Service Mesh Meetup

CLOUD NATIVE COMPUTING FOUNDATION

ServiceMesher

严选 ServiceMesh 实践

王国云

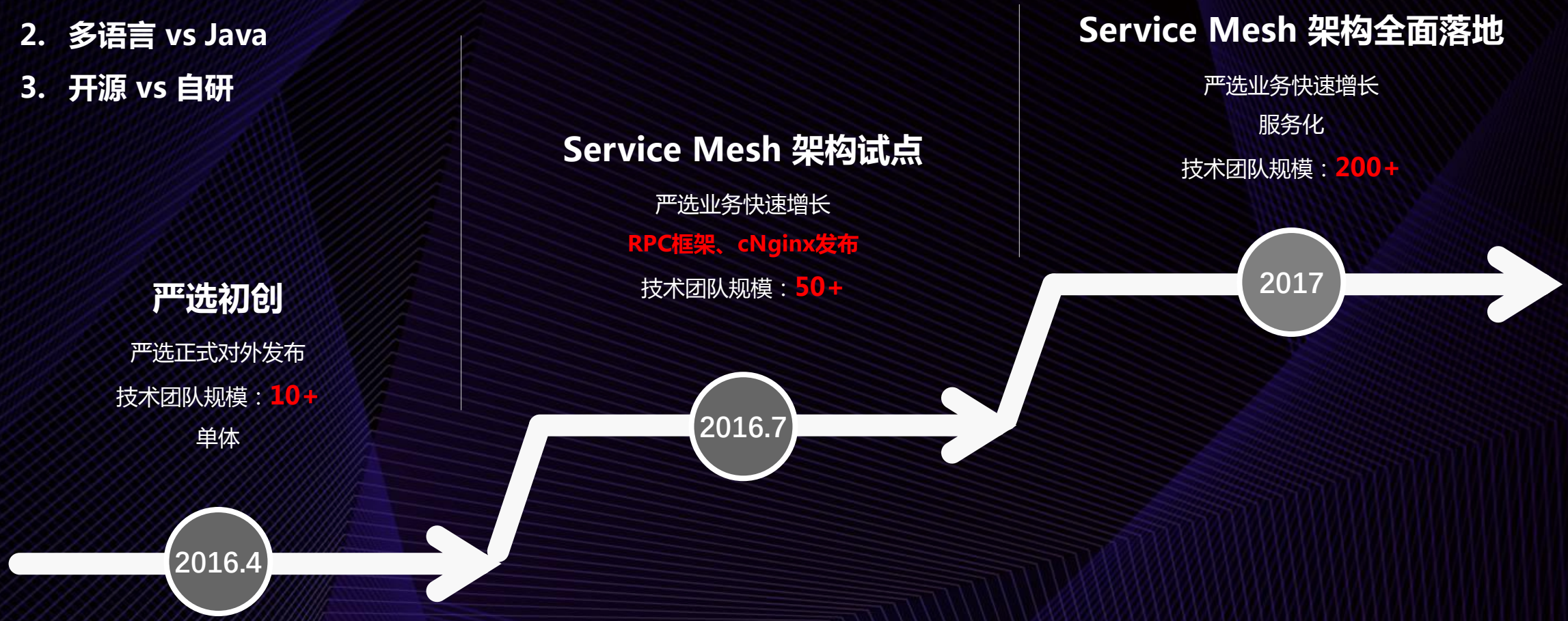
网易资深专家

严选中台技术团队负责人/容器化负责人

背景

基础架构三问：

1. **服务治理：RPC 框架 vs 服务治理平台**
2. **多语言 vs Java**
3. **开源 vs 自研**



CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

/01 严选 Service Mesh 演进

/02 混合云架构落地实践

/03 规划与展望

CLOUD NATIVE COMPUTING FOUNDATION

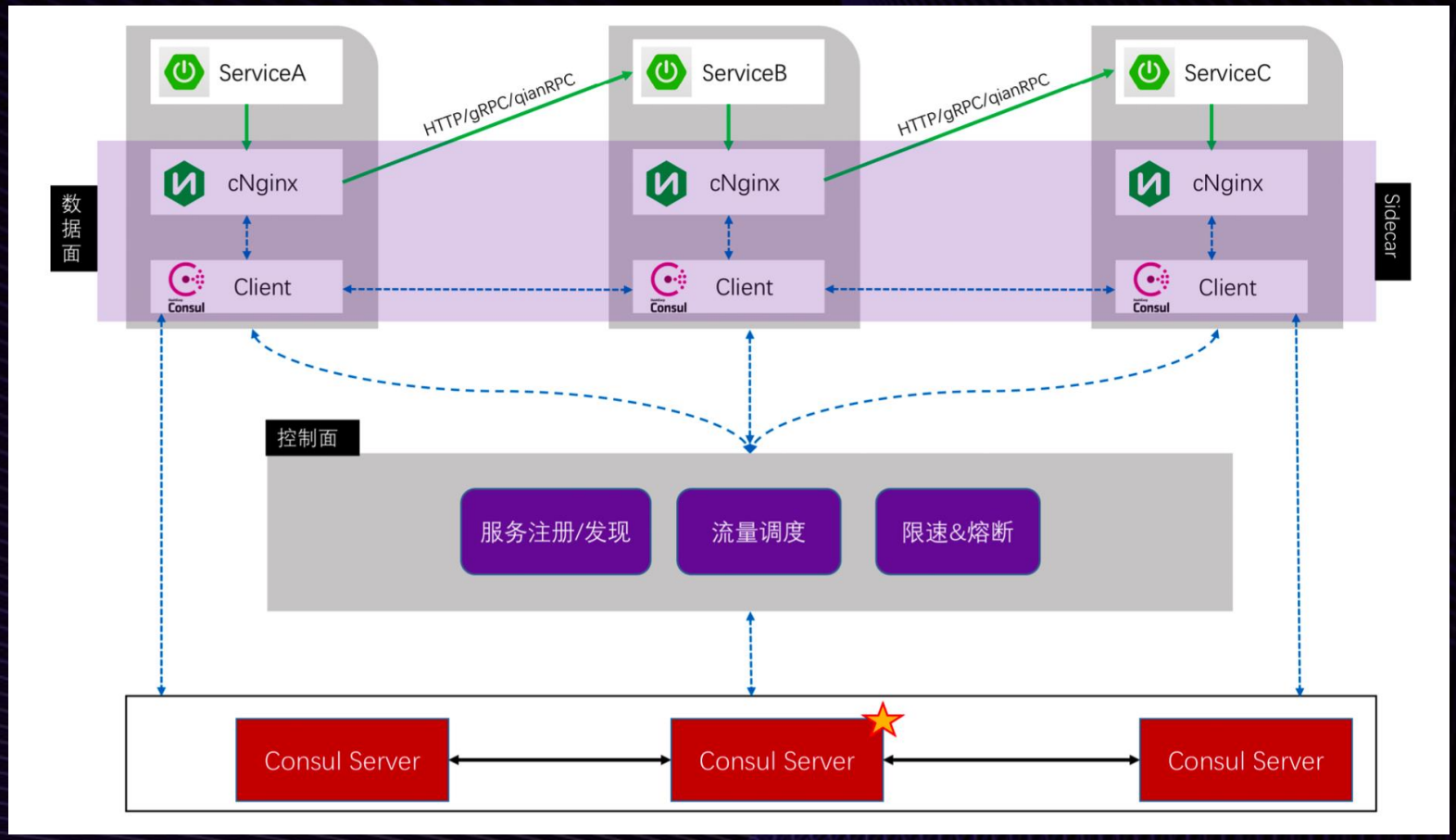
ServiceMesher

/01

严选ServiceMesh演进

严选第一代 Service Mesh 架构

- 使用 Consul 作为服务发现组件
- 数据面：cNginx
- 控制面：Consul 管理后台



服务治理能力 – 基于严选第一代ServiceMesh（cNginx）

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

类型	功能	能力提供方	
		服务调用方（Client）	服务提供方（Server）
服务注册与发现	注册发现：基于 Consul	√	
调用控制	协议支持：HTTP 1.X/2.X，可扩展至 TCP	√	
