APACHE APISIX

服务网格如何提供平台级的微服务治理能力

古琦

阿里云中间件技术专家

APACHE APISIX

CONTENT

01 微服务治理的现状

02 服务网格的介绍

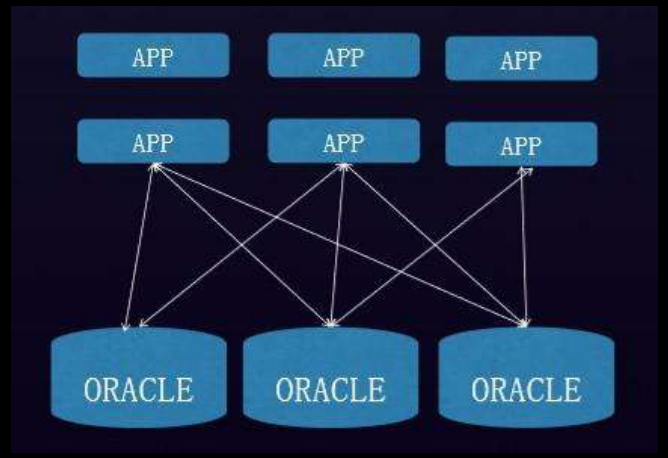
03 如何扩展服务网格的能力

APACHE APISIX

01 微服务治理的现状

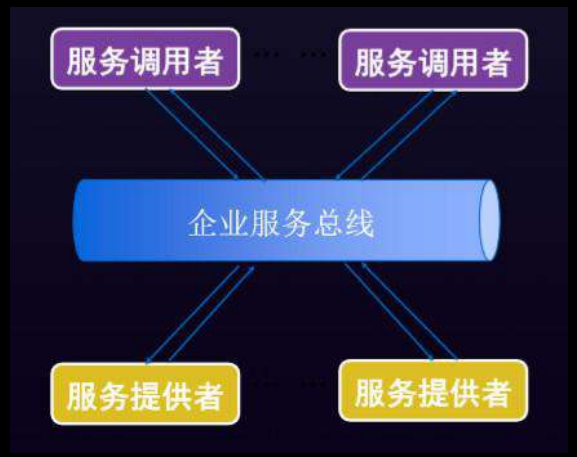
企业架构的演进之路

IOE 架构



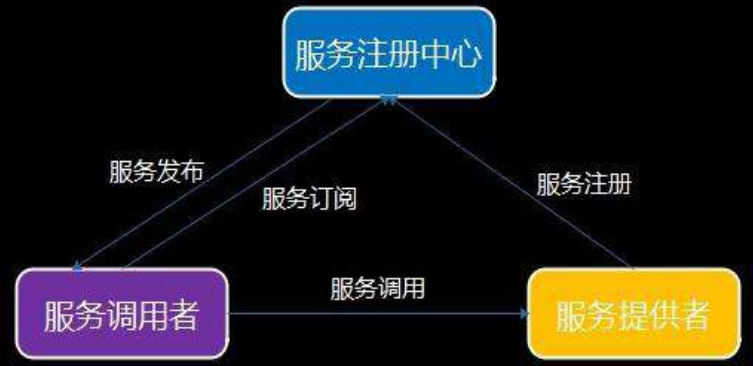
- 应用彼此相对独立，且规模庞大，同时链接多个数据库；
- 应用内部耦合性强，系统复杂，扩展性差；
- 架构封闭，重复制造轮子；

