阿里云
奥运会官方云服务合作伙伴

网络抖动的高密解法 & BPF开发编译平台LCC

毛文安　　阿里云高级技术专家
廖肇燕　　阿里云技术专家

2021/10/24

01 什么是网络抖动?

网络抖动案例

当前抖动

当下还在发生的
可复现的抖动问题

历史抖动

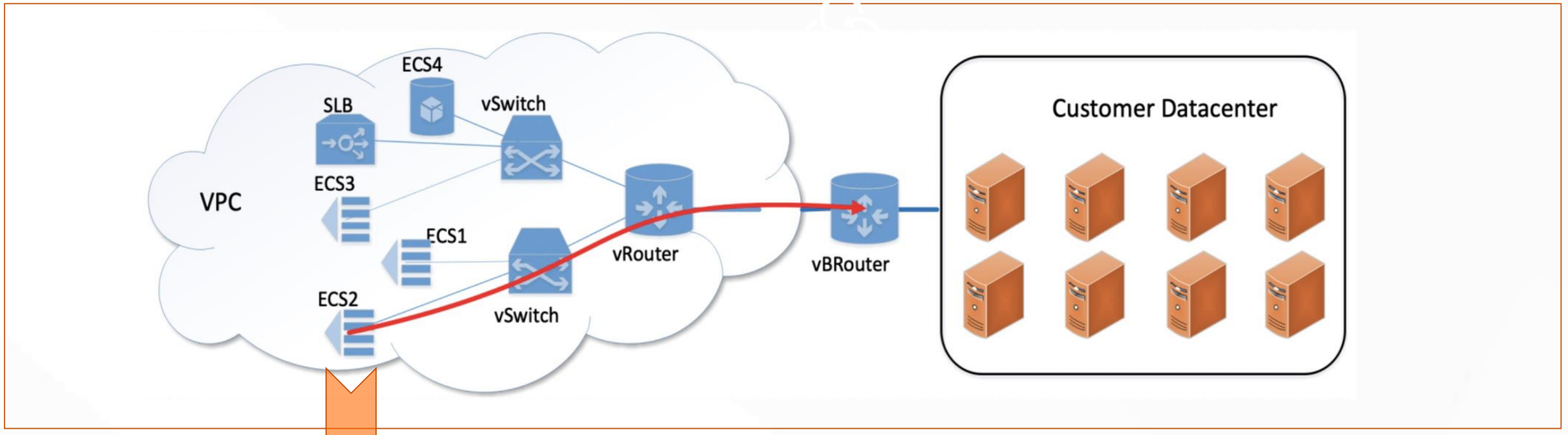
过去曾经发生的，当前
不再复现的抖动问题

ECS服务HTTP请求平均延迟正常水平维持在10ms，在某个时间点突然发生抖动，整体延迟增加至100ms，随后马上恢复。

