自研Service Mesh在趣头条的实践

徐鹏

基础架构部架构师

自我介绍

INTRODUCTION

徐鹏

在游戏行业沉浸近10年,对统一网关层有深入的理解, 目前在趣头条基础架构部,主要负责边缘网关和Service Mesh的研发和产品化

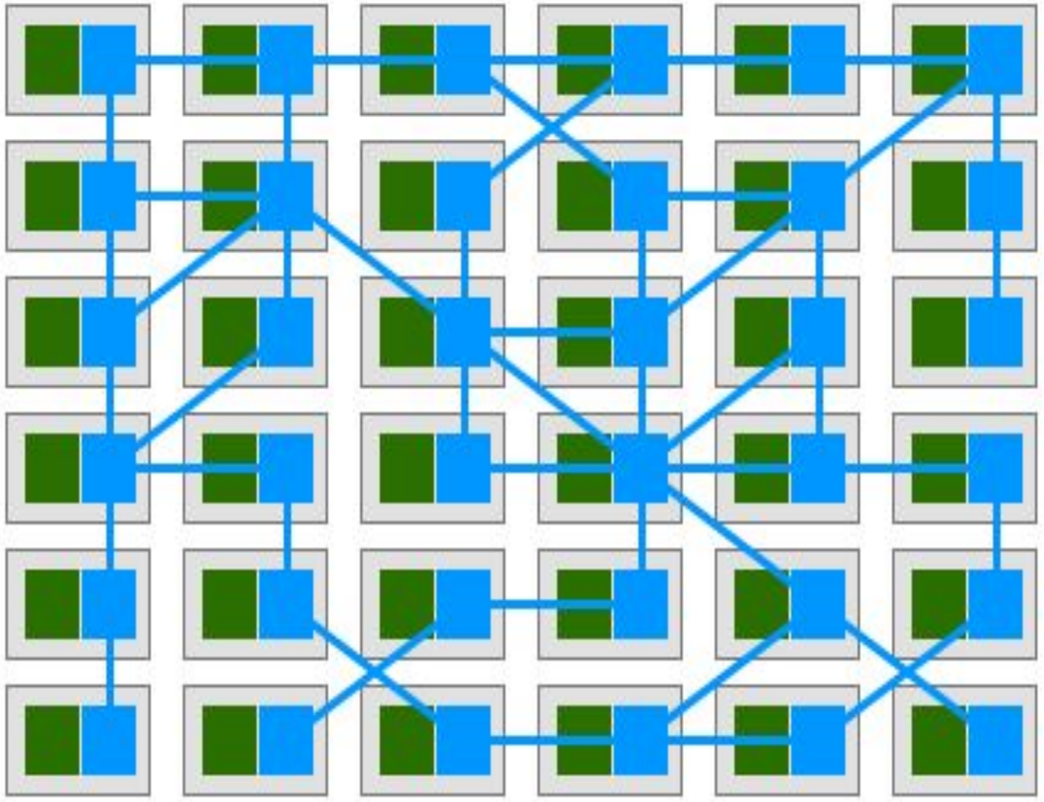
目　录

CONTENTS

01

简单介绍 Service Mesh

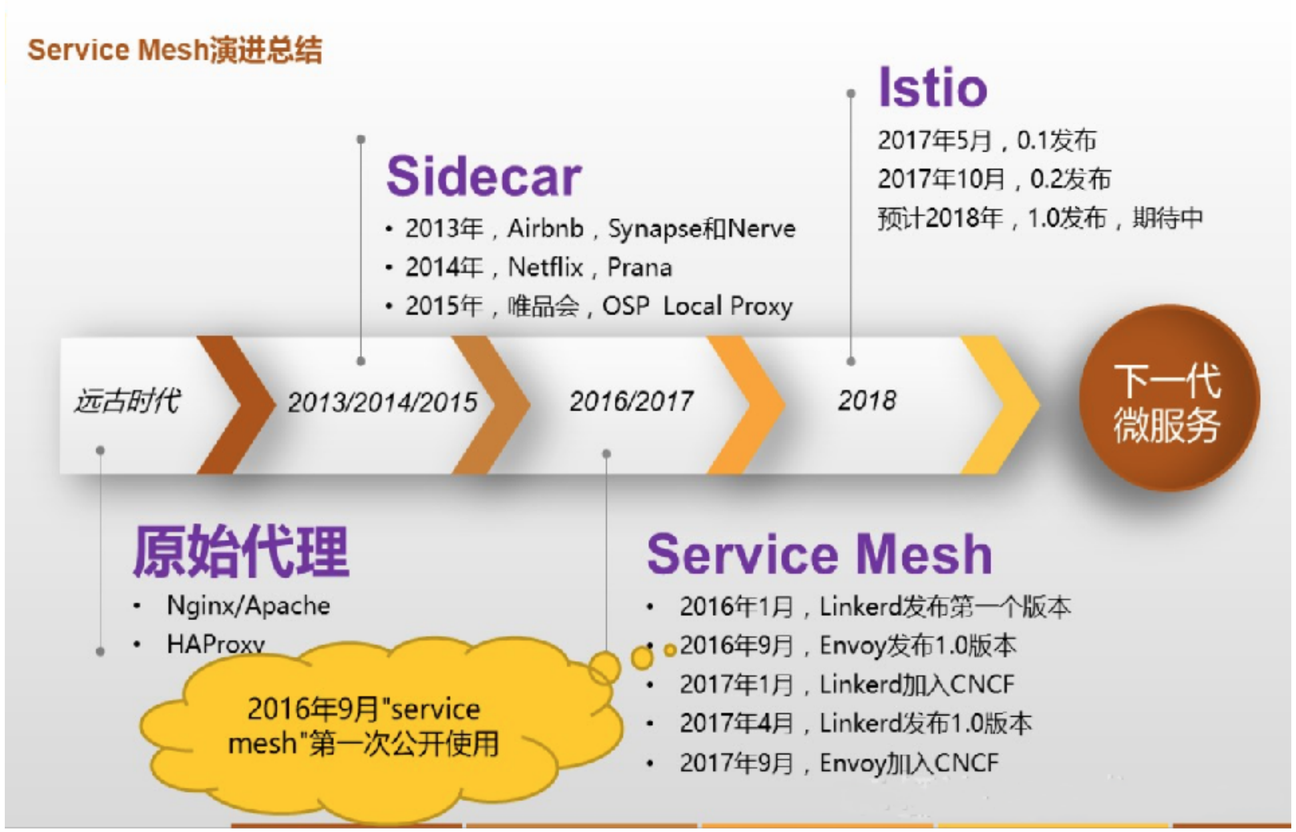
Service Mesh Sidecar模式



绿色部分代表服务程序

蓝色部分代表Sidecar

Service Mesh演进历程

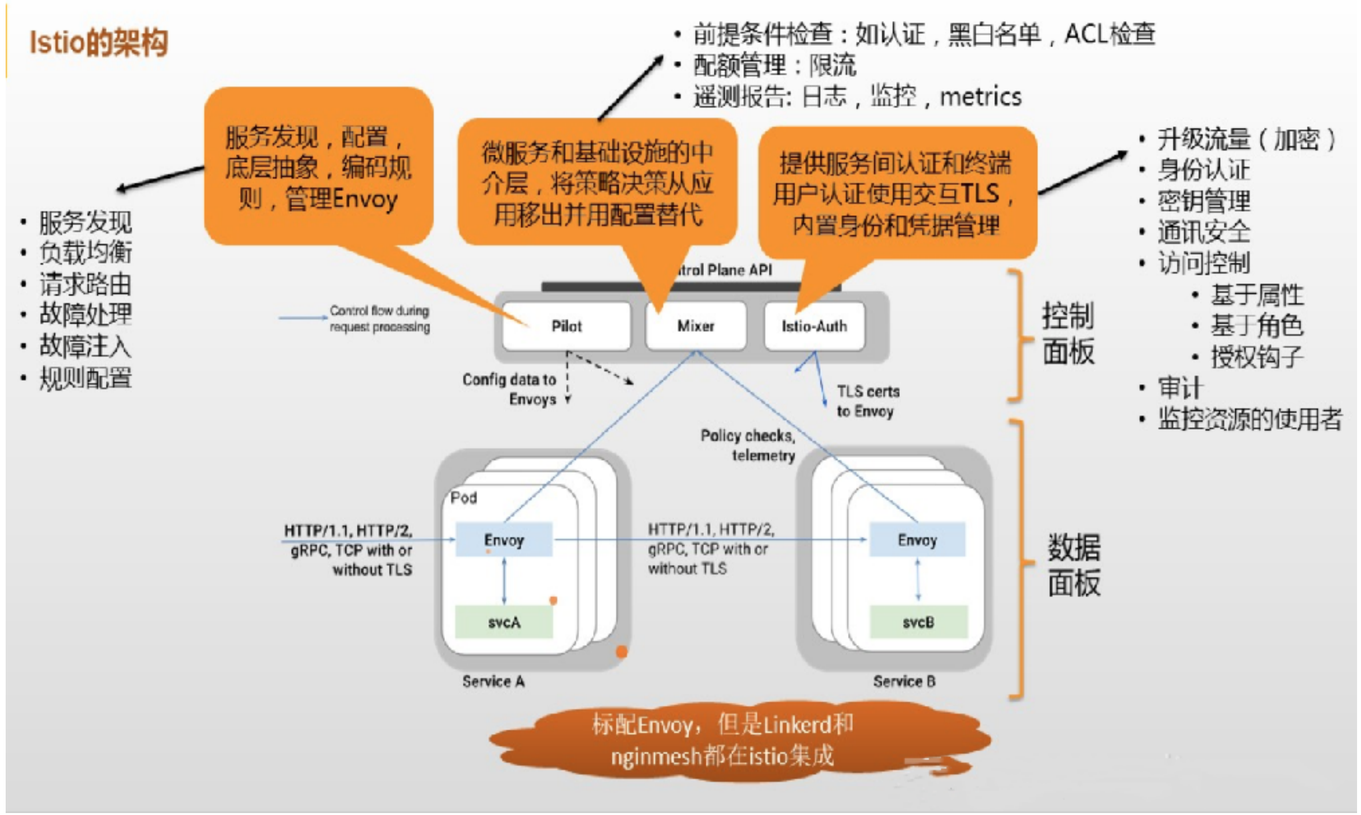


Sidecar并非新概念，

