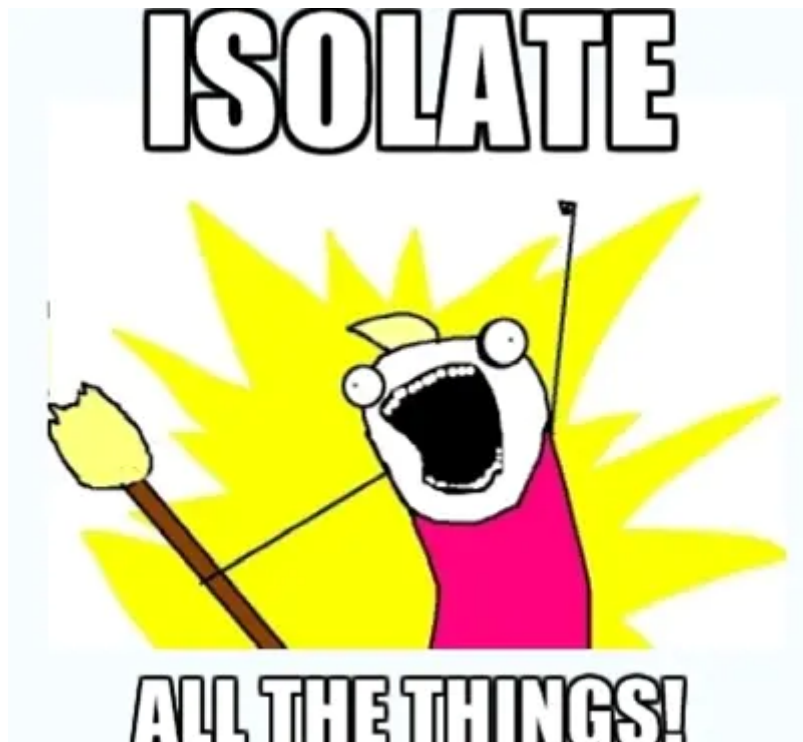


Alchemist5777 Lv3

2017年06月01日 阅读 9296

[关注](#)

简单聊聊SOA和微服务



架构设计中的朴素主义

前两天和一个朋友聊天，他向我咨询如何从零开始构建一个健壮、强大的软件系统，聊着聊着他忽然问我，「听大家都在说微服务（下文中有的地方会使用MSA），还有人会提到SOA，那么他们的区别到底在哪里？」。我想了想，一时也列不出来一个详细的列表，只能跟他讲说其实他们在概念上是相似的。

关于软件系统的架构设计，是一个太多人喜欢讨论的问题，尤其是对软件开发不了解的人士来说，总是被各种各样的概念绕来绕去。从更高的层次，更大的视野研究架构设计的一些专家（如Martin Fowler、Chris Richardson）能够很清晰的列出这些众多概念的区别，而身在开发一线的我们更多的是对这些概念有一些感性的、朴素的认识。将之内化到自己的工作中去，才能更好的发挥架构的优势。

本文将简单谈谈SOA和MSA的概念，并简单列举一下如果使用Java来构建一个微服务系统时所会遇到的一些概念。本文只是「Yet Another」叙述微服务的文章，而且观点并没有很严谨，希望能让读者对微服务和SOA留下一个感性的认识，如果有什么不对的地方，还请轻拍。

微服务和SOA的区别

所以，到底SOA和微服务有什么区别？

短答案：微服务是SOA发展出来的产物，它是一种比较现代化的细粒度的SOA实现方式。

微服务就是这样的一个「概念」，说白了，它不过就是近些年火起来的有一个名词而已，一时间仿佛整个行业都在讨论它，仿佛终于找到了银弹。这个时候SOA看待微服务大概就如同当年的Friendster和Myspace看待Facebook一样——大家都忘了社交网络这个东西并不是Facebook发明的。你再谈如何使用SOA去构建一个系统，就像是在谈一个过气的明星，人们会认为你已经落伍了。但真的是这样的吗？

