DOI：10.16667/j.issn.2095-1302.2024.05.017

微服务架构与容器化技术的软件开发实践

李　亮，舒　畅
（武昌工学院 信息工程学院，湖北 武汉 430065）

摘　要：随着云计算、大数据和高并发应用的兴起，传统的单体应用架构逐渐暴露出其弊端。微服务架构及容器化技术在此背景下应运而生，为企业提供了更加灵活、可扩展和高效的软件开发和部署方式。论文重点探讨了微服务架构的基本概念、特点和优势，同时深入分析了容器化技术如 Docker 和 Kubernetes 在微服务部署中的重要角色。首先，文章进一步研究了微服务与容器化技术的结合如何帮助企业更好地满足变化快速的业务需求；其次，文章探讨了如何提高系统的可维护性和降低开发及运维的复杂性；最后，文章提出了几种最佳实践，以指导开发者和企业如何更加高效地采纳和实施这两种技术。

关键词：微服务；容器化；Docker；Kubernetes；软件开发实践；虚拟化
中图分类号：TP319　　**文献标识码**：A　　**文章编号**：2095-1302（2024）05-0064-04

0　引　言

随着技术的日新月异，传统的软件开发模式已经无法满足当下的业务需求与市场挑战。微服务架构和容器化技术的出现，为此提供了革命性的解决方案。而这两者之间的深度结合，无疑打开了一扇通向更加高效、灵活和可扩展的软件开发与部署的大门。那么，如何确保我们能够完全利用这两者的潜力，并实现其在现实业务中的最大价值？本文旨在深入探索这一问题，分享先进的技术视角与实践经验，引领读者进一步理解这一趋势背后的核心技术和应用价值。

1　微服务架构的基本概念与特点

随着企业技术生态系统的快速发展，软件系统的复杂性日益增加，问题更加突出。在这样的背景下，微服务架构逐渐成为当今软件开发领域的研究热点和实践标准。

微服务架构是一种方法论，它是将一个大型的、单一的应用分解为一组小的、功能独立的服务。每一个这样的服务都运行在其自己的进程中，并与其他服务通过通常是轻量级的机制（如 HTTP RESTful API、gRPC 等）通信。这些服务是围绕业务功能构建的，可以通过全自动部署机制独立地部署和扩展[1]。

微服务的核心优势在于其解耦的特性，使得各个服务可以独立开发、测试、部署和扩展。此外，由于每个服务都较小，理解、维护和修改都变得更加容易。微服务架构的关键特点如下：

（1）模块性：微服务使应用程序更加模块化，这使得管理和扩展应用程序更容易。

（2）独立部署：由于每个微服务都是独立的，所以可以独立部署，这大大减少了部署风险。

（3）技术多样性：不同的服务可以使用不同的技术栈，允许团队选择最适合其特定服务的技术。

（4）可伸缩性：可以根据需要为特定的服务分配更多或更少的资源，而不必为整个应用程序扩展资源。

（5）容错性：单个服务的失败不太可能导致整个系统的故障。

为了更直观地对比微服务与传统单体架构，总结了两者的主要差异，见表 1 所列。

表 1　微服务与传统单体架构的差异

特　点	单体应用	微服务
部署	整体部署	独立部署
开发语言	通常单一	可多样化
数据存储	通常单一数据库	每个服务可能有自己的数据库
扩展性	水平扩展整体应用	单独扩展每个服务
容错性	一个模块的失败可能导致整体失败	模块之间隔离，单一模块故障不影响整体

尽管微服务架构具有诸多优势，但它也带来了一些挑战，例如服务之间的通信、数据的一致性、服务的发现和负载均衡等。

2　容器化技术概览

随着软件开发领域的发展，对高效、可扩展、易于维护的解决方案的需求不断增加，容器化技术逐渐成为前沿的技术选择。其中，Docker 和 Kubernetes 作为领先的技术，已在全球范围内被广泛采用，Docker+Kubernetes 已成为云计算的

