微服务的设计思考

寇宇

2017/12

PART ONE

01

微服务的设计

02

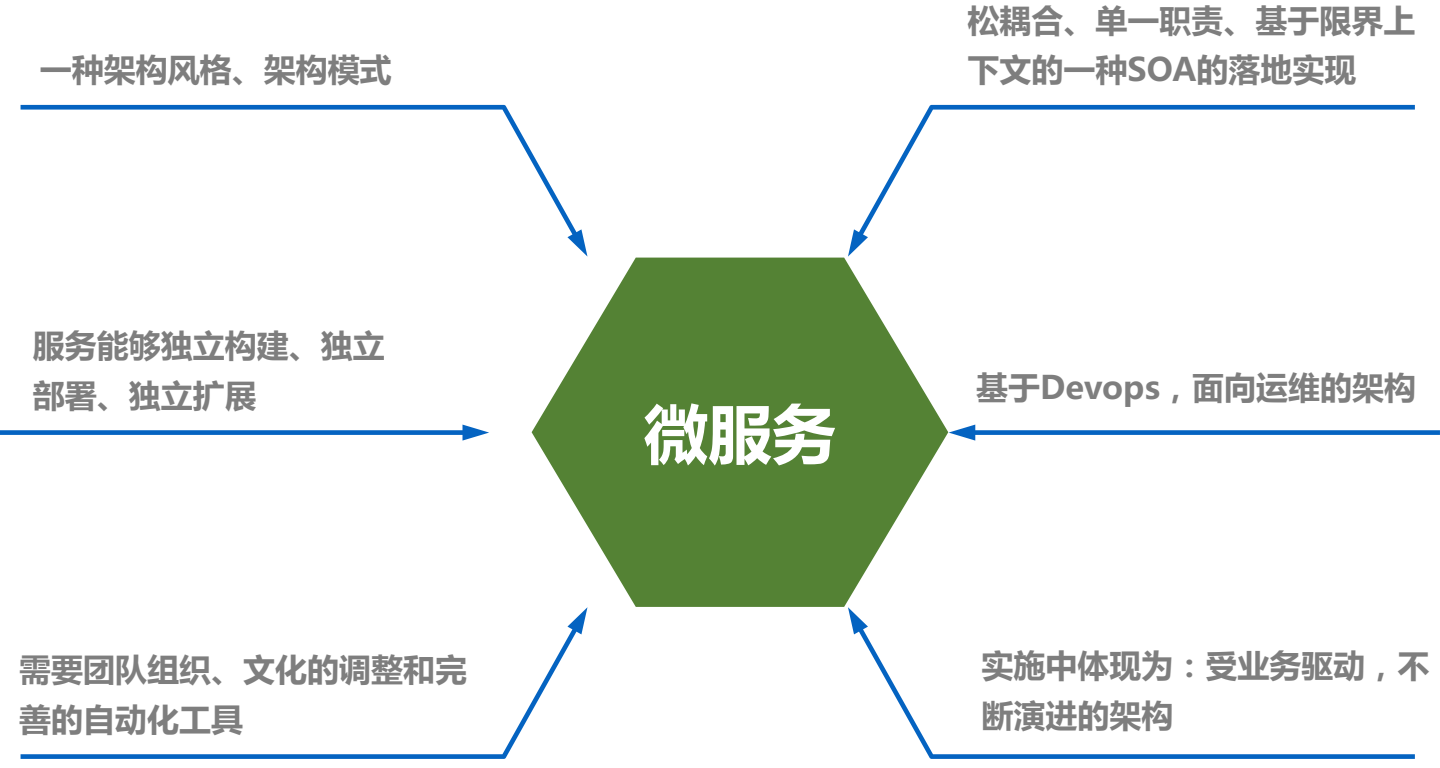
微服务的架构模式

03

微服务的的监控

PART ONE

微服务的设计： 概念



PART ONE

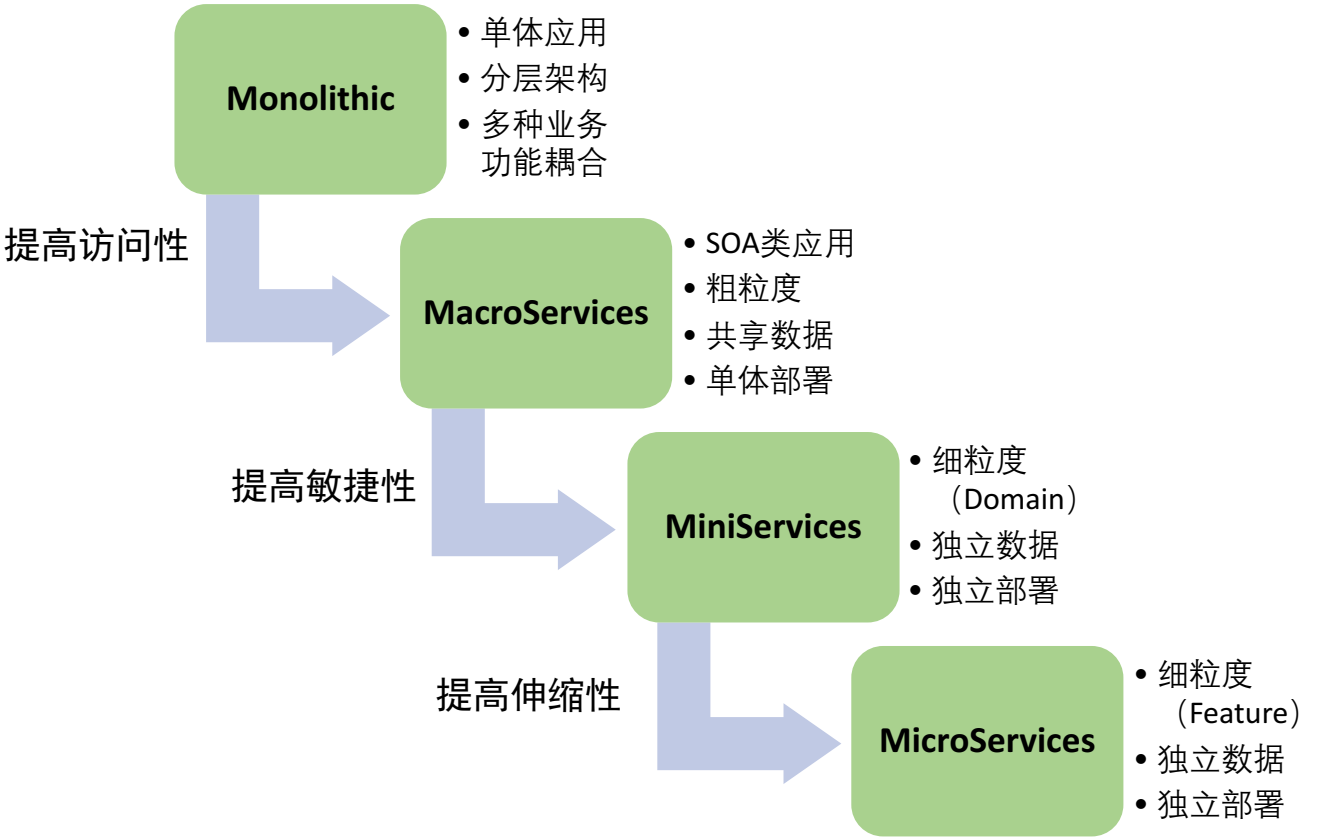
微服务的设计

常见误区：

- 我使用了Springboot或Dubbo等，所以我使用了微服务
- 微服务有助于提升应用性能
- 微服务只是一种新的架构模式，开发中改变下架构与设计方法就可以做到微服务
- 我使用了 Docker容器，所以我使用了微服务
- 或者，我们没上容器，所以没法使用微服务
- 通过在微服务框架上开发微服务，仍可以保证事务的实现

微服务的设计

