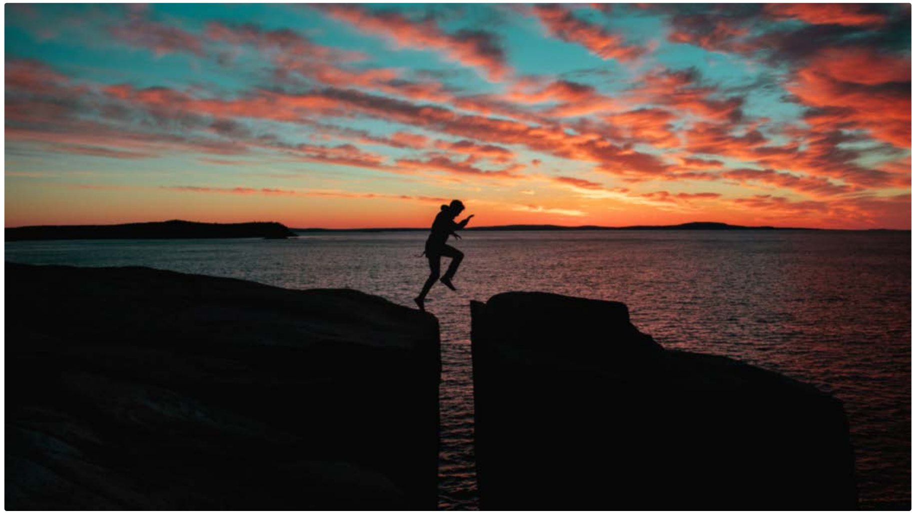
34 | 深入理解微服务架构：银弹 or 焦油坑？

2018-07-14 李运华



34 | 深入理解微服务架构：银弹 or 焦油坑？
李运华

微服务是近几年非常火热的架构设计理念，大部分人认为是Martin Fowler提出了微服务概念，但事实上微服务概念的历史要早得多，也不是Martin Fowler创造出来的，Martin只是将微服务进行了系统的阐述（原文链接：https://martinfowler.com/articles/microservices.html）。不过不能否认Martin在推动微服务起到的作用，微服务能火，Martin功不可没。

微服务的定义相信你早已耳熟能详，参考维基百科，我就来简单梳理一下微服务的历史吧（https://en.wikipedia.org/wiki/Microservices#History）：

- 2005年：Dr. Peter Rodgers 在Web Services Edge大会上提出了“Micro-Web-Services”的概念。
- 2011年：一个软件架构工作组使用了“microservice”一词来描述一种架构模式。
- 2012年：同样是这个架构工作组，正式确定用“microservice”来代表这种架构。
- 2012年：ThoughtWorks的James Lewis针对微服务概念在QCon San Francisco 2012发表了演讲。
- 2014年：James Lewis和Martin Fowler合写了关于微服务的一篇学术性的文章，详细阐述了微服务。

由于微服务的理念中也包含了“服务”的概念，而SOA中也有“服务”的概念，我们自然而然地会提出疑问：**微服务与SOA有什么关系？有什么区别？为何有了SOA还要提微服务**？这几个问题是理解微服务的关键，否则如果只是跟风拿来就用，既不会用，也用不好，用了不但没有效果，反而还可能有副作用。

今天我们就来深入理解微服务，到底是银弹还是焦油坑。

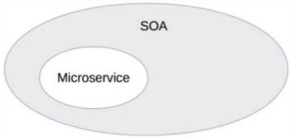
微服务与SOA的关系

对于了解过SOA的人来说，第一次看到微服务这个概念肯定会有所疑惑：为何有了SOA还要提微服务呢？等到简单看完微服务的介绍后，可能很多人更困惑了：这不就是SOA吗？

关于SOA和微服务的关系和区别，大概分为下面几个典型的观点。

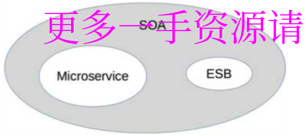
- 微服务是SOA的实现方式

如下图所示，这种观点认为SOA是一种架构理念，而微服务是SOA理念的一种具体实现方法。例如，“微服务就是使用HTTP RESTful协议来实现ESB的SOA”“使用SOA来构建单个系统就是微服务”和“微服务就是更细粒度的SOA”。



