基于 Spring Cloud 微服务架构的应用

文/李娜

摘要

本文介绍了微服务的基本概念、优缺点以及实现方式，对基于 Spring-Cloud 的微服务架构模式进行分析，最后给出微服务的应用场景。

【关键词】微服务 Spring Cloud 分布式

早期的系统开发，都采用了单体应用模式，比如淘宝、京东、豆瓣网等，这种模式是比较适合公司创业初期的，因为比较简单，一个工程，一个数据库，最后整体打包发布就上线了。但随着业务的发展，特别是系统的访问量、数据量的急剧增加，单体应用已经无法满足业务需求，因此将庞大的单体应用按照某种维度进行拆分，进行分布式部署，为了让这种分布式系统更加的规范、更容易管理，便形成了各种服务化的方式和工具，从基于 ESB 的 SOA（面向服务）的基础架构到当前流行的微服务架构模式，都是在不断适应越来越复杂的应用系统。

1 微服务简介

1.1 什么是微服务

微服务是一种新兴的软件架构模式，它把一个大型的单体应用或服务拆分为多个支持微服务。一个微服务的策略可以让工作变得更为简便，它可扩展单个组件而不是整个应用程序堆栈，从而满足服务等级协议。微服务最重要的就是这个“微”字，怎样才能成为“微服务”呢，其实没有标准，这要根据系统的实际功能需求而定，并不是拆分得越细越好，应该在业务层面上去划分，能够满足各方面的需求。

1.2 微服务的特点

1.2.1 微服务的优势

1.2.1.1 单个服务容易开发和维护

相比传统的单体应用，单个微服务的功能更加单一，只关注特定的功能实现，因此，在开发和维护上不需要多方的协调以及冗长的业务流程。

1.2.1.2 服务可以独立部署和扩展

微服务的开发、部署和维护都是相对独立的，互不干扰，服务之间通过标准的接口进行交互，可以很方便的对服务进行扩容。

1.2.1.3 可以由不同的团队来开发

在微服务的整体架构上，通常都是按照业务进行服务划分，可以由不同的团队进行开发，服务间通过接口进行交互。

1.2.1.4 服务开发技术的选项更加灵活

每个服务的技术选型都可以由相应的团队决定，可以尝试各种最新技术，更加的灵活。

1.2.2 微服务的不足

1.2.2.1 管理微服务是一件麻烦的事情

单个服务的开发和维护相对来说是很容易的，但从整个系统上看，这是一件很麻烦的事情，因为系统从单体变成了分布式，多个服务分布在不同的服务器中，需要完善的服务监控和管理的能力。

1.2.2.2 来自分区数据库带来的实现问题

每一个服务都有自己的数据库，这样才能达到真正系统微服务化的目的。由此会带来一个最为突出的问题就是分布式事务，实现上可以选择按照 ACID 的强一致性或者基于 BASE 理论的最终一致性。

1.2.2.3 服务间调用的成本更高

由于服务都是分布式部署，服务之间的调用相比传统的本地方法调用，需要更大的成本，调用过程中还会遇到安全、网络抖动等外在的问题。

2 微服务的实现方式

微服务并不是一种技术或者框架，而是一种设计理念或者架构模式，它基于模块化、组件化等架构思想。微服务的实现方式，目前主要有两种，一种是基于 RPC 的方式，另一种是基于 HTTP 的 Restful 方式，这两种实现方式各有利弊，可以选择其中的一种，也可以将两种结合起来使用。在实际应用中，系统内部服务之间的调用通过 RPC 方式，可以满足对性能方面的需求；面向客户端以及对外的服务输出采用 Restful 方式，一是调用简单，二是更加标准，降低调用成本。

3 Spring-Cloud的技术架构

Spring Cloud 是一种基于 Spring Boot 的微服务框架，它实现了微服务架构中常用的组件，目前比较常用的组件是基于 Netflix 对多个开源组件的封装，为微服务架构开发涉及的配置管理、服务治理、熔断机制、智能路由、微代理、控制总线、一次性 token、全局一致性锁、leader 选举、分布式 session、集群状态管理等操作提供了一种简单的开发方式，spring-cloud 的基础架构如图 1 所示。

3.1 网关

接受外部对服务接口的访问，屏蔽底层服务的具体实现，提供权限、认证、安全、监控、限流等基础服务，常见的有 Zuul、spring-cloud-gateway。

3.2 Ribbon

Spring-Cloud-Ribbon 是基于 HTTP 和 TCP 的客户端负载均衡工具，它基于 Netflix Ribbon 实现，可以轻松地将面向服务的 REST 模版请求自动转换成客户端负载均衡的服务调用。

3.3 Eureka

Spring-Cloud-Eureka 是对 Netflix Eureka 的封装以实现服务发现功能，它包含了 Server 端和 Client 端。Eureka Server 提供服务注册功能，各个节点启动后，会在 Eureka Server 中进行注册；Eureka Client 用于简化与 Eureka Server 的交互，可以方便地访问注册中心的服务。

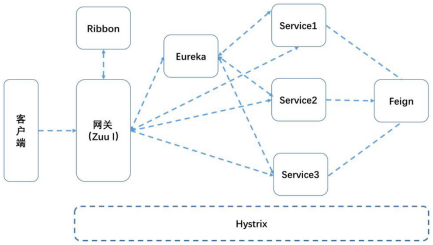

图 1：Spring-cloud 基础架构图

3.4 Hystrix

Hystrix 是一种熔断器，实现服务的限流、熔断、降级等功能，可以很好的保证服务在高并发情况下的稳定性。

4 微服务应用场景

任何的架构模式都需要根据实际的业务场景而定，不能盲目的追求最新的技术，最适合的就是最好的。对于微服务而言，以下的场景或者条件是比较适合的。

系统业务量越来越大，核心业务和非核心业务变得泾渭分明，这个时候将你的业务系统拆分为细颗粒的服务进行管理，通过断路由、降级、限流等服务管理措施保证系统高可用。

开发团队具有足够的实力，包括系统架构、开发、运维等方面，可以解决微服务带来的各种问题，充分利用好微服务带来的好处。

5 结论

综上所述，相对于传统的单体应用，微服务带来了系统整体架构上的转变，也给系统的设计和开发带来了很多好处，但也不可避免的存在一些问题，这需要根据系统自身业务场景来选择适合自己的架构模式。

参考文献

[1] 不甘于平凡的溃败的博客．微服务初探．https://blog.csdn.net/wohiusdashi/article/details/83957771

[2] 李忠民，齐占新．业务架构的微应用化与技术架构的微服务化 -- 兼谈微服务架构的实施实践 [J]．科技创新与应用，2016.

[3] 王玉良．基于 Spark 的短时交通流预测系统设计 [J]．桂林电子科技大学，2017.

[4] 赵善龙，孙婉婷．基于微服务架构的互联网 + 农业平台设计 [J]．通信管理与技术，2017.

作者简介

李娜（1988-），女，山东省新泰市人。研究生学历。主要研究方向为计算机应用技术。

作者单位

重庆青年职业技术学院　重庆市　400712