早在2014年Netflix等公司就

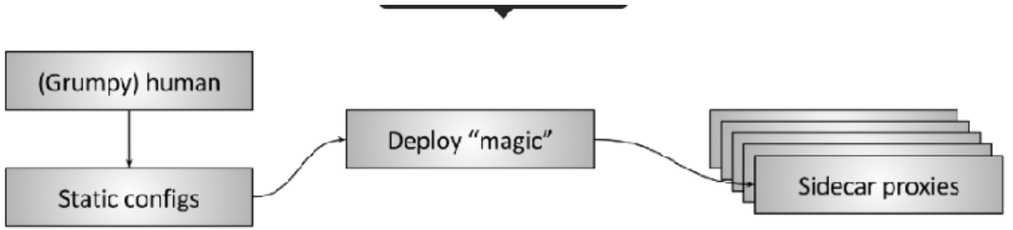
开始在公司内部大规模应用。

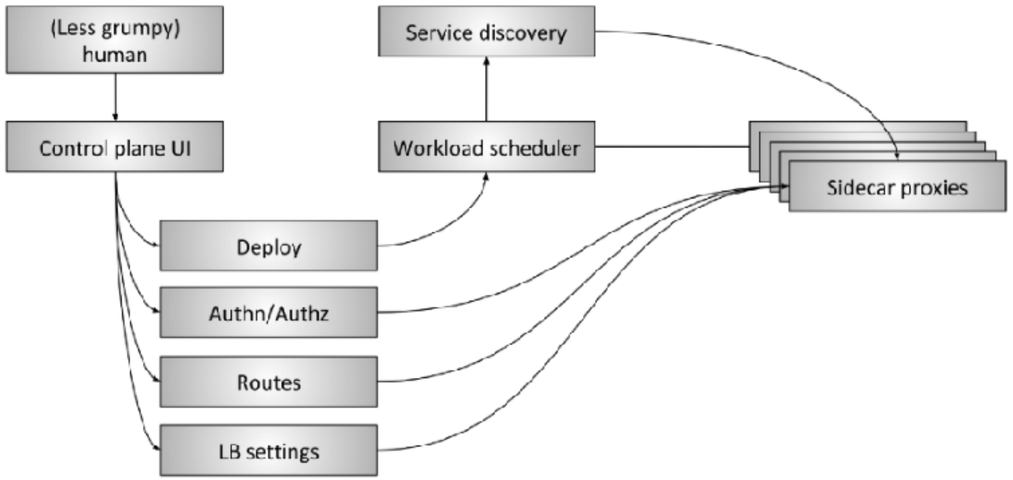
最典型的Service Mesh架构



- 这是一个最典型的 Istio+Envoy的 Service Mesh
- Istio作为控制平面
- Envoy作为数据平面
- 数据平面并非新概念，我们最常用的Nginx就是最典型的数据平面

什么是控制平面





控制平面很抽象?NO!这就是人类控制平面

Service Mesh要演进的控制平面：

1. 人，重大决策，比如服务降级
2. 认证,路由,负载均衡等配置
3. 资源调度程序,比如k8s,所有的机器资源可编程控制
4. 服务注册发现
5. Sidecar数据平面