	路由控制：提供简单的路由能力	√	
	负载均衡：支持 RR、权重、一致性 Hash 等	√	
	流量复制：**不提供**	×	
	故障转移：继承 Nginx 的 Failover 机制	√	
安全	访问控制：**主要依靠中间件**	×	中间件
治理控制	熔断降级：**主要依靠中间件**	中间件	
	限流：速率限制	√	中间件
	资源隔离：**主要依靠中间件**	中间件	
	故障注入：**不提供**	×	
	超时控制、重试、重写、重定向等：继承 Nginx 的 timeout 机制	√	
监控/故障诊断	链路追踪：**主要依靠中间件**	APM	APM
	性能监控：**主要依靠中间件**	APM	APM
	遥感数据：**主要依靠中间件**	APM	APM
	访问日志：**主要依靠日志平台**	日志平台	日志平台

Service Mesh 为严选带来了哪些架构收益

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

- 历史包袱：现有的服务在不改造的情况下引入了服务治理能力
- 大大降低了中间件的研发投入和演进成本，也降低了业务和中间件的耦合成本
- 基础架构与业务架构可以独立演进
- 为多语言栈提供了服务治理能力

持续演进的诉求

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

- **提供高质量的服务治理能力**
  - 增强流量管理能力
  - 将更多治理特性（如限流、熔断、故障注入）与业务架构解耦
  - 支持更多的协议
  - 增强控制面
- **配合业务容器化上云及混合云架构**

