微服务介绍

2020.05　研发A部 王振

Contents

什么是微服务

历程：

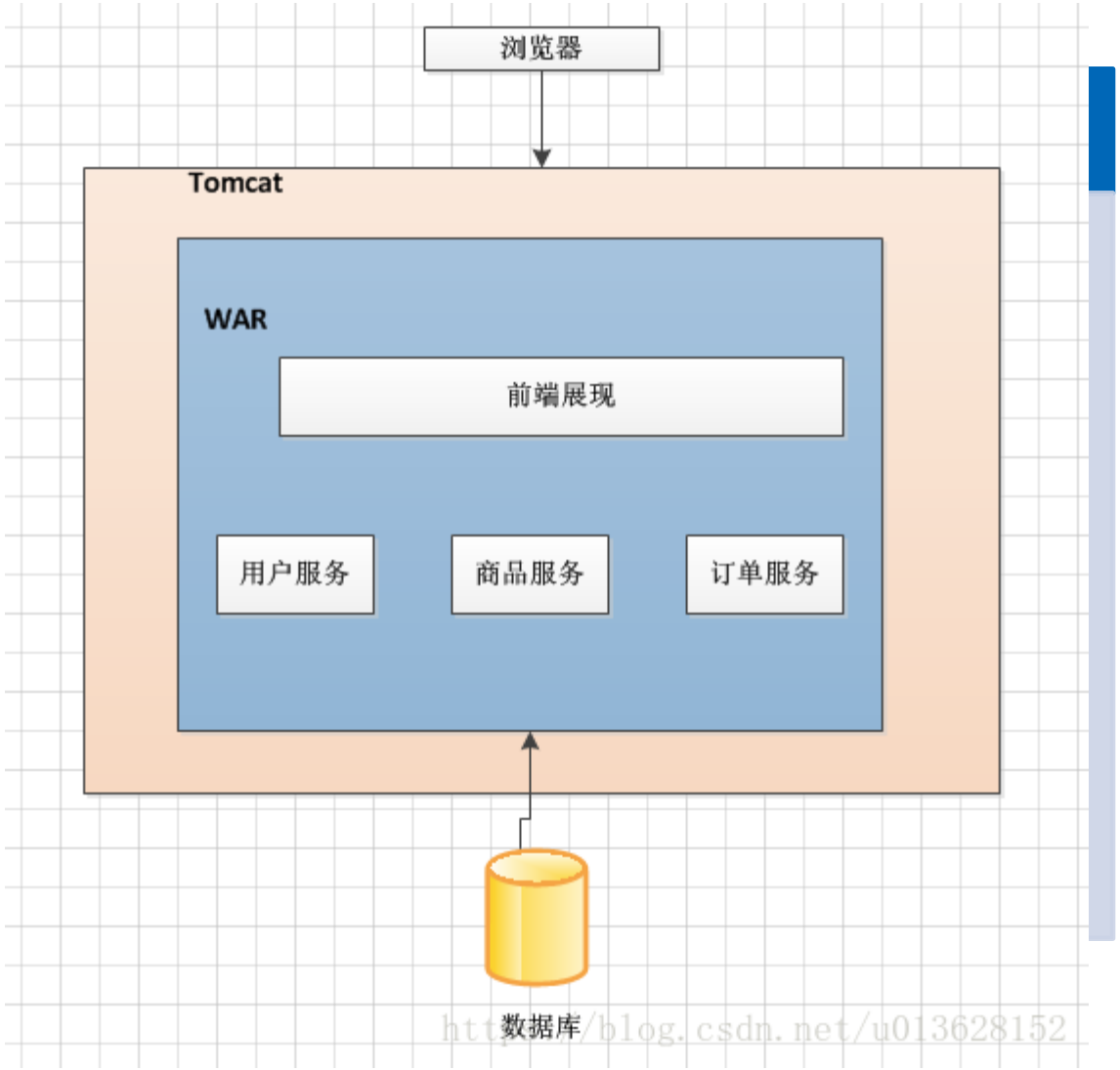
- 微服务的起源是由 Peter Rodgers 博士于 2005 年度云计算博览会提出的微 Web 服务 (Micro-Web-Service) 开始，Juval Löwy 则是与他有类似的前导想法，将类别变成细粒服务 (granular services)，以作为 Microsoft 下一阶段的软件架构，其核心想法是让服务是由类似 Unix 管道的访问方式使用，而且复杂的服务背后是使用简单 URI 来开放接口，任何服务，任何细粒都能被开放 (exposed)。这个设计在 HP 的实验室被实现，具有改变复杂软件系统的强大力量。
- 2014年，Martin Fowler 与 James Lewis 共同提出了微服务的概念，定义了微服务是由以单一应用程序构成的小服务，自己拥有自己的行程与轻量化处理，服务依业务功能设计，以全自动的方式部署，与其他服务使用 HTTP API 通信。同时服务会使用最小的规模的集中管理 (例如 Docker) 能力，服务可以用不同的编程语言与数据库等组件实现[1]。