02

自研Service Mesh(Negri)介绍

why Go

1. 公司技术栈主要基于Go

2. 社区优好,开发迅速

3. Go语言非常适合开发网络服务

Why自研Service Mesh

1

社区方案不合适目前公司环境

Istio等控制平面不成熟,Envoy c++开发,和公司技术栈不符合,扩展困难

2

公司的入口服务统一管理

自研的auth,加解密,abtest,trace等入口服务

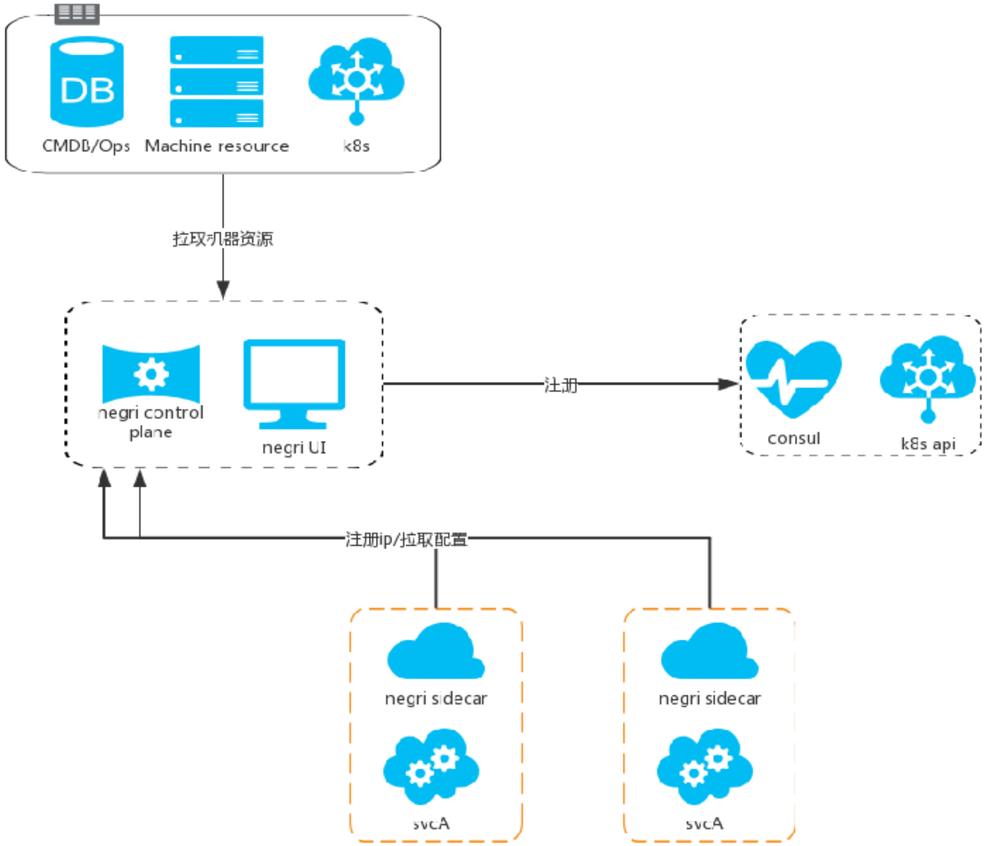
3

适合公司的研发/运维环境

在部署和研发维护上更友好,兼容传统主机和容器环境

自研Service Mesh(Negri)介绍

* negri控制平面负责管理机器资源
* 通过negri ui提供直观的图形界面
* 屏蔽后端注册中心,对k8s,consul等保持兼容



- Negri控制平面负责管理机器资源
- 通过Negri UI提供直观的图形界面
- 屏蔽后端注册中心，对k8s,consul等保持兼容
- 通过IP的方式获取配置,而不是服务名

自研Service Mesh Negri特性

① 语言无关

无需开发多套不同语言框架,PHP.Golang,Java,甚至Nodejs,Python都可以接入

② 服务注册发现

省去了slb部署环节,可以自动发现服务节点变化

③ 维护成本低

重框架维护成本非常高,DRY会引发大面积故障,升级成本高

④ 服务治理

限流,熔断,降级,trace,metrics,log,错误注入

⑤ 图形化控制面板

提供了统一的ui,控制服务治理的各种配置,展示服务的metrics,trace,日志,调用关系等

⑥ 自研业务的支持

abtest,trace,auth,加解密,sign验签等功能的支持

自研Service Mesh Negri特性

1. 对业务无侵入

只需要改动两行代码,便可以接入

2. 升级业务无感知

通过fork进程,配合systemd,实现平滑重启

3. 动态变更配置

通过配置中心 quconf 和微服务控制面板,动态更新配置,实时控制限流熔断降级等中间件

4. 可控权重灰度发布

通过单独部署低配置机器和动态修改服务流量,灰度发布服务

5. 资源隔离

通过cgroup限制资源(CPU,内存等)使用， 不影响业务机器

6. 统一的插件编写体系

无需区分client还是server端中间件,统一抽象为server端中间件,编写一次,多端运行

