基于 eBPF 的云原生可观测性深度实践

DeepFlow®：eBPF 之上的颠覆性创新，实现高度自动化的可观测性

向阳，研发 VP @ 云杉网络

关于

向阳

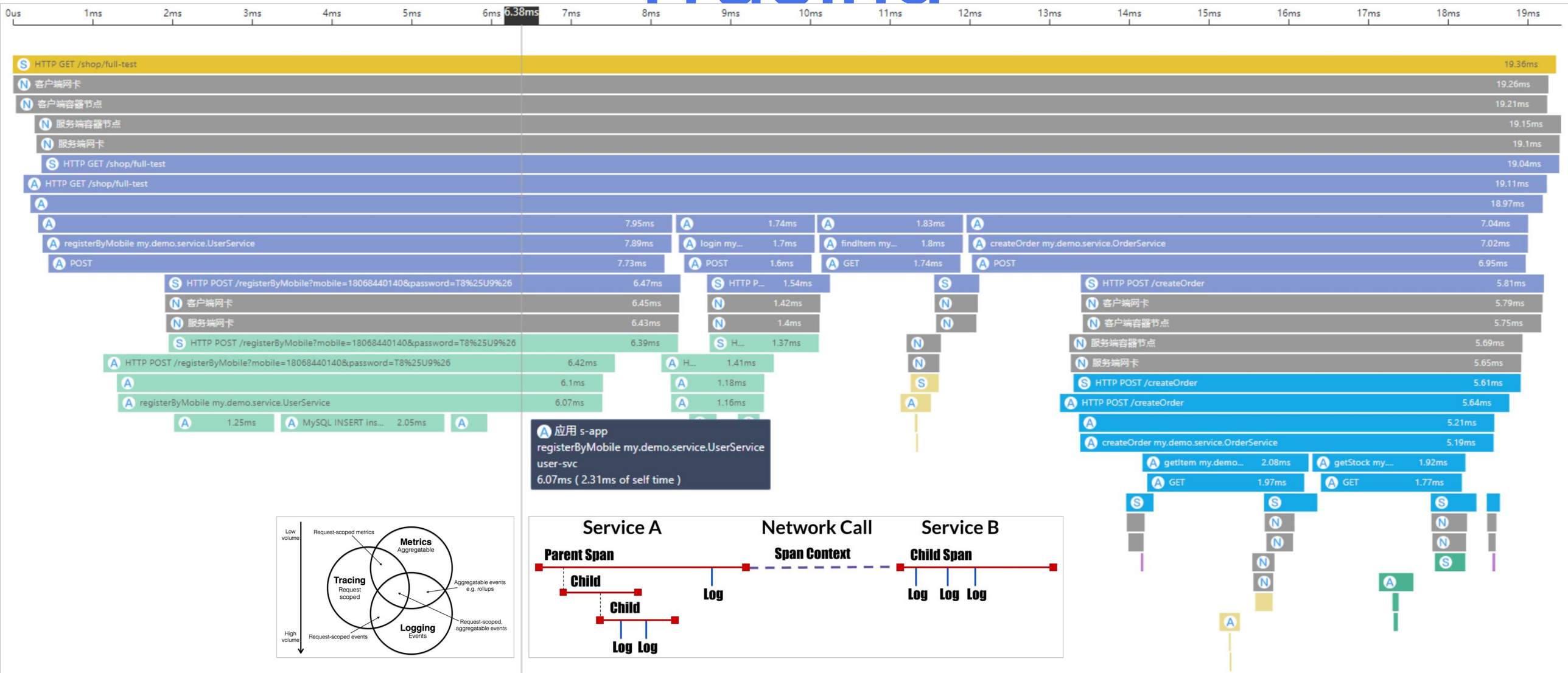
研发 VP @ 云杉

清华大学博士，毕业后加入云杉网络，现负责云原生可观测性产品 DeepFlow。2016 年我们发布了企业版第一个 Release，2022 年开源了 DeepFlow 核心（Apache 2.0），同年 7 月发布了社区版第一个 Release。DeepFlow 致力于让云原生开发者实现高度自动化的可观测性，让观测更自动，让开发者更自由！

目录

目录

分布式追踪 / Distributed Tracing



分布式追踪的十四年

更标准化
更自动化
覆盖更全

2022

2008

COMMUNITY SkyWalking has received contributions from 883 individuals now. >

Apache SkyWalking

Application performance monitor tool for distributed systems, especially designed for microservices, cloud native and container-based (Kubernetes) architectures.

Dapper, a Large-Scale Distributed Systems Tracing Infrastructure

Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson,
Manoj Plakal, Donald Beaver, Saul Jaspan, Chandan Shanbhag

InfoQ

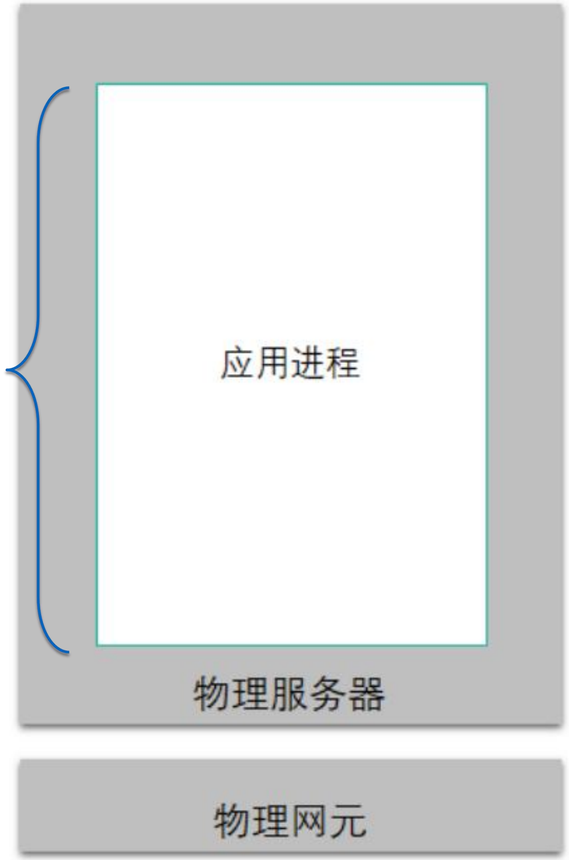
云原生时代的痛：插码插不全

微服务拆分、大型团队协作，有那么几个语言/框架永远也覆盖不全
没有了 Java 字节码增强技术的加持，每一次 SDK 的升级，感觉永远也部署不完

云原生时代的痛：链路追不全

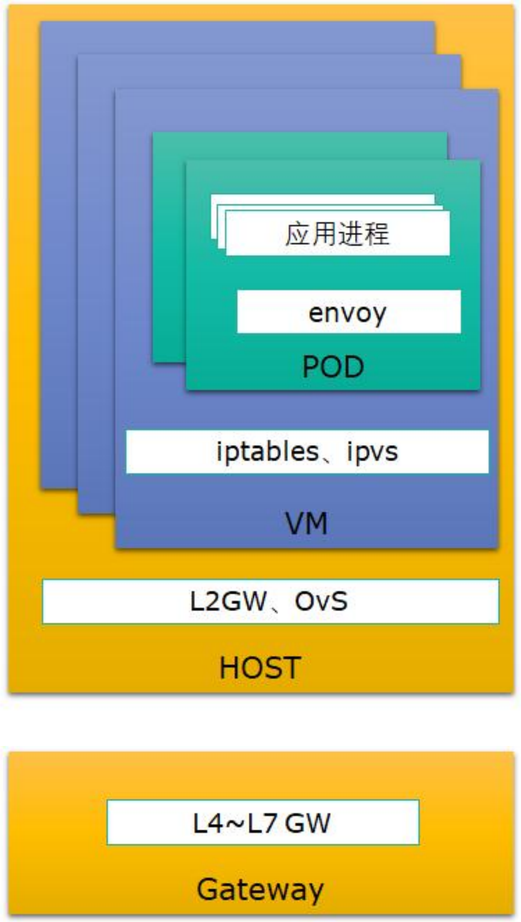
相对稳定的硬件

快速迭代的软件



业务代码 ➔
框架 / 库调用 ➔
系统调用 ➔
服务网格 sidecar ➔
容器网络 iptables/ipvs ➔
虚机网络 ovs/linuxbr ➔
网关、数据库