SOA 架构



- 通过ESB总线进行系统集成，松耦合
- 请求需要ESB转发，ESB容易成为性能瓶颈；
- 升级、扩容影响面较大；

微服务架构



- 业务单元松耦合，独行性强，扩展性强；
- 计算存储分离，数据层由各个微服务负责
- 业务发展更敏捷，迭代速度快；

微服务框架

为了应对复杂的业务场景和开发需要，服务化的框架被提出来的越来越多，比如以下我们常见的几种：

1、Spring Cloud

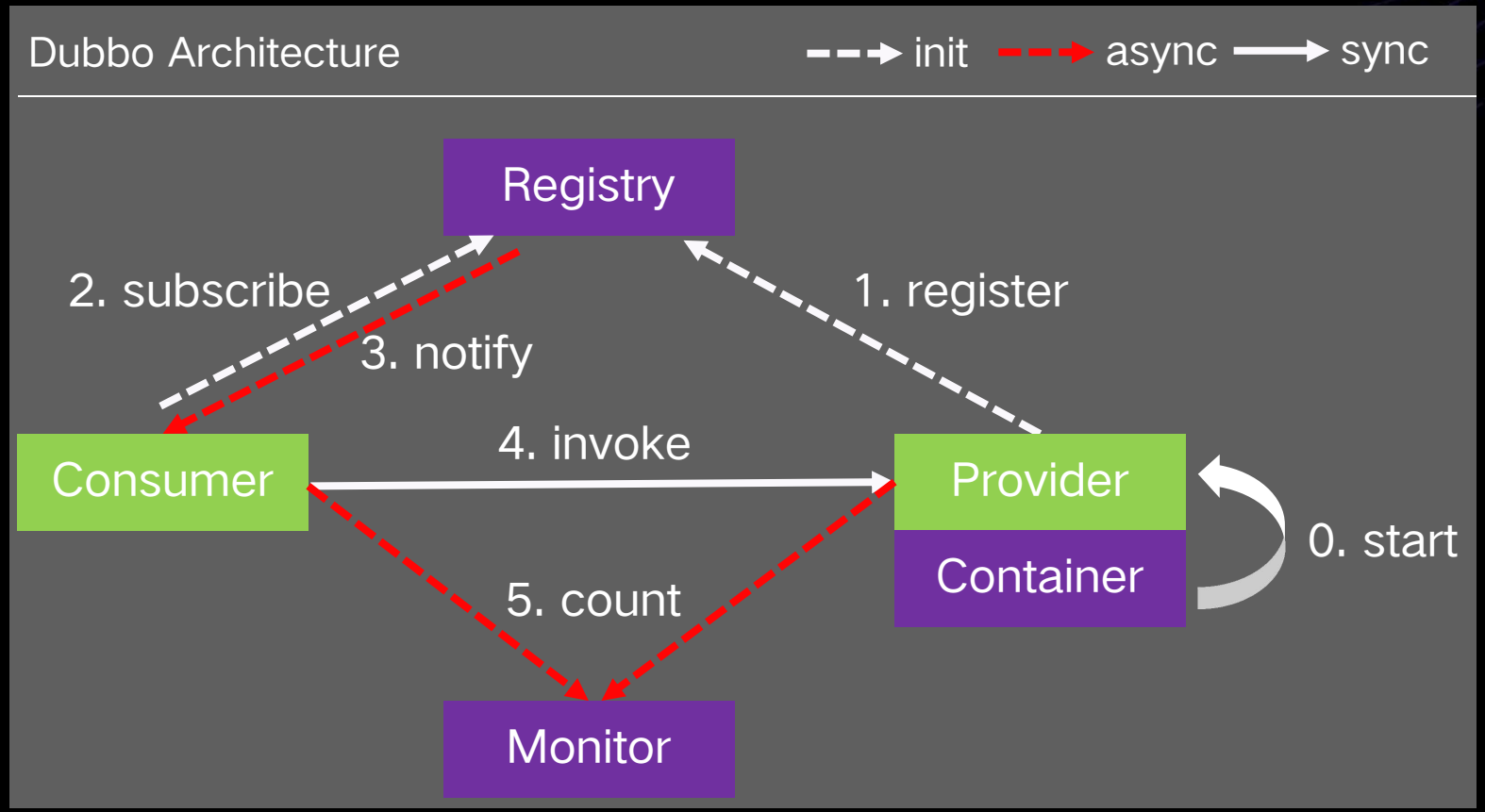
2、Dubbo

3、Kratos

4、Kitex

5、go-micro

...



企业架构的演进之路

- **微服务开发模式下主要面临什么样的棘手问题？**
  - 多语言问题：有多种编程语言，node.js, JAVA, GoLang…微服务需要为每种语言都维护一种中间件SDK
  - 升级推动难：SDK升级需要推动业务应用进行代码修改和发布，对业务有打扰，业务压力下推动成本高
  - 迭代速度慢：由于多语言多版本的存在，需要花费大量精力去维护历史版本，降低了迭代速度，一年只能升级1-2次 （在数据面选型上也考虑研发效能的问题）
  - 多版本问题：每种语言的SDK都面临版本升级，同时存在大量不同的版本互相访问，兼容性和测试维护成本巨大