MSA is a SOA

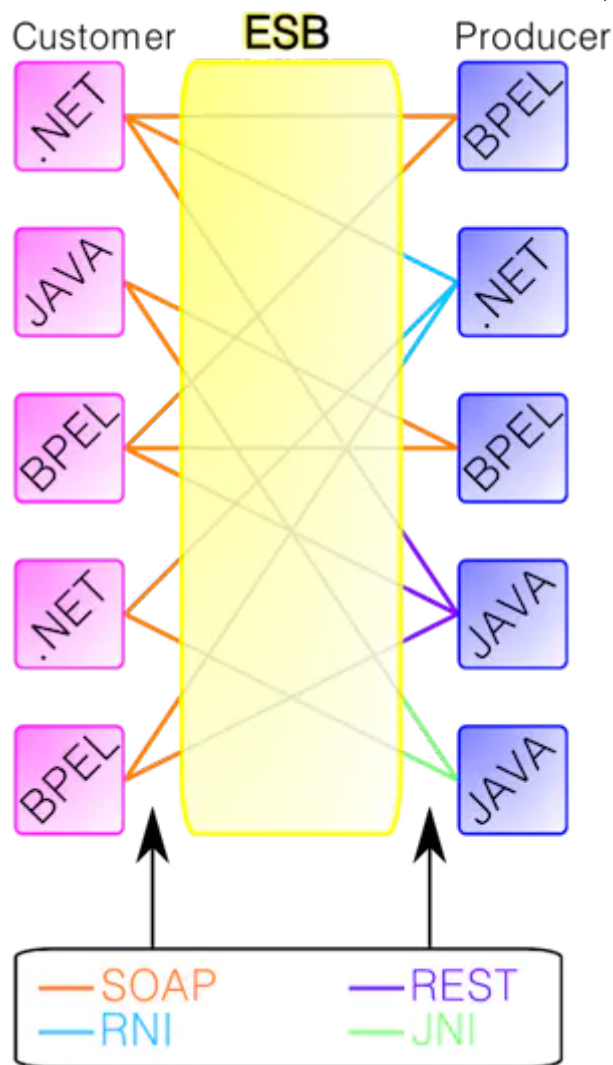
较早实践微服务的公司Netflix就曾经称他们构建的架构是「细粒度的SOA」

讨论「微服务和SOA的差别」的意义远不如讨论「微服务和单体系统的差别」更大，因为他们的区别实在有点微妙。此外，互联网近些年的发展，越来越朝去中心化的方向前进了，就像今天的IT工程师不需要像律师、教师那样，需要得到某些机构的认可才能更好的开展工作，这一方面意味着门槛的降低，另一方面也意味着更多的概念没有一个权威的声音来对它进行定义，使得每个人可以根据自己的需求做出不同的调整。

微服务和SOA都是这样背景下的产物，并没有一个权威的定义，来说明它们各自包含了什么东西，使用什么的方法进行系统的构建。但是，还是可以从最大的范围来对比它们的不同，当我们今天说出这两个概念时，其区别往往没有那么大，但SOA是有一定的历史了，在历史上的SOA往往意味着更多的东西，而这些都是现在很多人在做架构设计时不会采用的。

MSA vs SOA

1. 最多被人谈起的，应该算是ESB了



万能的ESB（图片来自Wikipedia）

ESB(enterprise service bus)曾经着实跟随着SOA火了一阵子，第一次知道它是一个朋友在一家物流公司工作，说他们用了个很高级的玩意儿，叫企业服务总线，接着给我灌输了一堆ESB的理念。当时感觉这玩意儿太牛了，设计理念也很厉害。

从名称就能知道，它的概念借鉴了计算机组成原理中的通信模型——总线，所有需要和外部系统通信的系统，统统接入ESB，岂不是完美地兼容了现有的互相隔离的异构系统，可以利用现有的系统构建一个全新的松耦合的异构的分布式系统。

但，实际使用中，它还是会有很多的缺点，首先就是ESB的本身就很复杂，大大增加了系统的复杂性和可维护性。其次就是由于ESB想要做到所有服务都通过一个通路通信，直接降低了通信速度。

而在现代的微服务中，往往是一个「富终端、瘦通信」（Smart endpoints and dumb pipes），使用轻量级的通信机制，而每个终端（服务）有自己的处理逻辑，它知道它要找的服务在哪里，不需要在通信的链路上做什么事情。

然而，ESB是一个历史产物，用今天的眼光看待它，并且将之当做SOA的一个标签是不合理不公平的。

2. 服务化的概念和服务的尺寸

如前所述，SOA的出现其实是为了解决历史问题：企业在信息化的过程中会有各种各样互相隔离的系统，需要有一种机制将他们整合起来，所以才会有上边所述的ESB的出现。同样的，也造成了SOA初期的服务是很大的概念，通常指定的一个可以独立运作的系统（这样看，好像服务间天然的松耦合）。这种做法相当于是「把子系统服务化」。

而微服务没有历史包袱，轻装上阵，服务的尺寸通常不会太大，关于服务的尺寸，在实际情况中往往是一个服务应该能够代表「实际业务场景中的一块不可分割或不易分割的业务实体」。将服务的尺寸控制在一个较小的体量可以带来很多的好处：

1. 更易于实现低耦合、高内聚
2. 更易于维护
3. 更易于扩展
4. 更易于关注实际业务场景

3. 通信协议（好吧，我实在编不下去了）

如今越来越多的工程开始使用RESTful来作为API的设计的基础，但仅仅几年前还有大把的API使用SOAP、WSDL等基于XML的重量级协议的Web Service。

