

《Istio大咖说》第4期

如何让 Istio 在大规模生产环境落地



主持人：宋净超（Tetrate）



嘉宾：陈鹏（百度）



6月23日

晚8:00 – 9:00

联合主办方：



扫码观看直播



tetrate



THE ENTERPRISE SERVICE MESH COMPANY

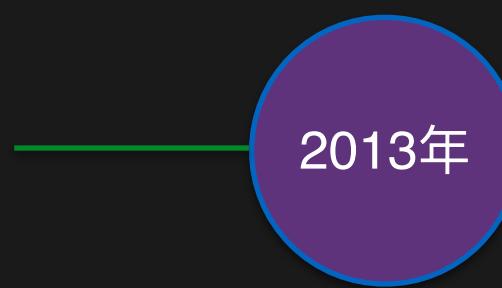


大纲

- 百度服务治理现状 & Istio 落地的挑战
- Istio 在百度生产环境大规模落地的实践分享
- 实践思考 & 落地 Mesh 的一些建议

百度 Service Mesh 编年史

内部探索



搜索并行服务系统

开始研发独立的 proxy 代理流量

大规模服役

Service Mesh 在百度的萌芽



百度网盘UFC系统

Proxy + 控制中心 设计

强大的能力
服务注册 / 发现
路由 / LB / 流量复制 / 故障注入
Trace / 仪表盘

超大规模落地
实例数 20w 千亿PV
机器节点10w+ / 10 IDC

跟进社区

Service Mesh 概念首次提出



自研 Service Mesh 系统 BMes

控制面 + 数据面 均采用 Go 语言实现
搜索服务前端模块上线



Istio 1.0版本发布

可用于生产环境

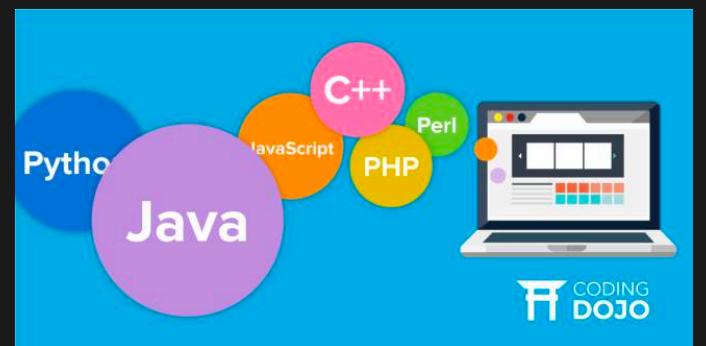


基于 Istio + Envoy 深度研发

多个核心业务大规模落地



为什么要引入 Service Mesh



多语言技术栈能力不统一

C++: bRPC

Java: Spring Cloud

PHP: RAL

Go: GDP

...

服务治理周期长

lib库和业务进程耦合

上线周期长、推动困难

策略调整周期长达数月

产品能力弱

以配置文件为主

可视化能力弱

缺乏平台化和产品化能力

QCon 分享上的一个小调查

你的业务是否全都是 http 协议？

几乎没有

你的业务是否全已经全都托管在 K8S 上？

比例稍微多一些，但占比还是比较少

Istio 落地的遇到的最普遍问题

Istio 原生几乎只支持 HTTP 协议

Envoy 虽然支持部分其他协议，如 thrift, dubbo 等

但存在一些问题：

1. 这些协议的 network filter 功能都比较弱，如不支持高级的 content based routing 等
2. 控制面 Istio 中 VirtualService 不支持非 HTTP 协议的策略配置，只能通过 EnvoyFilter 下发，体验不好

