Service Mesh + Java
能干哪些牛X的事

MAKING SOFTWARE ARCHITECTURE ALIVE

MegaEase 陈皓

企业云化架构解决方案提供商/用技术推动商业进步

个人简介

Self-Introduction

MegaEase
企业云化架构提供商

- **20+年工作经验，超大型分布式系统基础架构研发和设计**
- **擅长领域：金融、电子商务、云计算、大数据**
- **职业背景**
  - 阿里巴巴资深架构师（阿里云、天猫、淘宝）
  - 亚马逊高级研发经理（AWS、全球购、商品需求预测）
  - 汤森路透资深架构师（实时金融数据处理基础架构）
- **目前创业MegaEase，致力于为企业提供技术架构产品**
  - 支撑高可用高并发高性能的分布式系统架构
  - 为40+公司提供过软件技术服务

Weibo: @左耳朵耗子
Twitter: @haoel
Blog: coolshell.cn

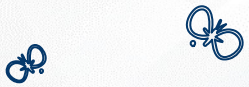
目录

CONTENT

新一代的数字化转型

The New Generation of digital transformation

MegaEase
企业云化架构提供商



企业内部数字化

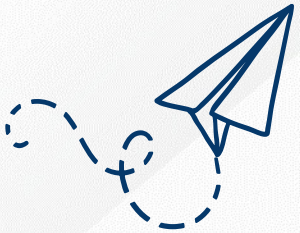
ERP， CRM， OA……

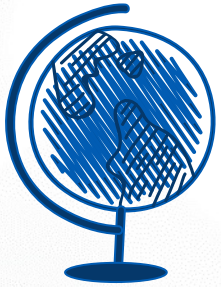
传统数字化主要完成了企业本身的数字化需求，以满足自身和管理为主

重心转移

用户需求数字化

用户营销、用户体验、用户行为，推荐系统……未来的数字化主要是用来感知用户和需求变化





企业数字化重心转移后的对技术的诉求

Business Uses Cases & Requirements

MegaEase
企业云化架构提供商

支持更多的用户活动（高并发）

商品秒杀、大促、抽奖、分销以及其它大规模营销越来越会成为获得客户的重要手段。但它会为网站带来超大规模的流量和负载。这需要后端的分布式架构和自动化流量和资源的调度。

