文章编号:2096-1472(2024)01-0006-04
DOI:10.19644/j.cnki.issn2096-1472.2024.001.002

云原生应用开发与部署面临的挑战及其应对方案

李 亮

(武昌工学院信息工程学院,湖北 武汉 430065)
✉ 543666826@qq.com

摘 要:随着云计算的发展和普及,云原生应用作为一种新的应用开发和部署方式备受关注,以其高度的可扩展性、可移植性和弹性成为现代云环境下的首选开发模式。文章首先分析了微服务架构管理的复杂性、持续集成与持续部署(CI/CD)的自动化难题及跨多云和混合云环境下存在的兼容性问题等带来的挑战,并提出应对方案;其次采用 Kubernetes 进行统一的微服务管理,利用开源工具实现 CI/CD 自动化流程,以及设计跨云应用的统一部署策略;最后分析和总结云原生应用的发展趋势,为软件工程在应用开发与持续部署领域提供了有益的参考和启示。

关键词:云原生应用;微服务;持续集成与持续部署(CI/CD);Kubernetes;跨云部署
中图分类号:TP393.2 **文献标志码**:A

Challenges and Solutions for Cloud Native Application Development and Deployment

LI Liang

(*Information Engineering Institute, Wuchang Institute of Technology, Wuhan* 430065, *China*)
✉ 543666826@qq.com

Abstract: With the development and popularization of cloud computing, cloud native applications have attracted much attention as a new way of application development and deployment. With their high scalability, portability, and resilience, they have become the preferred development mode in modern cloud environments. This paper first analyzes the management complexity of microservices architecture, the automation challenges of continuous integration and continuous deployment (CI/CD), and challenges brought by compatibility issues under multi-cloud and hybrid-cloud environments; then the solutions are proposed. Secondly, Kubernetes are used for unified microservice management, open-source tools are utilized to automate CI/CD processes, and a unified deployment strategy is designed for cross cloud applications. Finally, the analysis and summary of the development trends of cloud native applications provide useful reference and inspiration for the field of software engineering.

Key words: cloud native applications; microservice; continuous integration and continuous deployment (CI/CD); Kubernetes; cross cloud deployment

0 引言(Introduction)

随着云计算技术的蓬勃发展,一种新的应用开发与部署模式——云原生应用逐渐成为研究热点,受到业界的广泛关注[1]。云原生应用具备的弹性、可移植性及高度的可扩展性等特点使其成为现代云环境下的首选开发模式,在现代软件工程领域中占据了不可替代的地位,但云原生应用在兴起的同时也面临一系列开发与部署方面的挑战,例如微服务架构的管理具有一定的复杂性、持续集成与持续部署(CI/CD)的自动化难度大及跨多云和混合云环境的兼容性问题还未解决等,成为阻碍其进一步普及应用的关键因素。目前,很少有综述性的文献研

收稿日期:2023-05-10

究云原生应用开发与部署面临的挑战及应对方案,如何解决这些问题并将这种开发模式全面推广到社会应用化大生产中,以释放云原生应用的真正潜力,成为学者和工程师们亟待探讨的问题。基于此,本文主要围绕云原生应用开发与部署过程中面临的挑战进行了深入的分析,并提出了应对策略,希望能为相关研究者和技术应用工程师提供有益的参考。

1　云原生应用的核心特性及其对开发与部署的影响(The core features of cloud native applications and their impact on development and deployment)

云原生应用是近年来随着云计算技术逐渐成熟而兴起的一个概念,代表一种新的应用开发和部署方式。在云原生应用这种模式下,应用被设计成在云环境中运行,充分利用了云的弹性、可扩展性和可分布式处理等特性。

(1)云原生应用的核心特性之一是它的微服务架构。与传统的单体应用不同,微服务架构将应用分解为一组小型、自治、可独立部署的服务,这种分解使得应用的开发、部署和维护更为简便,同时支持更为灵活的扩展。但与此同时,云原生应用的广泛应用也带来了服务间通信、数据一致性和事务管理等多方面的一系列更加复杂的问题。

