msup

# 有赞Service Mesh实践

# 讲师简介

- 王健
- 有赞高级技术专家、技术总监
- 有赞中间件团队负责人
- 主要负责微服务架构、Service Mesh以及接入层网关等

# 目录


1 为什么选择Service Mesh?

2 Service Mesh 演进路径

3 取得的收益

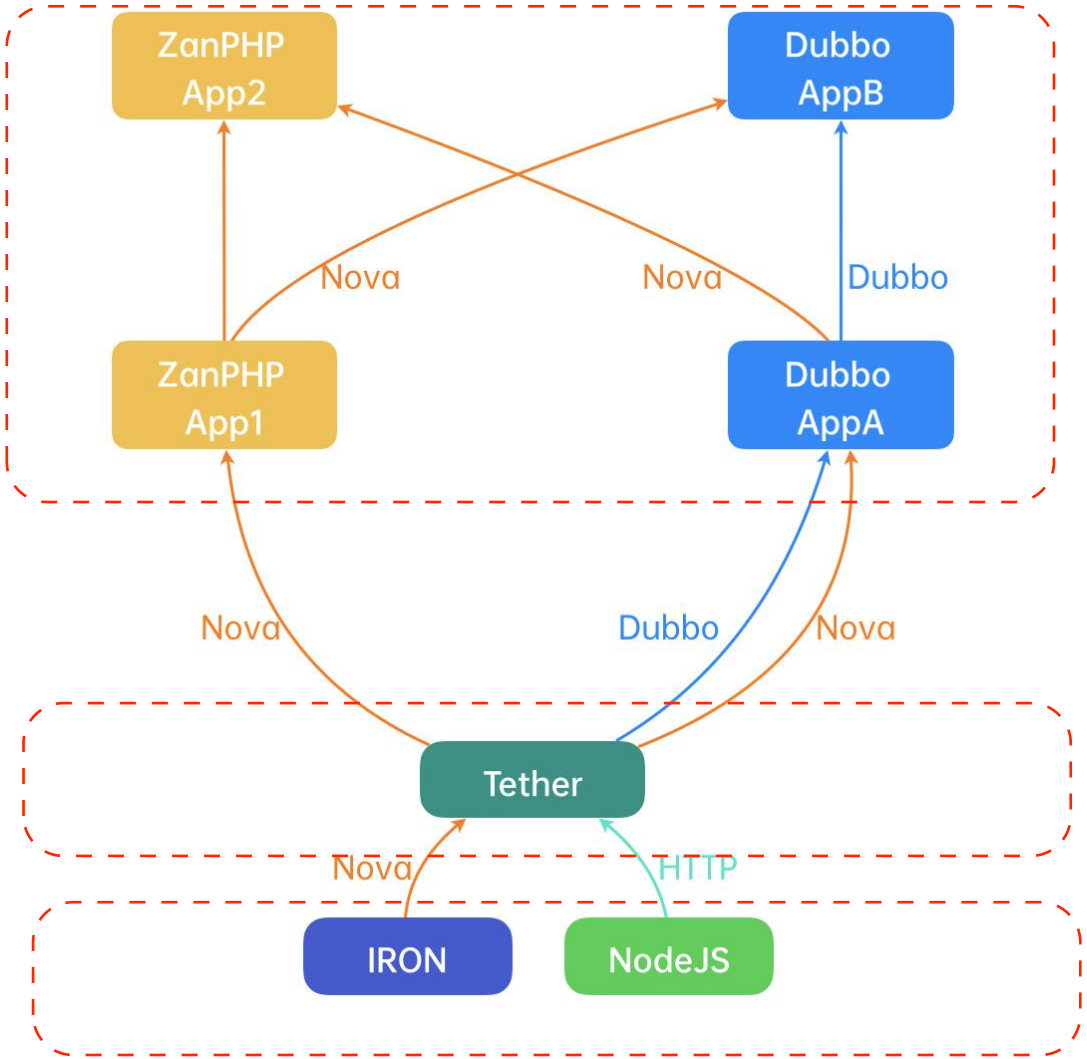
4 落地和实践建议

5 未来展望

# 为什么选择Service Mesh?

# 2019年初微服务架构

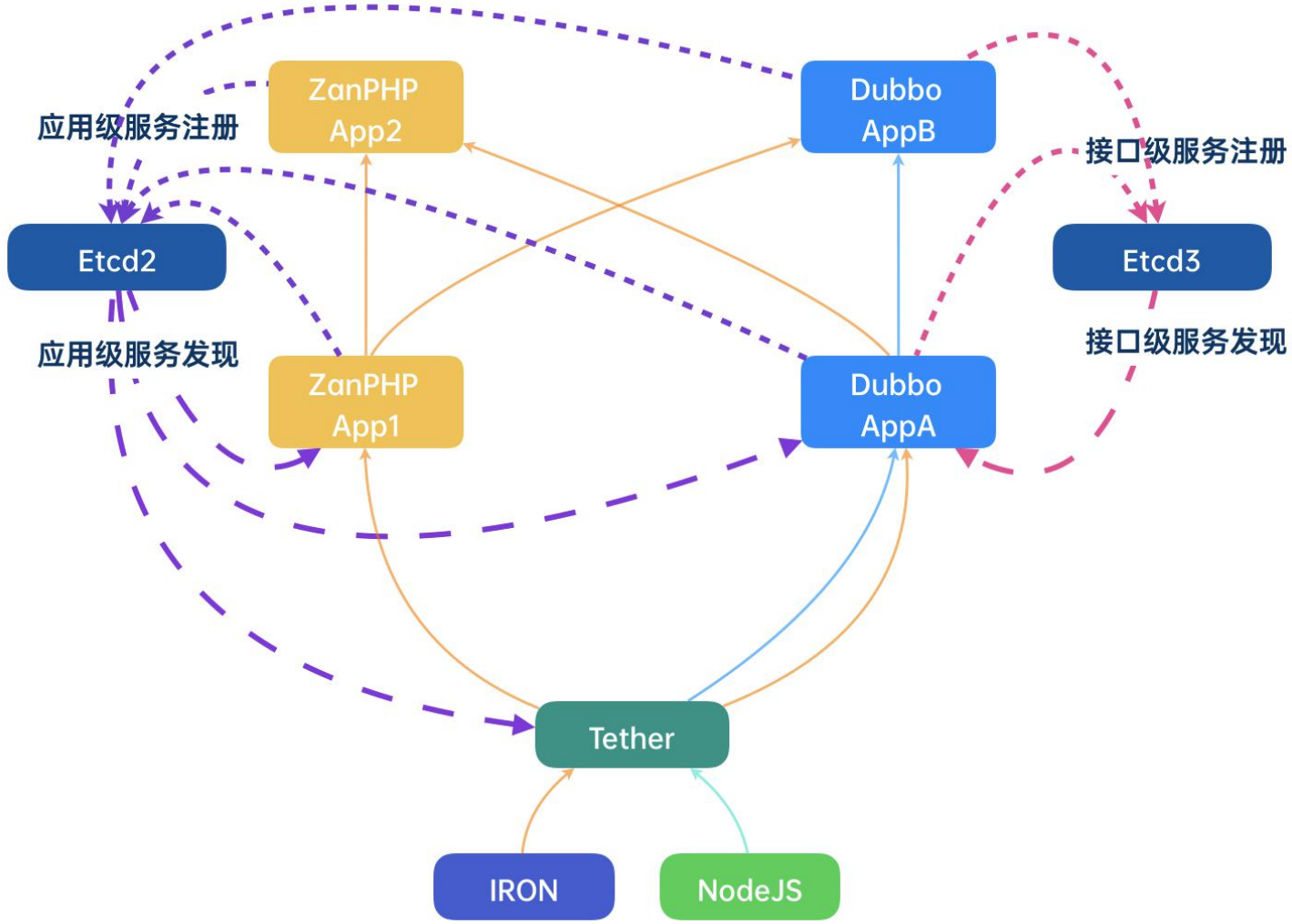
- 微服务应用
  - 两种语言：Java、PHP
  - 两种框架：Dubbo、Zan PHP
  - 两种协议：Dubbo、Nova(基于thrift自研)
- 前端应用
  - NodeJS：HTTP 协议访问后端微服务
  - PHP-FPM：通过Nova协议访问后端
- 自研Sidecar => Tether
  - 服务发现与请求路由
  - 协议转换：HTTP <—> Dubbo



