开源社
kaiyuanshe

基于eBPF构建无侵入的可观测系统

张海彬　　阿里云

张海彬

花名古琦，阿里云技术专家

负责阿里云应用监控eBPF版的产品研发工作，有多年的微服务治理、可观测实践经验，目前在阿里云云原生应用平台负责eBPF的应用监控工作。

目录

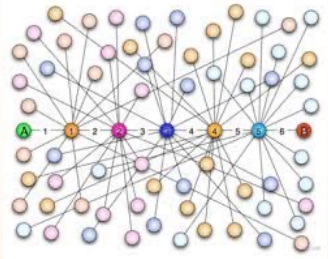
CONTENTS

开源社
kaiyuanshe

Part 01

微服务应用可观测的挑战

微服务应用可观测的挑战



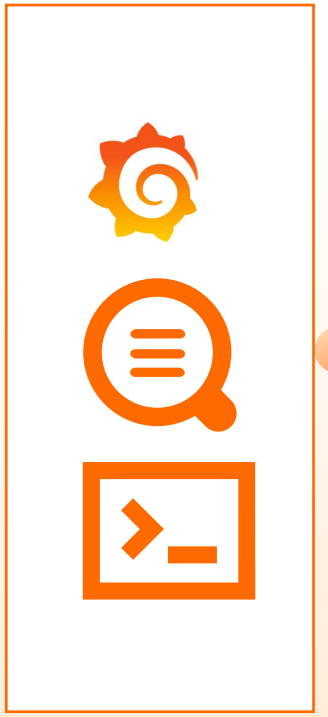
应用：微服务架构、多语言、多协议

挑战1：微服务、多语言、多协议环境下，端到端观测复杂度上升，埋点成本居高不下

- Kubernetes
- 容器
- 网络、操作系统、硬件

基础设施层复杂度日益增加

如何关联?



挑战3：数据散落，工具多，缺少上下文，排查效率低下

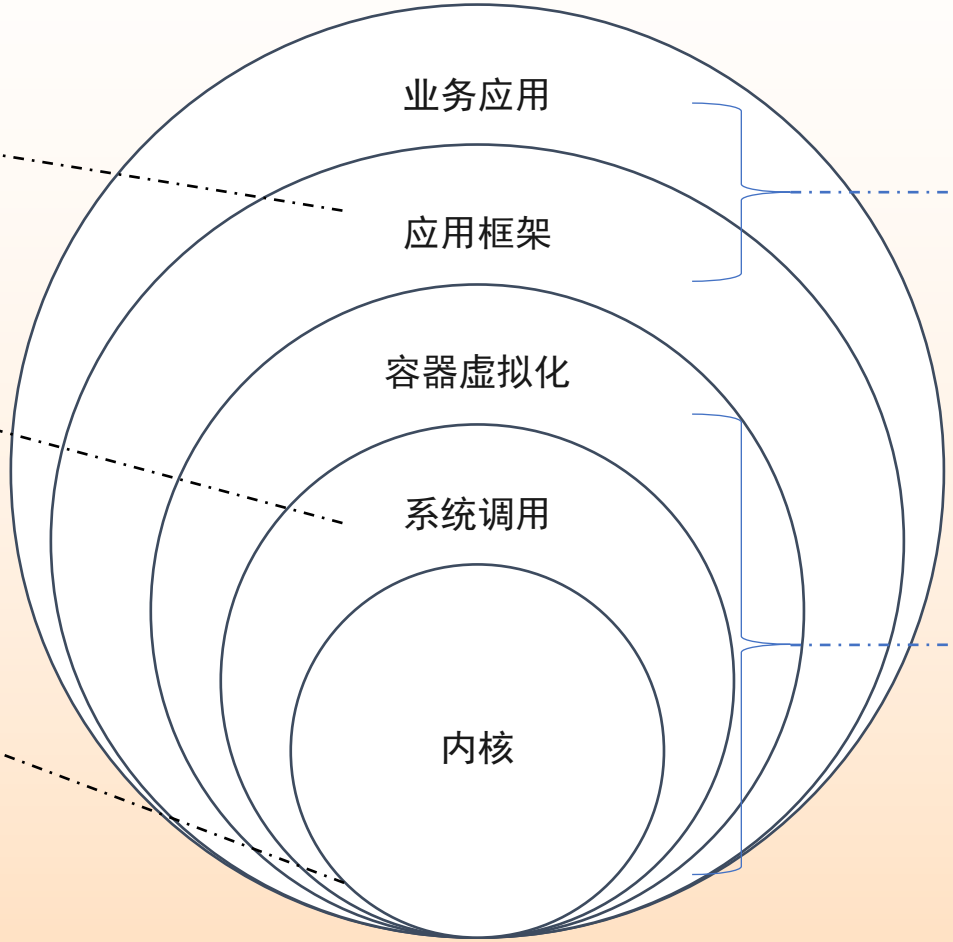
Kubernetes下的可观测

开源社 kaiyuanshe

Kubernetes组件异常：Scheduler，KCM，etcd，api-server，coredns…

系统调用异常：网络请求，内存申请，文件操作，CGroup…

内核异常：进程调度，内存管理，文件管理，夯机宕机，资源异常…



应用组件异常：线程池满，数据库连接无法获取，OOM，文件读取错误…

应用性能监控（APM）

Kubernetes监控

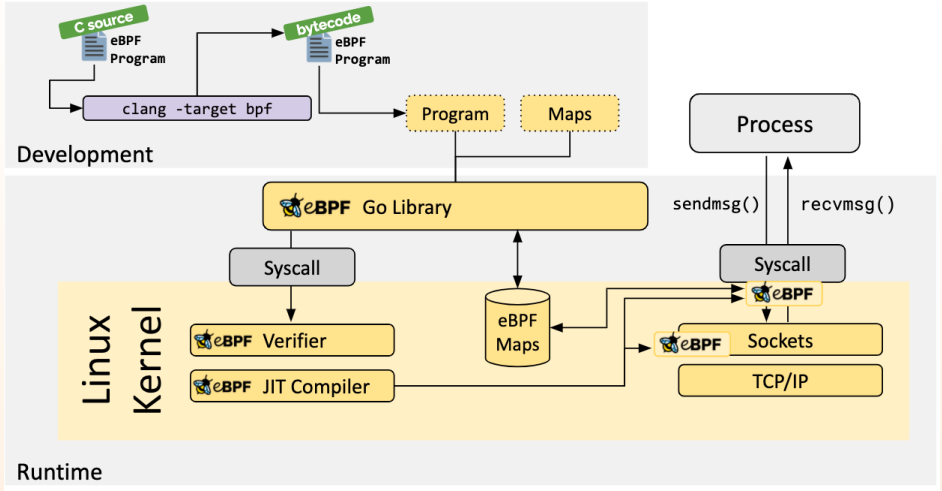
无法自顶向下端到端串联导致棘手问题频发。

开源社
kaiyuanshe

Part 02

使用eBPF进行监控数据采集

使用eBPF进行监控数据采集



数据处理流程

- 加载、编译BPF程序
- 监听内核事件
- 识别网络协议
- 聚合指标、输出Trace

extend Berkeley Packet Filter，在Linux 内核中运行沙盒程序，而无需更改任何源代码或加载任何内核模块，有以下特性：

- 无侵入：成本极低。应用不需要改任何代码。
- 动态可编程：不需要重启探针，动态下发采集脚本
- 高性能：自带JIT编译成native代码
- 安全：verifier机制保障内核运行稳定

