第二届 eBPF开发者大会
www.ebpftravel.com

阿里云使用eBPF深度监控k8s网络实践
--KubeSkoop

王炳燊（溪恒）

中国·西安

目录

① K8S网络问题复杂性

K8S网络问题复杂性

网络逻辑概念复杂

链路实现层次复杂

K8S网络逻辑概念

Kubernetes中包含很多网络逻辑概念导致配置复杂：

Ingress/Service/NetworkPolicy 配置

- LabelSelector选择到不预期的Pod
- 多个NetworkPolicy规则重叠
- Service NAT的端口和实际端口不符合

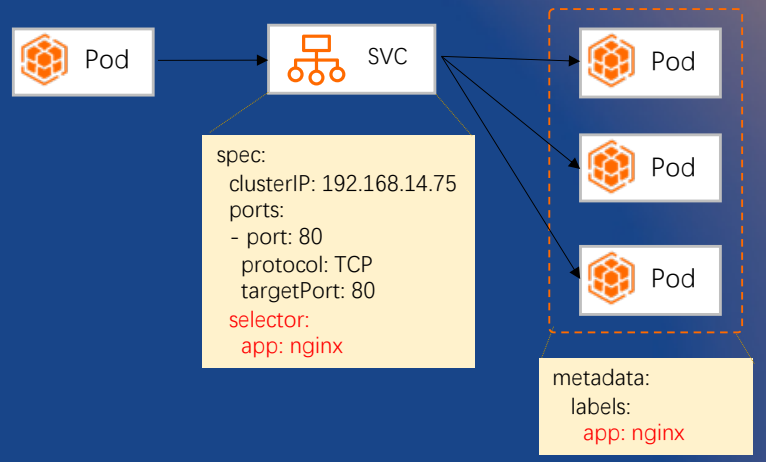
ServiceMesh

- 更复杂的7层网络策略

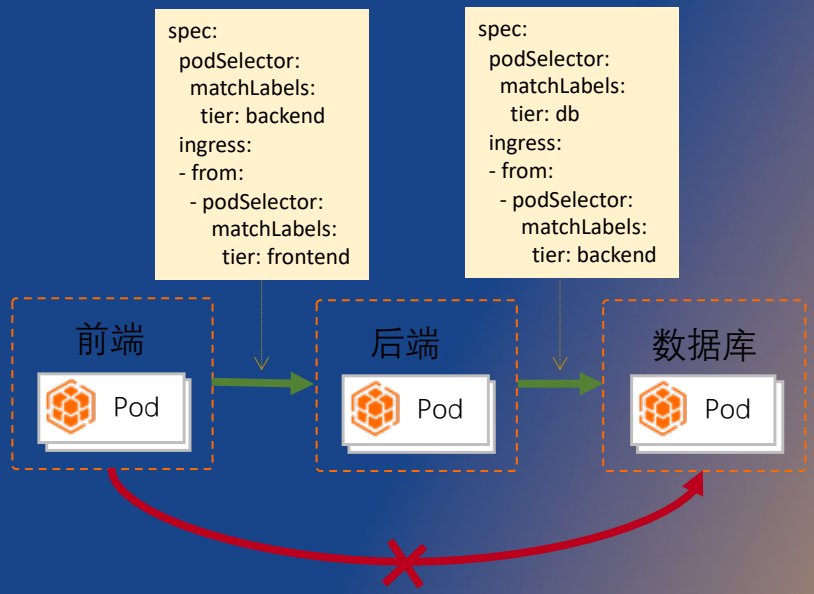
第三方网络/Ingress插件

- 自定义的网络扩展

K8S服务发现 -- Service



K8S网络策略 -- NetworkPolicy



容器网络的链路实现

数据平面复杂：

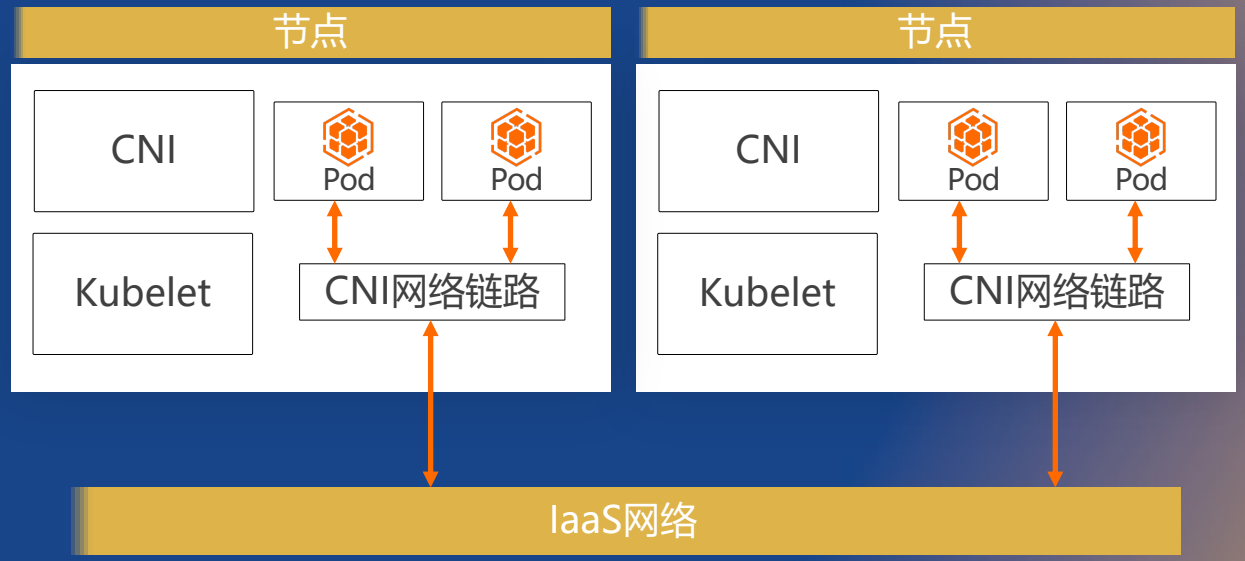
- 数据面多层处理，ServiceMesh/KubeProxy/CNI
- 多种CNI网络实现(Overlay/Underlay...)