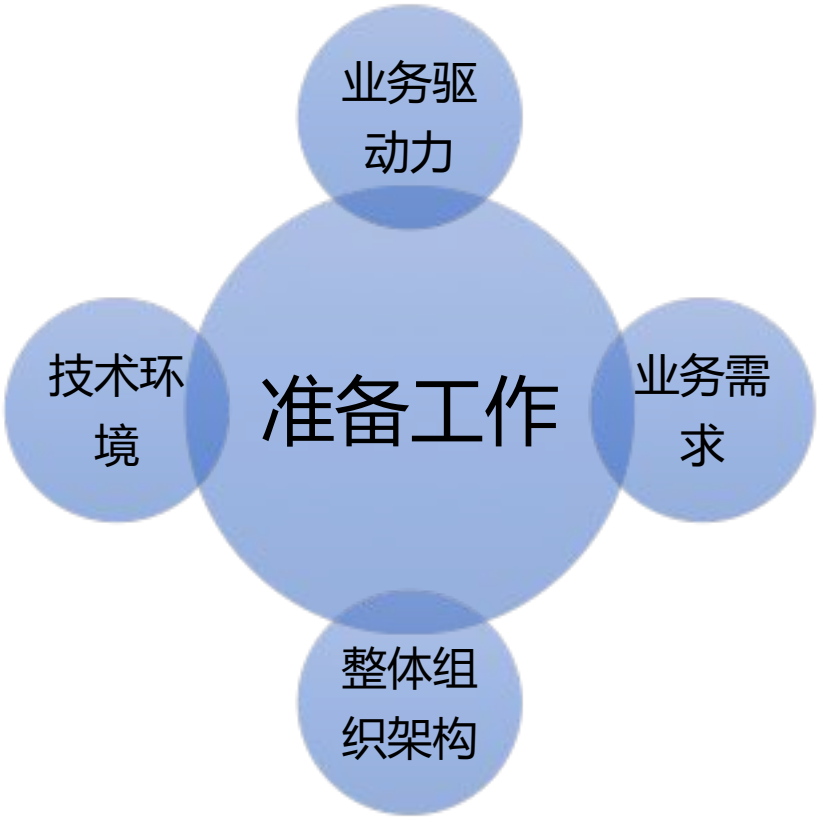
微服务构建的演进



PART ONE

关键问题（一）：该用微服务吗

- Do right things! 业务上真的有需要吗？
- 微服务不是“银弹”，并不适合于每个应用和所有环境；
- 原则：最好不拆！
- 何时采用微服务
  - 业务响应速度已受到严重影响，现有常规办法已无效果
  - 现有架构下，再怎样加硬件也无法改善应用指标
  - …



PART ONE

关键问题（二）：怎样设计出微服务

单体应用的分解方法： **拆**

策略

第一步：构建所有的新加特性作为微服务

不摧毁应用，也不加入新功能，而是使用微服务方式实现新特性

集成新的微服务：anti-corruption layer，隔离旧应用，提高扩展性

第二步：“扼杀旧应用”

