大厂面试课之Go语言

第10课：微服务

同学们好！从这节课开始我们将一起进入基于Go语言的Web应用与微服务开发的学习。Web应用开发主要是指在B/S架构模式下，基于浏览器的应用程序开发，比如京东、淘宝、当当网等。近年来随着技术的进步及移动端的普及，在Web应用开发领域出现了越来越明确的职责分工。一款基于Web的互联网产品基本上会分为Web UI设计、Web前端开发及Web后端开发等。在一些大型的互联网公司，还会设置独立的Web架构开发组，专门负责Web框架的维护与迭代。Web后端开发主要用到的编程语言包括Java、PHP、Python等。当然，随着Go语言的兴起，它渐渐成为近年来服务端开发的另一种选择，甚至在某种程度上已经取代了PHP，并不断撼动Java在这一领域的统治地位。Web后端服务器的架构形式主要包括，相对传统的单体架构和近年流行的微服务架构两种。相对于单体架构，体现更低耦合度的微服务架构，不仅提高了开发效率，而且在维护性和扩展性方面也具有显著的优势。

目录

在了解基于微服务架构的Web应用开发之前，我们首先必须明确到底什么是微服务，以及微服务概念的由来，通过与单体式服务的对比，更深刻地理解微服务的优势与价值。

什么是微服务?

那么究竟什么是微服务呢?

知识讲解

狭义的微服务

- 一个微服务就是一个可以被用户感知的独立的最小功能单元

广义的微服务

- 微服务是一种基于彼此协同的细粒度服务的分布式解决方案

狭义的微服务是指任何可以被用户感知的独立的最小功能单元。它可以是一台计算机，也可以是一个独立的进程或线程，甚至是一个能够为用户提供某种服务的库或函数。而广义的微服务则是指一种基于彼此协同的细粒度服务的分布式解决方案。比如一个典型的电商系统通常由很多业务模块组成。用户管理模块、商品管理模块、订单管理模块、支付模块、营销模块、物流模块等等。如果采用单体架构，模块与模块间的耦合度会非常高，后期的维护和扩展也会非常困难。而如果采用微服务架构，每个业务模块都可以做成独立的服务器程序，分布部署于多台计算机上，以远程调用的方式彼此协同工作。既提升了开发效率，又有利于维护和扩展，同时满足对现代分布式系统高可用、高并发、高性能的要求。

微服务概念的由来

从谷歌的搜索指数来看，微服务概念的热度是在进入2017年以后突然爆发的。国内各大会议和论坛有关微服务的讨论，也是在这以后如雨后春笋般层出不穷。各大一线互联网公司纷纷将微服务架构引入到实际业务系统中落地。那么有关微服务的概念究竟缘何而来呢?

微服务架构是一种基于分而治之，合而用之的设计理念

知识讲解

- 将一个大型应用拆分为一组小型服务
- 每个小型服务都运行在独立的进程中
- 小型服务之间通过轻量级机制（如HTTP）通信
- 这些小型服务是围绕业务功能构建的
- 每个小型服务都可以自动化独立部署
- 这些小型服务可以使用不同的编程语言开发
- 这些小型服务可以采用不同的数据存储技术
- 尽量避免对小型服务的集中式管理

首先，微服务架构是一种基于分而治之，合而用之的设计理念。微服务架构的核心思想就是将一个规模巨大的复杂应用拆分为若干个规模相对较小的简单服务。每个服务都可以运行在独立的进程中，并基于某种轻量级机制，如HTTP等，与其它服务通信。这些相对独立的小型服务是围绕业务而被划分和构建的，并可被自动化地独立部署。每个这样的小型服务都可以使用不同的编程语言开发，基于不同的数据存储技术。尽可能地避免对小型服务的集中式管理。

知识讲解

微服务是基于微服务架构设计理念被拆分出来的小型应用

- 分而治之，面向业务的拆分，让复杂的问题变得简单
- 合而用之，轻量级通信整合，让微小的力量变得强大

正因为如此，我们通常将微服务视为某种基于微服务架构设计理念而被拆分出来的小型应用。分而治之专注于面向业务的拆分，为的是让复杂的问题变得简单。合而用之则更强调基于轻量级通信的整合，所追求的是让微小的力量变得强大。

微服务和单体式服务的区别

微服务纵有千般好处但也绝非解决一切问题的灵丹妙药，单体应用纵有万般不是但也并非毫无用武之地。它们在不同的场合，满足不同的需求，体现不同的价值。脱离实际业务场景，盲目追求技术热点，往往会因缺乏对新理念的掌控能力而导致业务的整体稳定性受损。

单体架构在小规模系统中表现良好，系统规模越大问题越多

知识讲解

- 复杂度上升：十几年的老系统，几十万行代码，复杂度越高，越难以解决新问题
- 技术债上升：人员流动频繁，遗留bug众多，积累的技术债务越来越多
- 耦合度上升：修复旧bug，引入新bug，牵一发而动全身，沟通、管理成本与日俱增
- 交付周期长：构建部署耗时，人员培养困难
- 选型成本高：在统一技术平台上解决所有问题难度大，引入新技术新框架风险高
- 可扩展性差：成员负荷、数量的增加，对原系统造成较大影响，困难大风险高
  - 垂直扩展：成员数量不变，成员负荷增加
  - 水平扩展：成员负荷不变，成员数量增加