追踪盲点



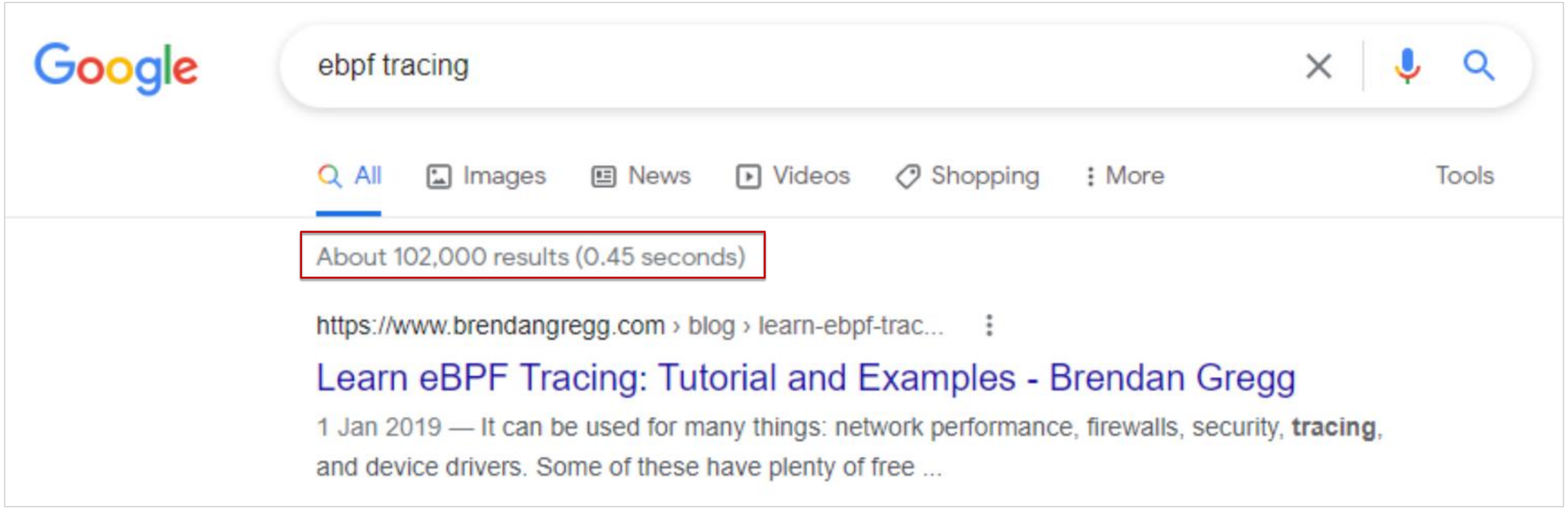
软件定义一切

高速迭代的软件

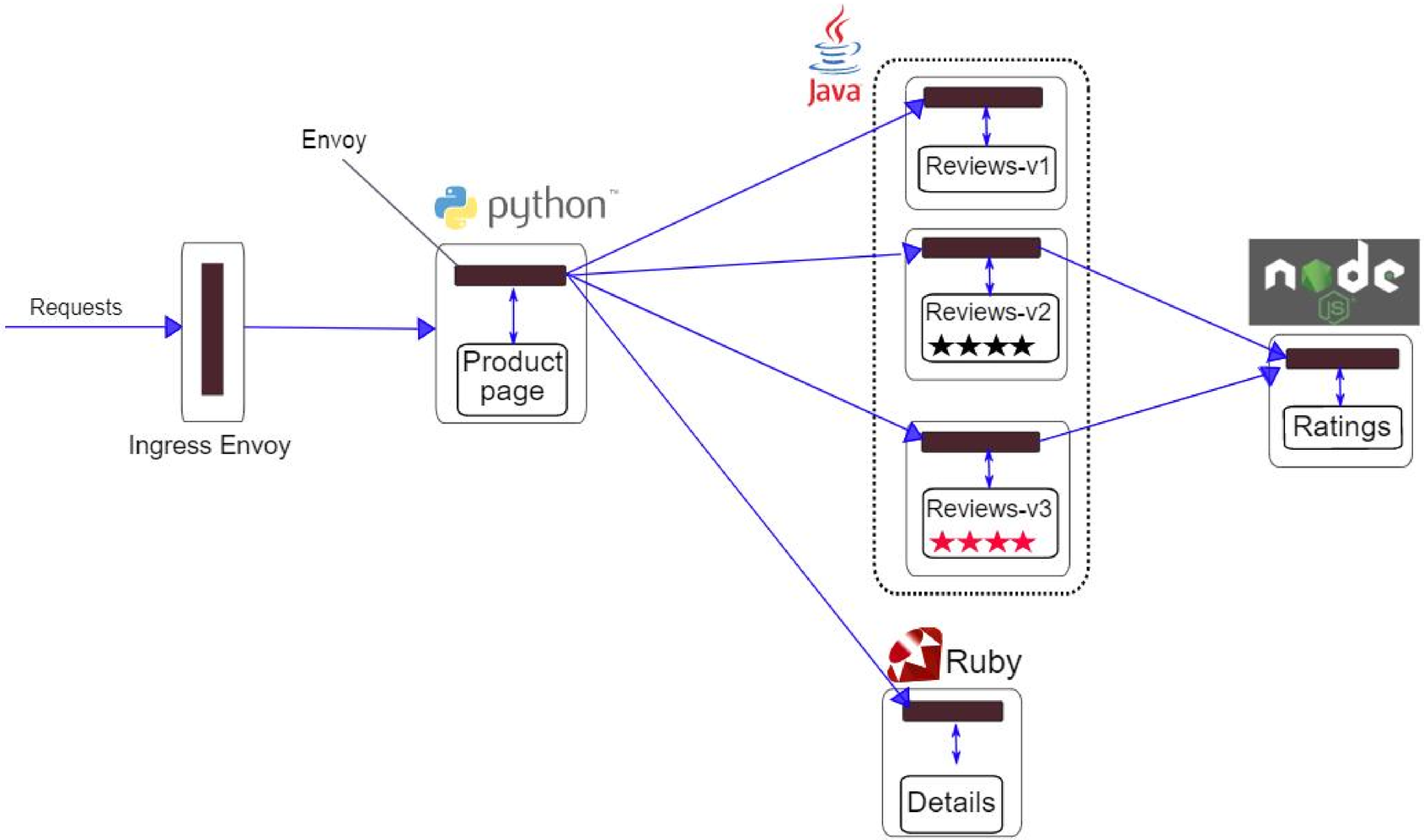
目录

- 分布式追踪：回顾十四年历史，剖析云原生时代的新痛点
- **AutoTracing：DeepFlow 基于 eBPF 之上的颠覆性创新**
- 让追踪无盲点：全栈、全链路，基于创新技术的产品方案
- 展望未来：开源共建，开启高度自动化的可观测性新时代

这会是一项创新吗?


Google
ebpf tracing
All
Images
News
Videos
Shopping
More
Tools
About 102,000 results (0.45 seconds)
https://www.brendangregg.com › blog › learn-ebpf-trac...
Learn eBPF Tracing: Tutorial and Examples - Brendan Gregg
1 Jan 2019 — It can be used for many things: network performance, firewalls, security, tracing, and device drivers. Some of these have plenty of free ...

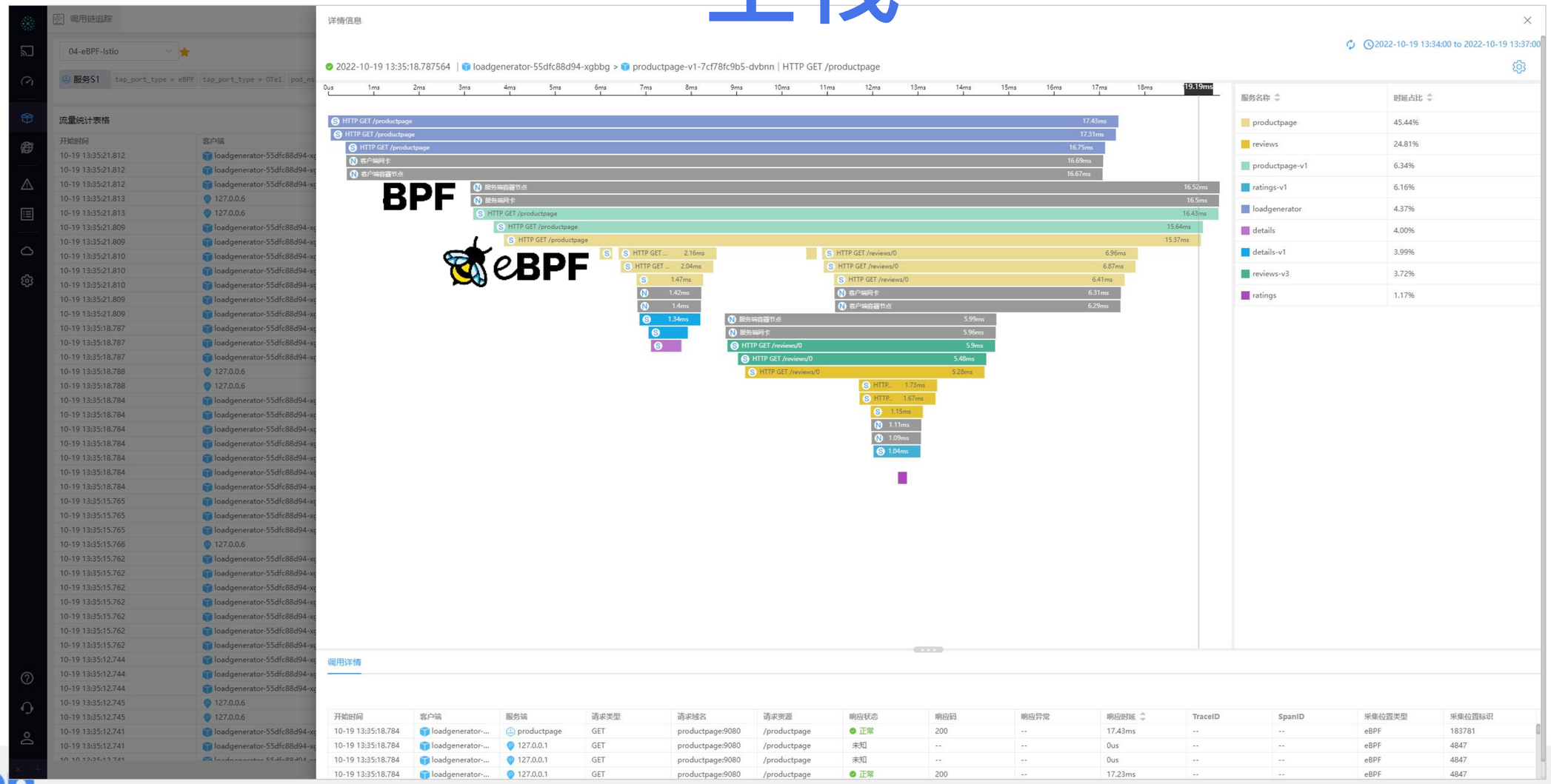
Istio Bookinfo 零插码追踪 Demo



Jaeger 追踪的怎样

- 单击此处添加文本

DeepFlow AutoTracing：零插码、全栈



感受 DeepFlow 的 AutoTracing

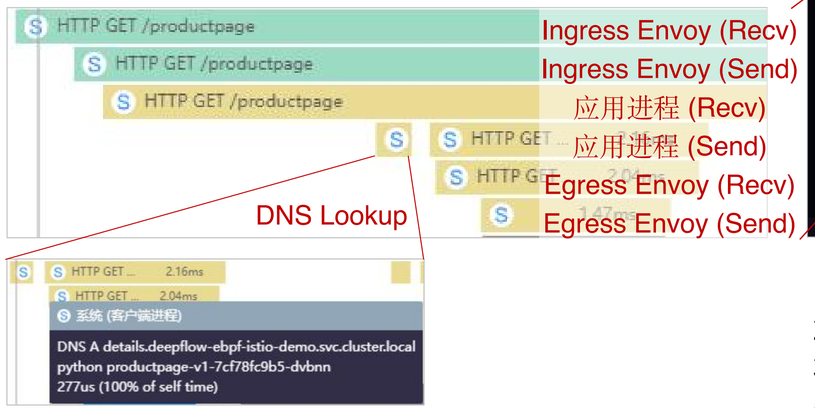
业务开发团队

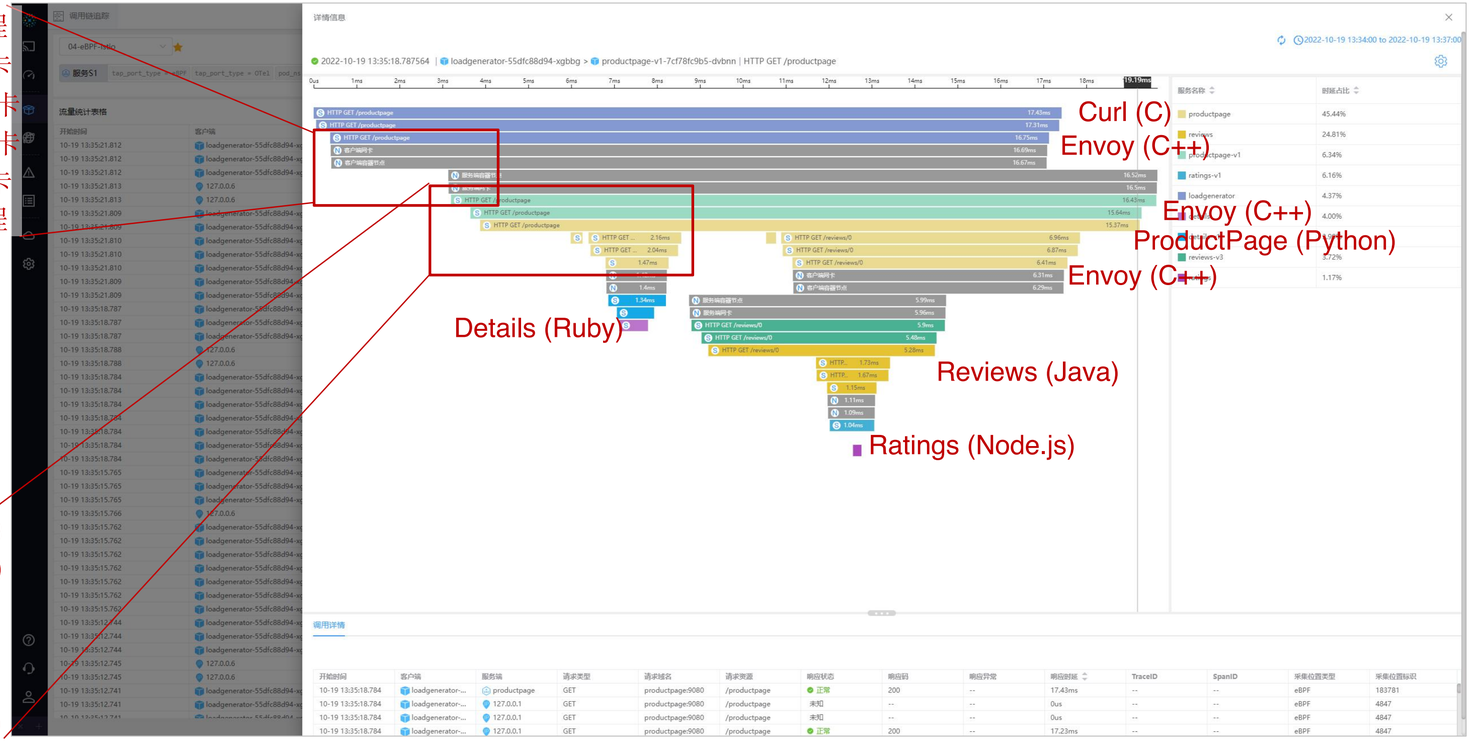
服务网格团队

容器运维团队

云运维团队

无需插码
快速定界

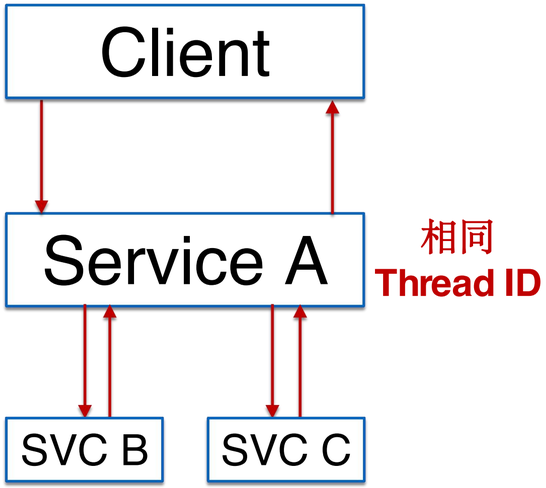




1. 零插码：且无需向 HTTP 头注入 TraceID 或 SpanID
2. 全链路：4 个调用、38 个 Span，分为 24 eBPF Span + 14 BPF Span
3. 多语言：Java、Python、Ruby、Node.js 及 C/C++ (curl/envoy)
4. 全栈：追踪两个微服务之间的网络路径，从 Pod 到 Node 到 KVM，IPIP、VXLAN、…
5. 全栈：追踪微服务内从 Envoy Ingress ➔ 服务 ➔ DNS ➔ Envoy Egress 全过程