(2)容器化是云原生应用的另一个核心特性。容器提供了一个隔离的环境,使应用可以在其中运行而不受外部环境的干扰。Docker 和 Kubernetes 等技术的兴起,使得容器的管理、编排和部署变得更加简单[2]。容器化使应用的部署和迁移更快捷,可以轻松地从一个环境迁移到另一个环境,如从开发环境迁移到生产环境。

(3)云原生应用通常设计为无状态或将状态分离。这意味着应用的逻辑和数据存储是分离的,可以让应用更容易地扩展和迁移,不必担心数据的同步问题,但也面临数据管理和持久化的挑战[3]。

表 1 总结了云原生应用的核心特性及其在开发和部署方面的优势和面临的挑战。

表1　云原生应用的核心特性、优势及面临的挑战

Tab.1 Core characteristics of cloud native applications

核心特性	优势	挑战
微服务架构	可以灵活地扩展、开发且维护简便	在服务间通信、数据一致性和事务管理等多方面面临挑战
容器化	在部署简便、高度的可移植性	在容器管理和资源隔离方面面临挑战
无状态设计	扩展和迁移简便	在数据管理和持久化方面面临挑战

(4)云原生应用的开发和部署需要考虑多云和混合云,这意味着应用可能需要在不同的云服务提供商之间迁移,或者在私有云和公有云之间迁移。虽然这为应用提供了更大的灵活性,但是也带来了兼容、数据迁移和网络连接等方面的问题。

综上所述,云原生应用的核心特性为其在现代软件开发中提供了巨大的优势,但也带来了开发和部署上的一系列挑战[4]。

2　微服务架构在云原生应用中的应用与管理挑战(Application and management challenges of microservice architecture in cloud native applications)

微服务架构作为云原生应用中的一个核心组成部分,已经逐渐成为现代软件开发的标准,它可以将大型复杂的应用拆分为多个独立、松散耦合的服务,每个服务都执行特定的业务功能,并可以独立开发、部署和扩展。这种模式为企业的微服务架构带来了更快的迭代、更好的可扩展性和更高的容错性。与此同时,微服务架构也给开发和运维团队带来了一系列新的挑战[5]。

在微服务架构中,服务间的通信变得尤为关键。云原生应用中的通信经常采用轻量级的协议,如 REST 或 gRPC。但随着服务数量的增加,跟踪和管理这些服务间的通信变得越来越困难。例如,如何保证服务间数据的一致性、如何处理网络延迟或失败、如何保证通信的安全性等,都是开发者必须解决的问题[2]。

此外,服务发现和负载均衡也是微服务架构面临的重要挑战。在传统的单体应用中,所有组件都在同一个进程中运行,而在微服务架构中,每个服务可能有多个实例分布在不同的主机或容器中。这就要求有一种机制确保请求被正确地路由到可用的服务实例,并在服务实例之间进行负载均衡。

如表 2 所示列出了微服务架构在云原生应用中面临的几个关键挑战及其潜在影响。

表2　微服务架构在云原生应用中的挑战及影响

Tab.2 Challenges and impacts of microservice architecture in cloud native applications

挑战	潜在影响
服务间通信	网络延迟、数据不一致、通信安全性等问题
服务发现和负载均衡	请求路由错误、单点故障、性能瓶颈等问题
服务的部署和版本管理	版本冲突、部署失败、服务降级等问题
分布式事务管理	事务不一致、数据丢失、复杂的回滚策略等问题

为了应对表 2 中的挑战,经常采用一些先进的工具和方法。例如,使用 Kubernetes 解决服务发现和负载均衡问题,Kubernetes 内置了服务发现和负载均衡的机制。服务网格技术 Istio 和 Linkerd 可以跟踪服务间的通信,为微服务提供了统一的通信层[6]。

尽管以上工具和方法的应用可以提高云原生应用的技术性能,但对其微服务架构的管理仍然是一项复杂的任务,例如如何设计服务的粒度、如何确保服务的独立性和自治性、如何处理跨服务的数据和事务一致性等,要解决微服务架构在云原生应用中的关键问题,需要开发者和运维团队具备扎实的基础知识和丰富的工作经验。