# 2019年初微服务架构

## 服务注册与发现

- 两种注册框架
- 两个注册中心
- 两种注册维度
- 三个服务发现与路由组件



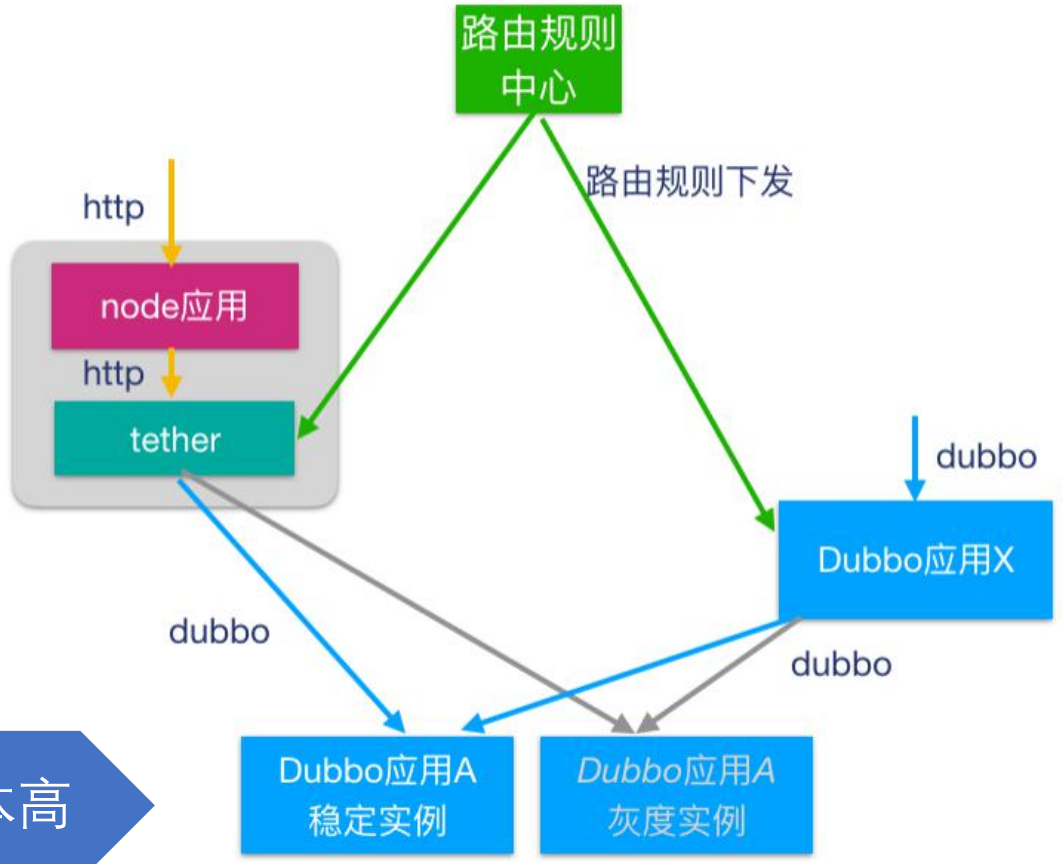
# 问题：

- 拓扑和组件依赖错综复杂、运维成本高
- 基础能力重复建设
- 注册中心瓶颈
- 新功能上线周期长

# 痛点案例

流量路由系统（2018H2 ~ 2019H1）

- 流量动态路由能力（支持灰度、蓝绿发布）
- Sidecar 为 Node 应用支持
- Dubbo 框架为 Dubbo 应用支持




开发成本高 > 测试成本高 > 升级周期长 > 运维成本高

# Service Mesh演进路径

# 核心考量

- 价值最大化
- 逐步落地、降低风险
- 平滑落地、降低对业务开发打扰

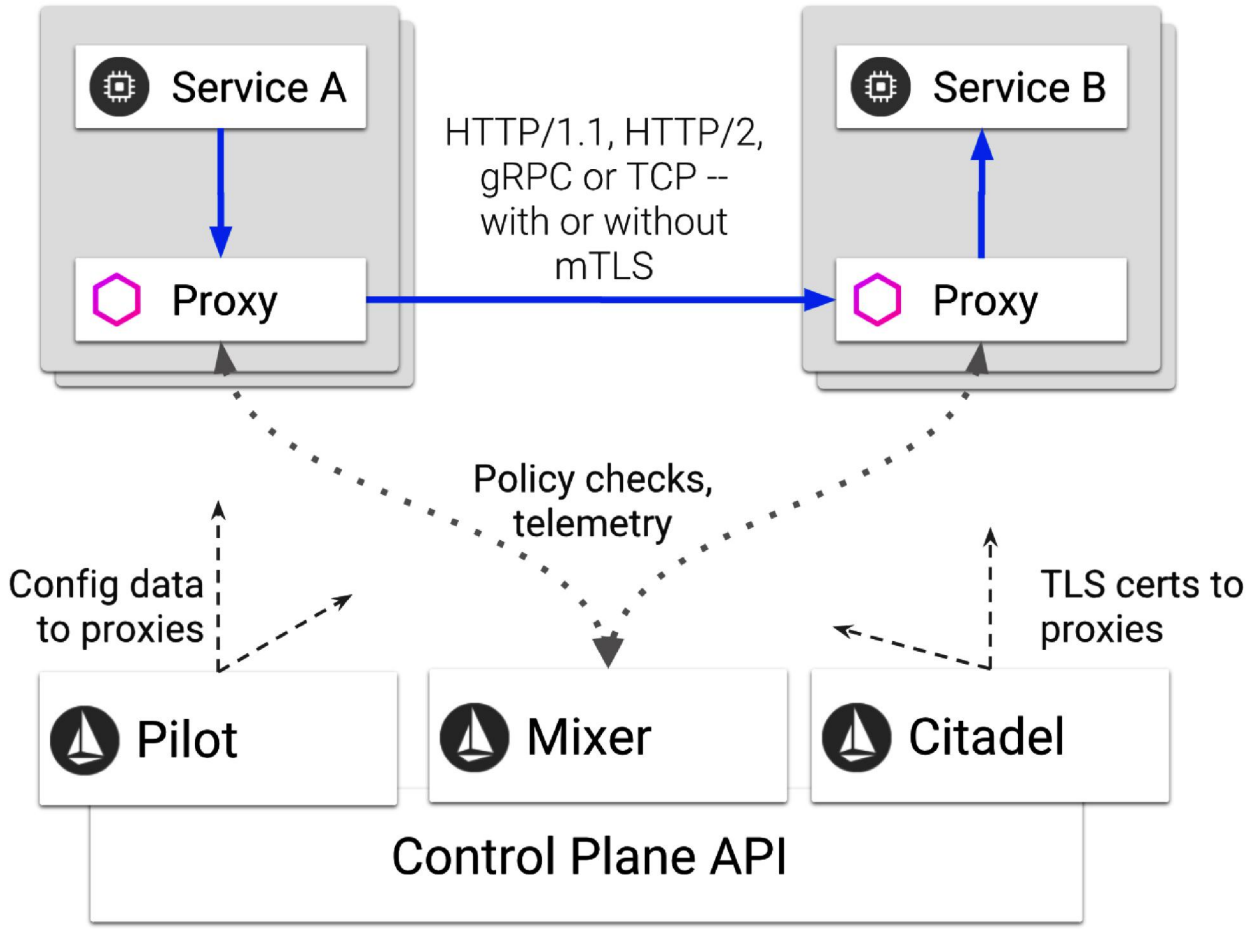
# 技术选型

## 开源方案

- 控制面 Istio
- 数据面 Envoy

## 问题

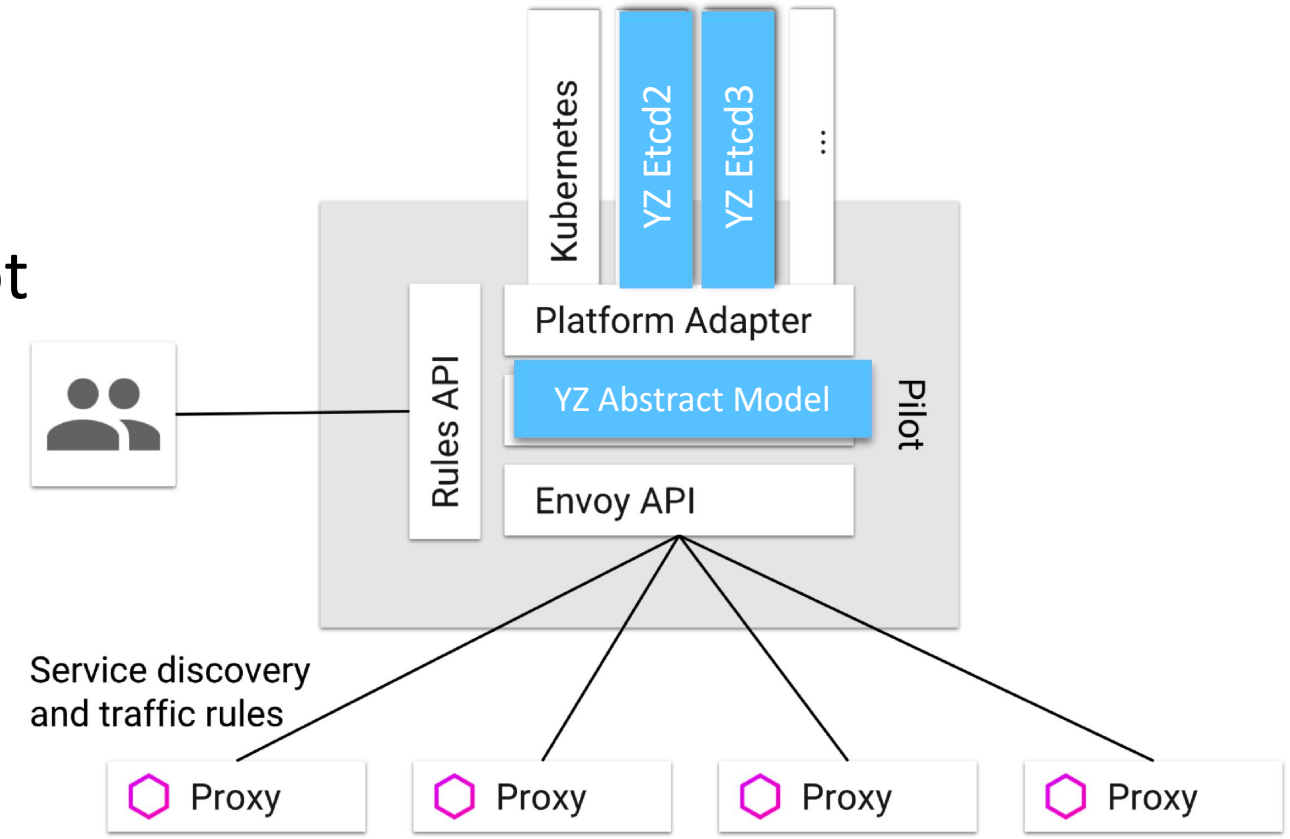
- 内部特性不支持
  - 项目隔离
- Dubbo不支持
  - 协议、服务注册与发现模型
- Envoy 难以把控
  - C++语言开发