不断地提取微服务，直到应用中全部的”限界上下文”都提取为微服务或其中所剩内容已无必要再提取。

提取组件为服务的标准：通过区分”限界上下文”，形成微服务

标准1：识别整体架构内的”限界上下文”,把不一致概念的分开。

标准2：处理优先级。在候选功能中，是否是优先的功能提取？

PART ONE

关键问题（三）：服务拆到什么程度

微服务的拆解粒度：how small is“small”？

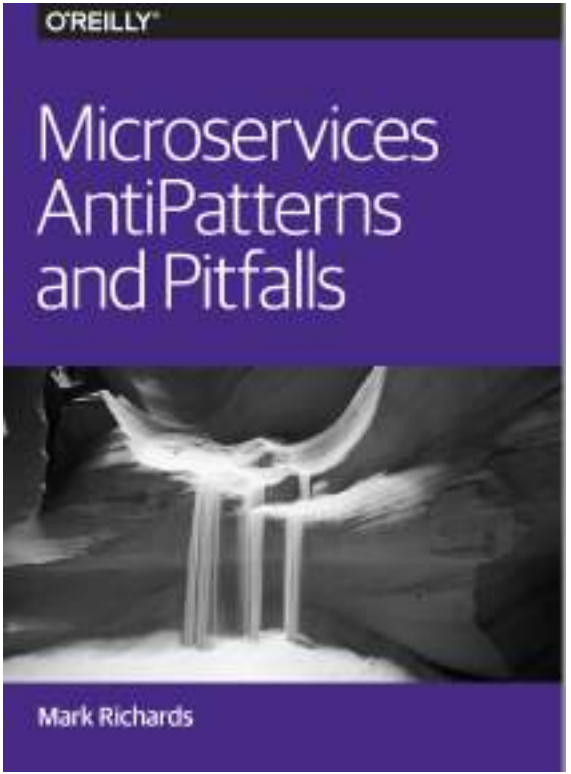
- 最佳实践：
  - 先粗后细：开始拆解时，很难一次性给出合适的粒度，可以先划分的粗些。
  - 不断调整：当对服务有了更多认识后，会不断调整粒度，进行服务的进一步拆分、合并。

- “类”与“服务”：类的数量不是粒度衡量的标准
  - 服务实际上是指服务组件，被认为是承担特定职责的架构组件；
  - 服务组件怎么实现和用多少类实现，要根据设计情况定；

- 确定服务粒度的基准测试
  - 服务的范围与功能：分析服务提供的操作的内聚层次，拆分指示词，“并且”、“此外”
  - 数据库事务：分布式的影响，ACID vs.BASE transactions ，是否服务粒度过细
  - 分析服务编织的层次：编织会降低整体性能；影响可用性与健壮性。太多的编织意味着服务粒度过细。请求响应能力与可靠性间的权衡
  - 考虑组织文化、团队规模：Two-pizza Team,Cross

PART ONE

关键问题（四）：反模式

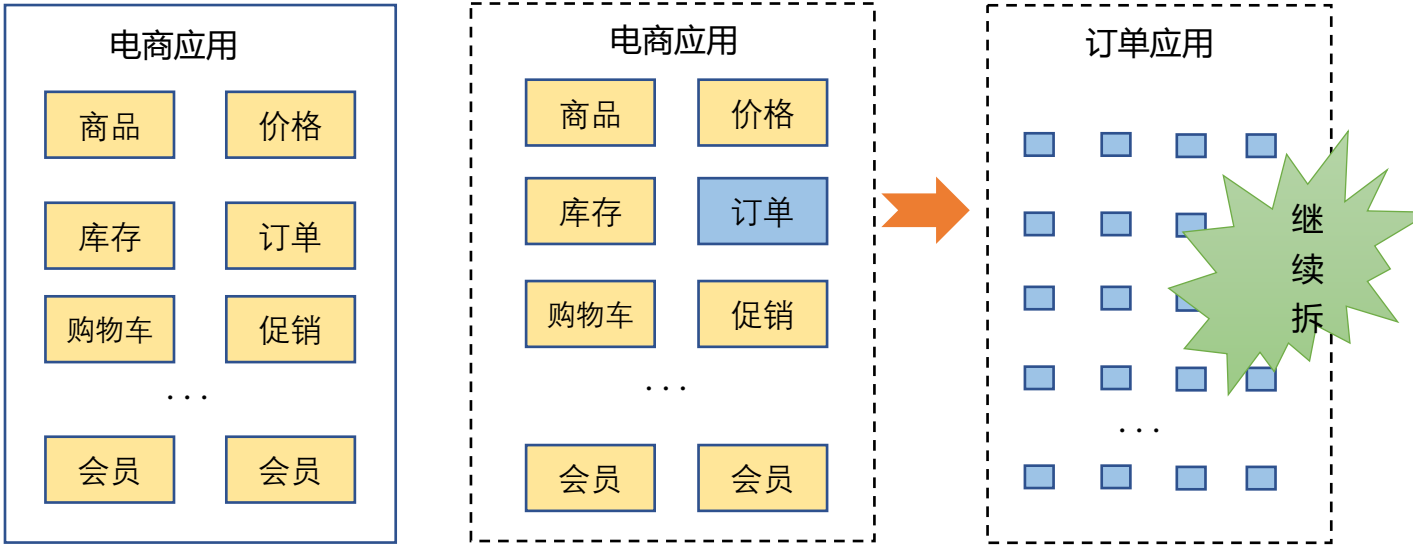


- “数据驱动迁移”反模式：Functionality First, Data Last
- “共享”反模式：打破了服务间的限界上下文
- “超时”反模式
- “Rest”陷阱
- “静态契约”陷阱
- …