案例：某互联网客户，使用 DeepFlow 5 分钟内定位客户端慢服务端不慢的经典扯皮问题。

AutoTracing 背后的关键洞察

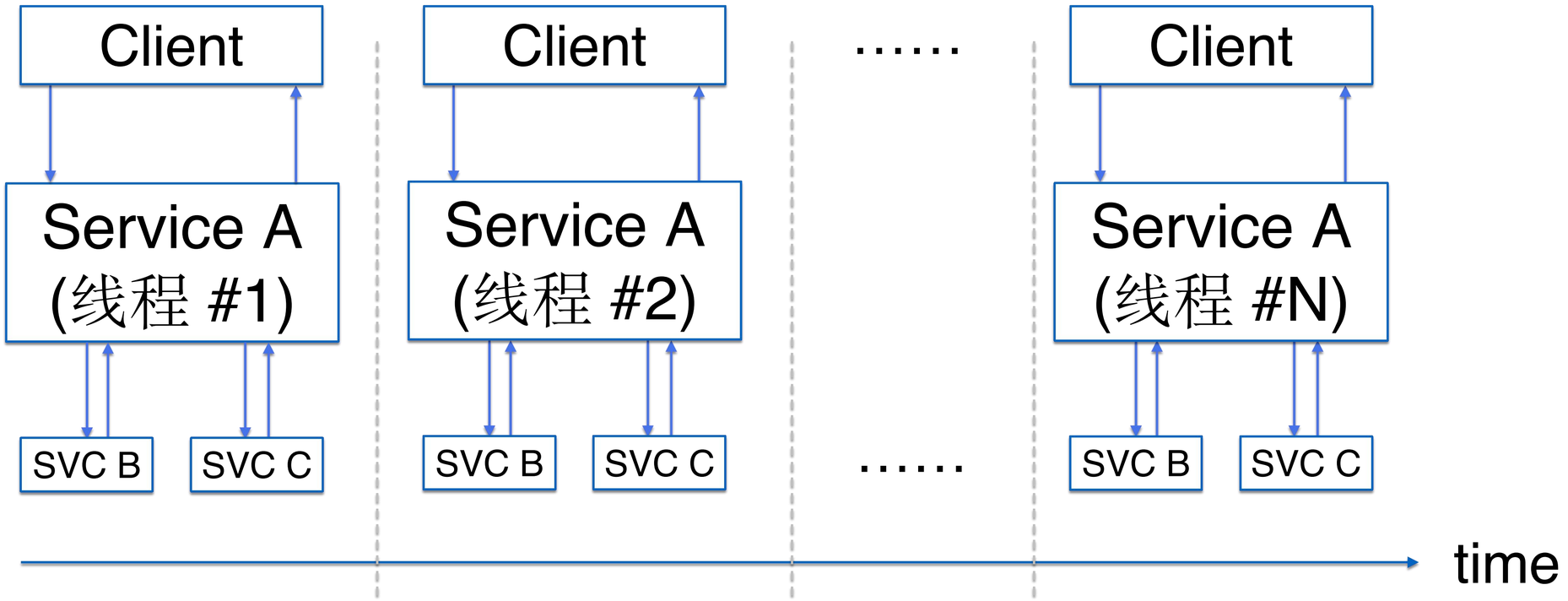


Coroutine
(Erlang, Golang)