Istio 几乎只能在 K8S 环境运行良好

最新版本 Istio 虽然支持了虚拟机，但支持程度有限

1. Istio 无法自动 watch K8S Service 资源，需要手动或其他系统生成 ServiceEntry / WorkloadEntry 资源
2. Sidecar 注入和管理困难，需要 PaaS 适配

Istio 在生产环境落地面临的问题

稳定性 产品能力 性能问题

服务发现集成 框架支持

生产级流量调度策略缺失 私有协议支持

Sidecar 注入 大规模 Sidecar 管理

K8S 依赖

问题：当我们说 Istio 依赖 K8S 时，
我们到底在讨论什么？

Istio 对 K8S 的深度依赖



Sidecar 注入和管理

透明注入提升用户接入体验
依托 K8S 对大规模 Sidecar 管理

升级内部 PaaS
支持 Sidecar 模式



Istio 配置存储

各类流量策略 CRD
通过 Api Server + Etcd 读写

予以保留，独立部署 K8S
Api Server + Etcd 两个组件

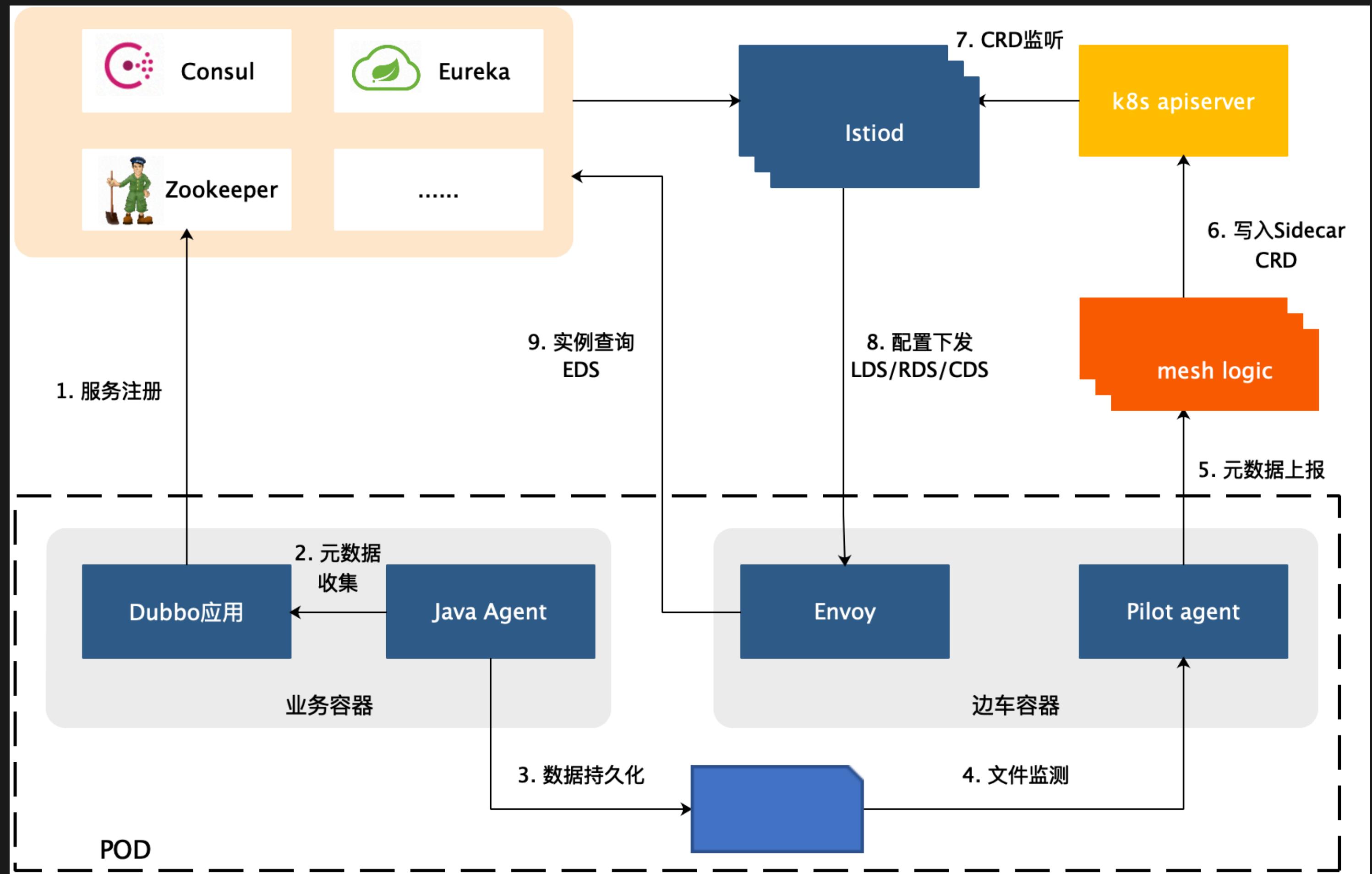


K8S 服务和流量模型

ClusterIP
容器网络
Iptables 流量劫持

内部环境无法使用 Iptables
因素：性能、可管控性、无容器网络
研发新的流量劫持模式

内部架构方案一



自动生成Sidecar

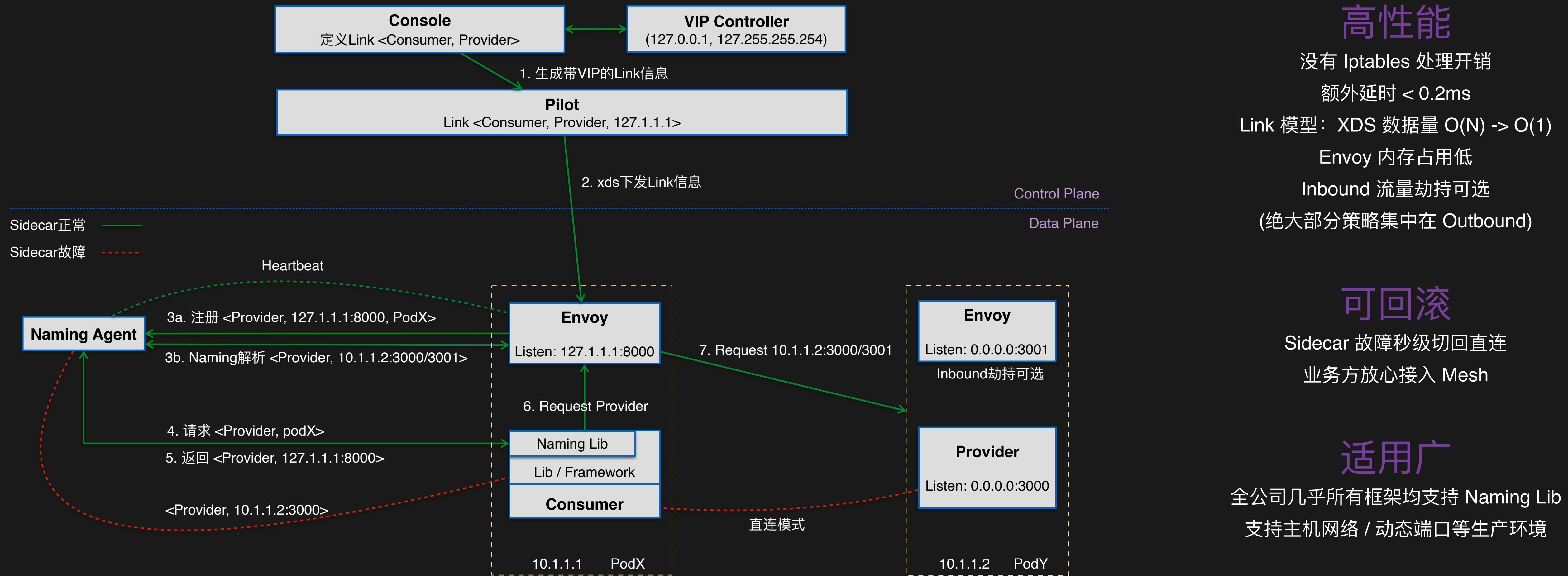
大幅降低 xds 推送量

降低 Envoy 内存占比

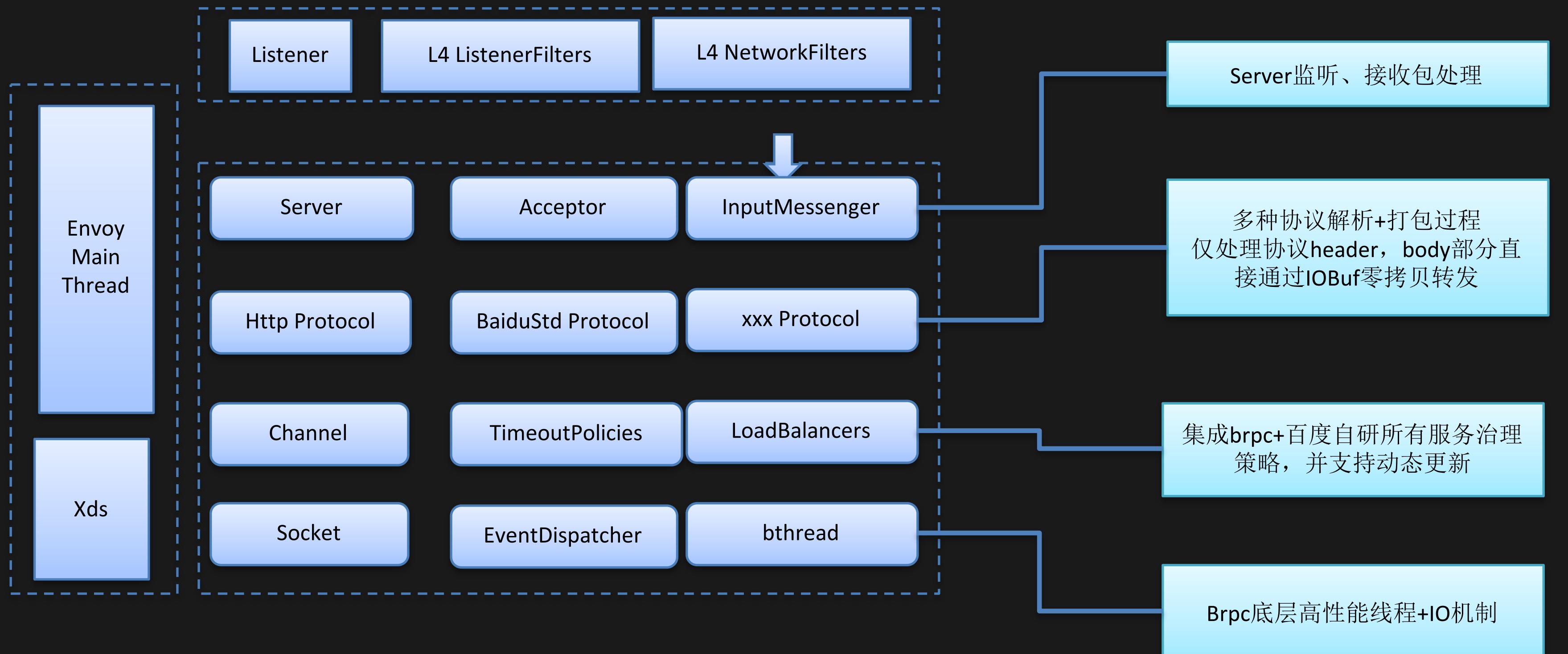
数据面服务发现

支持百万级别实例