目标

降低对业务干扰　　提升中间件研发效能　　统一版本

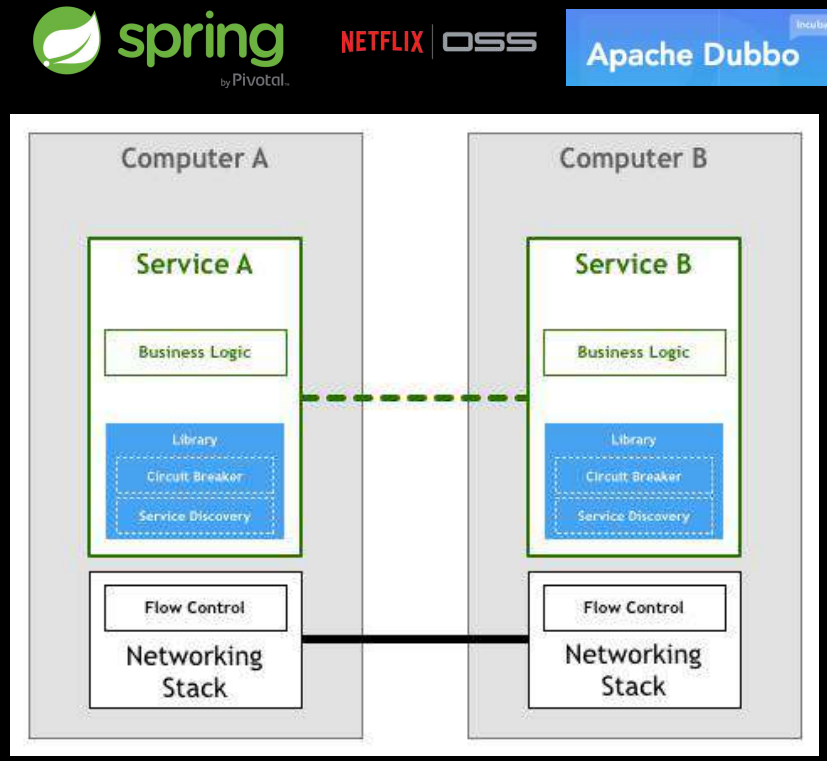
方式

SDK与业务应用解耦

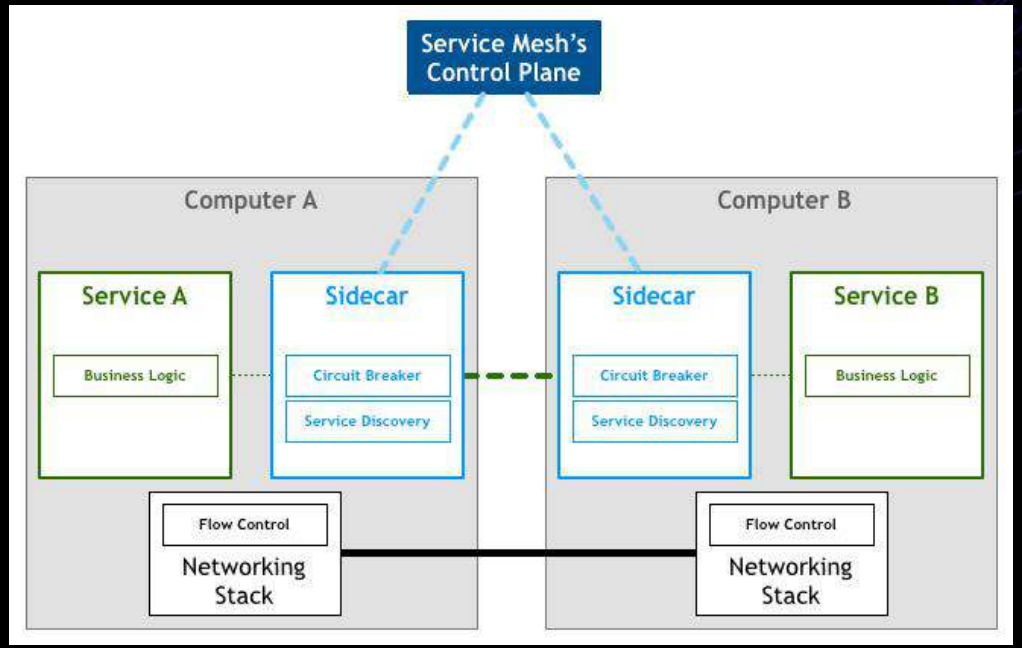
手段

APP　　Sidecar

从开发框架到服务网格



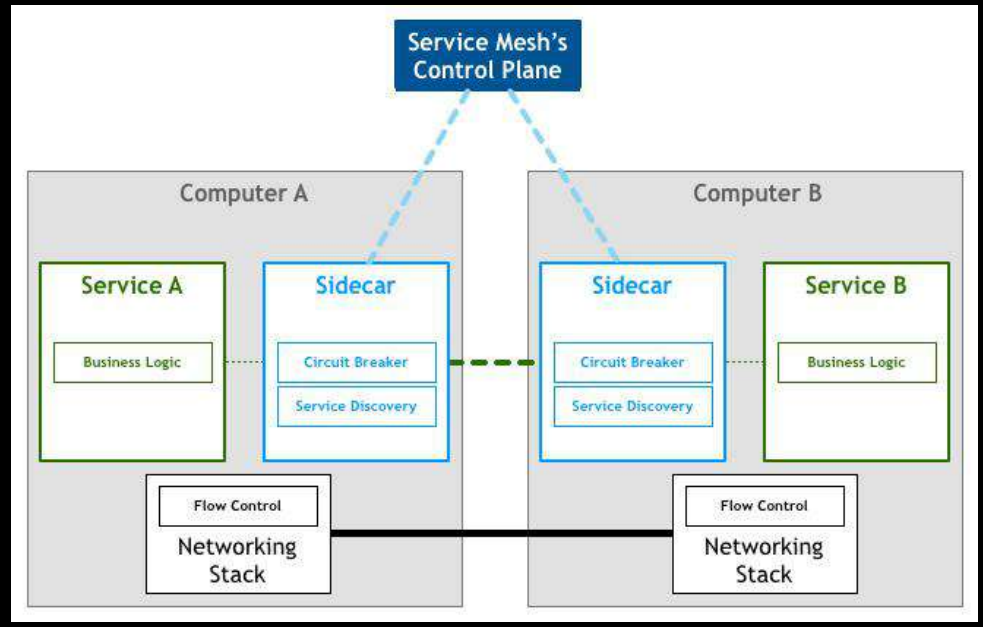
- ✓ 以代码库方式构建在应用程序中
- ✓ 版本不同往往带来冲突问题
- ✓ 版本变更，整个应用随之变更
- ✓ 不同开发语言、不同框架，方案差异性大