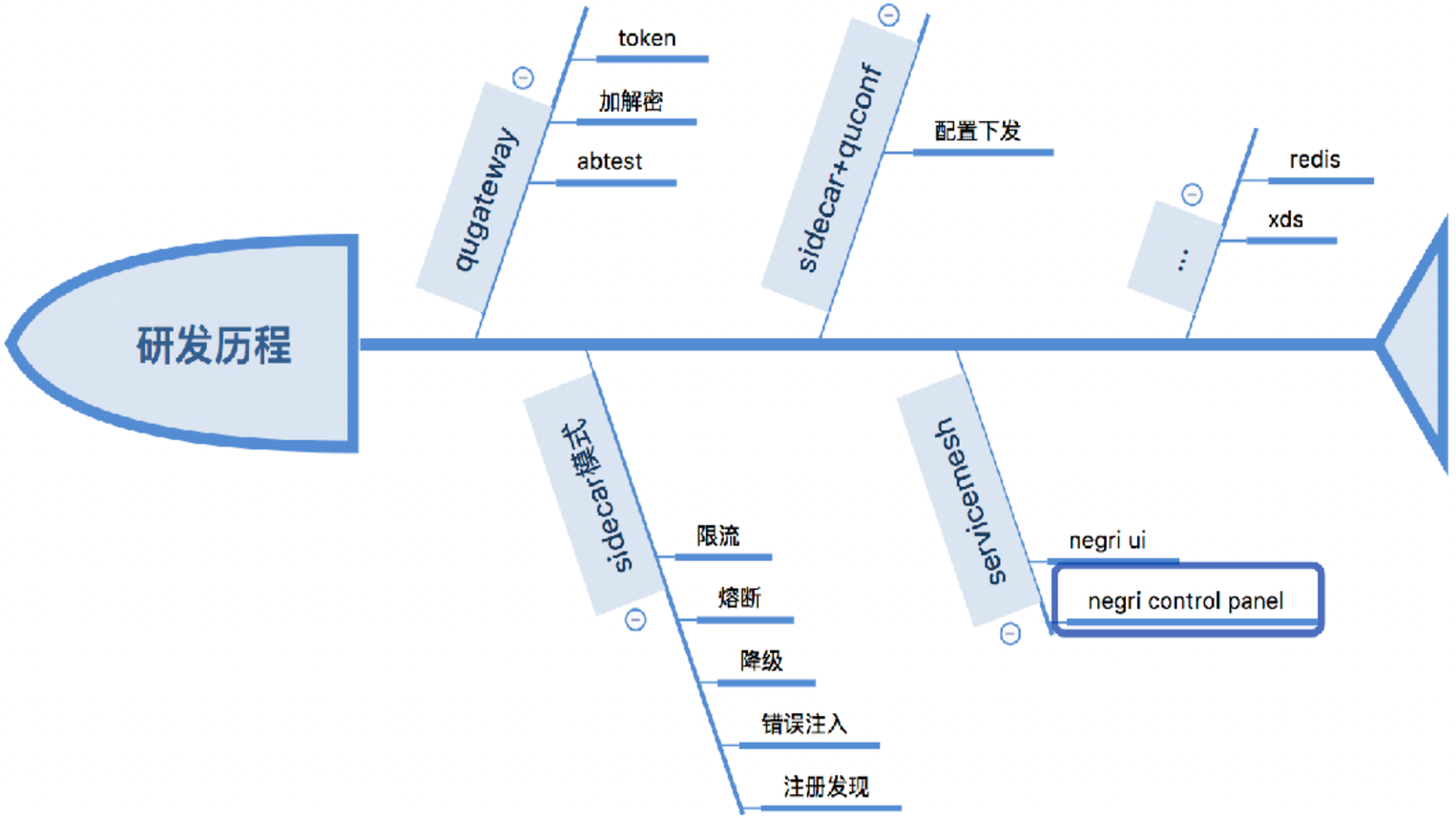
配置文件–插件模式参考kong

```
[services] #frontend
    ## >>==================sidecar=================================================================
    [services.logserver] # logserver 特定服务名
    upstream = "logserver.service.discovery"
        [services.logserver.routes.sidecar]
        rule = "Headers:X-Qtt-Meshservice, logserver" # 这里固定为 X-Qtt-Meshservice:[服务名]
        priority = 100
        enableAccessLog = true # 是否开启访问日志
        [services.logserver.routes.sidecar.logserver] # 启用 logserver 请求劫持
        enable = true
    [services.user]
    upstream = "qukan-user.service.discovery" #服务名
    passHostHeader = true
    retry = 3
    requestTimeout = "10s" # 请求后端服务超时时间, 默认为20s
        [services.user.routes.sidecar]
        rule = "Headers:X-Qtt-Meshservice, qukan-user" # 这里固定为 X-Qtt-Meshservice:[服务名]
        enableAccessLog = true # 是否开启访问日志
            [services.user.routes.sidecar.ratelimit] #限流
            extractorfunc = "request.host" #request.header
            rate = 3000
            burst = 1
```