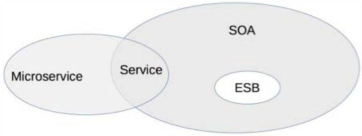
- 微服务是去掉ESB后的SOA

如下图所示，这种观点认为传统SOA架构最广为人诟病的就是庞大、复杂、低效的ESB，因此将ESB去掉，改为轻量级的HTTP实现，就是微服务。



- 微服务是一种和SOA相似但本质上不同的架构理念

如下图所示，这种观点认为微服务和SOA只是有点类似，但本质上是不同的架构设计理念。相似点在于下图中交叉的地方，就是两者都关注“服务”，都是通过服务的拆分来解决可扩展性问题。本质上不同的地方在于几个核心理念的差异：是否有ESB、服务的粒度、架构设计的目标等。



以上观点看似都有一定的道理，但都有点差别，到底哪个才是准确的呢？单纯从概念上是难以分辨的，我来对比一下SOA和微服务的一些具体做法，再来看看到底哪一种观点更加符合实际情况。

1. 服务粒度

整体上来说，SOA的服务粒度要粗一些，而微服务的服务粒度要细一些。例如，对一个大型企业来说，“员工管理系统”就是一个SOA架构中的服务；而如果采用微服务架构，则“员工管理系统”会被拆分为更多的服务，比如“员工信息管理”“员工考勤管理”“员工假期管理”和“员工福利管理”等更多服务。

2. 服务通信

SOA采用了ESB作为服务间通信的关键组件，负责服务定义、服务路由、消息转换、消息传递，总体上是重量级的实现。微服务推荐使用统一的协议和格式，例如，RESTful协议、RPC协议，无须ESB这样的重量级实现。Martin Fowler将微服务架构的服务通讯理念称为“Smart endpoints and dumb pipes”，简单翻译为“聪明的终端，愚蠢的管道”。之所以用“愚蠢”二字，其实就是与ESB对比的，因为ESB太强大了，既知道每个服务的协议类型（例如，是RMI还是HTTP），又知道每个服务的数据类型（例如，是XML还是JSON），还知道每个数据的格式（例如，是2017-01-01还是01/01/2017），而微服务的“dumb pipes”仅仅做消息传递，对消息格式和内容一无所知。

3. 服务交付

SOA对服务的交付并没有特殊要求，因为SOA更多考虑的是兼容已有的系统；微服务的架构理念要求“快速交付”，相应地要求采取自动化测试、持续集成、自动化部署等敏捷开发相关的最佳实践。如果没有这些基础能力支撑，微服务规模一旦变大（例如，超过20个微服务），整体就难以达到快速交付的要求，这也是很多企业在实行微服务时踩过的一个明显的坑，就是系统拆分为微服务后，部署的成本呈指数上升。

4. 应用场景

SOA更加适合于庞大、复杂、异构的企业级系统，这也是SOA诞生的背景。这类系统的典型特征就是很多系统已经发展多年，采用不同的企业级技术，有的是内部开发的，有的是外部购买的，无法完全推倒重来或者进行大规模的优化和重构。因为成本和影响太大，只能采用兼容的方式进行处理，而承担兼容任务的就是ESB。

微服务更加适合于快速、轻量级、基于Web的互联网系统，这类系统业务变化快，需要快速尝试、快速交付；同时基本都是基于Web，虽然开发技术可能差异很大（例如，Java、C++、.NET等），但对外接口基本都是提供HTTP RESTful风格的接口，无须考虑在接口层进行类似SOA的ESB那样的处理。

综合上述分析，我将SOA和微服务对比如下：

对比维度	SOA	微服务
服务粒度	粗	细
服务通信	重量级，ESB	轻量级，例如，HTTP RESTful
服务交付	慢	快
应用场景	企业级	互联网

因此，我们可以看到，SOA和微服务本质上是两种不同的架构设计理念，只是在“服务”这个点上有交集而已，因此两者的关系应该是上面第三种观点。

其实，Martin Fowler在他的微服务文章中，已经做了很好的提炼：

> In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery.

