微服务和 SOA（面向服务架构）区别

作者：xixiaoming
原文链接：https://ld246.com/article/1531359106778
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

相同点

1. 需要 Registry，实现 **动态的服务注册发现机制**；

2. 需要考虑分布式下面的 **事务一致性**，CAP 原则下，两段式提交不能保证性能，事务补偿机制需考虑；

3. 同步调用还是异步消息传递，如何保证消息可靠性？SOA 由 **ESB**（企业服务总线） 来集成所有消息；

4. 都需要统一的 **Gateway** 来汇聚、编排接口，实现统一认证机制，对外提供 APP 使用的 **RESTful** 接口；

5. 同样的要关注如何在分布式下 **定位系统问题**，如何做日志跟踪，如电信的信令跟踪

差别点?

1. 是持续集成、持续部署？对于 CI、CD（持续集成、持续部署），这本身和敏捷、DevOps 是交织一起的，这更倾向于软件工程的领域而不是微服务技术本身；

2. 使用不同的通讯协议是不是区别？微服务的标杆通讯协议是 RESTful，而传统的 SOA 一般是 SOA，不过目前来说采用轻量级的 RPC 框架 Dubbo、Thrift、gRPC 非常多，在 Spring Cloud 中也有 Fegn 框架将标准 RESTful 转为代码的 API 这种仿 RPC 的行为，这些通讯协议不应该是区分微服务架和 SOA 的核心差别；

3. 是流行的基于容器框架还是虚拟机为主？Docker 和虚拟机还是物理机都是架构实现的一种方式，是核心区别；

微服务架构的精髓在切分

1. 服务的切分上有比较大的区别，SOA 原本是以一种“集成”技术出现的，很多技术方案是将原有业内部服务封装为一个独立进程，这样新的业务开发就可重用这些服务，这些服务很可能是类似供应、CRM 这样的非常大的颗粒；而微服务这个“微”，就说明了他在切分上有讲究，不妥协。无数的例证明，如果你的切分是错误的，那么你得不到微服务承诺的“低耦合、升级不影响、可靠性高”之的优势，而会比使用 Monolithic 有更多的麻烦。

2. 不拆分存储的微服务是伪服务：在实践中，我们常常见到一种架构，后端存储是全部和在一个数据中，仅仅把前端的业务逻辑拆分到不同的服务进程中，本质上和一个 Monolithic 一样，只是把模块间的进程内调用改为进程间调用，这种切分不可取，违反了分布式第一原则，模块耦合没有解决，性却受到了影响。

分布式设计第一原则 — “不要分布你的对象”

1. 微服务的“Micro”这个词并不是越小越好，而是相对 SOA 那种粗粒度的服务，我们需要更小更适的粒度，这种 Micro 不是无限制的小

2. 一个简单的图书管理系统肯定无需微服务架构。既然采用了微服务架构，那么面对的问题空间必然比较宏大，比如整个电商、CRM