DUBBO应用实践

微服务应用优化过程

卢正贤

平安银行平台架构部中间团队负责人，曾任职IBM应用架构师, 系统架构师

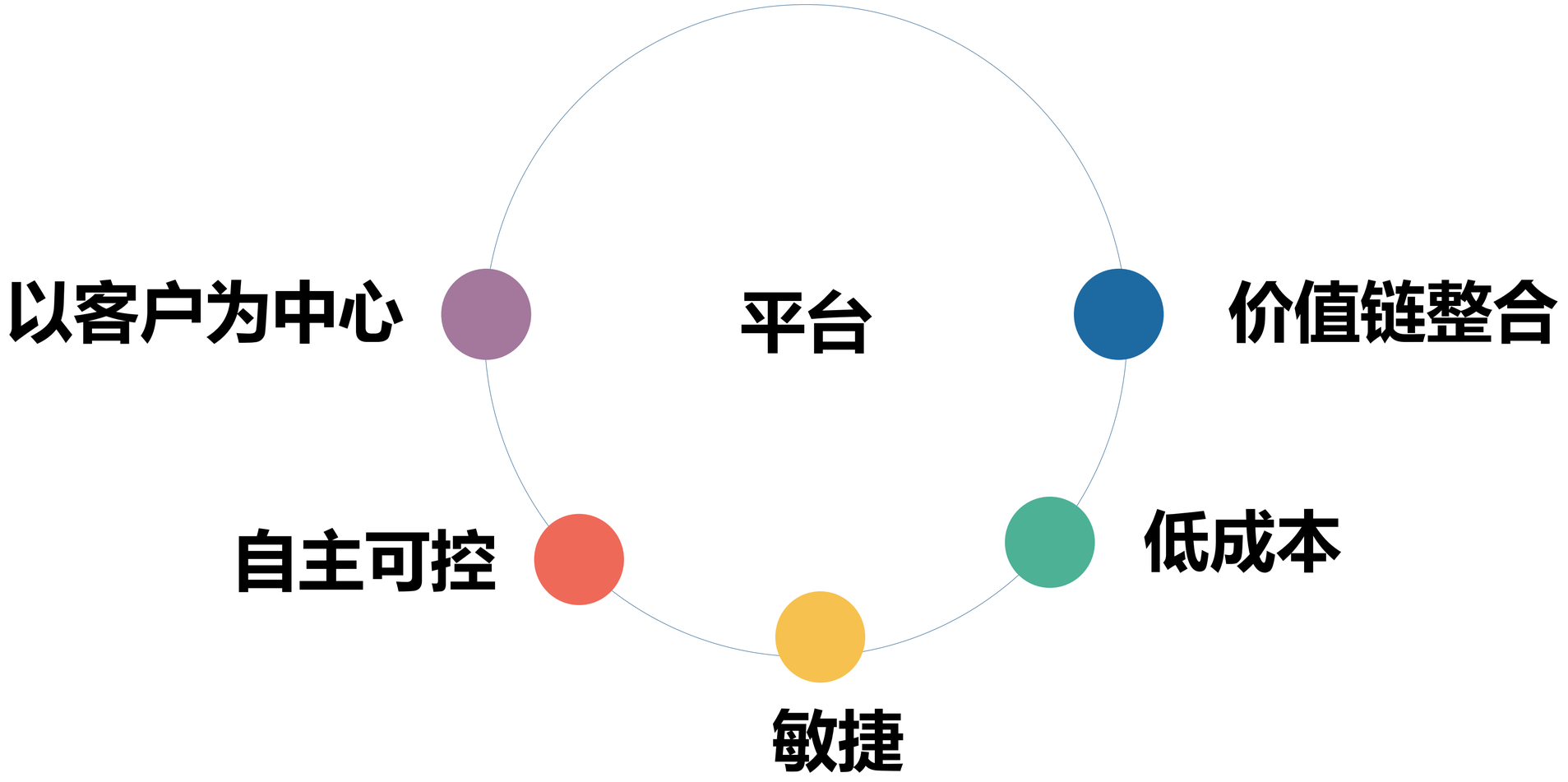
15年老派程序员

喜欢方案讨论，喜欢网络编程

目录

1 选择背景

“转型需要啥?”


