ServiceMesher

IT大咖说 知识共享平台

大规模微服务架构下的

Service Mesh探索之路

敖小剑

蚂蚁金服 ANT FINANCIAL

前言

6月初在深圳举行的GIAC全球互联网架构大会上，蚂蚁金服第一次对外透露了开发中的Service Mesh产品——**Sofa Mesh**。

今天我们将展开更多细节，详细介绍蚂蚁金服Sofa Mesh的**技术选型**，**架构设计**以及**开源策略**。

1

Technical

技术选型

Service Mesh落地要面临的实际要求

- ✓ 性能要求
  - 以蚂蚁金服的体量，性能不够好则难于接受
  - 架构与性能之间的权衡和取舍需要谨慎考虑
- ✓ 稳定性要求
  - 以蚂蚁金服的标准，稳定性的要求自然是很高
  - 高可用方面的要求很非常高
- ✓ 部署的要求
  - 需要用于多种场合：主站，金融云，外部客户
  - 需要满足多种部署环境：虚拟机/容器，公有云/私有云，k8s
  - 需要满足多种体系：Service Mesh，Sofa和社区主流开发框架

选择开源产品，还是选择自研？

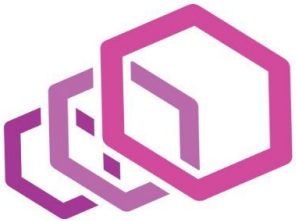
ServiceMesher

IT大咖说 知识共享平台

开源方案选择之第一代Service Mesh

Linkerd

- 无控制平面
- Scala编写，基于JVM资源消耗大
- 可扩展性有限，dtab不易理解和使用
- 功能不能满足蚂蚁的需求，没法做到类似envoy xds那样的扩展性
- 未来发展前景黯淡



Envoy

- 安心做数据平面， 提供XDS API
- 设计优秀，性能和稳定性表现良好
- C++编写，和蚂蚁的技术栈差异大
- 蚂蚁有大量的扩展和定制化需求
- 我们非常认可envoy在数据平面上的表现

开源方案选择之第二代Service Mesh

Istio

- 第一选择，重点关注对象
- 奈何迟迟不能发布生产可用版本
- 性能和稳定性远远不能满足蚂蚁的要求

- 但我们非常认可Istio的理念和方向

Conduit

- 只支持k8s，而蚂蚁尚未普及k8s
- 数据平面由Rust编写，过于小众，难于从社区借力。
- 同样存在技术栈问题
- 公司和产品在社区知名度和影响力有限

国内公司的选择之一：自研

华为：CES Mesher

- 使用Golang编写
- 由go chassis演进而来
- 走的是已有类库->加proxy->再加控制平面的路线
- 部分对接Istio
- 细节暂时不清楚，即将开源

新浪微博：Motan Mesh

- 也是使用Golang编写
- 全新实现（原有类库是基于Java）

老成持重的稳健思路：以proxy为切入口，第一时间获取跨语言和技术栈下沉的红利，立足之后再缓缓图之。

这个产品思路唯一的麻烦在于编程语言的选择

国内公司的选择之二：开源方案定制

腾讯：Tencent Service Mesh

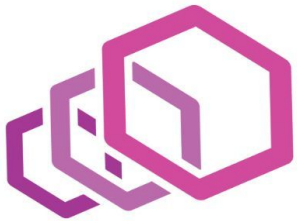
- 数据平面选择Envoy：成熟产品，符合腾讯语言体系，内部广泛使用
- 控制平面据传“挣扎了一下”，最终还是选择Istio，进行定制和扩展，解耦k8s

国内公司的选择之三：另辟蹊径

UCloud：Service Mesh

- 非常有意思的轻量ServiceMesh实践
- 从Istio中剥离Pilot和Envoy
- 去掉Mixer和Auth
- 定制Pilot，实现ETCD Adapter
- 脱离k8s运行

Sofa Mesh在技术选型时考虑



Envoy

- 数据平面：Envoy最符合要求
- XDS API的设计更是令人称道

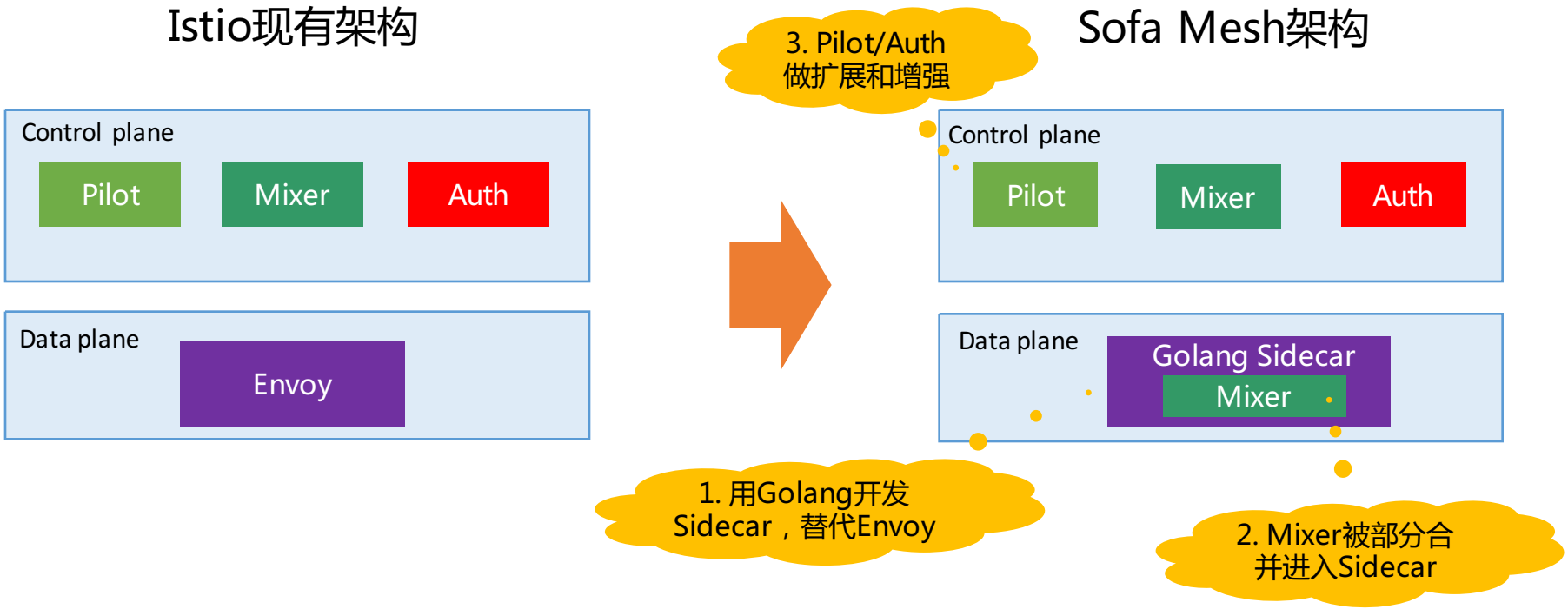
- C++带来的技术栈选择问题
- 我们有太多的扩展和定制
- 而且，proxy不仅仅用于mesh

