

加微信：642945106 发送“赠送”领取赠送精品课程

三 发数字“2”获取众筹列表

下载APP



## 01 | 到底什么是微服务？

2018-08-23 胡忠想

从0开始学微服务

进入课程 >



讲述：胡忠想

时长 08:58 大小 4.12M



从谷歌的搜索指数来看，微服务的热度在进入 2017 年后突然爆发，国内各大会议和论坛的相关讨论也如雨后春笋般层出不穷，各大一线互联网公司也纷纷将这一技术引入并在实际业务中落地。

● 微服务  
搜索字词

+ 比较

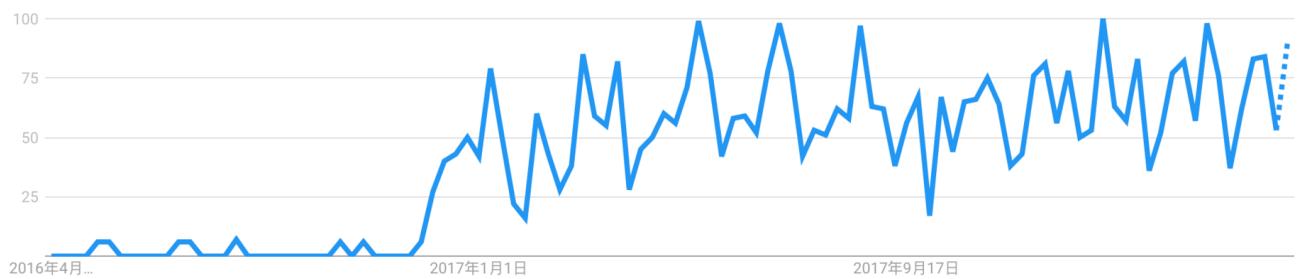
全球 ▾

2016/4/17 - 2018/5/17 ▾

所有类别 ▾

Google 网页搜索 ▾

热度随时间变化的趋势 [?](#)



然而据我所知，国内不少中小规模的技术团队对微服务的概念都不甚了解，对该不该引入微服务也不置可否。还有一些技术团队，没有考虑实际业务场景，只是为了追求技术热点，盲目引入微服务，但又缺乏相应的技术掌控能力，最后影响了业务的稳定性。

对于该不该引入微服务，以及微服务体系需要哪些技术，目前并没有适合中小团队的架构实践落地的指引。因此我结合自己在微博多年的业务实践，总结出了一套微服务落地经验，从基础理论到架构实践，再结合业界最新趋势分析，希望能帮助中小规模团队了解微服务的本质以及对业务的价值，从而做出正确的判断。

我们先来看看维基百科是如何定义微服务的。微服务的概念最早是在 2014 年由 Martin Fowler 和 James Lewis 共同提出，他们定义了微服务是由单一应用程序构成的小服务，拥有自己的进程与轻量化处理，服务依业务功能设计，以全自动的方式部署，与其他服务使用 HTTP API 通讯。同时，服务会使用最小规模的集中管理（例如 Docker）技术，服务可以用不同的编程语言与数据库等。

这个理论的定义看着有点晕？没关系，接下来[我来帮你理解到底什么是微服务？](#)

## 单体应用

在开聊微服务之前，我先要你和介绍下单体应用。如果你不知道单体应用的痛，那也不会深刻理解微服务的价值。

早些年，各大互联网公司的应用技术栈大致可分为 LAMP ( Linux + Apache + MySQL + PHP ) 和 MVC ( Spring + iBatis/Hibernate + Tomcat ) 两大流派。无论是 LAMP 还是 MVC，都是为单体应用架构设计的，其优点是学习成本低，开发上手快，测试、部署、运维也比较方便，甚至一个人就可以完成一个网站的开发与部署。

以 MVC 架构为例，业务通常是通过部署一个 WAR 包到 Tomcat 中，然后启动 Tomcat，监听某个端口即可对外提供服务。早期在业务规模不大、开发团队人员规模较小的时候，采用单体应用架构，团队的开发和运维成本都可控。

