微服务就是把一个应用程序，拆分为多个模块，每个模块可以单独运行，所有模块可以共同组成一个完成的应用服务

“微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相互协调、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，服务和服务之间采用轻量级的通信机制相互沟通（通常是基于HTTP的Restful API).每个服务都围绕着具体的业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应尽量避免统一的、集中的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构"---- Martin Fowler的博客



**优点**

1. 提升开发交流，每个服务足够内聚，足够小，代码容易理解；
2. 服务独立测试、部署、升级、发布；
3. 按需定制的DFX，资源利用率，每个服务可以各自进行x扩展和z扩展，而且，每个服务可以根据自己的需要部署到合适的硬件服务器上；每个服务按
4. 需要选择HA的模式，选择接受服务的实例个数；
5. 容易扩大开发团队，可以针对每个服务（service）组件开发团队；
6. 提高容错性（fault isolation），一个服务的内存泄露并不会让整个系统瘫痪；
7. 新技术的应用，系统不会被长期限制在某个技术栈上；

**缺点**

1. 没有银弹，微服务提高了系统的复杂度；
2. 开发人员要处理分布式系统的复杂性；
3. 服务之间的分布式通信问题；
4. 服务的注册与发现问题；
5. 服务之间的分布式事务问题；
6. 数据隔离再来的报表处理问题；
7. 服务之间的分布式一致性问题；
8. 服务管理的复杂性，服务的编排；
9. 不同服务实例的管理。

==================================================================

作者：铁原  
链接：https://www.zhihu.com/question/65502802/answer/741894748  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

上面答案各种长，堆砌各种名词形容词，敢拿效果出来说事么？

比如51是国内最早做微服务的，架构师出来走两步。

华为可以说用了微服务以后，虽然IPD很慢，但是开发效率提升了3倍……

阿里可以分享一下微服务后福报减少了多少……

业界我看了这么多年，做微服务没几个好死的。李笑来说啥：一个人说做区块链的，我们是朋友。你要是去做了，我没你这样的朋友。

懂这个意思么——微服务本身理念是好的，营销性质大于技术本身。就好像政治家们谈世界和平一样，忽悠SB用的。

////////////////////////////////

为什么会有微服务：简单讲是**吃饱了（过度SOA）**撑的。

IT系统在本世纪初的时候，还很稚嫩，会个静态html就已经算个高手，如果再会个动态语言，譬如PHP，JSP，那基本算是个IT专家。

中国当时的IT技术非常之稚嫩。上海当年地铁刷卡系统招标的时候，国外公司要价2亿。收单系统很复杂么，现在看起来很简单，当时中国人就是不会。08年的时候我初涉银行系统，当时银行核心系统几乎清一色的国外系统，核心都是C，java都是不敢用，现在看起来是个事么？。。。。。 就是这么稚嫩——虽然现在金融IT烂大街了,街上抓几个程序员就敢搞个金融系统出来，大数据，人工智能满地走。

这10多年来，中国IT技术发展的非常迅猛。

当时技术是瓶颈，但是现在都不是个事，现在是技术过剩，各种技术乱用滥用，技术不是问题，问题是如何使用技术。比如SOA的滥用就导致了系统的碎片化，数量膨胀动辄成百上千。

这个时候，有些公司清醒的架构师们提出了系统合并。14年左右的时候，支付宝架构组鲁肃等人是强烈推这个事的，记得当时有论断说：全支付宝的系统20个以内是绝对够用的，现在3000个左右，要合。

这事最初怎么起的呢？

10年初的时候，淘宝/支付宝和系统还是单体系统。随着业务量的迅猛增长，他们急需技术上对此进行支撑。

08-10年初的时候，是SOA疯狂盛行的年代——嗯，当时还盛行webservice技术，以及osgi，RCP，RIA、AJAX等很多技术。当时的SOA和我们现在的SOA显然不是一个概念。

