基于微服务的软件架构设计咨询

微服务架构的概念

微服务架构模式（Micro service Architect Pattern）是近两年在软件架构模式领域里出现的一个新名词。虽然其诞生的时间不长，但其在各种演讲、文章、书籍上所出现的频率已经让很多人意识到它对软件领域所带来的影响。微服务架构（Microservice Architect）是一种架构模式，概念源于 2014 年 3 月 Martin Fowler 所写的一篇文章“Microservices” (http://martinfowler.com/articles/microservices.html)

它提倡将单体架构的应用划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。

微服务的诞生并非偶然。它是互联网高速发展，敏捷、精益、持续交付方法论的深入人心，虚拟化技术与 DevOps 文化的快速发展以及传统单体架构无法适应快速变化等多重因素的推动下所诞生的产物。

微服务架构的优点

- 每个服务都比较简单，只关注于一个业务功能。
- 微服务架构方式是松耦合的，可以提供更高的灵活性。
- 微服务可通过最佳及最合适的不同的编程语言与工具进行开发，能够做到有的放矢地解决针对性问题。
- 每个微服务可由不同团队独立开发，互不影响，加快推出市场的速度。
- 微服务架构是持续交付(CD)的巨大推动力，允许在频繁发布不同服务的同时保持系统其他部分的可用性和稳定性。

微服务对企业的价值

微服务会为企业带来诸多价值，比如：

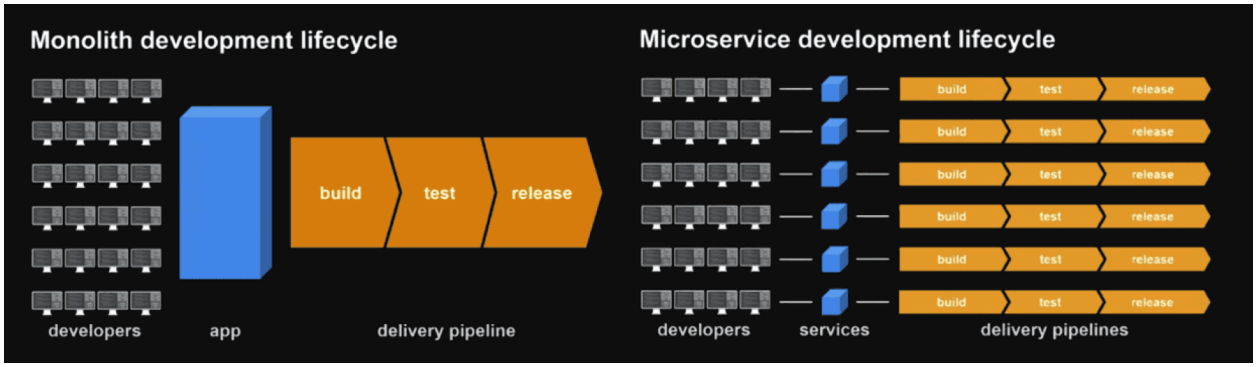
- 支持企业业务的敏捷性，随时可以增加新的业务功能，并快速上线。
- 在微服务架构的应用中添加新的功能，不需要重写整个应用；
- 更小的代码库维护起来更简单也更快捷，节省了开发成本和时间，从而提高了生产效率；
- 应用的各个模块可以分别扩展，而且更容易部署。支持灰度发布、金丝雀发布等，使得系统和应用在在线升级。

案例：Amazon 拥抱 DevOps 哲学并迁移到微服务基础设施

2001 年，Amzon.com 的零售网站是一个大型的单体，多层结构，每层都有多个组件，但是耦合度很高，运行起来就像一个整体。

『许多初创和企业项目开始时都采用这种架构，因为这种方式起步很快，但是随着项目的成熟和开发者的增多，代码库会变得越来越庞大，架构也会变得越来越复杂，单体架构会带来额外的开销，软件开发周期也会变慢。』

Amazon 有大量的开发者服务于一个大的单体架构的网站，尽管每个开发者只负责那个应用中的一小块，一旦有更新，也还是要在与项目中其它模块的协调中花费大量精力。

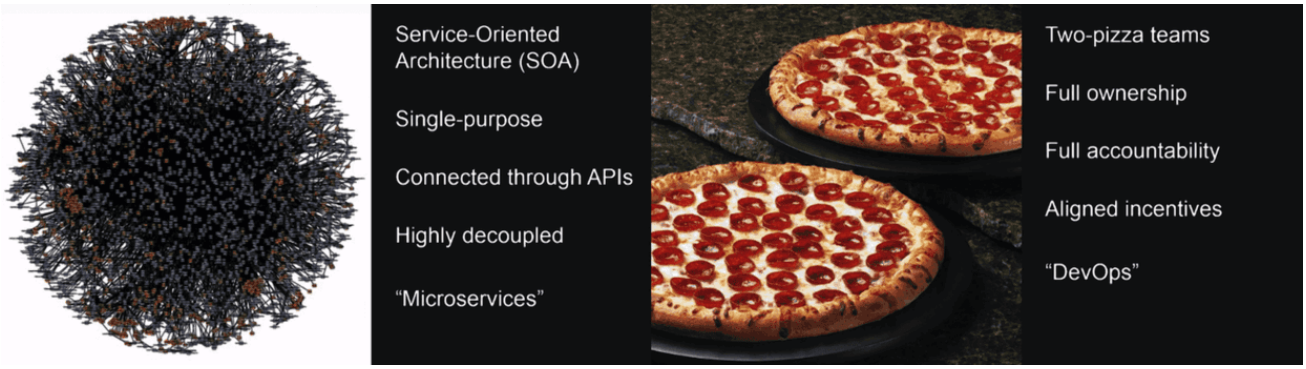


而添加新功能或修改 bug 时，还需要确保这些更新不会打断其他人的工作；如果要更新一个共享的库，需要通知每个人去升级这个库；如果要进行一个快速修复，不仅要协调自己的时间，还要协调所有的开发者。

即使每次的更新规模很大，依然给交付环节带来了很大的开销，整个新的代码库需要被重新构建，所有的测试用例需要重新运行，最终要将整个应用部署到生产环境中。

在 2000 年左右，Amazon 甚至有一个团队专门负责将应用的最新版本，手工部署到生产环境中。

对于软件工程师来说这很令人沮丧，最重要的是软件开发周期变得很慢，创新能力也减弱了，所以他们进行了架构和组织的重大变革。



这些变化开始于架构级别：Amazon 从单体架构迁移到了面向服务的架构。

这样 Amazon 就建造了一个高度解耦的架构，这些服务可以相互独立地迭代，只要这些服务符合标准的网络服务接口。

『那时这种架构还没有名字，现在我们叫它微服务架构。』

每个团队都对一个或几个微服务有绝对的控制权，在 Amazon 这意味着：要和顾客对话（内部和外部的），定义自己的 feature roamap，设计并实现这些 feature，测试这些 feature，最后部署和运维这些 feature。

如果在整个过程中的任何地方出了问题，这个小团队有责任去修复它。任何人的偷懒或者犯错都要自己承担责任，甚至在半夜加班修复这些服务。因此工程师团队有极大的动力去保证整个产品周期的高速运转。

『那时对这种团队，业界还没有统一的名称，现在叫 DevOps。我们负责开发，测试，运维并将所有服务 merge 到一个工程团队中。』

在经过架构和组织的重大变革后，Amazon 极大地提高了前端开发的效率。产品团队可以很快地做决定，然后转化为微服务中的新 feature。**现在 Amazon 每年要进行 5000 万次部署，这都多亏了微服务架构和他们的持续交付流程。**