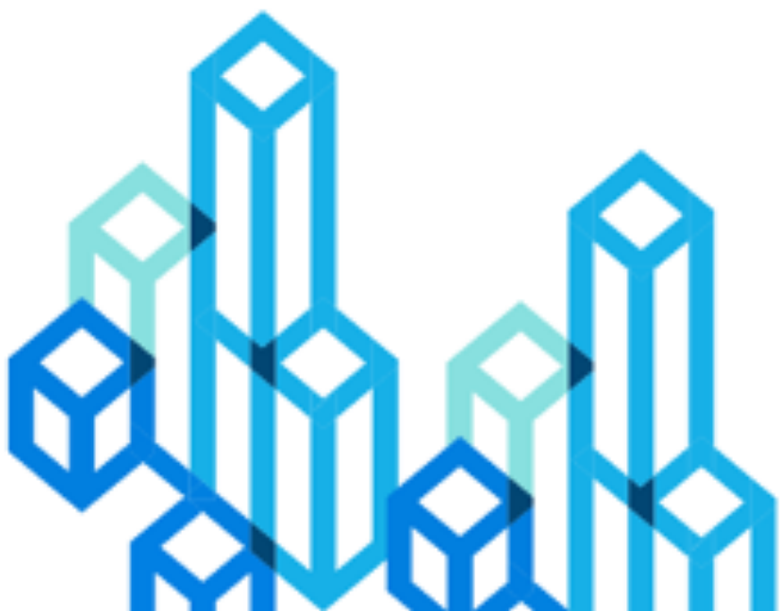
快手中间件Mesh化实践

姜涛



个人简介

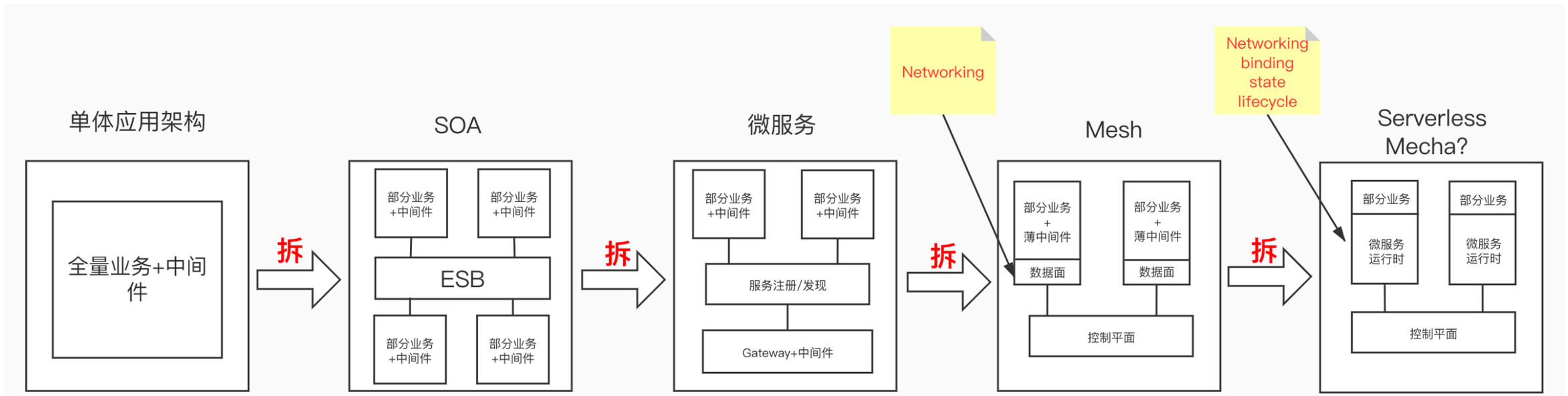
- 快手服务网格负责人
- 曾就职于阿里巴巴、美团、小红书
- Envoy、Istio、Dapr Contributor

目录

Part 01

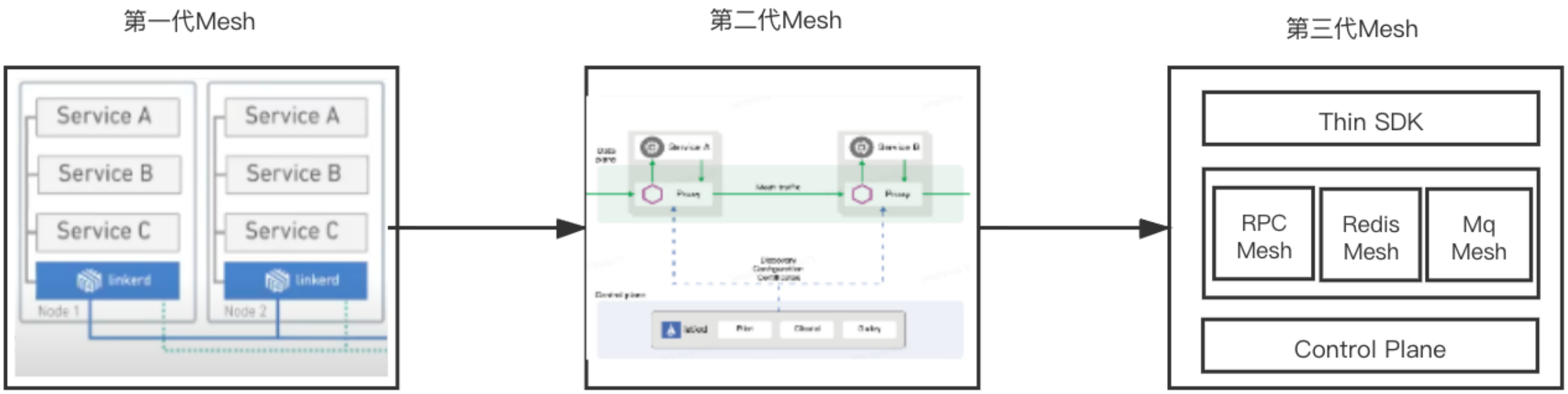
发展回顾

发展回顾



Mesh主要解决Networking的问题。
Mecha除了解决Networking的问题，还解决binding、state、lifecycle。同时配合轻量化SDK。例如：dapr