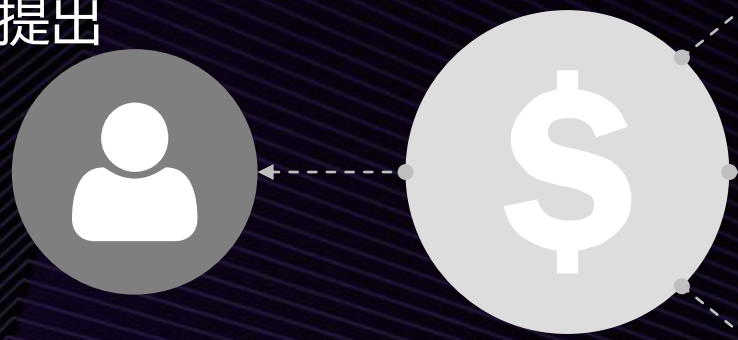
行业技术演进 - 通用型 Service Mesh 出现

CLOUD NATIVE COMPUTING FOUNDATION

ServiceMesher

Service Mesh 概念正式提出

2016年9月29日第一次被公开提出



Linkerd

2017年1月23日 加入 CNCF

2017年4月25日 1.0Releases

CONDUIT

Envoy

2016年9月13日 1.0Releases

2017年9月14日 加入 CNCF

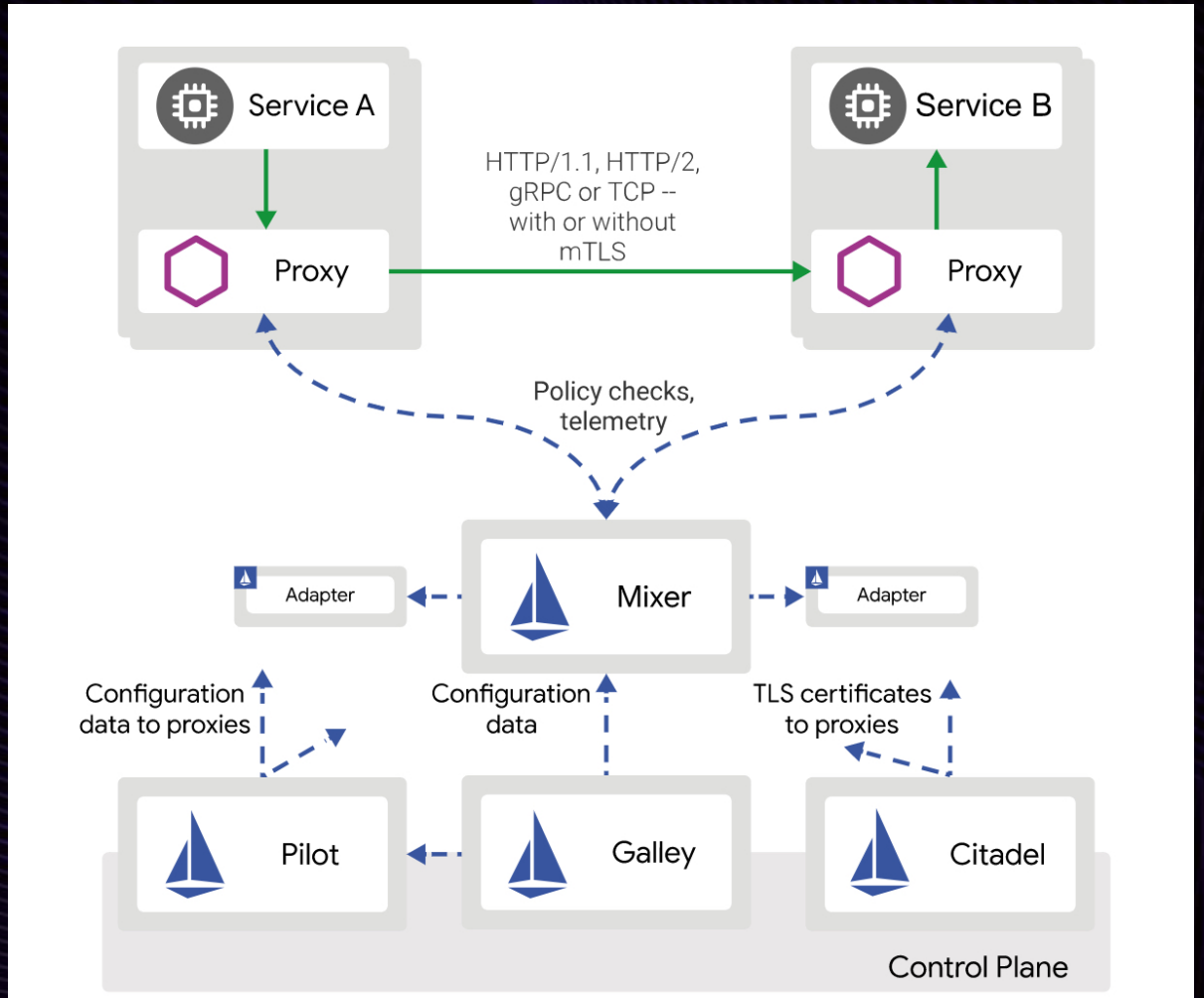
nginmesh

2017年9月15日 发布

最新版本：0.7.1

云原生 Service Mesh 框架 - Istio

- 由 **Google**，**IBM** 和 **Lyft** 联合开发，Go 语言，与 K8s 一脉相承且深度融合
  - K8s 提供了部署、升级和有限的运行流量管理能力
  - Istio 补齐了 K8s 在微服务治理上的短板（限流、熔断、降级、分流等）
  - Istio 以 Sidecar 的形式运行在 Pod 中，**自动注入，自动接管流量，部署过程对业务透明**
- 提供了完整的 Service Mesh 解决方案
  - 数据面：Envoy
  - 控制面：Pilot，Mixer，Citadel，Galley



Istio Architecture

功能视角 - 服务治理能力 – 基于Istio+Envoy

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

类型	功能	能力提供方	
		服务调用方（Client）	服务提供方（Server）