- ✓ 使用复杂度低，应用只需关注业务逻辑
- ✓ 支持多语言，无技术栈限制
- ✓ 代码无侵入，可单独升级
- ✓ 抽象为业务应用的网络层，低耦合

服务网格

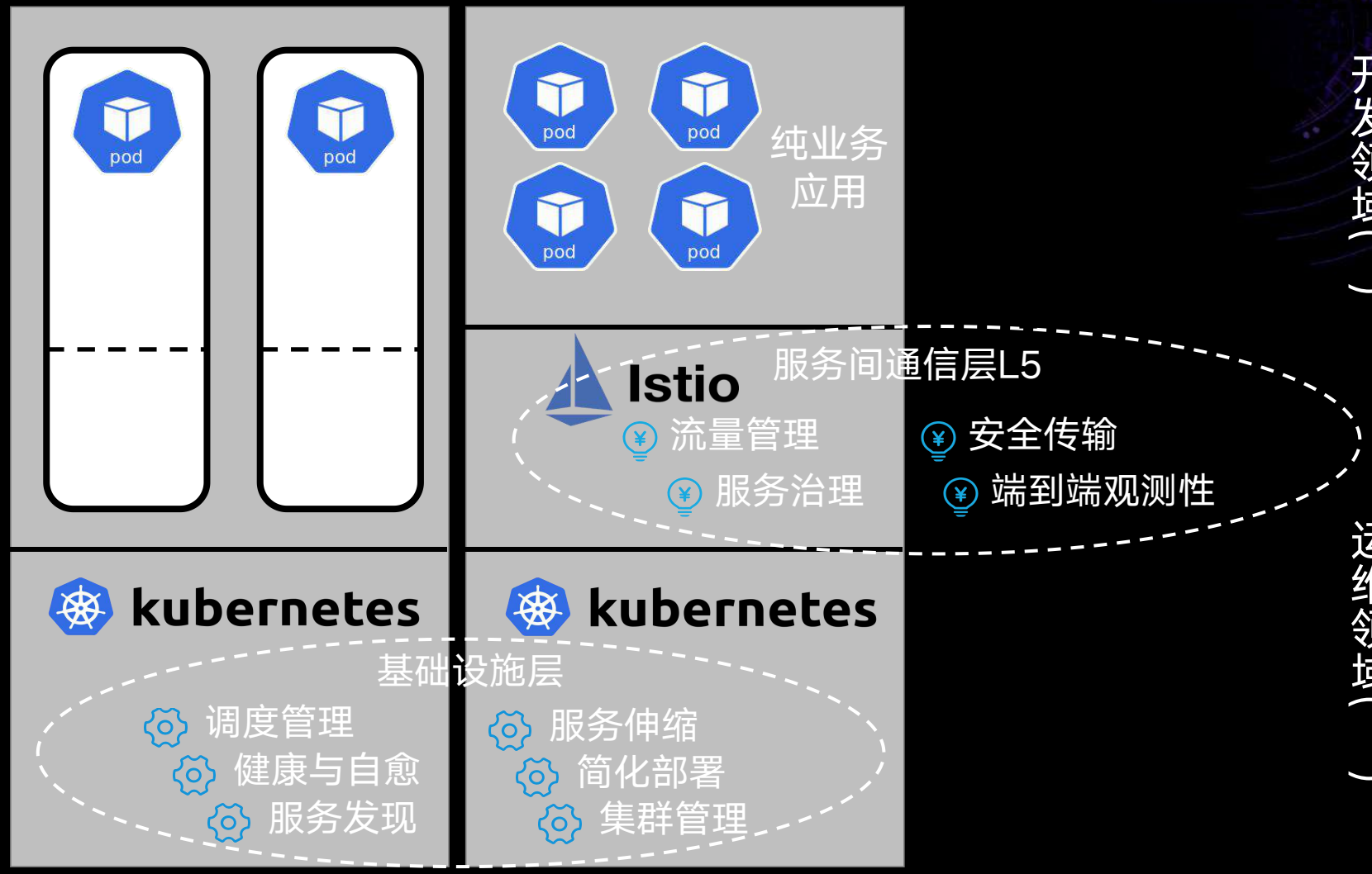
- **定义：** Service Mesh 是一个专门处理服务通讯的基础设施层。它的职责是在由云原生应用组成服务的复杂拓扑结构下进行可靠的请求传送。在实践中，它是一组和应用服务部署在一起的轻量级的网络代理，并且对应用服务透明。
- **功能：**连接， 安全，控制，可观察性
- 价值：**实现应用解耦平台级的服务治理能力**
  - 基础架构与应用解耦，降低升级和运维成本
  - 多语言多协议统一微服务治理
  - **服务通讯间安全性**
  - **应用运行的可观察性**