3　持续集成与持续部署(CI/CD)在云原生环境中的实践与难点(The practice and difficulties of continuous integration and continuous deployment (CI/CD) in cloud native environment)

持续集成与持续部署(Continuous Integration/Continuous Delivery,CI/CD)是现代软件开发中的关键环节,它为开发团

队提供了一种快速、可靠的方式集成和部署代码。在云原生的环境中，CI/CD更为关键，因为云原生应用经常需要在多个环境中部署、扩展和运行。然而在实践中，CI/CD在云原生环境中面临以下难题[7]。

(1)云原生应用的微服务架构需要部署大量的服务和组件，要求CI/CD流程能够支持多服务的部署，同时确保服务间的依赖关系得到正确的管理。例如，当一个服务的新版本部署后，它可能需要与其他服务进行通信，这就要求这些服务能够与新版本的Web系统更新兼容。

(2)云原生环境的动态和弹性特性意味着CI/CD流程需要处理动态的资源分配、服务发现和负载均衡。例如，当新的服务实例启动或关闭时，CI/CD工具需要自动将它们添加到服务器或从负载均衡器中移除。

(3)云原生应用常常部署在多云或混合云的环境中，这就要求CI/CD工具能够支持多种云服务提供商的API，例如AWS、Microsoft Azure和Google Cloud Platform的服务、API和部署模型都不相同，但是CI/CD流程需要适应这些差异。

表3列出了CI/CD应用于云原生环境中的主要问题及其可能造成的影响。

表3 CI/CD应用于云原生环境中的主要问题及其可能造成的影响

Tab.3 The main issues and potential impacts of applying CI/CD in cloud native environments

问题	可能造成的影响
多服务的部署	服务间的依赖关系管理、版本兼容性
动态资源管理	动态的服务发现、负载均衡和网络配置
多云部署	与多个云服务提供商的API和特性兼容
安全性	密钥管理、网络安全和服务间的通信安全

为了解决这些问题，许多CI/CD工具和平台已经开始为云原生环境提供特定的特性支持和插件。例如，Jenkins、Travis CI和CircleCI都提供了对容器和Kubernetes的原生支持，使开发者能够轻松地构建、测试和部署云原生应用。

开发和运维团队需要密切合作，确保CI/CD流程与云原生应用的特性和需求相匹配。此外，开发团队需要不断地学习和掌握新的技术和方法，以满足不断变化的业务和技术需求[8]。

4 跨多云和混合云环境下的应用部署策略与面临的挑战(Application of deployment strategies and challenges across multi-cloud and hybrid-cloud environments)

跨多云和混合云环境是企业应用部署的发展趋势，它们提供了强大的灵活性、冗余性和自由度。多云策略允许组织跨多个公有云服务提供者部署应用，而混合云策略结合了公有云和私有云(或传统数据中心)的优势，但在实施这些策略的过程中，开发和运维团队也面临一系列新的挑战[9]。

4.1 跨多云和混合云环境中组件与架构设计

模仿SSM(Spring、SpringMVC、MyBatis的缩写)是目前JavaEE企业级开发中最受欢迎的框架、组件的有序集合，拥有功能完善、轻量级的云服务实现组件，例如在服务发现治理、服务容错、服务网关、服务配置、负载均衡、消息总线服务跟踪等方面均有经过实践检验的成熟组件[10]。基于Spring、SpringMVC和MyBatis三个框架分工的SSM架构整合工作原理如图1所示。

