Atitit 微服务的优点和拆分

目录

[1. 微服务架构的好处四个方面的优点 简单 专业 扩展 语言无关 1](#_Toc20534)

[2. 微服务架构五大优势 崛起势头不可挡 4 2](#_Toc6758)

[2.1. 1、复杂度可控 6避免“盲人摸象” 7 3](#_Toc22761)

[2.2. 2、灵活可扩展 7 3](#_Toc18331)

[2.3. 3、独立部署 7 3](#_Toc14468)

[2.4. 4、开发针对性更强 7 3](#_Toc15294)

[2.5. 5、降低TCO 8 3](#_Toc18787)

[3. 微服务的优点 3](#_Toc20203)

[3.1. 微服务最大特点 独立部署 3](#_Toc5528)

[3.2. 增强稳定性，独立部署 3](#_Toc21426)

[3.3. 协调人力资源，使用不同的擅长的技术来实现不同的模块 4](#_Toc23346)

[3.4. 跨语言集成 提升效率 4](#_Toc22198)

[3.5. 性能负担 4](#_Toc26577)

[3.6. 拆分，有利于开发人员项目规模的轻量化，提升开发速度 4](#_Toc15481)

[3.7. 解耦：对于我们底层程序员而言，看得见的好处就是解耦 4](#_Toc11721)

[3.8. 提升启动速速 4](#_Toc13938)

[3.9. 提升扩展性 5](#_Toc18268)

[3.10. 利于持续性开发 5](#_Toc4571)

[4. 核心业务拆分　与　远程接口 6](#_Toc219)

[4.1. 拆分 6](#_Toc13781)

[4.2. 拆分粒度　　对于大部分项目来说，不到十 6](#_Toc2216)

[4.3. 按照不同的效率语言拆分不同的功能 6](#_Toc5083)

# ****微服务架构的好处****四个方面的优点 简单 专业 扩展 语言无关

[](http://s2.51cto.com/wyfs02/M01/06/32/wKiom1mzmwfi7KZpAABbFb65Ess69.jpeg)

我们总结了四个方面的优点，分别如下：

* 是每个微服务组件都是简单灵活的，能够独立部署。不再像以前一样，应用需要一个庞大的应用服务器来支撑。
* 可以由一个小团队负责更专注专业，相应的也就更高效可靠。
* 微服务之间是松耦合的，微服务内部是高内聚的，每个微服务很容易按需扩展。
* 微服务架构与语言工具无关，自由选择合适的语言和工具，高效的完成业务目标即可。

看到这里，大家会觉得微服务架构挺不错，然而还会有一些疑问，什么样的应用算是一个微服务架构的应用?该怎样设计一个微服务架构的应用?那我们来一起看看我们推荐的微服务应用的设计原则。

****三、微服务应用4个设计原则****

# 微服务架构五大优势 崛起势头不可挡 4

## 1、复杂度可控 6避免“盲人摸象” 7

## 2、灵活可扩展 7

## 3、独立部署 7

## 4、开发针对性更强 7

## 5、降低TCO 8

# 微服务的优点

## 微服务最大特点 独立部署

微服务特点的描述。

大概从以下四个方面来说：

* 根据业务模块划分服务种类。
* 每个服务可以独立部署并且互相隔离。
* 通过轻量的 API 调用服务。
* 服务需要保证良好的高可用性。

## 增强稳定性，独立部署

单体式应用另外一个问题是可靠性。因为所有模块都运行在一个进程中，任何一个模块中的一个bug，比如内存泄露，将会有可能弄垮整个进程。除此之外，因为所有应用实例都是唯一的，这个bug将会影响到整个应用的可靠性

## 协调人力资源，使用不同的擅长的技术来实现不同的模块

## 跨语言集成 提升效率

使得每个服务都可以有专门开发团队来开发。开发者可以自由选择开发技术，提供API服务。当然，许多公司试图避免混乱，只提供某些技术选择。然后，这种自由意味着开发者不需要被迫使用某项目开始时采用的过时技术，他们可以选择现在的技术。甚至于，因为服务都是相对简单，即使用现在技术重写以前代码也不是很困难的事情。

## 性能负担

## 拆分，有利于开发人员项目规模的轻量化，提升开发速度

## 解耦：对于我们底层程序员而言，看得见的好处就是解耦

。我要实现一个功能，可能并不需要很深入的了解别人的代码

## 提升启动速速

体式应用也会降低开发速度。应用越大，启动时间会越长。比如，最近的一个调查表明，有时候应用的启动时间居然超过了12分钟。我还听说某些应用需要40分钟启动时间。如果开发者需要经常重启应用，那么大部分时间就要在等待中渡过，生产效率受到极大影

## 提升扩展性

模块发生资源冲突时，扩展将会非常困难。比如，一个模块完成一个CPU敏感逻辑，应该部署在AWS EC2 Compute Optimized instances，而另外一个内存数据库模块更合适于EC2 Memory-optimized instances。然而，由于这些模块部署在一起，因此不得不在硬件选择上做一个妥协。

## 利于持续性开发

另外，复杂而巨大的单体式应用也不利于持续性开发。今天，SaaS应用常态就是每天会改变很多次，而这对于单体式应用模式非常困难。另外，这种变化带来的影响并没有很好的被理解，所以不得不做很多手工测试。那么接下来，持续部署也会很艰难。

1. 每个服务小，代码易于理解，启动更快，部署更快，开发更高效。   
     
   2. 每个服务独立部署，可以独立变更，无需协同，持续部署变得可行。   
   3. 每个服务可以独立进行X轴克隆和Z轴分区。独自选择适合自己资源需求的硬件。例如，CPU密集需求的和内存密集的组件，在整体架构中，必须部署在一起，但是微服务可以满足各自的优先需求来部署。   
     
   4. 可以团队分组，每个组开发独立开发部署一些小服务。   
     
   5. 故障隔离性。出问题的组件和服务，可能不会影响到全局。   
     
   6. 技术更新更容易，对每个服务可以独立选择语言和框架。   
     
   7. 如果新技术尝试失败，可以很快抛弃这个成果，不会对整个项目带来风险。

# 核心业务拆分　与　远程接口

## 拆分

## 拆分粒度　　对于大部分项目来说，不到十

微服务 优点

## 按照不同的效率语言拆分不同的功能

微服务笔记.html