Kernel Thread : User Thread = 1 : 1

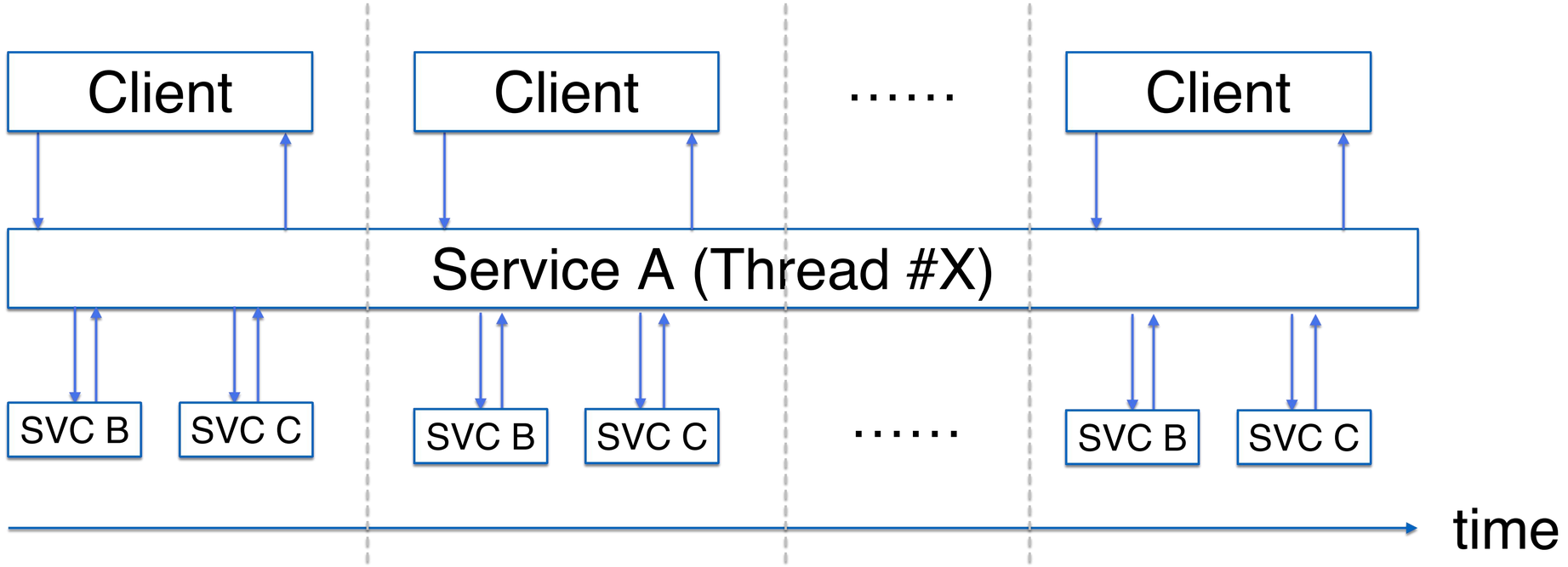
使用 Thread ID 关联同一个 Trace 在一个服务实例上的多个请求

理想情况



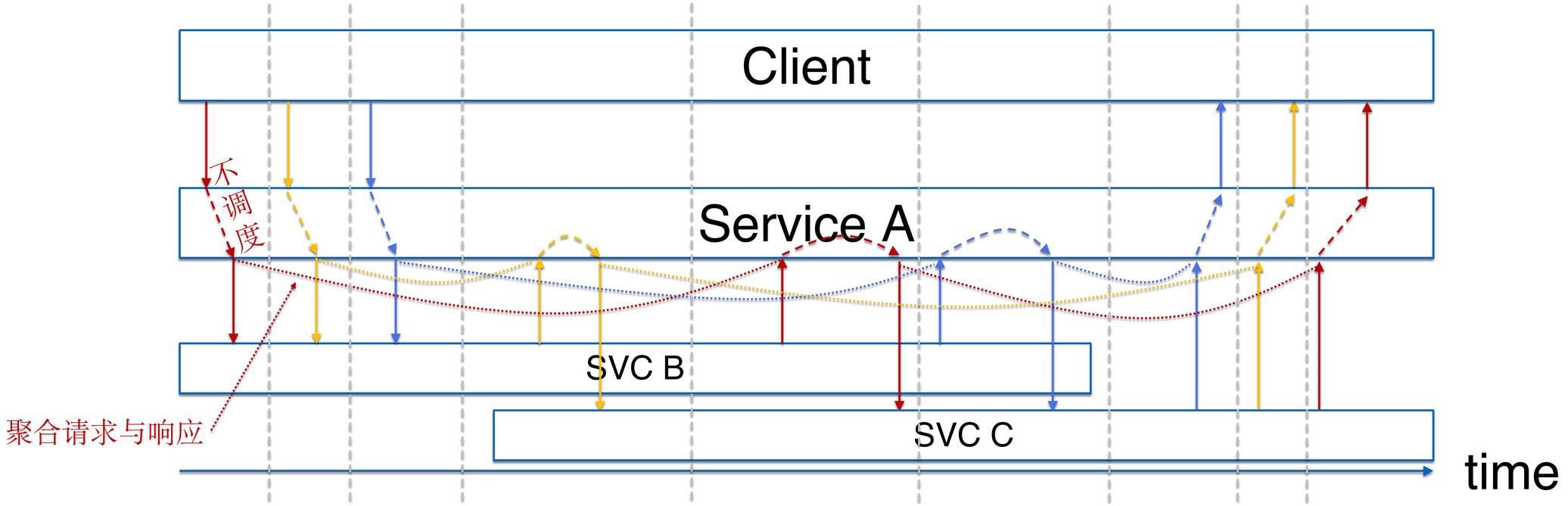
使用 Thread ID 切分 Trace

挑战一：如何处理线程复用



使用时间切分 Trace

挑战二：如何处理非阻塞 IO

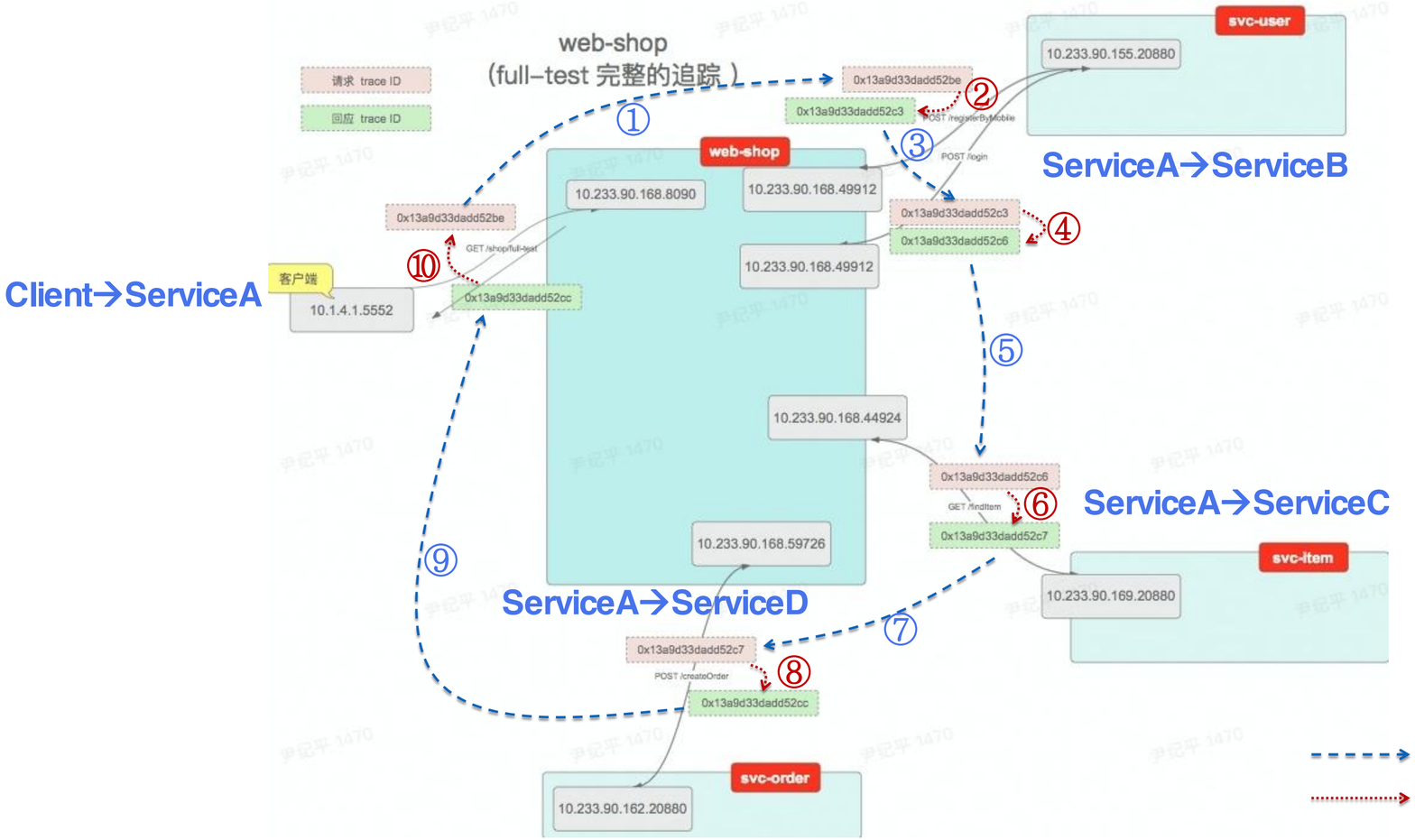


计算不会触发调度，IO 触发调度

利用时序关联“相邻”的两个调用

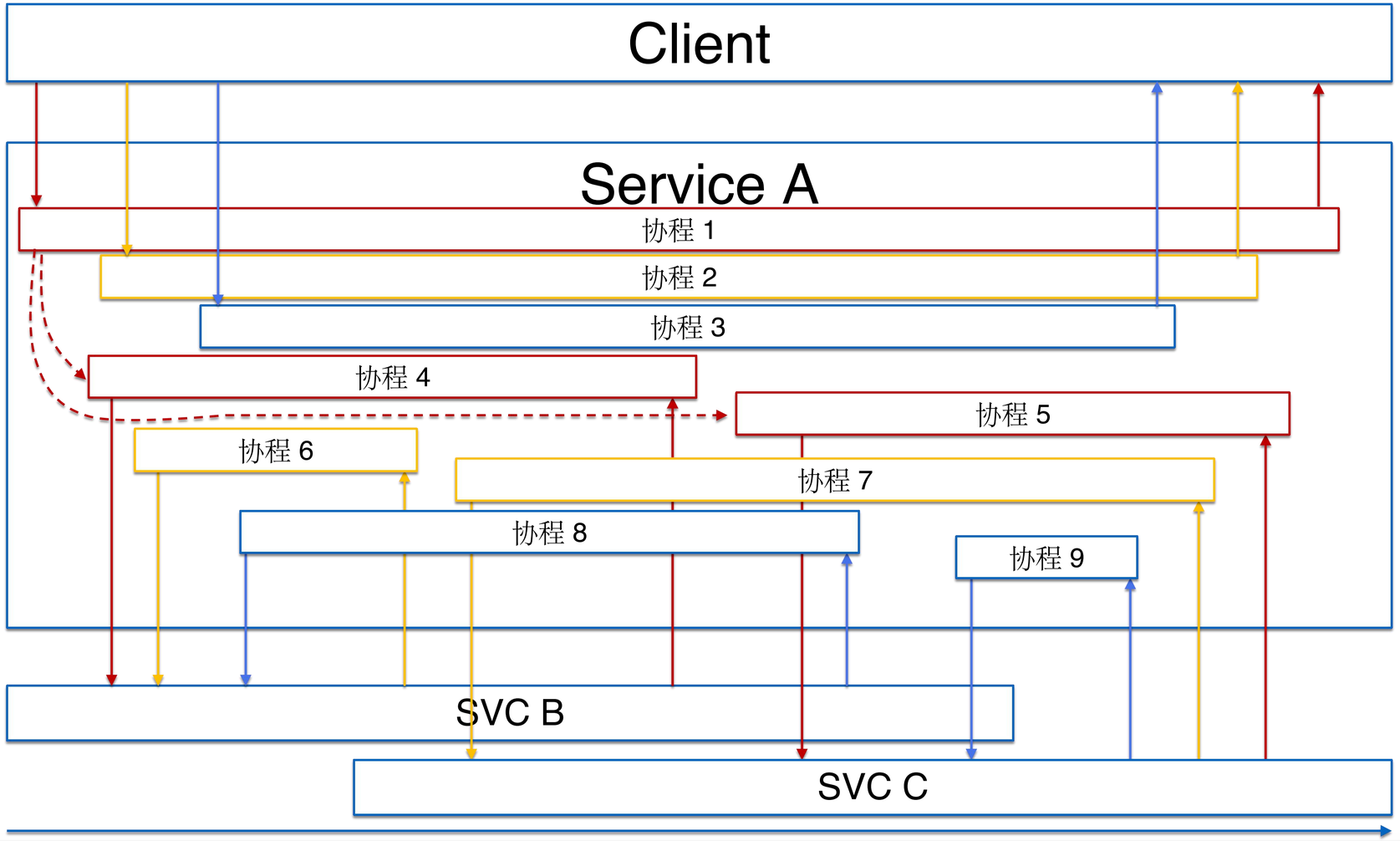
利用流聚合关联同一个调用的请求和响应

举一个实际的例子

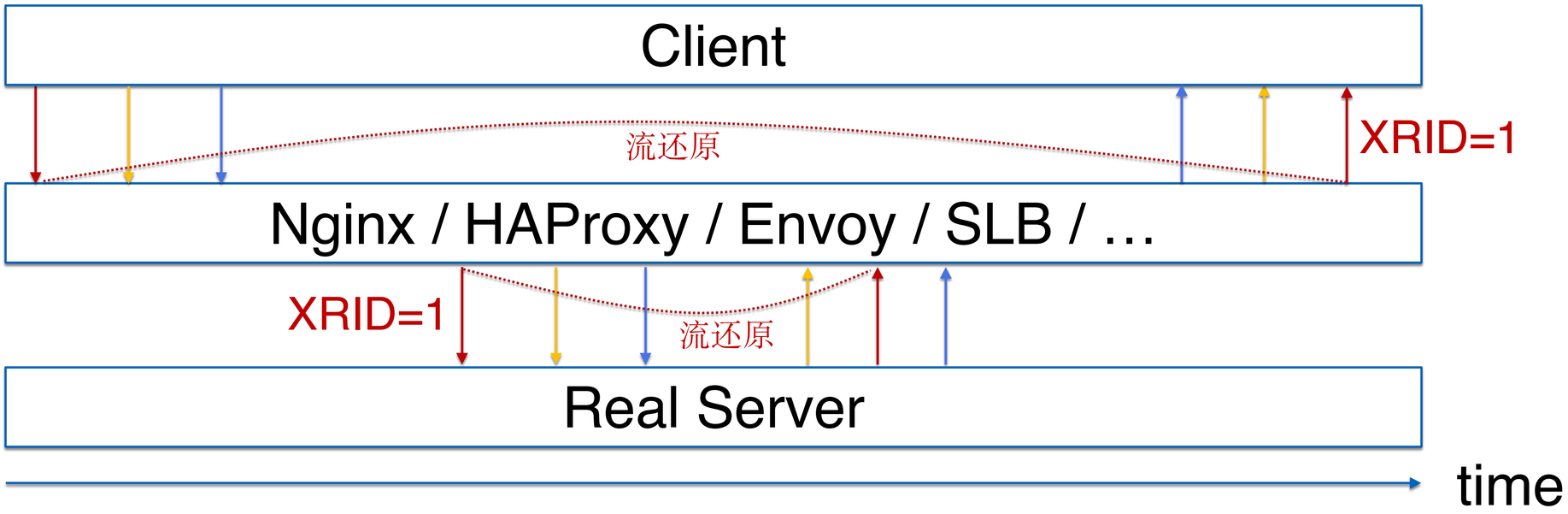

web-shop
(full-test 完整的追踪)
请求 trace ID
回应 trace ID
①
0x13a9d33dadd52be
②
0x13a9d33dadd52c3
POST /registerByMobile
svc-user
10.233.90.155.20880
ServiceA→ServiceB
③
POST /login
web-shop
10.233.90.168.8090
10.233.90.168.49912
0x13a9d33dadd52be
0x13a9d33dadd52c3
④
0x13a9d33dadd52c6
GET /shop/full-test
⑩
客户端
10.1.4.1.5552
0x13a9d33dadd52cc
10.233.90.168.49912
Client→ServiceA
⑤
10.233.90.168.44924
0x13a9d33dadd52c6
⑥
GET /findItem
ServiceA→ServiceC
0x13a9d33dadd52c7
svc-item
10.233.90.169.20880
⑨
10.233.90.168.59726
ServiceA→ServiceD
⑦
0x13a9d33dadd52c7
⑧
POST /createOrder
0x13a9d33dadd52cc
svc-order
10.233.90.162.20880
根据 syscall 时序追踪
根据 session 匹配追踪

挑战三：如何处理跨线程（协程）

协程染色
构造虚拟“线程”