https://istio.io/v1.0/docs/concepts/what-is-istio/

# 落地技术方案

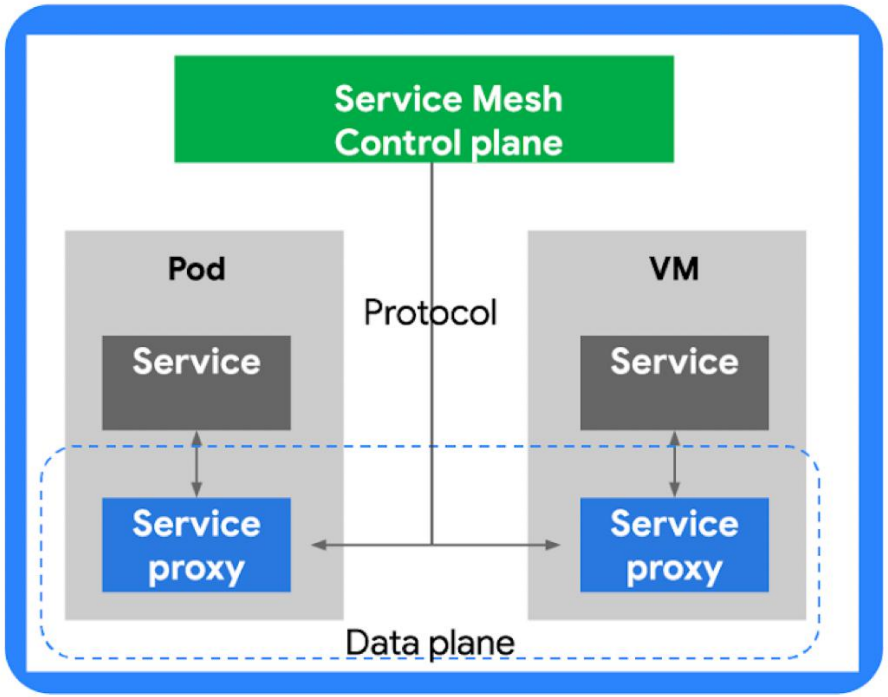
- 选择 Istio Pilot 组件
- 自研的Sidecar替代Envoy 对接 Pilot
- Pilot 对接内部注册中心



# A closer look at Service Mesh

**Service** Mesh

- Visibility
- Resiliency & Efficiency
- Traffic Control
- Security
- Policy Enforcement



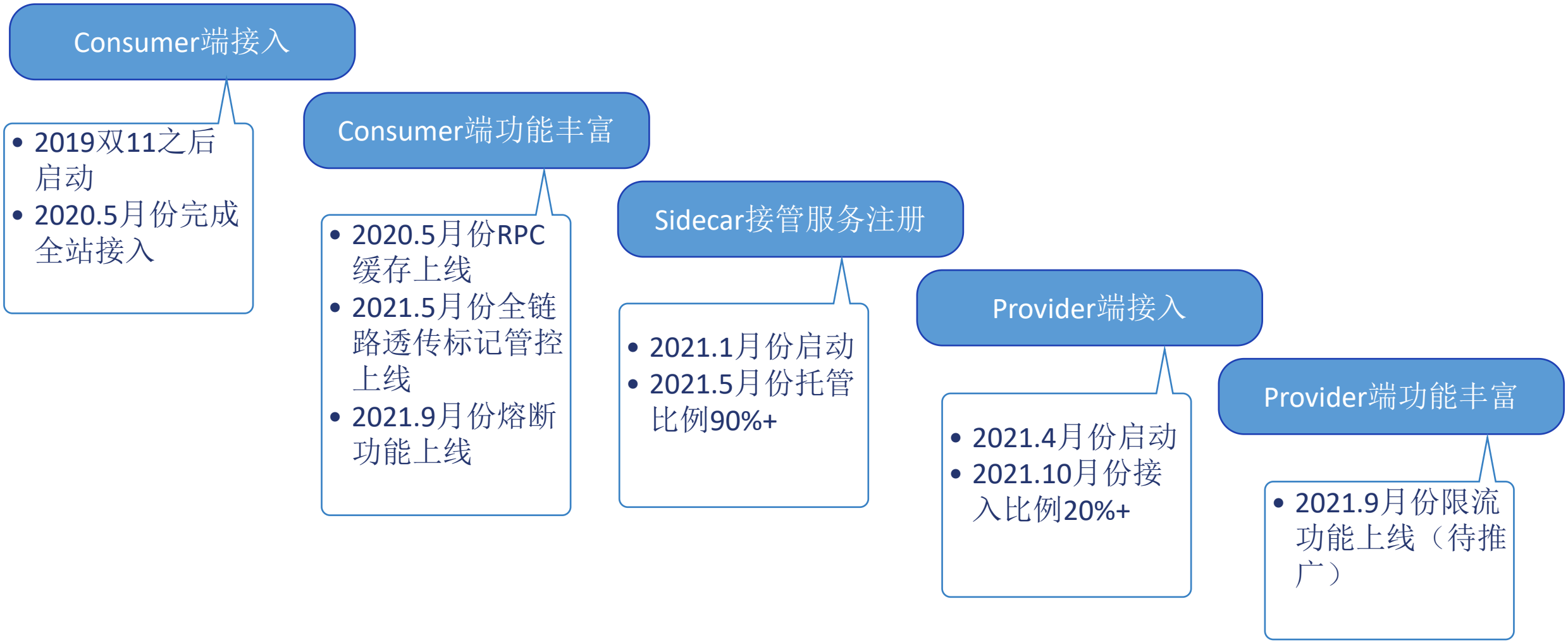

**Outbound features (client-side)**

- Service authentication
- **Load balancing**
- Timeouts, retries and circuit breakers
- Connection pool sizing
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection

**Inbound features (server-side)**

- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

# 演进路径概览

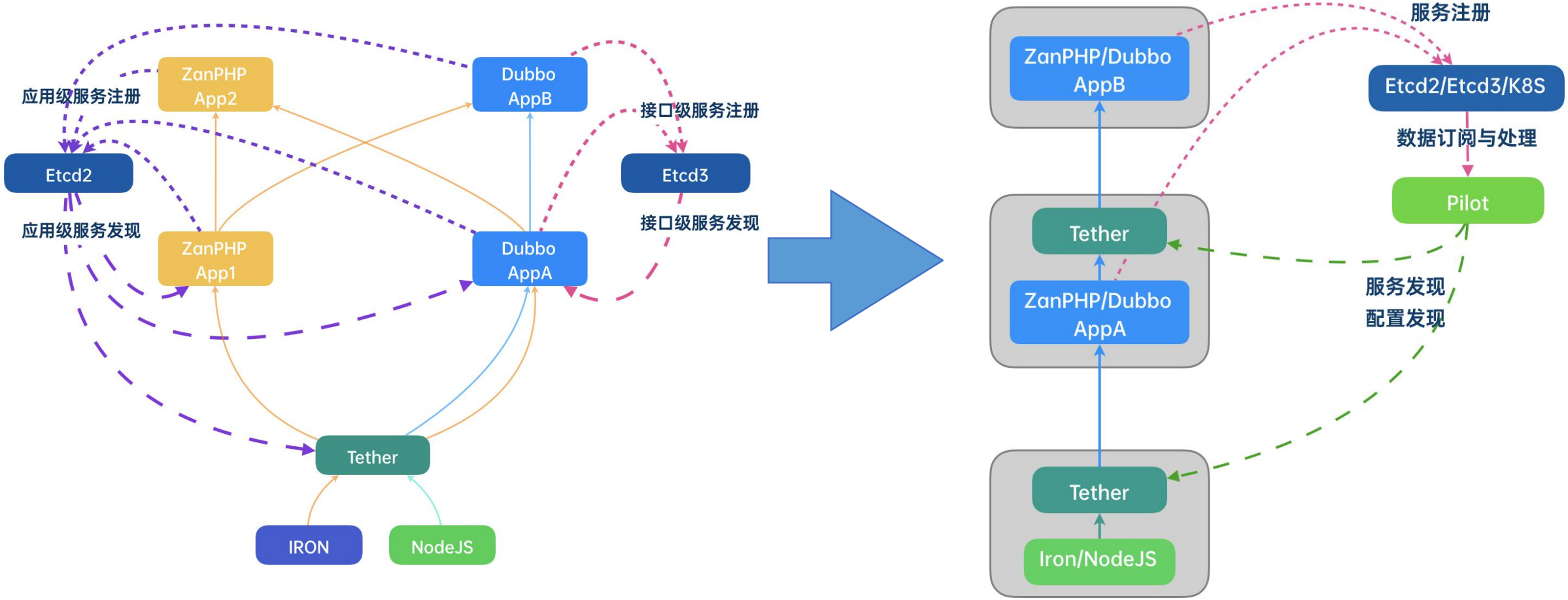

Consumer端接入
2019双11之后启动
2020.5月份完成全站接入
Consumer端功能丰富
2020.5月份RPC缓存上线
2021.5月份全链路透传标记管控上线
2021.9月份熔断功能上线
Sidecar接管服务注册
2021.1月份启动
2021.5月份托管比例90%+
Provider端接入
2021.4月份启动
2021.10月份接入比例20%+
Provider端功能丰富
2021.9月份限流功能上线（待推广）