收稿日期：2023-04-17　　**修回日期**：2023-05-15

主流[2]。Docker 与 Kubernetes 的关系与区别如图 1 所示。

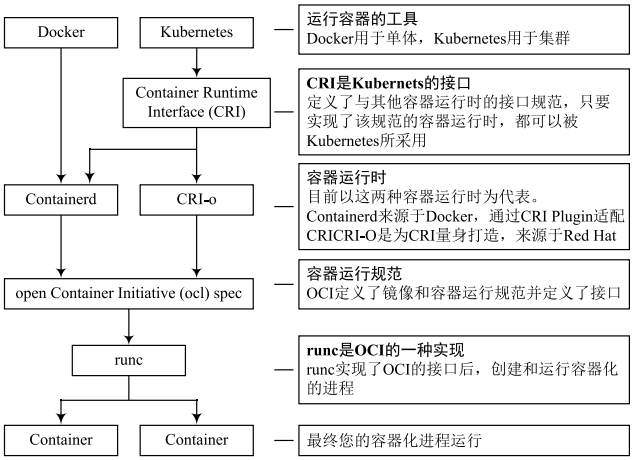

图 1　Docker 与 Kubernetes 的关系与区别

2.1　Docker——轻量级的容器化工具

Docker 是一个开源项目，旨在为开发、移植和运行应用程序提供一个轻量级的容器化解决方案。Docker 容器可以看作是轻量级的虚拟机，但与传统的虚拟机相比，它更为高效，因为它直接在宿主机的操作系统上运行，而不需要模拟整个操作系统。

Docker 的主要优势如下：

（1）可移植性：Docker 容器内的应用程序和其依赖项可以在任何支持 Docker 的环境中运行，无论是云端、物理服务器还是开发者的本地机器。Amazon ECS（Amazon Elastic Container Service）是一种高度可扩展的高性能容器编排服务，支持 Docker 容器，可以在 AWS 上轻松运行和扩展容器化应用程序，而不需要安装和操作自己的容器编排软件，不需要管理和扩展虚拟机集群，也不需要在这些虚拟机上调度容器[3]。其工作原理如图 2 所示。

（2）持续集成 / 持续部署（CI/CD）：Docker 与现代 CI/CD 工具（如 Jenkins、GitLab CI 等）无缝集成，使得应用程序的构建、测试和部署过程更为流畅。

（3）隔离性：Docker 的核心思想是利用扩展的 LXC（Linux Container）方案实现一种轻量级的虚拟化解决方案。Docker 主要利用 Kernel Namespace 来实现容器的虚拟化隔离性，保证每个虚拟机中服务的运行环境隔离。Docker 的隔离机制降低了内存开销，保证了虚拟化实例密度[4]。每个容器都在自己的环境中运行，确保了应用之间的隔离。

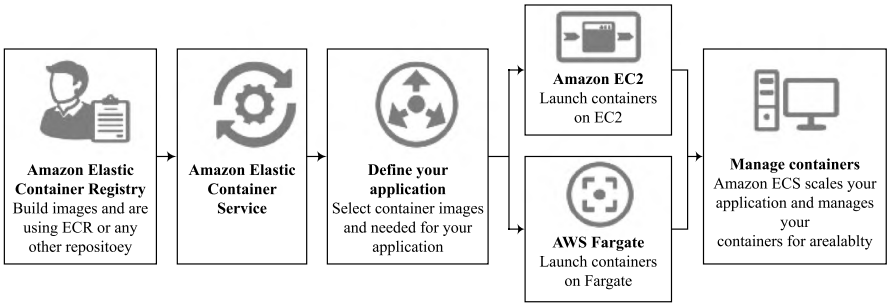

图 2　虚拟机上调度容器工作原理

2.2　Kubernetes——容器编排平台

随着容器化应用的日益普及，需要一个系统来管理、调度和扩展这些容器。Kubernetes 应运而生，作为一个开源的容器编排平台，它不仅能管理容器的生命周期，还能提供自动化伸缩和管理容器化应用的功能[5]。