以客户为中心
平台
价值链整合
自主可控
低成本
敏捷

Dubbo的优势

成熟可靠

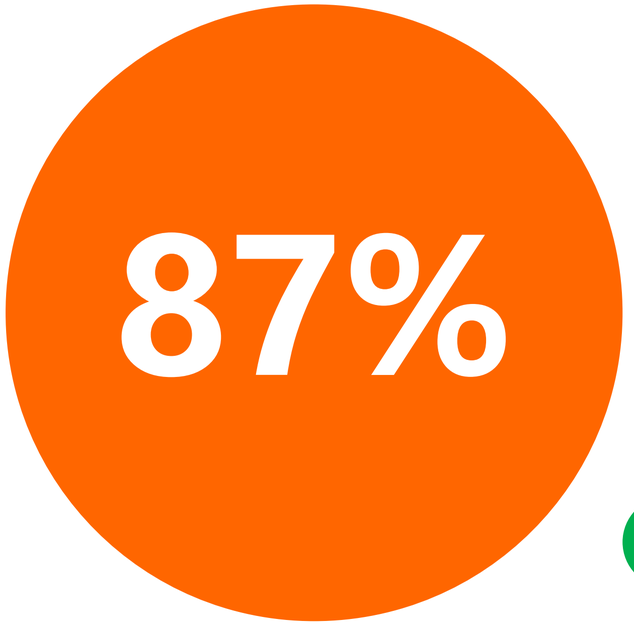
国产开源

微服务框架

高性能

流行的技术栈

Dubbo 应用现状



具有治理能力的Dobbo应用

10% 仅当PRC的Dobbo应用

3% 其他分布式应用

组合生态圈

- 注册中心 ZK
- 配置中心 Apollo
- API网关 自研
- 监控CAT
- DevOps 自研

效果

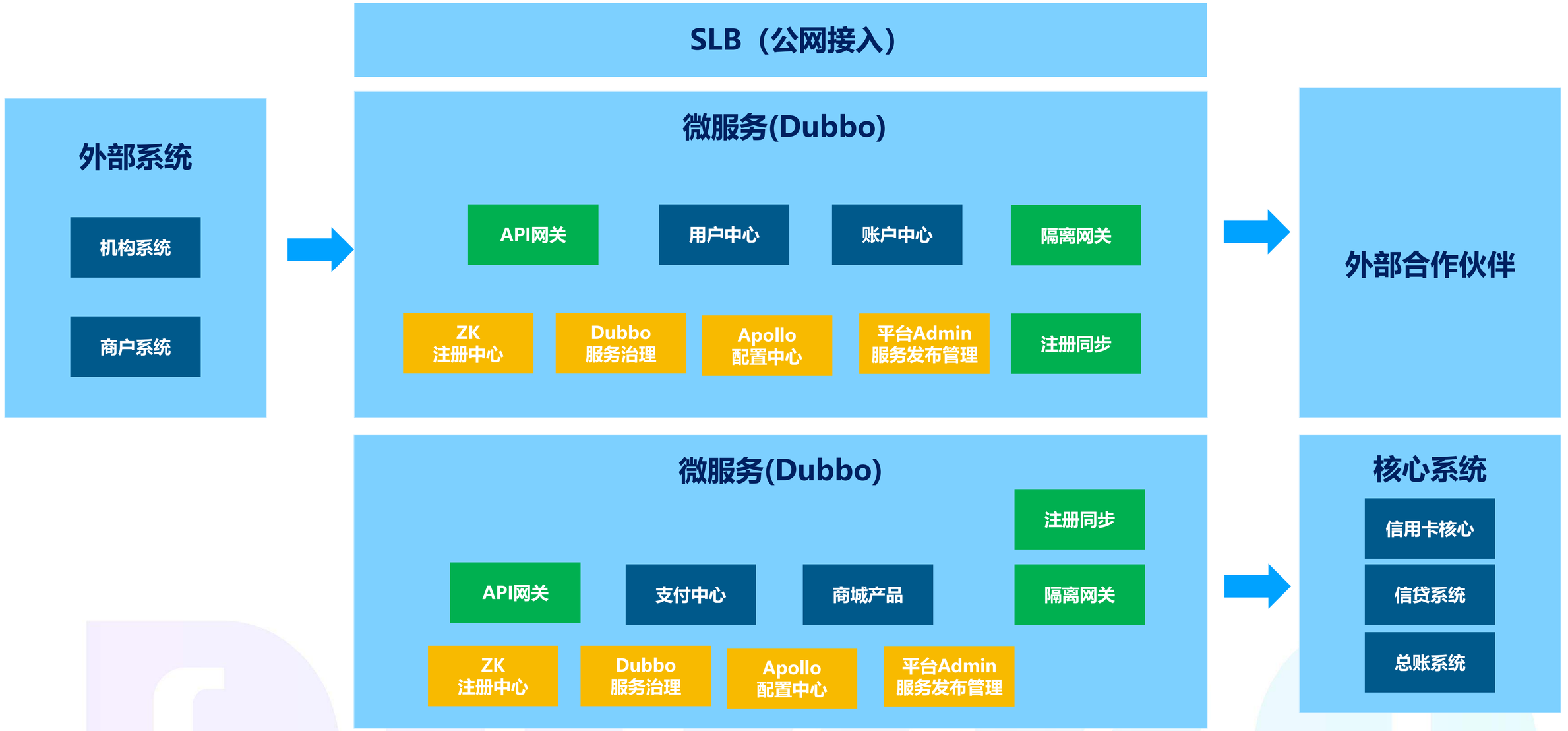
- 高效的公共平台
- 敏捷的客户端产品落地

2 系统架构

共享服务架构

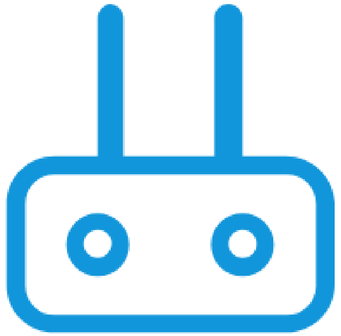

前台
外部系统
机构系统
商户系统
共享服务平台(Dubbo)
API网关
API网关
用户中心
账户中心
支付中心
隔离网关
ZK
注册中心
Dubbo
服务治理
Apollo
配置中心
ESB网关
平台Admin
服务发布管理
外部合作伙伴
核心系统
ESB总线
信用卡核心
信贷
总账
基础设施
CAT监控
Redis缓存

微服务架构(多中心)