# 取得的收益

# 取得的收益

msup

## 历史复杂架构收敛



# 取得的收益

## 服务通信基础能力收敛

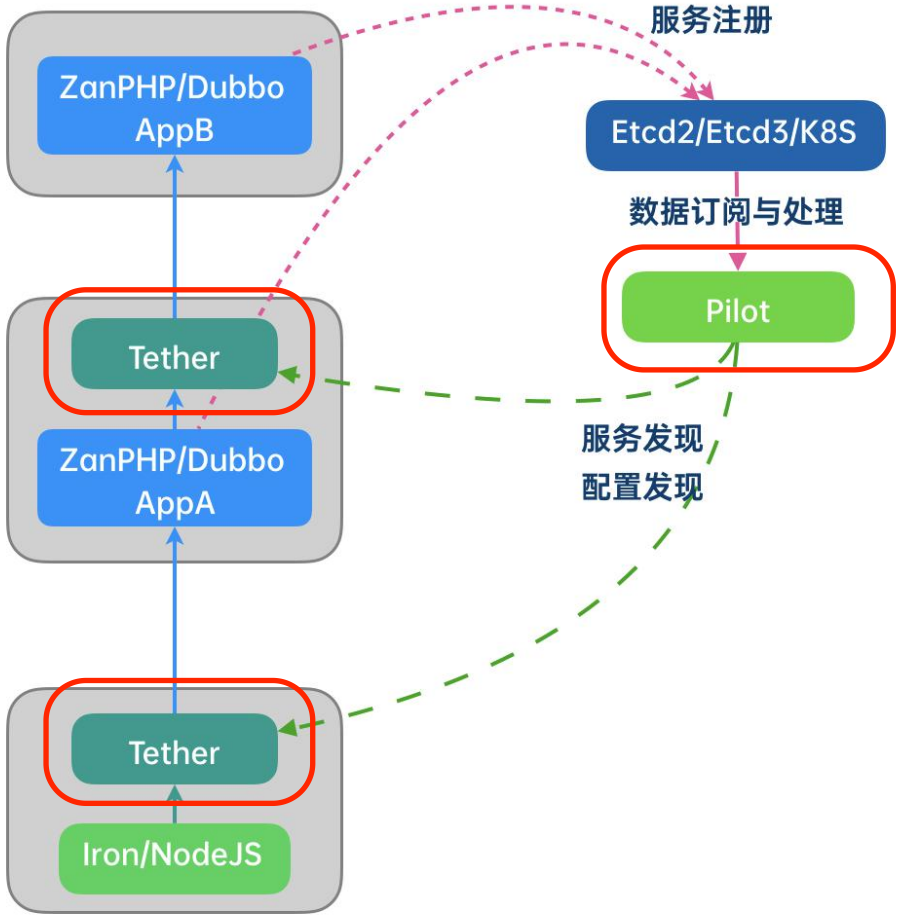
- 仅 sidecar支持即可

## 解决注册中心订阅瓶颈

- Pilot 无状态

## 统一服务发现模型

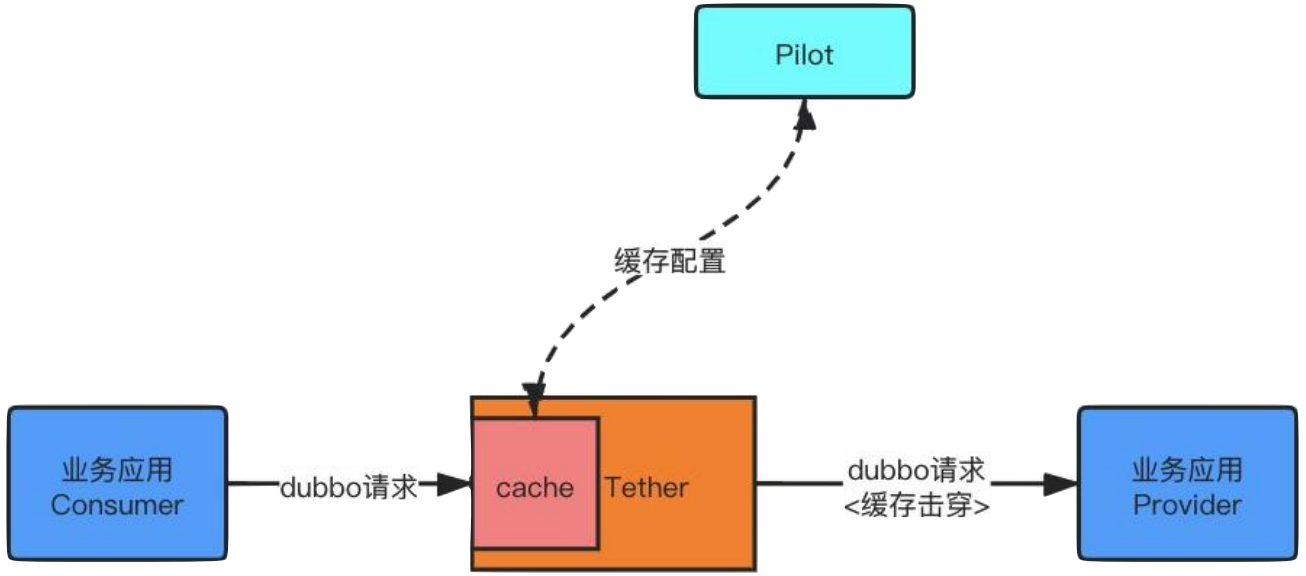
- 对齐主流解决方案，K8S 等
- 屏蔽注册中心细节



# 取得的收益

## RPC 缓存

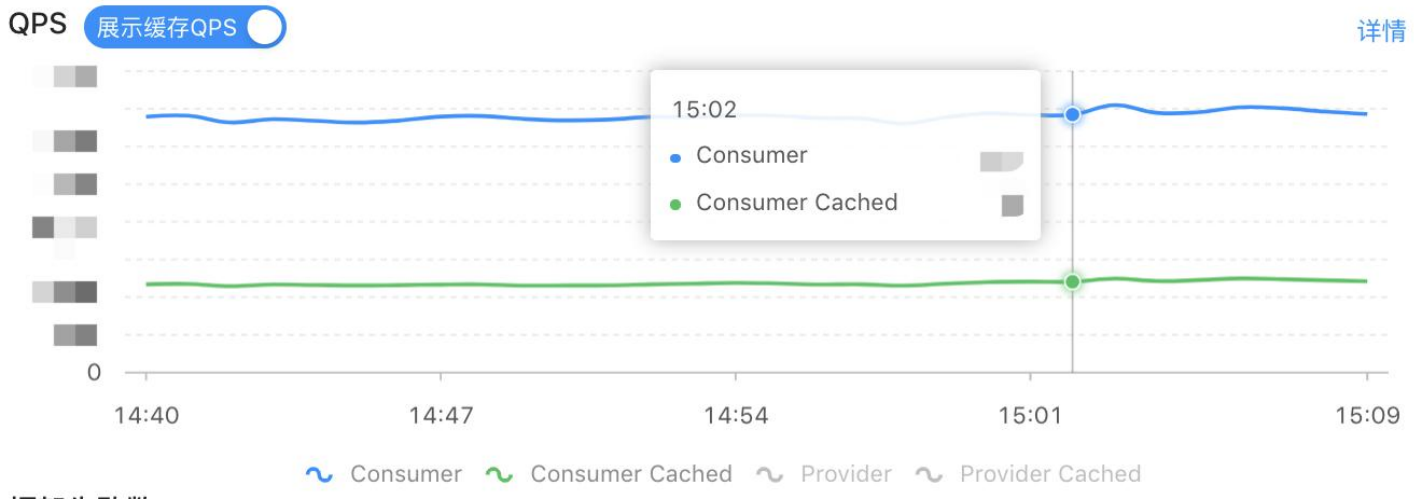
- 透明接入，不限开发语言
- 缓存过期、缓存击穿、缓存淘汰
- 运行时调整缓存策略
- 监控可观测性集成