事实上，单体架构在小规模系统中的表现是足够优异的，只是随着系统的规模越来越大，表现出的问题才越来越多。随着时间的流逝，单体架构系统的复杂度呈上升趋势。毕竟十几年的老系统，几十万行的代码量，复杂度越高越难以解决新问题。与复杂度上升相伴而行的是技术债的上升。随着人员的频繁流动，遗留bug众多，积累的技术债务越来越多。单体架构本身就存在耦合度过高的问题。一边修复旧bug，一边引入新bug，牵一发而动全身，沟通、管理的成本与日俱增。系统结构复杂，构建部署耗时，人员培养困难，共同导致单体架构系统的交付周期通常都比较长。单体架构还需要考虑在统一技术平台上解决所有问题的难度，为这样的系统引入新技术新框架的风险通常都很高。对于单体架构系统，无论是成员数量不变，负荷增加的垂直扩展，还是负荷不变，成员数量增加的水平扩展，对原系统都不可避免地会造成较大影响，困难大风险高，总之可扩展性差。

微服务架构的优点

知识讲解

- 职责单一：每个微服务都是独立的程序（进程），只负责一个业务功能
- 轻量级通信：不同微服务之间借助RPC协议实现跨语言的进程间（网络）通信
- 独立性：每个微服务相对独立，鲜有彼此依赖，可以同时开发，同时测试
- 迭代开发：增加新的微服务即可添加新功能，对整体设计水平的要求不高

与单体架构的缺点针锋相对的自然是微服务架构的优点。首先是职责单一。每个微服务都是一个独立的程序，即进程，只负责一个业务功能。其次是轻量级通信。不同微服务之间借助RPC协议实现跨语言的进程及网络间通信。再次是独立性。每个微服务都相对独立，彼此依赖很少，完全可以同时开发，同时测试。最后是迭代开发。增加新的微服务即可增加新的业务功能，对原有系统的影响很小，对系统的整体设计水平要求不高。

微服务架构的缺点

知识讲解

- 运维成本高：每个微服务都需要单独开发、编译、测试和部署
- 复杂度较高：分布式系统相对复杂
- 接口成本高：多个服务之间基于接口通信，远程过程调用导致性能损失
- 重复性工作：多个微服务有很多相似的代码，如访问配置、记录日志等
- 业务分离难：拆分粒度不好把控，拆得过碎，可考虑引入中台统一管理

当然，我们也要看到，微服务架构并不是十全十美的。首先，它的运维成本通常较高。毕竟每个微服务都需要被单独地开发、编译、测试和部署。其次，微服务架构系统的整体复杂度并不比单体架构系统低多少。分布式系统本身就要比集中式系统复杂。再次，微服务的引入从某种程度上增加了更多的接口成本。多个服务之间借助接口通信，远程过程调用必然导致更多的性能损失。另外，多个微服务之间会有很多相似甚至相同的代码，如访问配置、记录日志等，这就不得不增加很多重复性的工作。最后，微服务架构系统的拆分粒度很难把控，容易拆得过碎，可考虑引入中台系统统一管理。

微服务架构与单体架构对比

知识讲解

对比项	单体架构	微服务架构
设计	技术选型	服务间逻辑
难点	应用逻辑	架构逻辑
技术	单一且封闭	多样且开放
部署	不经常且容易	频繁且复杂
测试	简单	复杂
运维	简单	复杂

以下我们从设计、难点、技术、部署、测试、运维、速度、吞吐量、扩展性、管理重点和故障影响范围等几个维度，对单体架构和微服务架构做一个综合的横向对比。首先从设计上看，单体架构的重点在于技术选型，毕竟以后所有的业务都要在一个框架中运行，一开始的技术选型如果没有做好，后面会很难扩展。微服务架构就不同了，每个业务都是独立的程序，不同的业务可以选择不同的技术。当下的技术只要能满足目前的业务需求就可以了，不需要为未来的可能性考虑太多。因此微服务架构会把注意力更多地投入到服务间逻辑的设计上，从业务角度做出更加合理的服务划分，为今后的潜在需求留出足够的扩展空间。从这里也可以看出，单体架构的难点在于应用逻辑的设计，而微服务架构的难点则在于架构逻辑的设计。从技术层面看，单体架构的技术通常是单一且封闭的，毕竟一旦确定了整个平台的技术路线，后面无论再怎么扩展业务，始终还是运行在同一个平台之上，所依赖的技术工具不会有太多变化。

微服务架构就不同了，每一种业务所仰赖的技术支撑都是彼此独立的，新的业务为了适应新的商业模式，对技术也会提出新的要求。微服务架构并不会限制丰富的业务场景对技术多样性的追求，在这一点上比单体架构要开放得多。当然我们也要看到，单体架构系统的部署、测试和运维确实比微服务架构系统要简单。微服务架构源于分布式系统固有的复杂性，必然导致其部署会更加频繁，测试、运维的难度也更大。

微服务架构与单体架构对比

知识讲解

对比项	单体架构	微服务架构
速度	快	慢
吞度量	小	大
扩展性	差	好
管理重点	控制开发成本	服务治理与调度
故障影响范围	大	小

从应用系统的运行速度来看，单体架构通常会显得更快。微服务之间的远程调用，必然导致不可忽略的性能损失。但也正是因为分布式系统的高可用、低延迟和负载均衡等特性，使得微服务架构在业务吞吐量上要远胜于单体架构。基于分而治之的设计思想，专注不同业务的各个微服务之间的耦合度通常很低，这为系统扩展提供了极大的便利，不存在单体架构那种牵一发而动全身的维护风险。从管理角度看，单体架构的管理重点主要集中在对开发成本的控制上，而作为典型分布式系统的微服务架构，如何提升服务治理与调度的水平，才是其管理的重中之重。任何系统都不可避免地会发生故障。单体架构的高耦合性决定了，其故障波及的范围通常都会很大，而微服务架构却恰恰相反，它可以很轻松地将受故障影响的范围，限制在少数几个甚至一两个微服务的内部，而其它微服务丝毫不受影响，仍能正常工作。

Tedu.cn 达内教育

更多精彩，敬请期待

谢谢大家，我们下节课再见！