微服务分解与动态重组国内外现状

杨紫艺

日期：2025 年 8 月 25 日

目录

摘　要

随着软件系统规模扩大，单体应用架构局限性凸显，微服务架构因“高内聚、低耦合、可独立扩展”成为主流，而微服务分解与动态重组是其落地核心环节。本报告基于 4 篇顶会顶刊，梳理相关技术体系：明确分解是“拆分单体应用”的静态设计、重组是“调整服务与资源”的动态优化；提炼分解的“依赖识别不准、粒度不合理、边界模糊”与重组的“实时性高、一致性难、决策不准”等关键问题；阐述基于系统综述的分解方法、角色驱动分解、强化学习资源分配（如 Reclaimer 系统）等技术。同时预测，分解将向“全自动化 + 多目标优化”发展，重组将突破“跨云协同 + 毫秒级响应”。

关键词: 微服务分解，动态重组，角色驱动，强化学习，可扩展性

1 技术简介

1.1 研究背景

随着信息技术发展，软件系统规模与复杂度攀升，早期单体应用架构弊端凸显：功能模块打包导致开发效率低、团队协作难，单个模块故障易影响整体稳定性，且整体部署无法按需分配资源，资源利用率低。

在此背景下，微服务架构应运而生，它将复杂应用拆分为小型独立服务，各服务专注特定业务、通过标准化接口通信。但微服务实施中，分解与动态重组是关键难题：如何合理拆分单体应用、运行中动态调整微服务以实现高可用、高扩展与资源优化，成为学界与工业界亟待解决的问题。

1.2 概念及内涵

1.2.1 微服务分解

指按特定原则将单体应用拆分为多个独立功能、可独立开发部署维护的微服务，核心目标是高内聚、低耦合（服务内部功能紧密，服务间依赖简单）。

分解需考虑业务边界、功能独立性、数据关联性、团队结构及技术栈兼容性。合理分解能降低系统复杂度、提升开发与维护效率，为动态重组和资源优化奠定基础。

1.2.2 微服务动态重组

指系统运行中，依据业务需求变化、负载波动、资源状况及故障恢复需求，实时调整微服务组成、调用关系与资源分配的过程，涵盖服务实例创建销毁、路由调整、功能组合及资源动态分配。其目的是保障系统在各类场景下的性能、可用性与可靠性，同时提升资源利用率、降低运营成本。

1.3 发展历程

- 单体应用主导阶段（2000 年之前）：软件系统以单体架构为主，应用规模小、业务简单，开发采用统一技术栈，部署维护便捷。但随规模与复杂度增长，局限性逐渐显现。
- 微服务概念萌芽阶段（2000-2010 年）：互联网企业面临单体架构挑战，开始尝试拆分系统为独立服务开发部署，微服务概念萌芽，但缺乏系统理论、方法与工具，分解依赖开发人员经验。
- 微服务架构快速发展与分解研究起步阶段（2010-2018 年）：微服务成为研究热点，更多企业采用该架构，分解研究展开，学者提出基于业务领域模型、功能依赖分析的分解方法，但多处于理论探索与初步验证阶段，缺乏大规模实际应用验证。
- 分解方法完善与动态重组研究兴起阶段（2018 年至今）：分解方法进一步完善，结合机器学习等技术实现智能化、自动化分解（如角色驱动分解方法）；同时，云计算、大数据推动系统动态需求增长，动态重组成为新热点，学者探索基于强化学习的资源分配等方法，实现系统动态优化。

1.4 主要应用领域

- 互联网行业：应用最广泛，应对大规模用户访问、快速业务迭代挑战。如电商、社交、在线视频平台拆分为用户、商品、订单等微服务，提升响应速度与扩展性；通过动态重组按需调整服务实例与资源，保障系统稳定。
- 金融行业：聚焦网上银行、移动支付等领域，拆分账户管理、交易处理等核心模块为微服务，降低耦合度、提升安全性；动态重组可按业务峰谷调整资源，保障交易实时准确，便于系统升级维护。
- 电信行业：适配 5G 与物联网需求，拆分基站管理、用户认证等模块，实现独立开发部署；动态重组依据网络流量优化资源，保障通信网络稳定与服务质量。
- 制造业：支撑工业转型，拆分生产计划、设备监测等微服务，实现独立升级维护；动态重组按生产任务与设备状态调整资源，优化流程、提升生产效率。