ECS访问另一台RDS，在某个时间点突然出现大量业务日志打印的timeout，持续时间为10秒，随后马上恢复。

ECS服务的pps正常水平维持在800K，在某个时间点突然发生抖动，pps降低至400K，随后马上恢复。

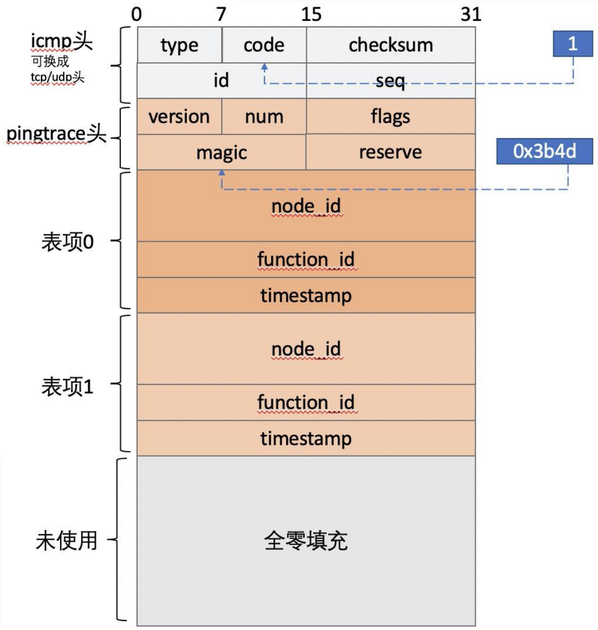
认识网络抖动

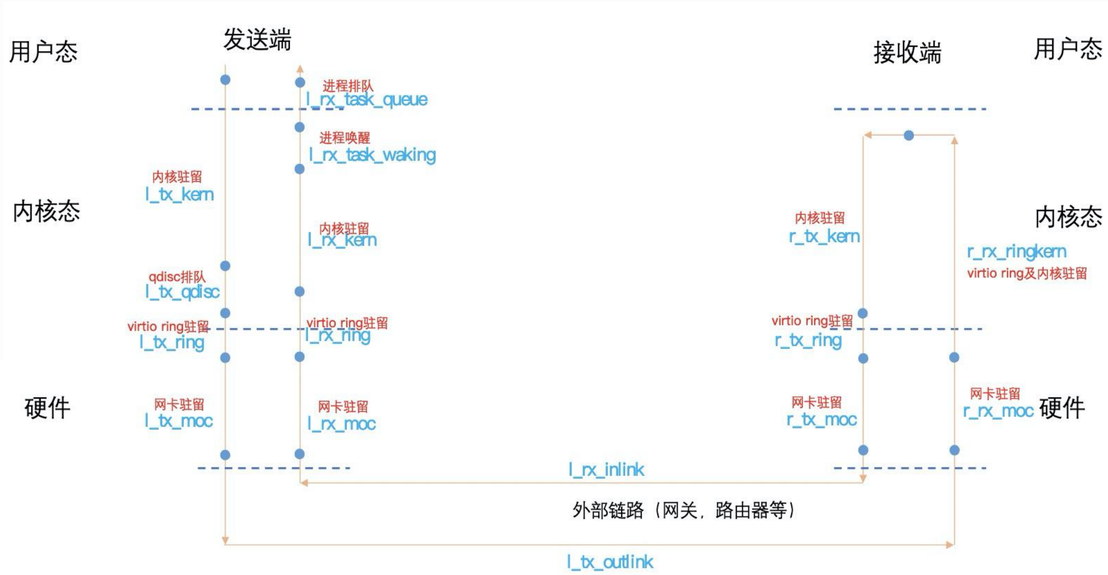


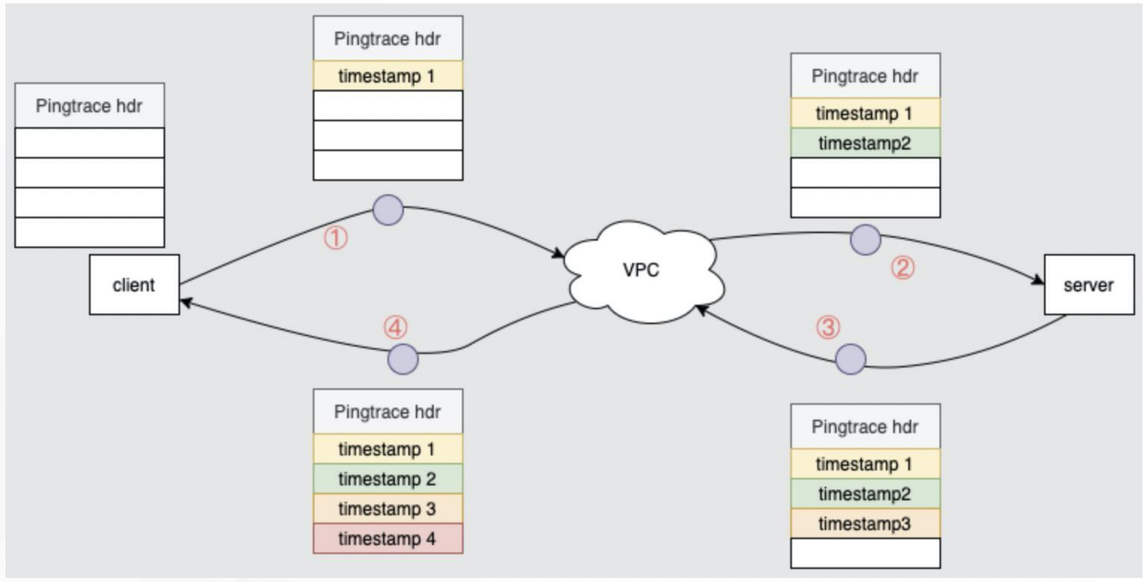
主机内部可能产生延迟的几个阶段：中断/软中断处理，唤醒进程/调度，qdisc排队等