PART ONE

微服务的设计： 服务拆分举例

因应业务发展而不断演变！



由一个商业套件实现全部应用功能

历经3-4年

采用SOA模式，整合各定制的单一分层应用

历经2-3年

开始按照限界上下文进行服务拆分，但粒度较粗

PART ONE

微服务的设计： 服务拆分举例

- 业务驱动力：
  - 单体应用性能差，越来越难以通过硬件扩展来提升服务水平
  - 难以快速开发、全量回归测试困难、难以快速部署上线，影响公司业务发展；
  - 希望大幅提升订单的开发效率，易于快速开发、快速测试，降低复杂度；
- 业务需求：
  - 接单：近200种场景的接单；
  - 审核与资源处理：处理会员权益、促销资格、价格、优惠、库存等；
  - 交易处理：支付相关操作；
  - 查询：按多维度；
  - 分发：同步必要的订单信息；
- 技术环境：
  - 基于虚机的私有云环境；
  - 处理单元化（可在分区内完成全部处理），利于跨机房部署；

PART ONE

微服务的设计： 服务拆分举例

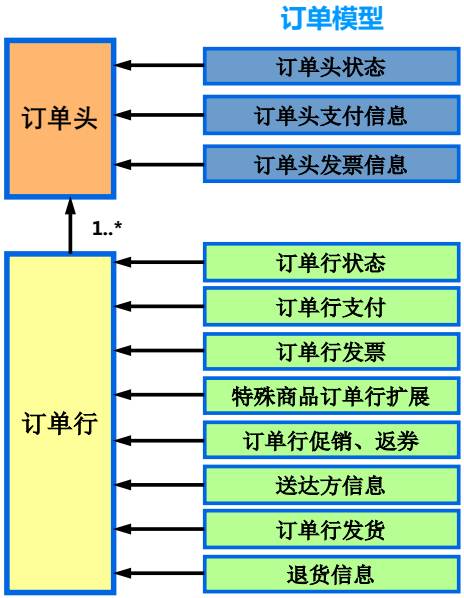
订单的场景：

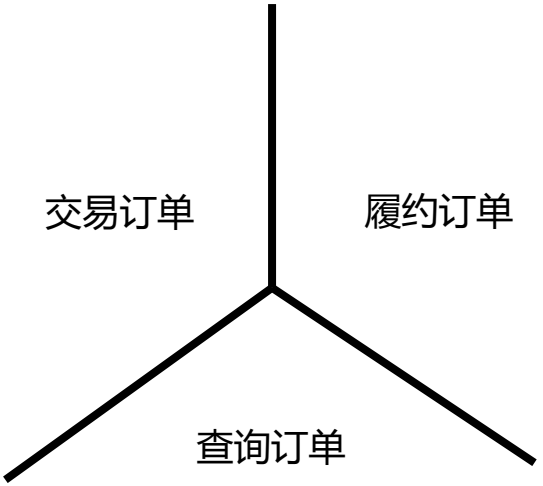
1.来源渠道	线下门店	线上	批发	对公/工程
2.下单终端	PC	门店	APP/WAP	电销
3.业态分类	零售	. . .		
4.业态来源	电器	超市		
5.商品分类	实体商品	虚拟商品	服务商品	
6.经营模式	自营	第三方	. . .	
7.配送方式	自营配送	商家配送	厂家配送	门店自提
8.支付方式	在线支付	货到付款	融合支付	
9.支付次数	一次支付	二次支付		

10.顾客	个人	企业	
11.自营销售方式	先销后采	先采后销	
12.线上购物	正常	抢购	闪拍
	名品特卖	海外购	
13.销售方式	正常	套餐	赠品
14.商品特性	正常	冷链	送装一体
15.发票	不打发票	电子发票	普通发票
16.结算	正常	分账	