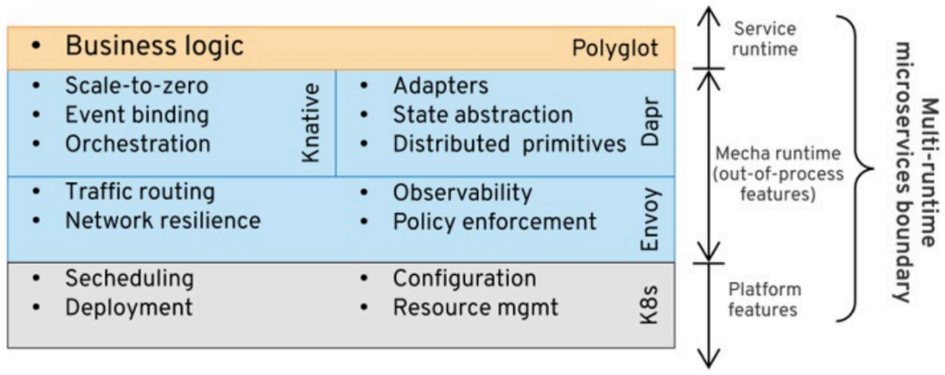
发展回顾



第一代mesh：以Linkerd为代表，主要是将rpc sdk（finagle）能力抽离到sidecar（Linkerd）中，实现解耦。

第二代mesh：以Istio、Linkerd2为代表，分离了控制面与数据面。主要处理rpc协议，强调完全无侵入、透明劫持与代理。

我们的mesh：强调应用基础设施能力下沉，统一解决Networking与Event binding(处于Istio与Multi-Runtime架构之间，我们称之为第三代mesh)。无侵入方案是关键手段但并非终态（不需要完全透明流量劫持与去sdk化）



关于网格的核心认知

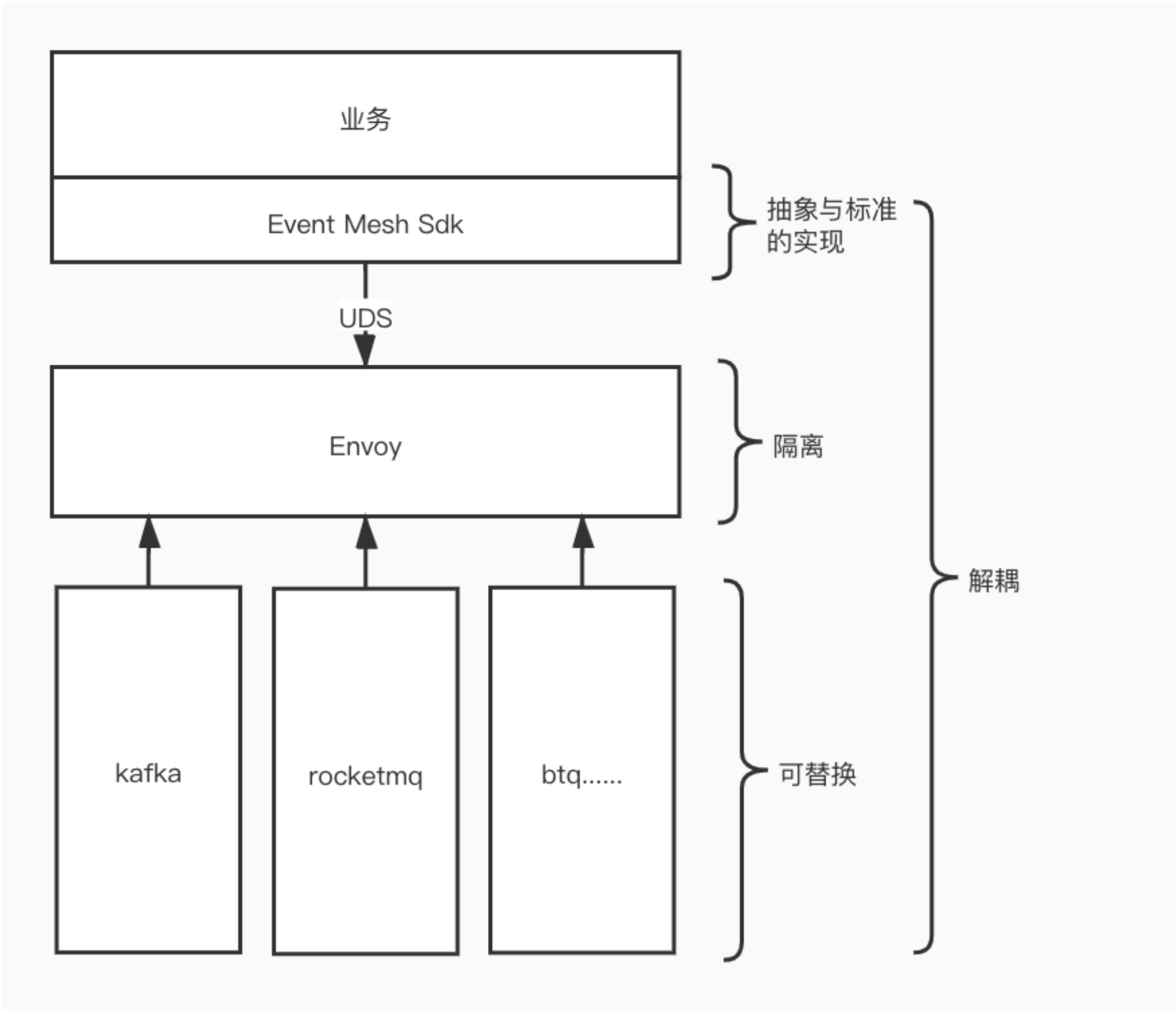
- 网格的核心价值：解耦
  - 解耦的价值：控制复杂度，将复杂的事务拆分成若干独立、简单的事务；应对不确定性；
- 解决中间件“重SDK”模式的痛点
  - 开发维护成本高
  - 新版本升级覆盖难
  - Bug召回给业务方的负担大
  - 业务方在架构改造项目上投入大

Part 02

选型思考与架构

选型思考

- 是否还需要SDK？需要，thin SDK
- Mesh的定位：承担中间件SDK能力下沉，代理业务常用通信协议，形成低耦合、标准化、成体系且稳定的通信基础设施。
- Thin SDK定位
  - 标准化
  - 抽象化
  - 易用性