SOA当时最火的是ESB这种模式，DUBBO这种P2P模式的在当时显然是不够高大上。ESB这种模式显然具有极大的魅惑性。它避免业务系统之间陷入迷宫般的调用拓扑，整个业务架构围绕在ESB总线旁边，条理清晰；所以业务系统之间无需处理其他系统的协议、报文等差异性，只需要关注自己的核心业务，ESB总线能帮业务系统处理各种通讯协议，报文协议的差异性。

但是现在有哪个人见过ESB系统呢？

历史终究选择了Dubbo这种模式的SOA，而不是ESB这种模式

任何一个技术发展的早期，都不能避免各种试错——不同的是，有些技术方向就是错误的，有些仅仅是道路是曲折的。比如OSGI、FLASH、RIA等各种溢美之词可能不逊于今天的微服务吧。可惜这些在历史的发展中几乎没有留什么痕迹。甚至今天我们熟悉的这些技术，其实跟当时的他们本来的面目也是截然不同，比如WEBSERVIE,SOA……可能他们的拥趸们也羞于提及这些事情吧

我总是为这种错误发展轨迹的“不留痕迹”感觉到很可惜。我们今天的人去解决今天的问题的话，过去的错误轨迹其实比成果，对我们的更有帮助。

SOA的曲折路程不仅仅是ESB，还有更大的弊病。  
淘宝、阿里、支付宝、……所有这些推行SOA的公司进展是非常迅速的，仅仅在1-2年时间就完成了系统的SOA化，系统迅速膨胀到了几百几千个（小公司的大部分在1000-，部分有3000，,大公司多数在5000-）

然后就发现随便做个事情，都关联到几十个系统，不说开发光协调工作都非常费劲。开发起来联调、部署、日志查询等更是困难重重。

这个时候，有些公司清醒的架构师们提出了系统合并。14年左右的时候，支付宝架构组鲁肃等人是强烈推这个事的，记得当时有论断说：全支付宝的系统20个以内是绝对够用的，现在3000个左右，要合。

不过这个事的进展一直很不顺利，这和SOA对的迅速化对比是非常鲜明的。

后来的事情大家其实都知道了，马云爸爸去supercell逛了以后回来以后说：我们过去创业的时候几个人几个月就弄个淘宝出来了，现在随便一点小事都几百号人半年看不见个水花。于是有了中台。

/////////////////////////////////////////////////////

以上是时代大背景。业务大发展，技术快速粗暴跟进，造成很多后遗症，然后SOA臭了。

下面是理论分析

/////////////////////////////////////////////////////

要考虑复杂度。

当业务复杂度提升的时候，团队必然存在一个无法cover的极限。势必需要拆分。

复杂系统通过拆分将复杂性分离为很多小模块的的办法。形式上不可避免只有7种形态：代码行，函数，对象，本地模块（项目中具有高度隔离性的模块），外置模块（dom4j，slf4j），SOA，Richclient。 这几种形态的复杂度依次增加。

拆分的问题在于会提升复杂度表现形态，增加总体复杂度。

比如一个业务新系统的业务复杂度都在一个系统内表示，都是用API或对象表示的话，拆分就可能让这个业务复杂度用系统来表现。这个表现形式本身的复杂度是提升了。整体上复杂度（ = 业务复杂度+技术复杂度）肯定是提升的。联调两个系统当然比两条连个对象复杂N倍。依此类推

需要注意

拆分的越多，总体复杂度增加的越多。  
表现形式提升的越高，总体复杂度增加的越多。

SOA显然就是犯了这个忌讳，从单体系统到拆分的时候，度没有掌握好，放任型的拆分，拆分粒度过细，整体复杂度提升过高。

解决这个问题，显然靠更细粒度的拆分，显然是吃错了药。

譬如现在大部分微服务的拥趸们干的事情——其中有部分拥趸认为SOA是不好的，因为不够微。所以他们居然做到了一个API一个系统。——他们如今的过的十分酸爽。

//////////////////////////////////////////

下面是又回到了不够理性的，感性的去分析微服务如今处境的文字

//////////////////////////////////////////

14年以后，显然SOA已经不够高大上了，弊病就燃烧在眉头。

这个时候微服务出现了，如同任何一个新技术一样，它和SOA早期一样针对让大家疼怕了的种种弊病（任意个技术早期的宣传套路）。不过如同SOA早期 一样，它自己面目是什么却是模糊不清。