PART ONE

微服务的设计： 服务拆分举例





PART ONE

微服务的设计： 服务拆分举例

交易服务

- 下单；
- 拆单；
- 校验；
- 支付；
- …

履约服务

- 库存调度
- 物流调度
- 售后服务调度
- …

查询服务

- 按用户查询；
- 按营业员查询；
- 按手机号查询；
- …

未来：使用Stratety模式，细分服务！

Agenda

01

微服务的设计

02

微服务的架构模式

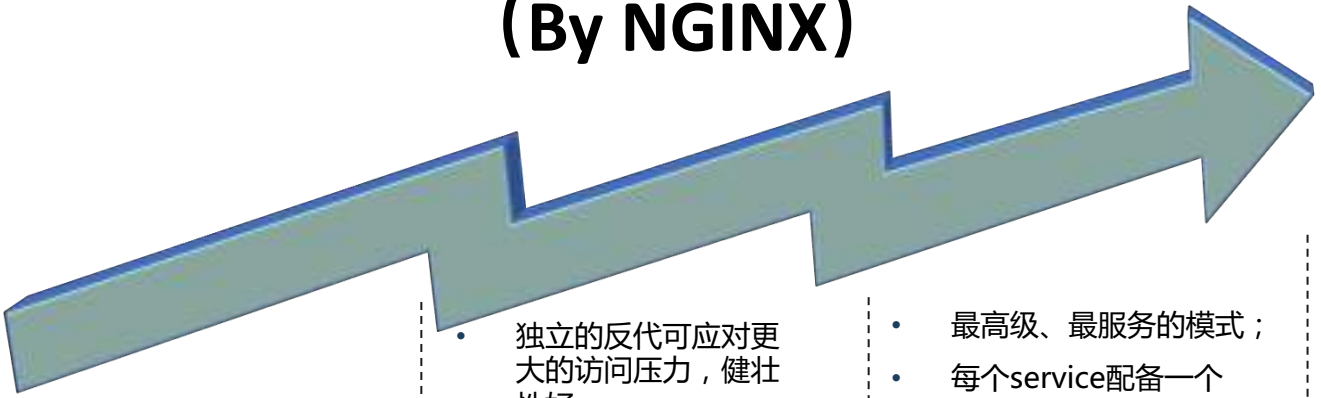
03

微服务的的监控

PART TWO

微服务的架构模式

Microservices Reference Architecture（By NGINX）



- Proxy作为API网关与控制器；
- 简单易行；
- 适用于从复杂度适中的单体应用转向微服务架构的起步阶段；

Proxy Model

- 独立的反代可应对更大的访问压力，健壮性好；
- router mesh hub处理服务间通信；
- 提供了更多的控制手段；
- 适用于从更大更复杂的单体应用转为微服务；

Router Mesh Model

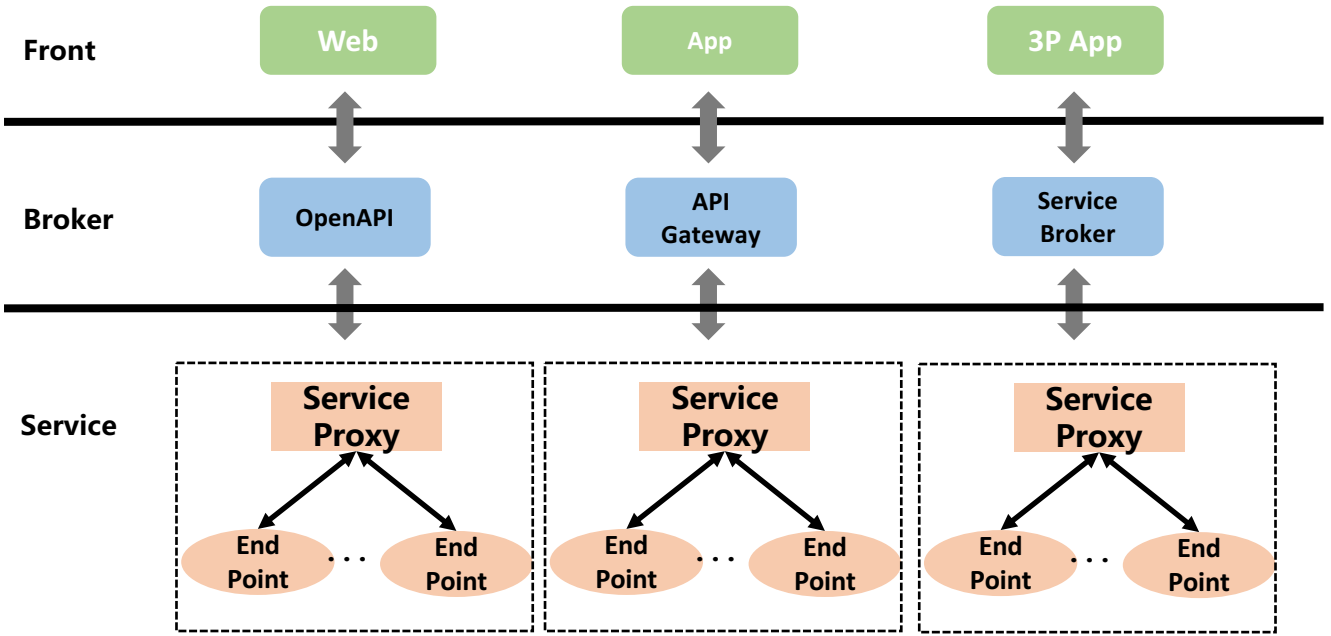
- 最高级、最服务的模式；
- 每个service配备一个Proxy；
- 服务间通信通过service Proxy，对服务进行有针对性的治理；
- 高性能、高弹性；
- 适用于高压力的场景；