选型思考

01 优先遵循业内标准，比如xDS协议，如不满足则对其扩展

02 由于服务实例数庞大且调用关系无法预知与规划，使用上需要改进，适配快手超大规模与复杂现状

03 复用现有设施，尤其是经过大规模验证得设施，不重复造轮子

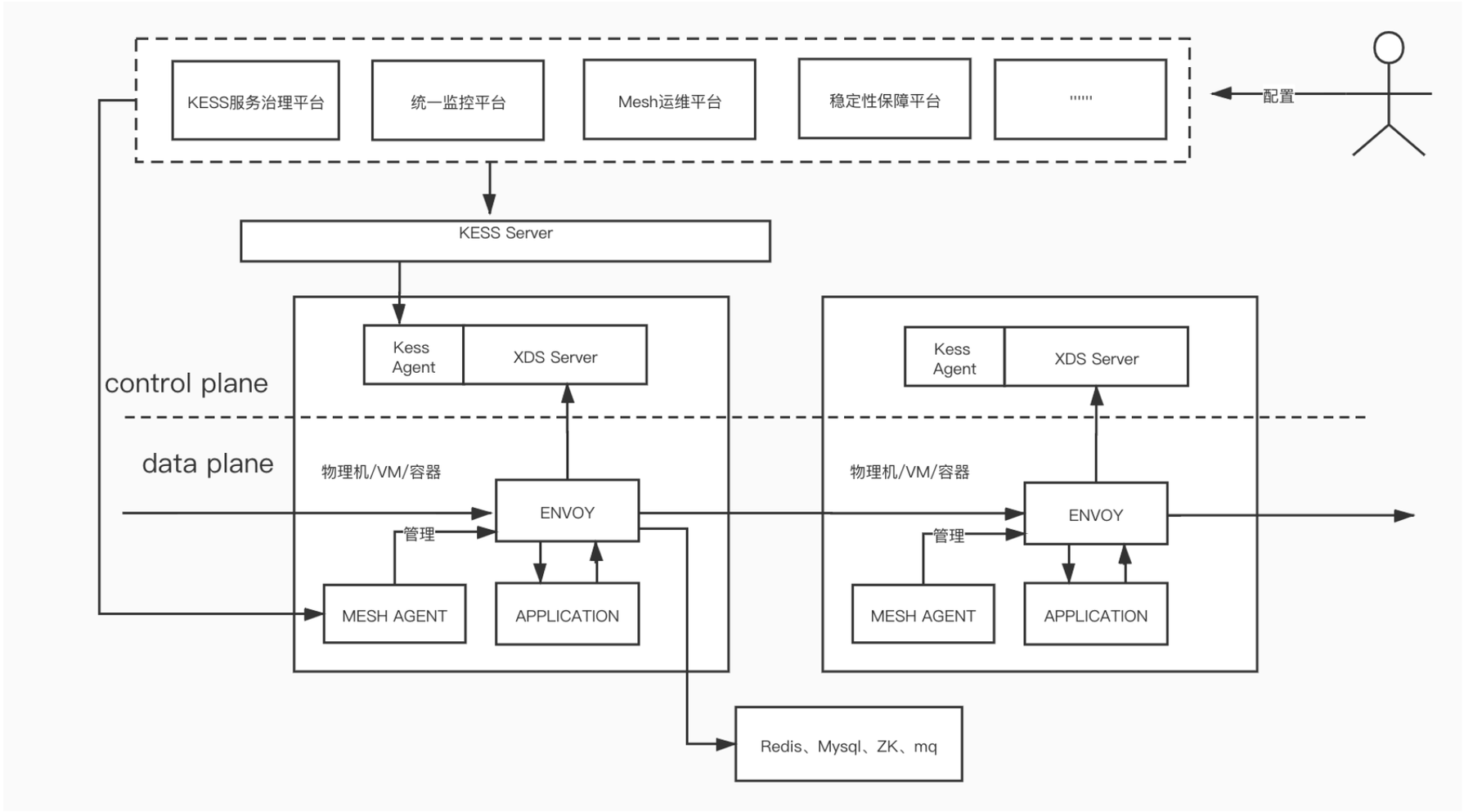
04 需要同时支持物理机、容器（Host、Overlay网络），方便大规模落地，且运维成本可控

05 中间件SDK Mesh化对资源消耗、性能更敏感，需要废弃掉社区影响性能的设计（Iptables流量劫持、Mixer等）

整体架构

- 主要特点

1. 物理机、VM、容器统一运维管理
2. 分布式 xDS server
3. delta、on-demand资源获取方式
4. 支持Http、Grpc、Redis、ZK、Mysql、Kafka等多种协议
5. 复用已有基础设施与平台、联动打通