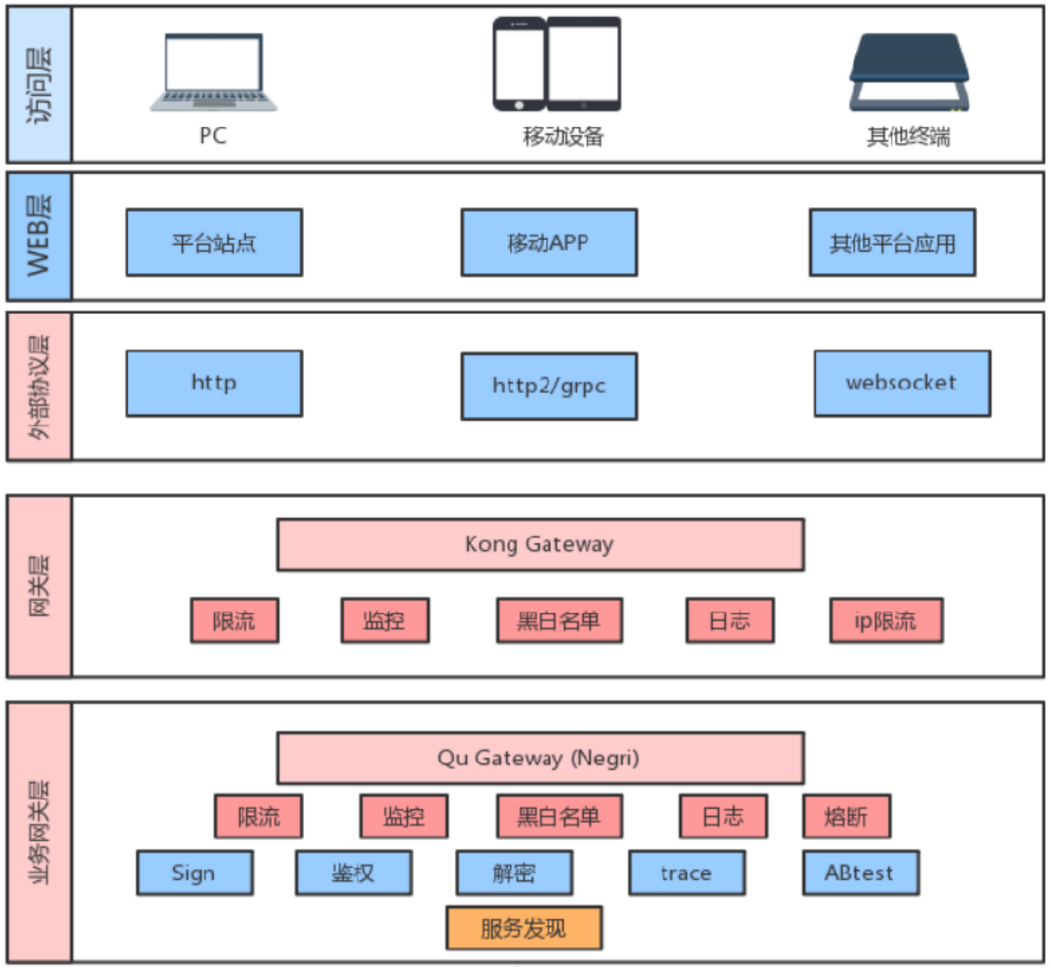
03

Negri研发历程

Negri研发历程


研发历程
qugateway
token
加解密
abtest
sidecar+quconf
配置下发
...
redis
xds
sidecar集合
限流
熔断
降级
错误注入
注册发现
servicemesh
negri ui
negri control panel