SLB（公网接入）
外部系统
机构系统
商户系统
微服务(Dubbo)
API网关
用户中心
账户中心
隔离网关
ZK
注册中心
Dubbo
服务治理
Apollo
配置中心
平台Admin
服务发布管理
注册同步
外部合作伙伴
微服务(Dubbo)
注册同步
API网关
支付中心
商城产品
隔离网关
ZK
注册中心
Dubbo
服务治理
Apollo
配置中心
平台Admin
服务发布管理
核心系统
信用卡核心
信贷系统
总账系统

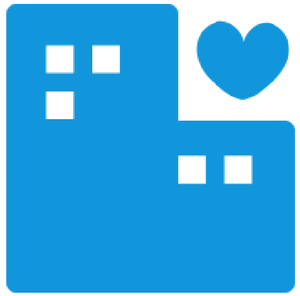
3 痛点及优化

我们发展过程的痛点及优化



网关

- 提供与异构系统之间的交互
- 提供访问核心系统的入口
- 提供对外服务的API发布



同城多活

- 多数据中心的注册同步
- 多数据中心的服务路由
- 多数据中心服务负载均衡
- 高可用的服务调用

测试多子环境

- 测试环境提供多子环境的能力

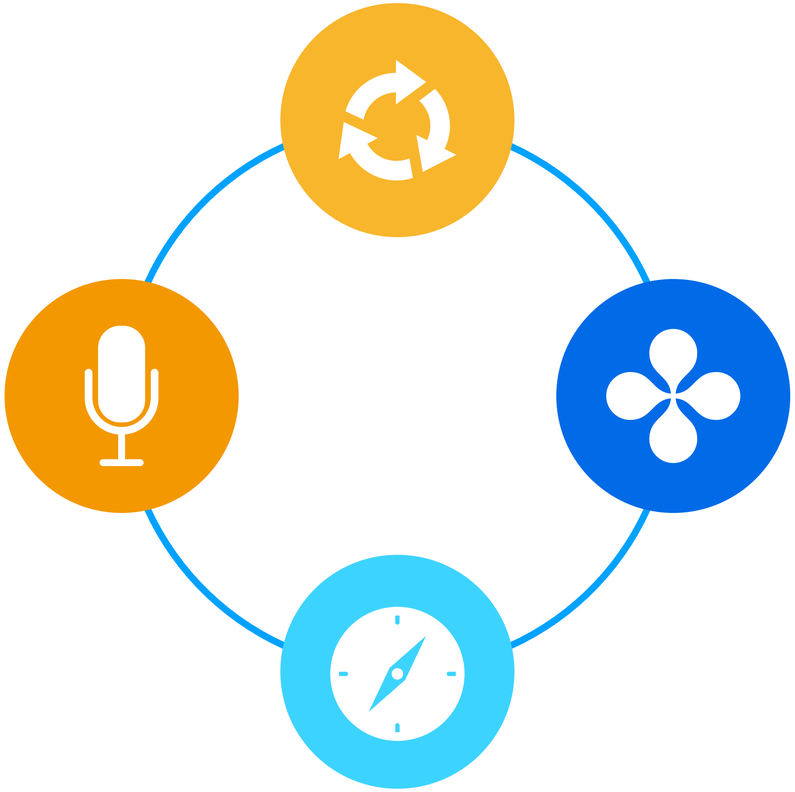
框架扩展

- 路由
- 负载均衡
- 高可用提升
- 注册中心

网关

API网关

- 为外部提供HTTP访问内部服务的入口
- 流量治理
- 访问控制



合作伙伴网关

- 提供微服务方式访问外部合作伙伴的API

隔离网关

- 代理访问核心系统服务
- 代理异构系统接口为微服务

智能设备网关

- 提供微服务方式访问物联网设备接口

网关性能需求?

高并发

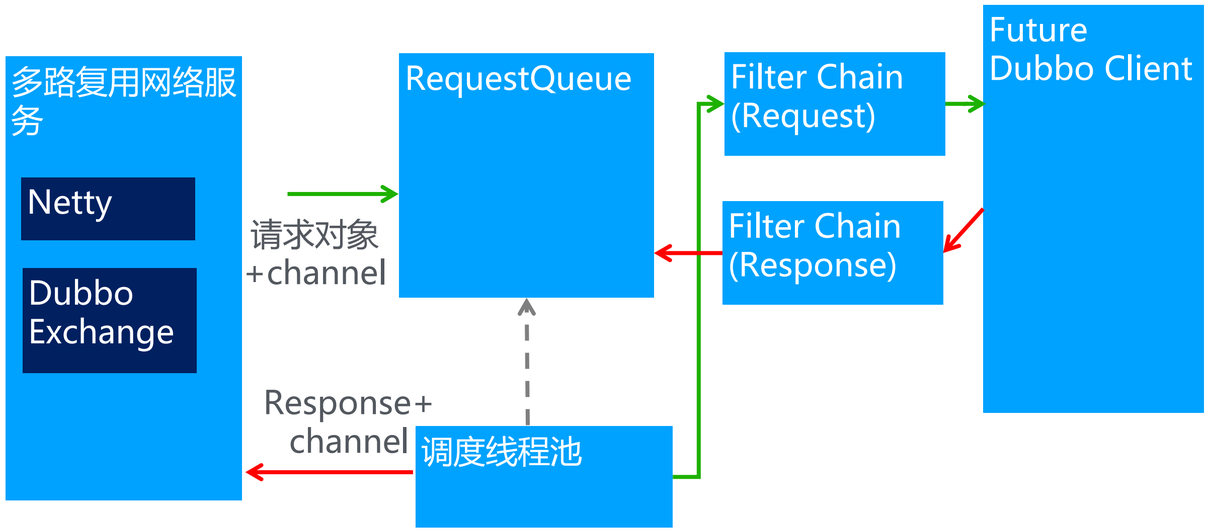
低延时

安全

性能需求

- 低延时
- 高并发
- 10ms的最大超时

网关异步化—高并发



高并发连接

全异步的Filter

Future请求
网络等待不占用线程

网关异步化

- Http/Tcp服务--Netty多路复用
- Dubbo服务--Exchange SPI接收channel对象,改成异步化
- 建立队列缓存正在处理的请求
- 上游服务代理用future避免占用线程