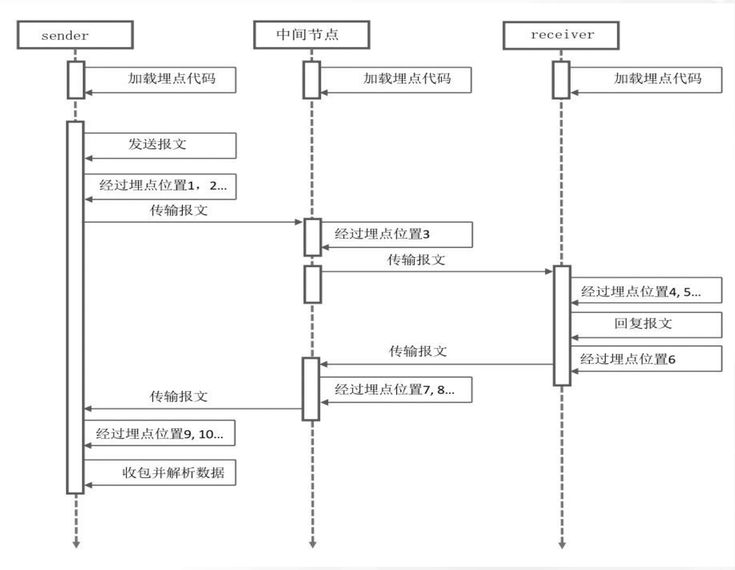
02 当前抖动的解法

Pingtrace实现

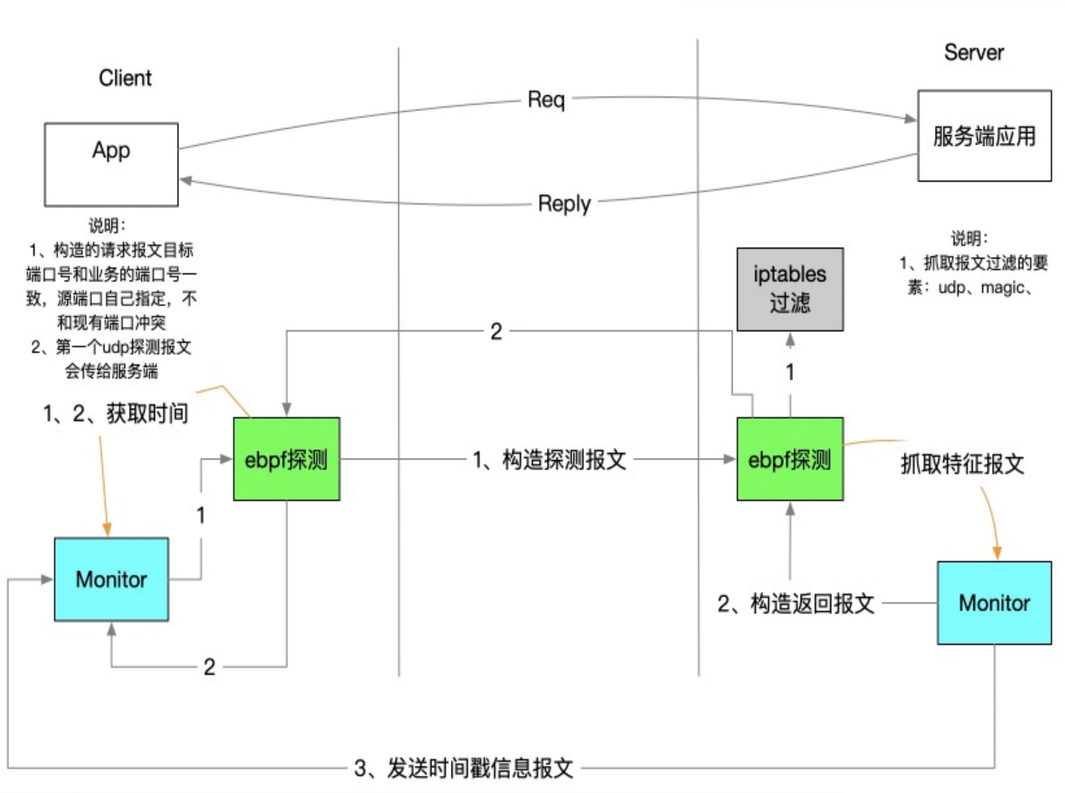




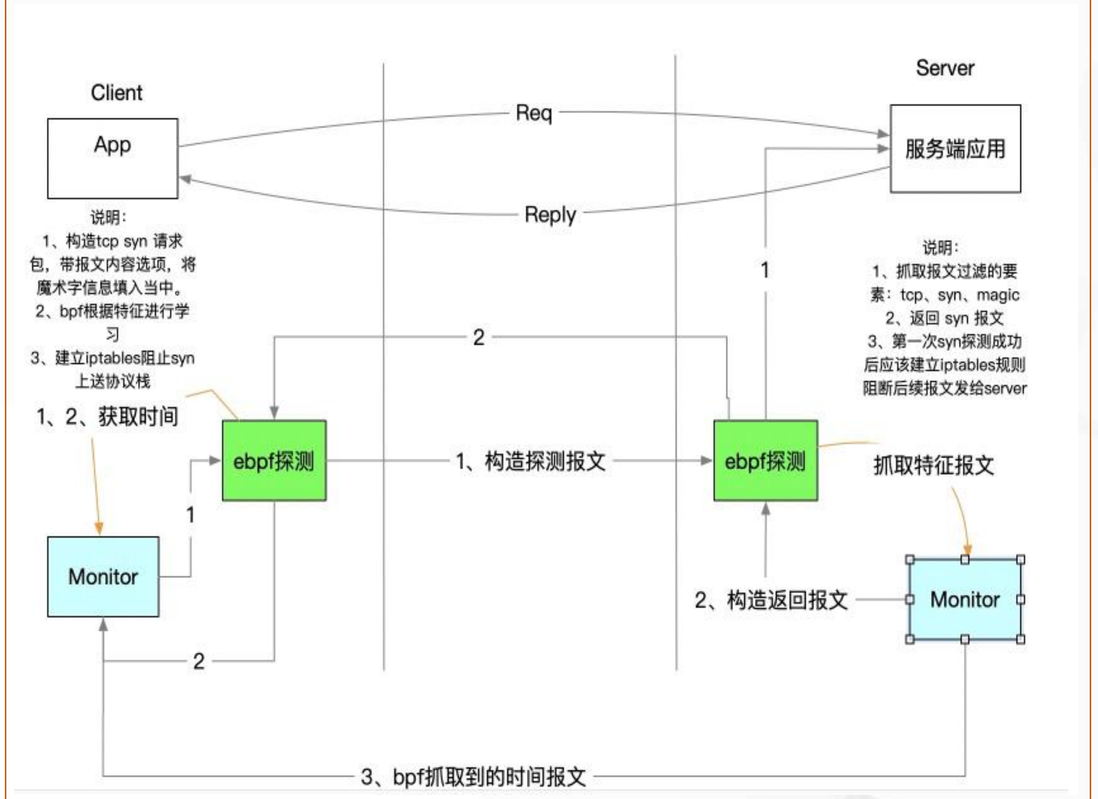




TCP&UDP pingtrace



udp探测流程



tcp探测流程