快速的生产流水线（实时发布）

要适应高速变化的市场、用户需求、以及竞争对手，需要建立高速的软件和技术生产流水线。这需要持续发布的DevOps，和高度可重用的技术和业务中台的建设。

技术自主可控（自由扩展定制）

技术自主可控越来越成为企业的核心竞争力，不被厂商绑架，拥有自己可控的技术协议和技术架构，才可能打造出更适合自己而且更高效的团队工程能力。

高可用及高稳定性（应对故障）

使用分布式架构，冗余结点和异地多活，让整体系统更为的稳定。通过采集、标注和分析数据，加以服务容错能力，能够做到自动化容忍和修复故障，让整个系统在错误下正常运行。

开放建立生态系统（进入场景）

通过分布式系统的PaaS平台和开放API平台可以很容易地把自己的能力开放出去，让更多的合作伙伴进行系统对接，以建立自己的生态系统，形成整个产业链的业务闭环。

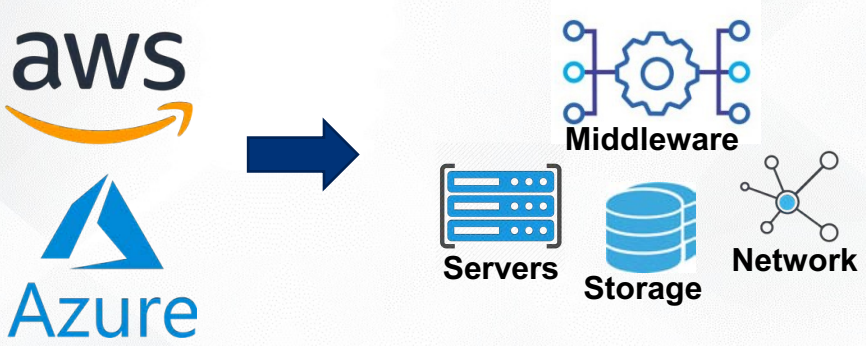
大数据智能运营基础设施（数字化）

通过技术和业务中台把异构系统标准化，可以把相关的数据规格化标准化，形成真正的大数据能力，为智能力运营打到坚实基础。

云计算架构的演进

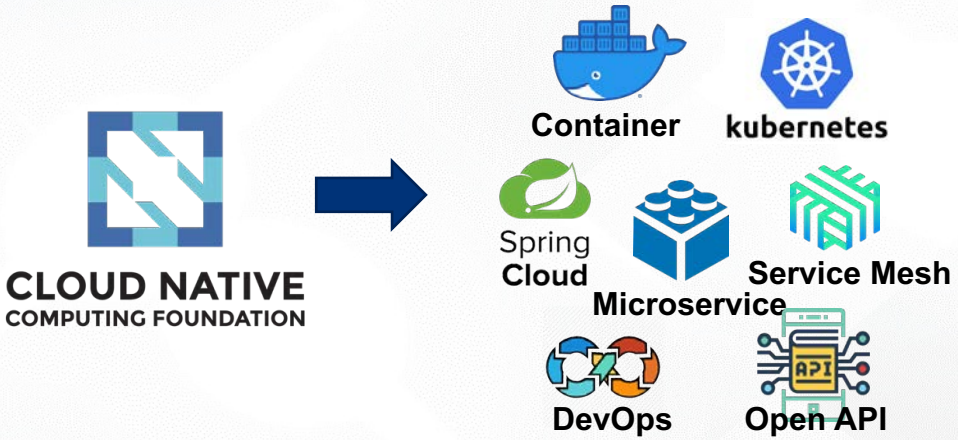
The Evolution of Cloud Computing

MegaEase
企业云化架构提供商



PaaS
IaaS

资源型（运维） Cloud 1.0



SaaS
PaaS

服务型（应用） Cloud 2.0

开源技术的蓬勃发展带来与公有云厂商的对标逐渐展开，已经进入cloud 2.0 时代数字经济时代

一个经典的例子

One Powerful Example

Google

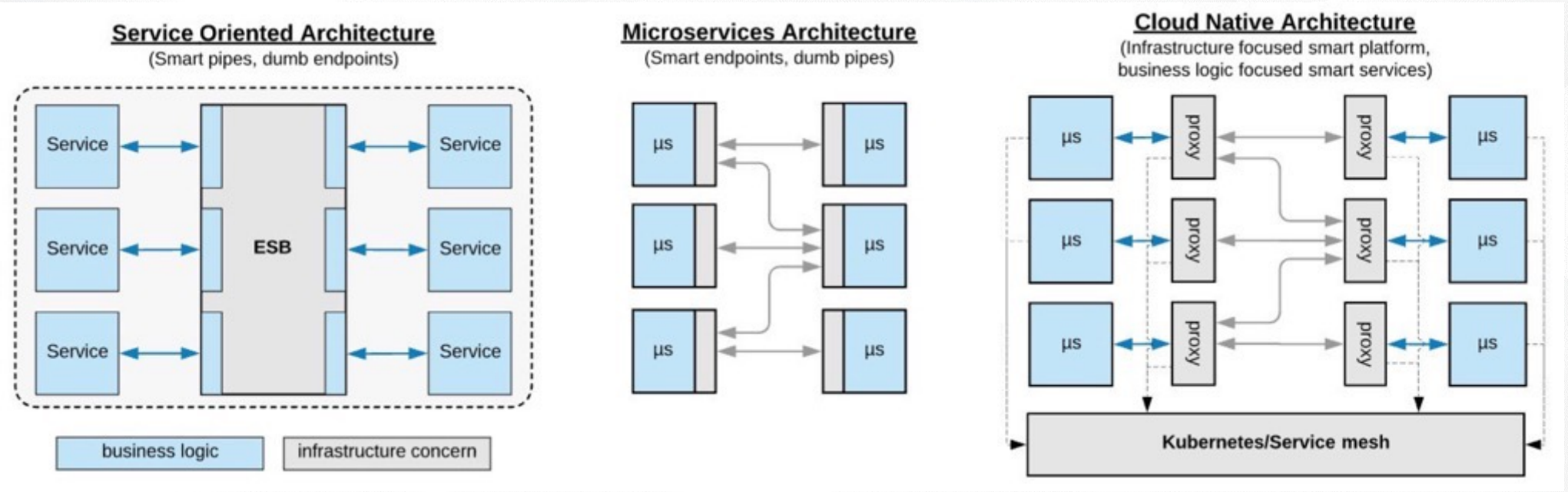
上世纪90年代，Google 运行在廉价不稳定的x86 + Linux 上

软件架构做得好，基础设施变得不重要了

企业服务化架构的演进

The Service Architecture Evolution

MegaEase
企业云化架构提供商



中心化的SOA架构

ESB中即有业务逻辑也有控制逻辑

ESB作为一个万能中间件可以进行服务注册发现、路由、限流、熔断、重试、监控、 认证、协议转换……

微服务架构

控制逻辑反过来入侵业务逻辑

每个微服务都有一个SDK完成服务注册发现、路由、限流、熔断、重试、监控、认证、协议转换……

服务网格架构

控制逻辑与业务逻辑分离

每个微服务都有一个边车，并由Kubernetes 进行管理和调度，完成服务注册发现、路由、限流、熔断、重试、监控、认证、协议转换……

服务化架构在折腾什么？

What's the Service-Oriented Architecture Looking for?