Kubernetes 的关键价值如下：

（1）自动化部署和回滚：当部署新版本的应用程序时，Kubernetes 可以确保部署不会导致系统停机，并能自动回滚到之前的版本。

（2）负载均衡和服务发现：Kubernetes 可以自动分配 IP 地址和选择一个方法来分发网络流量到不同的容器。

（3）自动存储编排：自动挂载所选的存储系统，无论是本地存储、云供应商，或是其他系统。

（4）自动扩展：基于 CPU 利用率或其他选择的指标，Kubernetes 可以自动调整应用程序的副本数量。

Docker 和 Kubernetes 的主要功能对比见表 2 所列。

表 2　Docker 和 Kubernetes 的主要功能对比

功　能	Docker	Kubernetes
用途	容器化应用程序	容器编排和管理
网络	Docker 网络	服务、Ingress 控制器、网络策略
存储	Volumes、bind mounts	Persistent Volumes、Storage Classes
扩展性	手动或 Docker Compose	自动扩展，基于配置的策略
部署	Docker ComposeDeployments，StatefulSets，DaemonSets	Docker 和 Kubernetes

Docker 与 Kubernetes 结合能够提供一个完整的框架

用于构建、部署、管理和扩展容器化应用。Docker 提供容器化的功能，Kubernetes 为这些容器提供必要的管理和编排工具 [6]。这两种技术的结合能够为开发者和运维团队提供强大的工具，使他们能够更快、更可靠地部署和管理应用程序，同时确保应用程序在生产环境中的稳定性和高可用性。

3 微服务与容器化技术的深度结合

近年来，微服务架构和容器化技术的应用已成为软件开发领域的重要趋势。其中微服务架构如图 3 所示。

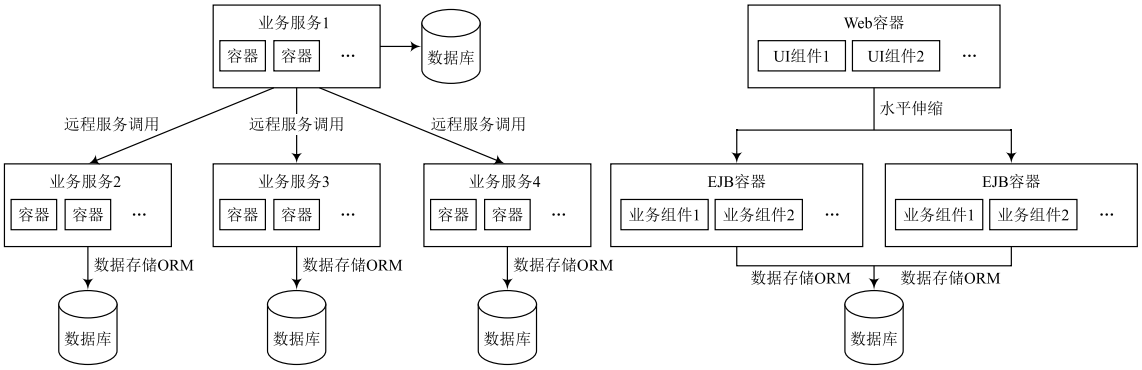

图 3 微服务架构

单独来看，微服务架构和容器化技术各自都具有很强的创新性。二者结合时，在现代软件交付中则更能显示出强大的优势 [7]。

（1）敏捷性与扩展性：微服务架构的分散性以及容器化技术的隔离和轻量性使得应用程序能够被更快地部署和扩展。每个微服务都可以独立地打包、部署和扩展，这大大提高了整体系统的敏捷性。

（2）跨平台和云原生：容器化技术，特别是 Docker，确保了应用程序在不同环境中的一致性和可移植性。这意味着微服务可以在任何平台包括云平台上运行，为企业提供真正的多云策略。