协议栈链路复杂：

- 链路长，涉及网卡驱动/netfilter/route/bridge等
- 配置繁琐难懂

底层网络配置：

- 每个云厂商配置不同
- 安全组、路由表等配置复杂



传统定位网络问题手段

定位流程长	大量定位时间开销	人员经验要求高
在Pod、Node等多处同时抓包	压测等手段复现问题	熟练的运维能力
定位丢包点	需要了解网络插件原理	精通Linux协议栈
分析丢包点配置(iptables/内核参数/云厂商网络等)	大量的抓包和信息采集	精通对应云厂商的网络配置

第二届 eBPF开发者大会

www.ebpftravel.com

② KubeSkoop项目介绍

中国·西安

KubeSkoop项目介绍

K8S网络问题诊断工具集


连通性诊断


异常回溯


访问流记录

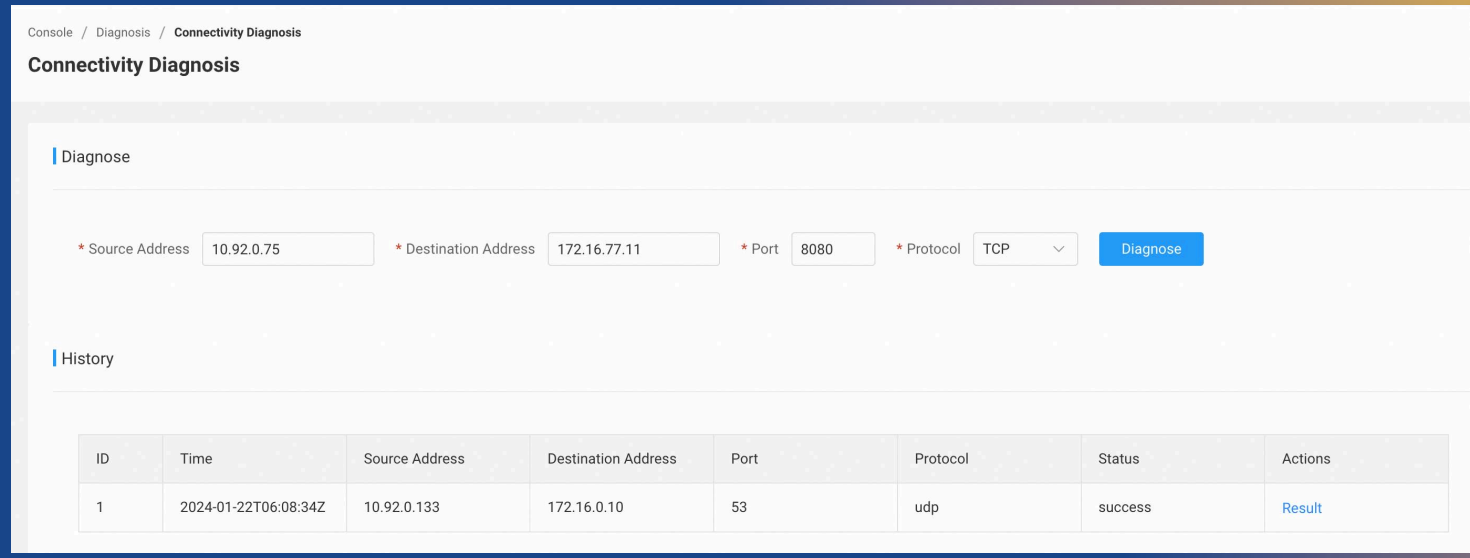

延迟探测

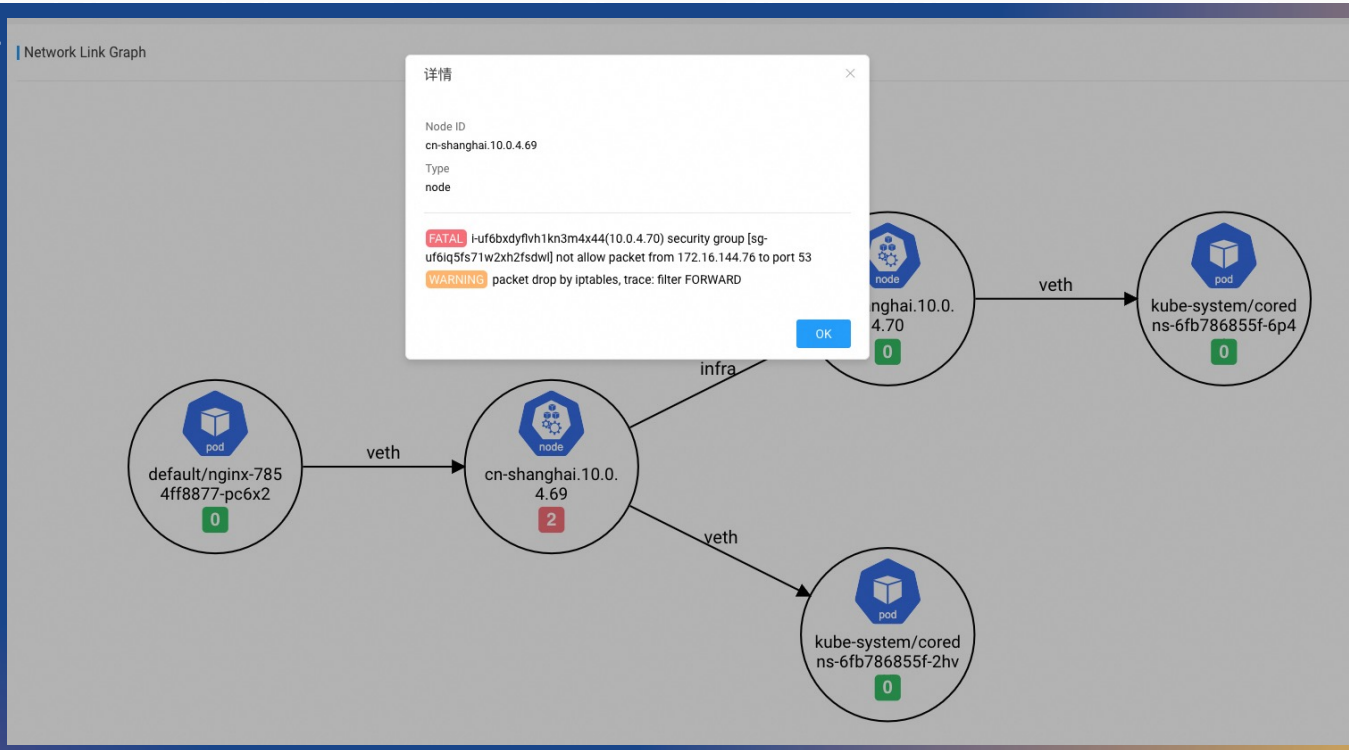

分布式抓包

项目地址： https://github.com/alibaba/kubeskoop

连通性诊断

- 用户指定不通的来源目的，等待输出诊断的结果
- 自动构建网络访问链路，分析链路问题
- 包含Kubernetes Service、NetworkPolicy等概念分析
- 全面覆盖协议栈、底层IaaS的连通性相关检查
- 无需了解网络插件实现和复杂的网络问题排查经验





监控和异常回溯

基于eBPF的容器网络异常监控：

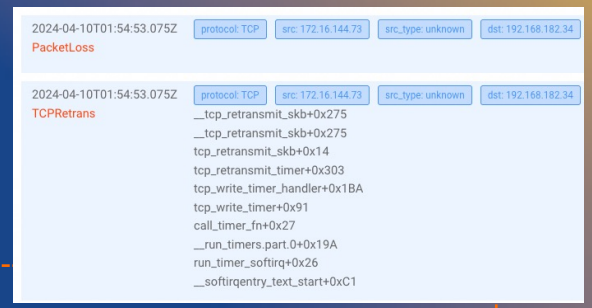
- 云原生部署，与Prometheus等可观测体系对接
- Pod级别的网络监控能力
- 精简、低开销的内核异常监控