然而随着业务规模的不断扩大，团队开发人员的不断扩张，单体应用架构就会开始出现问题。我估计经历过业务和团队快速增长的同学都会对此深有感触。从我的角度来看，大概会有以下几个方面的问题。

**部署效率低下。** 以我实际参与的项目为例，当单体应用的代码越来越多，依赖的资源越来越多时，应用编译打包、部署测试一次，甚至需要 10 分钟以上。这也经常被新加入的同学吐槽说，部署测试一次的时间，都可以去楼下喝杯咖啡了。

**团队协作开发成本高。** 以我的经验，早期在团队开发人员只有两三个人的时候，协作修改代码，最后合并到同一个 master 分支，然后打包部署，尚且可控。但是一旦团队人员扩张，超过 5 人修改代码，然后一起打包部署，测试阶段只要有一块功能有问题，就得重新编译打包部署，然后重新预览测试，所有相关的开发人员又都得参与其中，效率低下，开发成本极高。

**系统高可用性差。** 因为所有的功能开发最后都部署到同一个 WAR 包里，运行在同一个 Tomcat 进程之中，一旦某一功能涉及的代码或者资源有问题，那就会影响整个 WAR 包中部署的功能。比如我经常遇到的一个问题，某段代码不断在内存中创建大对象，并且没有回收，部署到线上运行一段时间后，就会造成 JVM 内存泄露，异常退出，那么部署在同一个 JVM 进程中的所有服务都不可用，后果十分严重。

**线上发布变慢。** 特别是对于 Java 应用来说，一旦代码膨胀，服务启动的时间就会变长，有些甚至超过 10 分钟以上，如果机器规模超过 100 台以上，假设每次发布的步长为 10%，单次发布需要就需要 100 分钟之久。因此，急需一种方法能够将应用的不同模块的解耦，降低开发和部署成本。

想要解决上面这些问题，**服务化**的思想也就应运而生。

## 什么是服务化？

这里我就不谈一些官方的、教条主义的概念了。在我看来，用通俗的话来讲，服务化就是把传统的单机应用中通过 JAR 包依赖产生的本地方法调用，改造成通过 RPC 接口产生的远程方法调用。一般在编写业务代码时，对于一些通用的业务逻辑，我会尽力把它抽象并独立成为专门的模块，因为这对于代码复用和业务理解都大有裨益。

在过去的项目经历里，我对此深有体会。以微博系统为例，微博既包含了内容模块，也包含了消息模块和用户模块等。其中消息模块依赖内容模块，消息模块和内容模块又都依赖用户模块。当这三个模块的代码耦合在一起，应用启动时，需要同时去加载每个模块的代码并连接对应的资源。一旦任何模块的代码出现 bug，或者依赖的资源出现问题，整个单体应用都会受到影响。

为此，首先可以把用户模块从单体应用中拆分出来，独立成一个服务部署，以 RPC 接口的形式对外提供服务。微博和消息模块调用用户接口，就从进程内的调用变成远程 RPC 调用。这样，用户模块就可以独立开发、测试、上线和运维，可以交由专门的团队来做，与主模块不耦合。进一步的可以再把消息模块也拆分出来作为独立的模块，交由专门的团队来开发和维护。