服务注册与发现	注册发现：**云外基于 Consul，云内基于 K8s 默认的 ETCD**	√	
调用控制	协议支持：HTTP 1.X/2.X，GRPC,WebSocket，Dubbo，Thrift	√	
	路由控制：**静态路由、动态路由、流量染色、分流控制等**	√	
	负载均衡：支持 RR、权重、一致性 Hash 等	√	
	流量复制：**Envoy 自带**	√	
	故障转移	√	
安全	访问控制：RBAC vs Mixer		√
治理控制	熔断降级	√	
	限流	√	中间件
	资源隔离	√	
	故障注入	√	
	超时控制、重试、重写、重定向等	√	
监控/故障诊断	链路追踪：主要依靠中间件	APM	APM
	性能监控：主要依靠中间件	APM	APM
	遥感数据：主要依靠中间件	APM	APM
	访问日志：主要依靠日志平台	日志平台	日志平台

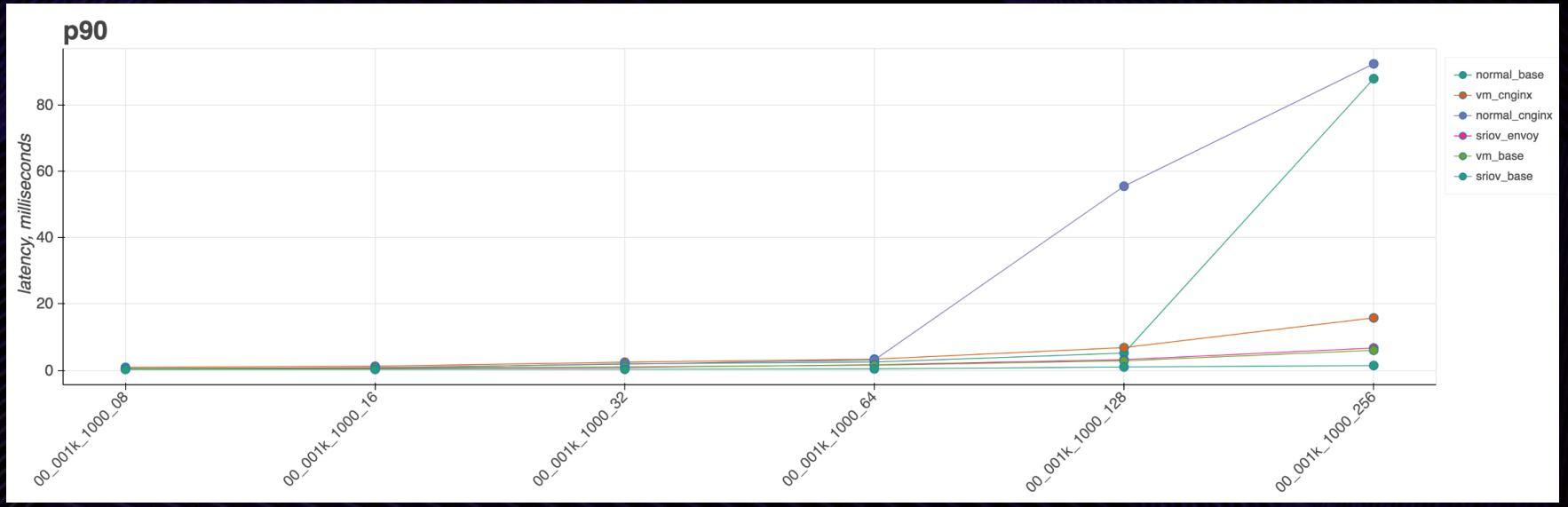
性能视角 – cNginx vs Envoy（优化前）

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

Samples	Total	Fails	FailRate	TPS	MRT(ms)	50%RT	90%RT	99%RT	MaxRT	RL(B)
sidecar-envoy - GET - ...	201646	0	0	1666.5	4.49	4	6	10	228	1668.19
sidecar-cnginx - GET -...	195723	0	0	1617.55	4.28	4	5	8	148	1668
direct - GET - http://1...	195701	0	0	1617.36	3.88	4	5	8	222	1668.19