挑战四：如何处理跨线程（队列）



网关生成随机 X-Requst-ID 并注入
到 Real Server 的请求，到 Client 的响应

挑战五：如何追踪一个调用

TCP.SEQ

业务代码 ➔

框架 / 库调用 ➔

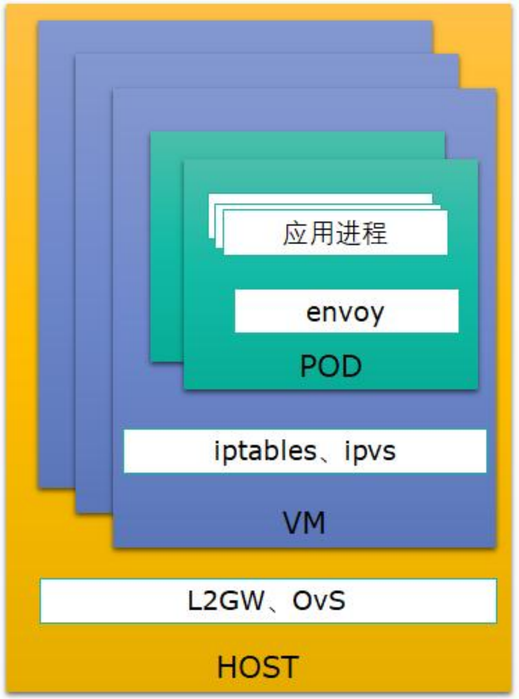
系统调用 ➔

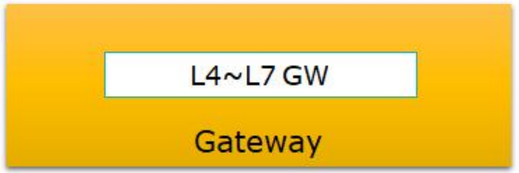
服务网格 sidecar ➔

容器网络 iptables/ipvs ➔

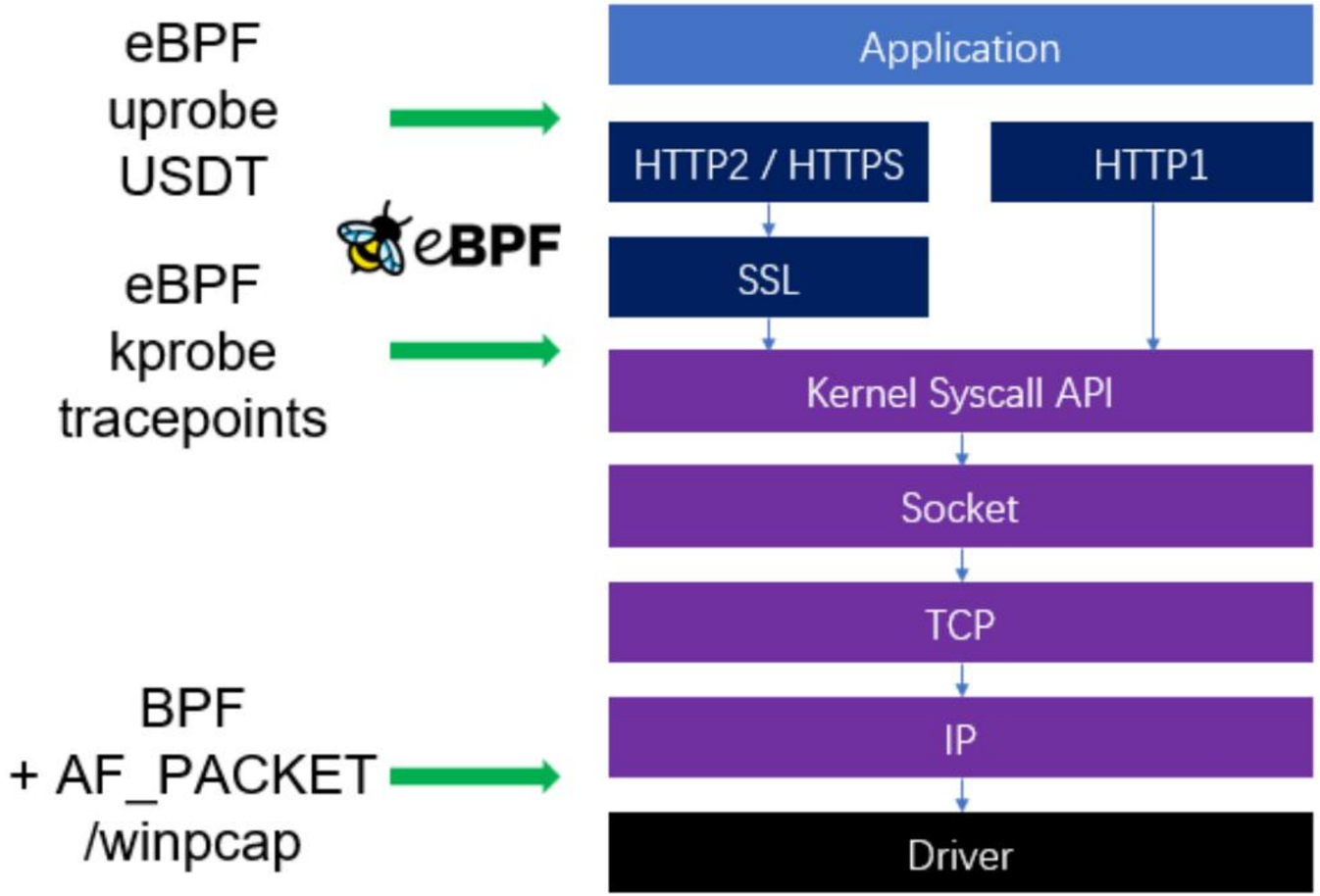
虚机网络 ovs/linuxbr ➔

网关、数据库



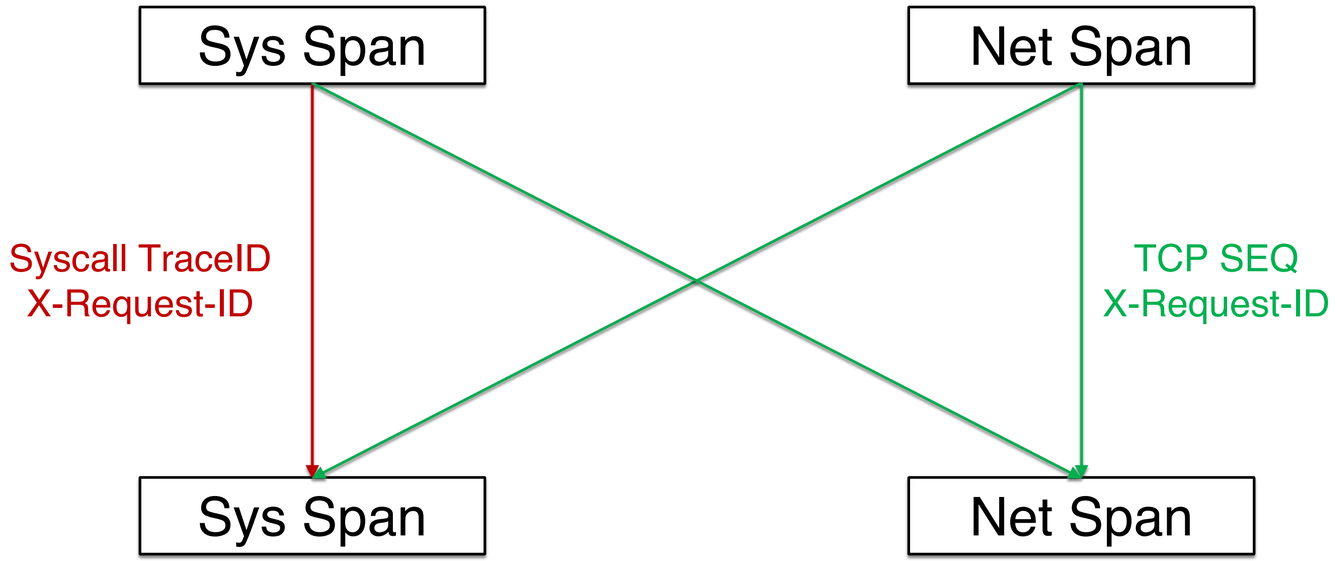


挑战六：如何追踪 HTTPS 调用



挑战七：如何查询 Trace

for {



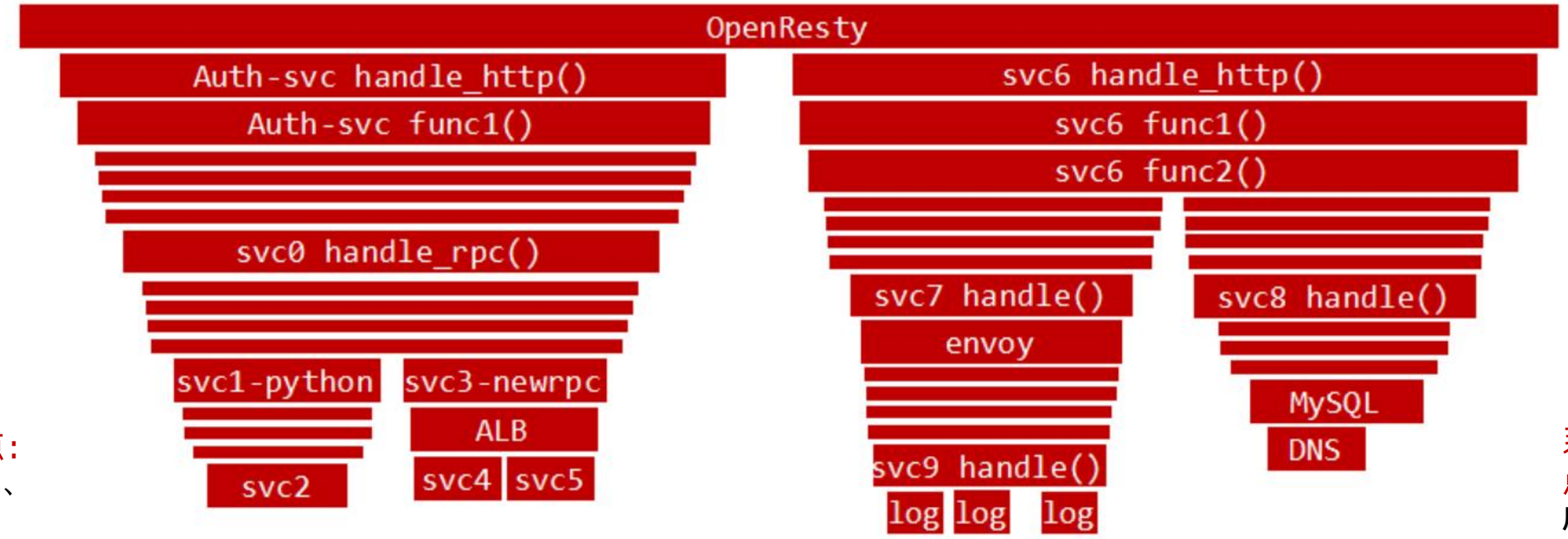
}

Span 里没有 TraceID、SpanID

目录

- 分布式追踪： 回顾十四年历史，剖析云原生时代的新痛点
- AutoTracing： DeepFlow 基于 eBPF 之上的颠覆性创新
- **让追踪无盲点：全栈、全链路，基于创新技术的产品方案**
- 展望未来： 开源共建，开启高度自动化的可观测性新时代

让追踪无盲点



业务代码 Span 盲点：

基础设施服务：APIGW、NLB/ALB、KVM/K8s、DB、DNS、Log、…

插码不完全：新语言、新框架、不同团队方案不一

Tracing without ~~blind spots~~

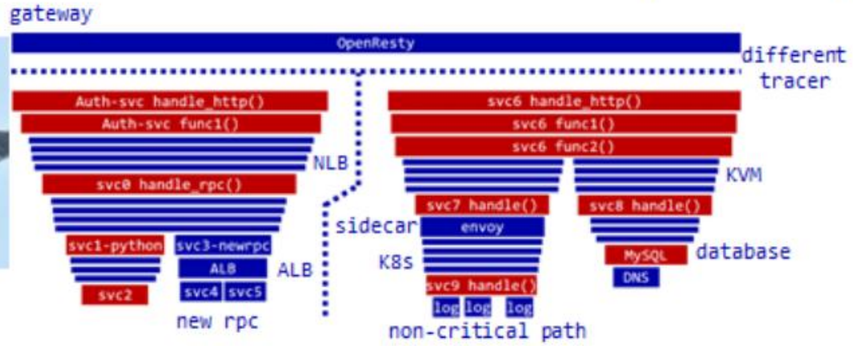
Auto Intrumentation by OpenTelemetry + AutoTracing by eBPF