MegaEase
企业云化架构提供商

可观测性

- 收集并关联数据
- 急诊和体验
- 服务调用链跟踪
- 资源中间件应用关联

服务治理

- 服务注册发现
- 服务配置管理
- 服务健康检查
- 服务弹力容错设计

流量管理

- 流量过滤和保护
- 灰度发布及路由
- 流量着色及调度
- 流量编排和降级

服务化架构的可观测性

The Observability of Service-Oriented Architecture

可观测性的重点

- 不是仅仅只是收集更多的监控数据，而是需要关联数据，数据不关联则没有意义
- 需要找到这样的关联：从API → 服务 → 服务调用链 → 中间件 → 基础资源
- 需要解决的问题是
  - “急诊” - 快速故障定位
  - “体检” - SLA报告 + 容量分析
- 需要关联的数据有
  - 基础资源的的Metrics（如：CPU，Memory, I/O，Network…）
  - 中间件的Metrics, Logs（如：JVM、Redis、MySQL、Kafka、Proxy/Gateway…）
  - 应用的 Metrices（如：吞吐量，响应时间，错误率，等等）
  - 应用的 Logs（如：Access Log, Application log, Throughput, Latency, Error…）
  - API的调用链跟踪（需要穿过应用内的异步调用，消息中间件）

微服务下的流量治理

The Traffic Orchestration of Service-Oriented Architecture

流量治理的重点

- 流量保护 – 按吞吐量限流，按响应时间限流
- 流量着色 – 能够进行流量着色处理，以便后台服务可以进行流量调度（把不同的流量分配给不同的服务实例）
- 灰度发布 – 能够进行七层的流量规则路由
- 流量过滤 – 能够进行流量过滤，比如：协议校验、白名单管理、权限认证……
- 流量编排 – 可以进行API流量编排、聚合
- 流量降级 – 可以对请求进行降级（使用降级版本的API，通过API响应缓存降低请求……）
- 容错管理 – 对后端的服务进行负载均衡，熔断，重试等操作

服务化架构的服务治理

The Observability of Service-Oriented Architecture

MegaEase
企业云化架构提供商

服务治理的重点

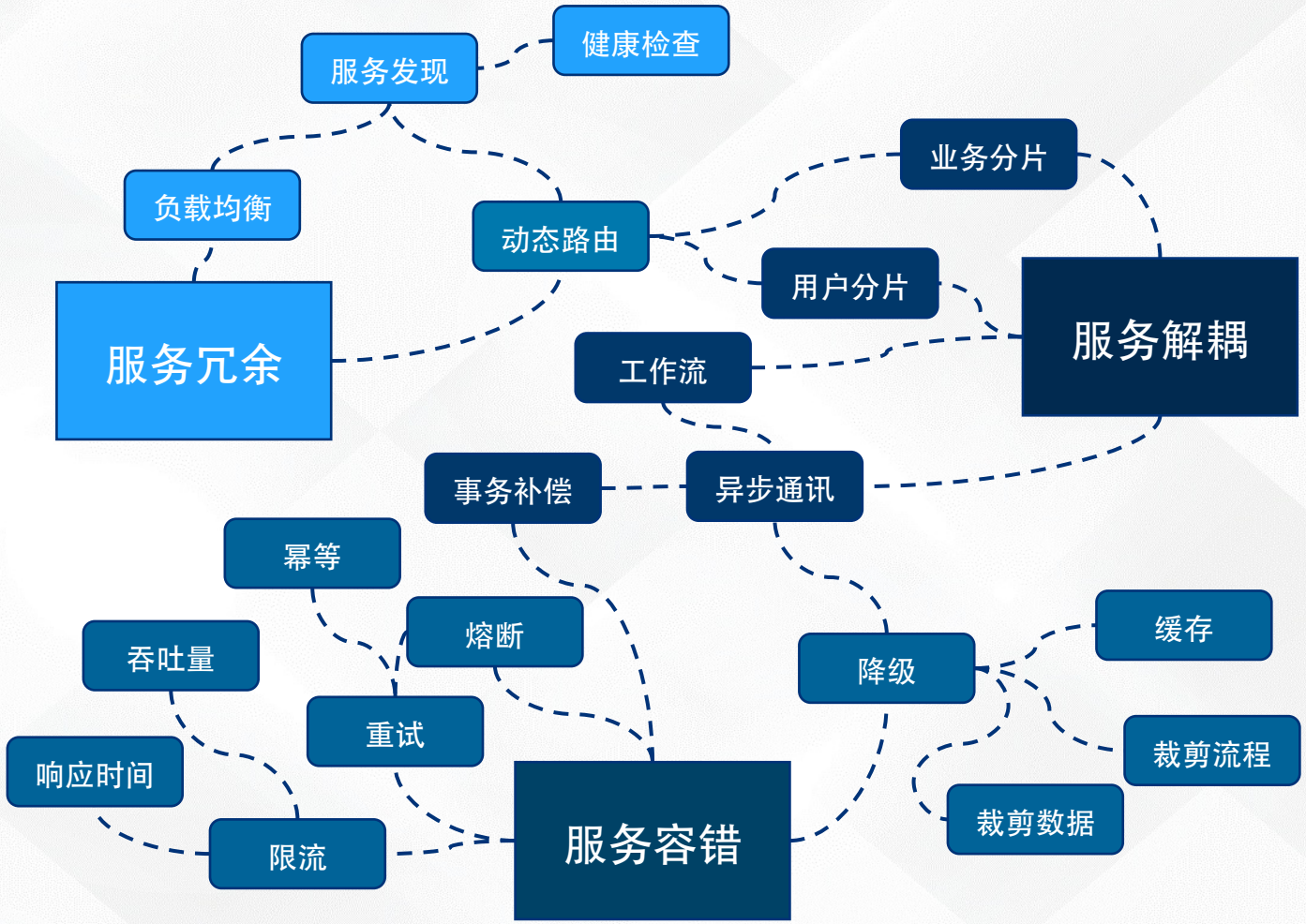
服务注册发现
- 服务需要进行注册和发现，因为需要动态的对服务进行管理

服务配置管理
- CMDB必需是服务为视角的，而不是资源为视角。
- 为了支持灰度发布，服务配置也需要是多版本的

服务健康检查
- 服务的健康检查 不是生死检查

服务的弹力及容错设计
- 服务流量治理和灰度发布
- 限流、重试、幂等、熔断、降级……

服务的编排和管理
- 服务间的异步通讯（Event Driven Architecture）
- 服务间的编排（工作流编排，API聚合）
- 服务的事务管理（TCC，事务补偿）