效果

- 并发数10k级别
- 线程稳定
- 很好的支持慢请求的服务

网关优化—低延时

高并发

信用侦测超时
50ms

安全

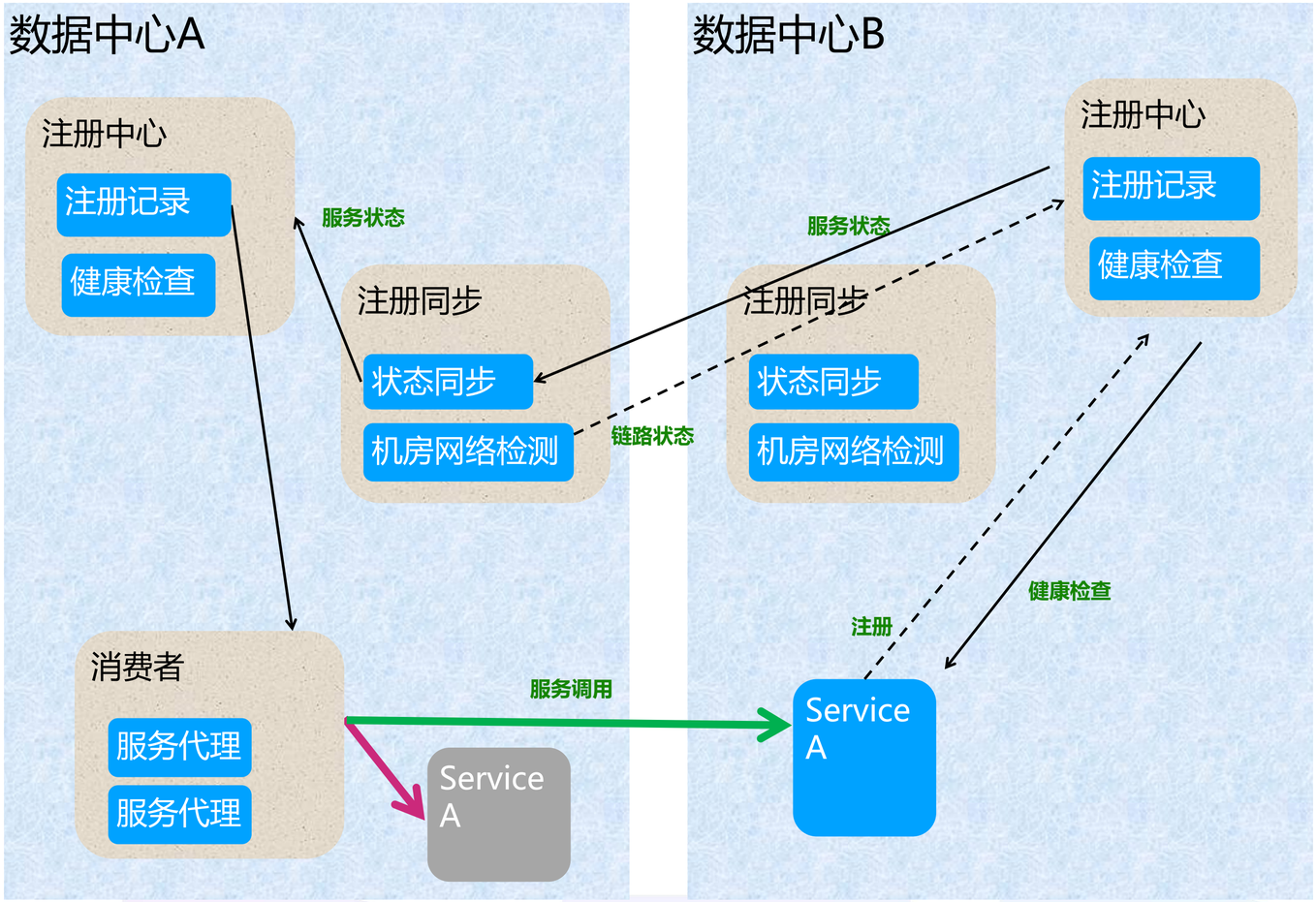
网关延时优化

- 异步IO
- 定时夜间Full GC
- 减小Minor GC时间
- 消费者分级定义

效果

- 提高核心业务可用性
- 降低尾部延时
- 降低最大损耗时间到10ms

同城多活



微服务下的同城多活需求

- 注册中心同步
- 跨机房故障转移
- 跨机房服务健康检测
- 跨机房调用性能
- 数据中心负载均衡

解决

- 机房数据监控和拉取
- 机房间网络检测+对方机房健康状态
- 网络连接失败优先转移其他机房
- 本地优先调用，配置机房负载

需要很多独立的测试环境?

