# 李豪微服务学习笔记

在《凤凰架构》书中提到了软件架构的发展过程，其实软件架构的发展和硬件系统是密不可分的，早在20世纪末就已经提出了分布式这个概念，当时背景是计算机性能十分有限，为了实现复杂逻辑的计算。但是局限于当时的硬件与合理的规则最终没有落地，就比如说远程调用，当时可没有如今的网络速度，带来的消耗是非常大的。后来盛行是单体架构，但是随着软件体量的增大，其部署和维护的代价是很大的，往往“会牵一发而动全身”，一点错误导致全盘崩溃很容易发生。再后来就是如今盛行的微服务架构。

结合《凤凰架构》和论文《Microservices: Yesterday, Today, and Tomorrow》，以及自己曾经做的一个微服务项目我对微服务的认识如下：

1、组件化

我认为组件化是微服务的核心，组件和单体架构中的“库”，其实都是为了将一个方法或者功能进行分离，但是组件可以独立运行部署，组件之间调用不在同一进程内，需要用到其他的机制，但是库是可以之间调用的。组件化可以很有效的避免我们上述提到的“牵一发动全身”问题，即使程序在运行中出错，最多我们也只是把其所在的组件服务撤回修改，其他组件依旧可以保持正常的运行，这样的效果在单体架构中是不可能实现的。

在刚开始学习微服务的时候也是逐步的把一个单体架构程序进行组件的逐步拆分，但是又产生了新的问题，拆分力度怎样确定，换言之，微服务到底有多微？这个问题没有统一的规范，但是拆分粒度确实对程序有很大的影响，组件太少体现不出微服务的优势，一个错误影响的范围可能依旧很大。但是组件太多又会影响系统的性能，因为组件之间调用是需要用到网络协议的，网络传输占用的时间和资源消耗往往是api调用延迟的主要因素。

2、去中心化治理

在单体架构中，项目开案前往往就确定了哪个语言进行编写，往往是一整套全部由一种语言编写而成（虽然目前运用了微服务的项目也经常是这样），但是微服务架构给我们提供了一种去中心化治理方案，其实不同的语言具有不同的擅长之处，我们完全可以按照不同组件的需求进行选择不同语言编写，最后只需要三方协议完成组件之间的调用。

微服务技术是一把双刃剑，上面也提到，开发和维护的便利是有一部分性能的牺牲，除此之外微服务系统比单体系统要消耗更多的资源，但是这好像并不能阻止微服务的发展，体量更大的系统、功能更多的软件目前最好的技术选型也只有微服务。