03 历史抖动的解法

历史抖动问题，难解的原因

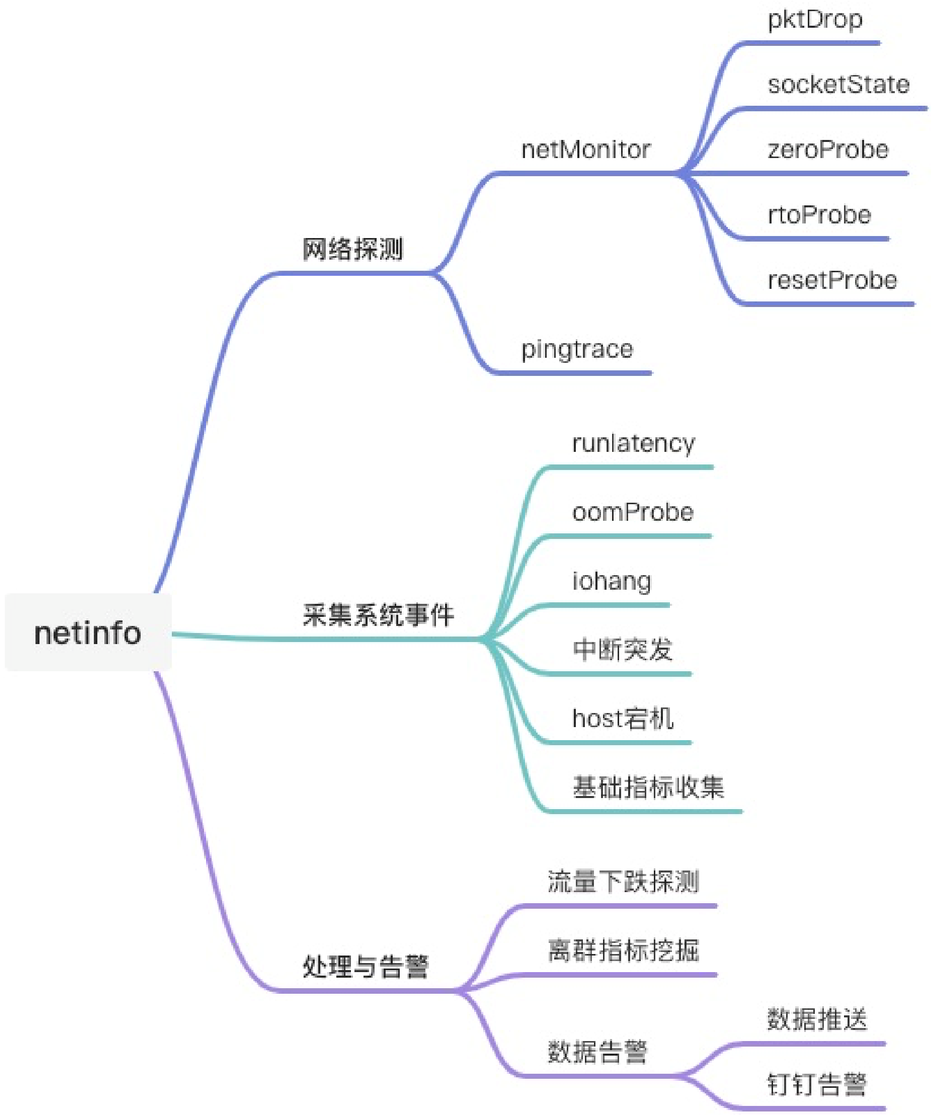
Netinfo--抖动监控利器

各项系统指标采集
NET/IO/MEM

分析业务抖动来源
确定抖动根因

指标展示和告警提示
主动运维

Netinfo--核心工具



Netinfo--连接和丢包检测

pktDrop：