Fabric Model

PART TWO

微服务的架构模式

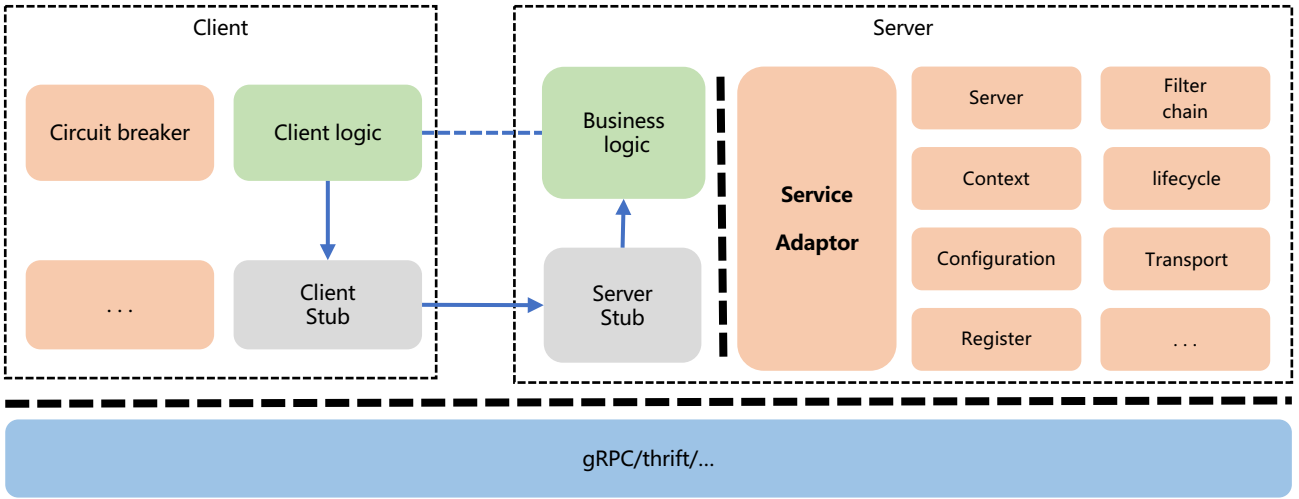
微服务框架的一种实现



PART TWO

微服务的架构模式

微服务框架的一种实现



- 一个协议无关的服务框架
- 基于契约优先方式开发服务接口
- 对远程调用协议进行了封装，优化了码生成的结构与调用方式，即采用形如“同步”的编码方式实来进行远程调用
- 提供LifeCycle，添加了服务注册，基于zipkin的调用链等功能
- 添加了Spring Starter，简化了服务启动和客户端调用

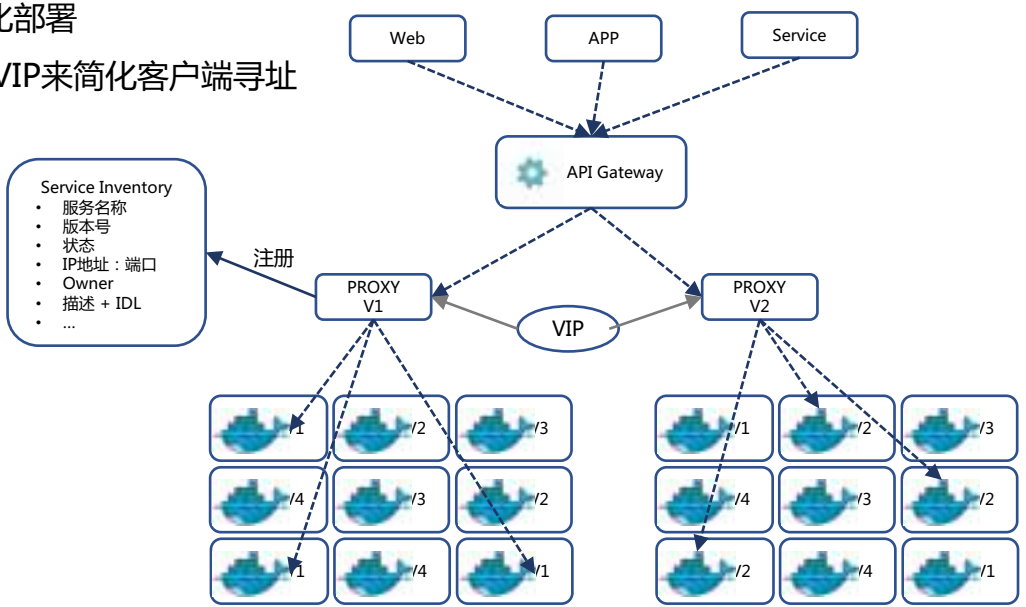
业务逻辑

通过IDL生成

微服务的架构模式

基于Proxy的服务端治理

- Proxy是Client访问的端点
- Proxy负责服务实例的信息收集和注册
- 基于Proxy的路由功能结合**语义化版本(X.Y.Z)**的概念进行不同服务的版本管理
- 利用Docker简化部署
- 利用Proxy绑定VIP来简化客户端寻址