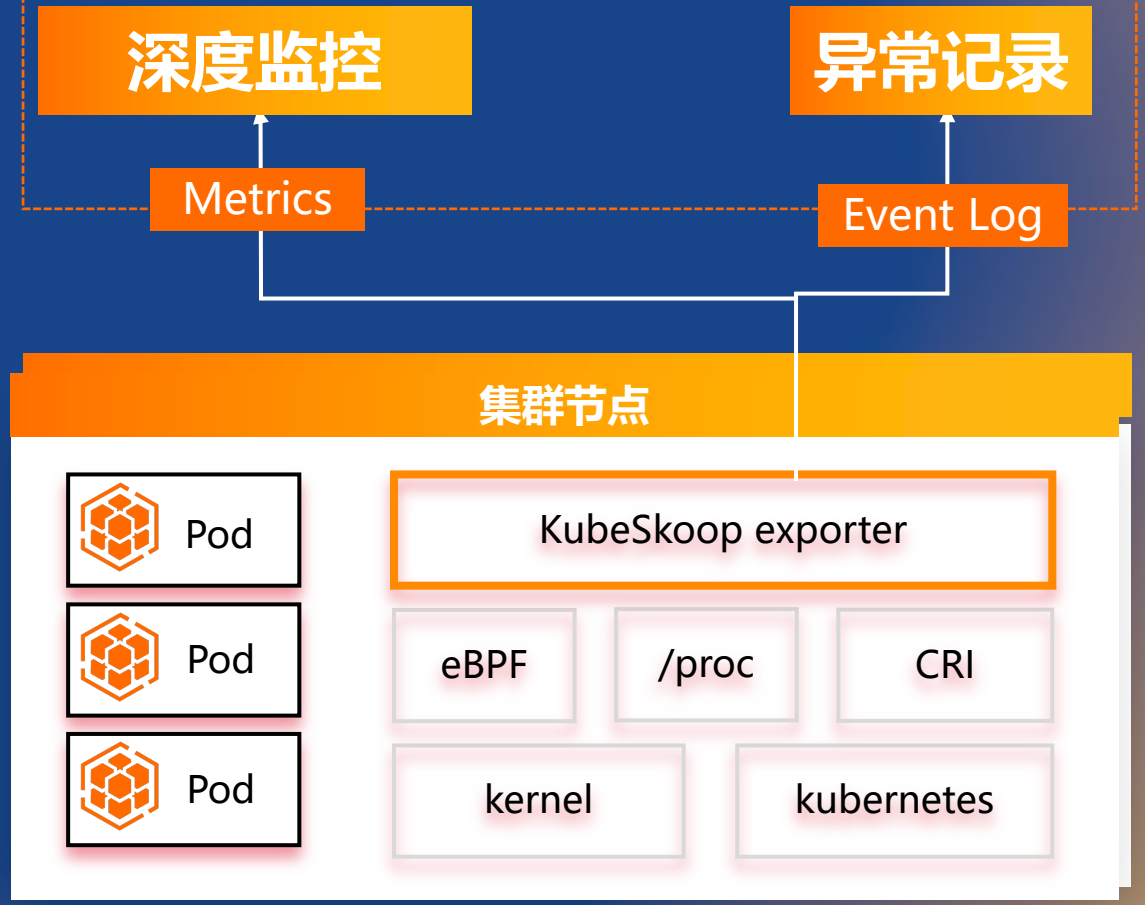
覆盖多种问题场景诊断：

- 网络偶发丢包重传诊断
- Flow级别异常识别
- 网络全链路延迟分析

多种类型异常透出：

- 反映访问关系和异常趋势的Metrics
- 记录异常现场的历史事件记录





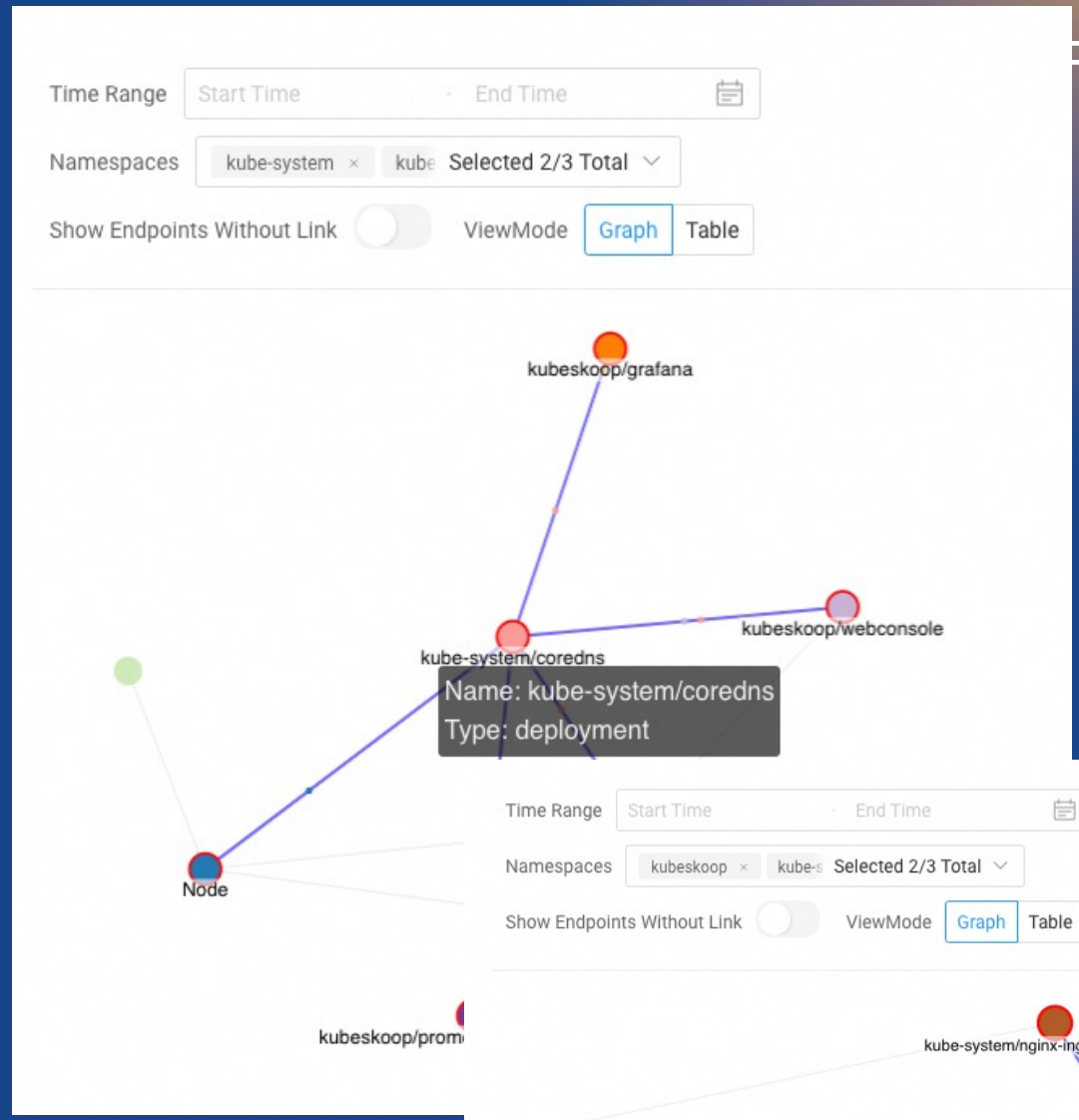
访问流记录

历史访问关系记录和回溯

- 记录流上连接数、包数量、吞吐
- 分析应用和网络性能占用和瓶颈
- 排查历史公用资源占用情况(CoreDNS)