（https://martinfowler.com/articles/microservices.html）

上述英文的三个关键词分别是：small、lightweight、automated，基本上浓缩了微服务的精华，也是微服务与SOA的本质区别所在。

通过前面的详细分析和比较，似乎微服务本质上就是一种比SOA要优秀很多的架构模式，那是否意味着我们都应该把架构重构为微服务呢？

其实不然，SOA和微服务是两种不同理念的架构模式，并不存在孰优孰劣，只是应用场景不同而已。我们介绍SOA时候提到其产生历史背景是因为企业的IT服务系统庞大而又复杂，改造成本很高，但业务上又要求其互通，因此才会提出SOA这种解决方案。如果我们将微服务的架构模式生搬硬套到企业级IT服务系统中，这些IT服务系统的改造成本可能远远超出实施SOA的成本。

微服务的陷阱

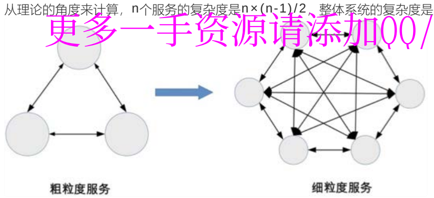
单纯从上面的对比来看，似乎微服务大大优于SOA，这也导致了很多团队在实践时不加思考地采用微服务——既不考虑团队的规模，也不考虑业务的发展，也没有考虑基础技术的支撑，只是觉得微服务很牛就赶紧来实施，以为实施了微服务后就什么问题都解决了，而一旦真正实施后才发现掉到微服务的坑里面去了。

我们看一下微服务具体有哪些坑：

1. 服务划分过细，服务间关系复杂

服务划分过细，单个服务的复杂度确实下降了，但整个系统的复杂度却上升了，因为微服务将系统内的复杂度转移为系统间的复杂度了。

从理论的角度来计算，n个服务的复杂度是n×(n-1)/2，整体系统的复杂度是随着微服务数量的增加呈指数级增加的。下图形象了说明了整体复杂度：



粗粒度划分服务时，系统被划分为3个服务，虽然单个服务较大，但服务间的关系很简单；细粒度划分服务时，虽然单个服务小了一些，但服务间的关系却复杂了很多。

2. 服务数量太多，团队效率急剧下降

微服务的“微”字，本身就是一个陷阱，很多团队看到“微”字后，就想到必须将服务拆分得很细，有的团队人员规模是5 ~ 6个人，然而却拆分出30多个微服务，平均每个人要维护5个以上的微服务。

这样做给工作效率带来了明显的影响，一个简单的需求开发就需要涉及多个微服务，光是微服务之间的接口就有6 ~ 7个，无论是设计、开发、测试、部署，都需要工程师不停地在不同的服务间切换。

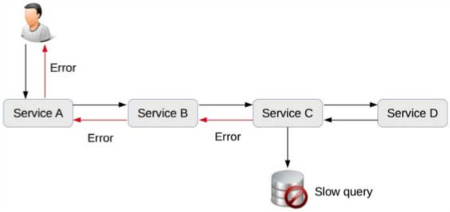
- 开发工程师要设计多个接口，打开多个工程，调试时要部署多个程序，提测时打多个包。
- 测试工程师要部署多个环境，准备多个微服务的数据，测试多个接口。
- 运维工程师每次上线都要操作多个微服务，并且微服务之间可能还有依赖关系。

3. 调用链太长，性能下降

由于微服务之间都是通过HTTP或者RPC调用的，每次调用必须经过网络。一般线上的业务接口之间的调用，平均响应时间大约为50毫秒，如果用户的一起请求需要经过6次微服务调用，则性能消耗就是300毫秒，这在很多高性能业务场景下是难以满足需求的。为了支撑业务请求，可能需要大幅增加硬件，这就导致了硬件成本的大幅上升。