Part 03

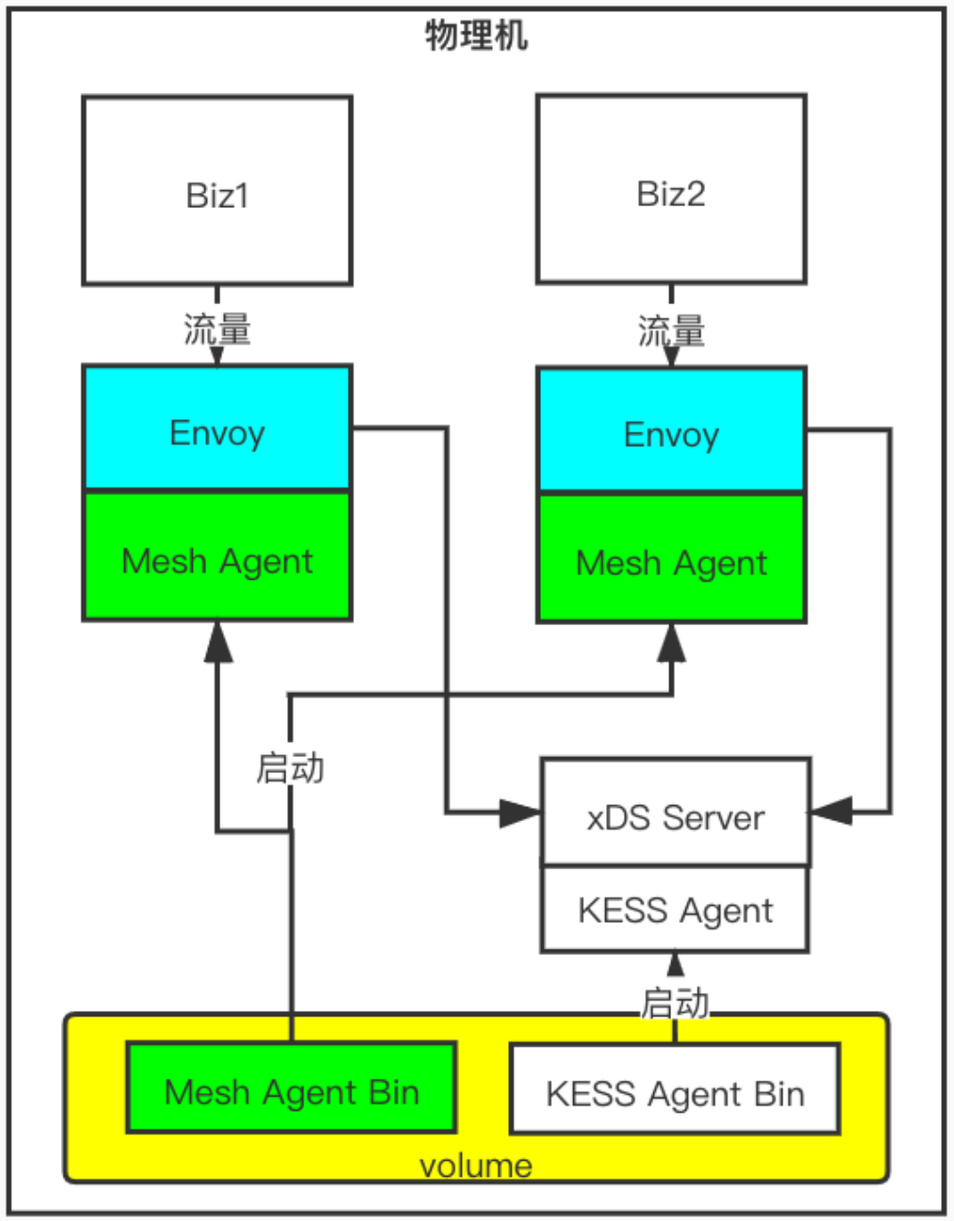
挑战与解法

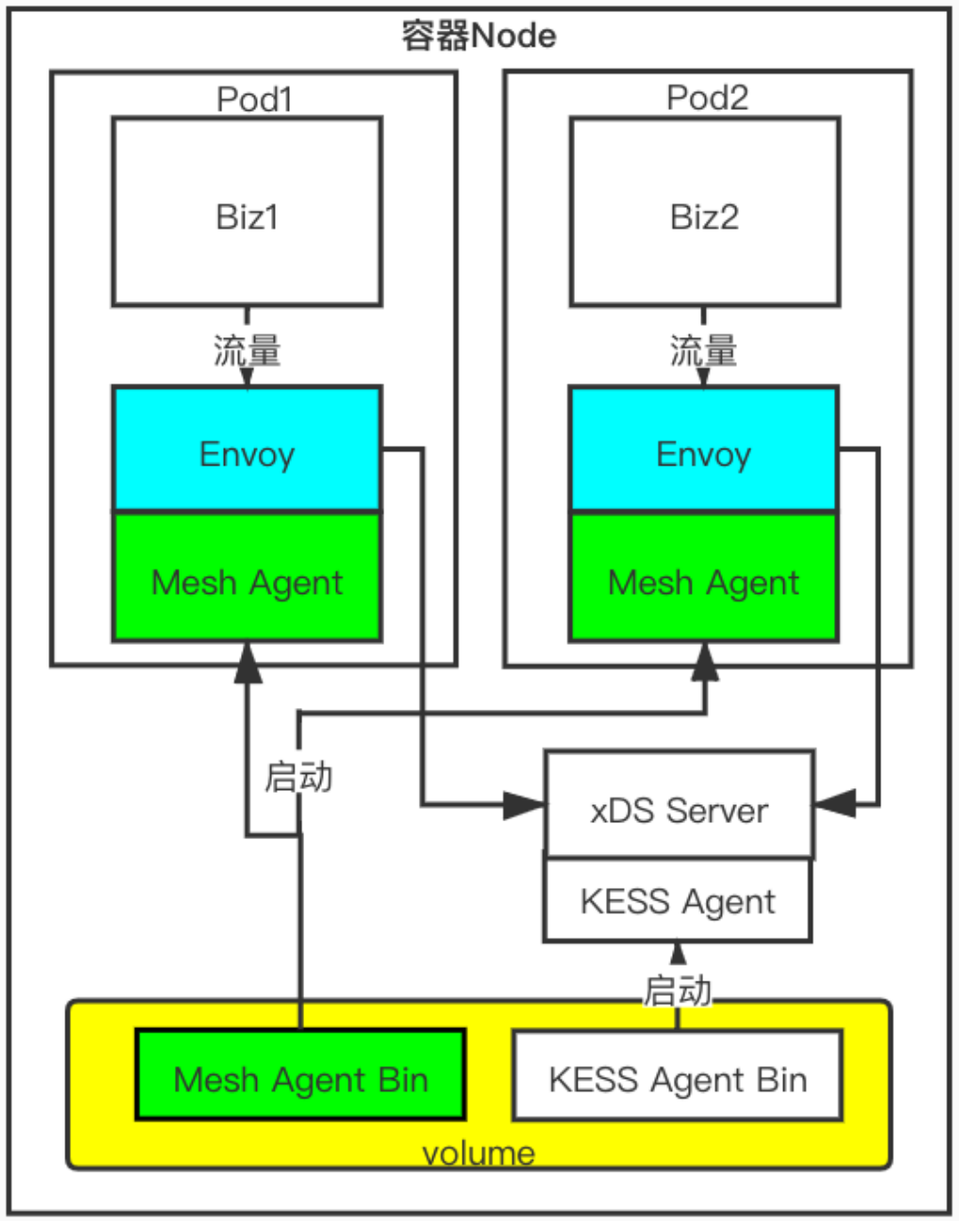
主要挑战

- 成本问题：复杂环境下的统一部署与运维
- 复杂度问题：规模大、性能要求高、策略复杂
- 落地推广：对业务来说不是强需求

统一部署

物理机、容器统一部署





统一运维

- 新建
  - 服务网格运维平台：部署、配置变更、热重启、内部监控。支持多sidecar。
- 接入
  - 服务发现与治理：服务发现信息、服务治理策略
  - 统一可观测性平台：提供Mesh层的监控
  - 稳定性保障平台：故障注入、流量录制等
  - 鉴权平台：访问控制等