Negri最早作为业务网关开发



- 早期作为业务网关开发
- 开发框架的过程中,发现对于服务发现和治理,PHP几乎无能为力
- 讨论发现Negri可以通过本地代理(Sidecar)模式,治理PHP服务

Negri网关模式开发过程

① 一周时间

基于现有的Go脚手架和oxy库(traefik),只用一周时间搭建出原型,并且性能不错

② 性能超越kong(Nginx)

裸跑100k qps,略低于kong的12w,但启用中间件后,得益于go的异步性能,全面超越kong

③ 问题/持续优化

1. 优化transport锁瓶颈
2. 建立对象池
3. cpu资源占用瓶颈

④ 基于net/http库

1. 代码质量高,官方库,稳定,同时支持http和http2
2. 可以代理,grpc,websocket,http
3. 直接提供反向代理模块

⑤ 所有都是中间件

抽象中间件层,限流,熔断,日志,metrics,都是基于中间件开发

性能优化–榨干CPU全核心性能

1 transport锁瓶颈

拆分多个transport,设置核心数相关的连接池数量

2 对象池

http proxy buffer pool

3 路由缓存

用了强大的路由规则库gorilla/mux,但sidecar模式种路由数量过多的时候性能下降严重,建立路由缓存解决

4 优化metrics

采用counter和hisgram类型,相比gauge和summery将计算放在proms

5 日志reopen

通过reopen日志的方式,大日志切分,不会阻塞进程