可见通过服务化，可以解决单体应用膨胀、团队开发耦合度高、协作效率低下的问题。

## 什么是微服务？

从 2014 年开始，得益于以 Docker 为代表的容器化技术的成熟以及 DevOps 文化的兴起，服务化的思想进一步演化，演变为今天我们所熟知的微服务。

那么微服务相比于服务化又有什么不同呢？

在我看来，可以总结为以下四点：

**服务拆分粒度更细。**微服务可以说是更细维度的服务化，小到一个子模块，只要该模块依赖的资源与其他模块都没有关系，那么就可以拆分为一个微服务。

**服务独立部署。**每个微服务都严格遵循独立打包部署的准则，互不影响。比如一台物理机上可以部署多个 Docker 实例，每个 Docker 实例可以部署一个微服务的代码。

**服务独立维护。**每个微服务都可以交由一个小团队甚至个人来开发、测试、发布和运维，并对整个生命周期负责。

**服务治理能力要求高。**因为拆分为微服务之后，服务的数量变多，因此需要有统一的服务治理平台，来对各个服务进行管理。

继续以前面举的微博系统为例，可以进一步对内容模块的功能进行拆分，比如内容模块又包含了 feed 模块、评论模块和个人页模块。通过微服务化，将这三个模块变成三个独立的服务，每个服务依赖各自的资源，并独立部署在不同的服务池中，可以由不同的开发人员进行维护。当评论服务需求变更时，只需要修改评论业务相关的代码，并独立上线发布；而 feed 服务和个人页服务不需要变更，也不会受到发布可能带来的变更影响。

由此可见，微服务化给服务的发布和部署，以及服务的保障带来了诸多好处。

## 总结

今天，我介绍了微服务的发展由来，它是由单体应用进化到服务化拆分部署，后期随着移动互联网规模的不断扩大，敏捷开发、持续交付、DevOps 理论的发展和实践，以及基于 Docker 容器化技术的成熟，微服务架构开始流行，逐渐成为应用架构的未来演进方向。

总结来说，微服务架构是将复杂臃肿的单体应用进行细粒度的服务化拆分，每个拆分出来的服务各自独立打包部署，并交由小团队进行开发和运维，从而极大地提高了应用交付的效率，并被各大互联网公司所普遍采用。

## 思考题

你在业务开发中是否也遇到过因单体应用过度膨胀所带来的问题呢？你觉得针对这些问题微服务能解决吗？

欢迎你在留言区写下自己的思考，与我一起讨论。



# 从 0 开始学微服务

微博服务化专家的一线实战经验

胡忠想 微博技术专家



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金奖励**。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 [开篇词 | 微服务，从放弃到入门](#)

下一篇 [02 | 从单体应用走向服务化](#)

## 精选留言 (91)

写留言



Colin

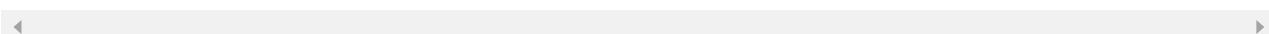
2018-08-23

90

服务化最难的是在数据库层，因为数据库中很多数据都是相互关联的，比如用户跟订单，订单和商品等等这些数据之间都是有关联的，服务拆分之后会面临以下问题：1当需要读取关联数据时，如果采用表连接的方式查询数据会出现跨数据库查询的可能，2如果是通过RPC的方式多次调用(比如要查订单，就需要查询商品及订单详细信息)，也会出现多次调用导致的频繁多次的数据库连接，而如果使用缓存，也会面临数据库与缓存之间的数据...

展开 ▼

作者回复：是的，服务化拆分最好不要涉及跨数据库表交互





少帅

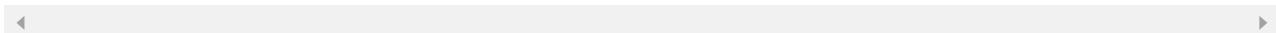
2018-08-23

37

看到留言里面好多正在经历从单体到微服务的拆分过程，拆分微服务先要把业务梳理清楚，做到心中有数，梳理清楚了那么业务的边界自然清晰了，自然而然对应拆分哪些服务也出来了，拆分的过程中一定是绞杀式的，新的需求自然放到拆分的微服务，老的逻辑按照优先级和重要程度一个点一个点的从单体迁移微服务，服务化上线之后，渐渐取代老的单体，等全部拆分完毕，线上稳定之后，自然就可以下掉老的单体应用