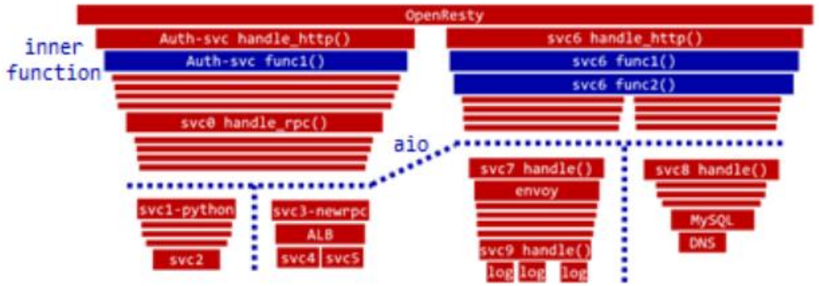
系统和网络 Span 盲点：

应用内部：进程内函数之间的调用链

业务逻辑：部分跨线程请求场景、异步 IO 场景

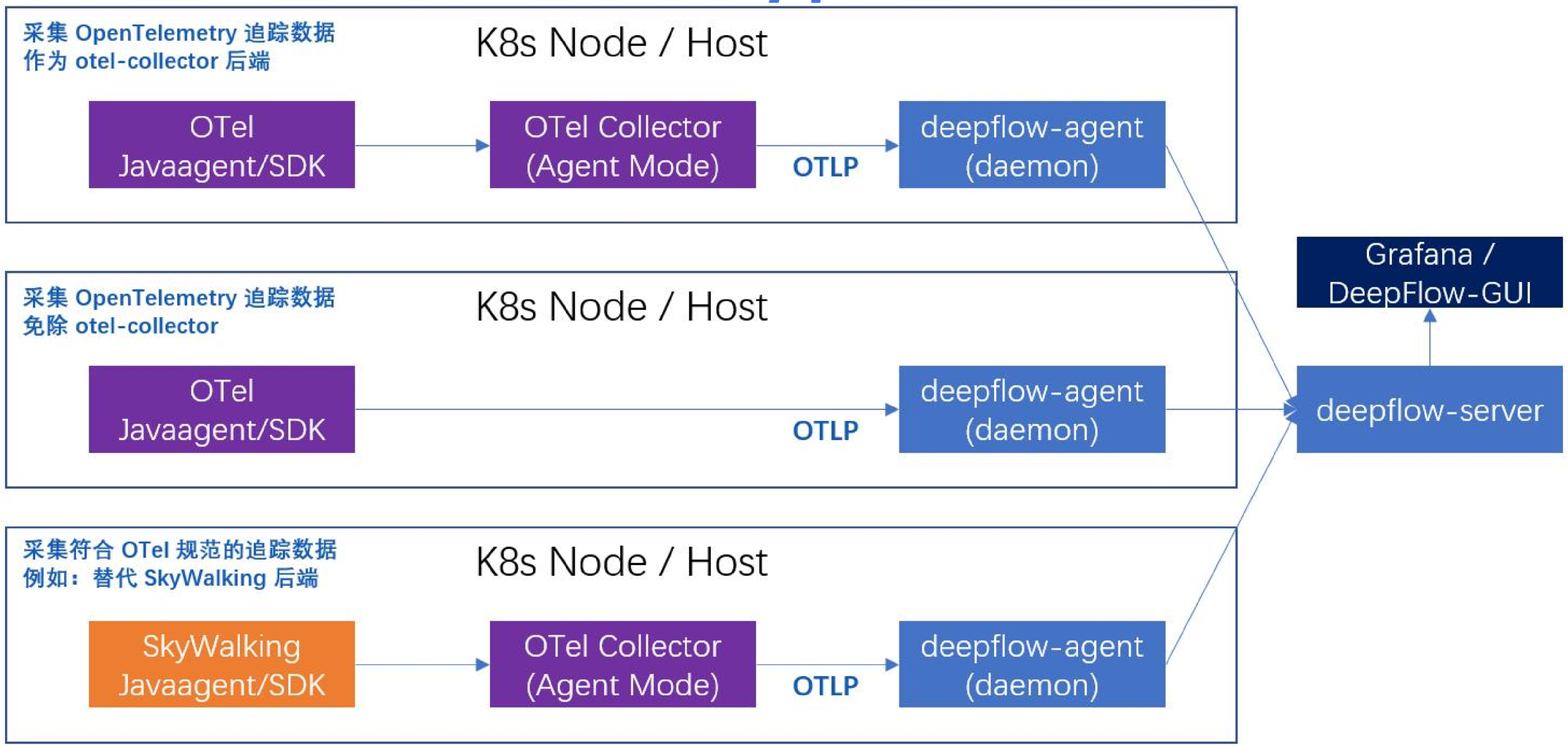
“火控雷达”：手动插码



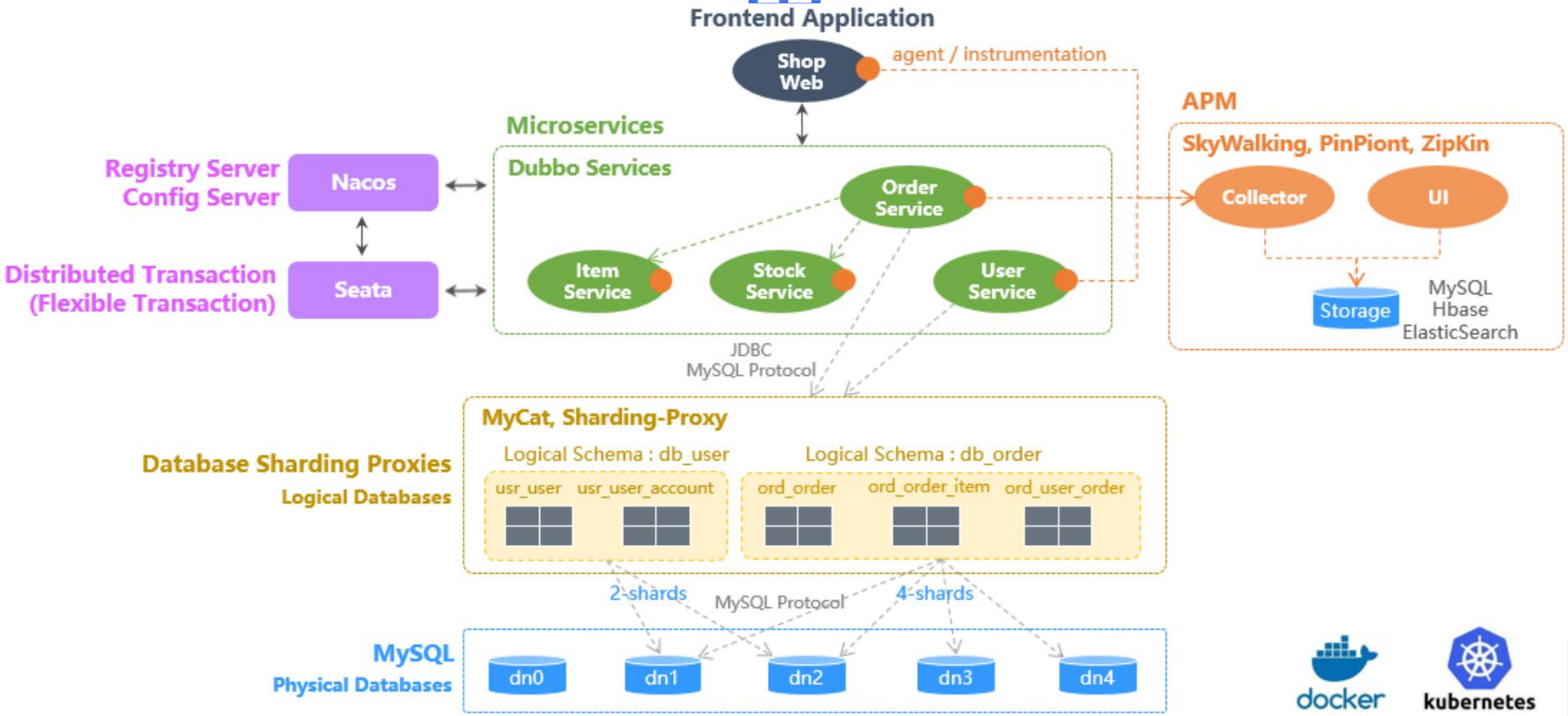


“预警雷达”：自动追踪

DeepFlow 的追踪数据集成、关联能力



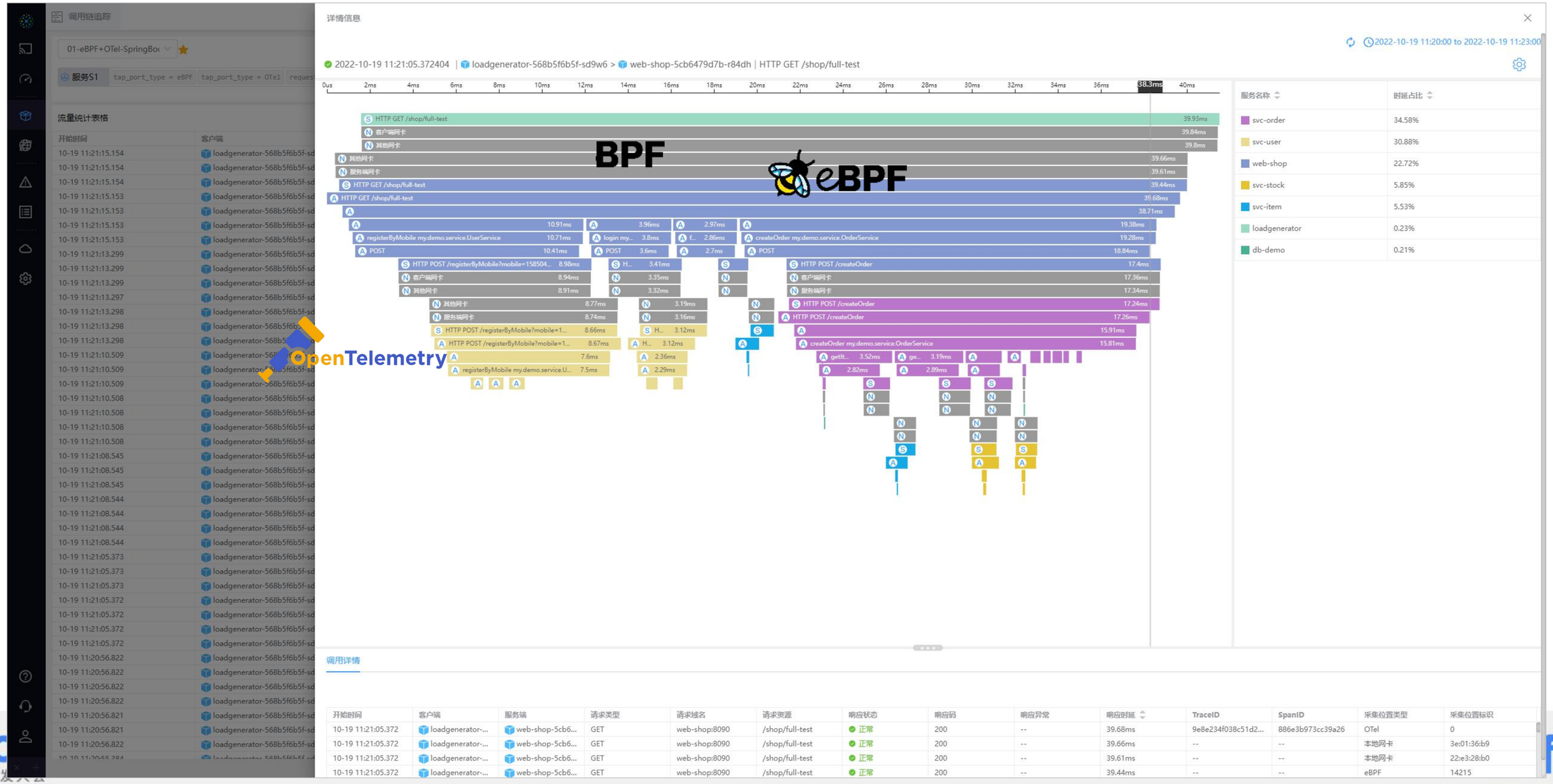
无盲点追踪一个 Spring Boot 应用



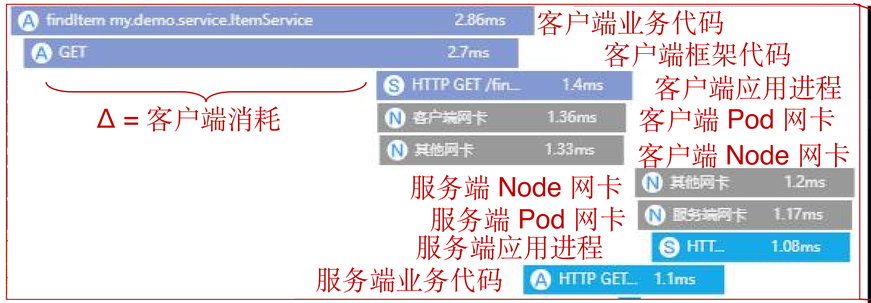
OpenTelemetry + Jaeger 追踪结果如何

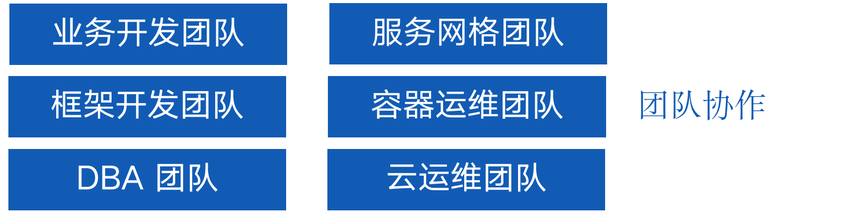
46 Spans

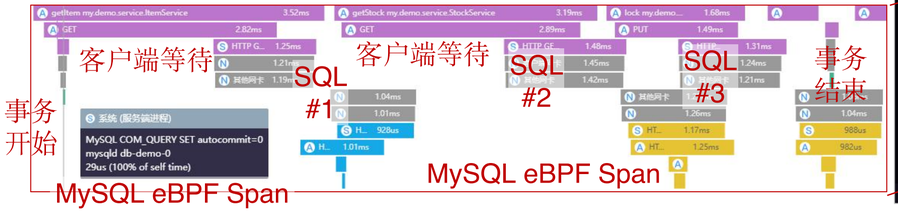
DeepFlow 无盲点追踪

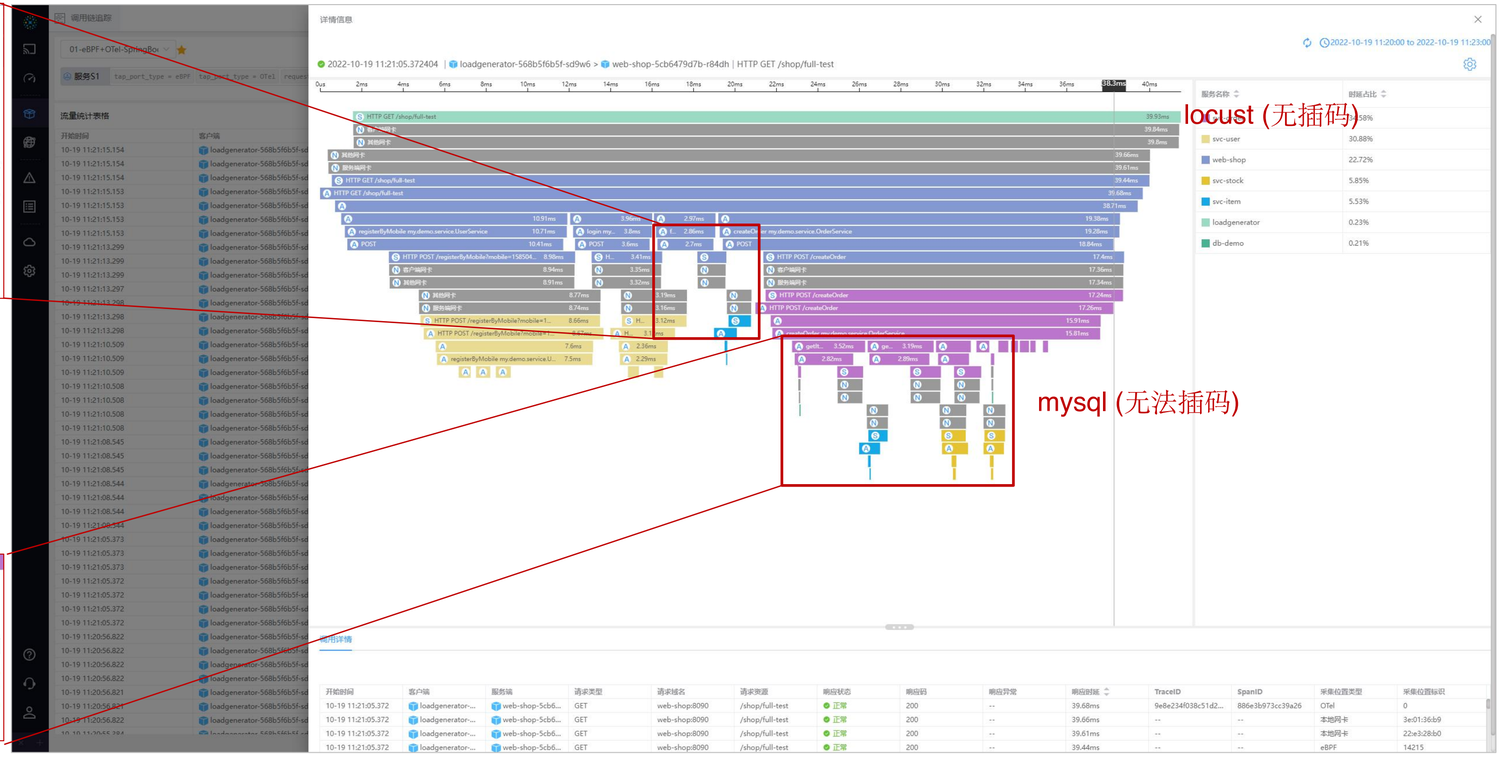


感受 DeepFlow 无盲点追踪