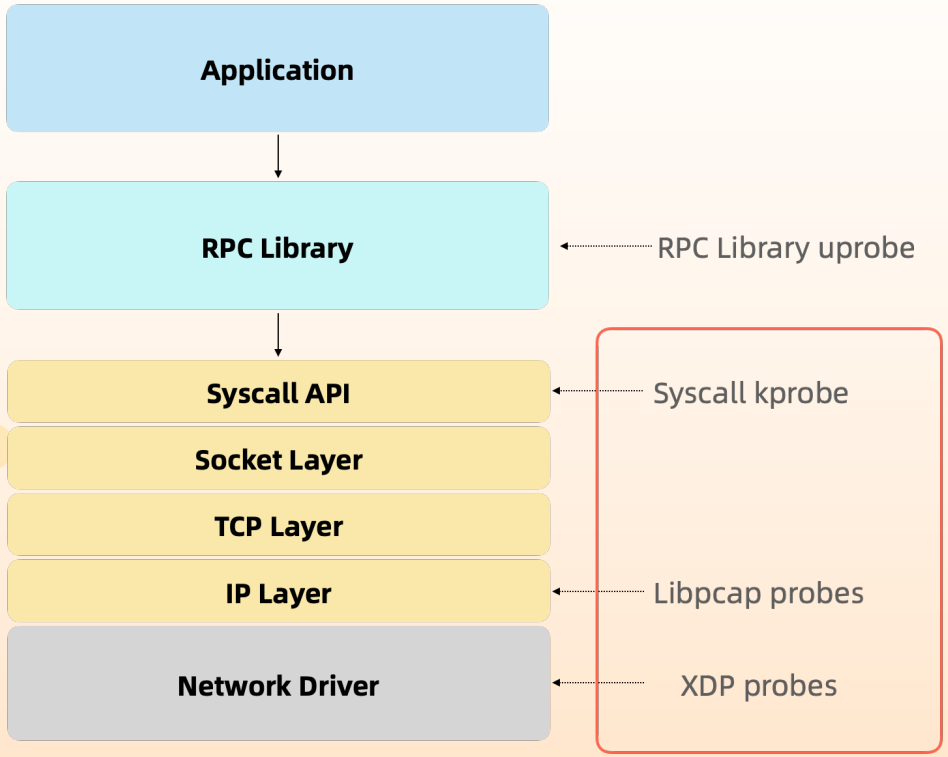
eBPF技术在可观测领域的优势

无侵入

多语言/多协议/多框架

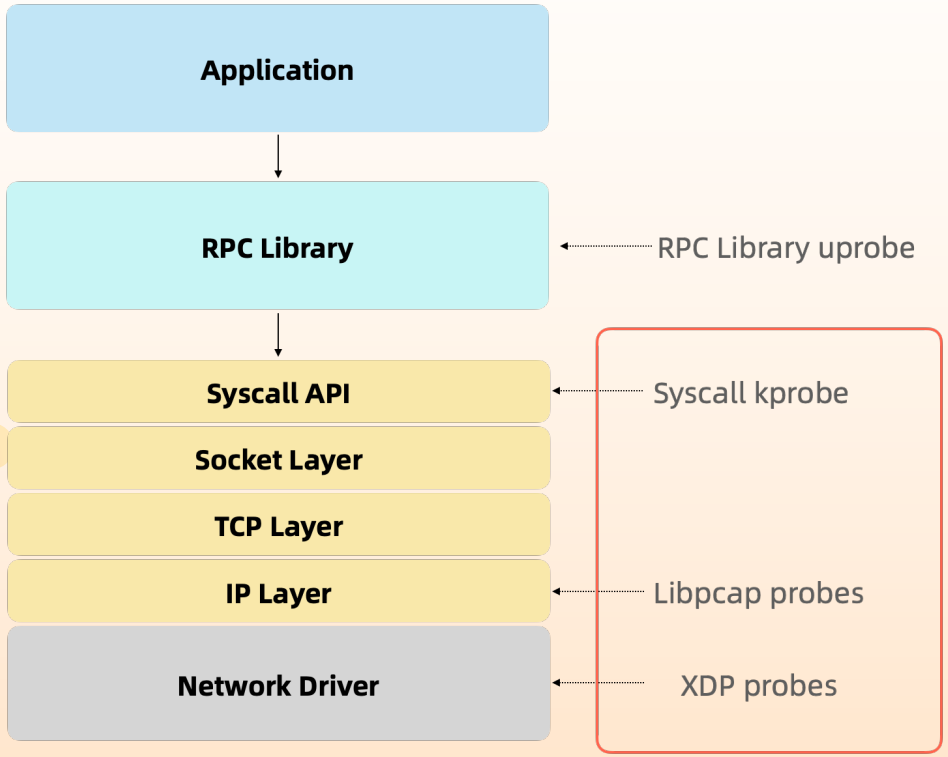
全栈覆盖

无侵入



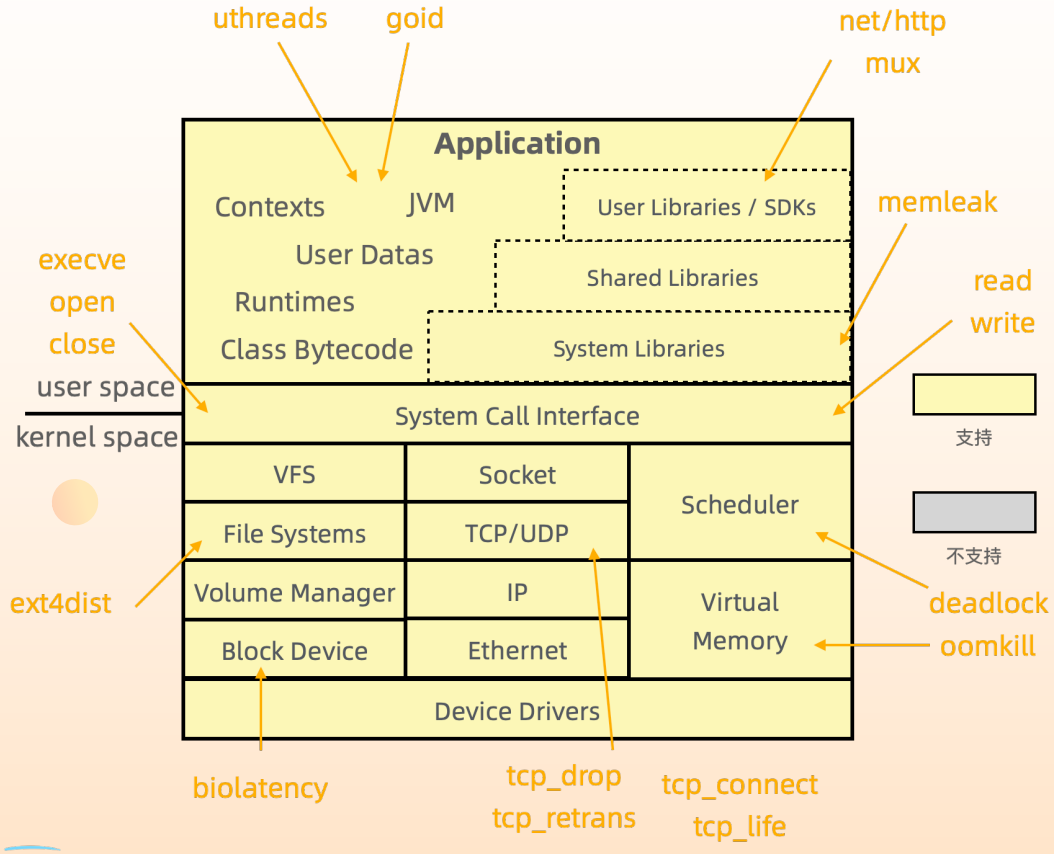
- 无需修改代码
- 无需重启应用
- Verifier保证运行安全

多语言、多协议、多框架



- 捕获网络字节流
- 无需适配编程语言
- 无需适配协议框架
- 同时支持用户态插桩

全栈覆盖



uprobe

kprobe

tracepoint

USDT

perf

...