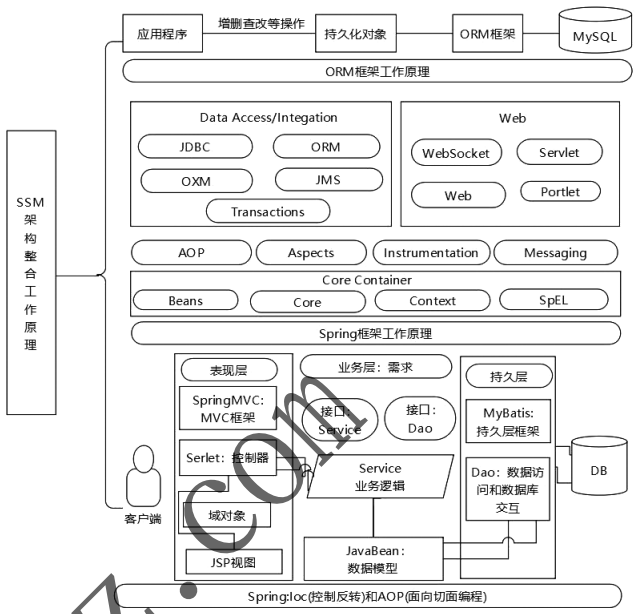

图1 SSM架构整合工作原理

Fig.1 Working principles of SSM architecture integration

(1)SpringMVC负责管理表现层的Handler。SpringMVC容器是Spring容器的子容器，因此SpringMVC容器可以调用Spring容器中的Service对象。

(2)Spring负责事务管理，Spring可以管理持久层的Mapper对象和业务层的Service对象。由于Mapper对象和Service对象都在Spring容器中，所以可以在业务逻辑层通过Service对象调用持久层的Mapper对象。

(3)MyBatis负责与数据库进行交互。

整合SSM框架后，当SpringMVC接收到请求，可以通过Service对象执行对应的业务逻辑代码，再由Service层装载Mapper对象，最终由Mapper对象完成数据交互。

在SSM框架整合过程中，SpringMVC和MyBatis没有直接的交集，所以只需要将Spring分别与SpringMVC和MyBatis整合，就可以完成SSM框架的整合设计。基于SSM架构案例设计实现思路如下：①搭建项目基础结构。首先在数据库中搭建项目对应的数据库环境；其次创建一个Maven Web项目，并引入案例所需的依赖；最后创建项目的实体类，创建三层架构对应的模块、类和接口。②整合Spring和MyBatis。在Spring配置文件中配置数据源信息，并且将SqlSessionFactory对象和Mapper对象都交由Spring管理。③整合Spring和Spring MVC。Spring MVC是Spring框架中的一个模块，Spring整合Spring MVC只需在项目启动时分别加载各自的配置即可。

案例编写完成之后,客户端向云服务器端发送请求,如果云服务器端能将数据库中的数据正确响应给客户端,就可以认为 SSM 框架整合成功。

4.2 跨多云和混合云环境中应用开发部署架构

每个云服务提供者都有其独特的 API、服务和特性,这意味着当应用需要在多个云环境中部署时,需要针对这些差异进行代码或配置的调整[11]。例如,云计算在 AWS 的 S3 和 Azure 的 Blob Storage 中的功能相似,但它们的 API 和使用方式可能不同,AWS 与 Microsoft Azure 两个计算行业巨头的云原生服务部署架构如图 2 所示。

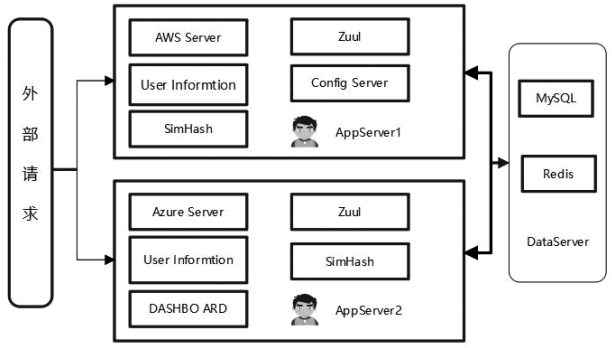

图 2　AWS 与 Microsoft Azure 两个计算行业巨头的云原生服务部署架构

Fig. 2 Cloud native service deployment architecture of AWS and Microsoft Azure, two giants in computing industry

4.3 跨多云和混合云部署中的关键因素

网络和数据传输是跨多云和混合云部署中的一个关键因素,跨多云和混合云部署中面临的主要挑战是在不同的云环境中确保数据的一致性和数据的高效传输。

安全性和合规性也是跨多云和混合云部署要重要考虑的问题。不同云提供商可能有不同的安全标准和工具,而跨云部署意味着组织需要确保所有的环境都符合统一的安全和合规性要求。同时,应用分布管理和监控也变得更加复杂。当应用分布在多个云环境中时,如何有效地跟踪资源使用、性能指标和故障情况,是组织需要解决的问题。