内部结构方案二（目前大规模落地）



极致性能优化 融合 Envoy 和 bRPC



Envoy 优点

灵活 Filter 扩展机制

丰富的策略

XDS 动态配置分发

bRPC 优点

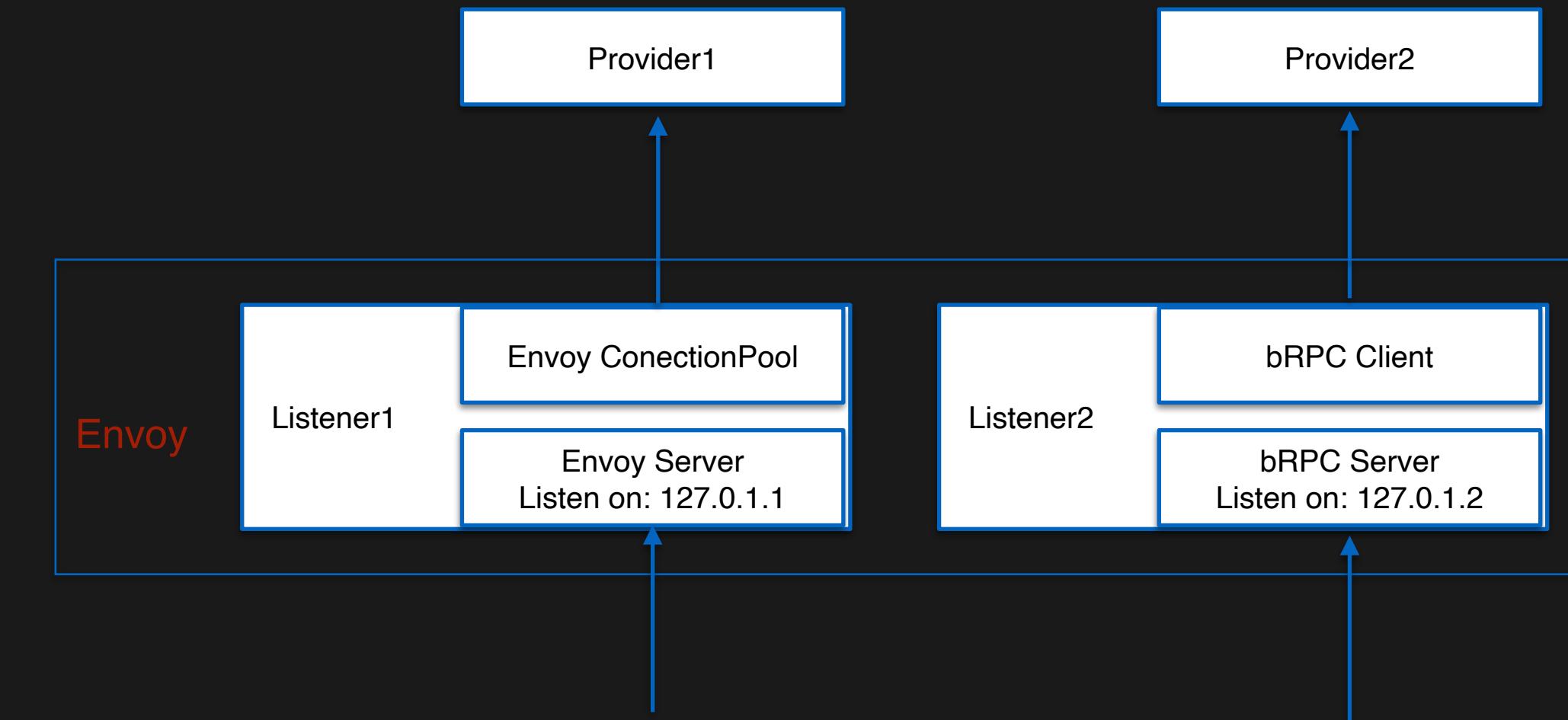
高性能用户态线程

高性能 IOBuf 库

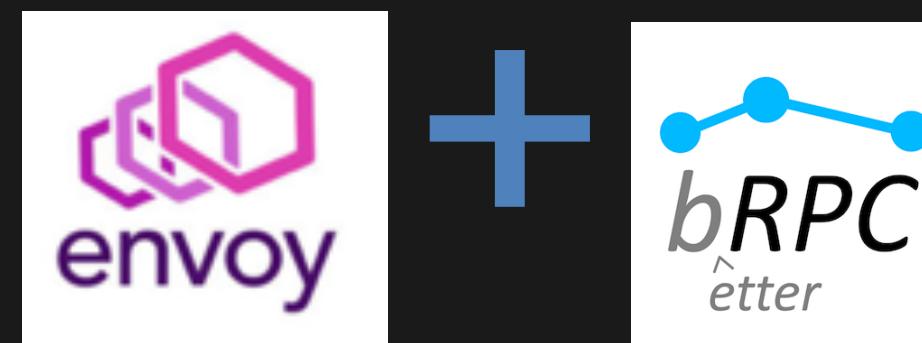
优秀的 IO 机制

Envoy 和 bRPC 内核动态切换

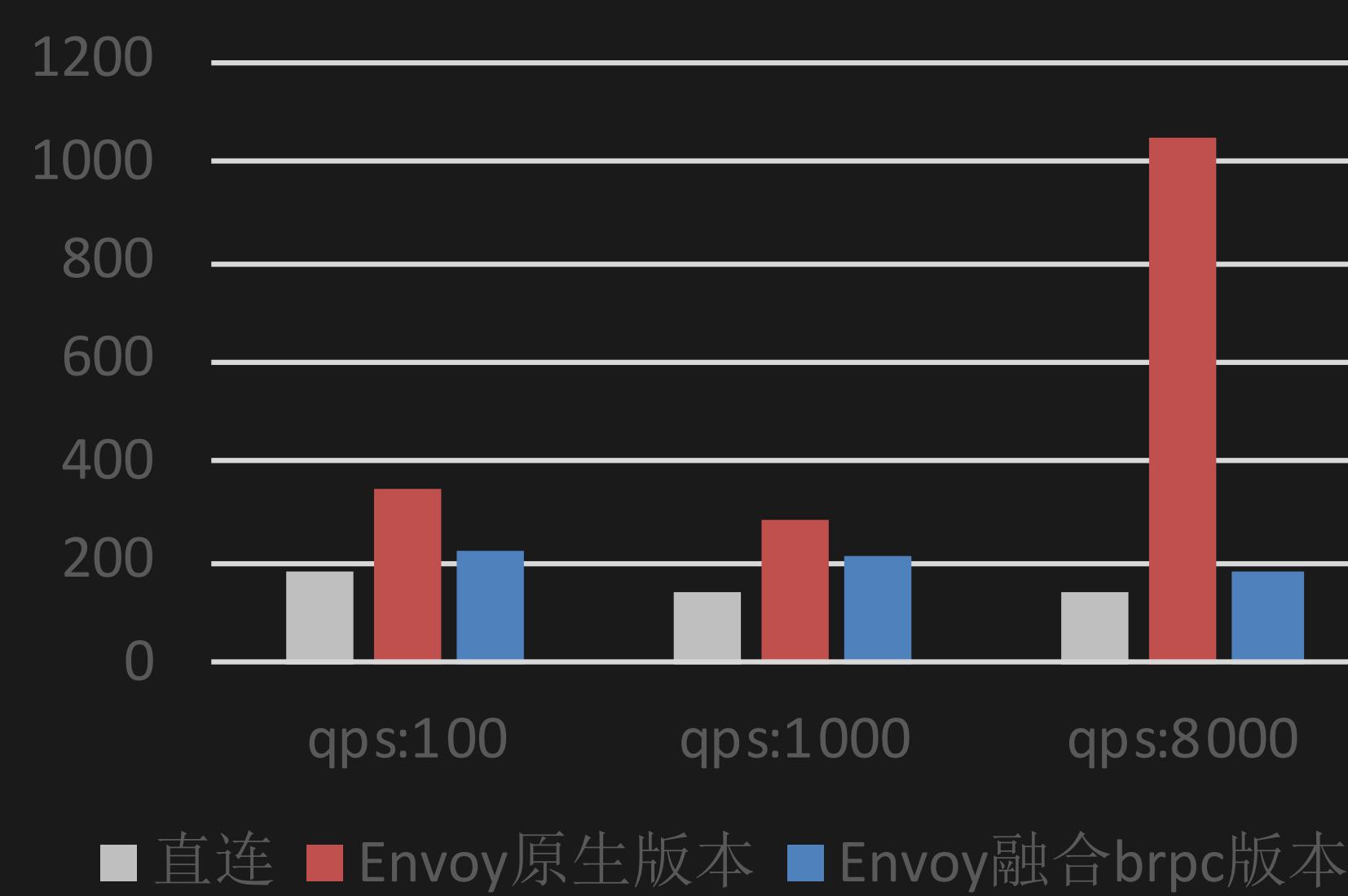
```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: Sidecar
3 metadata:
4   name: consumer
5 spec:
6   workloadSelector:
7     labels:
8       app: consumer
9   egress:
10  - hosts:
11    - "./provider1.default.bj"
12    port:
13      number: 1
14      protocol: http
15      name: main
16      bind: 127.0.1.1
17  - hosts:
18    - "./provider2.default.bj"
19    port:
20      number: 1
21      protocol: http
22      name: main
23      bind: 127.0.1.2
24  useBpcServer: true
```