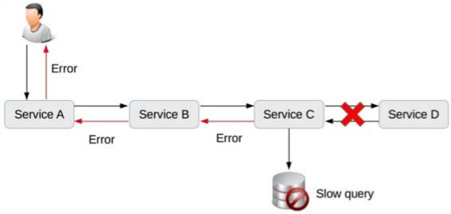
4. 调用链太长，问题定位困难

系统拆分为微服务后，一次用户请求需要多个微服务协同处理，任意微服务的故障都将导致整个业务失败。然而由于微服务数量较多，且故障存在扩散现象，快速定位到底是哪个微服务故障是一件复杂的事情。下面是一个典型样例。



Service C的数据库出现慢查询，导致Service C给Service B的响应错误，Service B 给Service A的响应错误，Service A给用户的响应错误。我们在实际定位时是不会有样例图中这么清晰的，最开始是用户报错，这时我们首先会去查Service A。导致Service A故障的原因有很多，我们可能要花半个小时甚至1个小时才能发现是Service B返回错误导致的。于是我们又去查Service B，这相当于重复Service A故障定位的步骤……如此循环下去，最后可能花费了几个小时才能定位到是Service C的数据库慢查询导致了错误。

如果多个微服务同时发生不同类型的故障，则定位故障更加复杂，如下图所示。



Service C的数据库发生慢查询故障，同时Service C到Service D的网络出现故障，此时到底是哪个原因导致了Service C返回Error给Service B，需要大量的信息和人力去排查。

5. 没有自动化支撑，无法快速交付

如果没有相应的自动化系统进行支撑，都是靠人工去操作，那么微服务不但达不到快速交付的目的，甚至还不如一个大而全的系统效率高。例如：

- 没有自动化测试支撑，每次测试时需要测试大量接口。
- 没有自动化部署支撑，每次部署6 ~ 7个服务，几十台机器，运维人员敲shell命令逐台部署，手都要敲麻。
- 没有自动化监控，每次故障定位都需要人工查几十台机器几百个微服务的各种状态和各种日志文件。

6. 没有服务治理，微服务数量多了后管理混乱

信奉微服务理念的设计人员总是强调微服务的lightweight特性，并举出ESB的反例来证明微服务的优越之处。但具体实践后就会发现，随着微服务种类和数量越来越多，如果没有服务治理系统进行支撑，微服务提倡的lightweight就会变成问题。主要问题有：

- 服务路由：假设某个微服务有60个节点，部署在20台机器上，那么其他依赖的微服务如何知道这个部署情况呢？
- 服务故障隔离：假设上述例子中的60个节点有5个节点发生故障了，依赖的微服务如何处理这种情况呢？

- 服务注册和发现：同样是上述的例子，现在我们决定从60个节点扩容到80个节点，或者将60个节点缩减为40个节点，新增或者减少的节点如何让依赖的服务知道呢？

如果以上场景都依赖人工去管理，整个系统将陷入一片混乱，最终的解决方案必须依赖自动化的服务管理系统，这时就会发现，微服务所推崇的“lightweight”，最终也发展成和ESB几乎一样的复杂程度。

小结

今天我为你讲了微服务与SOA的关系以及微服务实践中的常见陷阱，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，你们的业务有采用微服务么？谈谈具体实践过程中有什么经验和教训。

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

空档滑行

2018-07-14

之前一家公司搞了一次完整的微服务改造，享受到了一些好处，但是文中说到的问题，大部分都碰上了。
先说下好处，原来的单体应用都服务化了，扩容简单很多。功能隔离后之前一个bug导致系统挂掉的现象没了。问题责任定位划分的更清楚，比如之前大量慢sql无人管，现在通过监控快速找到开发责任人。
再说下坏处，1.服务太多了，人不够啊。之前的架构师按照小的原则，把数据层，服务层，应用层严格拆分。一个人手上超过10几个服务...
2.服务化不彻底，太多事手工干。服务监控只能监控一半指标，各种远程调用异常没人解决。运维只有打包发布做了自动化。可以想象下开发人员基本下改bug和发布的死循环中。服务网关没有，服务调用就是一张密密麻麻的网
3.培训不到位，直接上阵，开发人员对微服务理解不到位，服务质量可想而知
4.没有专职测试，自动化测试靠开发写脚本，谁有空啊，单元测试能写一个就算相当有觉悟了
总结下来，做服务化改造首先问自己这些问题，业务真的需要微服务来解决吗？真的所有模块的问题都要微服务来解决吗？技术人员的配置和水平达到要求了吗？