经典微服务架构 vs 新型服务网格架构

Classical Microservice Architecture vs Service Mesh Architecture?

MegaEase
企业云化架构提供商

	经典微服务架构	服务网格架构
技术栈	主要是基于Java, Spring boot 和 Spring Cloud	主要是基于 Kubernetes 的基本设施
服务发现	Eureka, Consul, Nacos, Zookeeper	Kubernetes DNS
配置管理	Apollo, Nacos, Spring Config server	Kubernetes ConfigMap & Secrets
健康检查	Spring Boot Actuator	Kubernetes Readiness & Liveness（可能导致重启）
弹力容错设计	SDK - Recilient4j, Sentinel	Sidecar - Envoy
服务网关	Zuul, Spring Cloud Gateway，Easegress	Kubernetes Service, Ingress Controller
API 网关	Envoy, Kong, Openresty, API Six, Easegress…	Ingress Controller
监控数据	JavaAgent, telegraf, EFK, Zipkins, Opentracing, Prometheus, InfluxDB, Grafana……	
自动化运维	通过Kubernetes 或 Docker	必须基于 Kubernetes

传统的 Service Mesh

The Pros/ Cons of Traditional Service Mesh

支持多语言

- 支持Python, Golang, Java 等

社区活跃度好

- 有很好的影响力，以及一定的标准化

不错的运维体验

- 丰富的运维命令

不完整的观测性

- 较弱的观测性能力， 缺失很多的关键指标数据
- 需要用户自己完成调用链追踪的功能，边车才会帮助传递

不完整的流量调度能力

- 流量着色并不是应用层的方案，在应用上不完整
- 对于灰度发布的策略，很难完成基于用户标签的真正的灰度

复杂的流量劫持

- 基于iptables的流量劫持，相当难Debug，也很难管理
- 消耗内核CPU，相当于操作系统级的Full GC

一个接地气的Service Mesh应该怎么做？

What's features need to be done for a great Service Mesh?