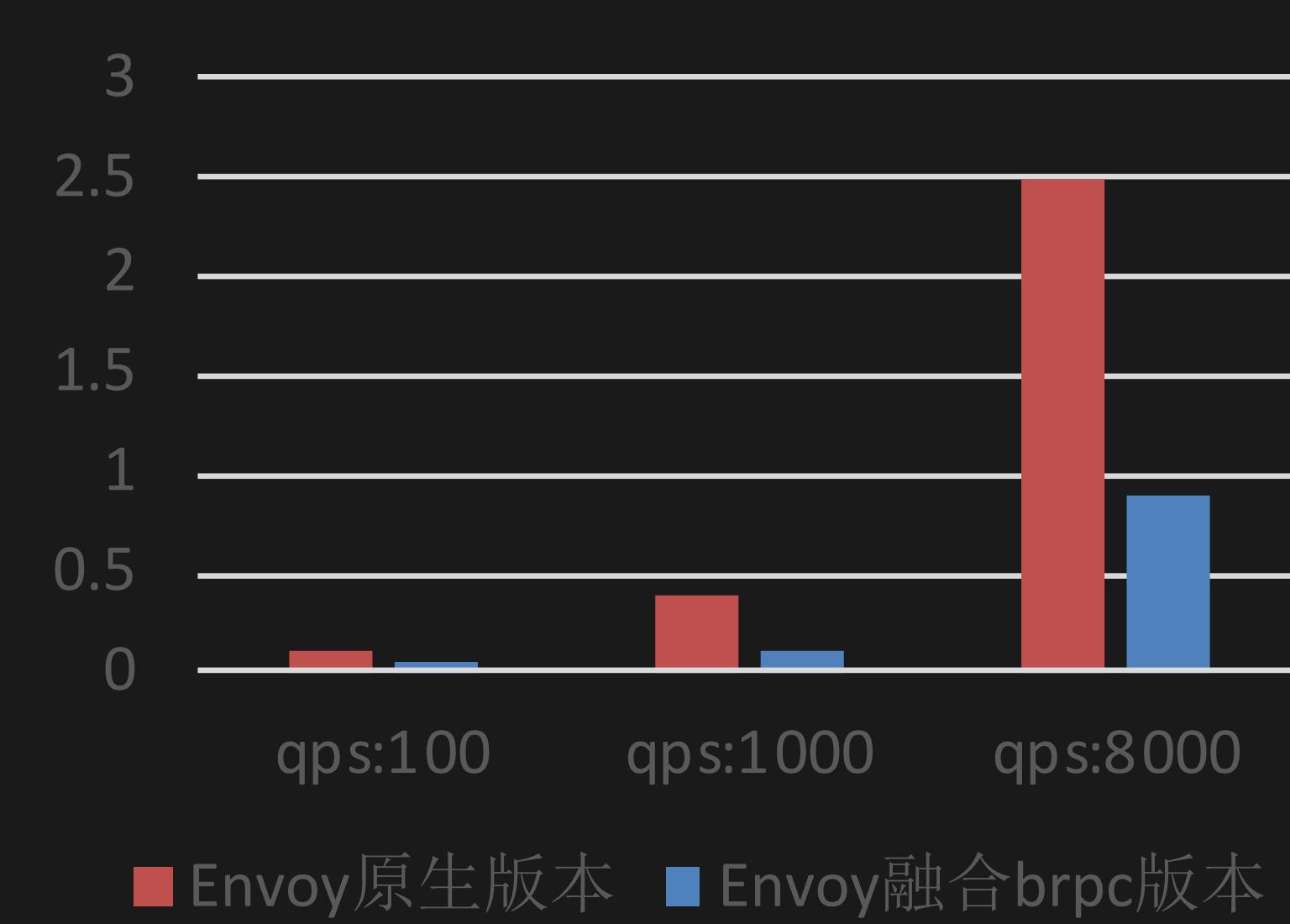
极致性能优化 延时和开销大幅降低



平均延时 (单位us)



CPU使用率 (%)



8k qps 延时
1ms -> 0.2ms

8k qps cpu
2.5 -> 0.9

问题：

Istio 不支持业务线的私有协议怎么办？

Istio 不支持业务线在用的高级服务治理策略怎么办？

支持私有协议和高级策略

私有协议支持

全面支持baidu std / nshead / http / dubbo

私有协议自动嗅探

流量黑洞

流量镜像

快速熔断 流控预案

线上流量 线下测试 / 训练模型

BackupRequest DynamicBP

快速重试 优化长尾

阈值保护 防止雪崩

降低业务接入成本

接入 Mesh 不会造成能力缺失

策略实时生效 提升服务治理效率

赋能其它框架 / 语言

Sidecar 模式 策略天然和框架及语言无关

一次实现 所有框架和语言复用

LocalityAware LocalityAwarePlus

综合吞吐、延时、错误码计算权值

负载流量更均匀

Istio原生产品能力缺失

VirtualService.yaml
DestinationRule.yaml
ServiceEntry.yaml
Sidecar.yaml
EnvoyFilter.yaml
Gateway.yaml
PeerAuthentication.yaml
...

Istio CRD众多

Kiali 能力较弱，不易扩展和集成其它基础设施（账户、权限、审计等）

Cloud Native = yaml工程师

内部产品 - 服务治理

基本信息

规则名称: rule-2

流量来源

服务调用方: echo-client-multi-1
集群标签 (LIDC): nj03
命名空间 (namespace): eks-aig-demo-space

流量目的

服务被调用方: echo-server-baidu-2
命名空间 (namespace): eks-aig-demo-space
总超时时间 (timeout): 1000 毫秒
重试次数 (max_retry): 1

筛选条件 (Key: Value) 权重 负载均衡 操作

集群标签 (LIDC)	值	权重	策略	操作
sz0	50	50	轮询 (Round_Robin)	更多设置 删除
nj	50	50	轮询 (Round_Robin)	更多设置 删除

网络协议: Baidu-std/Nshead

流量策略

连接池配置 (connection_pool) ?

连接超时时间 (connect_timeout) ?: 100 毫秒
最大连接数 (max_connections) ?: 10000

TCP长连接 (tcp_keepalive) ?

探测间隔 (interval) ?: 75 秒
探测次数 (probes) ?: 9
前置等待时间 (time) ?: 7200 秒

异常点检测和驱逐 (outlier_detection) ?

最短拒绝访问时长 (base_ejection_time) ?: 30 秒
连续失败次数 (consecutive_errors) ?: 5
探测间隔 (interval) ?: 10 秒
最大剔除占比 (max_ejection_percent) ?: 10 %
最小健康占比 (min_health_percent) ?: 10 %

主动健康检查 (health_check) 关 ?

请求备份 (backup_request) ?

请求备份模式
 静态backup_request 动态backup_request

静态请求备份时间 (backup_request) ?: 1000 毫秒

丰富的服务治理策略
易用的UI体验

内部产品 - 运维管理

The screenshot shows a Service Mesh management interface with two main sections: '批量切流' (Batch Traffic Switching) and '服务网格开关' (Service Mesh Switch).

批量切流 (Batch Traffic Switching):

- 调用方:** echo-client-multi-1
- 被调用方:** echo-server-baidu-1
- 命名空间 (namespace):** eks-aig-demo-space
- 服务链路ID:** sc-GpYEDb92
- 集群标签 (LIDC):** 重置

流量配置 (Traffic Configuration):

- 流量来源: bj**
bj: 30% (滑块: 30)
nj: 70% (滑块: 70)
- 流量来源: nj**
nj: 100% (滑块: 100)

服务网格开关 (Service Mesh Switch):

- 服务调用方:** echo-client-multi-1
- 开关状态:** 开 (On)
- 命名空间 (namespace):** eks-aig-demo-space
- 集群标签 (LIDC):** nj
- 服务路由开关:** 开 (On)

