设计虽然提高了数据的自治性和系统的可扩展性,但也带来了数据一致性的问题。在单体架构中,数据库事务可以确保数据的一致性,而在微服务架构中,分布式数据库使得事务管理变得更加复杂。

为了解决分布式数据一致性问题,通常使用两阶段提交(2PC)或补偿事务(Sagas)等分布式事务模式。两阶段提交可以确保分布式系统中的所有节点都达成一致,但它的实现复杂且性能开销较大。补偿事务则通过一系列可回滚的子事务来确保最终一致性,尽管它更为灵活,但设计和维护也相对复杂。

如何保证数据的实时同步和一致性,在需要多个微服务共享数据的情况下是个难题。比如,如何保证在用户信息更新的时候,更新的数据能够及时被所有需要这个信息的服务所获取?这种情况需要设计有效的数据同步机制,比如通过事件驱动架构和消息队列来实现数据的传播和同步。

3.3运维和部署

每一个微服务都是独立部署的,也就是说需要开发团队频繁地运营部署。为了保证系统的稳定性和可靠性,必须实现自动化的部署流程,这通常需要使用Jenkins、GitLabCI、CircleCI等持续集成和持续部署(CI/CD)工具。这些工具可以帮助团队对流程进行自动化的测试、构建和部署,但同样需要花费大量的时间和精力来对其进行配置和维护。

不同的服务可能需要在不同的时间发布不同的版本,如何确保版本之间的兼容性,这是一个需要解决的问题,如果在不同的时间发布不同的版本通常需要在API设计时引入语义版本控制、考虑向后兼容等版本控制策略。对于更大规模的微服务系

统来说,为了减少新版本上线对系统的冲击,蓝绿部署、金丝雀发布等策略也需要设计和实现。在微服务架构中,每一项服务都需要独立部署、独立运营,解决这一问题的首选是容器化技术。

每个微服务都可以通过Docker和其他容器技术打包成一个独立的容器,使部署和运行环境的配置变得更加简化。但是容器的管理也带来了新的挑战,特别是如何有效地对容器进行管理和编排是大规模容器应用中需要解决的问题。尽管Kubernetes这样的容器编排工具提供了强大的功能,但其复杂程度也要求运维团队更高。持续集成/持续部署,简称CI/CD,在微服务架构中必不可少,这是为了提高开发和部署效率减少人为错误而搭建的完善的自动化流程。团队需要对CI/CD流水线进行合理设计,在代码的单元测试用例执行完毕之后,自动进行构建和部署环节。既提高开发效率,又降低人为失误。但在CI/CD工具的配置和使用上同样复杂,需要对DevOps有比较深入地了解和经验。所以,为了更好地发挥CI/CD的功能,提高微服务架构的开发和部署质量,需要对相关技术和工具进行持续地学习和实践。

4 结论

凭借其鲜明优点,微服务架构日益普及并成为现代软件开发的主旋律。预计随着科技水准持续提升及企业应用涉猎领域更广,微服务架构将展现出更强韧的适应性,进一步凸显其重要性。经过反复尝试与优化,微服务架构必将在现代软件开发中扮演更为关键的角色,为企业创造更多价值与可能性。

[参考文献]

[1]李亮,舒畅.微服务架构与容器化技术的软件开发实践[J].物联网技术,2024,14(05):64-67.

[2]金勇.微服务架构在软件开发领域的应用与前景分析[J].市场周刊·理论版,2020,(33):6-7.

[3]张炼.基于微服务架构的企业级软件开发实践探讨[J].中国信息界,2024,(02):48-50.

作者简介：

曹严清(1988--),男,汉族,安徽明光人,硕士,高级软件工程师,计算机技术。