- 1600RPS+40个并发(主机配置均为 8C16G)
- cNginx 的 RT overhead 在0.4ms左右
- Envoy(client模式)的 RT overhead 是0.6ms左右

性能视角 – cNginx vs Envoy（优化后）

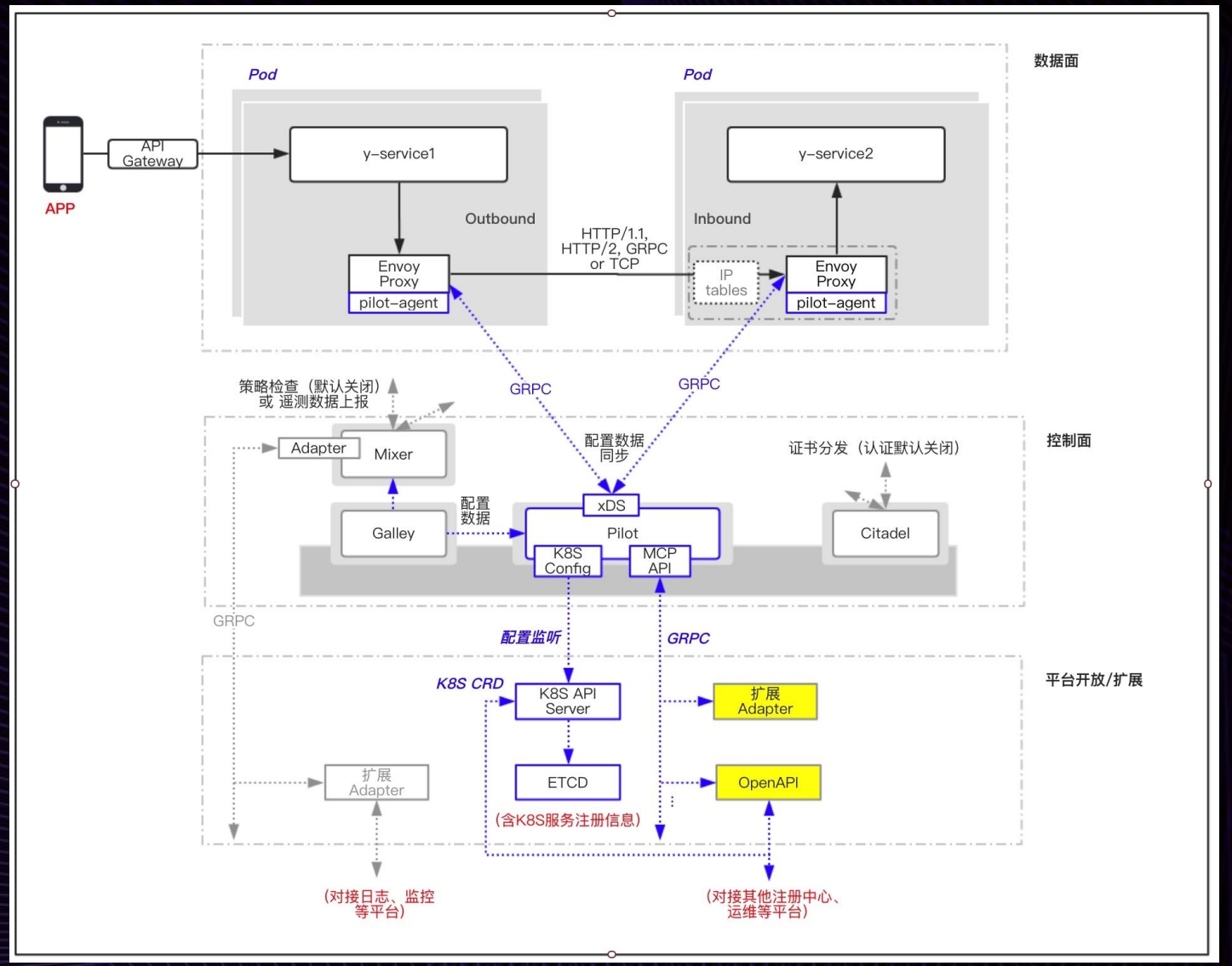


- 优化方案
  - 采用 sriov 容器网络
  - Envoy：将1.13版本中 connection loadbalancer 特性移植到 1.10.x 版本
- Envoy 优化后在低并发（<64）的情况下，容器网络 client sidecar 优于 VM 网络直连
- Envoy 优化后在高并发（>=64）的情况下
  - 容器网络 client sidecar **接近 VM 网络直连**
  - 容器网络 client sidecar 远远优于 VM cNginx（256并发 **6.707 vs 15.771**）