机房切流
链路级服务网格开关
快速故障止损

内部产品 - 版本控制

操作记录

2020-12-02 14:28:53 - 2020-12-03 14:28:53 1d 7d 操作类型: 已选择3项 搜索

记录ID	操作类型	操作者	操作时间	详情
re-OPHhxCE6	路由发布	[REDACTED]	2020-12-03 11:40:28	对 sc-h354HXHv 链路下的路由规则发布了 50 版本
re-WspQSS7w	批量切流	[REDACTED]	2020-12-02 17:07:05	对 sc-GK7xjGsS 链路进行了切流操作
re-cVJVsbMZ	路由发布	[REDACTED]	2020-12-02 17:04:51	对 sc-GK7xjGsS 链路下的路由规则发布了 54 版本
re-zWcTKefX	路由发布	[REDACTED]	2020-12-02 17:00:04	对 sc-3vZBQFkd 链路下的路由规则发布了 53 版本
re-cctBea28	路由发布	[REDACTED]	2020-12-02 16:54:55	对 sc-GpYEDb92 链路下的路由规则发布了 54 版本
re-Y2WN2Rws	路由发布	[REDACTED]	2020-12-02 16:50:54	对 sc-GpYEDb92 链路下的路由规则发布了 53 版本
re-45aRPB07	路由发布	[REDACTED]	2020-12-02 16:40:54	对 sc-GK7xjGsS 链路下的路由规则发布了 52 版本

每页显示 10 < 1 > 跳转至 GO

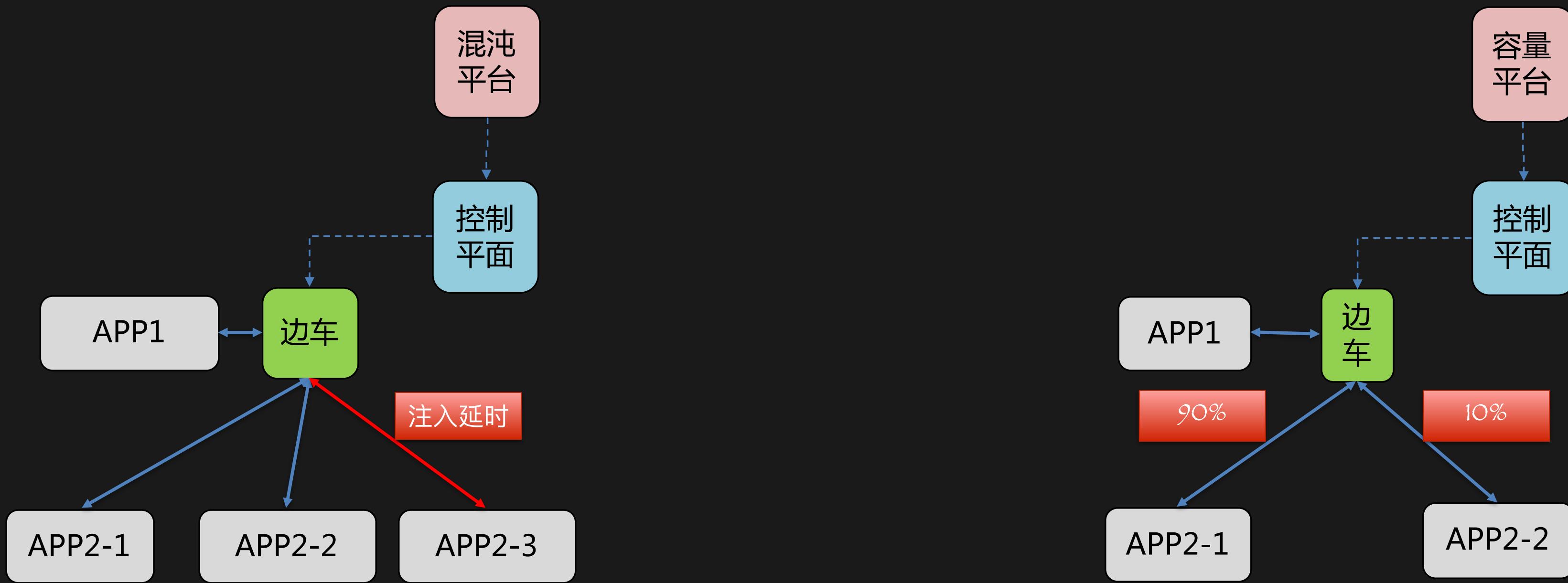
所有操作可溯源
一键回滚至任一版本

Mesh 落地后的更多价值

有了 Mesh 之后我们还能做什么？

Mesh 使整个服务网络变得可编程
因此，我们可以基于 Mesh 做更多的事情

平台化能力



混沌工程

混沌平台负责调度任务、效果评估

服务网格负责故障注入

容量压测

容量平台负责发起评估任务、评估效果

服务网格负责实施流量调度

线上效果 - 可用性提升



Service Mesh 落地全景图

赋能业务

手机百度 / Feed

百度地图

小程序

好看视频

公有云 / 私有化

落地场景

服务治理

运维管理

混沌工程

Mesh能力

流量路由

负载均衡

自动调参

故障止损

容量压测

故障注入

Log / Monitor / Trace

Mesh组件

Mesh Console

AI调参系统

控制面Istio

服务 + thin SDK

数据面Sidecar

K8S

自研 PaaS Pandora / Opera

Baidu Naming

Docker

自研容器 Matrix

基础设施

百度智能云 云原生微服务平台

产品概述和架构

云原生微服务应用平台（Cloud-Native Application Platform，简称CNAP）是一个为企业提供应用托管和微服务管理能力的PaaS平台，帮助企业简化部署、监控、运维等应用的生命周期管理工作，同时支持微服务注册、服务治理、服务监控、调用链、API网关和分布式事务等一站式微服务管理和运维能力。