Istio

- 控制平面：Istio是目前做的最好的
- 认可Istio的设计理念和产品方向

- 性能和稳定性是目前最大问题
- 对非k8s环境的支持不够理想
- 没有提供和侵入式框架互通的解决方案

Sofa Mesh：istio的增强扩展版



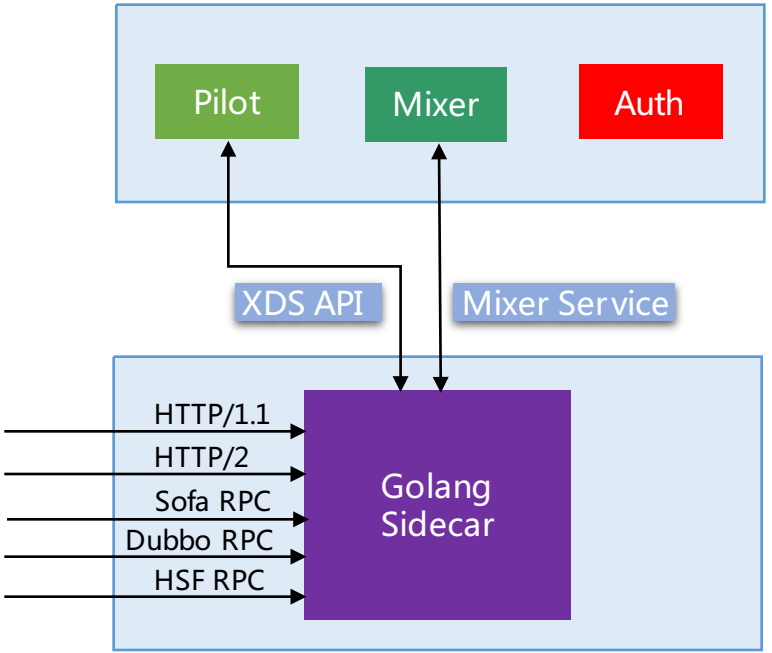
2

Architect

架构设计

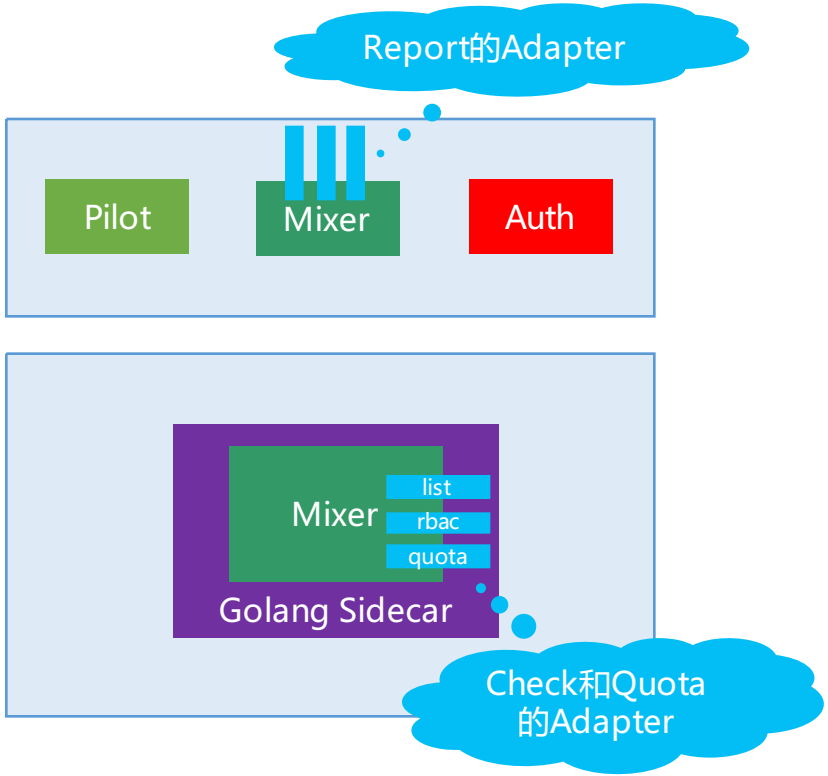
Golang版Sidecar

- ✓ 参照Envoy的设计
- ✓ 实现XDS API
- ✓ 兼容Istio
- ✓ 支持HTTP/1.1和HTTP/2
- ✓ 扩展Sofa/Dubbo/HSF