　用于异常包行为监控，如buffer满、错包、丢包、路由不存在等情形；

socketState：
　socket状态监控，用于检查网络性能瓶颈等场景；

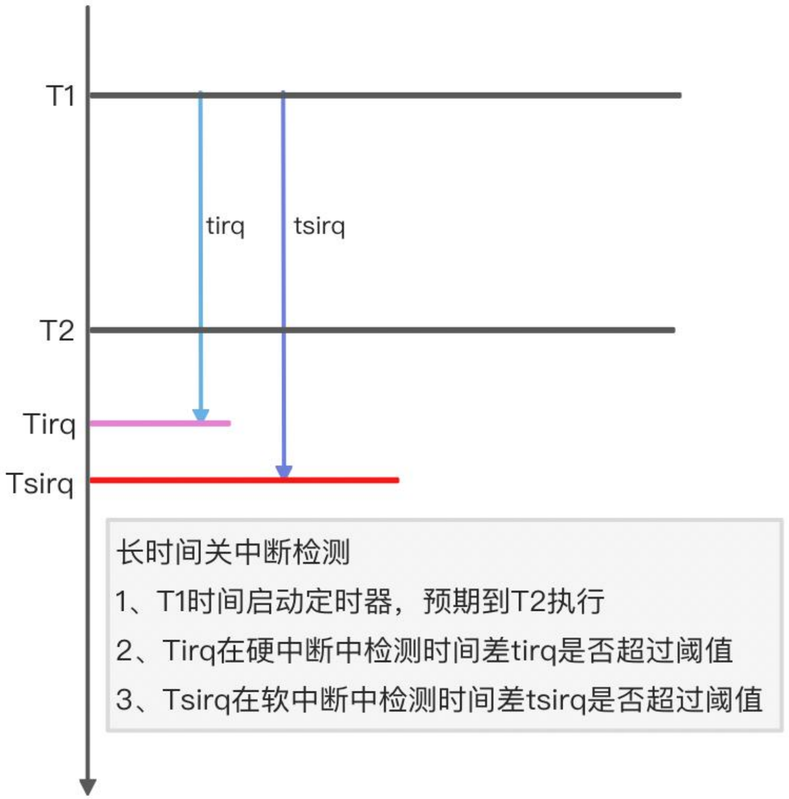
resetProbe：
　TCP rst 报文场景探测，有服务端监听套接字不存在、握手阶段不合法ACK 、错误SYN报文、TimeWait阶段发送reset、TCP状态机异常等场景；