这点和上文说到的2点其实大同小异，仔细想想，它们都是由于历史原因造成的，同样的，通信协议经过这些年的发展，现在主流的基本上了两种：

1. 文本协议
使用最广泛的多是基于HTTP的RESTful规范
2. 轻量级二进制协议
Thrift、Protobuf，或者任何自定义的轻量级协议

要解决的问题

大家都在讨论微服务，造成了当有人要构建一个系统，往往会第一时间想到使用它，较少有人会去想为什么要用微服务，是不是有更好的选择。

没有银弹

软件工程是一个处处有坑的事情，但MSA不是银弹，并不是所有的软件系统都适合。它有自己的适应的场景和不足之处，下面根据自己的理解列出一些系统的需求，如果你的系统有这方面的需求，那么可以考虑使用微服务来构建你的系统。

复杂性

顾名思义，微服务首先强调的是服务粒度比较小，这就带来了一个直接好处：实现简单。

如果你判断你的系统需要有很多截然不同的功能模块，复杂的业务逻辑，业务目标多种多样，那么你可以开始考虑使用MSA将你的系统分解成概念上独立的服务，这样一来开发工作的难度可以直接的降低了。当然，与此同时你要面对的是另外一种复杂性挑战——从概念上分解和管理这些服务本身就是一件十分复杂的工作。好在现在开源社区有很多较完善的服务治理的框架和解决方案，正可以用来解决这方面的复杂性问题。

相反，假如你要做一个论坛系统，或者一个简单的电商网站，并没有进行大面积推广的计划，预计用户规模在可预见的范围内不会超过10万。那么你需要的是不是微服务，而是可以快速开发并上线的架构，这种情况单体应用最合适，也有很多的开源解决方案可以用来开发MVP。

可维护性

微服务的细粒度服务和分布式部署的特点，带来的一个好处是：可独立维护。

如果你的系统需要持续的改进，以配合业务的不断发展，那么可以考虑使用MSA来做系统的架构。例如，你们正在做一个电商网站，经常要做各种各样的活动，优惠条件、优惠力度、甚至优惠的计算方式都经常在变，这所有的变化不可能在系统设计的时候完全考虑到，当需要开发介入时，MSA的优点就体现出来了。

服务的分离所遵循的很多原则（如SRP、开闭原则）正是为这种改变所提出的，根据这些原则，你的系统里可能有一个服务叫「订单服务」，所有的改动都只需要在这个服务内进行，或者更进一步，你有一个专门处理活动的服务，只需要很少的开发和测试的工作量就可以实现。

C10K、C100K。。。

高并发这个概念快变得烂大街了，不同的高并发场景需要的解决方案千差万别，同时它也是一个系统工程，而不是通过某一个方面就能解决的。如对于一个做内容的网站，高并发带来的直接挑战是大量数据

（图片、文本）的获取，而对于写的需求相对而言并不会很高，所以CDN、缓存、网页静态化等都是可以采用的软硬件解决方案；而对于电商类的交易系统，更重要的是保证系统的快速响应、持续可用、最终一致、和水平扩展，所以NIO、分库分表、缓存、API升降级、负载均衡都是可以采用的方案。

当然，本文的重点是MSA，所以这里只从架构的角度讲一下MSA对于解决高并发问题的优势。由于MSA通常是分布式部署，通过服务注册中心实现服务的发现，服务具体部署的机器对于服务的调用方来说是透明的，然后通过客户端或者服务端的软负载均衡，可以在某个服务的压力大的时候直接添加机器实现水平扩展，对于性能的提示在某个范围内可以认为是线性的。从架构的角度，微服务是解决某一类高并发的较好的解决方案。

同时，高层的架构设计只能从大局上解决这个问题，比如你设计了一个很完善的分布式系统，但好多的服务经常出现内存泄漏，动不动机器就当掉，那多好的架构都于事无补。

框架选择的玄学

虽然Java的框架还没有多到JS那么离谱，但相同或相似的功能通常都会有两个或两个以上的选择，有的需要多选其一，有的可能要组合使用。如何在这些名目繁多的框架中选择出来适合自己的框架十分困难，每个人心中都有自己的一套的价值观，快变成了玄学了。

Languages	
JavaScript	33,320
PHP	32,208
Java	27,752
Python	14,704
C#	11,373
C++	8,811
CSS	8,787
HTML	8,022
Ruby	5,501
Objective-C	4,395

在github搜索「framework」的结果

有一派的存在主义哲学认为，人是自由的，这体现在人有选择的自由，一个人做的每一个选择都成为他未来的自己的一个组成部分。放在架构设计上也适用，你在做各种各样的选择时，也一步步地成就了你要构建的那个系统的样子。