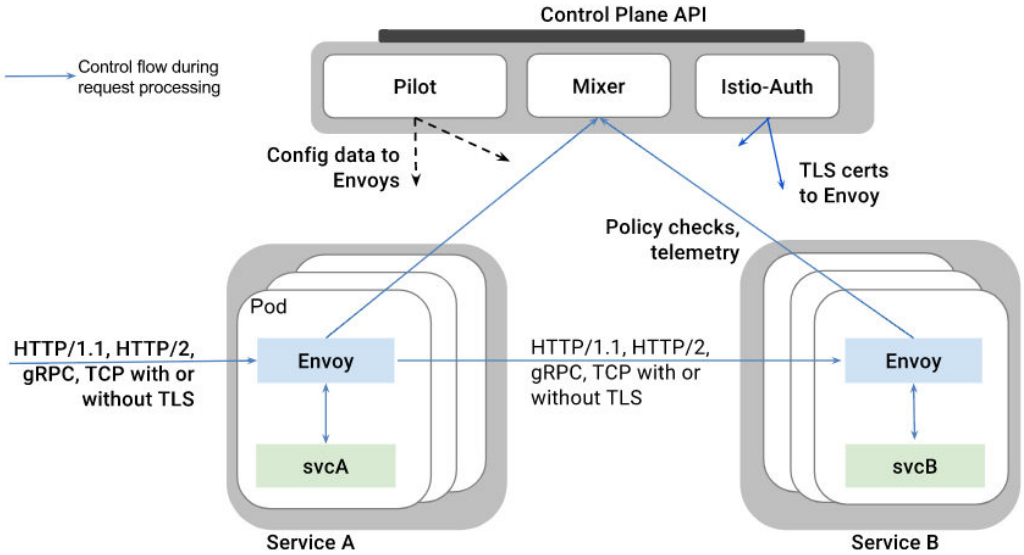
最大的改变：合并部分Mixer功能

- ✓ Mixer三大功能：
  - Check – 同步阻塞
  - Quota – 同步阻塞
  - Report – 异步批量
- ✓ 合并Check和Quota
- ✓ Report暂时保留在Mixer中



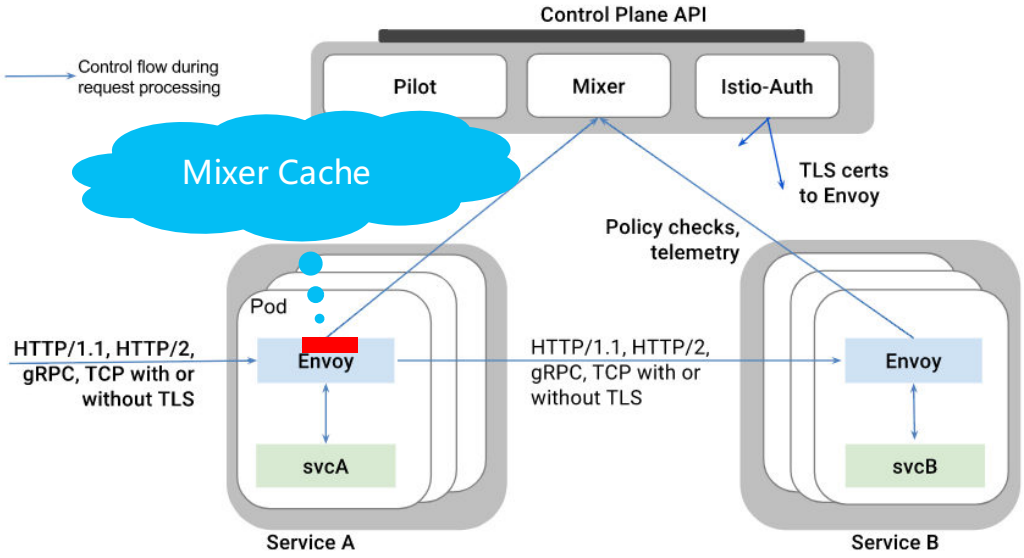
Mixer反省之一：对性能的影响

- ✓ 按照Istio的设计，每次请求Envoy都要执行对Mixer的两次远程调用：
  - 转发前执行Check(包含Quota)
  - 转发后执行Report
- ✓ 我们的观点：
  - 需要请求同步阻塞等待的功能都应该在Sidecar中完成
  - 远程调用带来的性能开销代价太高
  - 其他尽量优化为异步或者批量



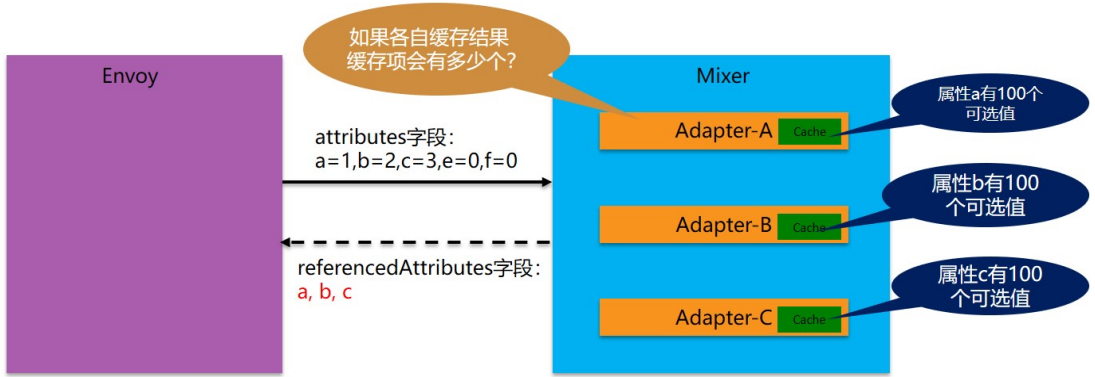
Istio的解决方案：添加Mixer Cache

- ✓ 缓存的工作方式：
  - Sidecar 中包含本地缓存，一部分的前置检查可以通过缓存来进行
  - 另外，Sidecar 会把待发送的Report数据进行缓冲，这样可能在多次请求之后才调用一次 Mixer
  - 前置检查和配额是同步的
  - Report数据上送是使用 fire-and-forget 模式异步完成的



