redhat.

白皮书

成功构建微服务架构

优势

- 简易开发和部署
- 更快捷、独立的部署
- 应用组件的独立可扩展性
- 避免长期技术锁定

执行摘要

微服务架构是一种创建松散耦合且自主化服务的新型架构风格。技术领域的新兴趋势， 例如 DevOps、平台即服务 (PaaS)、容器和持续集成与交付 (CI/CD) 方法，正在帮助组织超越面向服务的架构 (SOA) 等传统方式，实现更大规模的模块化系统创建和管理。然而，企业在将单体式应用以微服务形式重构时的收获确不尽相同。有效采用微服务的关键在于必须深刻理解为什么、以及应该如何采用微服务来构建应用。

改进面向服务的架构

面向服务的架构 (SOA) 的常规定义是：通过网络与其他组件通信来为其提供服务的应用组件。采用 SOA 的目的在于创建具有弹性的分布式应用，有效消除各类复杂的中央组件。

然而，SOA 与组织结构高度耦合，被用于支持新的内部结构。因此，该架构能否获得成功在很大程度上依赖于最终、经过重新规划的组织能力，以及设计该架构的团队结构。SOA 创建的并非是兼具松散耦合和自主化特点的系统，而是一个需要复杂基础架构的高度脆弱的系统。此外，早期的 SOA 实施会造成供应商锁定，因为专有中间件往往只专注于集中化逻辑、持久化、治理和管理。

相反，微服务架构则在构建应用的每个步骤（从开发、部署到运营）中兑现了 SOA 的最初承诺。微服务架构专注于简化技术，利用精简化的组件构建复杂系统。该架构通过基于异步 HTTP 或消息传递协议的简单、标准化的管道通信，来取代集中化逻辑和集成基础架构（基于重量级、非标准化的平台）。SOAP、XML 和其他重量级协议和数据格式将被轻量级 JSON（通过基于 HTTP 的 REST）所取代。每项微服务都有自己的数据存储，不需要集中化管理和持久化。

微服务采用持续集成 (CI) 和持续交付 (CD) 的方法和实践，以及多种在 SOA 中不常见的组件，例如：

- 多语言编程和持久化。
- 容器或不变虚拟机 (VM)。
- 具有弹性、可编程的基础架构即服务 (IaaS) 和 PaaS。

是否已为采用微服务做好准备？

您的组织是否符合以下条件：

- 构建了结构良好的单体式应用？
- 明确了将微服务用于满足哪些特定需求？
- 围绕微服务架重新调整了团队结构？
- 采用了最佳 DevOps 和 CI/CD 实践方法？
- 明确界定了应用的业务范围？
- 已实施微服务编排和管理工具与流程？

面向灵活、快速响应式 IT 的创新解决方案

加速部署

微服务的特点在于小而专，这是因其专注于域边界和一致性域建模等特点，且所需的代码更少。部署战略包括集中、能独立运行的数据存档（通常以 Linux 容器和 CI/CD 形式打包），以实现更快、更可靠的部署。该做法的结果是有效缩短了软件开发周期。这也推动和加速了各类新功能、bug 修复，以及经全面测试的安全补丁的发布速度。

模块化控制

凭借微服务，每项服务都能独立扩展，以满足临时或季节性的访问流量激增、完成批处理，以及其他各种业务需求。同时，改进后的故障隔离功能，可将各种服务问题（例如内存泄漏或数据库连接泄露）的影响限制在一定范围内，并确保不对其他服务产生影响。微服务的可扩展性补充了云服务的灵活性 — 让您能在有效改进服务的同时，在不中断服务的情况下同步处理更多客户请求。

更多选择

开源技术解决方案和组织方法正在引领微服务市场的发展。因此，微服务可以减少供应商锁定，免受专业技术的长期束缚，让您重获产品的选择自由，以充分满足企业的 IT 和业务需求。

为微服务奠定稳固基础

要想成功采用微服务架构，组织必须首先为其单体式架构创建一个稳固的基础。必须考虑和建立模块化、域边界和基本分布式系统理论，以发挥微服务的全部优势。

此外，微服务能为较复杂的系统创造更多优势。即便每项服务都完全独立，我们还必须满足一些运营方面的需求，例如以下功能：

- DevOps（开发运维）。
- PaaS。
- 容器或不变 (Immutable) VM。
- 服务复制、注册和发现。
- 主动监控和警告。

由于满足这些需求往往涉及较大的投入，而且不能立即获得回报。 因此利用微服务可能并非对于每个团队或项目都属于经济高效之举。对“单体优先的方法”进行评估，可确保应用的构建遵照以下可靠的设计原则、以明确域边界划分，让您可在需要可扩展性时逐步过渡到微服务架构。例如，即便是最基本的购物车应用也应包含：