Agenda

01

微服务的设计

02

微服务的架构模式

03

微服务的的监控

PART THREE

微服务的监控

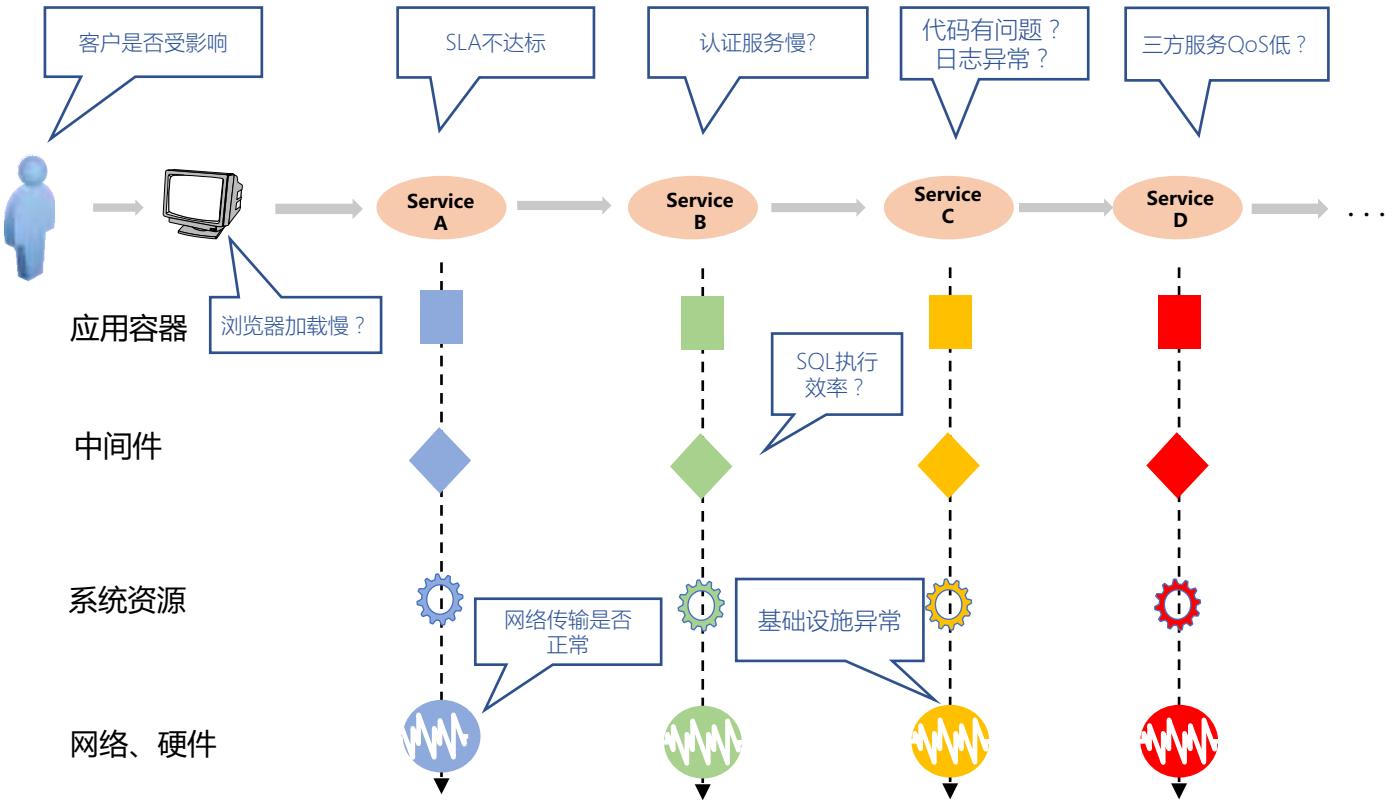
微服务的规模和动态性使得数据收集的成本大幅度提高，例如（cpu、内存和网络传输的开销）

大量的监控数据对后台数据处理分析的产生影响

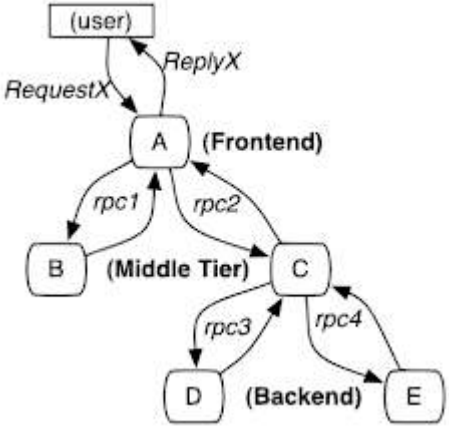
对于可视化和关联分析的要求方面，传统APM缺少好的手段

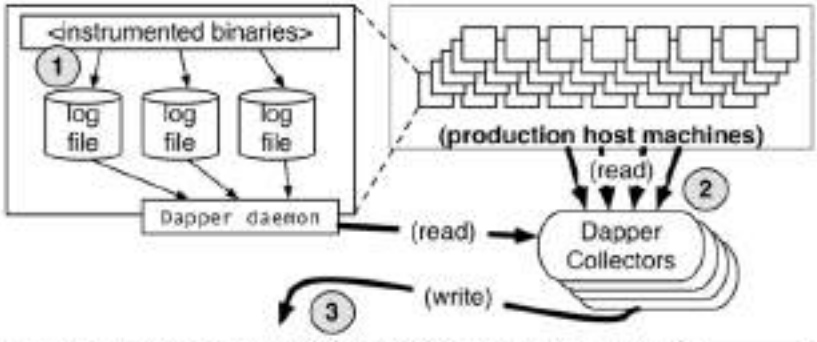
PART THREE

微服务的监控：端到端的全链路监控



分布式追踪 – Google Dapper



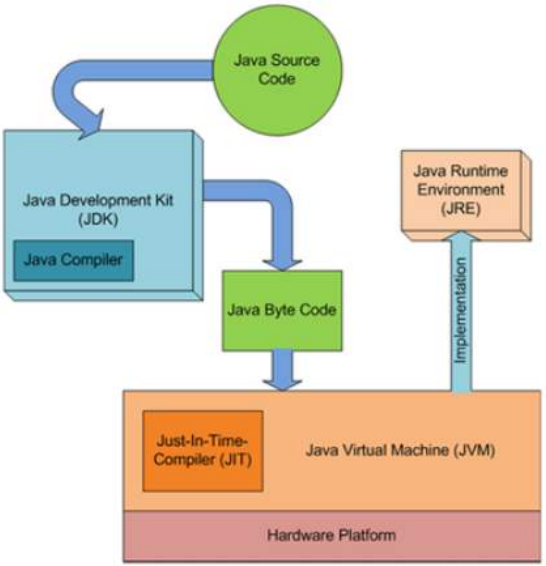


trace id	span 12	span 23	span 34	span 45	span 56	...
123456	*nil*	*nil*	**<data>**	**<data>**	*nil*	...
246802	**<data>**	*nil*	*nil*	*nil*	**<data>**	...
357913	*nil*	**<data>**	*nil*	*nil*	*nil*	...
...	...	...	...	...	...	...

(Central Bigtable repository for trace data)

Figure 5: An overview of the Dapper collection pipeline.

