链滴

# 01 微服务简介

作者：pleaseok

原文链接：https://ld246.com/article/1544359207735

来源网站：链滴

#01微服务简介

# 单体架构存在的不足

● 业务越来越复杂，单体应用的代码量越来越大，代码的可读性、可维护性和可扩展性下降，新人接代码所需的时间成倍增加，业务扩展带来的代价越来越大

● 随着用户越来越多，程序承受的并发越来越高，单体应用的并发能力有限。

● 测试的难度越来越大，单体应用的业务都在同一个程序中，随着业务的扩张、复杂度的增加，单体用修改业务或者增加业务或许会给其他业务带来一定的影响，导致测试难度增加

##什么是微服务

2014年，Martin Fowler 与 James Lewis 共同提出了微服务的概念，定义了微服务是由以单一应用序构成的小服务，自己拥有自己的行程与轻量化处理，服务依业务功能设计，以全自动的方式部署，其他服务使用 HTTP API 通讯。同时服务会使用最小的规模的集中管理 (例如 Docker) 能力，服务可用不同的编程语言与数据库等元件实作。

####1.微服务单元按业务来划分

根据 Martin Fowler 的定义，微服务的“微”是按照业务来划分的 。一个大的业务拆分为若干个小业务，一个小的业务也可以拆分为若干个更小的业务。并且这些微服务单元是高度组件化的模块，服与服务之间没有任何的耦合。也就是说，一个小的业务的微服务需要传统开发中的一整个团队，包括UI团队、服务端团队、数据库和运维团队，对开发人员提出了更高的要求。

####2.微服务通过HTTP来互相通信

微服务是按业务划分，并运行在各自的进程中。微服务单元之间的通信方式一般使用HTTP这种简单通讯机制，更多的时候是使用RESTful API风格。服务与服务之间也可以通过轻量级的消息总线来通，例如RabbitMQ、Kafaka等，通过发送消息或者订阅消息来达到服务与服务之间通信的目的。服务服务通信的数据格式，一般为JSON、XML，这两种数据格式与语言、平台、通信协议无关。有时也用Protobuf。

####3.微服务的数据库独立

微服务的一个特点就是按业务划分服务，服务与服务之间无耦合，就连数据库也是独立的 。这样做的处在于，随着业务的不断扩张，服务与服务不需要提供数据库集成，而是提供 API接口相互调用：还一个好处是数据库独立，单业务的数据盆少，易于维护，数据库性能有着明显的优势，数据库的迁移很方便。

####4.微服务的自动化部署

单体架构的应用程序只需要部署一次，而微服务架构有多少个服务就需要部署多少次。随着服务数量增加，如果微服务按照单体架构的部署方式，部署的难度会呈指数增加。这时需要更稳定的部署机制随着技术的发展，尤其是Docker 容器技术的推进，以及自动化部署工具（例如开源组件Jenkin s ）出现，自动化部署变得越来越简单。自动化部署可以提高部署的效率，减少人为的控制，部署过程中现错误的概率降低，部署过程的每一步自动化，提高软件的质量。

####5.微服务集中化管理

微服务系统是按业务单元来划分服务的 ，服务数量越多 ， 管理起来就越复杂，因此微服务必须使用中化管理。目前流行的微服务框架中，例如 Spring Cloud 采用 Eureka 来注册服务和发现服务，另 ， Zookeeper、 Consul 等都是非常优秀的服务集中化管理框架。

####6.分布式架构

分布式系统是集群部署的，由很多计算机相互协作共同构成，它能够处理海量的用户请求。分布式系

通过网络协议来通信，所以分布式系统在空间上没有任何限制，即分布式服务器可以部署不同的机房不同的地区。微服务架构是分布式架构，分布式系统比单体系统更加复杂，主要体现在服务的独立性服务相互调用的可靠性，以及分布式事务、全局锁、全局Id等，而单体系统不需要考虑这些复杂性。外，分布式系统的应用都是集群化部署，会给数据一致性带来困难。分布式系统中的服务通信依赖于络，网络不好，必然会对分布式系统带来很大的影响。如果一个服务出现了故障或者是网络延迟，在并发的情况下，会导致线程阻塞，在很短的时间内该服务的线程资源会消耗殆尽，最终使得该服务不用 。由于服务的相互依赖，可能会导致整个系统的不可用，这就是“雪崩效应”。为了防止此类事件发生，分布式系统必然要采取相应的措施，例如“熔断机制”。

####7.熔断机制

为了防止“雪崩效应”事件的发生，分布式系统采用了熔断机制。当某个服务出现故障，请求失败次超过设定的阀值之后，这个服务就会开启熔断器，之后这个服务不进行任何的业务逻辑操作，执行快失败，直接返回请求失败的信息。

熔断器还有另外一个机制，即自我修复的机制。当某个服务熔断后，经过一段时间，半打开熔断器。打开的熔断器会检查一部分请求是否正常，其他请求执行快边失败，检查的请求如果响应成功，则可判定该服务正常了，就会关闭此服务的熔断器；如果此服务还不正常，则继续打开熔断器 。

# 微服务的优势

TODO

# 微服务的不足

TODO

# 微服务和SOA的关系

SOA即面向服务的架构。SOA 的实施思路是根据 ESB 模式来整合集成大量单一庞大的系统，这是 SO 主要的落地方式 。 然而， SOA 在过去 20 年并没有取得成功 。在谈到微服务时，人们很容易联想它是一个面向服务的架构。的确，微服务的概念提出者 Martin Fowler

没有否认这一层关系。