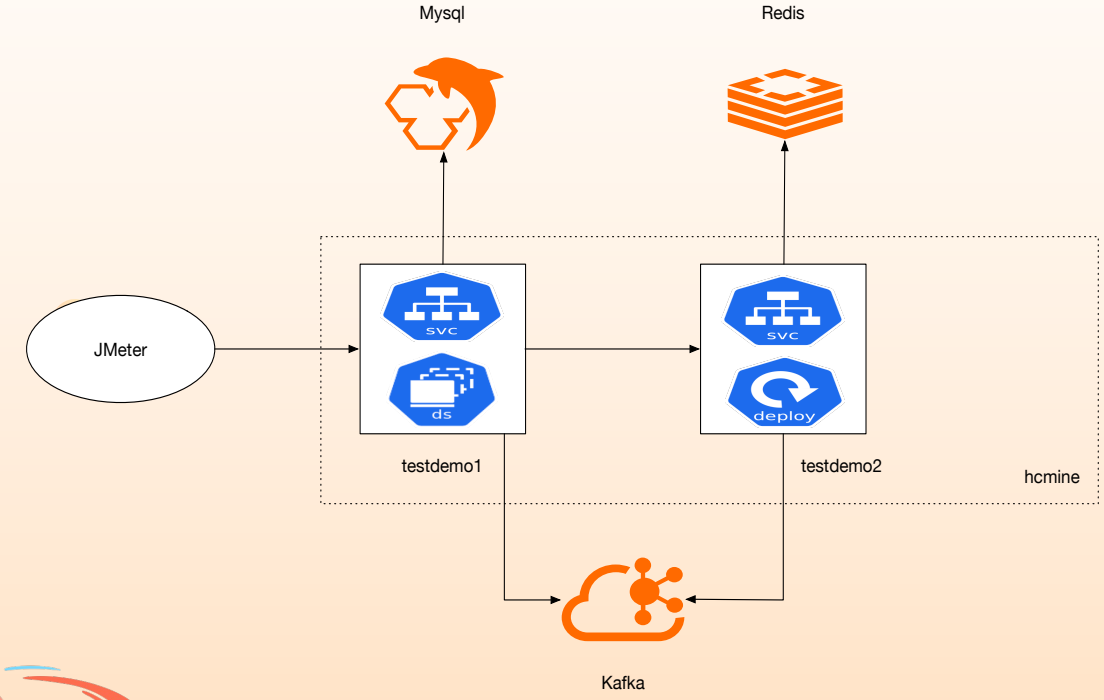
开源社
kaiyuanshe

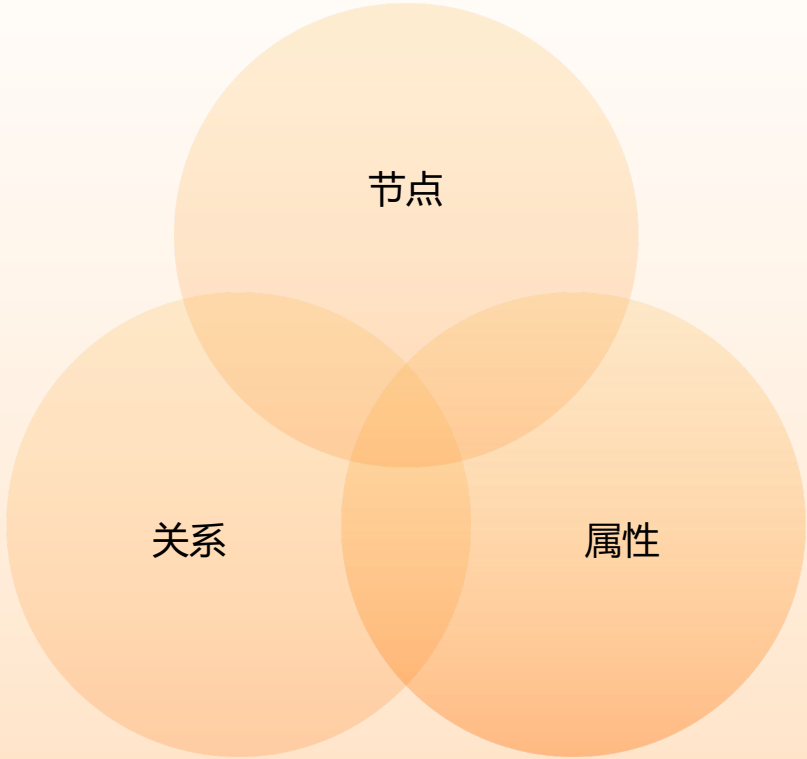
Part 03

构建完整的可观测系统

架构感知

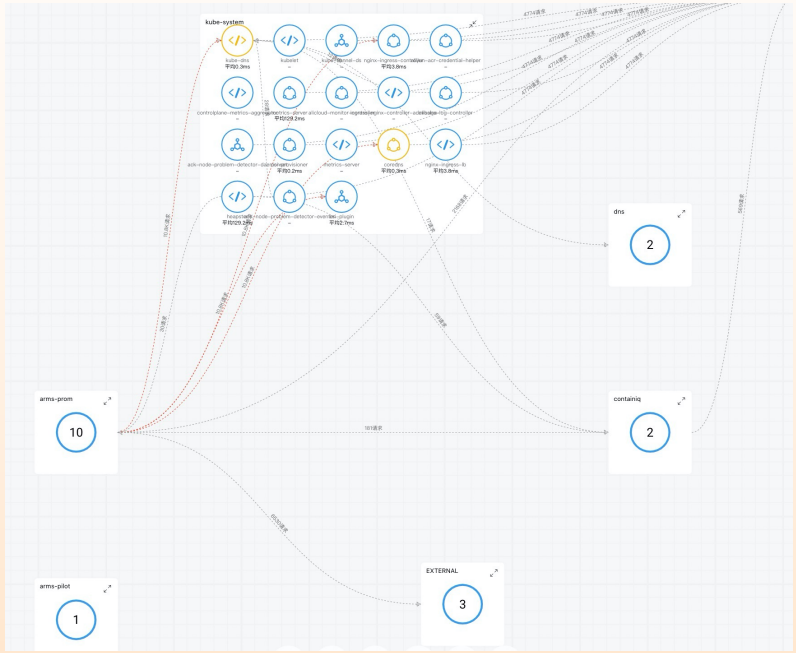
架构感知，节点和关系以及他们的属性，能够正确地反应当前运行的网络关系，帮助用户感知架构，通过对比期望架构，发现问题，通常在新应用上线，新地区开服，整体链路梳理等场景使用。