展开 ▼

作者回复: 实践出真知凸



Neo\_PJ

2018-08-23

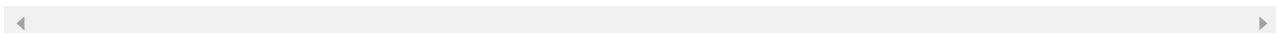
36

现在的系统代码量50w，单体应用，但是拆分成微服务太麻烦了，领导不敢动，系统是金融管理基金托管计算的，太多复杂场景耦合，10几年的代码，好几个团队的代码堆积，俗话就是 又臭又硬的系统，只能不断维护，不敢重新搭建新的框架，不敢拆，拆不得，拆了谁负责，谁敢保证拆后所有功能按照原有一样逻辑呢？

...

展开 ▼

作者回复: 不好整...



Eric\_Zhan...

2018-08-23

17

看到老师提到微服务粒度，看到一个面试题是这样说的：微服务划分的粒度？这个问题，该从什么方面回答？

展开 ▼



YC

2018-08-23

16

看到有人说dao层要不要抽出来作为微服务，我觉得没必要，每个小应用都应该有自己独立的数据库，如果dao抽出来，意味着其他服务也可以调用这个dao，那就会造成数据库数据的混乱。最好是在dao层上面封装service层，提供给其他应用使用，这样能保证数据的读写可控

展开 ▼

---



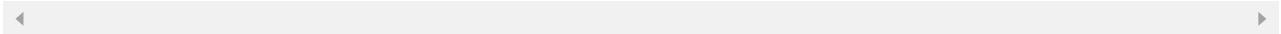
钟悠

2018-08-24

14

在我看来，用通俗的话来讲，服务化就是把传统的单机应用中通过 JAR 包依赖产生的本地方法调用，改造成通过 RPC 接口产生的远程方法调用。这句话真的讲的很透彻凸

作者回复: 实践出真知



云学

2018-08-23

13

这篇文章内容比较实在，“进程内调用改成Rpc调用”，一语直达本质，期待更多干货



WolvesLead...

2018-08-23

12

老师，问一下，一般和数据库交互的dao层，是单独拿出来做成服务好还是不做好，还有一些公共的模块是不是也要做成服务呢？谢谢啦

展开 ▼

作者回复: 我理解你这个需求中间件的需求



日拱一卒

2018-08-25

10

看了文章和留言，讲的很清楚，请教一个问题，之前也有同学问了。

对于典型的复杂web应用，当由单体应用改造成微服务时，一般会按照业务进行服务划分，彼此独立的业务拆分到不同的微服务中。但是在数据层面，涉及到下面两个问题：

1. 数据库是否拆分？从数据架构的角度看，可能有三个选择：1) 一个db实例，所有微服务...

展开 ▼

作者回复: 第一个问题：数据库拆分其实不是微服务拆分的关键，如果会引入分布式事务的话，最好不要拆分。

第二个问题：长远来看，建议方案2，这样会减少每个微服务同db的连接，在集群规模上千台的情况下，db的连接数不会成为瓶颈，服务启动也会变快，因为初始化连接也会变少。



**Amorfati**

2018-08-23

10

如何确定是过度膨胀也是比较难的问题，在膨胀前切分很有可能随着业务的发展，发现切错了，然后好尴尬，按照我的理解是，单体应用中已经经过足够时间业务上的检验，将其中模式比较稳定，整体高内聚的业务进行抽离。如有错误，还请老师指正。

展开 ▾

作者回复: 对



**null**

2018-08-23

7

5年的php单体应用，整个项目结构复杂混乱，php前端和php后端各种函数调用。业务逻辑代码也穿插着大量已过期或无关的代码，项目可维护性越来越低。每次需求开发完毕合并分支时，需要花费大量的精力去解决代码冲突。开发的速度远跟不上需求的速度。

在去年所有新的需求使用java进行开发，同时一点一点将php的业务迁移到java。...