低开销的访问关系采集

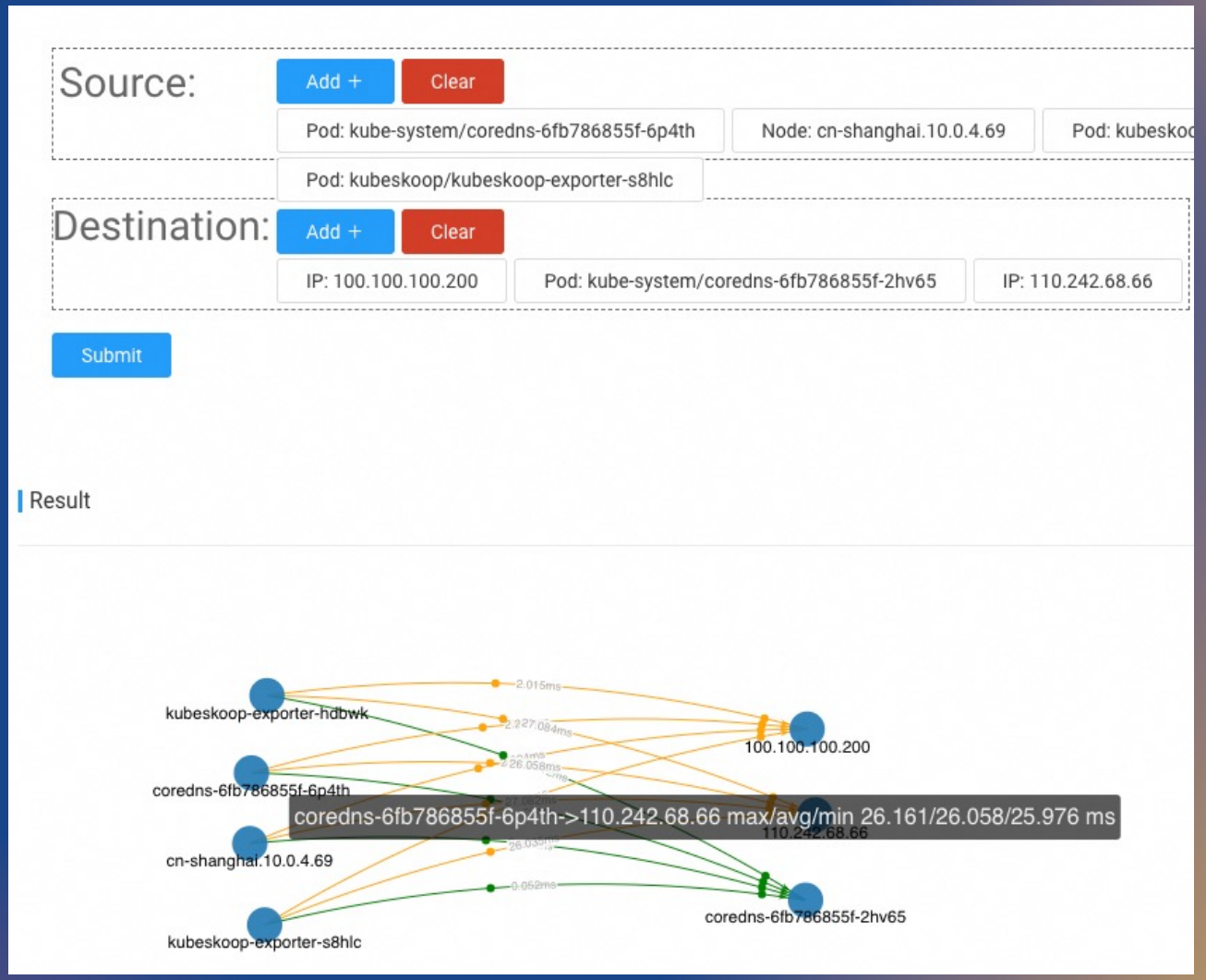
- 采用eBPF的访问流采集



延迟探测（PingMesh）

集群延迟探测

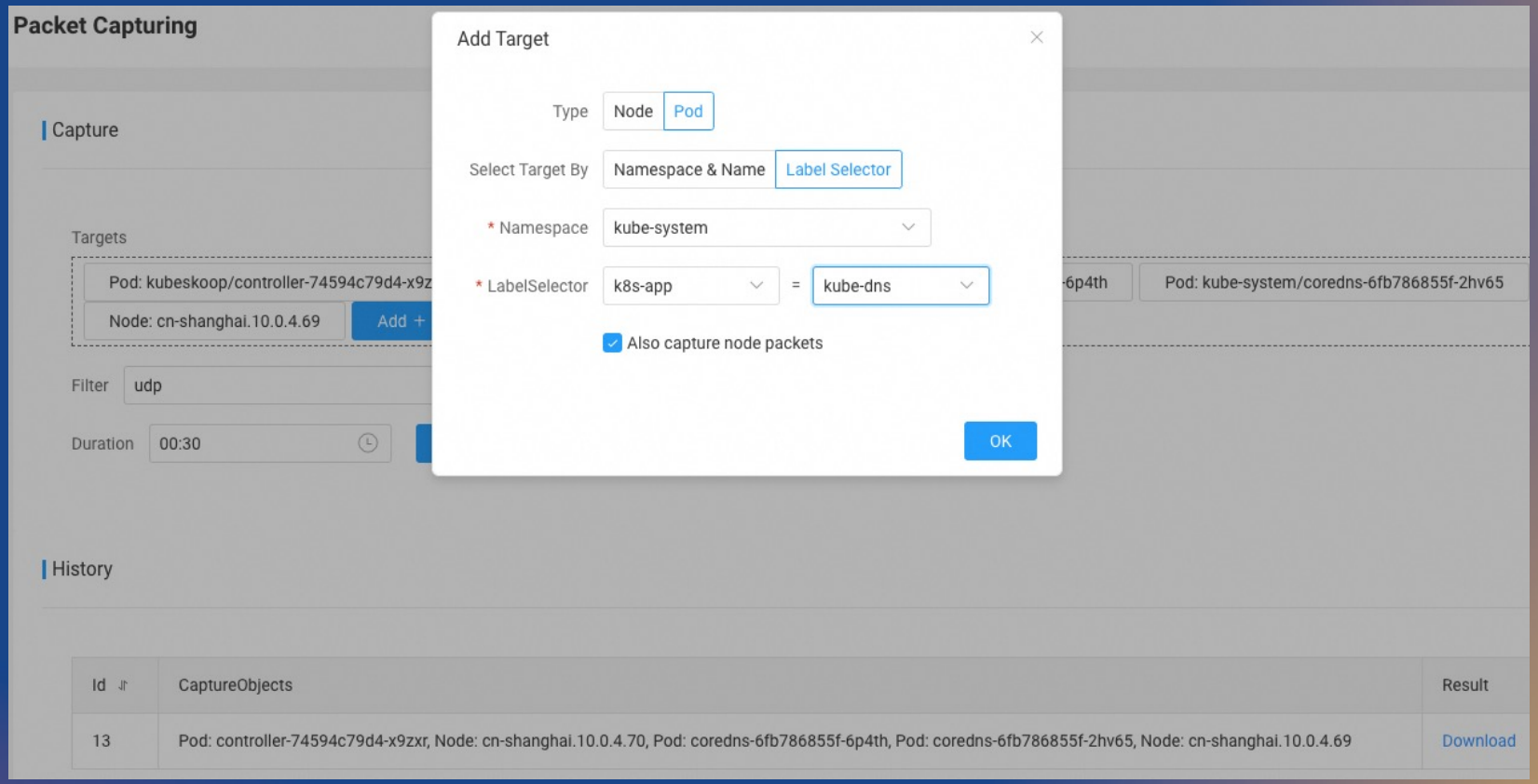
- 一键生成集群中延迟情况
- 排查性能瓶颈
- 优化应用部署拓扑



分布式抓包

集群多点同步抓包

- 一键抓取&采集全链路数据包
- 未知的原因丢包问题的定界



第二届 eBPF开发者大会

www.ebpftravel.com

③ KubeSkoop基于eBPF的K8S网络监控

中国·西安

为什么采用eBPF分析k8s网络问题

K8S使用的协议栈复杂