使用容器和 Kubernetes 可以简化跨多个云环境的应用部署。容器提供了一种标准化的方式打包和运行应用,而 Kubernetes 则为跨云部署提供了一套统一的管理和编排工具。此外,服务网格技术(如 Istio)提供了一种跨多云环境的通信和管理解决方案[12]。

4.4 跨多云和混合云环境中为企业提供管理平台

当前,多数企业选择使用多云管理平台,这些平台提供了跨多个云环境的统一管理、监控和自动化工具,这些工具帮助组织简化操作、减少错误和提高效率,特别是多云和混合云模式为企业管理平台提供了更大的灵活性和选择性,但也提高了跨多云和混合云环境中管理的复杂性。为解决管理的高复杂性问题,企业需要制定明确的策略、使用适当的工具和方法,以及注重数据安全性和成本控制,还需要有效地管理多个云平台上的应用,实现更高效的云计算环境。在不断发展的云计算领域,有效的多云和混合云策略将成为企业取得成功的关键[13]。

5 结论(Conclusion)

跨多云和混合云部署为现代企业带来了前所未有的机会,但伴随而来的技术和策略方面的挑战也不容忽视。云原生应用是由微服务、容器、DevOps、服务网络、不可变基础、声明式 API 等关键技术构建,面对多样化的云环境,网络性能及数据的一致性和安全性成为决策中需要考虑的核心因素。选择合适的工具和平台,实施统一的配置管理策略,以及加强对网络与安全的重视,是确保应用成功部署的关键。正如本文探讨,前瞻性的策略和持续的技术适应是企业在多云与混合云时代中得以蓬勃发展的基石。

参考文献(References)

[1] 赵阮彬. 基于混合云架构的广播电台融媒体技术平台[J]. 信息记录材料,2022,23(10):201-203.

[2] 唐安宏,刘建兴. 江苏有线混合云部署架构设计[J]. 广播电视信息,2020(4):97-99.

[3] 曾理,胡晓勤,龚勋. 面向多云环境的虚拟私有云安全通信研究[J]. 现代计算机,2021(17):19-24.

[4] 彭兴,徐辰宇,陈俊,等. 可与 iFIX 一体化集成的 EMS 高级应用软件系统的设计与应用[J]. 工业控制计算机,2014,27(8):1-2.

[5] 刘晓庆. 基于 EUHT 技术的城轨高速线路车地无线网络解决方案[J]. 铁路技术创新,2018(2):30-34.

[6] 张国生. 基于 Kubernetes 的无服务器计算与微服务集成架构研究[J]. 中国电子科学研究院学报,2023,18(1):48-55.

[7] 詹翊春. 电信级以太网技术讲座 第 4 讲 电信级以太网的应用与部署[J]. 中国新通信,2008,10(7):68-72.

[8] 于尔铿,刘广一,周京阳,等. 能量管理系统(EMS)中高级应用软件的设计与开发[J]. 电网技术,1995,19(5):10-14.

[9] 翁列恩,王振,楼佳宁. 集成化、信息化与标准化的居家服务创新研究:以杭州市上城区为例[J]. 公共管理学报,2013,10(3):1-10,137.

[10] 王方旭. 基于 Spring Cloud 实现业务系统微服务化的设计与实现[J]. 电子技术与软件工程,2018(8):60-61.

[11] FERNANDEZ ALVAREZ L, DATSKOVA O, JONES B, et al. Managing the CERN batch system with Kubernetes [J]. EPJ web of conferences, 2020, 245: 07048.

[12] TRUYEN E, XIE H J, JOOSEN W. Vendor-agnostic reconfiguration of Kubernetes clusters in cloud federations[J]. Future internet, 2023, 15(2): 63.

[13] 何震苇,黄丹池,严丽云,等. 基于 Kubernetes 的融合云原生基础设施方案与关键技术[J]. 电信科学,2020,36(12):77-88.

作者简介:

李　亮(1985-),男,硕士,讲师。研究领域:软件工程,服务器技术。