MegaEase
企业云化架构提供商

兼容于主流的业务成熟方案

兼容于主流Java技术栈的服务治理
零成本应用服务迁移及改造

完整的无侵入式的观测性

从应用内部的关键指标
到应用外部调用链的追踪

更好的流量劫持方案

网络层可以使用eBPF
应用层仅劫持服务注册/发现

完整的流量调度

在整个调用链传递着色流量标签
基于用户标签的灰度发布

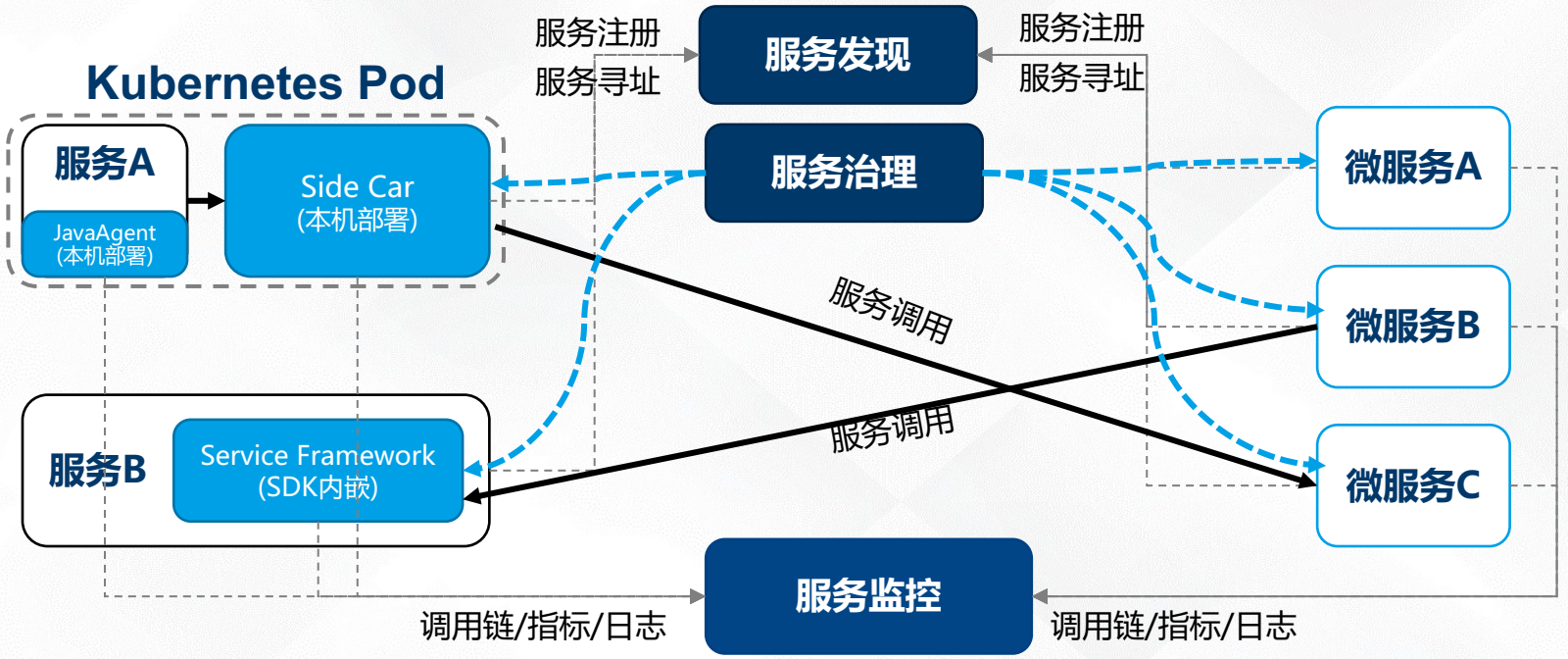
如何让Java应用无缝迁移

How to transfer the

MegaEase
企业云化架构提供商

Mesh方式

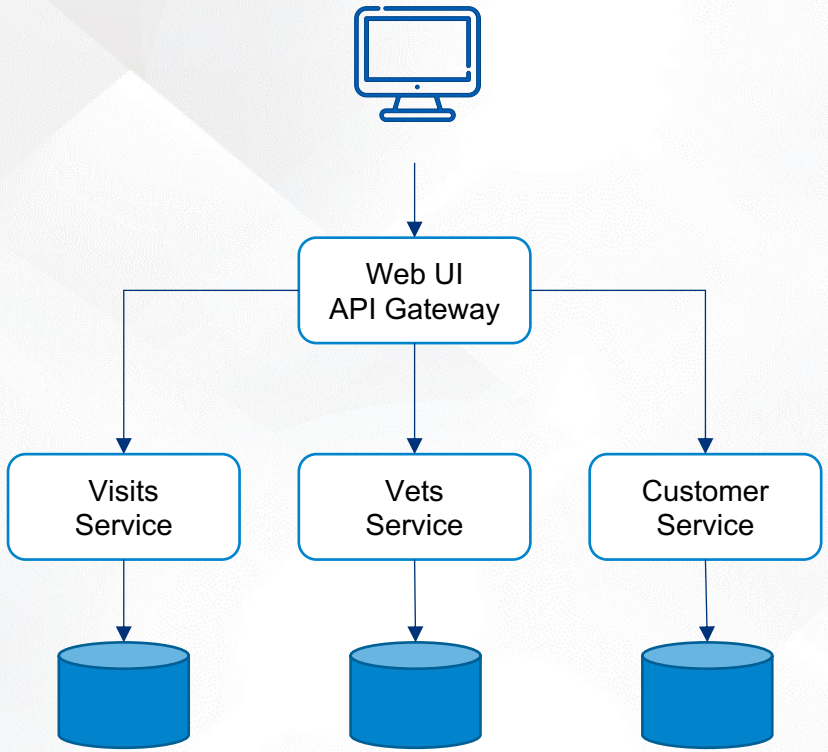