- Paul, 我们部门需要3套独立的环境
- 我要独立的压测环境
- XX部门需要独立的环境
- XX 系统需要11套...

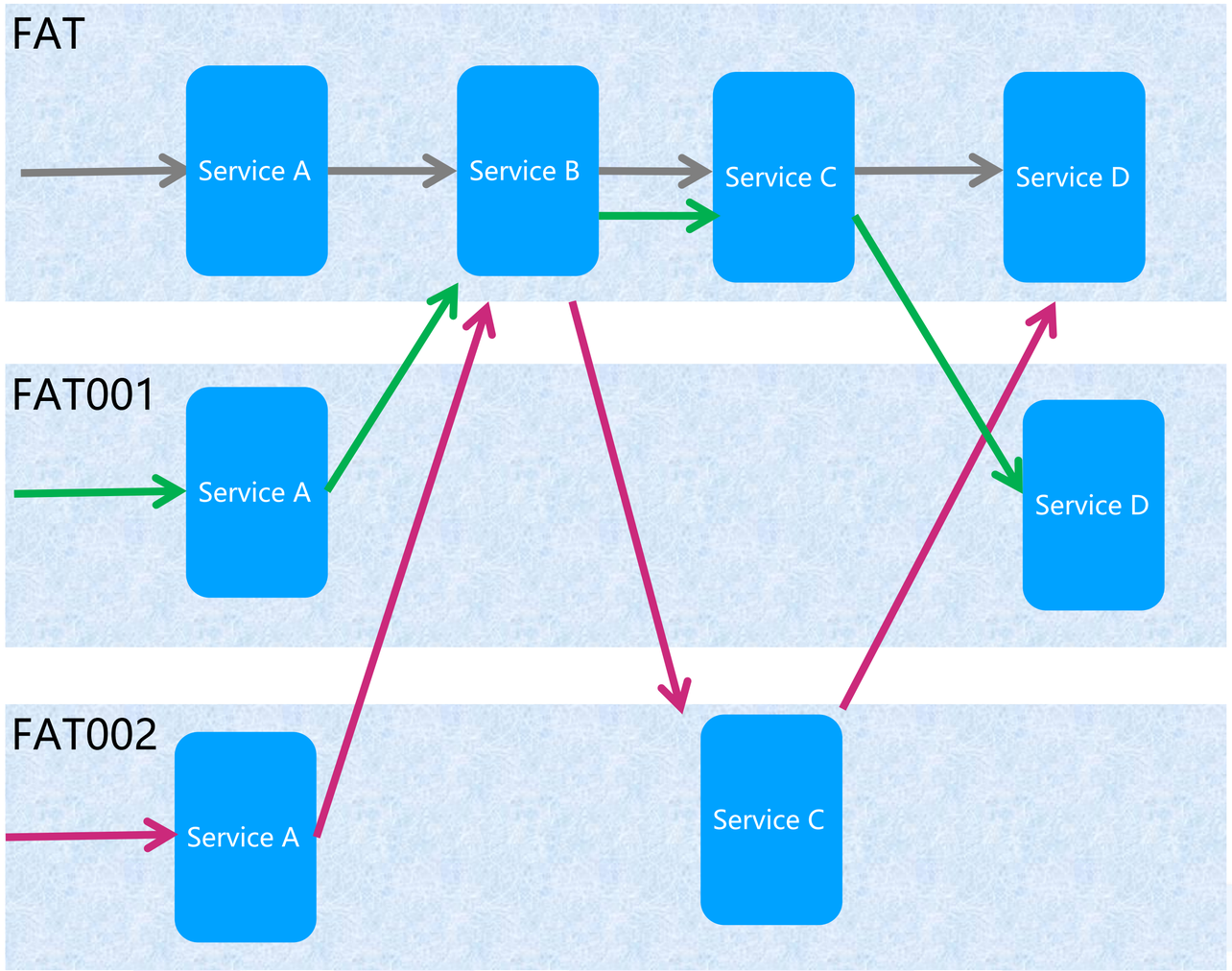
微服务下的架构问题

- 环境问题引起bug相互block
- 资源永远不够
- 搭建环境越来越多
- 人力远远不够维护这些环境

最终的问题

- 低效的公共平台

FAT环境



多子环境调用链

- 子环境的流量优先走子环境应用服务
- 子环境不存在的服务走FAT
- 应用提供方可用配置强制所有流量走某个子环境

实现关键

- 环境参数定义
- 请求带环境标识，流量染色
- 定制路由

框架拓展

- 路由
- 负载均衡
- 高可用提高
- 重构注册中心

路由很简单?

