Atitit 必须要上微服务的指标 应用体积与启动时间

而这种应用的尺寸也会拖慢开发进度。应用越大，启动时间越长。譬如在最近的调查中，不少开发者指出启动时间长达 12 分钟。我也听说有的应用启动时间居然得 40 分钟。如果开发者不得不频繁重启应用服务器，那大量时间就被浪费，生产效率也饱受其害。

一旦你的应用变得庞大、复杂，你的开发团队将饱受折磨，苦苦挣扎于敏捷开发和交付。一大原因就是应用已经格外复杂，庞大到任何一个开发者都无法完全理解。最后，修复 bug 和实施新功能也就极其困难且耗时颇多。更可怕的是，这是一个向下的螺旋发展。代码库越难理解，正确的修改就越难。最后你会深陷庞大的、无法估量的泥淖之中。

采用此种风格的应用非常普遍。由于 IDE 和其他工具擅长构建单一应用，这类应用也易于部署。这类应用也非常容易测试。你可以非常轻松地进行端到端测试，使用 Selenium 测试 UI 。整体应用也便于部署，只需将软件包复制到服务器。你也可以通过运行多个包和负载均衡实现扩展。在项目早期这么做非常有效。

几年后，原本小而简单的应用会变得臃肿。举个极端的例子，我最近与一位开发者交流，他正在开发一款小工具，来分析他们应用（包括几百万行代码）中的几千个 JARs 的依赖。我相信每年都会有大量开发者不遗余力地对付这种麻烦。

而这种应用的尺寸也会拖慢开发进度。应用越大，启动时间越长。譬如在最近的调查中，不少开发者指出启动时间长达 12 分钟。我也听说有的应用启动时间居然得 40 分钟。如果开发者不得不频繁重启应用服务器，那大量时间就被浪费，生产效率也饱受其害。

庞大且复杂的单体应用的另一大问题就是难以进行持续部署。现在， SaaS 应用的发展水平足以在单日内多次将修改推送到生产环境。然而要让复杂的单个应用达到此水平却极为棘手。想更新应用的单个部分，必须重新部署整个应用，漫长的启动时间更是雪上加霜。另外，由于不能完全预见修改的影响，你不得不提前进行大量人工测试。结果就是，持续部署变得不可能。

单体应用的另一问题就是可靠性。由于所有模块都运行在同一进程中，任何模块中的一个 bug，比如内存泄漏都可能弄垮整个进程；此外，由于应用中的所有实例都是唯一，这个 bug 将影响整个应用的可用性。

微服务架构范式对应用和数据库的关系影响巨大。每个服务都有自身的数据库计划，而不与其它服务共享同一个数据库。一方面，这个方法类似企业级数据模型。同时，它也导致部分数据的重复。然而，要想从微服务中获益，为每个服务提供单个的数据库计划就非常必要，这能保证松散耦合。下面的图表展示了示例应用的数据库架构。

intro-microservices

用是分布式系统，由此会带来固有的复杂性。开发者需要在 RPC 或者消息传递之间选择并完成进程间通讯机制。此外，他们必须写代码来处理消息传递中速度过慢或者不可用等局部失效问题。当然这并不是什么难事，但相对于单体式应用中通过语言层级的方法或者进程调用，微服务下这种技术显得更复杂一些。

作者：xuliang87  
链接：https://www.jianshu.com/p/a8d5bef3ff5a  
来源：简书  
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。