集成各类功能和协议扩展、性能优化
支持 公有云 / 私有化部署

落地规模及收益

业务线

规模

收益



手机百度

服务：数千个

跨语言统一服务治理能力



Feed

实例：10w

服务可用性提升



百度地图

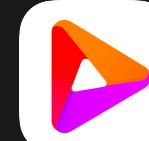
流量：百亿+

服务治理周期：月级 -> 分钟级



小程序

变更效率提升：10X



好看视频

服务治理体验提升

总结：Service Mesh 落地的核心要素

务实

适配公司基础设施

实现生产线服务治理策略

让业务方能低成本
接入 Mesh

性能

极致性能优化

打消业务方性能顾虑

让核心业务能够
接入 Mesh

稳定

完善监控

链路级 Mesh 开关

Sidecar 故障自动秒级回滚

让业务方敢大规模
落地 Mesh

体验

平台化

产品化

拒绝做 yaml 工程师

让业务方觉得
Mesh 好用

关于需不需要引入 Mesh

不一定要引入 Mesh 的场景

- 业务几乎都是单语言，有成熟的框架提供服务治理能力（如 Java + SpringCloud）
- 治理策略变更不频繁，可以通过上线解决
- 框架升级不频繁，不会和业务升级相互耦合、影响

引入 Mesh 反而会带来技术复杂度和性能开销

适合引入 Mesh 的场景

- 业务场景跨多语言，各语言框架服务治理能力不统一，对业务整体可用性造成影响
- 策略变更频繁，有动态服务治理的强需求
- 框架迭代频繁，需要和业务解耦
- 想基于 Mesh 做更多上层的服务治理场景（混沌、智能调参、容量压测等），需要整个网络通信层可编程
- 业务规模庞大，需要一种可复制的基础设施，满足规模化服务治理需求

引入 Mesh 可以解决若干痛点问题，提升效率

引入 Mesh 的成本

几乎没有成本的场景

- 业务是 HTTP 协议
- 托管在原生 K8S 上
- 业务本身是长耗时请求，或对 Mesh 引入的延时可以容忍（2ms）

基本可以直接接入

有一定成本的场景

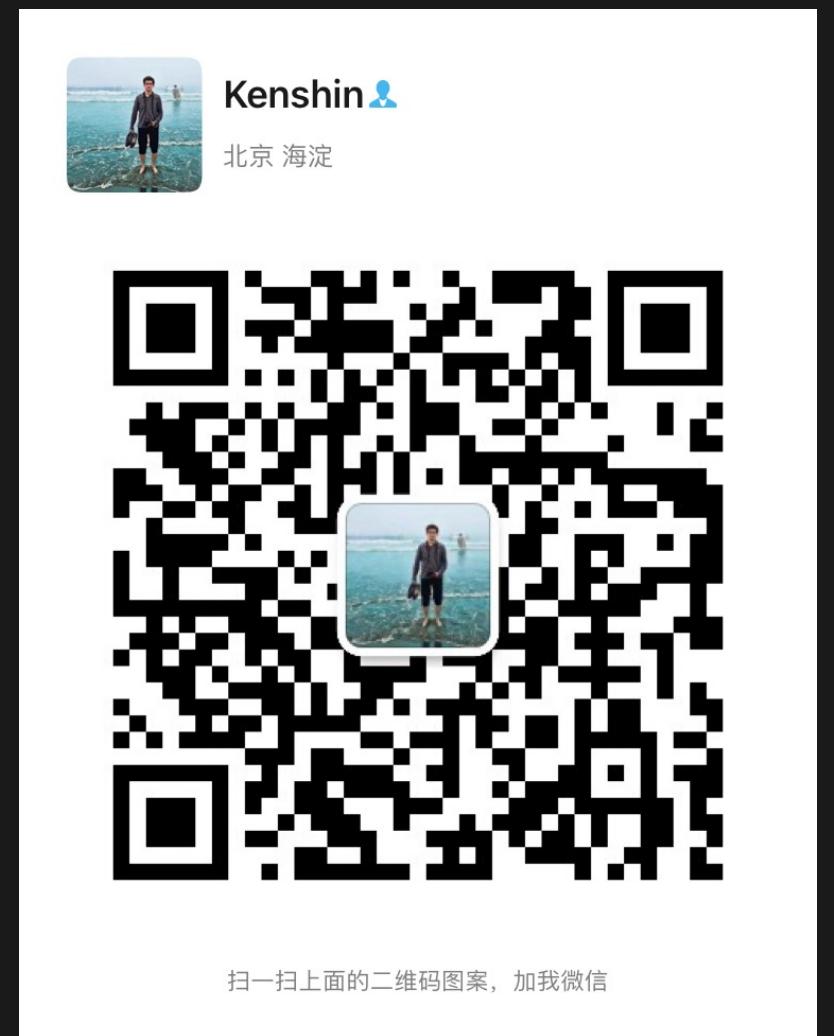
- 有私有协议（开发 NetworkFilter，控制面 API 支持）
- 托管在 K8S + 虚拟机（手动生成管理虚拟机相关 CRD 资源，解决 Sidecar 注入和管理问题）

需要一定的研发人力支持

高成本的场景

- 业务规模大，大量私有协议需要开发支持
- 托管在 K8S + 虚拟机 + 无容器网络 + 私有注册中心（还需要额外处理流量劫持问题）
- 大量业务高级流量调度策略（需要开发支持）
- 对性能、延迟有要求（需要深入优化 Istio + Envoy 性能）
- 服务配置需要权限控制、审计，有用户体验要求（需要一个产品团队开发上层产品）

需要一定规模的研发+产品团队支持



个人微信
欢迎大家线下交流



团队公众号
百度系列Service Mesh等云原生相关文章

Thank you!



Tetrate 中国



云原生社区