- 有效的问题服务隔离。
- 利用明确定义的应用编程接口 (API) 实现高内聚和低耦合功能。
- 独立接口、API 和实施，遵循得墨忒耳定律或“最少知识准则”。
- 用于集合相关对象的域驱动设计。

企业之所以能够成功采用微服务在于其具有合理的组织架构，而非其使用的技术产品。构建一只采用扁平化组织结构、具备灵活性、多功能和自主化的高效团队是成功的关键。

一旦我们遵循可靠的软件架构准则创建了某个用于扩展的单体式应用，则可以将其以微服务的形式进行重构。而重构的最有效方法包含了以下步骤：

1. 首先确定应用域中的业务边界和业务逻辑差别，然后将各域分解为独立的微服务。
2. 识别出那些需要频繁改动的组件（例如由价格计算或法规变更而导致的业务规则更新），或者往往为解决安全漏洞而进行的修补部件。
3. 在定义了基本的、基于域的微服务后，我们还需进一步完善用于支持服务交互的 API。我们可以使用聚合器、代理、链式、分支、事件驱动和其他设计模式构成这些 API。

协调一致、技能精湛的团队

企业之所以能够成功采用微服务在于其具有合理的组织架构，而非其使用的技术产品。构建一只采用扁平化组织结构、具备灵活性、多功能和自主化的高效团队是成功的关键。

构建高效率、高技能的团队需要围绕业务功能，而非企业架构来重新协调人员配置 — 例如，Amazon 的“两张披萨团队”原则， 每组团队由 8 到 10 人组成，负责创建和维护其服务。此外，康威定律还指出，组织只能创造出与其组织结构相仿的设计。例如，如果对一个团队按照开发、运营、质保及安全几大部分进行划分，则会对团队的业务灵活性造成一定的限制，并可能导致进度延误。

因此，企业在过渡到微服务之前，应先创立 DevOps 实践，以预先明确其通信策略，从而有效缓解或防范这些问题的发生，并避免创建失败的 SOA、非高效的微服务架构。

高效的管理工具

除了设计合理的软件和组织有序的高效团队之外，构建高度可扩展的架构还需要恰当的工具来帮助管理额外的服务和应用组件，其中包括：

- 服务注册和发现工具，例如 Kubernetes。
- 容器化应用的标准化打包格式，例如 Docker，以及大规模复制容器的编排工具，例如 Kubernetes。红帽 OpenShift 可提供满足上述需求的可靠开源技术产品。
- CI 持续集成环境创建工具，例如 Jenkins 或 Shippable 出品的 Docker and Kubernetes。
- 依赖性解决方案，例如 Nexus。
- 故障切换和弹性工具，包括 Hystrix 和 Ribbon 等库。
- 服务监控、提醒和事件工具，例如 ELK（ElasticSearch、LogStash 和 Kibana）堆栈。

数据管理

过渡到微服务的另外一个重要考虑因素是数据管理。不同于 SOA，微服务不会共享数据。每项微服务都具备独立物理数据存储和混合持久化，让多种数据库引擎都能在各个微服务下运行。因此，用户可以以每项服务为基础选择数据存储，而不必将所有数据都存储在一个企业关系数据库管理系统 (RDBMS) 中。

但维护企业数据库的多个副本可能会增加许可成本和复杂性。此外，数据存储也需要通过协调实现一致性。常规的提取、转换和加载 (ETL) 或数据虚拟化工具有助于数据规范化，事件溯源则是一种广为认可的设计模式，有助于协调数据存储，以适应追溯更改。

结论

微服务架构可为组织带来诸多优势，例如各种应用组件的独立可扩展性，以及更快捷、更简便的软件开发与维护。但微服务可能无法为所有团队或项目带来同等的优势。对部分组织而言，还可能成为无法立即获得回报的重大投资。过渡到微服务应是一个循序渐进的过程，仅重构部分现有应用（而非全面过渡）是较为行之有效的做法。为成功采用微服务架构，组织必须首先根据现有平台标准构建设计合理的应用，随后根据需要将应用重构为微服务集合，以满足业务需求。配以恰当的人员、流程和工具，微服务将促进更快的开发和部署，并让企业摆脱长期技术锁定，从而获得业务发展和产品选择自由。

销售及技术支持

800 810 2100
400 890 2100

红帽软件（北京）有限公司

北京市朝阳区东大桥路 9 号侨福芳草地大厦 A 座 8 层 邮编: 100020
8610 6533 9300

红帽官方微博　红帽官方微信

cn.redhat.com
INC0336100_0216