对基础能力的诉求

我们如何拓展路由

路由扩展

多维需求

- 测试子环境穿梭
- 跨数据中心调度
- 应用分流
- 点火内测流量
- 白名单流量
- 业务自定义规则

拓展负载均衡

实现

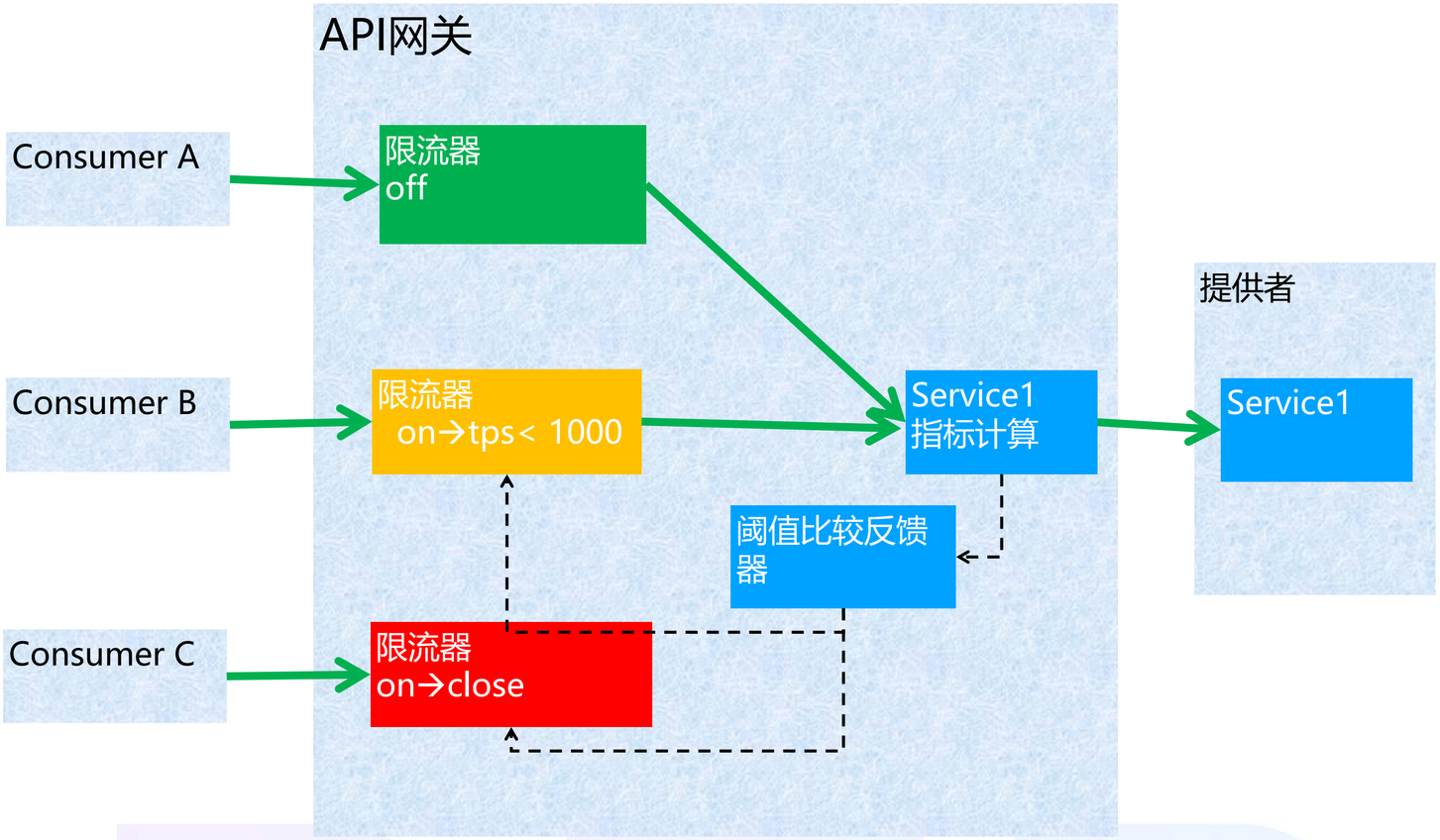
- 灰度
- 新实例热身
- 权重
- 机房流量均衡

动态权重指标

- 耗时
- 并发数

高可用提高?