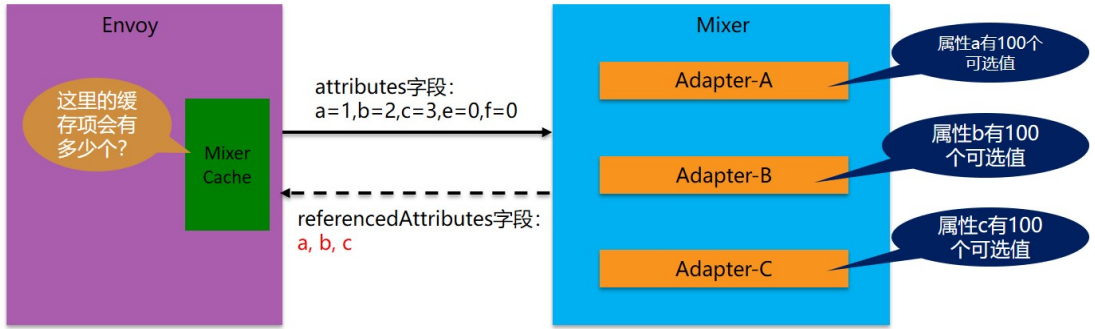
Mixer Cache的隐患

a + b + c = 300



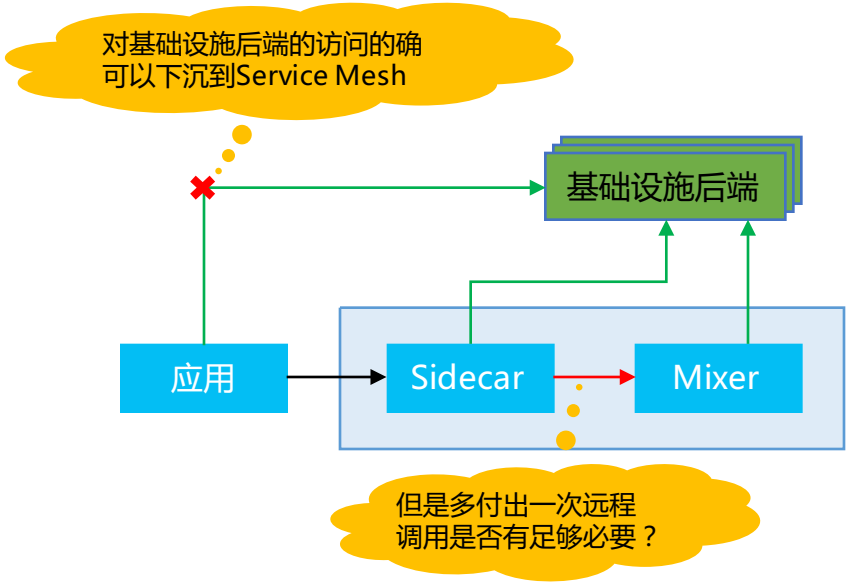
a * b * c = 1000000

笛卡尔乘积



Mixer反省之二：隔离和抽象的层次

- ✓ Mixer的设计目标：
  - 提供统一抽象(Adapter)
  - 隔离基础设施后端和Istio其他部分
  - 容许运维对所有交互进行精细控制合并Check和Quota
- ✓ 我们的反思
  - 认可这样的抽象和隔离，确实有必要从应用中剥离出来
  - 但是要加多一层Mixer，多一次远程调用
  - 抽象和隔离在Sidecar层面完成，也是可以达到效果的
  - 对于Check和Quota，性能损失太大，隔离的效果并不明显