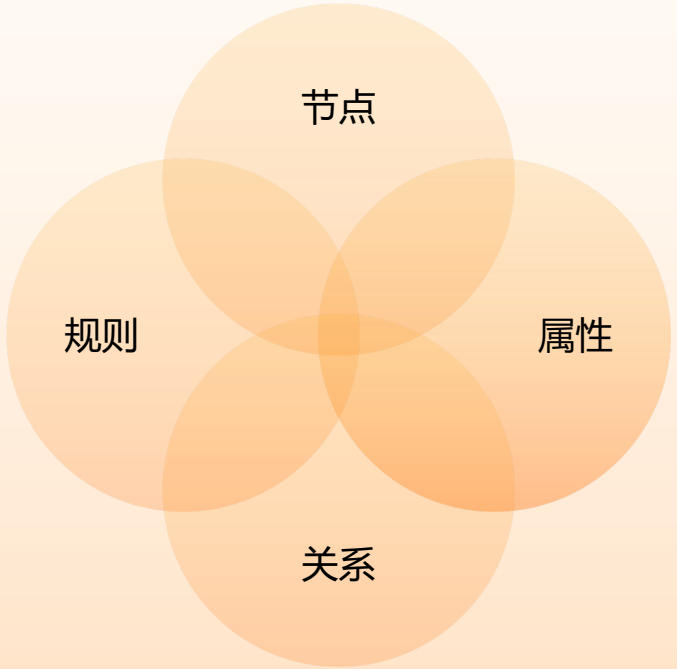




异常发现

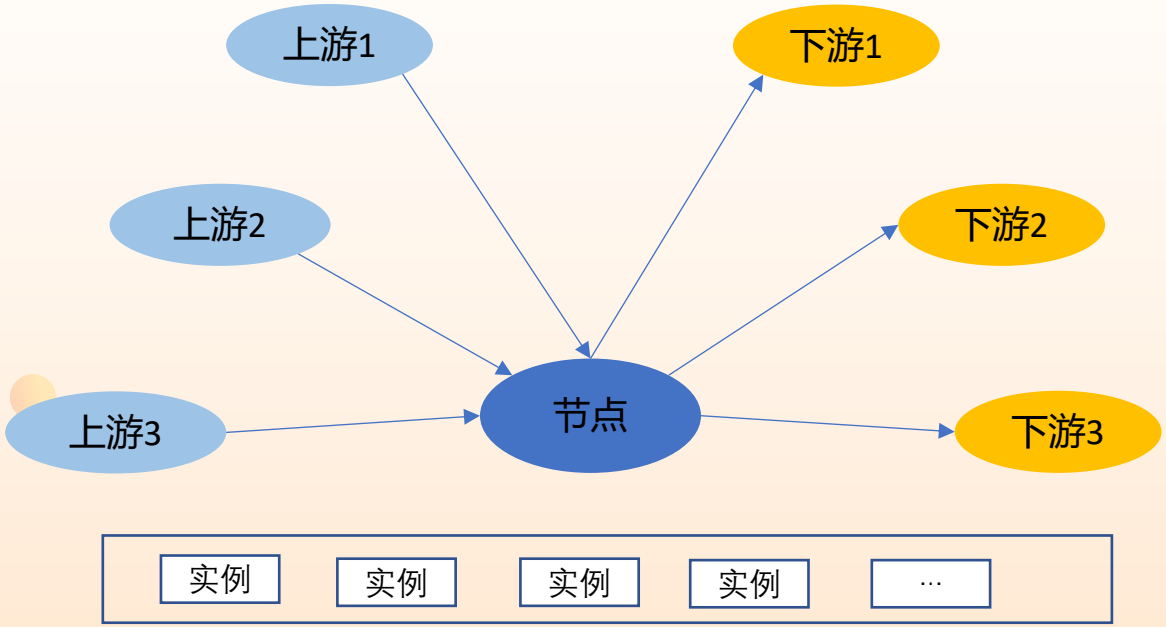
异常发现，通过节点和关系颜色表达，能够快速地发现特点的节点和关系异常，进一步提升问题发现和定位的效率，通常在应用运行时整体链路梳理和特定问题节点上下游分析等场景使用。

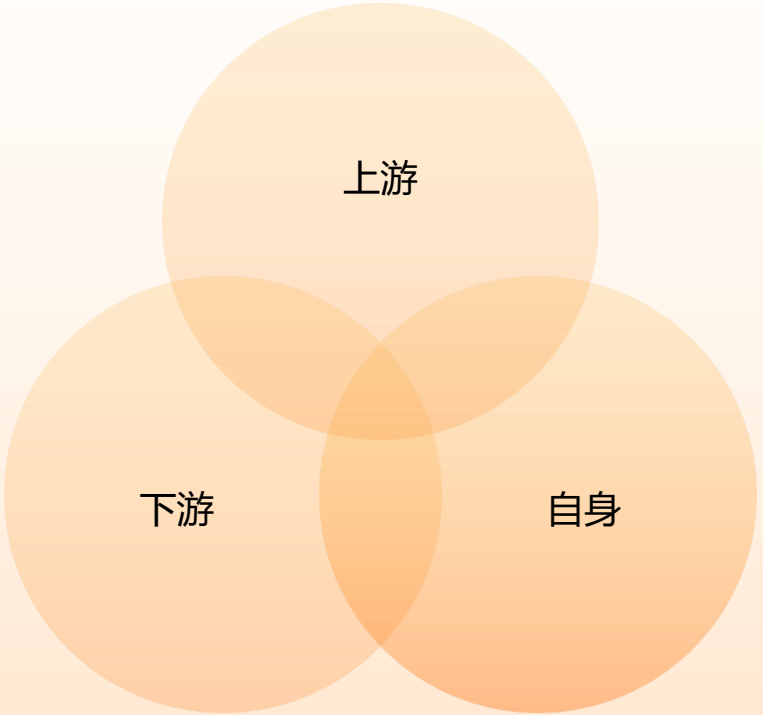




关联分析

关联分析，通过关联关系的切换，可以快速查看上游请求和下游依赖，以及自身服务实例的运行情况，进一步提升问题定位能力，通常在已经定位到某个异常节点后使用。

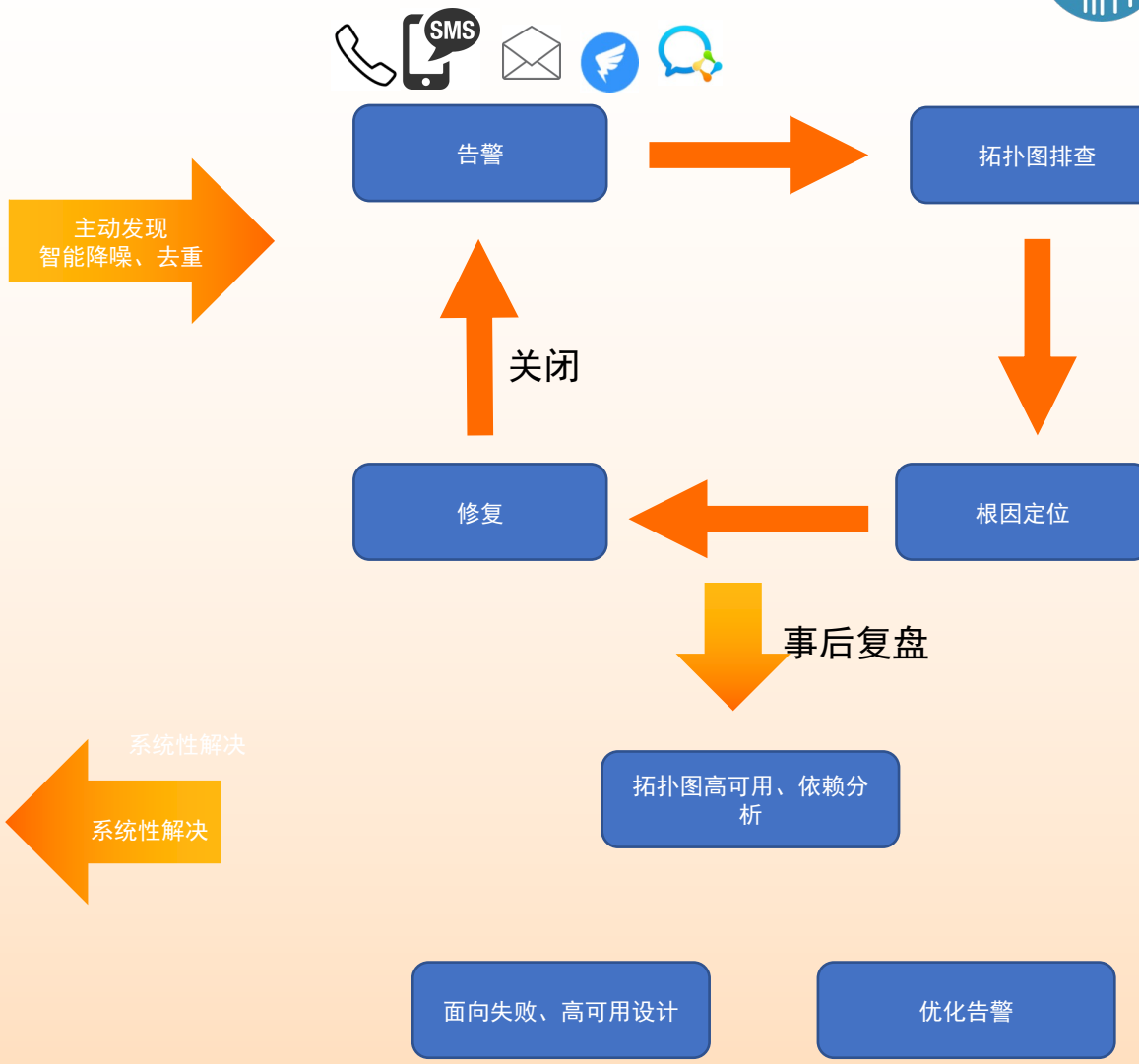




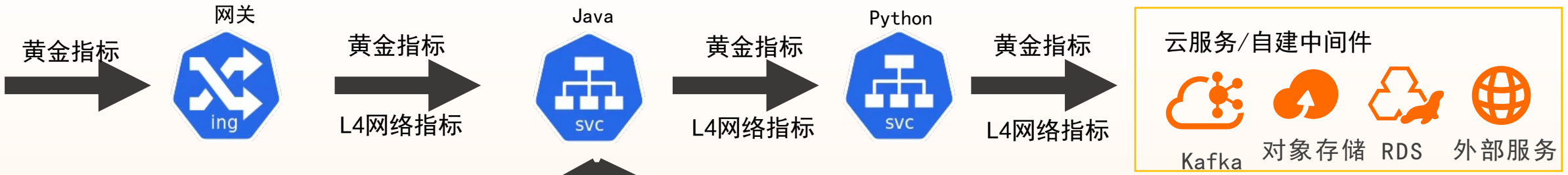
智能告警

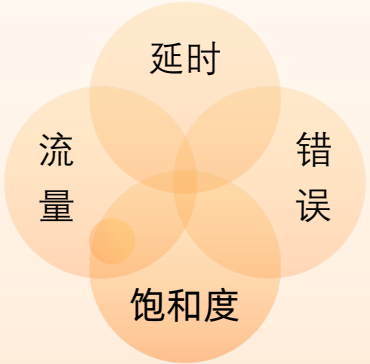
全栈数据源，70+个告警模板开箱即用：
应用级别：Pod/Service/Deployment
K8S控制面：apiserver/ETCD/Scheduler
基础设施：节点、网络、存储
云服务界别：Kafka/MySQL/Redis/