高可用--保护核心业务



机制

- 消费者提供者分级模型
- 提供者多级阈值
- 阈值出发对级别低的进行降权，限流，拒绝服务

阈值指标

- 并发数 --- 效果最好
- 耗时
- TPS

高可用--重试

老系统不支持重试，需要更高的可靠性

连接异常

流量繁忙

消费者总体服务时间超时

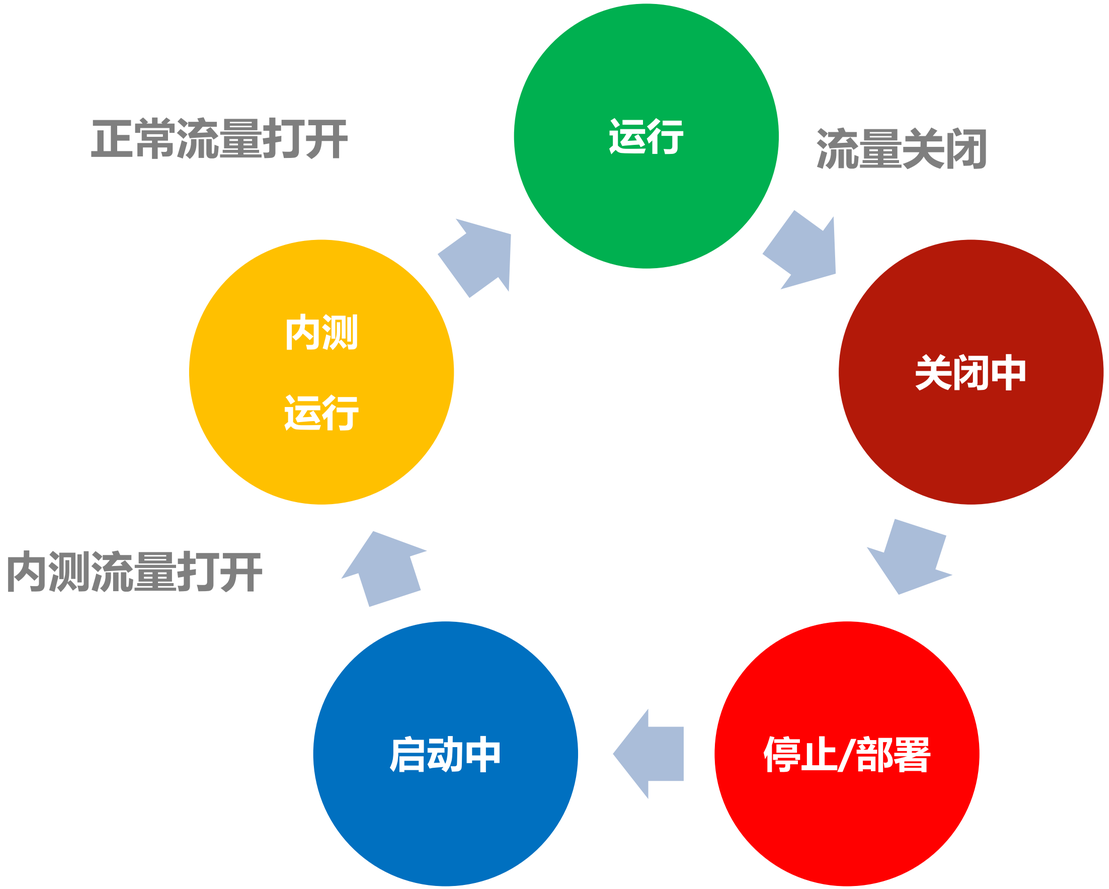
故障转移

- 非幂等→
  网络连接失败重试
  框架异常重试
- 幂等重试 多通道广播重试

重试问题

- 流量繁忙时
- 消费方的服务总体超时限制
- 网络，框架，业务错误码要严格的分类

提高可靠性--实例状态管理



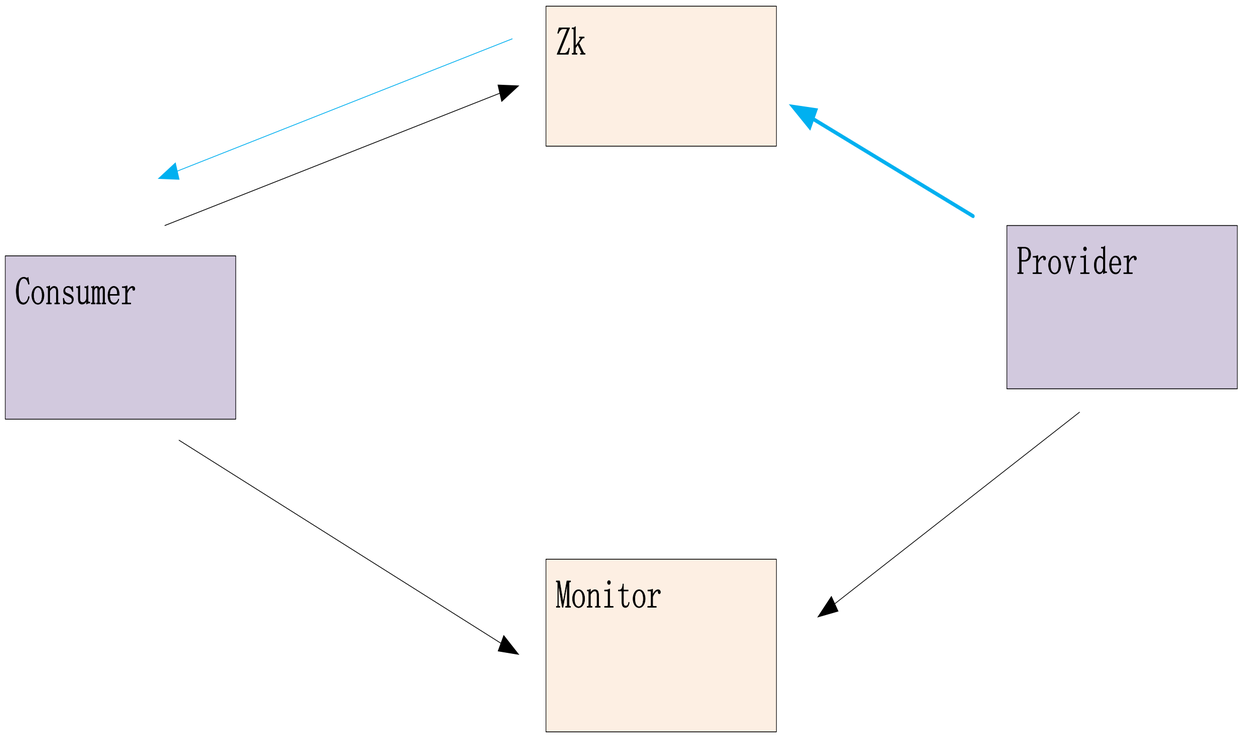
改进

- 定义实例的状态
  服务状态继承实例状态
- 定义每个状态的流量规则
- 测试状态给实例启用前能够测试
- 关闭中，避免流量进入
- 所有流量处理后才能进入停止状态