探讨：何为基础设施后端？是否可以

ServiceMesher

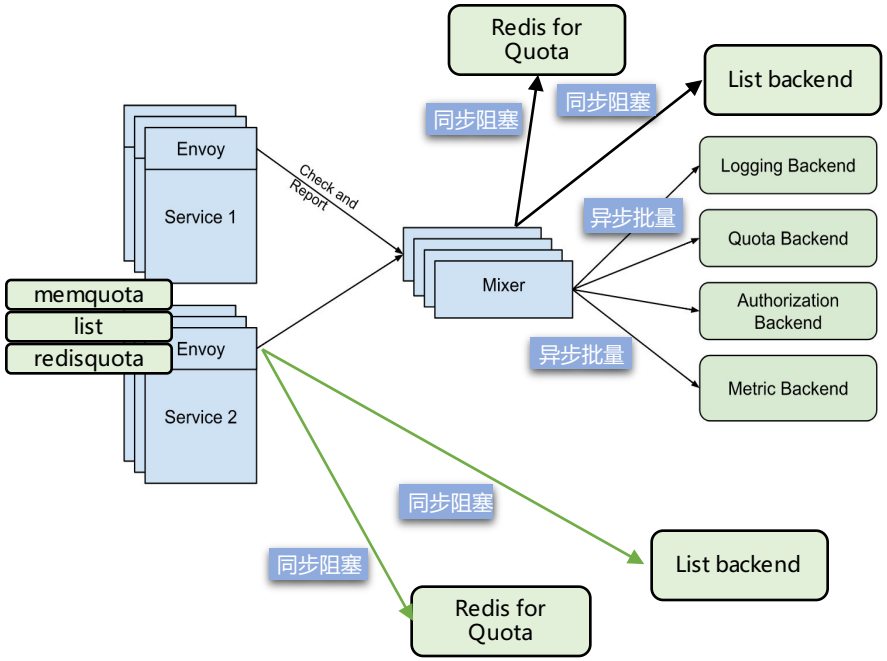
IT大咖说 知识共享平台

Istio现有的Mixer Adapter：

- ✓ 实现Check的Adapter：
  - listchecker (黑白名单)
  - opa (Open Policy Agent)
  - rbac（连接到Istio CA）
- ✓ 实现Quota的Adapter
  - Memquota (基于单机内存)
  - Redisquota （基于外部redis）
- ✓ 实现Report的Adapter
  - Circonus
  - Cloudwatch
  - Dogstatsd
  - Fluentd
  - Prometheus
  - Solarwinds
  - Stackdriver
  - Statsd
  - Stdio

但是这些？？更应该视为基本能力，直接做成Mesh内置功能

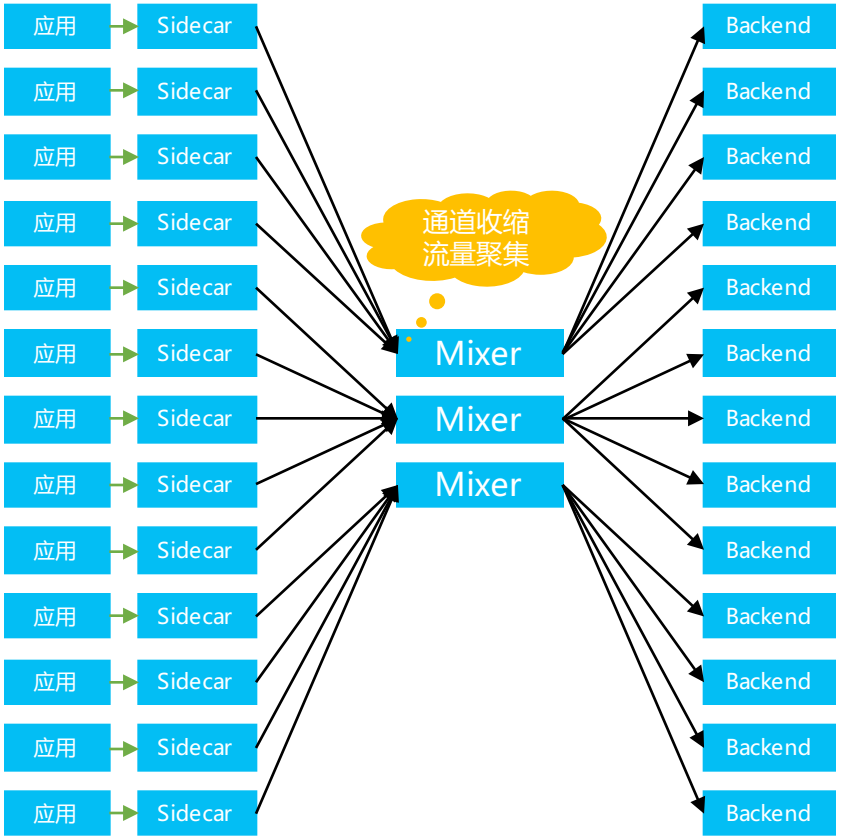
同意视为基础设置，甚至可能集成更多，这里的抽象隔离是我们认可的



更多资料，深入了解

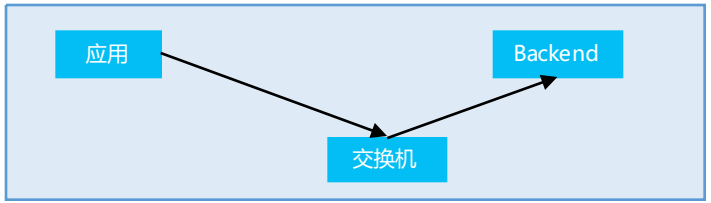
- ✓ 有关数据平面和控制平面的
  - Service Mesh架构反思：数据平面和控制平面的界线该如何划定？
- ✓ 有关Mixer Cache的详细介绍和源码解析
  - Mixer Cache: Istio的阿克琉斯之踵?
  - Istio Mixer Cache工作原理与源码分析(1) - 基本概念
  - Istio Mixer Cache工作原理与源码分析(2) - 工作原理
  - Istio Mixer Cache工作原理与源码分析(3) - 主流程
  - Istio Mixer Cache工作原理与源码分析(4) - 签名