zeroProbe：
　当tcp发生零窗口事件后，意味着接收缓冲区已满导致网络传输暂停；

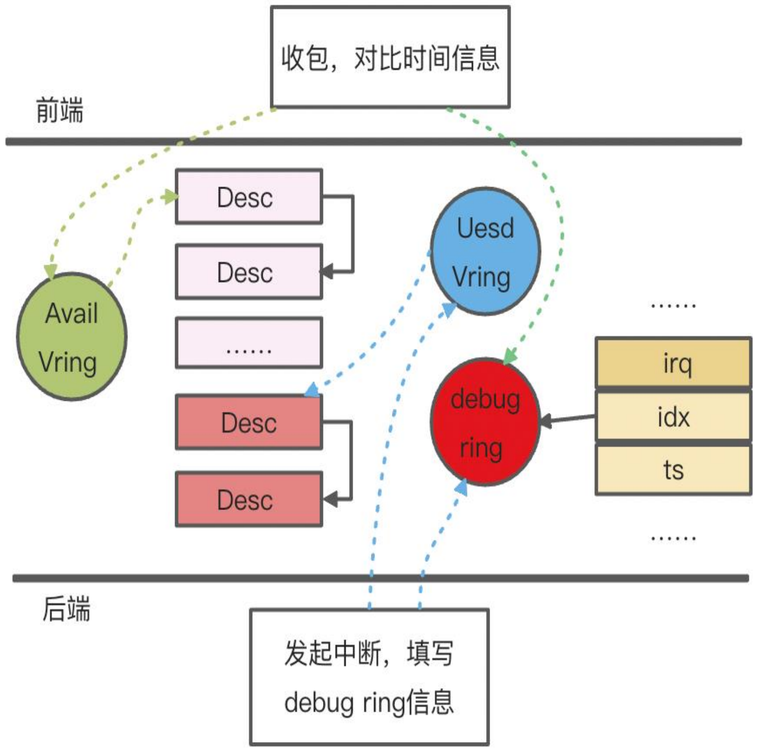
rtoProbe：
　在快速重传无法激活时，依赖RTO重传定时器来触发重传，这个定时器间隔一般为200ms，该场景对网络性能影响较大。
　可在监控点获取重传的源和目的ip、进程名等；通过tcp_sock可将连接的发送和接收信息提取出来，如snd_nxt, rcv_nxt, srtt_us等，确定buffer使用情况、连接状态、绘制srtt曲线图等。