告警收敛，幸福感UP

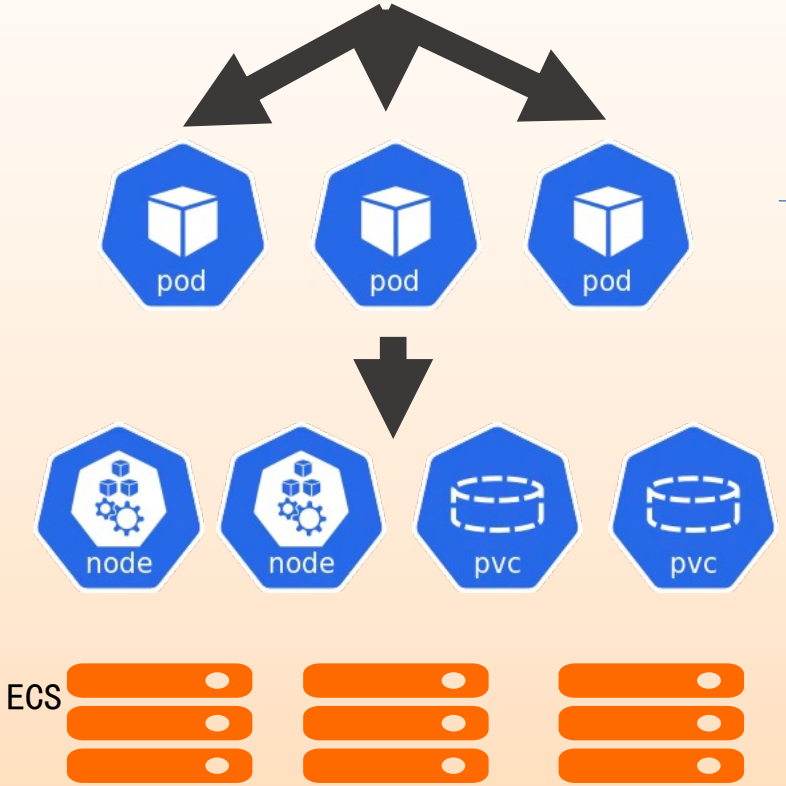


完整的应用可观测





- L4网络指标
  - 4元组
  - 重传
  - 丢包
  - RTT



基础资源指标
日志、事件、调用链

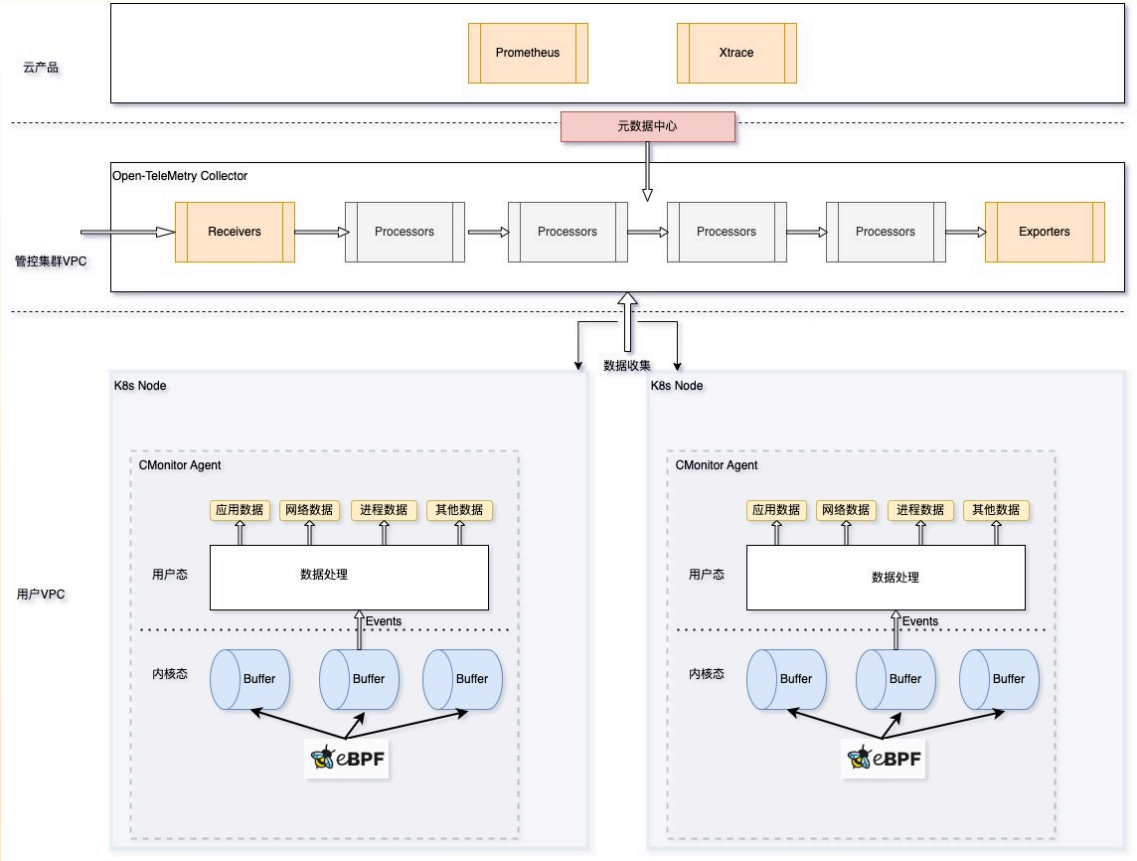
eBPF在阿里云可观测的实践

开源社 kaiyuanshe

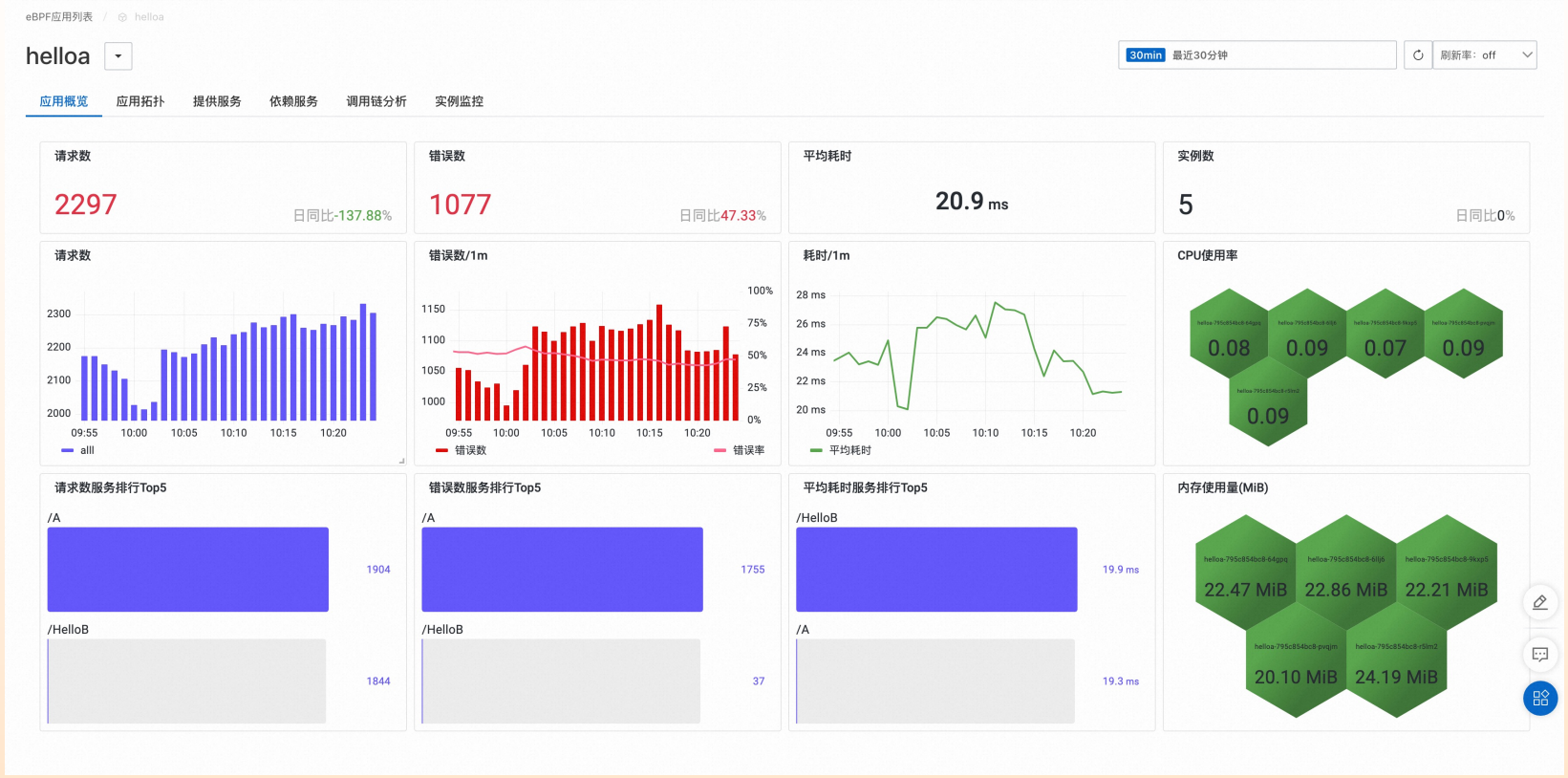
数据展示

数据处理

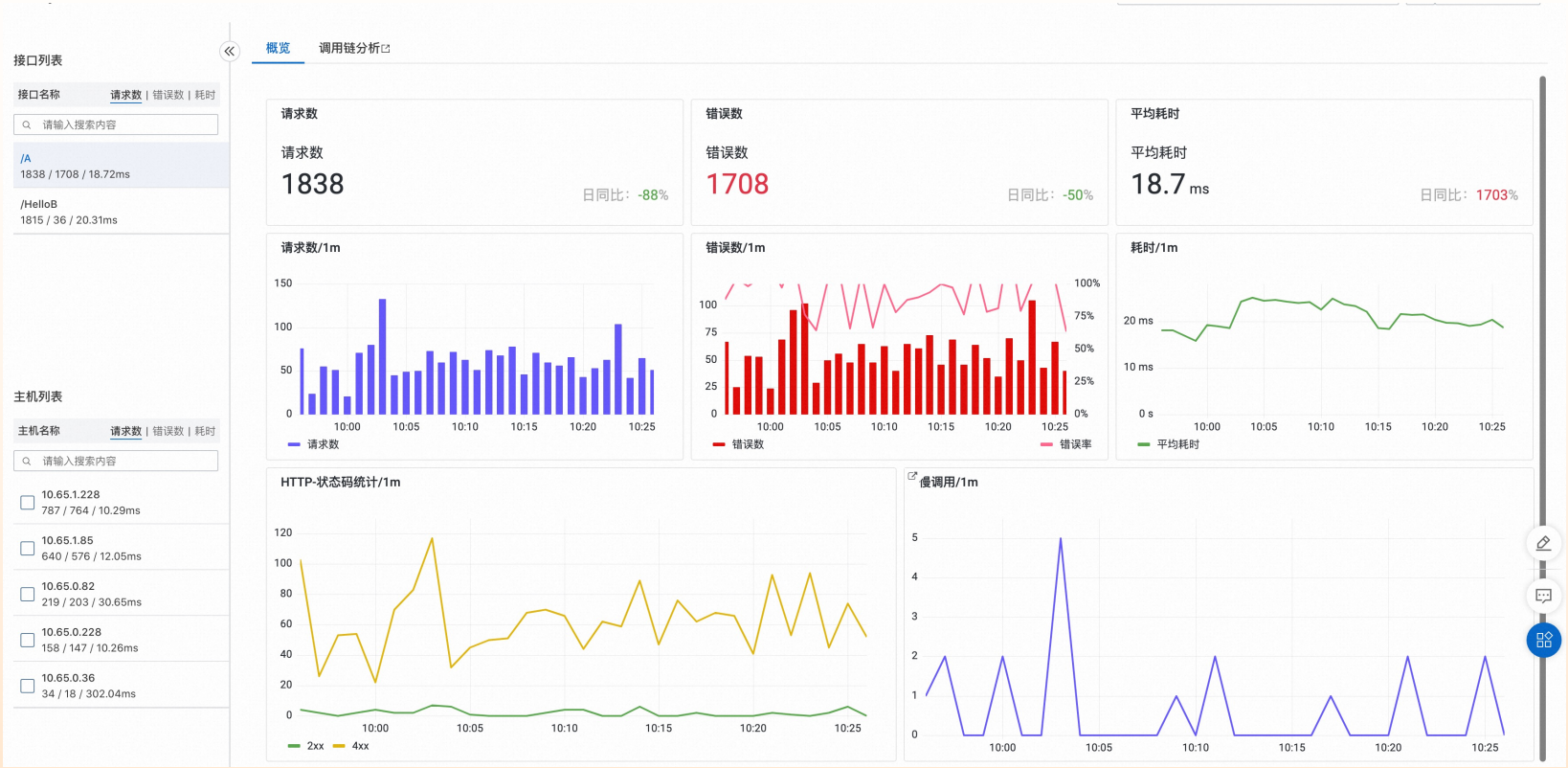
数据采集



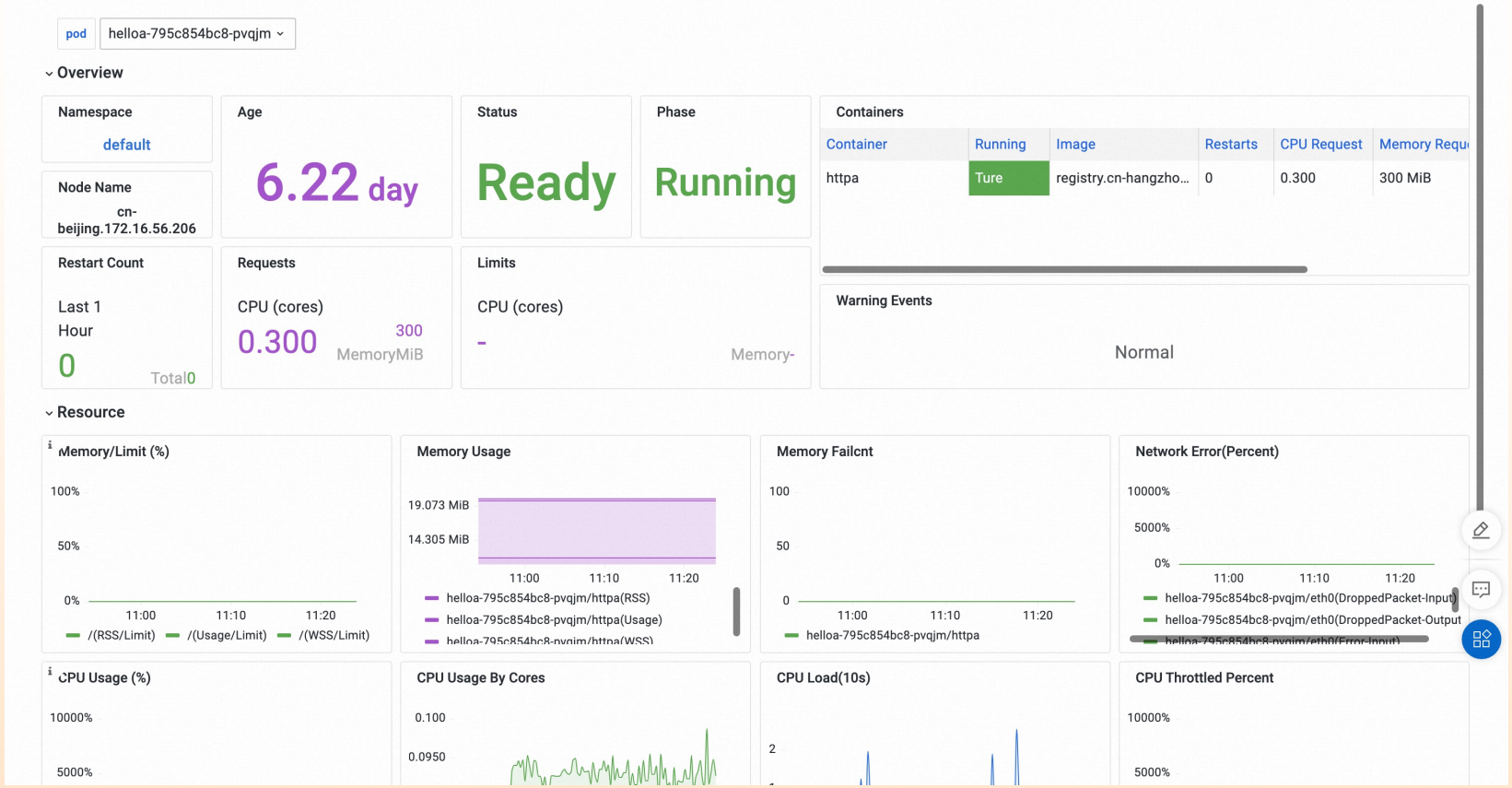
应用详情概览


eBPF应用列表 / helloa
helloa
30min 最近30分钟
刷新率：off
应用概览
应用拓扑
提供服务
依赖服务
调用链分析
实例监控
请求数
2297
日同比-137.88%
错误数
1077
日同比47.33%
平均耗时
20.9 ms
实例数
5
日同比0%
请求数
2300
2200
2100
2000
09:55
10:00
10:05
10:10
10:15
10:20
alll
错误数/1m
1150
1100
1050
1000
100%
75%
50%
25%
0%
错误数
错误率
耗时/1m
28 ms
26 ms
24 ms
22 ms
20 ms
平均耗时
CPU使用率
0.08
0.09
0.07
0.09
0.09
请求数服务排行Top5
/A
1904
/HelloB
1844
错误数服务排行Top5
/A
1755
/HelloB