Report部分的隐忧：网络集中

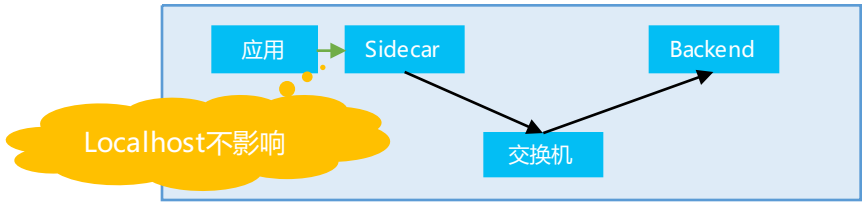


Report部分的隐忧：网络吞吐量

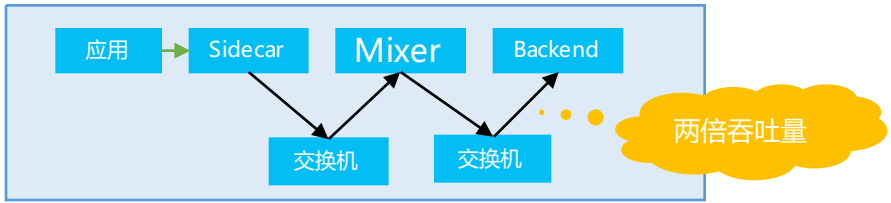
应用直连基础设施后端：



Sidecar连接基础设施后端：



Mixer连接基础设施后端：



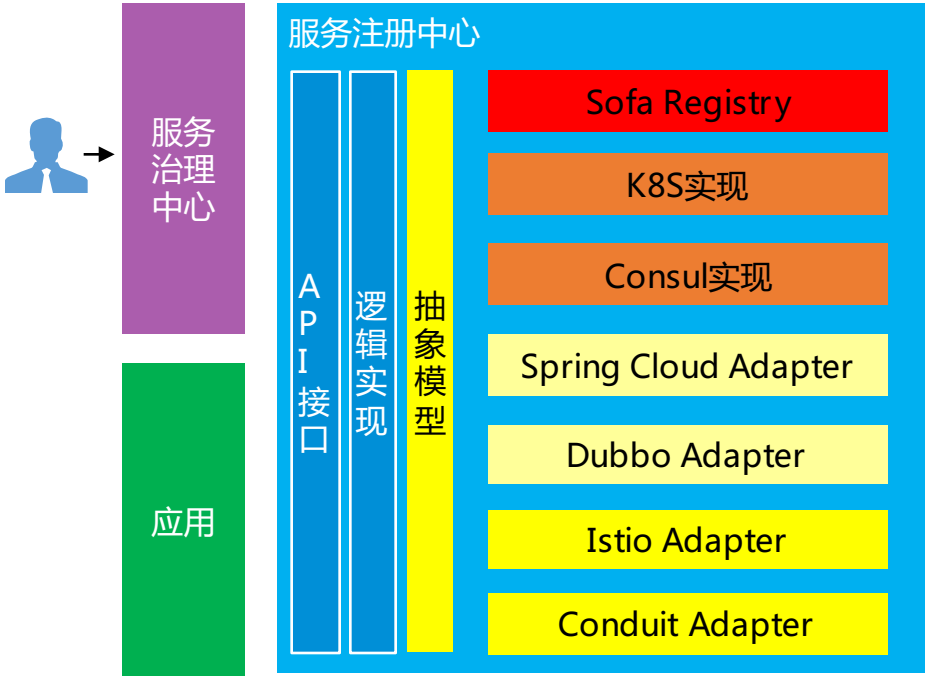
- ✓ 决策：
  - 暂时不确认是否会造成直接影响，先不动
  - 等待实际验证后再决定是否合并report部分
- ✓ 参考
  - Conduit已经在新版本中将report类的功能合并到Sidecar
  - 国内的华为/新浪微博等都选择在Sidecar中实现功能，没有mixer