SDK方式

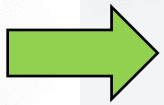


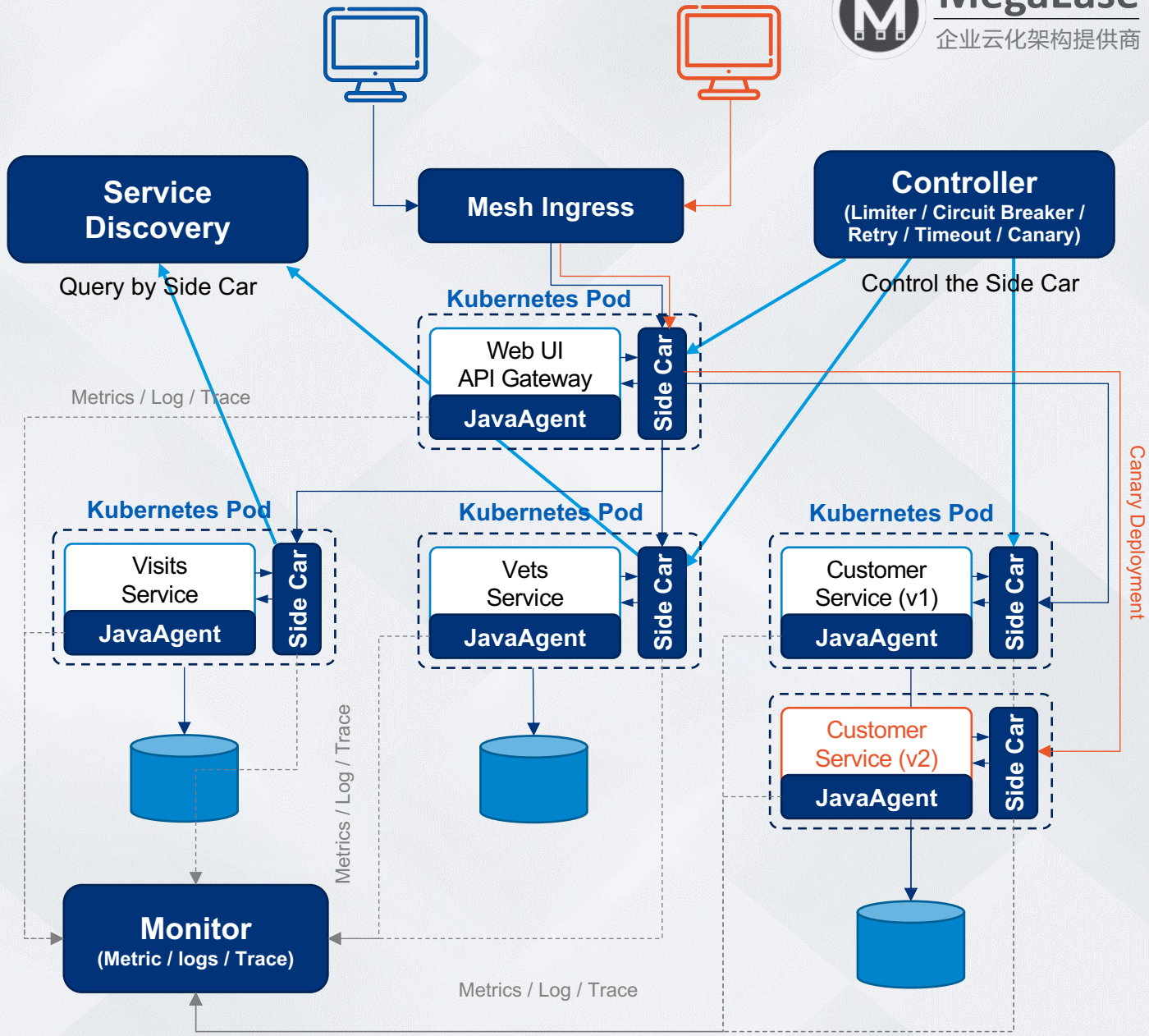
- 通过在服务的机器上安装一个Side Car边车服务
  - 兼容于Java技术栈的主流的服务发现协议
  - 劫持【服务A】的进站出站的流量，进行限流，熔断，重试、降级等服务治理。
- 通过Java Agent的字节码注入技术配合Side Car边车
  - 对【服务A】的内部情况进行观测，追踪服务调用链，采集应用运行的指标和调用中间件的数据
- 这样就可以把一个服务在无侵入的情况下进行服务治理和应用监控