挑战2：规模大、性能要求高、策略复杂

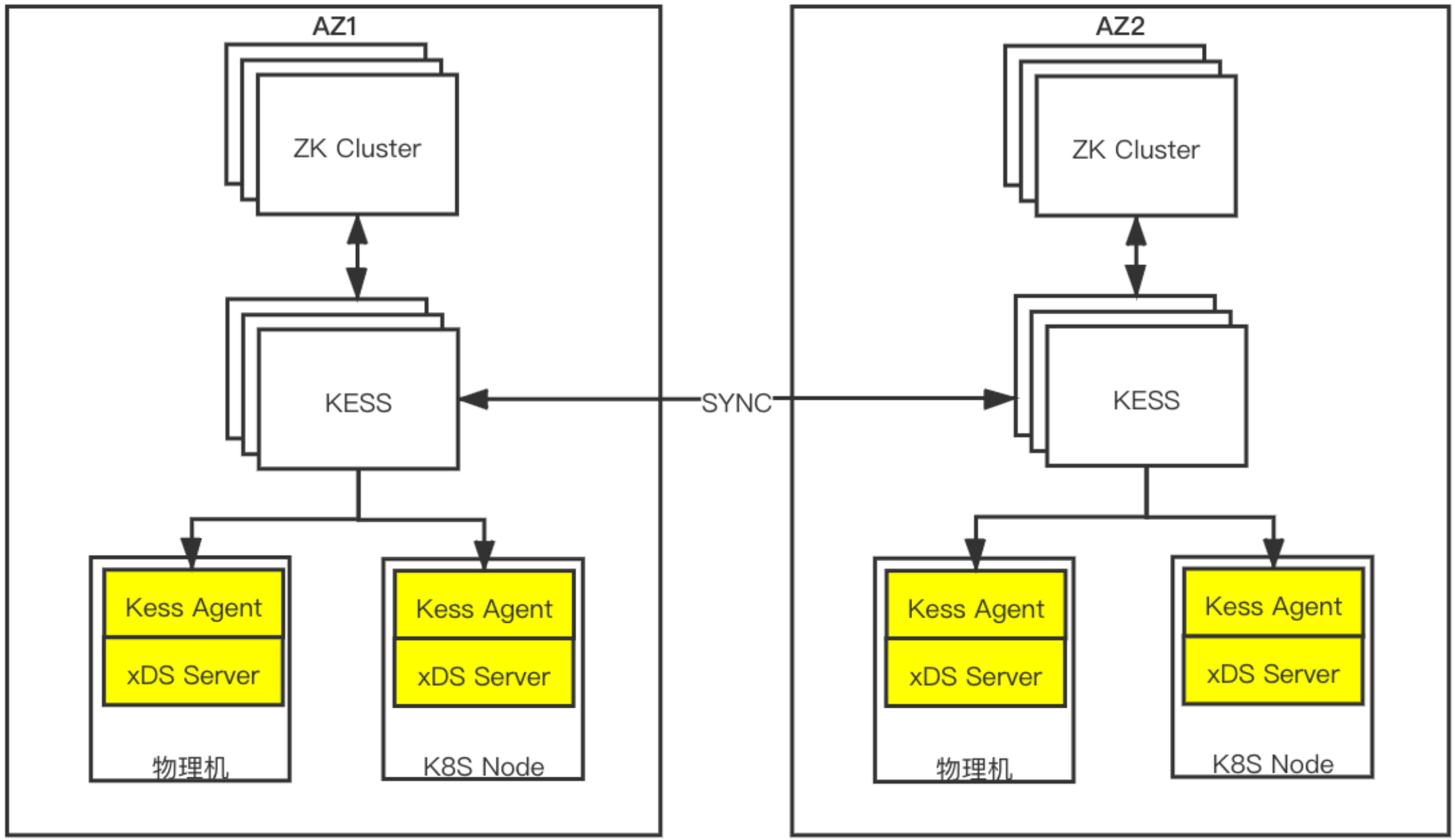
- 规模大

01 负载分散

02 数据切分

规模大问题

- 负载分散：分布式控制面



规模大问题

数据切分

Delta xDS、On-Demand xDS

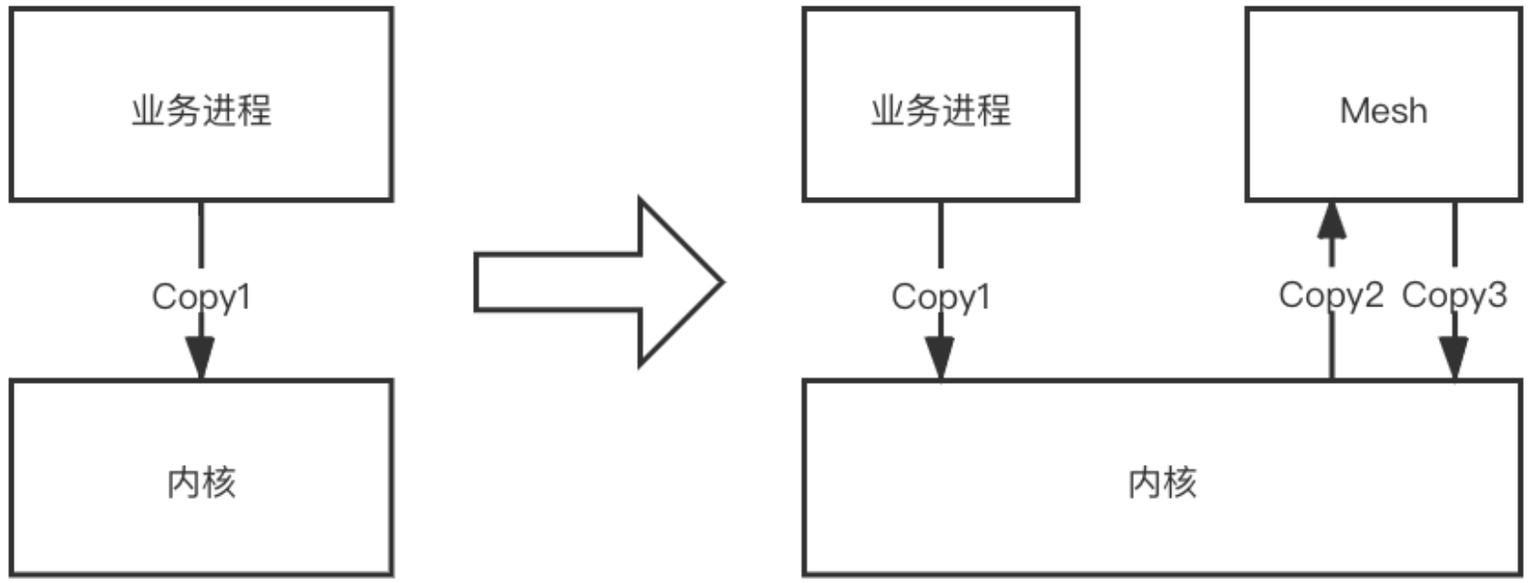
Delta xDS

只传输变更的增量信息

On-Demand xDS

本质是按需获取。主要优化CDS，初始化CDS通过wildcard watch管理，可根据请求动态触发CDS请求，是CDS也支持通过named watch管理

性能要求高



- Copy1、Copy2优化：基于”共享内存无锁环形队列“的IPC
- Copy3优化： MSG_ZEROCOPY

性能要求高

已经做的事

- Mget、Mset拆分优化
- 连接数优化
- 可观测性精简
- SDK协程化接口（stackful、sdk改造）

正在做的事

- 协议精简与标准化（部分完成）
- 无拷贝序列化（类似Capnproto）
- IO_URING探索
- 探索Envoy之间使用Quic，降低跨机房通信延迟

策略复杂集成

需要集成三部分复杂策略：服务治理策略、Mesh与SDK联动性策略、容错性策略

服务治理策略：

	功能点	Envoy不足点	优化点
1	负载均衡	• 一致性hash的负载均衡性不优 • 没有连接数均衡的负载均衡算法	• 添加HRW Hash • 添加“最少连接“负载均衡策略
2	治理策略	• 没有隔板、随机丢弃等策略 • 不同的协议（Filter）分开实现，重复开发且行为不一致，控制复杂（分散在CDS、RDS、RTDS）	• 添加隔板、随机丢弃治理策略，与kess对齐 • 统一实现，通过增强rtds统一控制
3	熔断	• 不区分协议 • 实现与Netflix熔断不一致，没有熔断打开后暂定请求一段时间	• 按协议进行区分 • 复用KESS进行重新实现
4	自适应限流	• 只支持HTTP协议，其他协议无法复用 • 应用在Cluster级别（服务级别） • 策略对象是Thread Local的，性能上有优势	• 支持各种协议 • 应用到Method级别 • 策略对象优化为全局，控制更准确
5	限流	• 只细化到path力度 • 一个限流配置却需要通过RDS + RTDS相互配合下发，过于复杂	• 细化到方法级别 • 通过增强RTDS统一控制