37
平均耗时服务排行Top5
/HelloB
19.9 ms
/A
19.3 ms
内存使用量(MiB)
22.47 MiB
22.86 MiB
22.21 MiB
20.10 MiB
24.19 MiB

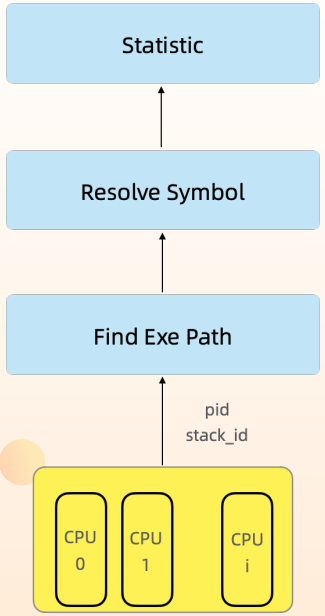
异常接口查询


接口列表
概览
调用链分析
接口名称
请求数 | 错误数 | 耗时
请输入搜索内容
/A
1838 / 1708 / 18.72ms
/HelloB
1815 / 36 / 20.31ms
主机列表
主机名称
请求数 | 错误数 | 耗时
请输入搜索内容
10.65.1.228
787 / 764 / 10.29ms
10.65.1.85
640 / 576 / 12.05ms
10.65.0.82
219 / 203 / 30.65ms
10.65.0.228
158 / 147 / 10.26ms
10.65.0.36
34 / 18 / 302.04ms
请求数
请求数
1838
日同比：-88%
错误数
错误数
1708
日同比：-50%
平均耗时
平均耗时
18.7 ms
日同比：1703%
请求数/1m
错误数/1m
耗时/1m
HTTP-状态码统计/1m
慢调用/1m
2xx
4xx

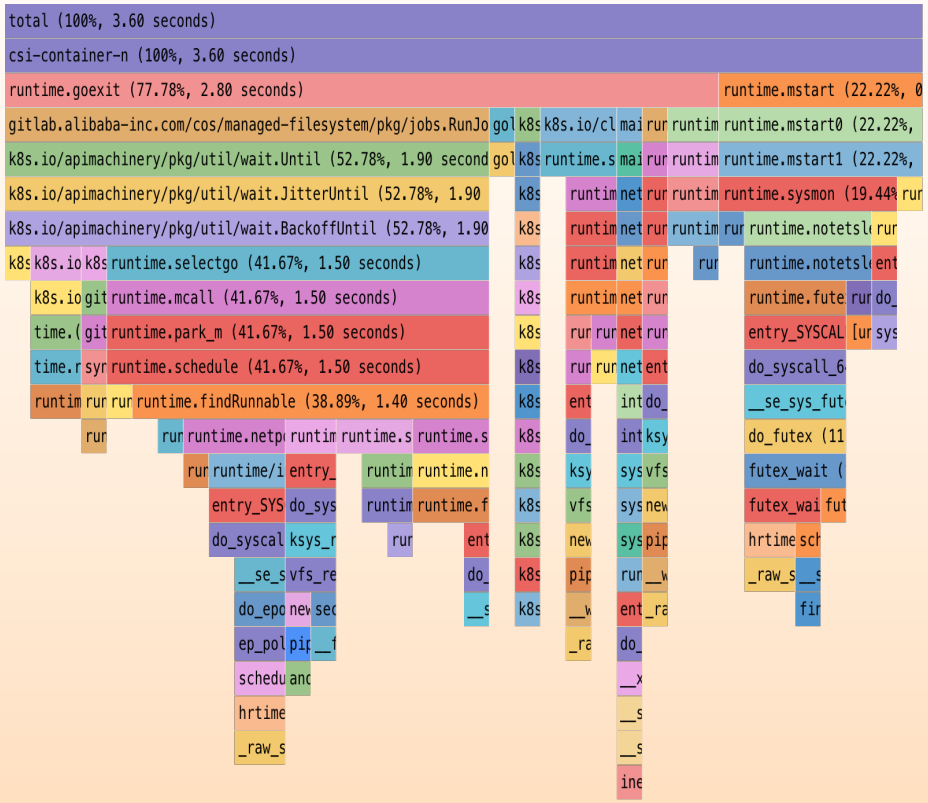
应用实例监控


pod
helloa-795c854bc8-pvqjm
Overview
Namespace
default
Node Name
cn-beijing.172.16.56.206
Age
6.22 day
Status
Ready
Phase
Running
Containers
Container
Running
Image
Restarts
CPU Request
Memory Requ
httpa
Ture
registry.cn-hangzho...
0
0.300
300 MiB
Restart Count
Last 1 Hour
0
Total0
Requests
CPU (cores)
0.300
300 MiB
Limits
CPU (cores)
-
Memory-
Warning Events
Normal
Resource
Memory/Limit (%)
100%
50%
0%
11:00
11:10
11:20
/(RSS/Limit)
/(Usage/Limit)
/(WSS/Limit)
Memory Usage
19.073 MiB
14.305 MiB
helloa-795c854bc8-pvqjm/httpa(RSS)
helloa-795c854bc8-pvqjm/httpa(Usage)
Memory Failcnt
100
50
0
helloa-795c854bc8-pvqjm/httpa
Network Error(Percent)
10000%
5000%
0%
helloa-795c854bc8-pvqjm/eth0(DroppedPacket-Input)
helloa-795c854bc8-pvqjm/eth0(DroppedPacket-Output)
CPU Usage (%)
10000%
5000%
CPU Usage By Cores
0.100
0.0950
CPU Load(10s)
2
CPU Throttled Percent
10000%
5000%