当前演进方向

整体基于 **Envoy+Istio** 方案：

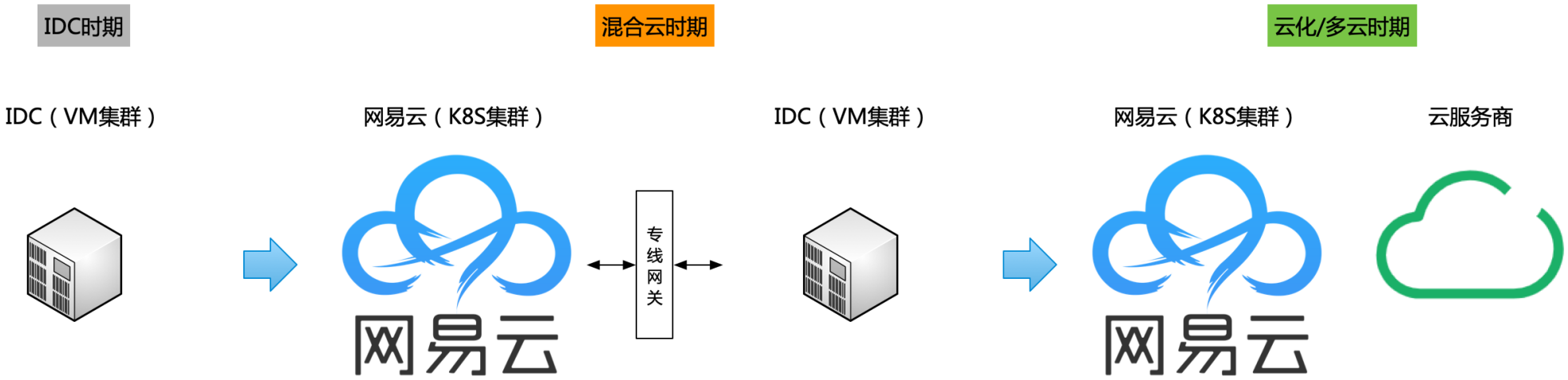
- 数据面以 Envoy Proxy 作为代理组件
- 控制面以 Pilot 为核心组件
- 平台开放与扩展主要通过 Kubernetes CRD与Mesh Configuration Protocol（简称为 MCP，一套标准 GRPC 协议）
- 高可用设计主要基于 Kubernetes 及 Istio 机制实现