策略复杂集成

- 联动性策略
  - SDK的参数设置与Mesh的联动与打通，并通过专有Header格式来处理
  - 可观测性的联动性
  - 热升级SDK的联动

策略复杂集成

- 可靠性策略：采用面向失败的架构设计（Failures can and will happen, anytime, anywhere. Fail fast, fail small, fail often and recover quickly）
  - xDS Server分布式化，控制Bug影响范围。
  - SDK 对主动/被动健康检查。
    - 主动检测Envoy admin是否健康
    - 被动检测请求response是否有envoy内部错误。
  - SDK自动fallback。
    - Fat SDK自动fallback，切换直连模式。
    - Thin SDK自动fallback，切换为Proxy模式。
- SDK Traffic Shift流量平滑迁移

挑战3：对业务不是强需求

- 业务对性能、尤其稳定性比较敏感。
- 业务很难投入人力配合架构升级。
- 从Mesh能力上看，给业务带来的收益并不明显。

挑战4：落地推广实践

- 稳定性是重中之重，铺量与放量解耦
- 选取典型落地场景
- 基于WASM扩展性，与业务共建
- 搭车公司重大项目（单元化、机房逃生）

Part 04

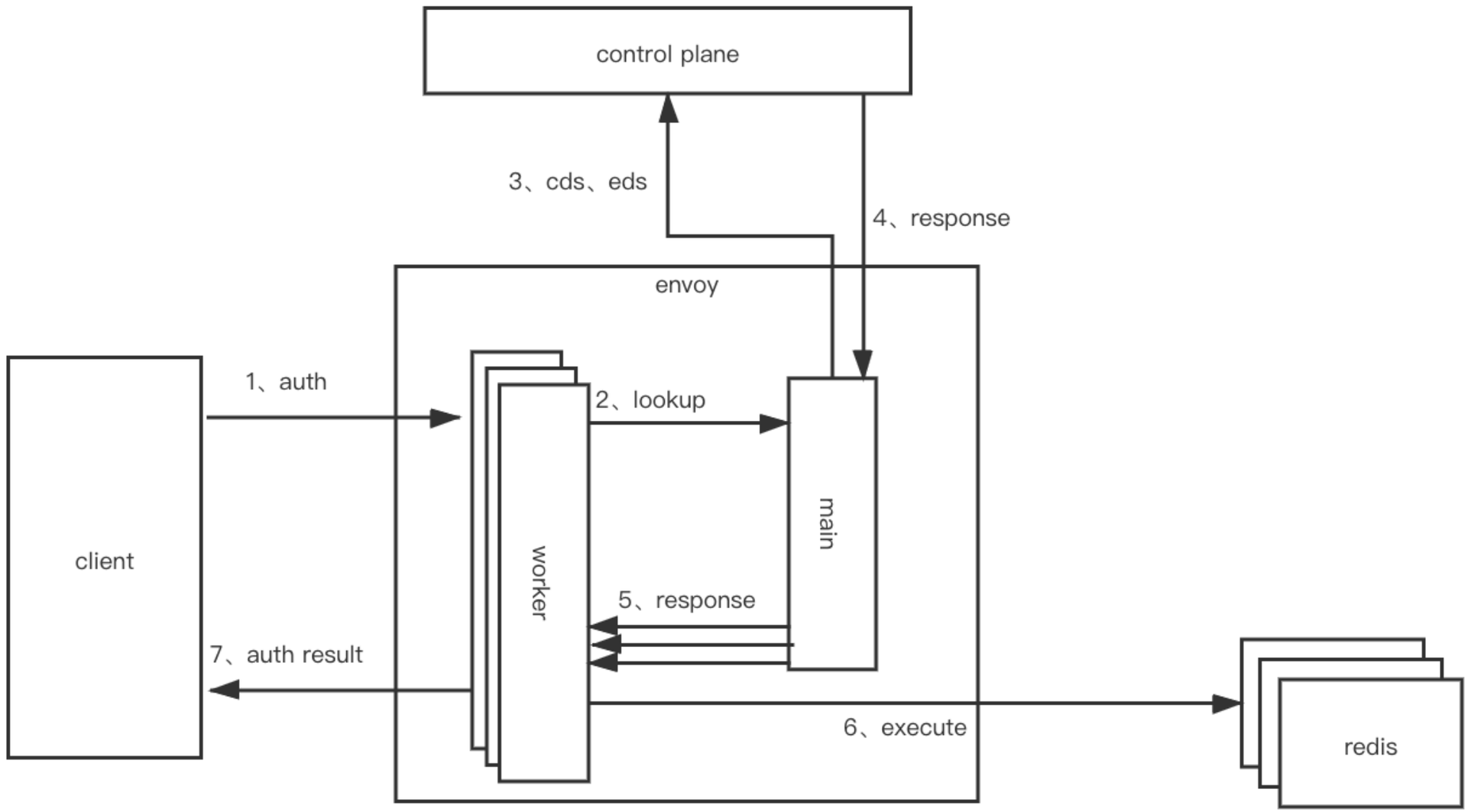
落地情况与展望

落地情况

- 规模情况：落地实例数数万
- 延迟情况：延迟增加60~140微秒，p90 100~180微秒
- 资源消耗情况：Redis单核跑满10.2万qps；Rpc单核跑满