Ease Mesh 工作原理

The mechanism of Ease Mesh







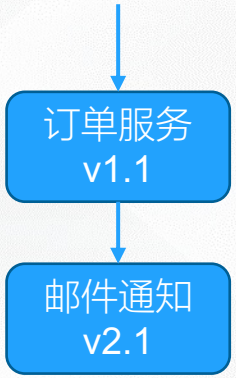
灰度发布的挑战 – 同时灰度发布

The Challenge of Canary Release

增加订单信息

- 修改订单服务
- 修改邮件通知格式

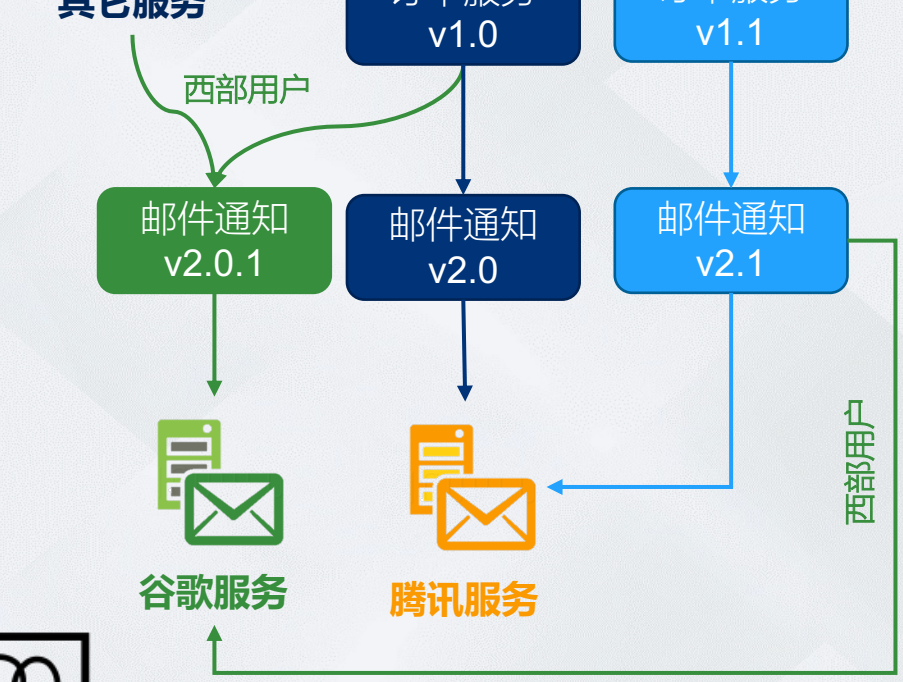
苹果用户灰度

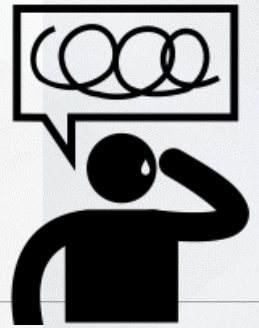


更换邮件提供商

- 需要修改邮件服务

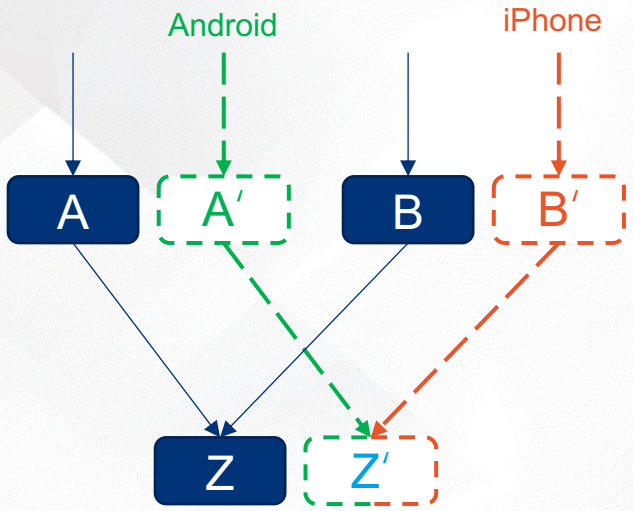
西部城市用户灰度





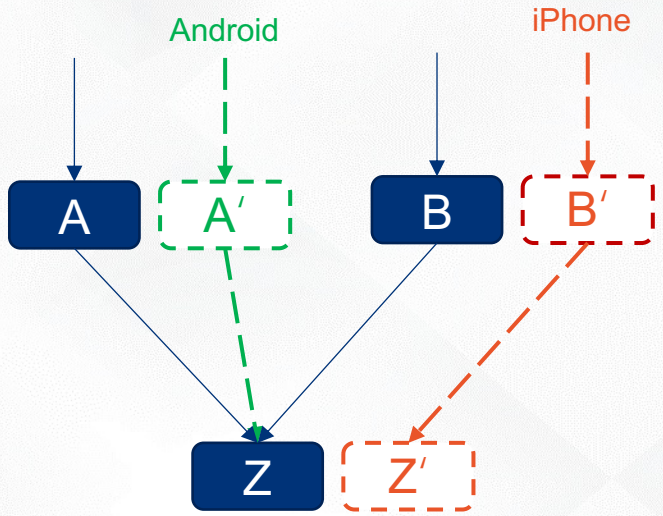
灰度发布的挑战 – 同时灰度发布

The Challenge of Canary Release



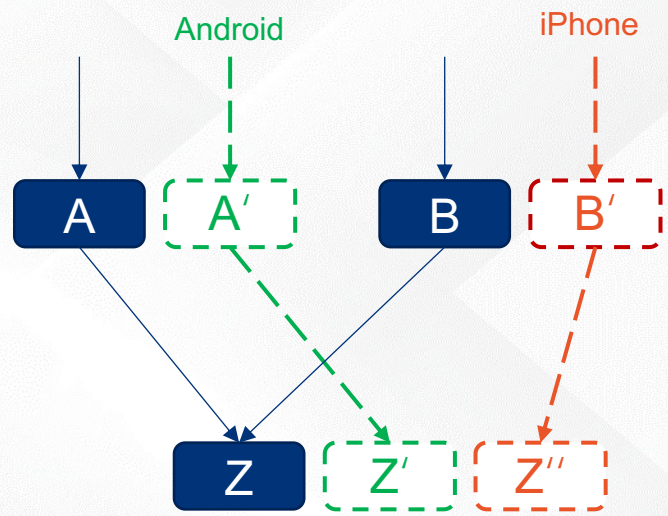
Two canary deployment
- A′ and Z′
- B′ and Z′

Z′ got all traffic from A′ and B′



Two canary deployment
- A′
- B′ and Z′

Z′ only get traffic from B′

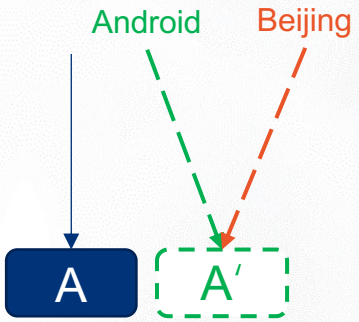


Two canary deployment
- A′ and Z′
- B′ and Z″

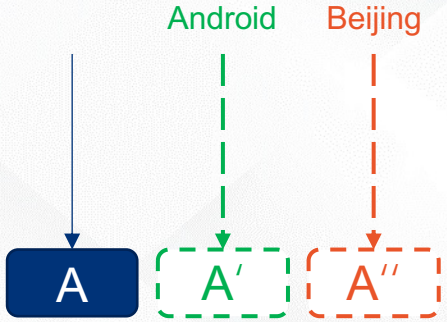
Z′ only get traffic from A′
Z″ only get traffic from B′