/02

Service Mesh 在混合云架构落地

严选上云 Roadmap



落地关键步骤

CLOUD NATIVE COMPUTING FOUNDATION　ServiceMesher

拥抱云原生

- 大势所趋
- 容器化是**微服务的最佳载体**
- 容器化是 **Service Mesh 高效落地的基石**

建设服务治理平台

- **无缝衔接** VM 集群和容器集群的服务治理能力
- **最大化**发挥 Service Mesh 的优势

部署平台

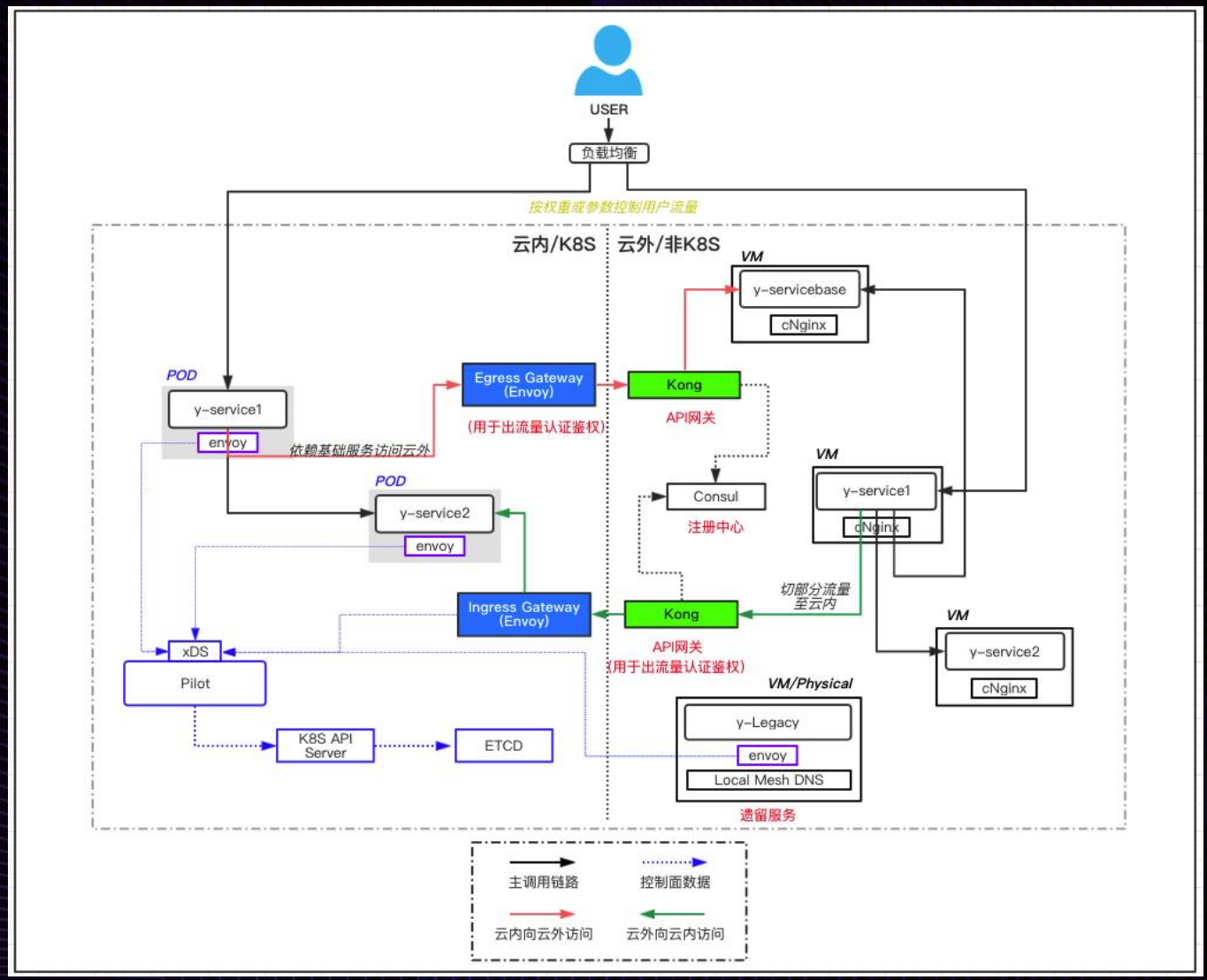
- **Sidecar 注入，业务无感知**
- **加速云化**

灰度引流

- 服务间调用灰度引流
- 外域调用灰度引流
- **平滑迁移是 Service Mesh 在混合云架构落地的关键**

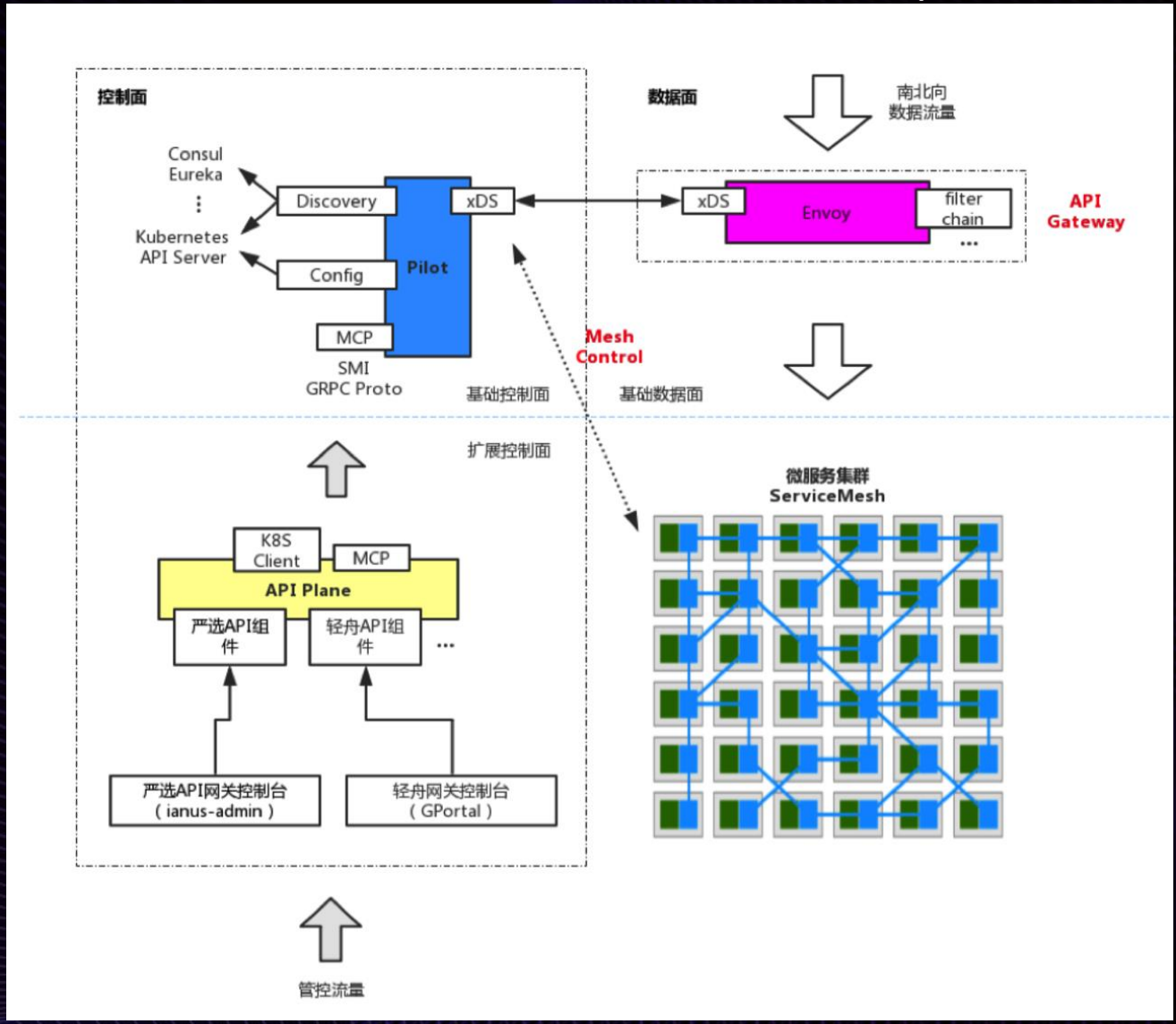
平滑迁移

- 边缘网关支持跨 LDC 访问
- 兜底路由：云内云外互备
- 访问控制（如白名单能力）
- 灰度引流使架构透明化
  - 服务间调用灰度引流
  - 外域调用灰度引流



API 网关

- 整体基于 **Envoy+Pilot** 方案：
- 数据面以 Envoy Proxy 作为代理组件
- 控制面以 Pilot 为核心组件
- 平台开放与扩展主要通过 Kubernetes CRD与Mesh Configuration Protocol（简称为 MCP，一套标准 GRPC 协议）



质量保障体系

- CICD
- 单元测试
- 性能基准自动测试
- 监控报警
- 版本升级机制
  - Envoy 热更新机制
  - 灰度发布机制：业务灰度+流量灰度
  - 演练测试
- 业务回归验证

一些坑

- Envoy 目前编译版本存在 Bug
  - 在 Istio pilot 升级到加入 accesslog 相关配置下发功能版本后，Envoy 在一定压力访问或有客户端主动断开请求时，会进入一段存在问题的断言（assert）逻辑，导致 envoy crash，此时请求方体现为 502 异常
  - 社区目前给出的优化建议是在 envoy 编译选项使用 -opt（默认为 -dbg）
  - 社区已在新版本清理这段问题断言逻辑：https://github.com/envoyproxy/envoy/issues/9083
- Mixer 性能陷阱