# 取得的收益

## RPC缓存

- 业务零成本接入
- 2020 双11 性能提升利器
- 全站日常命中率15%左右，活动命中率近70%



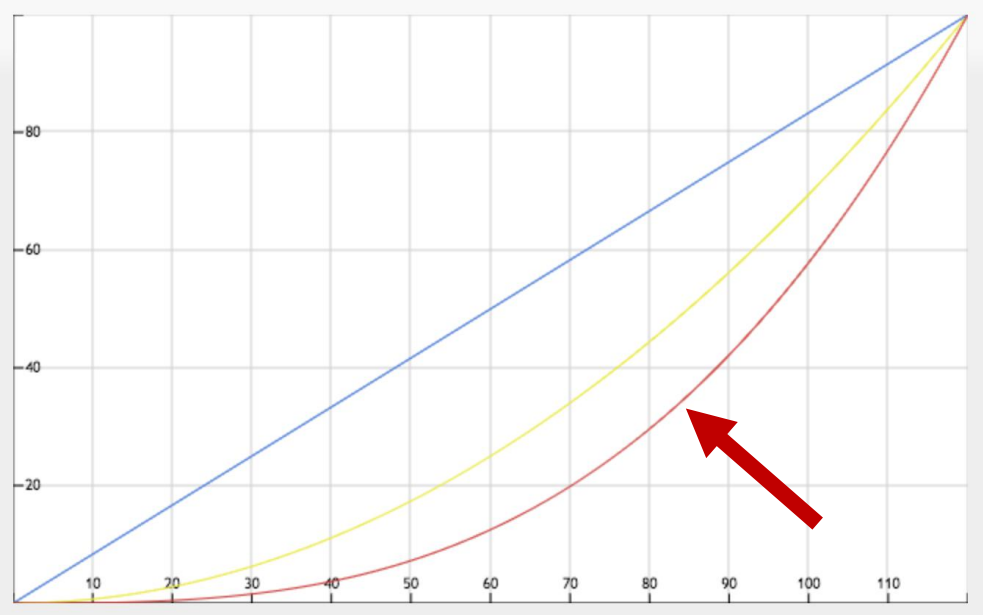
# 取得的收益

msup

## 其他...

- 异常服务实例快速熔断
- 请求超时时间管控
- 全链路透传信息管控
- 功能快速上线

**更新上线3天时间**



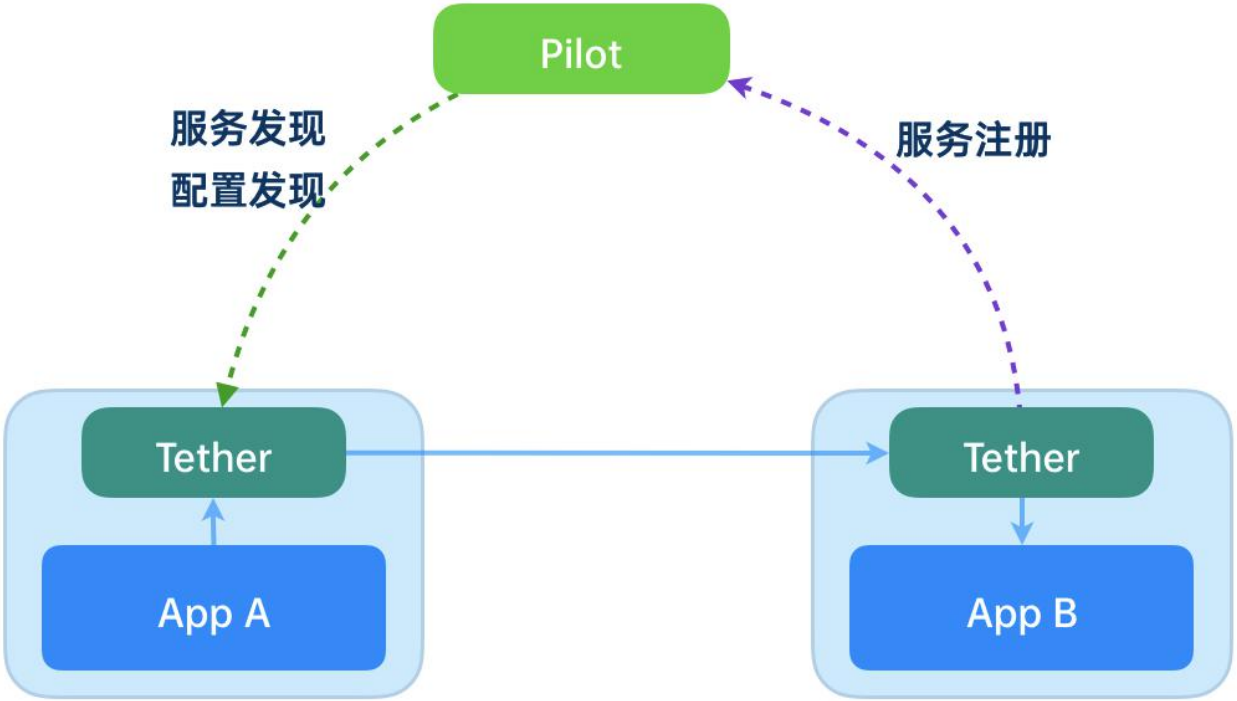

**预热算法**

# 取得的收益

## Sidecar 接管服务注册

服务注册与发现闭环

- 应用完全解耦注册与发现
- 低成本支持多语言应用
- 更多透明演进能力

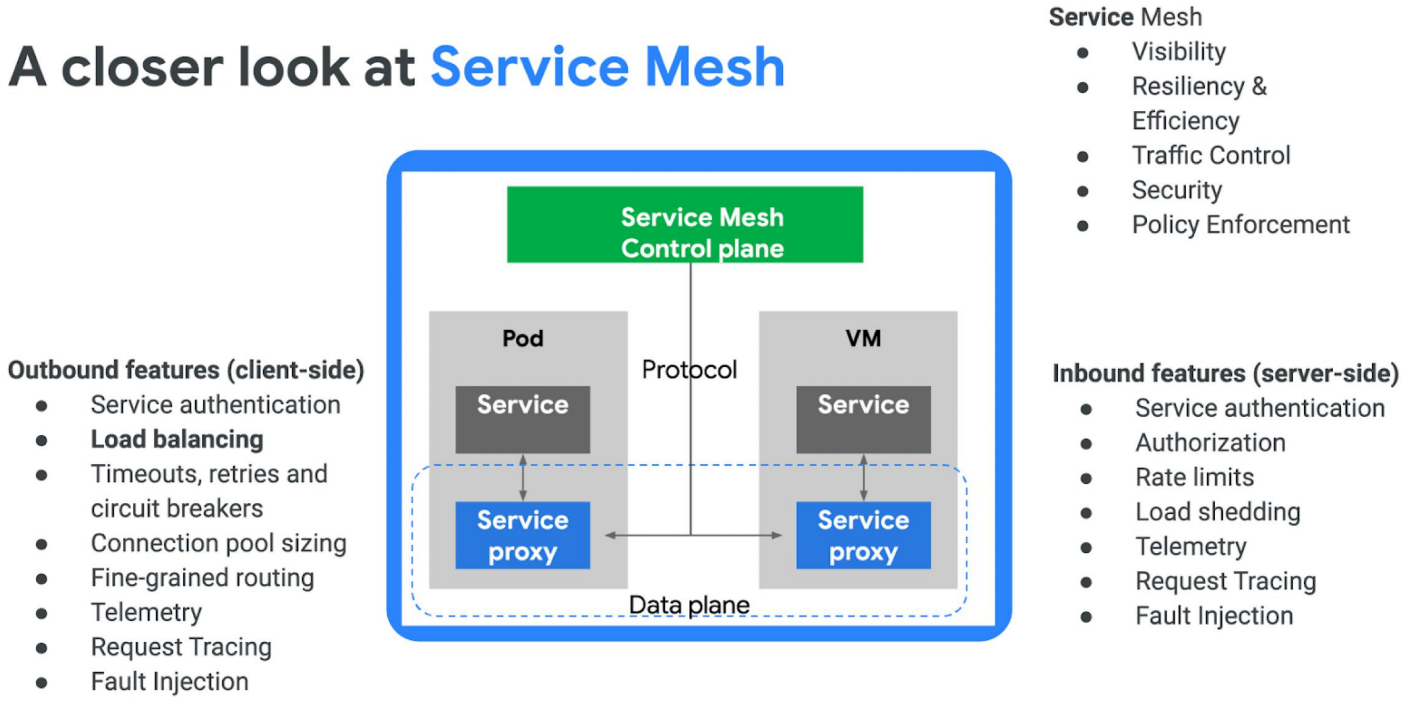


# 取得的收益

## Provider 端接入

全面接管出入口流量，Provider端提供更多能力

- 限流
- 鉴权
- 流量镜像
- 故障注入
- 可观测性



# 落地建议

# 落地建议

- 使用成熟产品，避免完全自研
- 关注业务价值
- 稳定优先