调度和中断延迟--runlatency

调度延迟

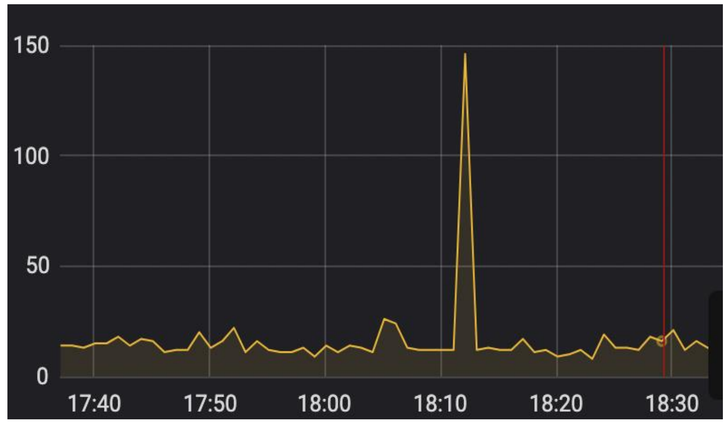


关中断检测



前后端debug ring中断检测

离群指标挖掘

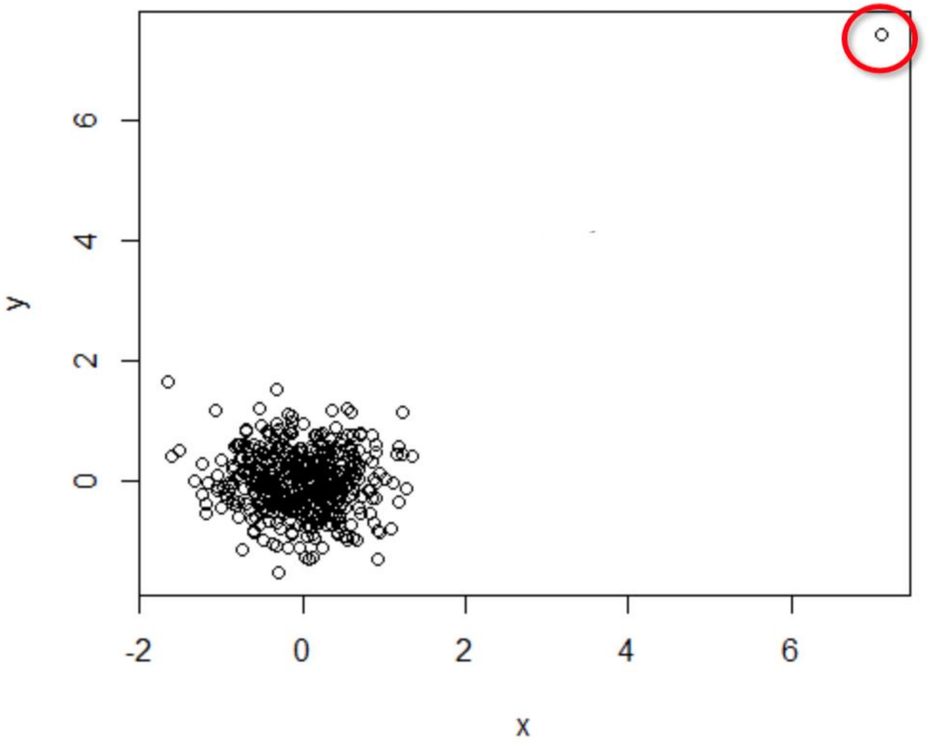


出现业务抖动毛刺，可能的关联指标有数十项，需要快速筛查

手工排查
关联指标过多，容易忽略或者出错

阈值筛查
很多场景下阈值并不固定，容易误检

离群检测
与正常的实例作离群检测，快速查出存在异常的指标



在发生抖动期间，对所有关联指标进行离群检测，以辅助快速发现抖动指标源，进而锁定分析分析根因

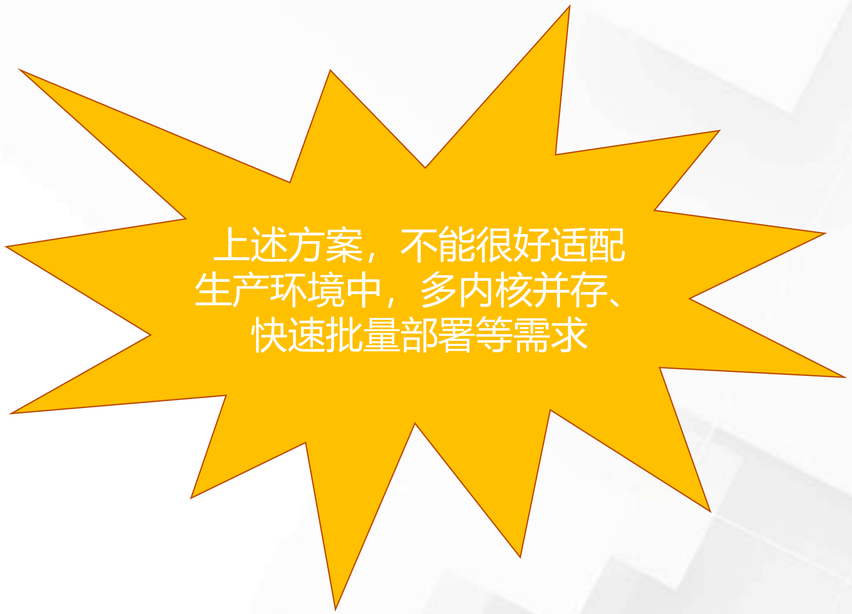
04 BPF开发编译平台LCC

BPF开发常用方案对比

原生libbpf
优势：资源占用量低
缺点：
1、需要搭建代码功能、开发效率低；
2、不同内核版本兼容性差；

BCC（bpf compile collection）
优势：开发效率高，可移植性好，支持动态修改内核部分代码
缺点：
1、部署依赖的Clang/LLVM；
2、每次运行都要执行Clang/LLVM编译，争抢内存CPU内存等资源；
3、依赖目标环境头文件；

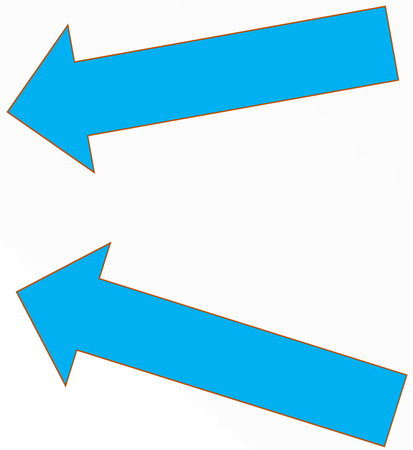
BPF CO-RE
优势：不依赖在环境中部署Clang/LLVM，资源占用少
缺点：
1、仍需要搭建编译编译工程；
2、核部分代码相对固定，无法动态配置；
3、用户态开发支持信息较少，缺乏高级语言对接；



05 什么是LCC?