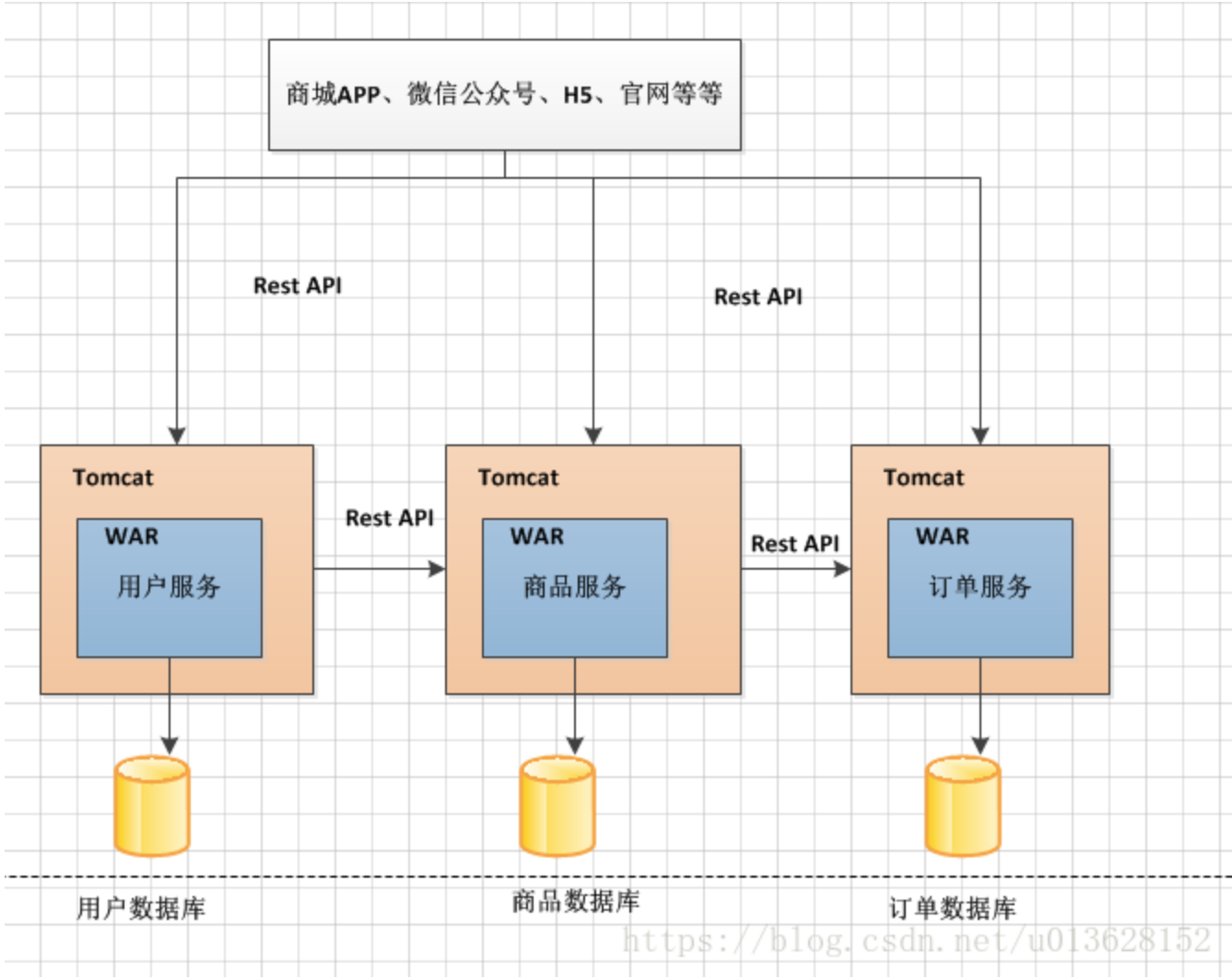
概念：

- 维基百科：**微服务** (Microservices) 是一种软件架构风格，它是以专注于单一责任与功能的小型功能区块 (Small Building Blocks) 为基础，利用模块化的方式组合出复杂的大型应用程序，各功能区块使用与语言无关 (Language-Independent/Language agnostic) 的 API 集相互通信。
- 百度百科：所谓的微服务是SOA架构下的最终产物，该架构的设计目标是为了肢解业务，使得服务能够独立运行。微服务设计原则：1、各司其职 2、服务高可用和可扩展性

什么是微服务

北控伟仕





为什么要用微服务

1.复杂性高

- 项目代码多几十万到几百万行，各个模块之间定义模糊，逻辑比较混乱，代码越多复杂性越高，越难解决遇到的问题。
- 各系统之间通过webservice交互，接口多，难维护。

2.部署及测试问题

- 启动慢：单体架构模块非常多，代码量非常庞大，导致部署项目所花费的时间越来越多，四险项目启动一次3-5分钟左右，页面访问加载3-5分钟。
- 部署复杂：单体架构中，整个系统运行在同一个进程中，当应用进行部署时，必须停掉当前正在运行的应用，部署完成后再重启进程，无法做到独立部署。
- 整体性：在单块架构中所有功能都在同一个代码库，功能的开发不具有独立性；当不同小组完成多个功能后，需要经过集成和回归测试，测试过程也不具有独立性；当测试完成后，应用被构建成一个包，如果某个功能存在 bug，将导致整个部署失败或者回滚