展开 ▾

作者回复: 第一点，可以抽离出哪些是公共的模块，并且请求量和重要性值得这么做。

第二点，接口定义最好不要随便变更，这是原则，可以添加参数，新增返回字段，也可以新增接口，但最好不要改参数名或者已有字段。



**CC**

2018-08-23

5

我的理解，微服务的重点在于拆分治理，spring boot，容器，这些只是实现拆分的一个工具而已，如果你想，使用传统的tomcat启动发布也是可以实现这样的目标的。

作者回复: 是的，微服务架构的核心在于服务治理，而容器是对运维方式的改变，通过容器标准化发布部署。



常玉棋

2018-08-23

1 5

子系统越来越多，重复造轮子的情况也越来越频繁，但我以为小团队还是不要轻易采用微服务方案，毕竟，创业初期人员有限，应该以项目目标实现为第一要务。

展开 ▼



jlogin

2018-08-25

1 4

现在好多公司强推微服务，不考虑实际情况，单体应用并不是就一定很脆弱，系统不稳定，服务拆分了也不一定稳定，开发效率不高，没准是流程问题，系统本身设计问题耦合严重，单体应用设计不好，玩不转，服务拆分后更乱。

...

展开 ▼

作者回复: �恩，微服务架构的技术门槛非常高，没有充裕的技术人员不要轻易入坑



WestonLee

2018-08-23

1 4

我们目前的系统就是从单体到服务化再到微服务化的，第一次转变是为了减少大计算量的模块带来的阻塞，提高负载，第二次转变是因为系统已经比较庞大，需要切分的更细，并且便于开发维护。目前用了spring cloud和docker，遇到的最大的问题是数据库伴随服务的切分以及一致性带来的复杂度提升

展开 ▼

作者回复: 微服务有利有弊，总体来看，利大于弊



瘦是为了帅...

2018-08-25

1 3

以前都是接触的单体小项目，还没有遇到膨胀到不行的，目前在这里新公司，用到了微服务！多学习，多了解

展开 ▼

---



70

2018-08-24

冂 3

公司现在的系统采用了按功能进行划分，不同功能的模块，不同Service，不同Service之间依靠rpc进行通信，如果其他服务需要访问的不属于他的数据库，需要对应的Service提供出服务进行访问，这样会有一个问题，本来需要事物执行的逻辑可能会被打散到多个Service执行，那要不要在事物中包含Rpc调用，我的理解是不行的，这样可能会导致数据库事物卡着，那行怎么进行保证一致性喃？通过异步的去确认吗？

展开 ▼

---



青青木

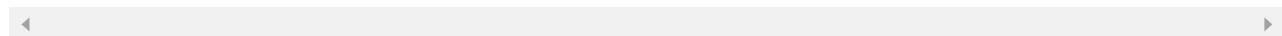
2018-08-23

冂 3

老师好，我所在的公司正在进行微服务化的系统改造，原有的系统是一个巨大的单体架构业务系统。在改造过程中我的困惑是微服务的拆分粒度 差的大了感觉效果不好，差的细了业务调用链太长 造成系统响应变慢 问题定位不容易等问题，老师您对微服务的业务划分有什么建议吗？谢谢

展开 ▼

作者回复: 下一节会讨论



lwt

2018-08-23

冂 3

一个应用包含太多功能，每次改动一块的功能，测试需要整体测试，周期很长。  
同一时期很多项目都需要去改同一个应用，一个项目上线后，其他项目的需要合并这个项目的代码，然后整体回归测试。  
因为应该集成的功能很多，新人很难在短时间内熟悉整个应用。

展开 ▼

---



哈迪斯

2018-08-27

冂 2

“用通俗的话来讲，服务化就是把传统的单机应用中通过 JAR 包依赖产生的本地方法调用，改造成通过 RPC 接口产生的远程方法调用。” 直击本质

展开 ▼