msup

# 关注性能、提升核心压测数据

- 优化协议解析性能
- 大规模服务集群场景请求路由优化
- 服务依赖、服务发现预热

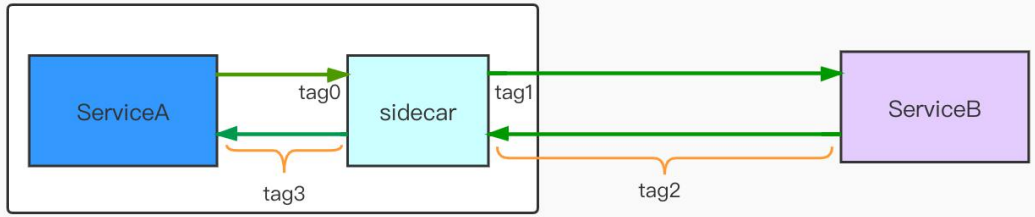
# 落地建议

msup

## 数据面接入调用链

调用链中记录关键时间信息，提升业务排查问题的效率

调用链中默认不显示sidecar的span信息，减少干扰

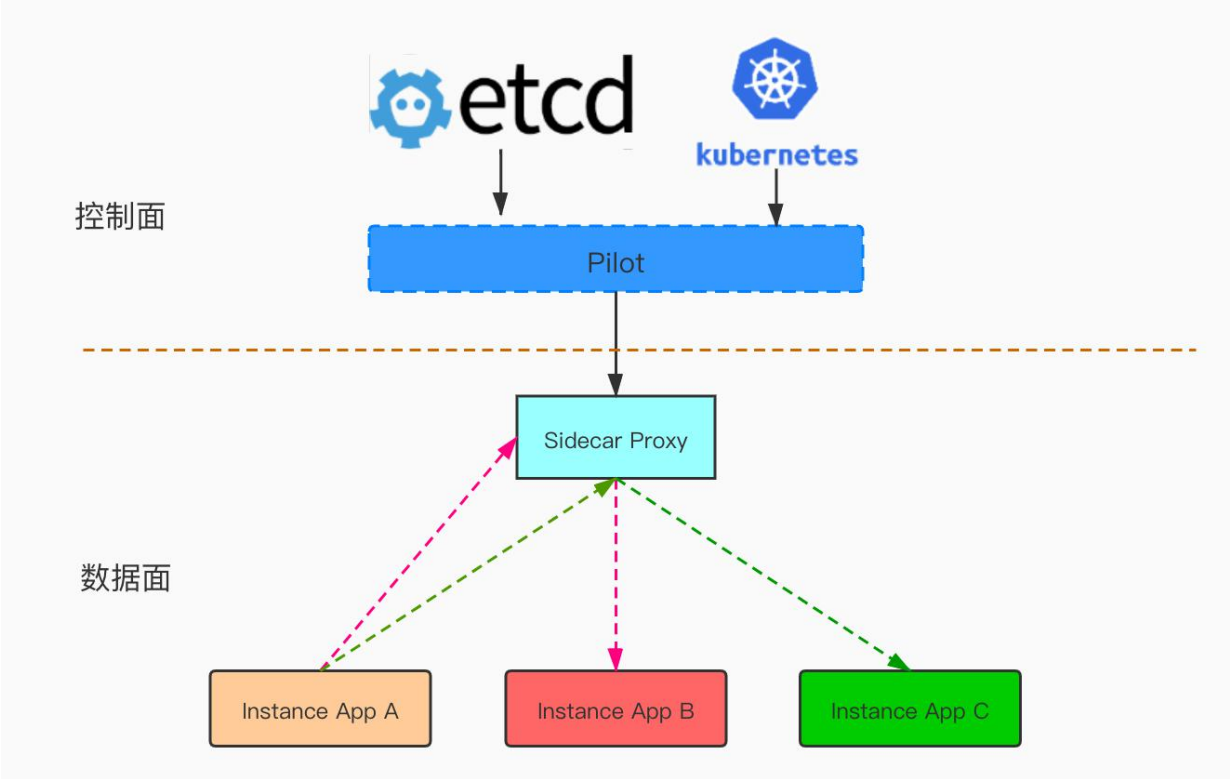



tag0: parse_request_at　　接收并完成请求解析的时间戳
tag1: proxy_request_at　　向后端发送请求的时间戳
tag2: receive_response_after　向后端发送请求至收到响应的时间
tag3: finish_proxy_after　　接收请求至响应写回业务服务的时间

**Logs**


```
[ 2 items
  0 : { 2 items
    "proxy_request_at" : "2021-01-17 18:31:32.366301"
    "receive_response_after" : "835.513µs"