（3）增强的持续集成 / 持续部署（CI/CD）：微服务和容器的结合为自动化的构建、测试和部署流程提供了更好的基础，这促进了更快和更频繁的代码发布。

（4）资源利用率：与传统虚拟机相比，容器对资源的需求更少，这允许更高密度的部署，从而提高资源利用率。

微服务架构和容器化技术的结合带来的挑战如下：

（1）复杂性：微服务的分散性和容器的管理都可能增加系统的复杂性。这需要新的工具和方法来确保系统的稳定性和可维护性。

（2）服务发现与负载均衡：在微服务环境中，服务的数量可能会非常多。这需要有效的服务发现和负载均衡机制来确保服务的可用性。

（3）安全性：每个微服务都可能成为潜在的安全风险点。与此同时，容器本身也有其独特的安全挑战，如镜像安全和运行时安全。

（4）数据的一致性和管理：微服务通常有自己的数据存储，这可能会导致数据的一致性问题。同时，容器的短暂性存储也给数据管理带来了挑战。

微服务和容器化技术的结合为现代软件开发带来了巨大的机遇，也迎来了一些挑战。因此，在实践中需要在技术和策略层面进行更细致的考虑 [8]。

4 微服务和容器化技术的应用案例

近年来，随着企业数字化转型的加速，微服务和容器化技术已经成为支持这一变革的关键工具。许多企业不再满足于传统的单体应用，而是转向微服务架构来提高其应用程序的灵活性和扩展性。同时，容器化技术，尤其是 Docker 和 Kubernetes，能够为企业提供一个高度一致、可移植和可伸缩的运行环境。下面介绍一些微服务和容器化技术在真实业务场景中的应用案例 [9]。

电商平台巨头，例如亚马逊和阿里巴巴，都已经采用微服务和容器化技术来支持其庞大的在线业务。以亚马逊为例，为了支持 Prime Day 这样的大型购物活动，亚马逊需要一个能够快速扩展的系统来处理数百万的并发请求。通过采用微服务架构，亚马逊可以独立地扩展其各种服务，如订单处理、支付和库存管理。同时，容器化技术能够确保每个服务都在一个一致的和隔离的环境中运行，具备高可用性和高性能 [10]。

在金融领域，许多银行和金融机构通过微服务和容器化技术来支持其数字化服务。例如，某银行为了给客户提供更好的体验，决定对其在线银行应用程序进行重构。传统的单体应用已经无法满足客户对于响应速度和功能的要求。通过将应用程序拆分成多个微服务，如账户管理、交易处理和客

户支持，银行能够更快地迭代和发布新功能。容器化技术确保了这些服务可以在多个云平台和数据中心一致地运行。

医疗保健行业也看到了微服务和容器化技术的价值。某医院集团面临着来自不同医疗设备和系统的数据集成问题。传统的集成方法复杂且难以维护。通过采用微服务架构，医院可以为每种设备或系统建立独立的集成服务[11]。这不仅简化了集成过程，而且具有更高的灵活性。容器化技术进一步简化了服务的部署和管理，使医院能够更加集中精力提供高质量的医疗服务。

此外，制造业、零售业和许多其他行业也在积极探索微服务和容器化技术的应用[12]。无论是在供应链管理、客户关系管理还是智能制造方面，微服务和容器化技术都能够为企业提供一个更加敏捷、可靠和高效的解决方案。

5 结 语

随着企业数字化转型的推进，微服务和容器化技术已逐渐成为当前软件开发的核心。这两种技术的应用并不局限于特定行业或应用场景，从电商、金融到医疗和制造行业，都能为企业带来巨大的变革和价值。它们使企业数字化系统具有更高的灵活性、一致性和可伸缩性，确保现代企业可以迅速响应市场变化和客户需求。