凡凡

2018-07-16

说到微服务，切分的粒度和基础设施都致关重要。

经历的项目有创业初期的单体服务，也有不太完善的服务切分的系统，也有微服务基础设施相对完善的公司。

单体服务致命就致命在随着项目的发展，项目会越来越臃肿，越不利于扩展开发。

微服务过程怕就怕基础不完善，人员配备不够盲目切分，导致工程师开发和维护的战线拉长，特别疲惫和泄气，容易产生一种抱怨的大气氛，从而导致微服务失败，重新合并一部分服务。

微服务做的好的，也有所经历，公司的基础设施完全云化和统一管理，申请几台机器，一套缓存集群，一套mq，sql/nosql...特别容易，工程师愿意建独立的工程，因为很容易构建和部署。这种感觉有点像svn和git，git建分支特别轻量，大家都愿意用分支管理自己的代码，迭代开发，应急处理都能自由切换，得心应手。

现在遇到一个问题，就是微服务内部系统大多使用rpc，但对接外部系统，或者跨外网传输到客户端就需要http-rest类的协议。也就是我们常说的网关，如果是纯http就很容易做到通用的转发机制，但是http转rpc就不知道有什么方式可以做到通用转发了，内部每增加一个rpc，网关就需要增加一个对应的服务处理逻辑。不知道，这个问题，有没有好的解决办法？

作者回复

2018-07-17

把HTTP转RPC做成规则，别硬编码每个接口，例如，规定HTTP URL为/service/interface/method?para1=xxx&para2=yyy

---

鹅米豆发

2018-07-16

对于一个新事物的诞生，本能地套用已有的知识。特别是一个并不简单的东西，这算是一种高效的入门方法。微服务架构其实相当复杂，我是分成好几个阶段理解。
1、第一阶段，微服务架构就是去掉了ESB的SOA架构，只不过是通信的方式和结构变了。对于初级的使用者而言，这样理解没有太大问题。
2、第二阶段，没有了ESB，原本很多由ESB组件做的事儿，转到服务的提供者和调用者这里了。他们需要考虑服务的拆分粒。大体仍然算是SOA架构。
3、第三阶段，随着服务的数量大幅增加，服务的管理越来越困难，此时DevOps出现了。这个阶段的微服务架构，已经是跟SOA架构完全不同的东西了。
之前给一家大国企分享过一些经验。他们想从传统架构，转向微服务架构。
1、建设好基础设施，RPC、服务治理、日志、监控、持续集成、持续部署、运维自动化是基本的，其它包括服务编排、分布式追踪等。
2、要逐步演进和迭代，不要过于激进，更不要拆分过细，拆分的粒度，要与团队的架构相互匹配。（康威定律）
3、微服务与数据库方面，是个很大的难点，可以深入了解下领域驱动设计，做好领域建模，特别是数据库要随着服务一起拆分。
说完上面这些，他们的研发负责人说，我说的跟他们的架构师说的不一样，他们的架构师说，微服务就是各种拆分，不顾一切地拆分。

作者回复

2018-07-17

他们架构师是水货��你的理解和分析是对的，后一篇就讲了

波波安

2018-07-15

我们用的是dubbo。最开始系统要快速上线，所以服务拆分的不彻底。订单，商品，店铺等这些服务都没有进行拆分。就把支付和营销两个服务拆分出来了。服务拆分不彻底经常导致一个业务有问题。整个系统都用不了。

作者回复

2018-07-16

不是拆分有问题，是配套基础设施有问题

3.27。

2018-07-15

我们公司的平台就是使用微服务架构，差不多二十个微服务，但是没有自动化部署，监控，自动化测试这些，而且每次报错日志也特别难找，但是我们那架构师却不重视这些，只想继续升级平台的功能