3.技术债务

- 单体项目代码量庞大的惊人，开发过程中留下的坑很难被发觉

4.可扩展性弱

- 技术升级和新技术引进很难做到。

为什么要用微服务

1. 单一职责

- 微服务架构中的每个服务，都是具有业务逻辑的，符合高内聚、低耦合原则以及单一职责原则的单元，不同的服务通过“管道”的方式灵活组合，从而构建出庞大的系统。

2. 轻量级通信

- 服务之间通过轻量级的通信机制实现互通互联，而所谓的轻量级，通常指语言无关、平台无关的交互方式。对于轻量级通信的格式而言，我们熟悉的 XML 和 JSON，它们是语言无关、平台无关的；对于通信的协议而言，通常基于 HTTP，能让服务间的通信变得标准化、无状态化。目前大家熟悉的 REST（Representational State Transfer）是实现服务间互相协作的轻量级通信机制之一。使用轻量级通信机制，可以让**团队选择更适合的语言、工具或者平台来开发服务本身**。
- 技术异构性：使用最适合该服务的技术，降低尝试新技术的成本；

3. 独立性

- 在微服务架构中，每个服务都是独立的业务单元，与其他服务高度解耦，只需要改变当前服务本身，就可以完成独立的开发、测试和部署。

4. 进程隔离

- 每个服务在应用交付过程中，独立地开发、测试和部署。
- 在微服务架构中，应用程序由多个服务组成，每个服务都是高度自治的独立业务实体，可以运行在独立的进程中，不同的服务能非常容易地部署到不同的主机上。
- 伸缩性强：每个服务都可按硬件资源的需求进行独立扩容

那么微服务有哪些挑战呢?

