微前端 概念 && qiankun框架

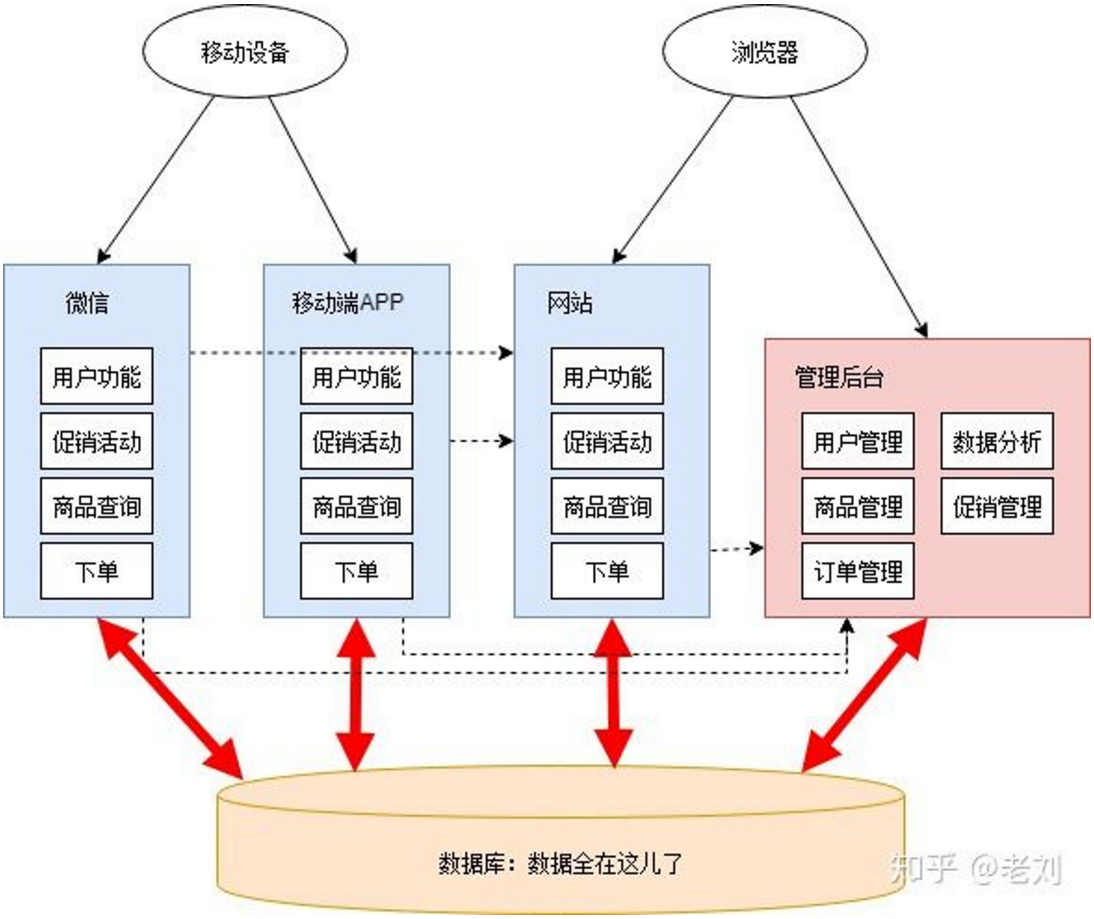
2020年3月12日 15:39

- fengliang

微前端

微服务架构

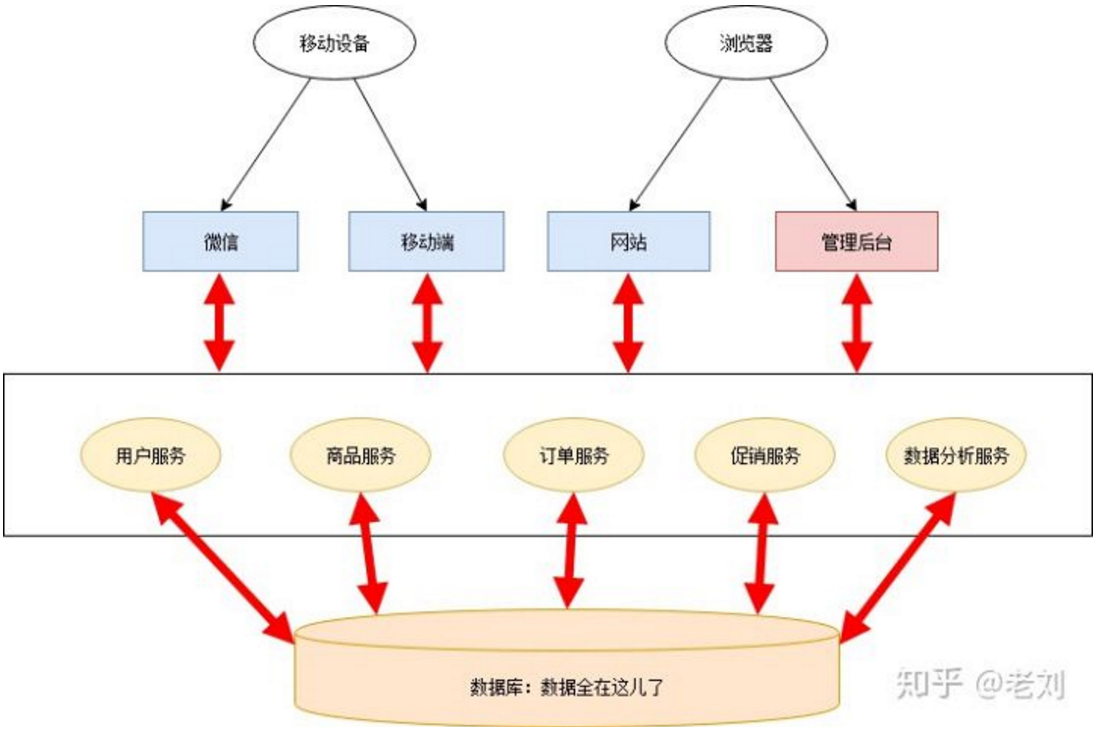
传统单体应用：

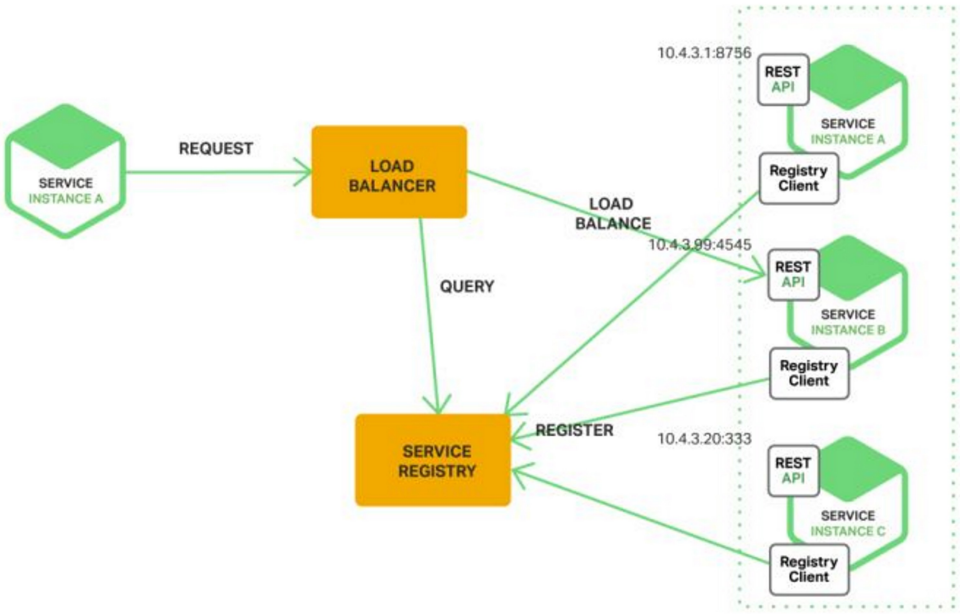


微服务架构：

将每一个功能模块拆分出来，作为一个服务模块，对外提供能力。
提倡将单一应用程序划分成一组小的服务，服务之间相互协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务和服务之间采用轻量级的通信机制相互沟通（通常是基于HTTP的Restful API).每个服务都围绕着具体的业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。
https://www.zhihu.com/question/65502802
（分布式 ） + FaaS(函数即服务)/BaaS(后台即服务)

IaaS PaaS SaaS Serverless (FaaS BaaS)https://juejin.im/post/5d42945ff265da03a715b2f0





优缺点：

优点

1. 提升开发交流，每个服务足够内聚，足够小，代码容易理解；
2. 服务独立测试、部署、升级、发布；
3. 按需定制，每个服务可以根据自己的需要部署到合适的硬件服务器上；
4. 容易迭代新增功能，以及扩大团队，细分团队主导某一业务，可以针对每个服务（service）组件开发团队；