持续剖析



提供基于eBPF的应用Continuous Profiling能力，基于微服务应用符号表，自动标记微服务调用栈，轻松获取整个容器集群应用Profiling数据，快速查询应用CPU热点

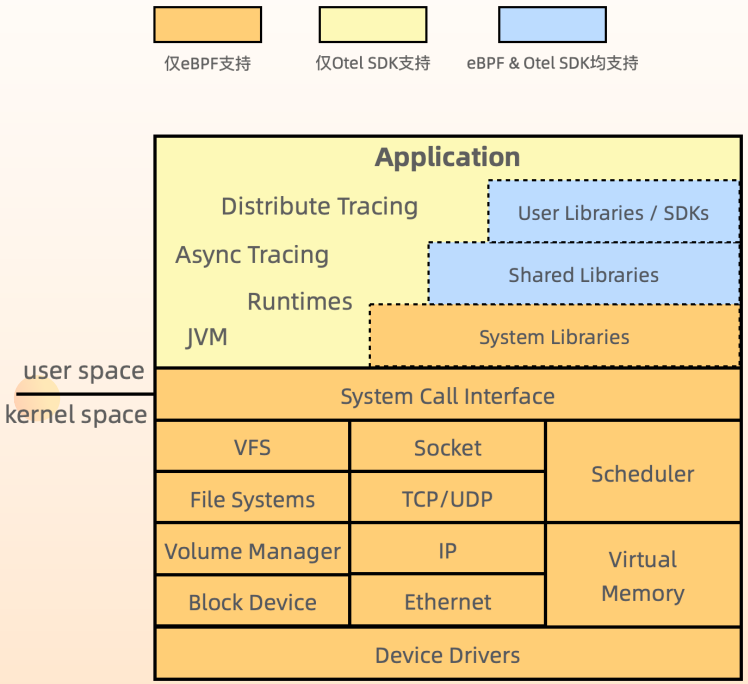


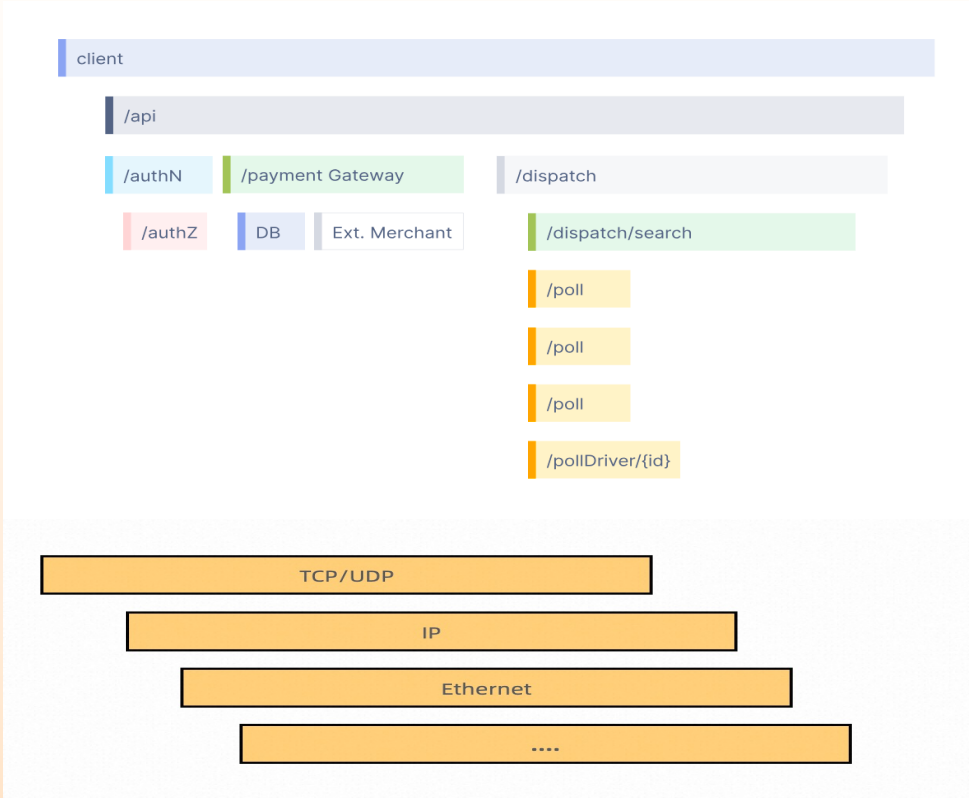
Part 04

未来与展望

结合OtelSDK构建全链路Trace能力

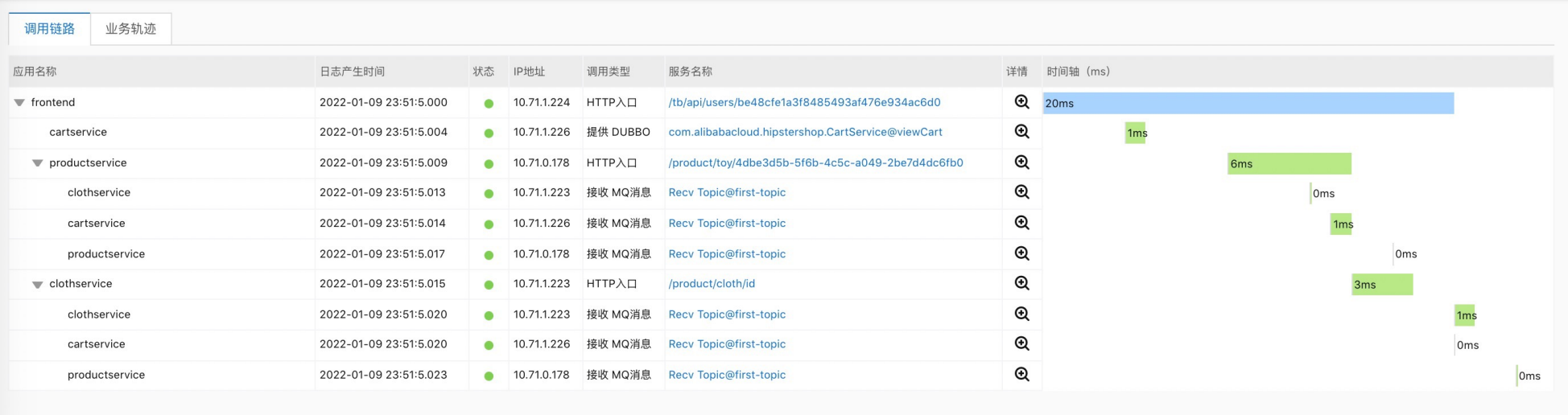
Otel SDK提供应用Trace能力，eBPF提供网络层Trace能力，打通这两种实现全栈的Trace





无侵入的TraceId透传

eBPF因为它本身的无侵入性，在应用内部的Trace Id透传上无法满足要求，特别是异步场景下的Trace Id 透传，当前同步场景下的Trace Id透传已经解决，类似golang、java这些高级语言的异步场景还无法满足客户的生产可用



拥抱开源、贡献社区

预计今年12月份左右相关工作将贡献给alibaba开源项目KubeSkoop，构建Kubernete下的全栈可观测能力，尽情期待。

https://github.com/alibaba/kubeskoop

THANK YOU

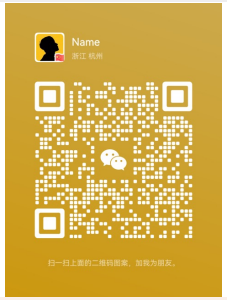
QUESTIONS?

欢迎扫码打卡
积分可兑换对应礼品哟！

微信公众号：开源社KAIYUANSHE
视频号：开源社KAIYUANSHE
新浪微博：开源社
B站：开源社KAIYUANSHE

简书：开源社
头条：开源社
Facebook：KaiyuansheChina
Twitter：开源社KAIYUANSHE

扫码关注开源社公众号



扫码添加讲师联系方式