1. 分布式系统的复杂性

微服务架构是基于分布式的系统，而构建分布式系统必然会带来额外的开销。

- **性能：** 分布式系统是跨进程、跨网络的调用，受网络延迟和带宽的影响。
- **可靠性：** 由于高度依赖于网络状况，任何一次的远程调用都有可能失败，随着服务的增多还会出现更多的潜在故障点。因此，如何提高系统的可靠性、降低因网络引起的故障率，是系统构建的一大挑战。
- **异步：** 异步通信大大增加了功能实现的复杂度，并且伴随着定位难、调试难等问题。
- **数据一致性：** 要保证分布式系统的数据强一致性，成本是非常高的，需要在 C（一致性）A（可用性）P（分区容错性） 三者之间做出权衡。

2. 运维成本

运维主要包括配置、部署、监控与告警和日志收集四大方面。微服务架构中，每个服务都需要独立地配置、部署、监控和收集日志，成本呈指数级增长。

3. 自动化部署

在微服务架构中，每个服务都独立部署，交付周期短且频率高，人工部署已经无法适应业务的快速变化。因此如何有效地构建自动化部署体系，是微服务面临的另一个挑战。

4. DevOps 与组织架构

在微服务架构的实施过程中，开发人员和运维人员的角色发生了变化，开发者将承担起整个服务的生命周期的责任，包括部署和监控；而运维则更倾向于顾问式的角色，尽早考虑服务如何部署。因此，按需调整组织架构、构建全功能的团队，也是一个不小的挑战。

5. 服务间的依赖测试

单体架构中，通常使用集成测试来验证依赖是否正常。而在微服务架构中，服务数量众多，每个服务都是独立的业务单元，服务主要通过接口进行交互，如何保证依赖的正常，是测试面临的主要挑战。