灰度发布的挑战 – 多个流量策略

how to do an good Canary Release



If a user matches Android but not Beijing,
Which one should go? A or A’

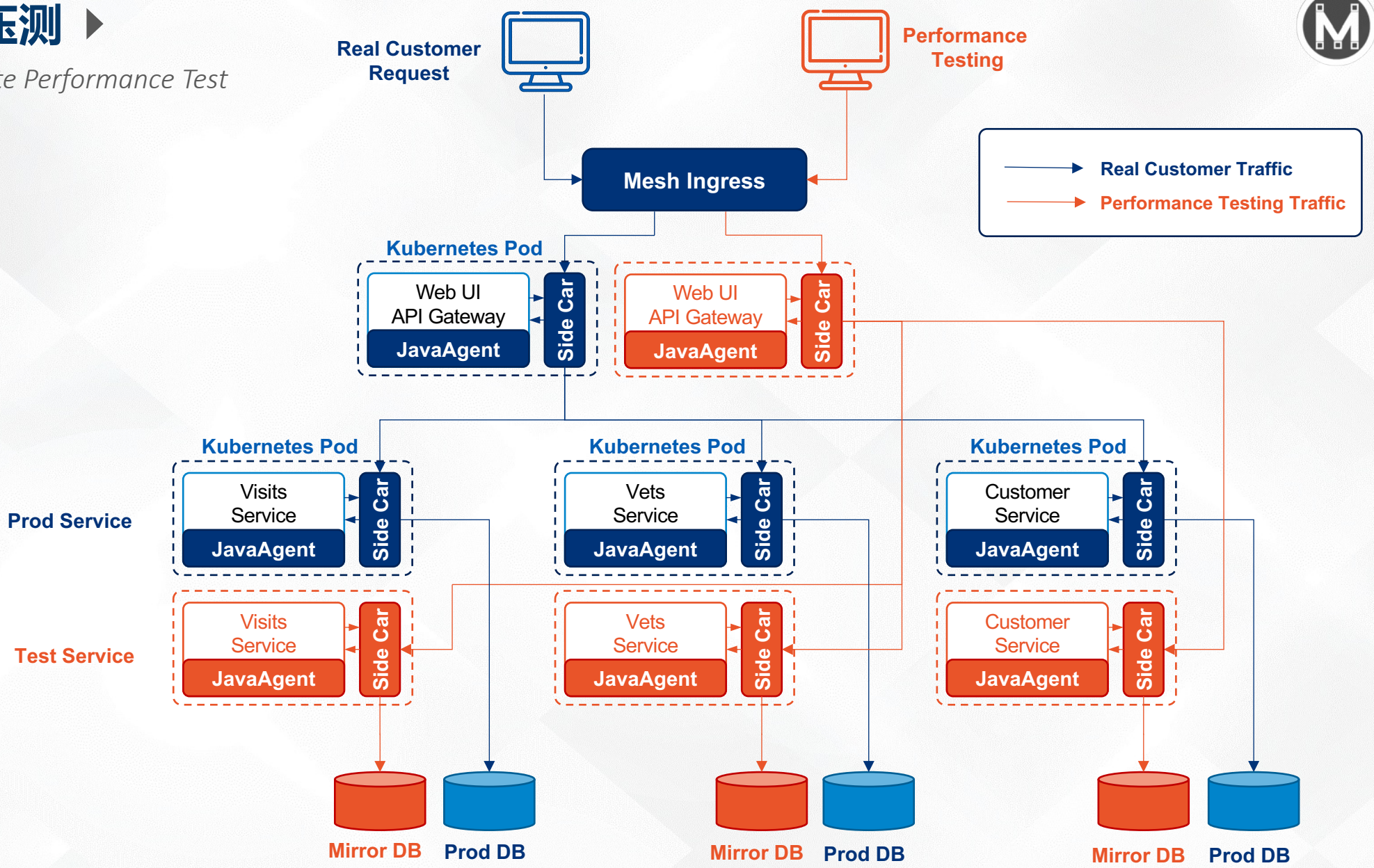


If a user matches Android and Beijing,
Which one should go? A’ or A”

欢迎一起讨论 https://github.com/megaease/easemesh/issues/52

全链路压测

The Whole Site Performance Test



Service Mesh的更多玩法

More Enterprise Solutions with Service Mesh

MegaEase
企业云化架构提供商

一行代码不改做服务治理

一行代码不改做秒杀

一行代码不改做整体监控

一行代码不改高可用

一行代码不改做异地多活

一行代码不改做全链路压力测试

一行代码不改同步转异步

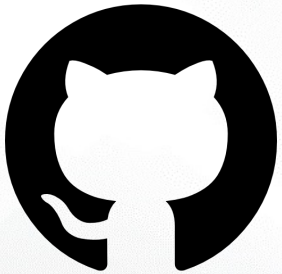
一行代码不改做灰度发布

一行代码不改做跨云管理

一行代码不改做老应用迁移

欢迎关注我们的开源项目

https://github.com/megaease



THANKS

用技术推动商业进步

https://megaease.com