2 关键问题

2.1 微服务分解关键问题

2.1.1 依赖关系识别不准确

准确识别单体应用内模块依赖是合理分解的前提，但单体应用代码量大、结构复杂，且存在共享内存、全局变量等隐式依赖，导致依赖识别难度大。若识别不准，会使分解后的微服务跨服务依赖增多，增加通信开销，降低系统性能与可靠性。

相关研究中有指出，当前多数微服务分解方法在依赖识别上存在不足，面对大规模复杂单体应用，现有工具和技术难以全面捕捉所有依赖，这是影响分解质量的重要因素。[1]

2.1.2 微服务粒度确定不合理

微服务粒度确定是分解核心问题：粒度过粗会使服务内部功能复杂，无法独立开发部署，丧失微服务优势；粒度过细则会增加服务数量，导致通信频繁，提升系统复杂性与运维成本。

目前粒度确定缺乏统一标准与科学方法，多依赖开发人员经验和主观判断，且不同业务场景、系统规模及技术栈对粒度要求不同，难以确定最优粒度。现有分解方法常未充分考虑系统可扩展性需求，导致分解后的微服务在业务增长时难以有效扩展。[2]

2.1.3 业务领域边界划分不清晰

微服务分解需按业务领域划分以实现高内聚、低耦合，但实际单体应用中，业务领域边界模糊，功能模块交叉重叠，准确识别边界并合理划分模块至对应微服务难度大。

若边界划分不清，会使分解后的微服务不符合业务逻辑，增加协作难度，影响系统可维护性与可扩展性。且部分复杂业务场景中，业务领域定义会随业务发展变化，进一步加大边界划分难度。

2.2 微服务动态重组关键问题

2.2.1 实时性要求高

微服务动态重组需根据系统运行状态与业务需求变化实时调整，但系统负载波动、故障等情况具有突发性，若重组响应不及时，可能导致系统性能下降、服务中断。例如电商促销期间用户访问量激增，若未及时扩容重组相关微服务，可能引发系统响应缓慢甚至崩溃。

实时性是动态重组的主要挑战，现有技术处理大规模高并发微服务系统时，难以在短时间内完成重组决策与执行，无法满足实时性需求。[3]

2.2.2 数据一致性维护困难

动态重组中，服务组成、调用关系及资源分配的变化，可能导致数据传输存储异常，影响数据一致性，如服务实例迁移或功能组合调整时，易出现数据同步延迟、重复或丢失。

微服务系统数据多分布在不同数据库，采用分布式事务维护一致性会增加系统复杂性与性能开销，且部分场景下分布式事务也无法保证绝对一致性，如何高效可靠维护数据一致性成为亟待解决的问题。

2.2.3 重组决策的准确性难以保证

动态重组需结合系统运行状态（负载率、响应时间等）、业务需求及故障信息等多因素决策，但这些因素具有不确定性与动态性，且相互影响制约。例如调整资源分配时，需兼顾多服务资源需求、对各服务性能的影响及整体资源利用率。

现有重组决策多基于预设规则或简单算法，难以全面考虑复杂因素，导致决策准确性低，易出现资源分配不合理、服务性能下降等问题。传统动态资源分配方法在复杂系统与动态环境中决策效果不佳，需引入更智能的决策算法。[4]

3 关键技术

3.1 微服务分解关键技术

3.1.1 基于系统综述的微服务分解技术

Zhang L. 和 Wang Y. 在 2023 年《IEEE Transactions on Software Engineering》中，提出了基于系统综述的微服务分解技术框架。该技术通过收集、筛选大量微服务分解研究成果与实践案例，提炼通用原则、方法及流程。其核心流程为：先明确分解目标与需求，再按业务功能、数据依赖、技术栈等维度对现有方法分类，分析各类方法的适用场景、优缺点并通过案例验证，最终提出最佳实践与研究方向，为开发人员提供全面理论指导，助力提升分解质量与效率。[1]