Sidecar模式转变

1

单端口还是多端口

中间件编写过程中,发现单端口,需要重复编写客户端和服务端中间件,多端口只需要一次

2

配置如何差异化下发

独立出服务私有化配置的概念,通过viper.merge的方式加载

3

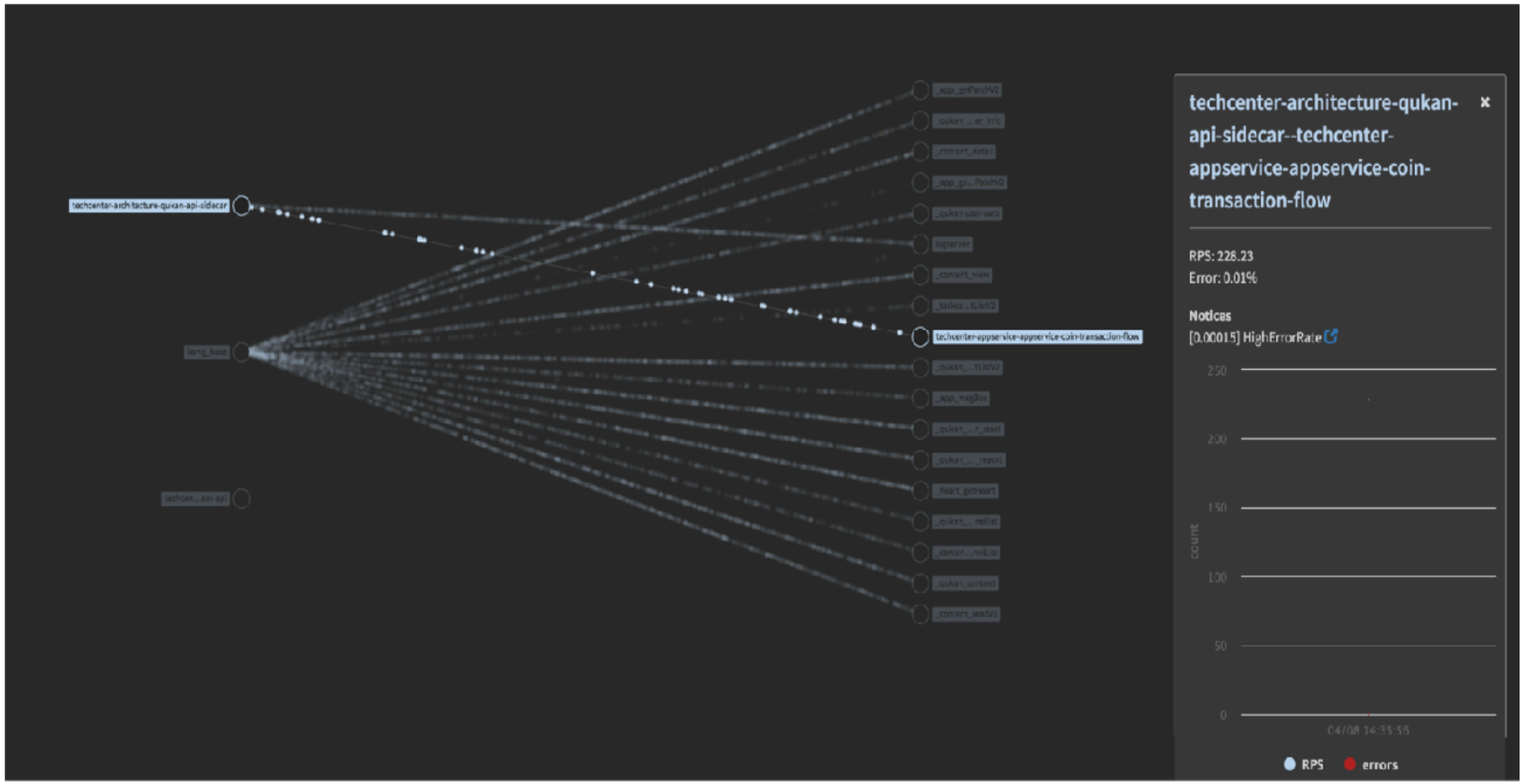
从高并发到低延时

sidecar模式相对网关,需要更关注低延时,而不是高并发

04

Negri的最佳实践

最佳实践–金币详情服务改造–服务调用关系



最佳实践–自动接入metrics面板

最佳实践–实时查看服务状态

用户服务

控制面板 / 用户服务

服务详情 监控数据 调用链 日志 配置

点击根据tag筛选： 所有 sidecar

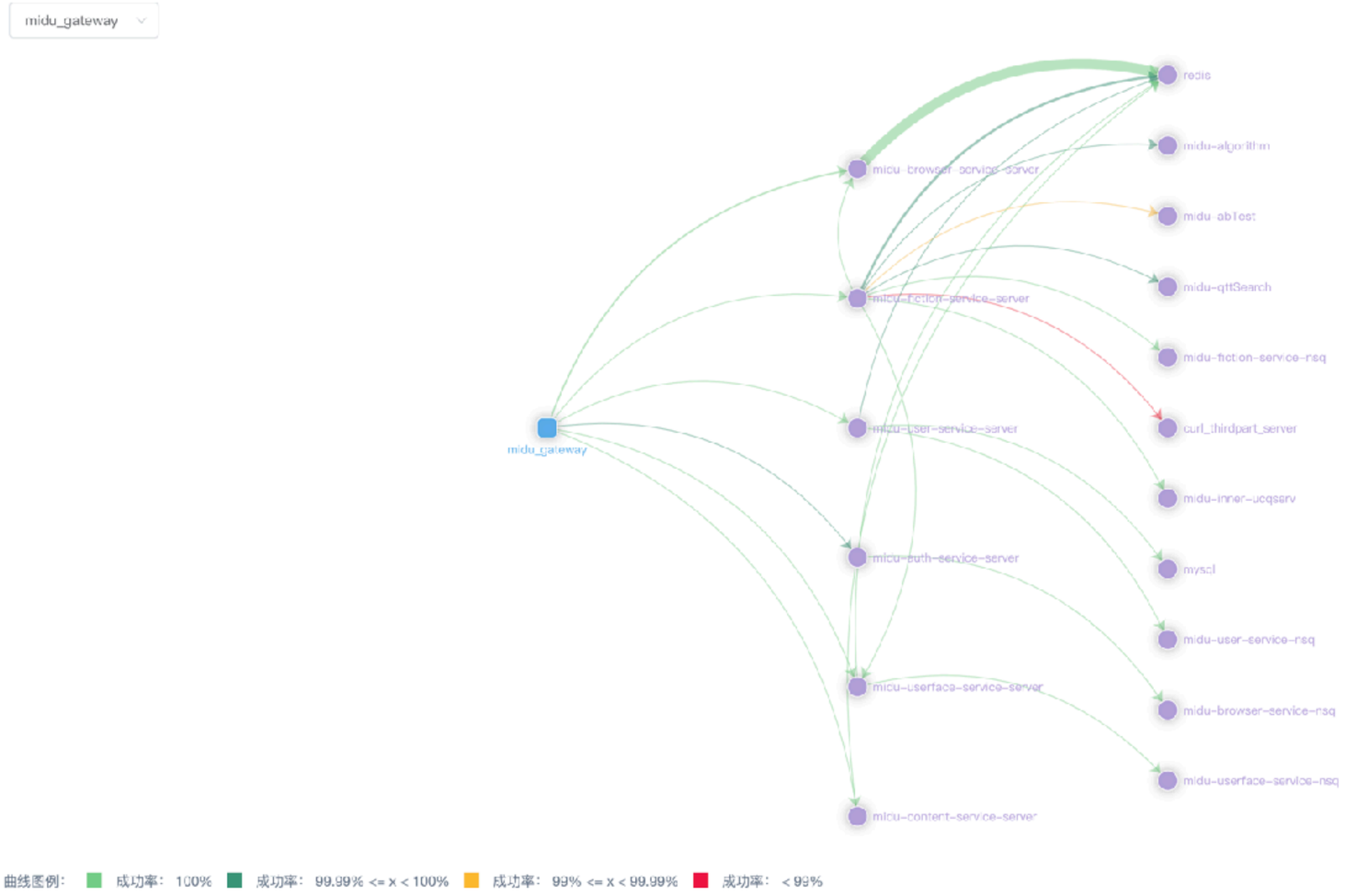
ip	健康状况	tag	meta	权重
172.16.51.142:8103	✓	["sidecar"]	{"Hostname":"bjc-techcenter-appservice-appservice-coin-transaction-flow-01"}	{"passing":1,"warning":1}
172.16.51.141:8103	✓	["sidecar"]	{"Hostname":"bjc-techcenter-appservice-appservice-coin-transaction-flow-02"}	{"passing":1,"warning":1}