APACHE APISIX

02 服务网格的介绍

有了 Kubernetes 还需要 Service Mesh 么？



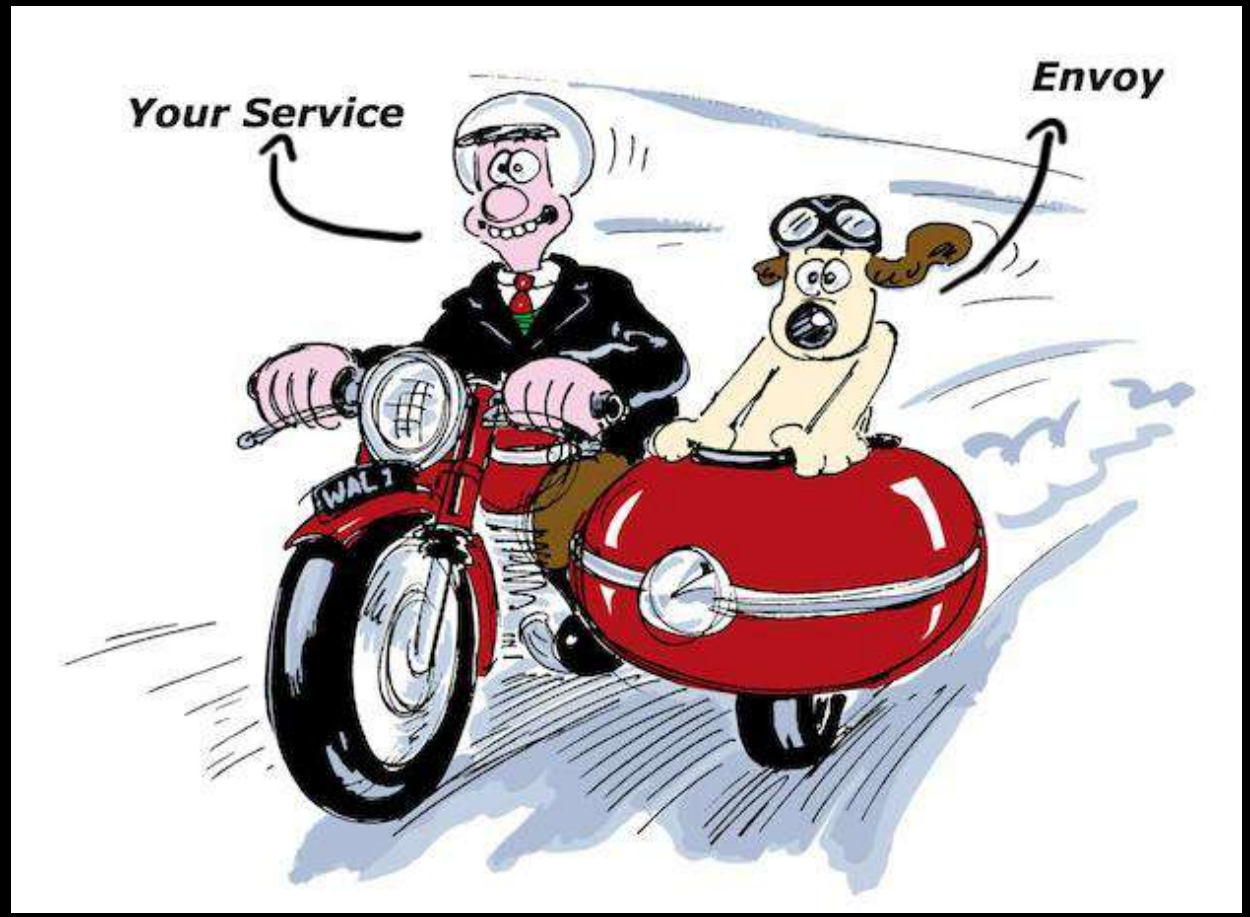
APACHE APISIX

通过服务网格将服务治理标准化、统一化

功能	类别	需要更改应用程序吗？
支持多种协议的流量控制	流量管理	No
支持mTLS	安全性	No
负载均衡	可靠性	No
故障注入	可靠性	No
金丝雀部署	发布管理	No
A / B测试支持	发布管理	No
流量镜像	流量管理	No
服务之间的RBAC授权	安全性	No
HTTP请求指标	监控	No
TCP连接指标	监控	No
请求跟踪	性能监控	需要少量工作，每个微服务必须将传入请求中的标头添加到传出请求中
ó .	ó .	ó .

APACHE APISIX

Sidecar



1.Service Discovery

2.Observability (metrics)