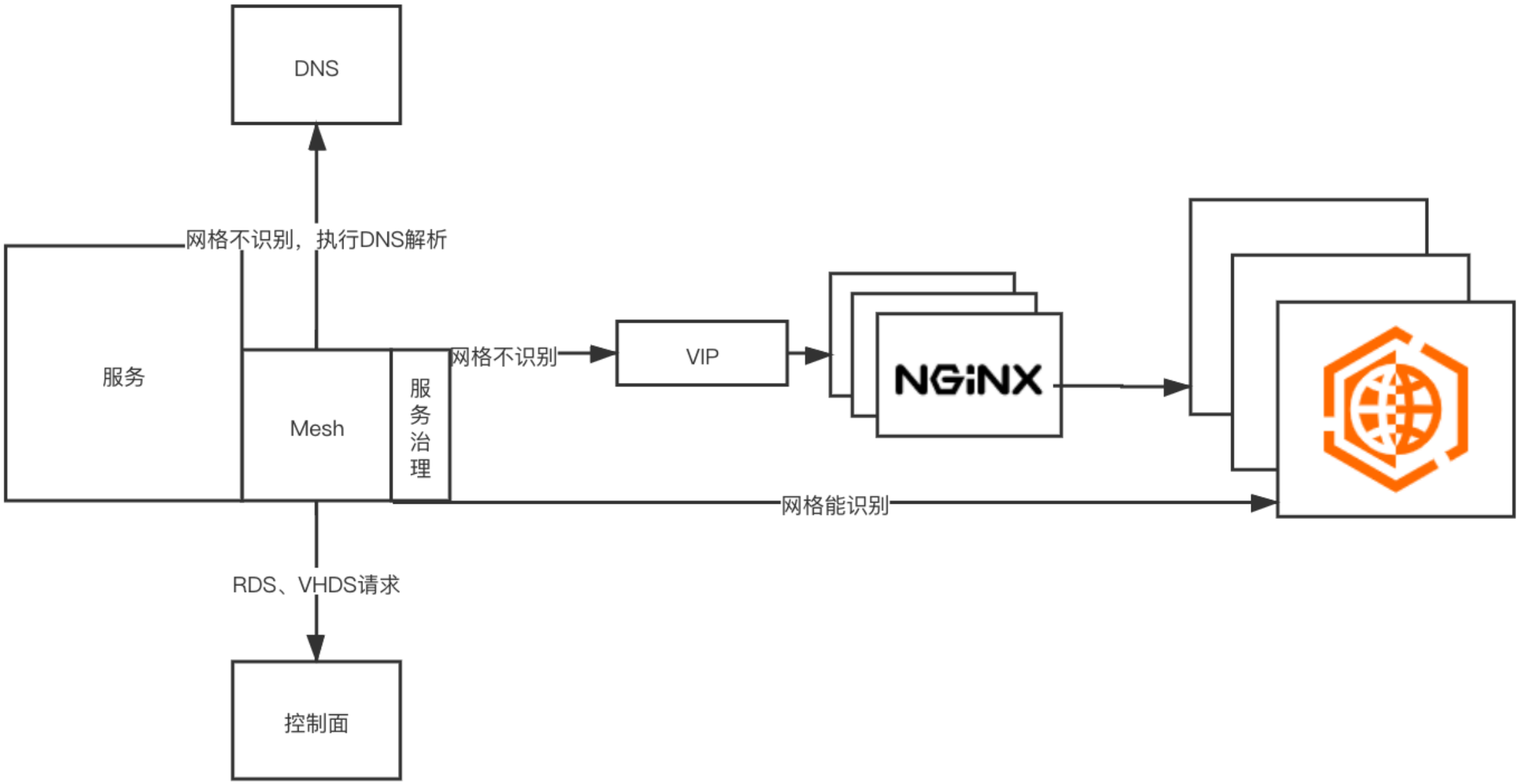
落地案例：Redis

- 通过Auth来发起on-demand CDS请求



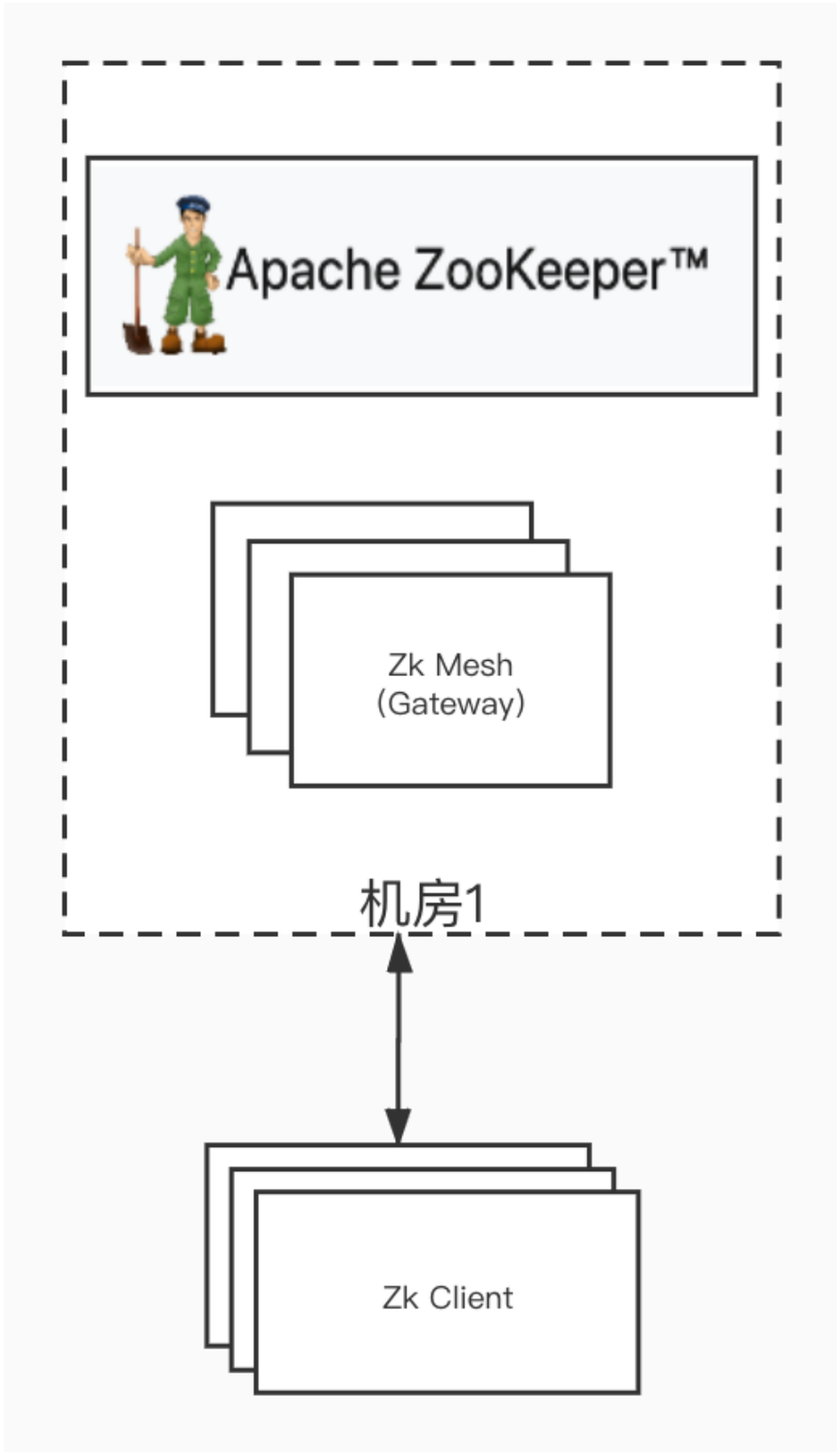
落地案例：HTTP

通过VHDS、on-demand CDS获取配置，且自动fallback回DNS



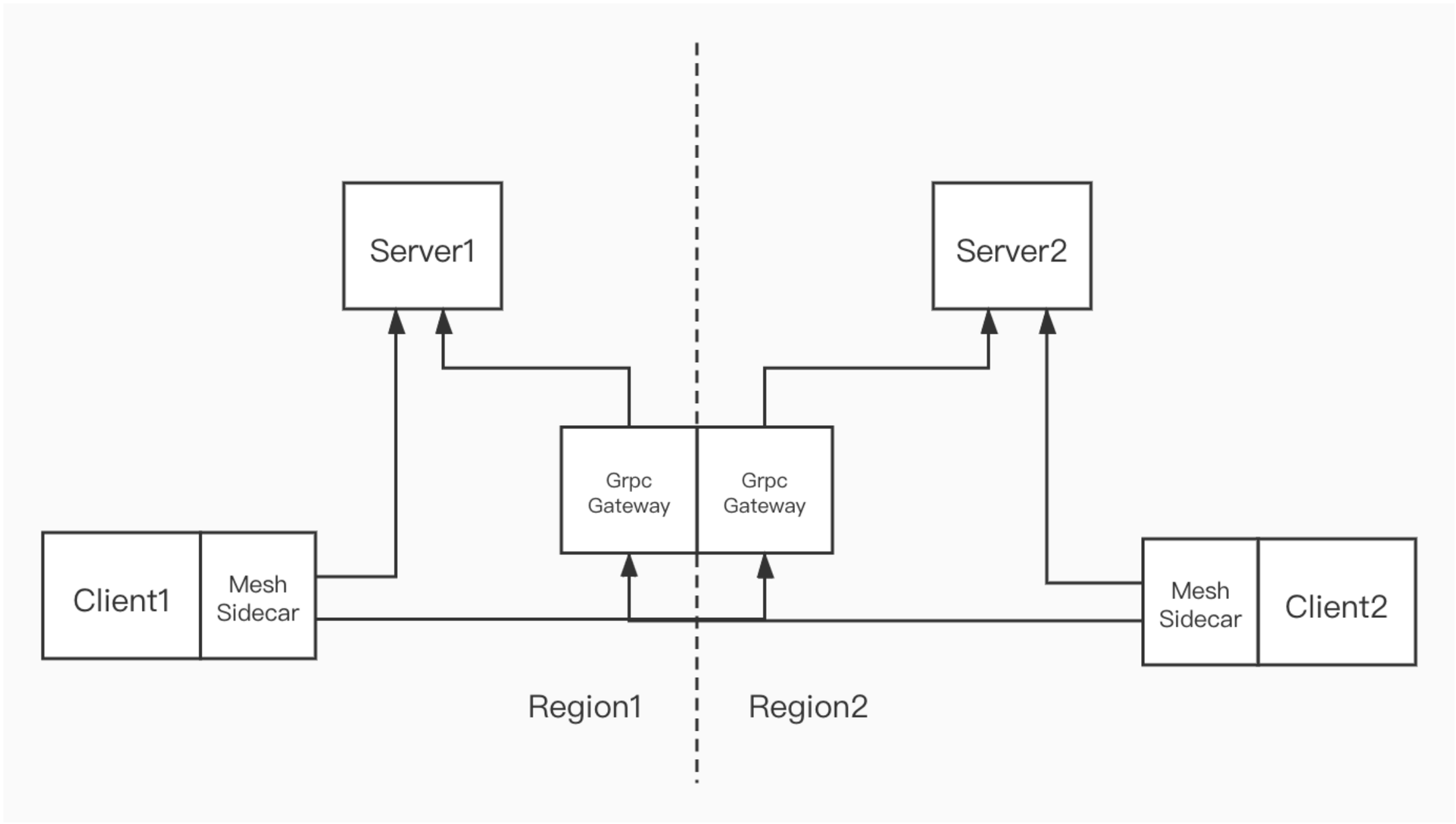
落地案例：Zookeeper

对ZooKeeper集群提
供读写限流保护（尤其
是写保护），以及可观测性。



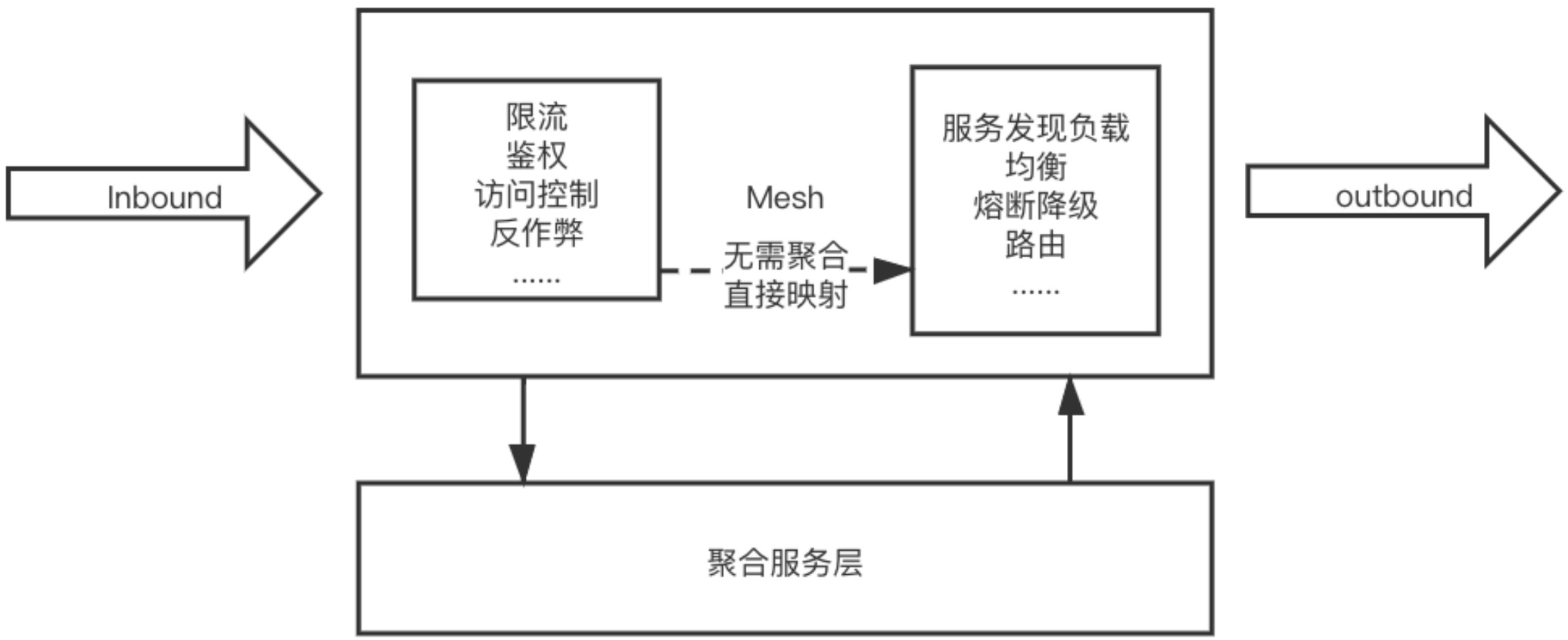
落地案例：Grpc

基于On-demand CDS，提供Mesh Sidecar、Mesh Gateway两种形态，其中MeshGateway用于跨Region流量管控。



落地案例：ApiGateway

去中心化ApiGateway，与聚合服务一起部署。



未来规划

- 继续完善产品能力矩阵（2020年8月~2021年12月）

Service Invocation

HTTP、Grpc

01

Event

Kafka、RocketMq、Btq

02

Storage

Redis、Mysql、ZK、Mongo

03

未来规划

生态建设

品牌建设

核心能力建设与落地

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。

高可用架构主要关注互联网架构及高可用、可扩展及高性能领域的知识传播。订阅用户覆盖主流互联网及软件领域系统架构技术从业人员。 高可用架构系列社群是一个社区组织，其精神是“分享+交流”，提倡社区的人人参与，同时从社区获得高质量的内容。