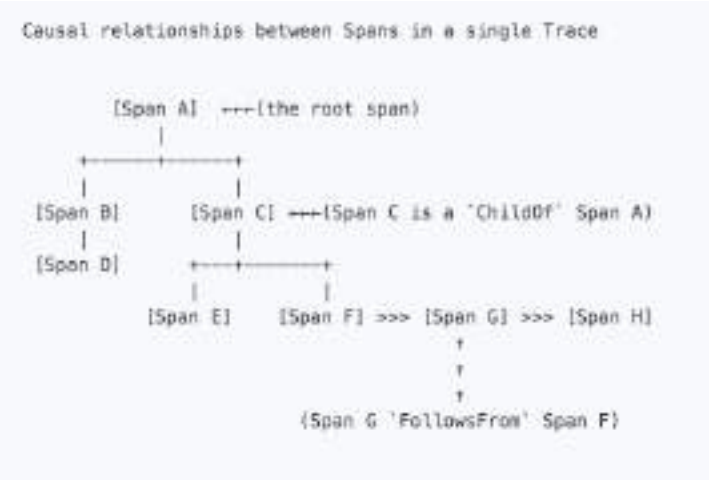
APM探针的基本原理 (Java Instrument)

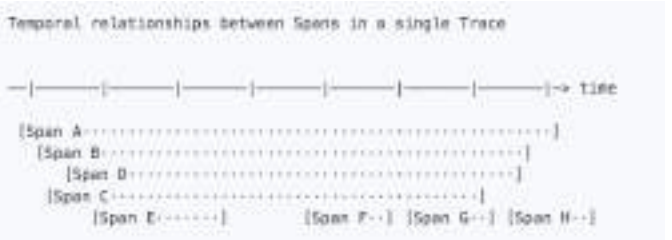


PART THREE

微服务的监控

分布式追踪 – OpenTracing



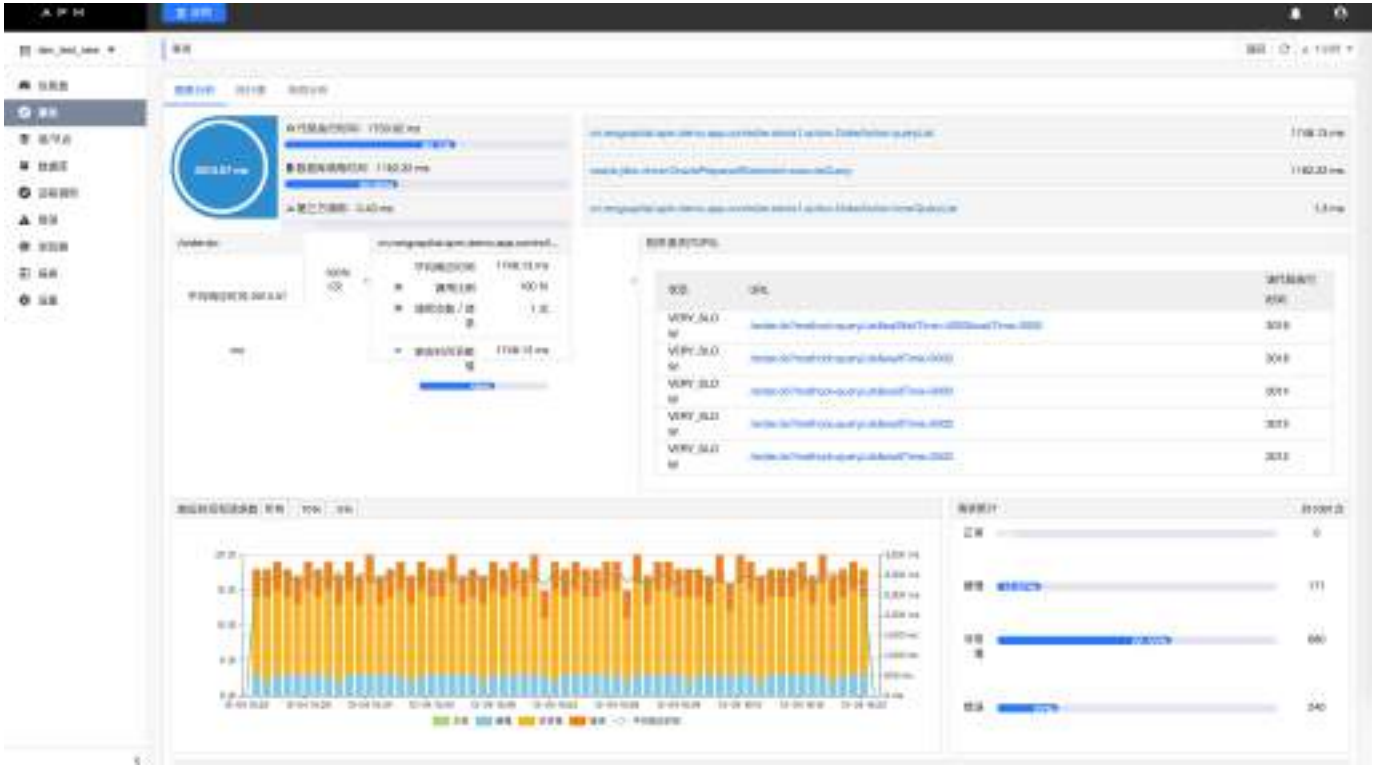


PART THREE

微服务的监控：动态拓扑

PART THREE

微服务的监控：事务分析



THANKS