3.Rate Limiting

4.Circuit Breaking

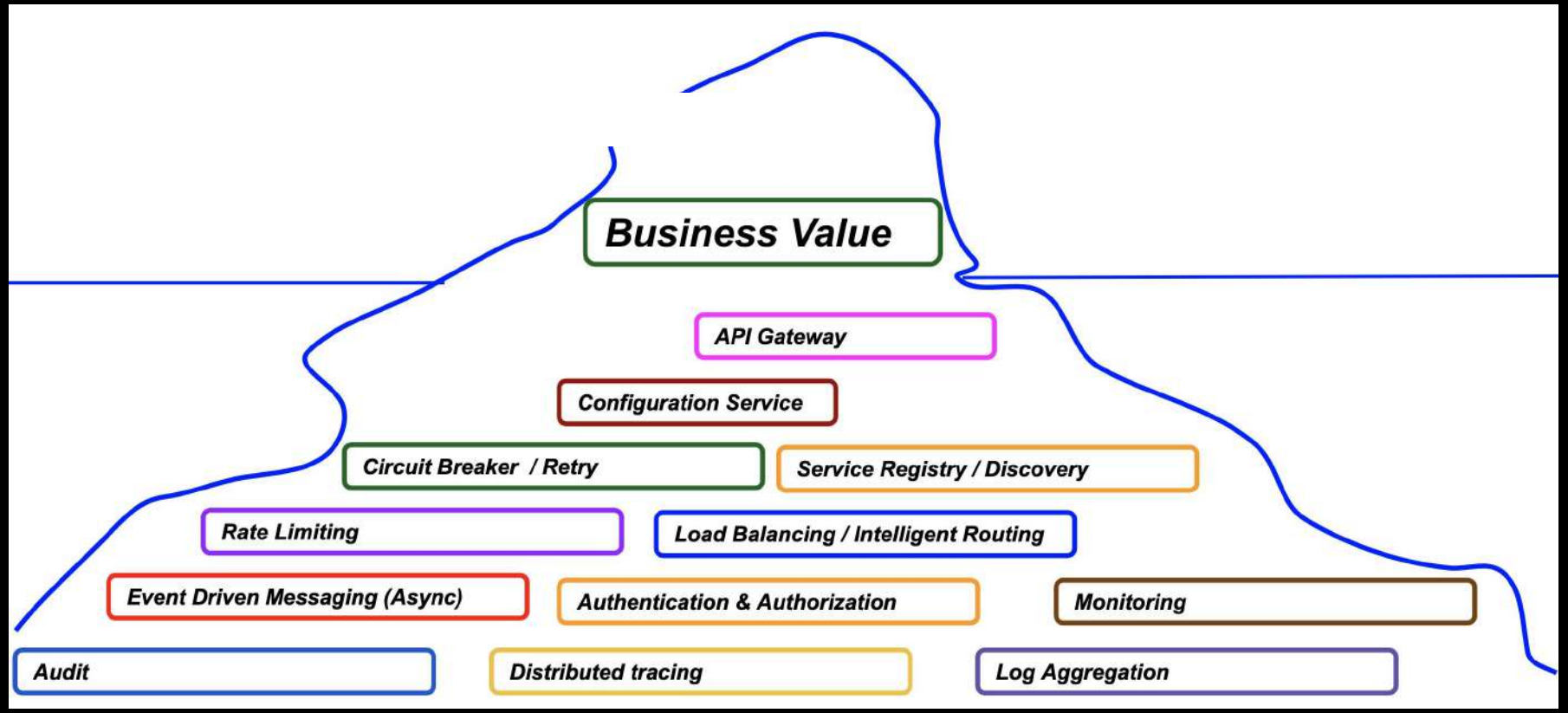
5.Traffic Shifting

6.Load Balancing

7.Authentication & Authorization

8.Distributed Tracing

抽象应用服务通信能力到基础设施



开发人员聚焦于业务应用本身开发

抽象化、自动化

重新定义应用服务网络通信

APACHE APISIX

03 如何扩展服务网格的能力

APACHE APISIX

服务网格的缺点

1、针对已有微服务框架的服务支持不友好

目前服务网格的能力只针对使用 K8s 作为服务发现的微服务支持完整，针对其他的微服务迁移到服务网格中，需要做代码修改

2、资源的问题

每个 Pod 一个 Sidecar 对于很多的用户而言是一个比较大的资源消耗，这也导致了服务网格不能更好的普及

3、性能问题

Sidecar 会在客户端、服务端都同时存在，RPC 的过程中相比于 SDK 的模式多了两跳处理流程，也会导致很多用户质疑服务网格的性能问题

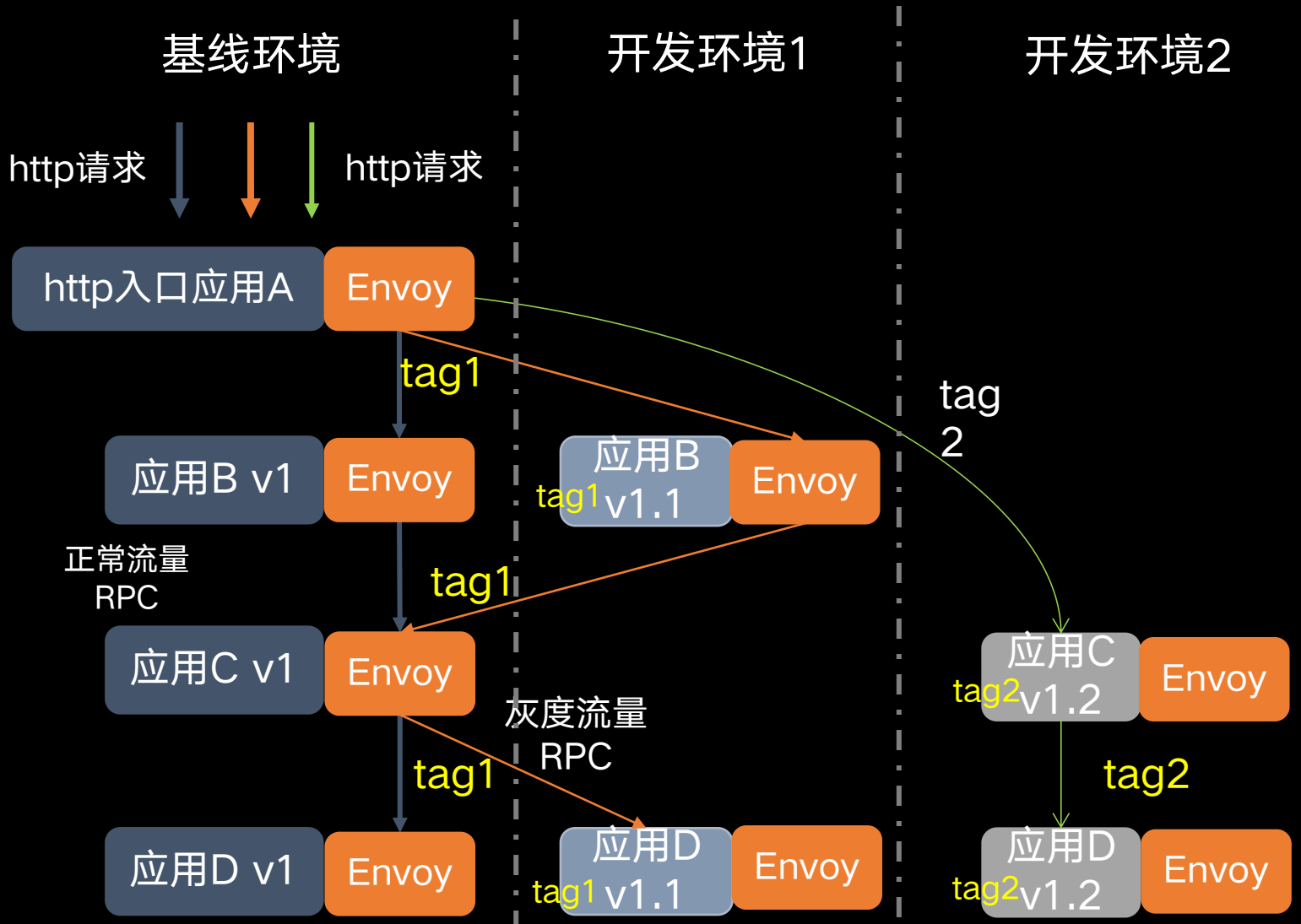
APACHE APISIX

对接不同类型的注册中心

通过 MCP Over XDS 的协议，扩展 Istio 的能力，对接类似 Eureka、Zookeeper、Nacos 这几个常见的注册中心，这样能支持类似 Spring Cloud、Dubbo 等服务迁入服务网格