- TCP
- Bridge
- Netfilter
- TC
- …

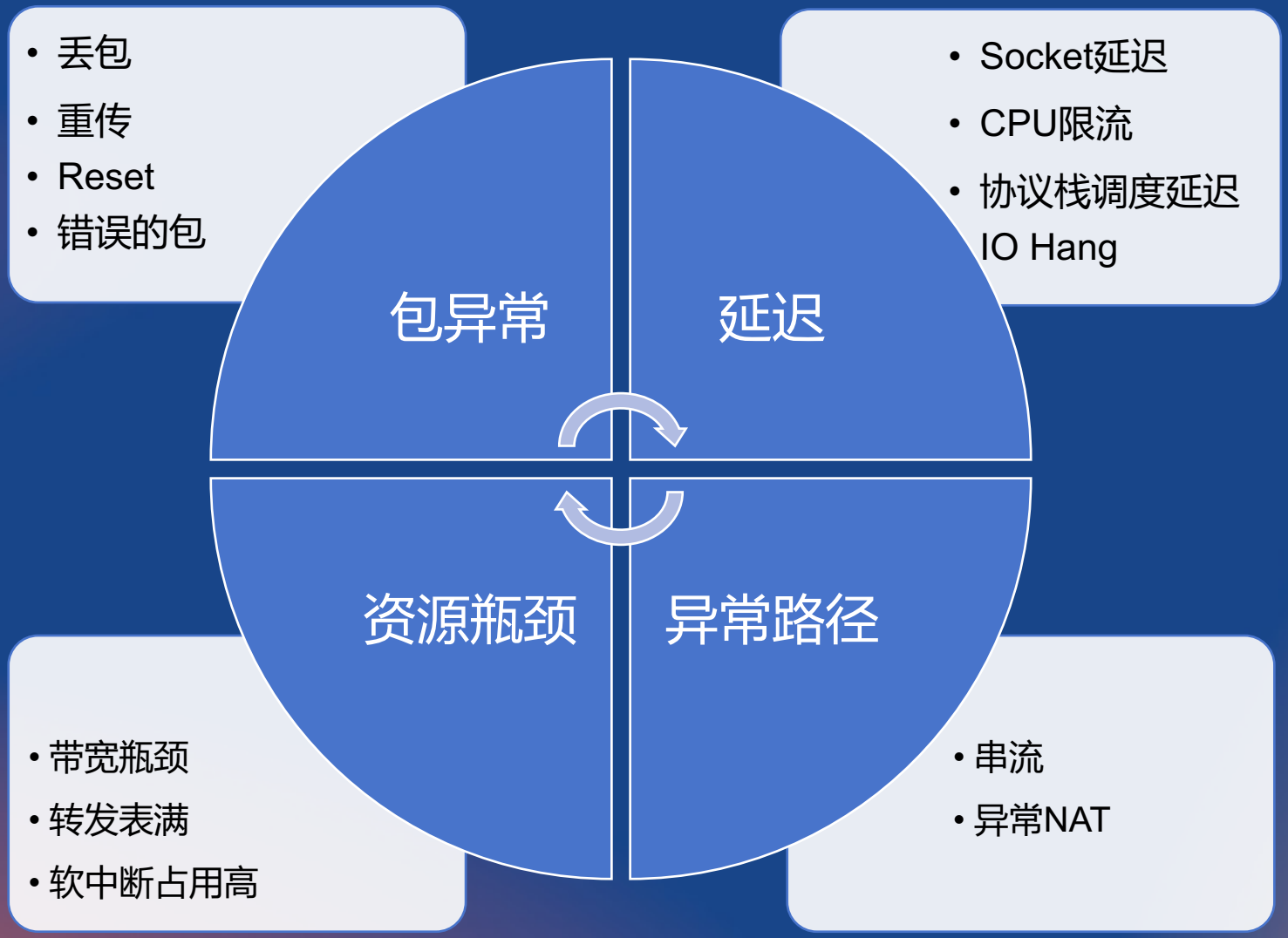
内核原生透出的信息少

- 缺少netns信息透出
- 关键链路缺少记录
- 没有上下文信息
- …

eBPF程序分发部署便利性

- 兼容性好
- 安全性高
- 内核态过滤统计开销低
- 让排查问题经验可复制

K8S网络问题关键特征

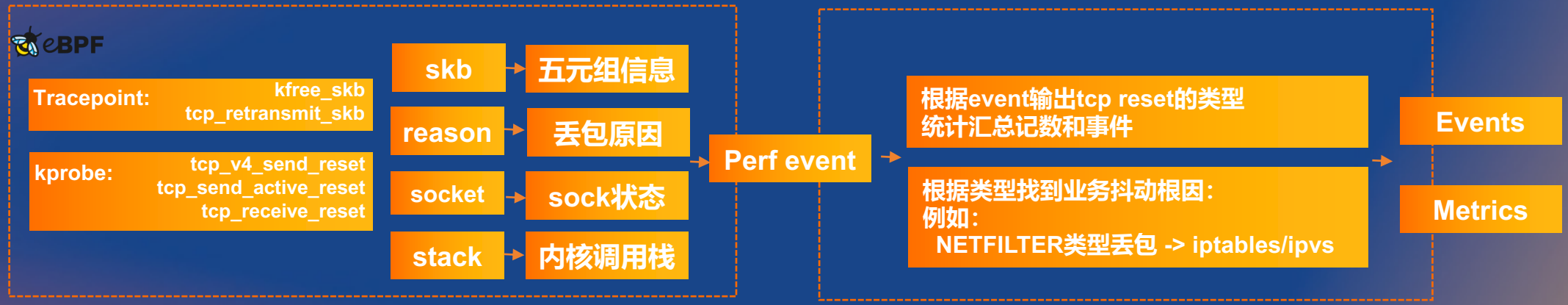


eBPF监控包异常

包异常导致业务偶发的抖动

- 丢包&重传
- tcp reset

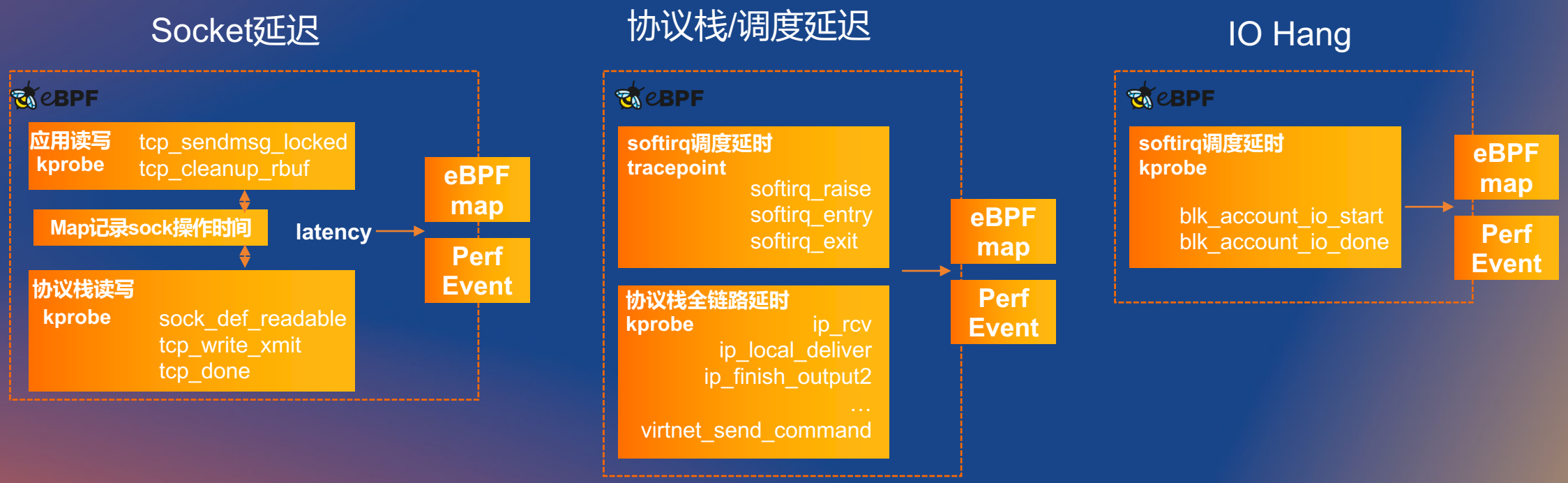