作者回复

2018-07-16

让你们架构师来订阅架构专栏������

---

kyll

2018-07-18

公司在决定使用微服务的时候，连服务注册发现，配置中心等基础设施组件都没有，业务就同步开发，最后造成了框架组件需要改造，业务却抵制的状况。

作者回复

2018-07-18

掉到坑了��

---

kyll

2018-07-18

公司正在进行微服务的改造，过程确实很痛苦，不光是要在框架上做变革，开发运维的思维意识也需要改变。

作者回复

2018-07-18

是的，微服务是开发，测试，运维都需要改变，刚开始肯定不习惯

星座

2018-07-17

我们用微服务改造了系统，老师文中说的问题基本都遇到了，开始没有服务治理，导致定位问题非常困难，还有就是每服务的版本维护是一个问题，现在服务越来越多，版本要手工一个一个去改，老师有没有好的办法

作者回复

2018-07-17

搞好微服务基础设施建设，后面章节会讲

凡凡

2018-07-17

http转rpc我们卡在了怎么去掉模型和接口这两个骨架代码这里，有没有办法可以不加入骨架代码，按照既定的url自动解析，然后转发？或者说只能使用协议解析实现？

我们用的thrift，回头我需要确认一下thrift的protocol在编解码的时候，有没有写入模型的类型信息，如果没有的话，实现起来感觉不会太复杂，如果包含模型的类型信息，就和骨架代码强耦合了。

阿里如何处理的这种情况呢？

作者回复

2018-07-17

HTTP协议解析很简单的呀，可以试试，阿里也有MTOP承担类似职责

云学

2018-07-17

软件的复杂度由业务逻辑和控制逻辑决定，前者决定了复杂度上限，微服务架构仅能帮助改善控制逻辑复杂度。

作者回复

2018-07-17

控制逻辑具体指什么？

王维

2018-07-17

按照这样说，用Web API方式构造的服务，也是属于微服务吗？

作者回复

2018-07-17

不一定，微服务不单单是将ESB改为HTTP，后面会深入阐述

---

hello

2018-07-17

请问李老师，esb是单独的一个系统吗？esb需要知道每个系统的接口和协议吗？

作者回复

2018-07-17

是的，需要适配每个系统的每种接口协议和数据格式

---

krugle

2018-07-17

go语言有什么微服务全家桶之类的吗，找了几个不理想,没java多

作者回复

2018-07-17

我了解是没有��

小田

2018-07-16

\# 微服务特点小结
细粒度服务化: 依赖服务治理系统
快速交付: 依赖敏捷开发技术(自动化测试,持续集成,自动化部署)
轻量级通信: RESTful API，RPC
场景: 互联网应用(相对于"企业内部应用")

Neal

2018-07-16

1. 数据库拆不开
2. 人手不够就两人

---

ant

2018-07-16

我们是一家社交公司，后端加厚的演变符合dubbo官网的那张图，在Mvc架构坚持了一年后，业务越来越大，工程越来越臃肿。后面我们一致同意进行服务话，开始用了dubbo。后面由于决策层的原因没有上，后面来了个架构师，又重启了服务拆分，到现在已经用于生产。我们使用的是Spring cloud，现在拆分暴露了很多问题：
1、个别服务没有熔断出来，出现过雪崩效应
2、服务拆分过细，服务调用链过长
3、开发人员能力不一样，代码水平不一样
4、没有监控措施
5、每个服务部署多台，日志查询就会死人的感觉
6、开发过程中经常出现访问不到该访问的接口，这是因为开发人员经常启动本地服务，就会导致30%的概率访问不到

7、使用了不合理的持久层框架，使用了JPA访问
大概就有上面的问题，总之微服务不是银弹，用得不好会发现无穷无尽的坑。当然，出现问题解决问题就是了。唯一的就像CEO说的：你们他妈的是完全在拿用户当小白鼠使用