效果

- 实例部署过程流量无损
- 确保实例启用前做充分测试
- 优雅停机

注册中心问题



Zk注册中心成为阻碍

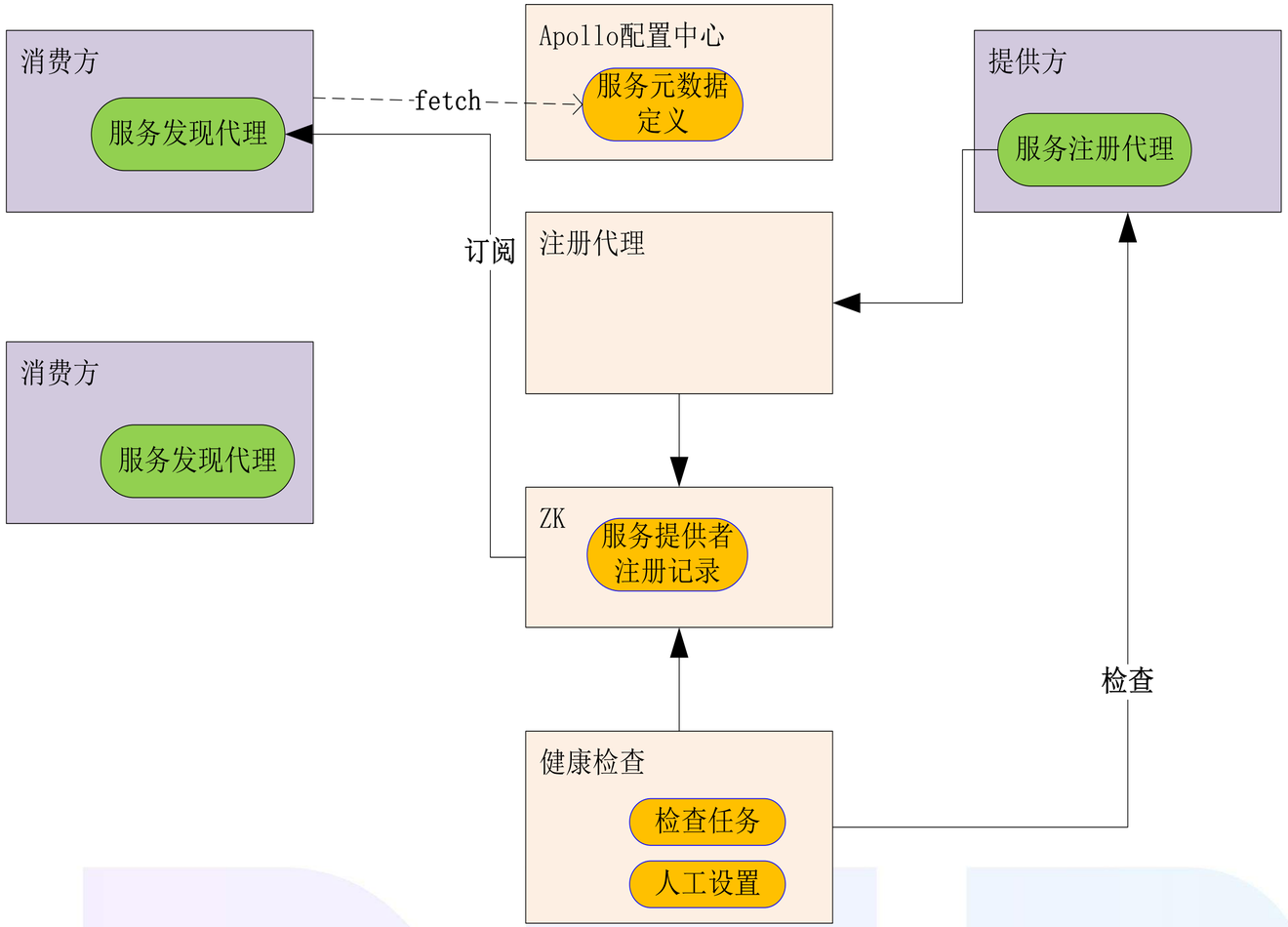
模型臃肿

不能灵活管控

使用中的问题

服务断开后不能实时反映到消费端

重构注册中心



增强功能

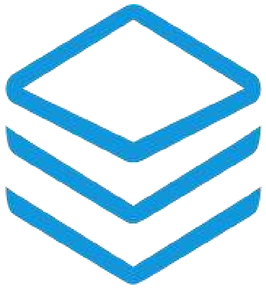
- 提供服务注册
- 提供服务发现
- 提供健康检测

调整模型

- 配置与注册信息分离
- 服务注册和服务发现分离
- 重构元数据模型，以解决方案为主线
- 重构服务发现

4 强化路上

转型中—解决方案即服务



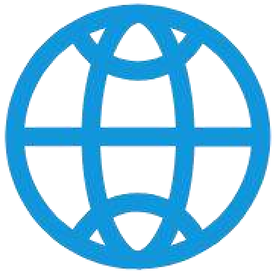
解决方案为中心

服务治理升级

开放平台

升级运维

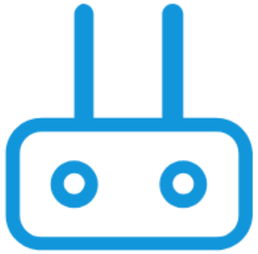
解决中



开源后版本兼容

大文件传输

契约管理



路由

监控接口开放

未来---Service Mesh?

框架与业务解耦，下沉到基础设施中

多参与大家Service Mesh应用实践

Thank you !