采用eBPF可以追踪调用栈，从而找到丢包根因



eBPF监控网络延迟问题

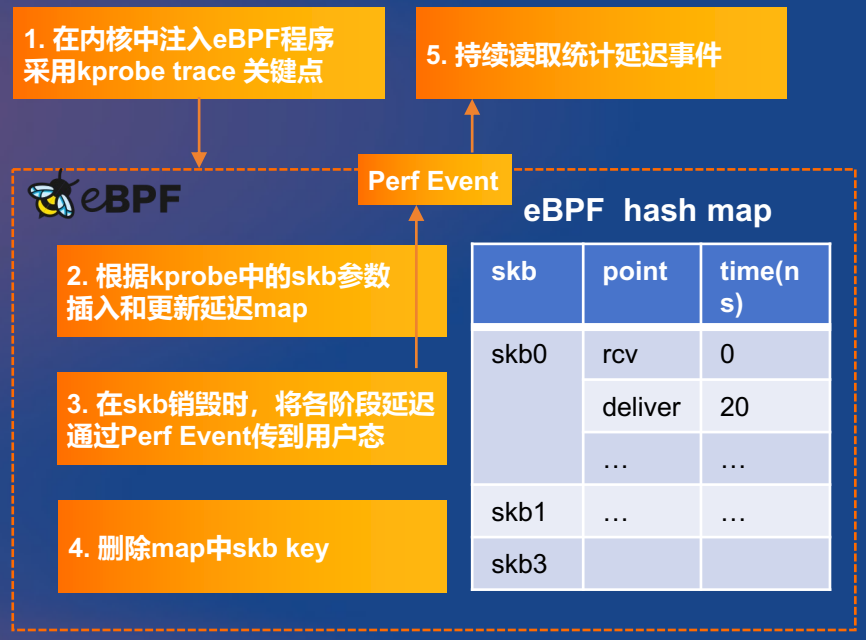
网络延迟问题
- Socket延迟
- 协议栈/调度延迟
- IO Hang

eBPF采用skb,pid作为key追踪，在各个点中采集延迟，定位延迟根因



eBPF监控网络延迟问题

eBPF协议栈延迟检测方式

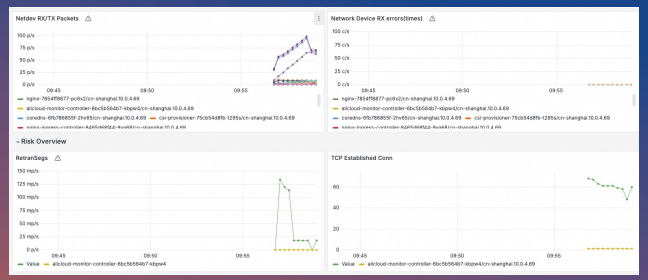


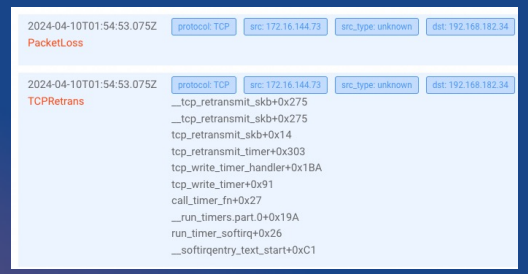
协议栈全链路延时检测

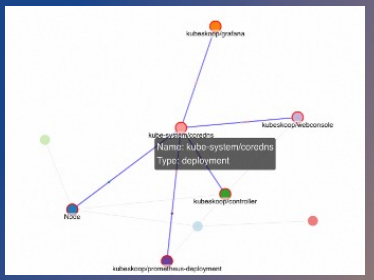
- 发包链路
- 收包链路

KubeSkoop eBPF指标采集和处理

	Metrics	Events	Flow
	• 延迟 • 异常计数	• 事件 • 调用栈	• 五元组 • 统计信息
采集	• Tracepoint • kprobe&uprobe	• Tracepoint • kprobe	• TC • sockops
同步	ebpf map异步读取	Perf Event	ebpf map异步读取
存储	Prometheus 时间序列	• Loki • 日志存储	Prometheus时间序列
展示	Grafana Dashboard	回溯查询事件展示	查询指标组成访问关系图