APACHE APISIX

全链路路由



基于 标签的路由控制：

抽象出了 TrafficLabel CRD

动态地对流量和应用实例打标

标签灰度：

按标签路由

基线环境流量回退机制

优点：

可以任意路由到服务链路中间服务

研发/测试人员可独立部署一套环境

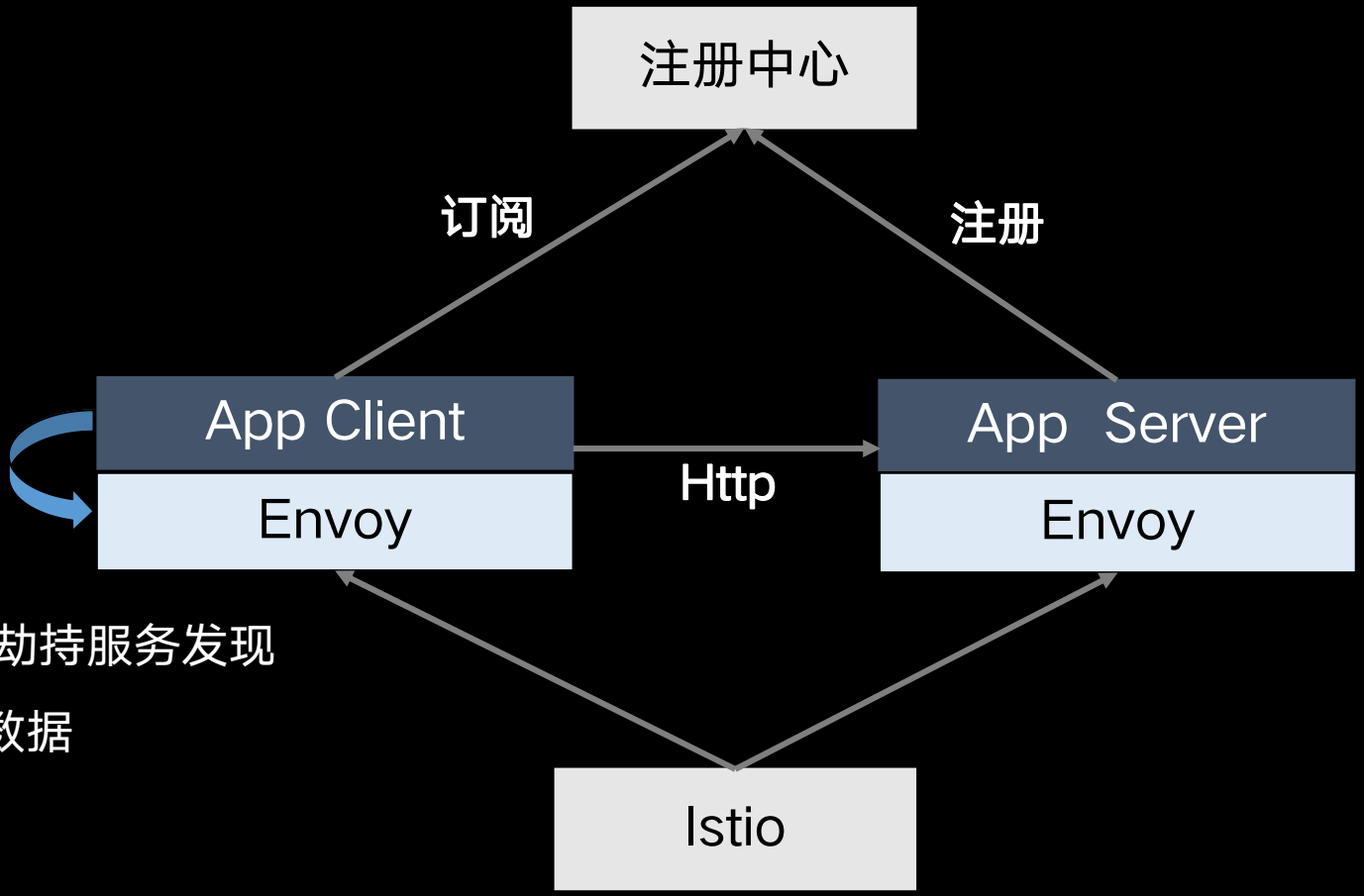
提升研发/测试效率，降低运维成本

依赖：

技术实现本身依赖 trace 完整链路

(x-b3-traceid, x-request-id etc.)