最佳实践–trace调用链

米读 | 服务名 | 2019-04-08 | 搜索

服务名	成功次数	失败次数	成功率	最大耗时	最近耗时	平均耗时	99线	最大QPS	操作
midu-fiction-service-server	17731729	1484	99.99%	4043.648 ms	2.864 ms	2.914 ms	<=30ms	974	明细 主调 被调 调用图
midu_gateway	32346792	268	99.99%	1033.315 ms	6.968 ms	6.900 ms	<=50ms	1268	明细 主调 调用图
midu-auth-service-server	184386	1	99.99%	592.268 ms	8.465 ms	11.006 ms	<=200ms	57	明细 主调 被调 调用图
midu-user-service-server	11077821	1	99.99%	1687 ms	1.413 ms	1.367 ms	<=20ms	581	明细 主调 被调 调用图
midu-browser-service-nsq	1400	0	100%	6.556 ms	4.268 ms	3.829 ms	<=10ms	2	明细 被调 调用图
midu-userface-service-server	1934346	0	100%	1004.901 ms	2.197 ms	2.179 ms	<=20ms	149	明细 主调 被调 调用图
midu-user-service-nsq	94219	0	100%	35.55 ms	1.643 ms	1.621 ms	<=5ms	15	明细 被调 调用图
midu-trade-service-server	2436	0	100%	3.132 ms	1.139 ms	0.947 ms	<=3ms	6	明细 调用图
midu-userface-service-nsq	70726	0	100%	103.698 ms	3.689 ms	3.643 ms	<=10ms	12	明细 被调 调用图
midu-fiction-service-nsq	2810	0	100%	12.539 ms	3.671 ms	3.730 ms	<=10ms	3	明细 被调 调用图
midu-content-service-server	2390	0	100%	488.225 ms	1.326 ms	2.536 ms	<=3ms	10	明细 主调 被调 调用图
midu-browser-service-server	61416861	0	100%	973.582 ms	1.192 ms	1.174 ms	<=20ms	2749	明细 主调 被调 调用图

最佳实践–trace调用关系图



05

Negri的未来演进方向

未来规划

1

兼容xds协议

兼容社区xds协议,支持istio作为控制平面

2

支持更多协议,代理更多服务

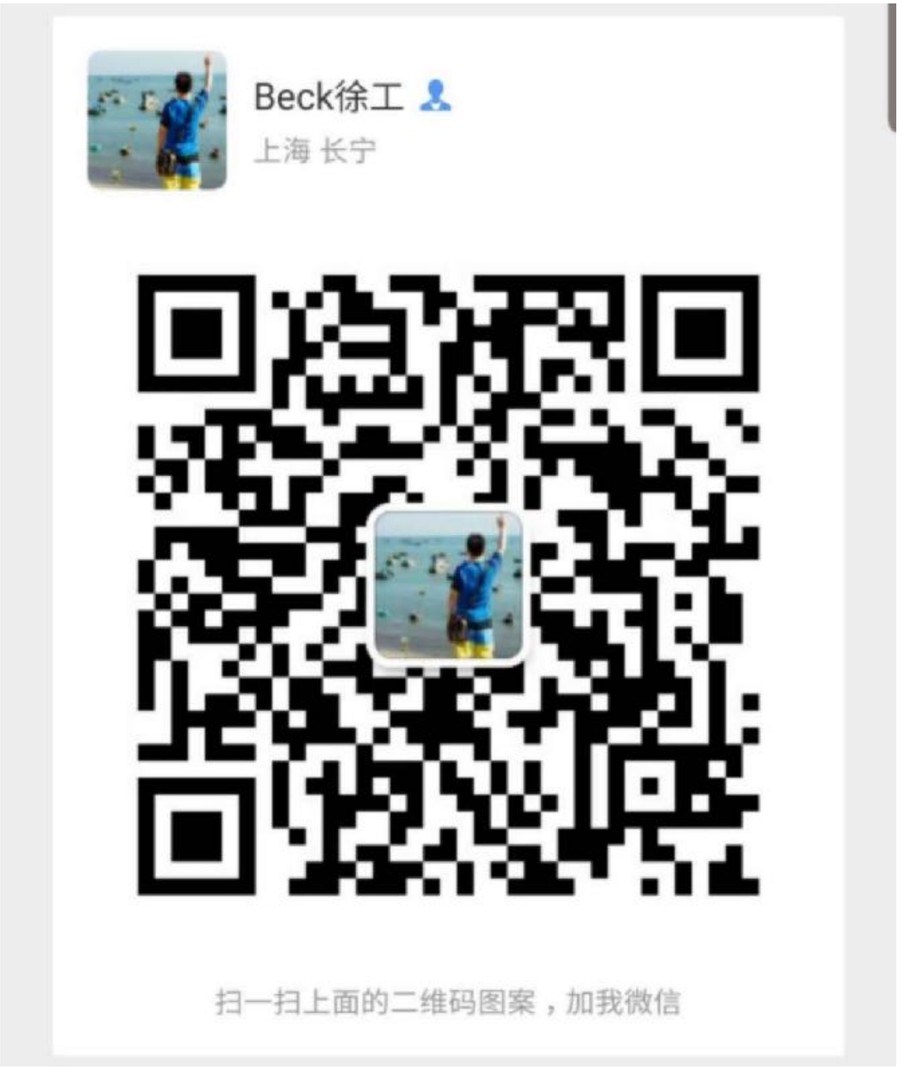
支持redis,mysql,nsq,kafka协议,能够对这些服务做限流熔断,metrics,trace

3

服务授权认证

通过下发服务间调用规则/appsecret,让服务间调用更安全

欢迎加微信交流



THANKS

联合主办：趣头条 Geekbang 极客邦科技