5. 提高容错性（fault isolation），一个服务的内存泄露并不会让整个系统瘫痪；
6. 新技术的应用，系统不会被长期限制在某个技术栈上；

缺点

1. 微服务提高了系统的复杂度；
2. 开发人员要处理分布式系统的复杂性；
3. 服务之间的分布式问题，设计注册，返现，调用，通信，一致性，升级，编排，容灾，隔离等等；
4. 不同服务实例的管理。

微服务 + 前端 = 微前端

原存在的问题

控制台/中台应用： 周期长、业务发展快 -> 模块多，耦合大， 难以理解 -> 巨石应用：难以维护、开发成本大、新增需求困难

价值：

微前端架构旨在解决单体应用在一个相对长的时间跨度下，由于参与的人员、团队的增多、变迁，从一个普通应用演变成一个巨石应用(Frontend Monolith)后，随之而来的应用不可维护的问题。

1. **技术栈无关：** 子应用间技术栈不需要在统一，只需要接入主应用
2. **独立开发、独立部署：**子应用可以进行独立开发，开发完成后接入主应用；分模块部署，子应用可以独立部署在不同的服务器上，独立部署后主框架自动完成更新。
3. **独立运行时：**每个子应用之间的state相护隔离，运行时不共享，也就是子应用的状态不会受到影响。

微前端的现状以及一些微前端框架实现中需要考虑到的技术问题：
https://tech.antfin.com/community/articles/536

框架 qiankun

github：https://github.com/umijs/qiankun

qiankun怎么实现微服务的 - 待研究

Examples - 例子