Java中的微服务

虽然MSA其本身不排斥多语言的异构系统，实际场景中也会有多语言开发的例子，但多数情况下，很多人还是习惯于系统中大多数的服务都是用相同的语言进行开发。在Java的生态中，已经有很多十分成熟的，可以拿来实现MSA的中间件，比较出名的有：Spring全家桶（基于Netflix）、Dubbo（国内十分流行）、Thrift（需要自己实现一些基础的东西）。

通常，MSA的实现通常要满足下面几个条件：

1. 服务足够的小，需要根据自己的实际需求决定服务的大小。
2. 服务可以独立开发、部署、测试。
3. 使用轻量级通信方式。
4. 数据分离

实际场景中的选择

在实际的MSA的实践中，会遇到各种各样的选择，To be or not to be，被这个问句困扰着的不止哈姆雷特，还有很多开发者。实际场景中会遇到的选择包括但不限于以下几点，可以拿来作为一个checklist。这并不是一个完整的列表。

- 具体细节的实现方式

从代码级的角度看，主要是具体细节的实现方案，代码的好与坏很难用一个定量的标准去衡量，但GOF的设计模式是一个很好的开始，「重构」那本书中也讲到了很多「代码中的坏味道」。多数情况下，这里讲到的坏代码并不是指的性能，而是指其可读性与可扩展性。

- 基础设施

任何软件系统都是要部署到具体的基础设施上去的，关于基础设施的部署也有很多的选择，如：PC vs Docker、Apache vs Nginx、Tomcat vs Jetty。

- 微服务框架

1. Spring全家桶

用起来很舒服，只有你想不到，没有它做不到。

2. Dubbox

很多国内的企业还在用，可以支持RESTful风格的API，但更多的还是会使用Dubbox的默认的基

于RPC的API，调用远程API像调用本地API一样。这样做无疑带来了优势，但同时其基于接口的方式增加了服务间的耦合，怎么说呢，各有利弊。

3. Thrift

如果你比较高冷，完全可以基于Thrift自己搞一套抽象的自定义框架。

- 同步vs异步

在跨服务的业务逻辑的实现上，使用基于消息的异步调用，还是使用保证结果的同步方案。

- 数据服务

内存数据库 vs 持久化数据库 (Redis vs MySQL)

关系型数据库 vs 非关系型数据库 (MySQL vs Mongo)

传统数据库 vs 分布式数据库 (MySQL vs F1)

非关系型数据库又有如KV数据库，文档数据库，图数据库等。

由于MSA提倡服务间的数据隔离，往往不同的服务使用不同的数据源，这就会直接导致数据聚合查询比较困难的问题。进行数据聚合又有几种不同的方案（如CQRS）。

- 日志分析

日志分析也是有很多的成熟解决方案（如ELK）。

关注下面的标签，发现更多相似文章

[Java](#)[架构](#)[微服务](#)**Alchemist5777** Lv3

后端工程师

获得点赞 2,681 · 获得阅读 64,531

[关注](#)

安装掘金浏览器插件

打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

石在心中 java新手

顶顶顶



8月前

  回复

zproo

写的最好的一篇。所有人都是soa和微服务对比，微服务最该和单体系统对比。

2年前

  回复

Docker精选 Lv2 DockOne.io @ DockO...

你好，我是 DockOne 社区发起人李颖杰，刚看到你写的这篇文章，可以加一下微信吗？我的微信号是：liyingjiesz

3年前

 1  回复

Zhang_J JAVA研发 @ 微链

好文

3年前

  回复

Jhonwill


赞一个

3年前

相关推荐



古时的风筝 · 7小时前 · Java

『JVM』我不想知道我是怎么来滴，我就想知道我是怎么没滴

 17  4



小姐姐味道 · 8小时前 · Linux / 架构

别小看tail 命令，它难倒了技术总监

 19  3



MacroZheng · 8小时前 · Java / MySQL

MySQL如何实时同步数据到ES？试试这款阿里开源的神器！


 29  4

程序猿DD_ · 6小时前 · Java

仅用六种字符来完成Hello World，你能做到吗？


 3  4

23张图！万字详解「链表」，从小白到大佬！

 13 

是小齐呀 · 8小时前 · Java / 算法

快速排序为什么这么快？

 8 

程序那些事 · 7小时前 · Java

JDK8中的新时间API:Duration Period和ChronoUnit介绍

 4  3

JavaGuide · 8小时前 · Java / 面试

系统设计面试指北

 7  1

程序那些事 · 7小时前 · Java

理解分布式一致性:Paxos协议之Multi-Paxos

 5  1

MacroZheng · 2天前 · Java / Spring Boot

还在手动整合Swagger? Swagger官方Starter是真的香！

 57  10