SOA号称自己是更微更轻量级，独立部署，restful……

单单这些形容词看起来都没什么问题。但是内部却有很深层次的矛盾。

如果它是通过拆分的方式解决问题的话，显然它和SOA会是一个路子——但显然SOA已经臭了——微服务自然极力要和SOA撇清关系。很多微服务的拥趸用restful等细枝末节的差异来说明自己和SOA的差异——当然还有很多其他理由，但是也仅仅是理由，目的只有一个，和已经臭的SOA撇清关系。

但是关键问题它不能交代自己到底是什么。

理论上，模块只有7中表现形态  
代码行，函数，对象，本地模块，外置模块是逻辑复杂性全部在本地的东西   
Soa是一种本地什么逻辑都没有的，复杂性全部在远端的东西。   
Richclient是一种混合模式。

微服务不可能是代码行，函数，对象，本地模块，外置模块。它又否认自己是是SOA，那么它只可能是Richclient。

但是，如果他是Richclient的话，这违反他自己的定义“能够被独立的部署到生产环境、类生产环境等”，“每个服务运行在其独立的进程中”。 但是如果符合这个标准，显然就是SOA，并且用这个标准去看SpringBoot提供的几大微服务神器，没一个符合的。全是符合Richclient的模式。

某种程度上，我个人认为微服务就是Richclient。

如果按照马丁的这个定义的话，它其实就是SOA。不过这显然会让拥趸们心理难以接受，于是自然要拼命找和SOA的不同，不过似乎也找不到什么确切的依据，于是大多数微服务的拥趸们有两条路  
1，抱大腿，拼命和赵老爷子攀亲戚。多使用市面上所谓的微服务框架SpringBoot，Docker，或者市面上的大厂谁号称微服务，谁的营销热闹，谁的名气大，谁的嗓门大…

2，原教旨主义：抓住教义的原始教文——微，restful等，但是什么才叫微呢？拆成模块还不够，一个api一个系统才够极致。

那么正确的“修正主义”微服务应该是什么呢？

1，相比过去碎片化的拆分，他应该是一种更克制的拆分  
2，相比纯粹的独立部署的性能更差的SOA，他更拥抱本地嵌入本地调用

遵守几个原则

1，宁简不繁  
2，宁胖不瘦  
3，宁少不多

宁简不繁：业务逻辑复杂度一定的情况下，选取最简单的模块形态。能用api不用函数，能用模块不用SOA。哪些将API表达为SOA的同学们注意了。

宁胖不瘦：同样的组件形态，它要有更丰富的业务表达能力。这样后续业务扩展，各种变化就会非常从容。

宁少不多：组件的总体数量尽可能少。对单个系统而言，理论上少的极限是1，但是N个系统整体看，1就未必合适，彼此的粒度未必合适，要取一个整体的所有系统的最小值。理论上这实际上会导致领域驱动的组件。

满足这三条，即可保证：当前业务复杂度总量一定的前提下，我们的实现是花费最小代价，满足了最高的业务扩展性，最好的性能，运维成本……

这就是所谓的修正主义微服务。

//////////////////////

不是任何时候使用微服务都是正确的。

微服务是一种应对复杂性的产物，只有当问题复杂到一定程度的时候，他才是有益的。相比简单的问题，它本身就是复杂度过剩的。是不值得的。

市面上90%的所谓微服务，都是扯淡。都是在复杂化问题或者在顺势卖东西。

当前的技术方案，前一阶段的技术债都是存在的，正确的微服务微更应该是上一个阶段SOA的优化，是一种更节制、更加拥抱技术深层次内核——概念/领域问题/复杂性问题等产物。