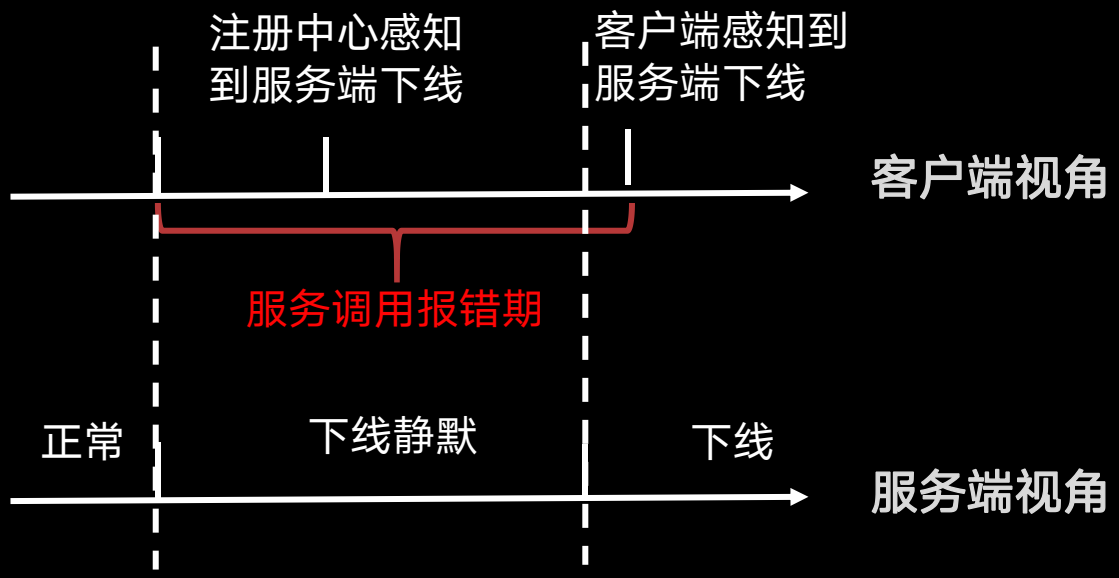
微服务如何无缝接入服务网格

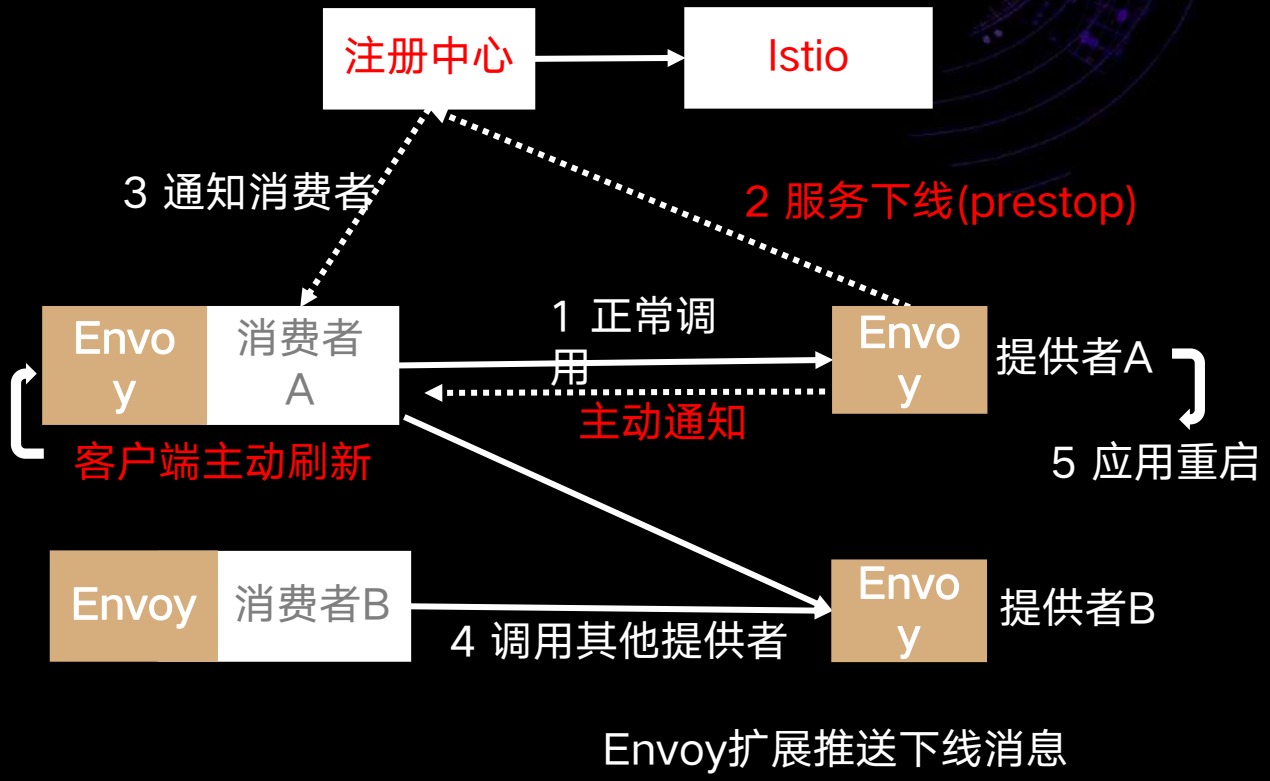


扩展 Envoy，劫持服务发现

替换服务发现数据

无损的上下线





Envoy扩展推送下线消息

APACHE APISIX

性能&资源问题

1、使用的 Fat SDK 变成一个 Thin SDK，只做序列化、反序列化、RPC 的事情，减少资源消耗

2、ebpf + sidecar 的模式，去除 Iptables 在劫持流量时候的损耗

3、Pre Node 的 Sidecar 或者 ProxylessSidecar 也被提出的越来越多

APACHE APISIX

感谢聆听

THANKS