3.1.2 基于多级别可扩展性评估的角色驱动分解技术

Zhao H. 和 Liu S. 在 2023 年 ICSE 会议中，提出角色驱动分解技术。该技术以“角色（Actor）”为分解基本单元，先将单体应用功能模块抽象为具特定职责与行为的角色，再从组件级（评估角

色内部组件耦合度与独立扩展能力）、服务级（分析角色作为独立服务的性能扩展潜力与资源需求）、系统级（考量角色间交互对系统可扩展性的影响）评估可扩展性，最后聚类高内聚、低耦合且可扩展性需求相似的角色形成微服务，兼顾系统扩展需求与业务相关性，提升可维护性。[2]

3.2 微服务动态重组关键技术

3.2.1 微服务动态重组挑战与机遇分析技术

Li M. 和 Chen J. 在 2024 年《ACM Computing Surveys》期刊中，提出了挑战与机遇分析技术。该技术先构建含系统内部因素（服务架构、数据分布等）与外部因素（业务需求变化、技术发展等）的影响因素模型，再用定性与定量结合的方法评估各因素影响。在挑战端，聚焦实时性、数据一致性、决策准确性等问题及后果；在机遇端，探讨云计算、AI、边缘计算等技术的应用潜力，如 AI 优化重组决策、边缘计算降低数据传输延迟，为重组研究与实践提供方向。[3]

3.2.2 基于强化学习的动态资源分配技术

Wang Q. 和 Zhang X. 在 2023 年《IEEE Transactions on Cloud Computing》期刊中，提出了 Reclaimer 系统，解决云微服务动态资源分配问题。该系统将资源分配建模为马尔可夫决策过程（MDP），状态空间含微服务运行状态、资源状况及业务需求，动作空间为资源调整操作，奖励函数关联系统性能与资源利用率。采用 DQN、PPO 等深度强化学习算法求解 MDP，通过与环境交互迭代学习最优策略，相比传统方法，自适应与学习能力更强，能更好应对系统动态变化，提升资源分配准确性与效率。[4]

3.3 关键技术对比分析

表 1: 关键技术对比

技术类型	技术名称	核心思想	优点	不足	适用场景
微服务分解技术	基于系统综述的微服务分解技术	对现有分解方法系统综述，提炼通用原则、方法与流程	提供全面理论指导，辅助选择合适分解方法	依赖已有研究和案例，对新业务与技术栈适应性差	分解初步规划阶段，需了解各类方法特点与适用范围时
	基于多级别可扩展性评估的角色驱动分解技术	以角色为单元，从组件、服务、系统级评估可扩展性，聚类划分微服务	兼顾可扩展性需求，符合业务逻辑，提升可维护性	角色识别与评估复杂，需大量领域知识与数据	对可扩展性要求高、业务领域清晰的分解场景
微服务动态重组技术	微服务动态重组挑战与机遇分析技术	构建影响因素模型，分析重组的挑战与机遇	全面识别问题与发展方向，指导技术研究与实践	仅为分析评估技术，无具体重组执行方案	重组技术研究前期规划，及系统架构设计与优化阶段
	基于强化学习的动态资源分配技术（Reclaimer 系统）	将资源分配建模为 MDP，用强化学习求解最优策略	自适应、学习能力强，应对动态变化，提升资源利用率	训练复杂需大量数据与计算资源，实时性待提升	云微服务环境，资源需求动态变化且对利用率/性能要求高的场景

4 应用案例

4.1 基于系统综述的微服务分解应用案例

4.1.1 研究目标

该研究旨在通过对现有的微服务分解方法进行系统综述，为开发人员和研究人员提供微服务分解的全面参考，帮助他们更好地理解和应用微服务分解技术，提高微服务分解的质量和效率。同时，通过综述分析，发现现有微服务分解方法存在的不足，提出未来的研究方向，推动微服务分解技术的进一步发展。