增强版Pilot：梦幻级服务注册和治理

ServiceMesher

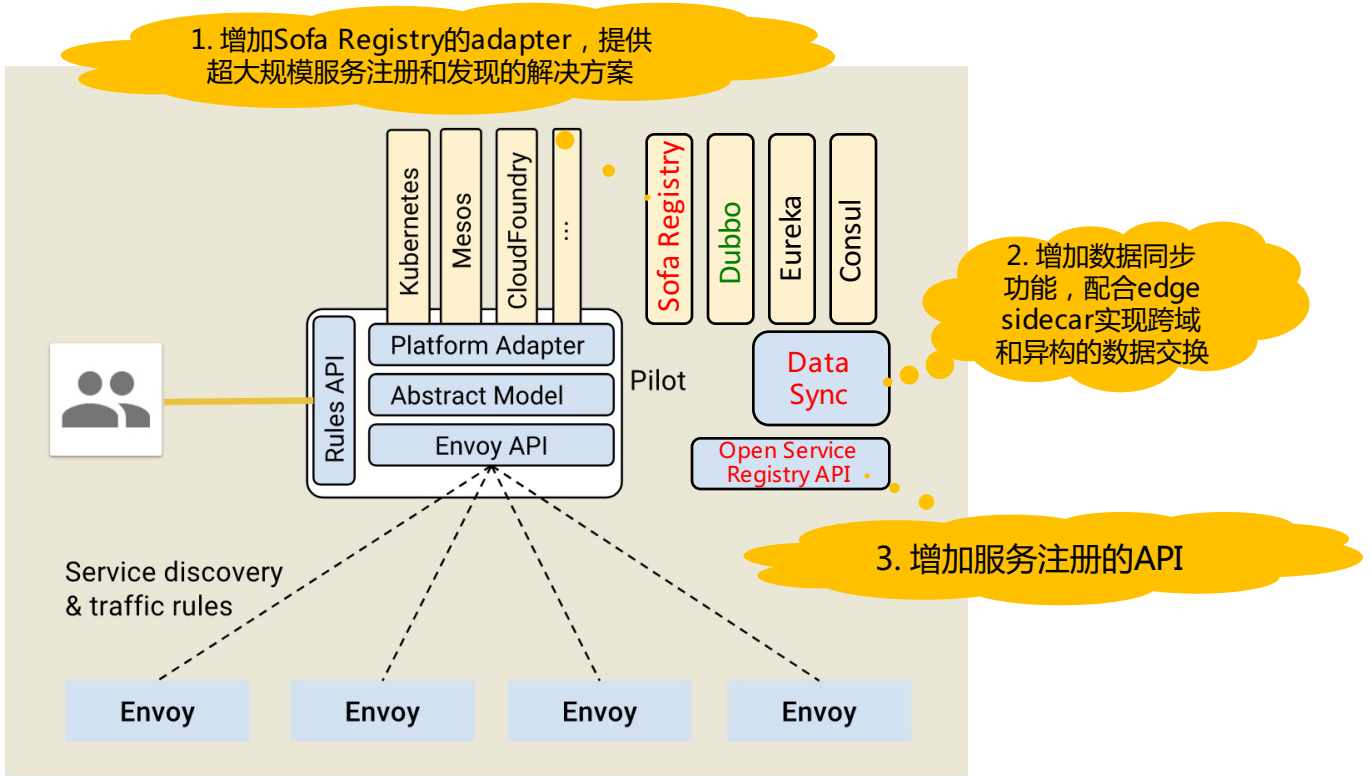
IT大咖说 知识共享平台

- ✓ 支持跨集群
  - 打通多个服务注册中心
  - 支持多个注册中心同步信息
  - 实现跨注册中心的服务调用
- ✓ 支持异构
  - 实现方式不同的注册中心
  - 向Service Mesh的过渡
  - 两个非Service Mesh的打通
- ✓ 终极形态
  - 跨集群 + 异构同时支持
  - 配合其他模块实现更灵活的服务间通讯