作者回复

2018-07-16

用户会用脚投票的����

问题究竟系边度

2018-07-16

算是用了微服务。
没有保证足够人力不要实施微服务。
需求的功能拆分需要更资深的人做规划。
业务没有到一定的复杂度不要实施微服务，不要追时髦。
监控，部署，机器资源，服务接口管理是实施重点。

衣申人

2018-07-15

同问:dubbo和motan这类rpc框架，是微服务框架？？

作者回复

2018-07-16

dubbo可以算微服务基础设施的一部分，主要承担服务注册发现等

霍涌

2018-07-15

很多情况都是为了为微服务而微服务，。我们公司也是采用了微服务，但结果很不理想，改造后的系统每天都会有10到30分钟出问题，使用者苦不堪言

作者回复

2018-07-16

你们需要一个架构师��

---

fiseasky

2018-07-15

搭建微服务架构需要搭建那些基础设施呢?

作者回复

2018-07-16

有专门章节介绍

乡工鸟

2018-07-15

个人感觉soa与微服务的最大区别在他们出现的背景就可以可清晰的标明了。一个面向多系统集成，一个主打系统内拆分。至于微服务的坑，部署上微服务了，思想上还没有(除了把系统拆小之外，而且拆的原则还不清不楚，剩下所有的一切还按老路走)��

作者回复

2018-07-16

焦油坑��

---

mini希

2018-07-14

基于Spring Cloud或者其他框架是不是可以减少一些成本

作者回复

2018-07-16

减少很多，但是不灵活

Aec

2018-07-14

看到上面的留言几个服务，就引入微服务。。杀鸡用牛刀啊。。还是先跑业务吧。。华而不实。。无论业务复杂度，流量，需求迭代。。4.5个人拆10，20个服务想啥呢

作者回复

2018-07-16

很多团队或者领导需要一些给自己镀金的东西����

Snake

2018-07-14

服务多了之后用的资源反而比之前更多，没有监控系统和日志聚合系统查看问题就像大海捞针

小西

2018-07-14

李老师，Dubbo是属于SOA架构还是微服务架构?

作者回复

2018-07-16

微服务

---

yungoo

2018-07-14

我们也在采用微服务，后端团队大概20人，十来个服务，4/5条产品线，基础服务由4/5人团队维护。碰到的最大问题是服务部署和应用监控的问题，其次是部分服务没有恰当的抽象，还在为不同的业务定制接口。

正是那朵玫瑰

2018-07-14

我们的业务也算是微服务吧， 接手项目时，已经有好多服务，主要采用的语言有java，php，nodejs，存在以下问题:
1、没有一个统一的网关服务，前端请求后端服务都需要后端的服务A来充当安全校验，权限校验等，A服务充当了多重职责，变的职责不明确了，后来抽出网关系统，负责平台统一的流量入口。在构建微服务网关系统是至关重要的。
2、监控系统不完善，调用链跟踪，异常报警都不完善，对微服务是巨坑，查找问题如同一场噩梦，调用链很长，一旦发生异常不知道到底哪里出现问题，得一个一个去找。后来慢慢完善，变得好了很多，问题很快定位。
3、加入网关后，没有一个统一的服务发现注册中心，网关的路由依靠人工手动配置，变的很麻烦，也很容易出错。后来引入consul，得到改善。

作者回复

2018-07-16

spring全家桶，你值得拥有��

张玮(大圣)

2018-07-14

采取一些措施把坑填平，生态中有一些通用措施，

不能满足再走内部开发，

做这些事之前就是得评估下团队人的情况和产品项目的发展情况。

---

hansc

2018-07-14

请问老师soap是esb的一种吗?

作者回复

2018-07-14

soap是协议，ESB是系统

---

narry

2018-07-14

我弄微服务遇到最大的坑，就是jvm与docker兼容性不好，导致每个微服务会消耗过多不必要的内存，我感觉从资源消耗上来说，java开发的微服务都不能算是微服务了，最近在转向go来改造

作者回复

2018-07-14

这还真是第一次听说jvm与docker不兼容，看看是不是有bug