KubeSkoop eBPF监控案例

ebpf分析业务抖动根因问题案例 -- Nginx-Ingress-Controller访问偶发超时

监控回溯
Metrics

事件记录
Events

```
INFO  March  9 21:21:05 VIRTCMDEXCUTE nginx-ingress-
controller-5c7fb5594-jwhnw protocol=TCP
saddr=100.121.88.193 sport=27057 daddr=10.33.0.11
dport=443  stacktrace: virtnet_send_command+0x1
virtnet_set_rx_mode+0x251  __dev_change_flags+0x9c
dev_change_flags+0x21  do_setlink+0x257
__rtnl_newlink+0x600  rtnl_newlink+0x44
rtnetlink_rcv_msg+0x119  netlink_rcv_skb+0x4e
netlink_unicast+0x1d7  netlink_sendmsg+0x240
sock_sendmsg+0x5f  ____sys_sendmsg+0x232
___sys_sendmsg+0x75  __sys_sendmsg+0x49
do_syscall_64+0x30
entry_SYSCALL_64_after_hwframe+0x61
```

业务异常时间点 21：25 左右
通过回溯KubeSkoop监控
业务Pod重传↑，Virtnet指令延时突增
然后通过事件分析到调用栈和根因
为底层虚拟化执行超时导致软中断卡住而延迟重传

KubeSkoop eBPF程序管理问题

监控能力部署分发

- 依赖内核的Header编译
- clang/llvm编译环境大(300MB+)

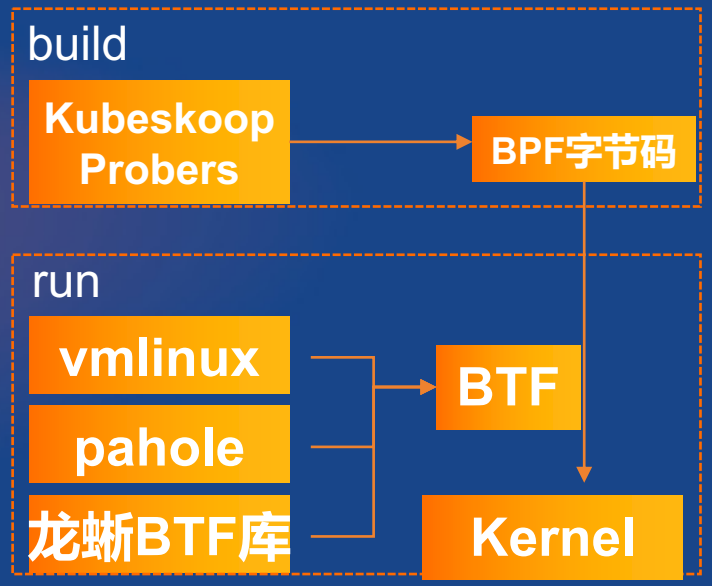
eBPF程序管理

- 数十个eBPF程序动态插拔
- 各种功能开关带来的动态配置

大量数据的有效采集

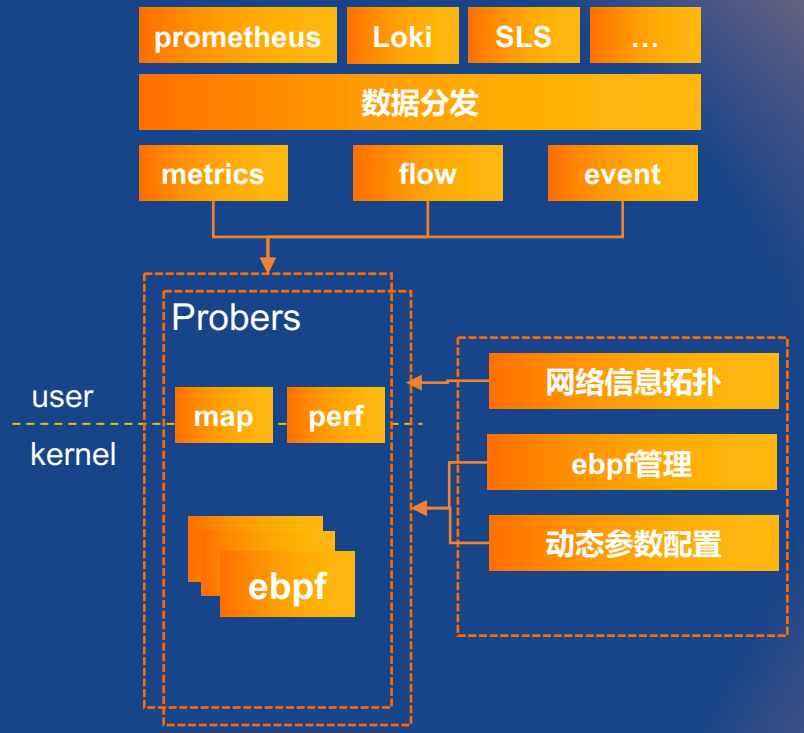
- 各种程序的数据标签不同
- 事件和监控指标需要输出多种存储

KubeSkoop eBPF程序管理



采用eBPF CO-RE

- 不需要clang/llvm动态编译
- 减少OS的头文件等环境依赖
- 使用btfhack多途径获取BTF文件
- 分发大小300MB -> 33MB



Prober扩展性设计

- Metrics、Event、Flow接口抽象
- cri+netns生成共享网络拓扑信息
- 基于.rodata的ebpf程序的动态插拔，变配
- 统一的数据采集分发
- 简化新增监控能力开发量

KubeSkoop 未来规划

- 增加更多云厂商和网络插件的支持
- Flow图中支持RTT、wnd等信息展示，用于发现集群中网络瓶颈
- 延迟探测支持多种协议类型、支持同时抓包分析延迟路径
- 支持采用bpftrace脚本语言扩展Prober，降低开发门槛
- 提供KubeSkoop Analysis 工具，智能分析KubeSkoop的指标和事件，降低问题理解门槛
- 应用层感知，对7层协议(http,redis等)的感知和处理

谢谢！