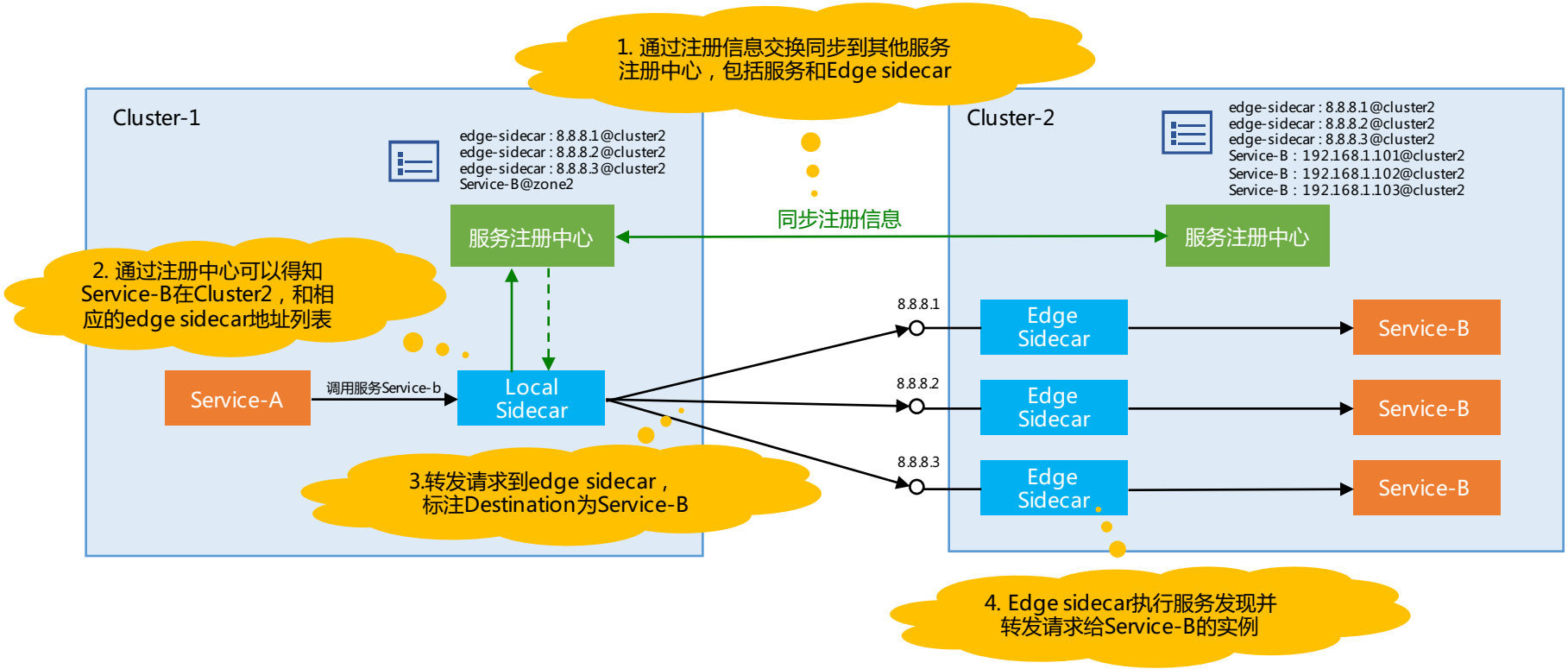


曾经构想过的服务注册中心理想架构

Pilot架构类似：以Pilot为基础做扩展



Edge Sidecar: 东西向服务间通讯的



下回分解：增强版Pilot和Edge Side

7月底北京，第二次Service Mesher线下Meetup

3

Open Source

开源策略

蚂蚁金服，开源开放

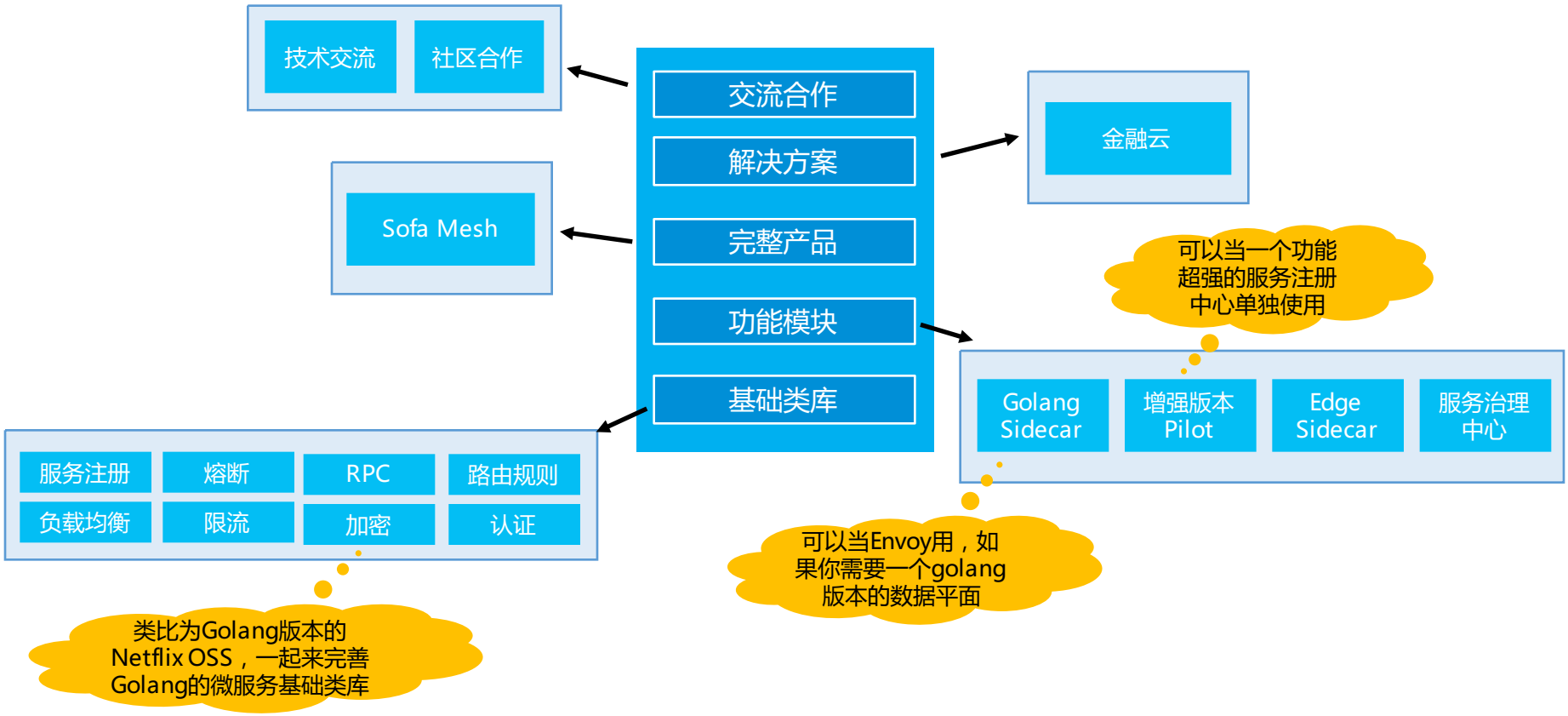
- ✓ 从 4 月份开始逐步开源金融级分布式架构中的各个组件：
  - SOFA Boot
  - SOFA RPC
  - SOFA Tracer
  - SOFA Lookout
- ✓ 科技开放，走出去看更大的生态
  - 蚂蚁有丰富的业务场景，技术体系也经历了很长时间的发展，沉淀了很多自研产品
  - 蚂蚁本身业务上的开放策略，要求技术也要开放，而且要在更丰富的场景下去磨炼
  - 在此期间，我们趟坑无数，走了N多弯路，演进了N个版本，我们期望能过通过开源和开放，让社区跑的更快，节省更多时间
  - 我们认为金融领域下的分布式架构设计有独特的原则，作为实践者，我们期望能在标准化上跟社区一起沉淀和共建，期望做些贡献，有些建树

Sofa Mesh的开源态度

- ✓ 开源的时机
  - 产品完成甚至使用多年之后
- ✓ 开源的内容
  - 陈旧的技术，过时的架构
  - 放弃不再使用的产品
  - 新产品，但是自己不用
- ✓ 开源的动机
  - 秀肌肉，博名声
  - 沦为KPI工程，面子工程
- ✓ 开源项目的维护
  - 被抛弃，或者发展停滞无人维护

- ✓ 开源的时机
  - 直接开源，摆明态度
- ✓ 开源的内容
  - 业界最新的技术
  - 业界最好的架构（努力中☺）
  - 内部使用同样产品落地
- ✓ 开源的动机
  - 吸引社区，谋求合作，开源共建
- ✓ 开源项目的维护
  - 内部使用，保证持续投入
  - 请放心

Sofa Mesh的合作模式：多层次全方



Sofa Mesh开源宣言

我们认可Service Mesh的方向

我们看好Service Mesh的前景

我们勇敢探索

我们耐心填坑

我们积极推进技术进步

我们努力打造优秀产品

我们愿意分享

我们寻求合作

我们的口号

ServiceMesher

IT大咖说 知识共享平台

集结中国力量，共建开源精品

蚂蚁愿意在Service Mesh领域，积极而务实的推进技术进步，以开放的姿态寻求共赢。

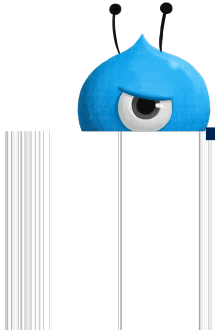
蚂蚁希望联合所有对Service Mesh技术感兴趣的国内厂商/企业/技术媒体，开展不同层面上的交流与合作。

Sofa Mesh on the way !

开源准备中，七月，github见！

Service Mesher社区网站开通！

http://www.servicemesher.com



THANKS