参 考 文 献

[1] 林景朗 . 基于微服务架构的金融资讯服务平台的设计与实现 [D]. 上海：华东师范大学，2023.
[2] 董子奇，刘淇，高原，等 . 基于容器技术的微服务部署研究 [J]. 信息技术与标准化，2023，65（Z1）：93-98.
[3] 周姣 . 基于微服务架构的高校二手物品交易系统设计与实现 [J]. 电脑知识与技术，2023，19（22）：67-70.
[4] 夏丽娟，乌日娜，潘新 . Kubernetes 容器云编排系统的研究与实现 [J]. 自动化应用，2023，64（14）：224-227.
[5] 芮祥麟 . 敏捷方法：源于最佳实践，升级软件开发 [J]. 中国制造业信息化，2011，48（24）：54.
[6] 周至 . 最佳实践经验助力软件企业发展——微软“软件开发‘模式与实践’论坛”首次登陆中国 [J]. 程序员，2009，10（1）：28.
[7] 陈少锋，倪小凤 . 微信平台在校园服务领域的应用——以宿州学院“微校街”服务平台为例 [J]. 六盘水师范学院学报，2015，27（6）：58-61.
[8] 王昊旸 . 2015 年国际标准化组织低温容器技术委员会（ISO/TC 220）年会在北京召开 [J]. 中国特种设备安全，2015，31（7）：53.
[9] 张玲红 . 高职《ERP 软件开发实训》课程项目化教学设计 [J]. 电脑知识与技术，2014，10（30）：7144-7145.
[10] WEISSMAN B，NOCENTINO A E. A Kubernetes primer [M]. [S.l.]：[s.n.]：Apress eBooks，2021.
[11] BUCHANAN S，JOYNER J. Azure arc-enabled Kubernetes and servers [M]. [S.l.]：[s.n.]：Apress eBooks，2022.
[12] ALVAREZ L F，DATSKOVA O，JONES B，et al. Managing the CERN batch system with Kubernetes [J]. EPJ web of conferences，2020，245（7）：07048.

作者简介：李 亮（1985—），男，湖北孝感人，硕士，讲师，主要研究方向为软件工程、服务器技术。
舒 畅（1966—），男，安徽潜山人，教授，主要研究方向为计算机科学与技术。

（上接第 63 页）

[8] KHALIFA A，KERMORGANT O，OMINGUEZ S，et al. Platooning of car-like vehicles in urban environments：an observer-based approach considering actuator dynamics and time delays [J]. IEEE transactions on intelligent transportation systems，2021，22（9）：5684-5696.
[9] ZHENG Y，LI S E，WANG J，et al. Stability and scalability of homogeneous vehicular platoon：study on the influence of information flow topologies [J]. IEEE transactions on intelligent transportation systems，2015，17（1）：14-26.
[10] XU L，ZHUANG W，YIN G，et al. Energy-oriented cruising strategy design of vehicle platoon considering communication delay and disturbance [J]. Transportation research Part C：emerging technologies，2019，107：34-53.
[11] CHEN J，LIANG H，LI J，et al. Connected automated vehicle platoon control with input saturation and variable time headway strategy [J]. IEEE transactions on intelligent transportation systems，2021，22（8）：4929-4040.
[12] SHI M，YU Y，XU Q. Delaydependent consensus condition for a class of fractional-order linear multi-agent systems with input time-delay [J]. International journal of systems science，2019，50（4）：669-678.
[13] SADEK B A，EL HOUSSAINE ，NOREDDINE C. Small-gain theorem and finite-frequency analysis of TCP/AQM system with time varying delay [J]. IET control theory & applications，2019，13（13）：1971-1982.
[14] 李旭光，张颖伟，冯琳 . 时滞系统的完全稳定性究综述 [J]. 控制与决策，2018，33（7）：1153-1170.
[15] 郑洋 . 基于四元素构架的车辆队列动力学建模与分布控制 [D]. 北京：清华大学，2015.
[16] CHEHARDOLI H，HOMAEINEZHD M R，GHASEMI A. Control design and stability analysis of homogeneous traffic flow under time delay：a new spacing policy [J]. Proceedings of the institution of mechanical engineers，Part D：journal of automobile engineering，2019，233（3）：622-635.

作者简介：张琳虎（1998—），男，河南开封人，硕士，研究方向为自主车辆队列控制。
杨景凯（1998—），男，陕西宝鸡人，硕士，研究方向为智能交通、深度学习。
刘维宇（1987—），男，陕西西安人，博士，副教授，研究方向为电动微纳流体、人工智能。