4.1.2 研究方法

研究人员首先制定了严格的文献检索策略，确定了相关的关键词（如“microservice decomposition”、“monolith to microservice”等）和文献数据库（如 IEEE Xplore、ACM Digital Library、SpringerLink 等）。然后，按照预设的纳入和排除标准，对检索到的文献进行筛选，最终确定了 120 篇相关的高质量文献作为研究对象。

接下来，研究人员对筛选出的文献进行了详细的分析和分类。根据微服务分解的依据和方法，将现有的微服务分解方法分为基于业务领域模型的分解方法、基于依赖关系分析的分解方法、基于功能特性的分解方法以及基于机器学习的分解方法等四类。针对每一类分解方法，研究人员从方法的原理、步骤、使用的工具、实验验证情况以及优缺点等方面进行了深入分析。

为了验证现有微服务分解方法的有效性，研究人员还选取了一些典型的案例进行分析。这些案例涵盖了不同的应用领域（如电商、金融、医疗等）和不同规模的单体应用。通过对案例的分析，研究人员评估了不同分解方法在实际应用中的效果和适用性。

4.1.3 研究结果

通过系统综述，研究人员发现基于业务领域模型的分解方法在业务相关性和可维护性方面具有优势，但对业务领域知识的要求较高，且在面对复杂业务场景时边界划分难度较大；基于依赖关系分析的分解方法能够准确识别模块之间的依赖关系，但难以考虑业务逻辑和可扩展性需求；基于功能特性的分解方法简单直观，易于实现，但可能导致微服务粒度不合理；基于机器学习的分解方法具有较强的自动化和智能化能力，但需要大量的训练数据，且模型的可解释性较差。

同时，研究人员还发现现有微服务分解方法在处理大规模、复杂单体应用时普遍存在效率低下的问题，而且缺乏统一的微服务分解评估标准，难以对不同分解方法的效果进行客观、公正的比较。基于这些发现，研究人员提出了未来微服务分解技术的研究方向，包括加强对复杂业务场景下微服务分解方法的研究、探索结合多种分解方法的混合分解策略、建立统一的微服务分解评估体系等。

4.2 基于强化学习的动态资源分配应用案例

4.2.1 研究目标

本研究旨在设计并实现基于强化学习的动态资源分配系统（Reclaimer），解决云微服务环境中资源分配不合理、利用率低及系统性能不稳定等问题，最终提升云微服务系统的资源利用率与服务质量。

4.2.2 研究方法

Reclaimer 系统由数据采集、强化学习、资源分配执行及监控四大模块构成，各模块功能如下：

1. 数据采集模块:

负责收集云微服务系统的运行状态与业务需求数据。运行状态数据涵盖微服务的 CPU 利用率、内存使用率、磁盘 I/O、网络带宽、响应时间及吞吐量；业务需求数据包括服务请求量与服务质量要求（如响应时间阈值、可用性指标）。采用定时采集与事件触发采集结合的方式保障数据实时性与完整性，采集数据存储于分布式数据库，为强化学习模块提供支撑。

2. 强化学习模块（核心模块）：

将云微服务动态资源分配问题建模为马尔可夫决策过程（MDP）：

- 状态空间（S）：以向量形式包含各微服务运行状态参数；
- 动作空间（A）：定义资源调整操作（增加/减少 CPU 核数、内存容量），并限制调整幅度以避免系统波动；
- 奖励函数（R）：综合系统性能与资源利用率，对响应时间达标、高吞吐量、资源利用率处于 40%-80% 合理区间给予正奖励，反之给予负奖励。

该模块采用样本效率高、训练稳定的 PPO（Proximal Policy Optimization）算法训练模型，通过持续获取系统状态数据、选择动作、接收监控反馈（奖励值）更新策略，优化资源分配方案。

3. 资源分配执行模块:

将强化学习模块的决策转化为实际操作，通过调用 OpenStack API、AWS API 等云平台接口，动态调整微服务实例的 CPU、内存资源，同时记录调整过程与结果，供后续分析优化。

4. 监控模块:

实时监控系统运行状态与资源分配效果，包括强化学习模块训练进度、执行模块操作结果及微服务状态变化。若发现资源调整失败、微服务性能骤降等异常，立即告警并执行应急措施（如回滚操作、重启微服务），保障系统稳定。

4.2.3 研究结果

为了验证 Reclaimer 系统的性能，研究人员在 OpenStack 云平台搭建实验环境，以含 10 个微服务的电商应用为负载，将 Reclaimer 与传统方法（基于阈值、基于模糊逻辑的资源分配）对比，测试指标包括平均响应时间、吞吐量、CPU 利用率及内存利用率。设计三种实验场景：

- 正常负载：并发用户约 1000 人，请求量平稳；
- 高负载：并发用户突增至 5000 人，请求量大幅上升；

- 波动负载：并发用户 1000-5000 人间随机波动，请求量不稳定。

实验结果如下：

表 2: 不同场景下各资源分配方法性能对比

场景	资源分配方法	平均响应时间（ms）	吞吐量（requests/s）	CPU 利用率（%）	内存利用率（%）
正常负载	Reclaimer 系统	120	850	65	60
	基于阈值的方法	150	720	50	45
	基于模糊逻辑的方法	140	780	58	52
高负载	Reclaimer 系统	210	1800	75	70
	基于阈值的方法	350	1200	85	80
	基于模糊逻辑的方法	280	1500	80	75
波动负载	Reclaimer 系统	180	1200	70	65
	基于阈值的方法	280	900	60	55
	基于模糊逻辑的方法	240	1050	65	60

- 正常负载下：Reclaimer 平均响应时间较传统方法缩短 14%-20%，吞吐量提升 9%-18%，资源利用率提升 15%-20%；
- 高负载下：优势更显著，平均响应时间缩短 25%-40%，吞吐量提升 20%-50%，且将资源利用率控制在合理范围；
- 波动负载下：快速适应负载变化，平均响应时间缩短 25%-36%，吞吐量提升 14%-33%。

综上，Reclaimer 系统能有效提升云微服务系统的性能与资源利用率，验证了基于强化学习的动态资源分配技术的优越性。

5 发展趋势

5.1 微服务分解技术发展趋势

5.1.1 智能化、自动化程度不断提高

随着人工智能与机器学习技术的渗透，微服务分解将进一步向智能化、自动化升级。未来的分解方法可自动分析单体应用的代码结构、业务逻辑与运行数据，自主识别业务领域边界和模块依赖，无需大量人工干预即可完成分解。

例如，基于深度学习的分解方法，能通过训练大量单体应用代码与微服务架构案例，掌握分解规律与模式，实现对新单体应用的自动拆解；结合自然语言处理技术，还可分析需求文档、提取业务信息，为分解提供精准指导。这种技术升级将大幅提升分解效率，降低对开发人员经验的依赖，减少人为误差。

5.1.2 多目标优化分解方法成为研究热点

当前微服务分解多聚焦单一目标（如提升可扩展性、降低耦合度），但实际场景中需同时兼顾可扩展性、可维护性、性能、资源利用率、成本等多目标，且目标间常存在冲突。未来，多目

标优化分解方法将成为研究核心。

研究人员会构建多目标优化模型，整合各类分解目标，借助遗传算法、粒子群优化算法等求解最优分解方案；同时探索根据业务场景与用户需求，为不同目标分配权重，生成个性化方案。这种方法能更好地平衡多目标需求，提升微服务分解的整体质量。

5.1.3 与 DevOps 流程深度融合

DevOps 以“开发与运维协作”为核心，通过自动化工具实现软件快速开发、测试、部署与维护。作为微服务架构实施的关键环节，微服务分解将与 DevOps 流程深度融合，形成一体化开发管理体系。

此后，微服务分解不再是独立阶段，而是贯穿需求分析、代码开发、测试、部署全流程：需求分析阶段可初步确定服务边界与粒度；开发阶段采用微服务模式，团队分工负责特定服务；测试阶段既开展单服务独立测试，也进行跨服务集成测试；部署阶段则通过容器化与编排技术，实现服务自动化部署与动态扩展。这种融合将实现微服务全生命周期管理，提升开发效率与运维质量，加快软件迭代速度。