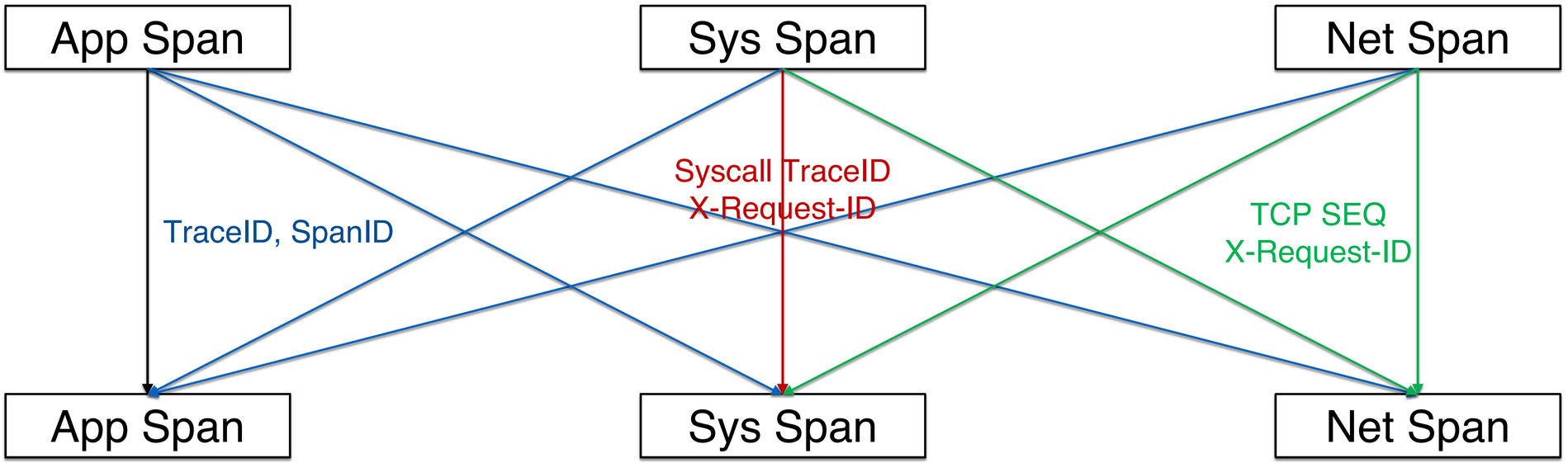


1. 全链路：追踪到了 96 个 Span：46 个 OTel Span、20 个 eBPF Span、30 个 BPF Span
2. 全栈：支持追踪两个微服务之间的网络路径，即使有隧道
3. 全链路：对 OTel 无插码的服务（loadgenerator，C），通过 eBPF 自动追踪补齐
4. 全链路：对 OTel 无法插码的服务（MySQL，autocommit），通过 eBPF 自动追踪补齐
5. 无盲点：eBPF 和 BPF Span 穿插在 OTel Span 之间，让追踪无盲点

案例：某互联网客户，使用 DeepFlow 5 分钟内定位应用慢 DBA 找不到慢日志的经典扯皮问题。

如何查询完整的 Trace

for {

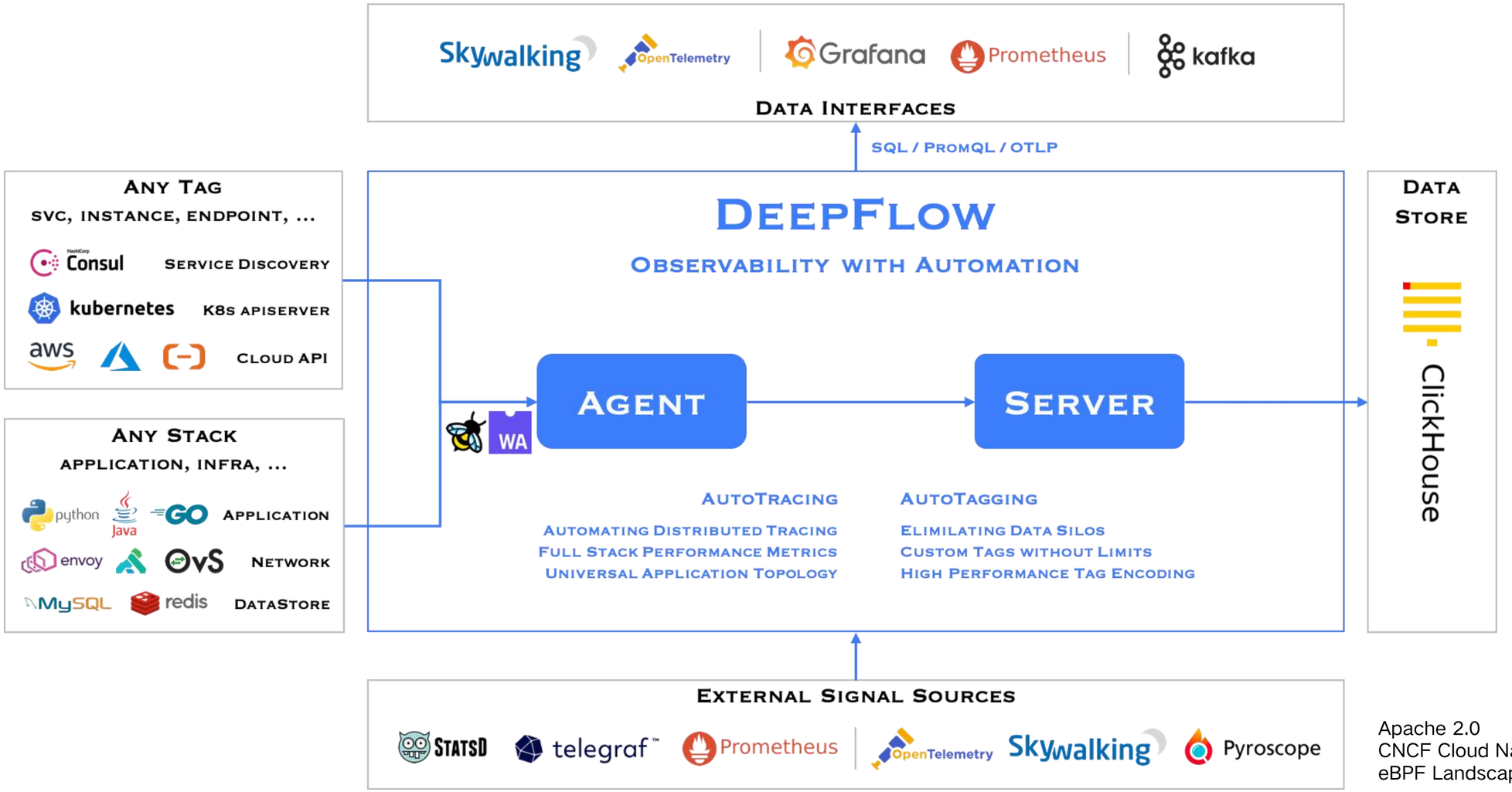


}

目录

- 分布式追踪：回顾十四年历史，剖析云原生时代的新痛点
- AutoTracing：DeepFlow 基于 eBPF 之上的颠覆性创新
- 让追踪无盲点：全栈、全链路，基于创新技术的产品方案
- **展望未来：开源共建，开启高度自动化的可观测性新时代**