6. 服务间的依赖管理

微服务架构中，服务数量众多，如何清晰有效地展示服务间的依赖关系也是个不小的挑战。

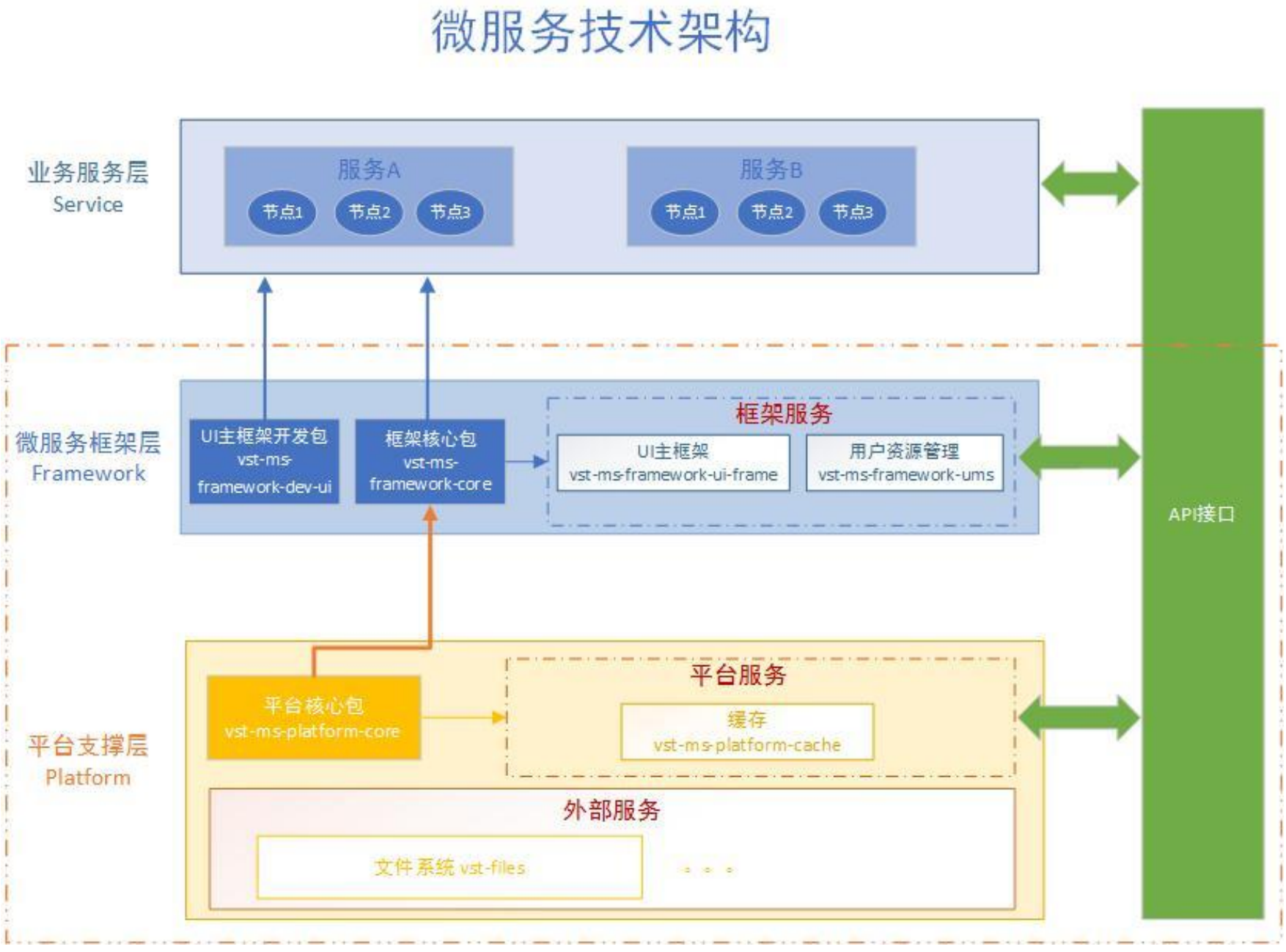
微服务的落地需要经过全面的考察和完善的试验，并不是每个场景都适合使用微服务架构，也不是每个企业都有能力或者精力去面对这些挑战，所以对于管理层来说，是否选择微服务，是个值得商榷的问题。

中台和微服务的关系

中台，简单地说，就是企业级能力的复用，一个种方法论，企业治理思想。
微服务，是可独立开发、维护、部署的小型业务单元，是一种技术架构方式。
可见，中台并不是微服务，中台是一种企业治理思想和方法论，微服务是技术架构方式

中台化的**落地**，需要使用微服务架构
中台强调核心基础能力的建设，基础能力以**原子服务**的形式来建设，并通过将原子服务产品化，支撑业务端各种场景的快速迭代和创新；原子服务和微服务所倡导的服务自闭环思想不谋而合，使得微服务成为实现原子服务的合适架构。支撑业务场景的应用也是通过服务来实现，其生命周期随业务变化需要非常灵活的调整，这也和微服务强调的快速迭代高度一致，所以业务应用服务也适合通过微服务来实现。

中台化系统建设不是一蹴而就的，需要长期动态的演进，加上其技术体系已经在**互联网领域**被证明且相当成熟，其在企业落地、执行的土壤已经具备。


微服务技术架构
业务服务层
Service
服务A
节点1
节点2
节点3
服务B
节点1
节点2
节点3
微服务框架层
Framework
UI主框架开发包
vst-ms-framework-dev-ui
框架核心包
vst-ms-framework-core
框架服务
UI主框架
vst-ms-framework-ui-frame
用户资源管理
vst-ms-framework-ums
API接口
平台支撑层
Platform
平台核心包
vst-ms-platform-core
平台服务
缓存
vst-ms-platform-cache
外部服务
文件系统 vst-files
. . .

感谢您的聆听！