  }
  1 : { 2 items
    "parse_request_at" : "2021-01-17 18:31:32.366248"
    "finish_proxy_after" : "912.629µs"
  }
]
```

# 落地建议




- QA环境集中化部署
- 方便运维和升级
- 大流量、高并发测试

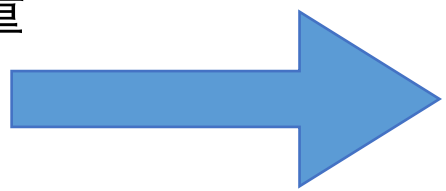
# 展望

# 未来展望

msup

## Service Mesh 完全落地

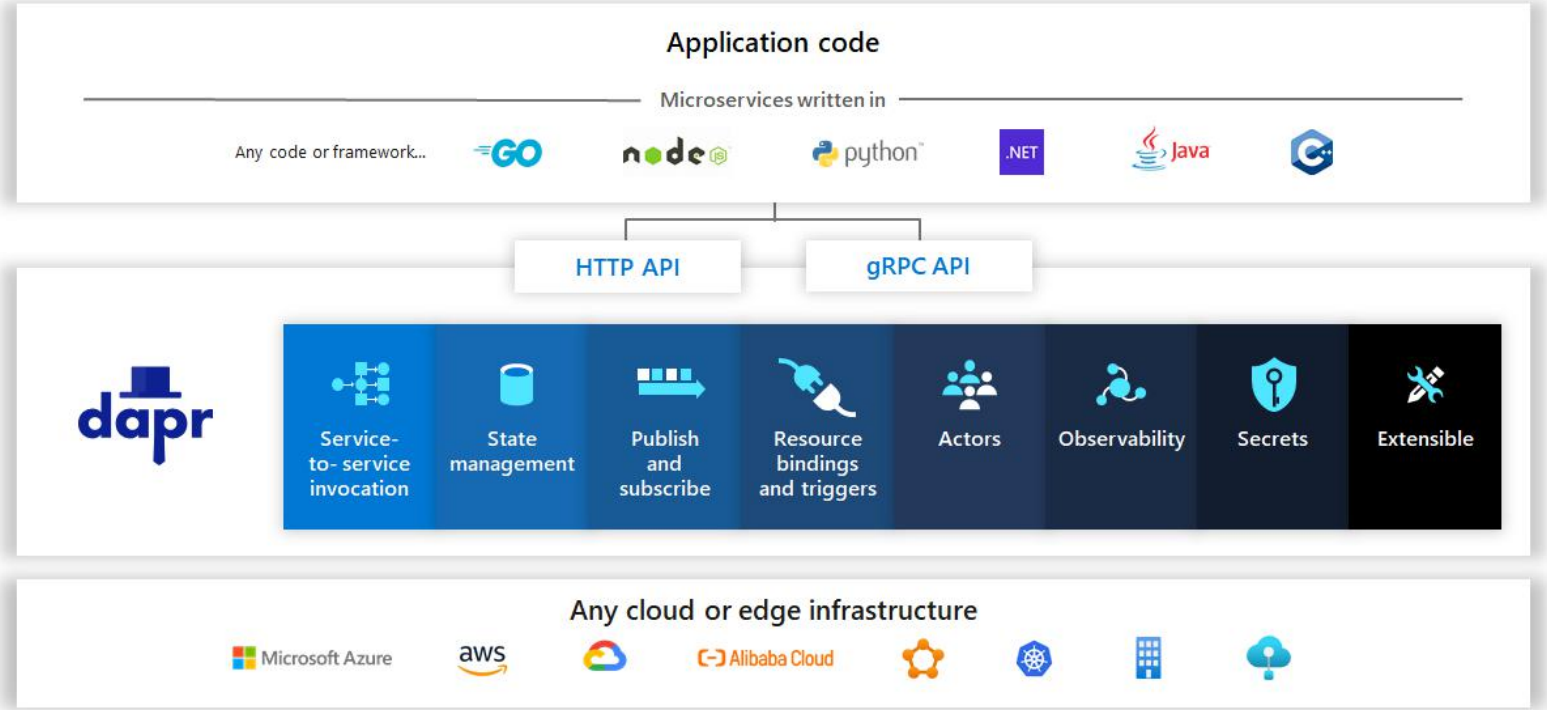
- Sidecar 接管服务注册
- Sidecar 接管 Provider 端流量



## 挖掘更多业务价值

- 自适应限流
- 自适应负载均衡
- 服务可见性、权限控制
- 更完善的可观测性
- 产品化、挖掘数据价值
- 。。。

# 分布式应用运行时




Maybe FaaS?

https://docs.dapr.io/concepts/overview/

msup®

# 感谢