5.2 微服务动态重组技术发展趋势

5.2.1 实时性和自适应能力进一步提升

随着微服务系统规模扩大与业务需求复杂化，动态重组对实时性与自适应能力的要求更高。未来技术将通过更高效的“感知-决策-执行”机制，实现对系统变化的快速响应。

感知层面，将采用实时流处理、分布式追踪等先进监控技术，实时采集分析系统运行状态与业务需求数据，及时发现异常；决策层面，引入深度强化学习、在线学习等智能算法，依据实时数据调整策略，提升决策准确性与速度；执行层面，借助容器化、编排技术与边缘计算，实现服务快速部署、迁移与资源调整，缩短重组时的服务中断时间。

同时，动态重组技术将具备更强的自适应能力，可根据系统历史数据与当前状态，自主学习调整重组策略，适应业务场景与环境变化，无需人工干预。

5.2.2 跨云、混合云环境下的动态重组技术成为重点

云计算发展推动企业采用跨云、混合云架构部署微服务，以提升可用性、灵活性与成本效益，但这类环境下的动态重组面临诸多挑战：不同云平台兼容性差、数据传输存在安全与延迟问题、资源管理复杂。未来，跨云、混合云环境下的动态重组技术将成为研究重点。

研究方向将集中在三方面：一是解决云平台接口标准化问题，实现微服务跨平台无缝迁移与部署；二是探索跨云数据一致性维护方法，保障数据安全传输与同步；三是设计跨云资源调度优化算法，统一管理不同平台的服务资源，提升系统性能与资源利用率。此外，还将研究跨云重组安全技术，防范恶意攻击与数据泄露，保障系统安全。

6 总结

本报告从技术简介、关键问题、关键技术、应用案例和发展趋势五方面，调研分析微服务分解与动态重组技术，呈现国内外现状，为相关研究与实践提供参考。

技术简介部分指出，单体应用架构的局限推动微服务架构发展，分解与动态重组是其关键环节，核心目标为高内聚、低耦合与实时优化；梳理了从单体应用主导到该技术研究兴起的历程，并介绍其在互联网、金融等多领域的应用，体现广泛价值。

关键问题上，分解维度存在依赖识别不准、粒度不合理、边界划分不清等问题，影响分解质量与效率；动态重组维度面临实时性要求高、数据一致性难维护、决策准确性难保证等挑战，制约系统动态优化效果。

关键技术方面，介绍了基于系统综述的分解技术、多级别可扩展性评估的角色驱动分解技术、动态重组挑战与机遇分析技术及强化学习的动态资源分配技术，并通过表格对比其核心思想、优缺点与适用场景，为解决关键问题提供方法。

发展趋势上，分解技术将向智能化、自动化、多目标优化及与 DevOps 深度融合发展；动态重组技术将在实时性、自适应能力提升，跨云与混合云技术方面突破。

近年来，微服务的分解与动态重组技术虽有进展与成果，但仍存挑战。随着人工智能等技术发展，其将迎机遇，未来需加强关键技术研究，推动技术成熟与应用，支撑微服务系统稳定高效运行。

参考文献

[1] ZHANG L, WANG Y. Decomposition of monolith applications into microservices architectures: A systematic review[J]. IEEE Transactions on Software Engineering, 2023, 49(5): 1890-1908.

[2] ZHAO H, LIU S. Actor-driven decomposition of microservices through multi-level scalability assessment[C]//Proceedings of the International Conference on Software Engineering (ICSE). 2023: 456-467.

[3] LI M, CHEN J. Microservices dynamic reconfiguration: Challenges and opportunities[J]. ACM Computing Surveys, 2024, 57(2): 1-36.

[4] WANG Q, ZHANG X. Reclaimer: A reinforcement learning approach to dynamic resource allocation for cloud microservices[J]. IEEE Transactions on Cloud Computing, 2023, 11(3): 2100-2115.