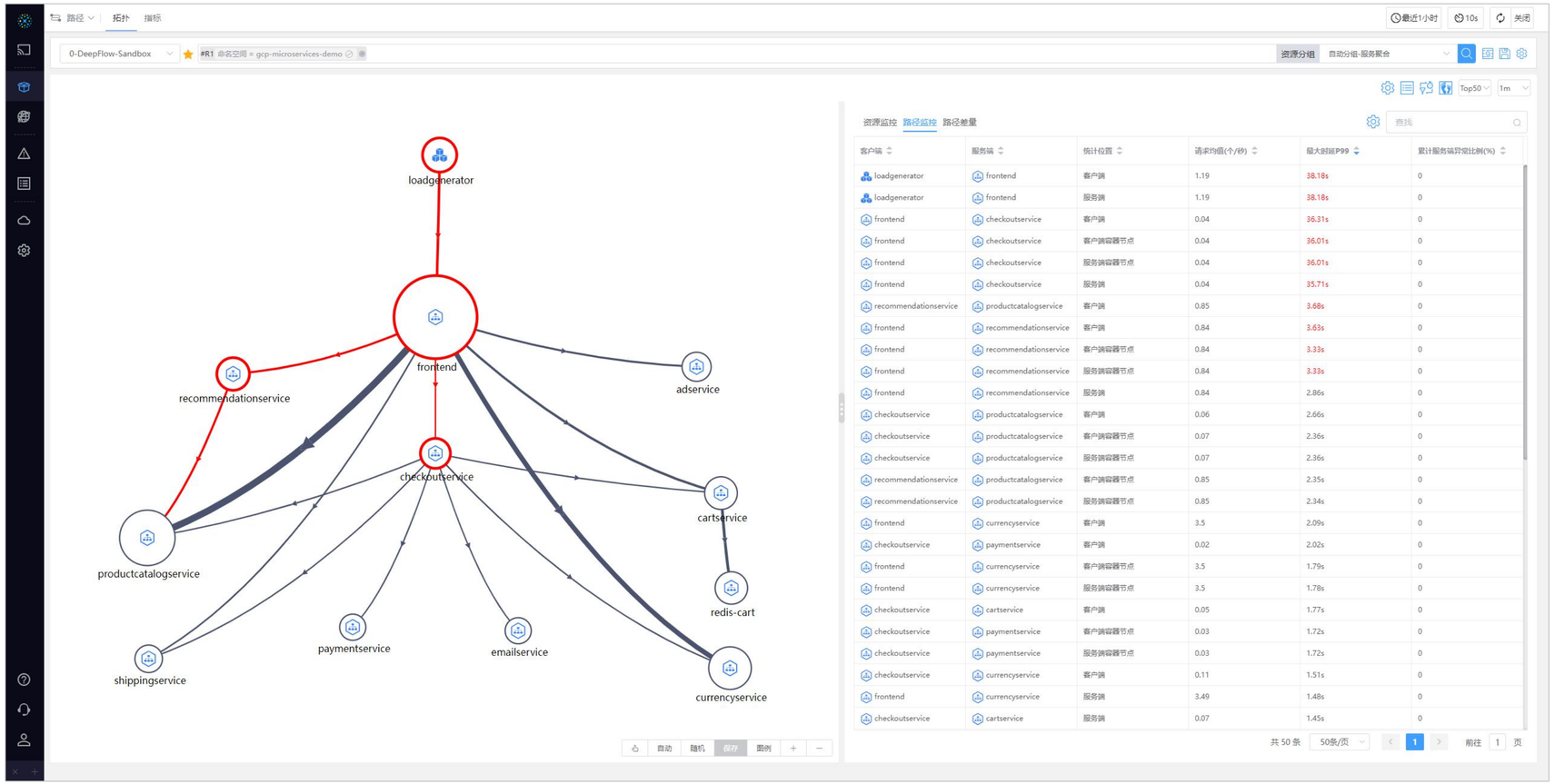
DeepFlow：高度自动化的可观测性平台



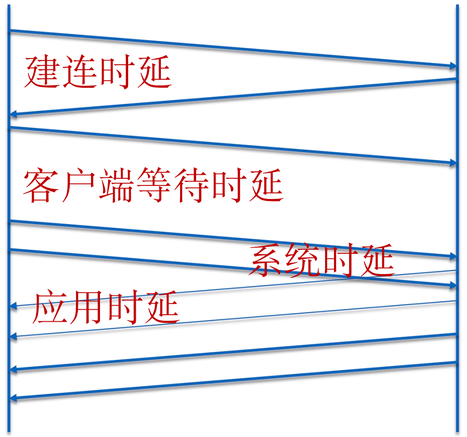
github.com/deepflowys/deepflow

Apache 2.0
CNCF Cloud Native Landscape Project
eBPF Landscape Project

全景应用拓扑



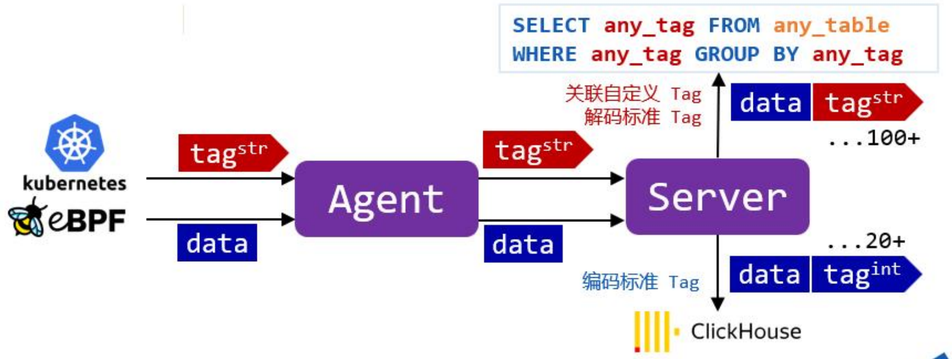
Linux Kernel 2.6+
Pod / 进程粒度的应用访问拓扑
穿透 L4 网关、还原 NAT
关联资源、服务、业务自定义 Tag
关联应用、系统、网络全栈指标



磁盘 IO、重传、…

不插码，你真的知道谁在访问你吗？
某互联网客户，使用 DeepFlow 5 分钟内从数万个 Pod 中定位 RDS 访问量最大的Pod、服务、团队。
即便插码，你真的知道你在访问谁吗？
某银行客户，信用卡核心业务上线受阻，使用 DeepFlow 5 分钟内发现两个服务之间 API 网关是性能瓶颈，检查配置后发现缓存设置不合理。

AutoTagging + SmartEncoding
高性能的数据标签自动注入机制

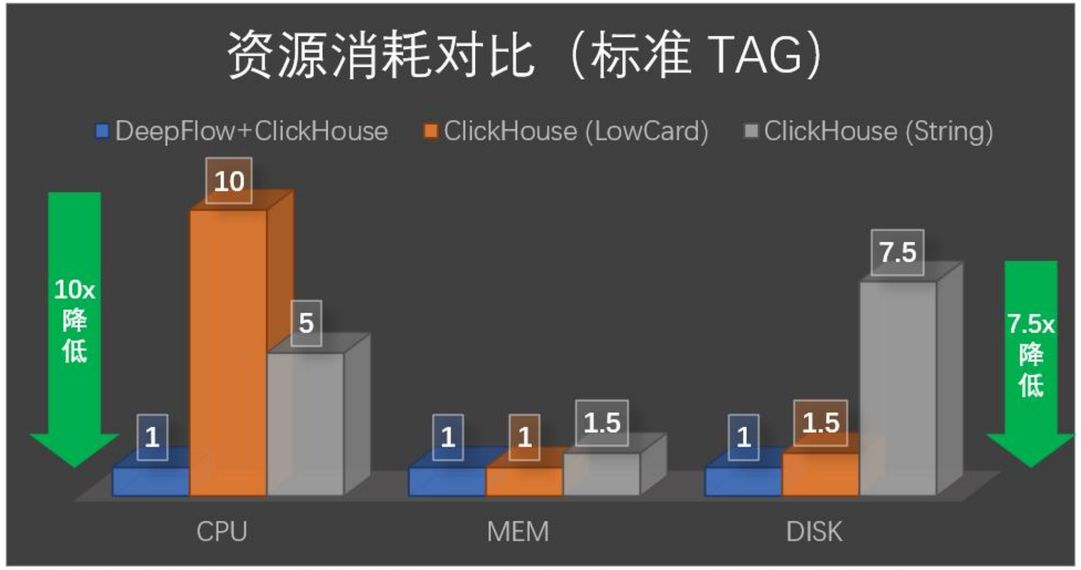


标准 Tag：开销 10x 降低

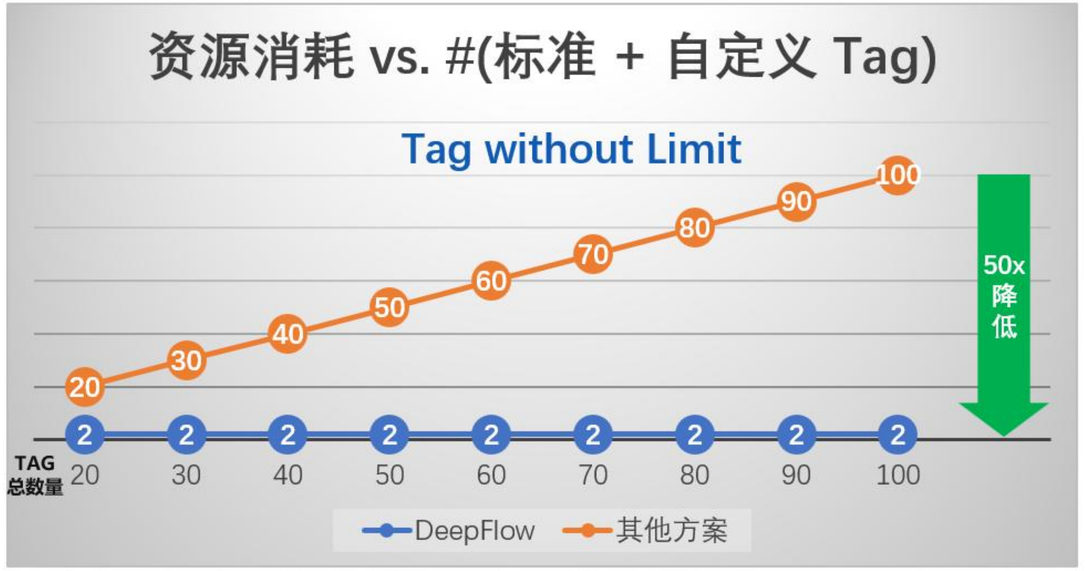
资源池	网络资源	容器资源	Application
区域	VPC	容器集群	ServiceName
可用区	子网	容器节点	FunctionName
云平台	CIDR	命名空间	Endpoint
租户	IP地址	容器服务	TraceId
云资源	NATGW	Ingress	SpanId
宿主机	ALB	Workload	RequestId
云服务器	...	POD	...

自定义 Tag：零开销

K8s labels	Annotations *
app	biz/terminalType
version	cicd/deploymentId
env	...
owner	**OS ENV ***
stage	MODULE_NAME
commitId	...
...	



随机生成一组长度为 16 字符串标签，Cardinality 为 5000，持续极限速率写入。



假设标准 TAG 固定 20 个，自定义 TAG 从 0 个增长到 80 个。

DeepFlow 的演进

跨线程
AutoTracing

文件读写
…

RPC、MQ
DB、…

WASM
LUA

……

THANKS

软件正在重新定义世界

Software Is Redefining The World

DeepFlow 开源社区微信群

DeepFlow®：eBPF 之上的颠覆性创新，实现高度自动化的可观测性

AutoTracing、Universal Application Topology