  - 如打开 Mixer 的策略执行功能，每一次调用 Envoy 都会同步调用 Mixer 进行一次策略检查，导致性能下降的非常迅速
  - 社区也已经意识到并着手进行优化
  - 作为 Mixer 策略执行的替代品， Istio 的 RBAC 也是可以满足一部分功能的，比如服务白名单我们就是通过 RBAC 来实现

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

/04

规划与展望

性能优化方向

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

- 方案1: 采用 eBPF/xDP(sockops)，优化路径为 SVC <-> Envoy，延迟性能提升10-20%。Envoy 部署方式 per-pod，跟社区方向一致，也是目前严选采用的部署方案。
- 方案2: 采用 DPDK+Fstack 用户态协议栈，优化路径为 Envoy <-> Envoy，延迟性能提升 0.8-1 倍。Envoy 部署方式为 per-node，功能和运维层面的限制还在评估当中。

- Sidecar 模式采用方案1进行优化，gateway 模式采用方案2进行优化。

服务治理平台 – 升级严选服务治理能力

CLOUD NATIVE COMPUTING FOUNDATION ServiceMesher

01.服务定义

- 服务元数据定义：服务等级、服务集群、服务器规格及环境
- 服务地图：可视化的服务关系，如业务拓扑、服务依赖拓扑，集群视图

02.服务管控

- 常用服务管理功能：服务上下线、服务实例管理
- 服务生命周期管控与查询
- 服务扩缩容：服务副本数、配额、扩缩容策略；调整后自动应用至 K8s 集群

03.调用控制

- 限流、资源隔离、熔断、降级等配置
- 负载均衡：流量调配、分流、切流量等
- 服务路由
- 访问控制配置

04.服务监控

- 服务监控项设置并对接基础监控平台
- 服务质量指标（SLI）定制并监控：如 Latency、QPS 等

05.配置管理

- 不可变信息配置
- 动态配置、实时下发
- 支持环境、集群等区分维度

06.问题定位与诊断

- 借助APM的能力
- 发现异常调用链
- 分析请求来源及去向

Service Mesh Meetup

CLOUD NATIVE COMPUTING FOUNDATION

ServiceMesher

感谢聆听

严选技术团队

PTC Product Technology Center

科技赋能制造，共创美好生活

扫描二维码关注