简单总结：复杂性问题需要模块化来解决复杂性控制，拆分本身会引入额外复杂性。过去拆过头了，修正主义微服务是一种更低复杂度的拆分方案。最低的复杂性引入当然是不拆分——如果复杂性没大到不能控制就不应该选择他。

////////////////题外话，为什么历史选择了Dubbo这种模式的SOA

前文我们讲了，各个公司早期是非常倾向于ESB模式的。这种模式真正可以让业务系统开发简单，不过他有两个弊病  
1，ESB中间件有单点问题。架构上有风险  
2，最关键的，ESB中间件开发门槛高。 dubbo这样的RPC框架其实还是比较简单的（一个工作三年的程序员在指导下1-2周就能开发出一个简单能用的RPC框架，从使用上一般看不出和Dubbo的区别，但是工业级的稳定性，扩展性等则需要非常长时间、使用量和人力积累）。ESB则没那么容易。

在本世纪初的10年，bat等IT公司业务快速发展，中国技术和技术大咖们显然还比较青涩，当然是先保证能先跑起来，而更少考虑这些高大上的东西。

、、、、、、、、、、、

本质上，软件开发是工程性质的，而不是科学性质的。他没有什么真正的科学问题要突破，他不过把原来人肉完成的东西换成Object来完成。从业务的角度来看，一个10年经验开发的架构完美的软件和一个1年经验开发的能run的没什么区别。

最初中国的软件开发更多的是业务追求，业务导向。在这种初级追求阶段，技术追求更多的是比较low的高并发，性能等指标——这些指标老板们也能理解。

你要老板们理解扩展性、抽象，架构扩展性，显然超出他们的理解范畴了。也超出这个阶段IT人才的理解范畴了，因为大家没见过这个东西。

但是当It公司的业务复杂度到一定阶段，按起葫芦起了瓢以后，现实就完成了对老板和工程师的初级教育 ，这个时候非常合适考虑架构设计，业务架构，灵活性等问题。

这当然是对的。在初级阶段考虑高级阶段的问题显然也并不明智。但是到了高级阶段还是草莽心态，马上得天下马上治之显然也是不行。

简而言之， 不论是初级阶段，还是高级阶段，始终都是以业务为目标。是目标清晰的，有些技术，譬如OSGI等 ，本身是有点过于技术show的。整体上来看他们是在复杂化问题而不是简化问题的。——当然，任何技术最初都是只说自己好的一面，回避自己不好的一面，OSGI们也一定会给自己很多貌似很好的外衣……。每一个一个清醒的架构师、技术人员在学习、使用一个新技术的时候，都要绷紧这根弦。另外，看清楚自己和技术的位置，合适的使用它，并不是好的任何情况下都是好的。这是纯粹技术层面的分析。

在纯粹的技术层面，无论是ESB，还是SOA，还是OSGI，还是……，技术本身是没有什么高下之分的，他们只是一个客观，受到哲学原理的制约，尺有所短寸有所长。但是使用人的人的不正确的使用，比如过度的SOA，显然最后背锅的一定是不会说话的那个，比如SOA技术本身。而那些从前的技术导师，布道师们显然会毫不犹豫拥抱“更先进的技术”。好比那个行业的演变，显然一群莺们需要一个比他们身份更高大上的title，于是群莺们就开始糟蹋小姐这个词，然后小姐这个词也必然烂，他们必然再回去找一个新的枝头。显然错不在词语本身啊。跟这群人玩，还以为自己玩的是纯粹的技术？？怎么可能，这里都是玩人的。

IT圈现在显然也是资本推崇的行业。钱多的地方是非就多，不分行业。显然现在的技术圈乱像比其他行业要厉害的多，不过被假以技术之名神圣化了……古人说圣人之名之下最容易那啥？

技术是纯粹的，但是技术后面的人是不纯粹的。

技术不会自己张嘴说话