LCC（Libbpf Compiler Collection）技术优势

LCC
以bpf CO-RE为基础，融合了资源占用低、可移植性强、动态编译等优点



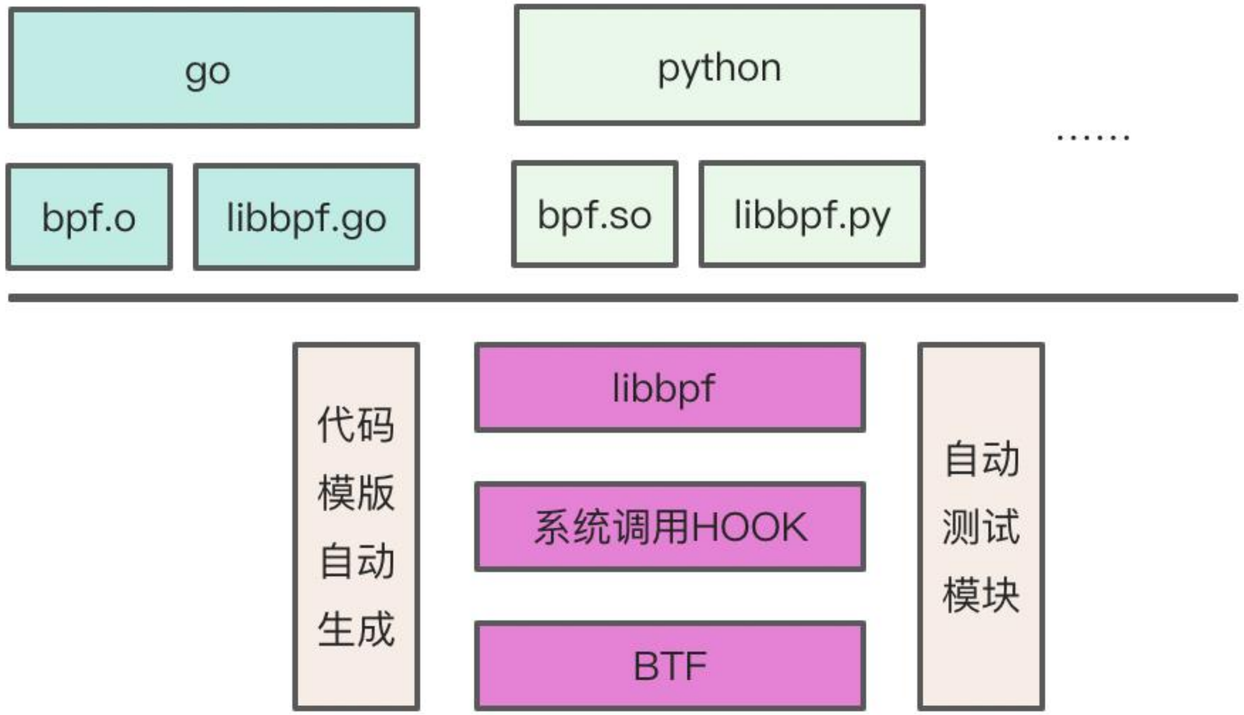
BCC
优势：开发效率高，可移植性好，支持动态修改内核部分代码

BPF CO-RE
优势：不依赖在环境中部署Clang/LLVM，资源占用少

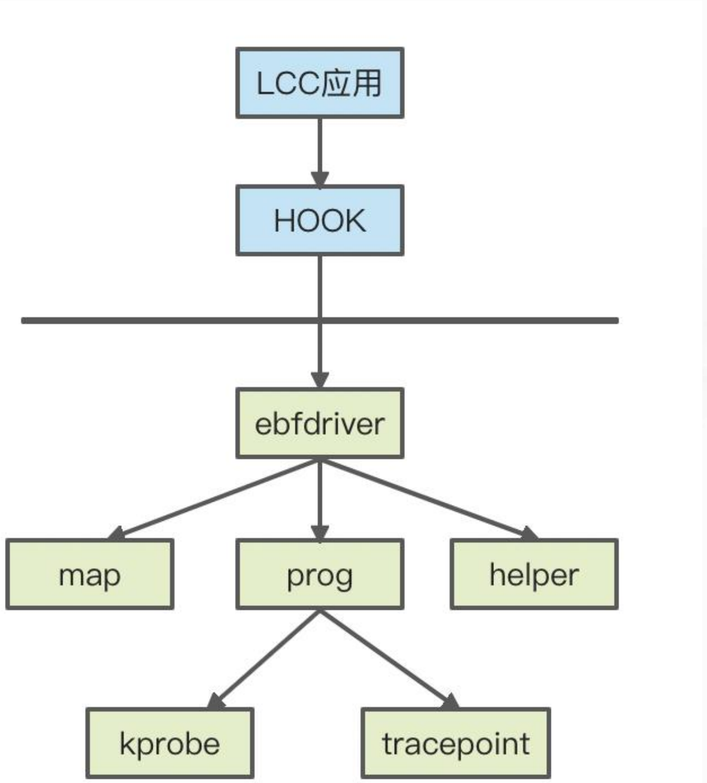
方案优势：
1、开箱即用：内核侧仅提供bpf.c即可，完全剥离出内核编译工程；
2、复用编译成果：本地侧无编译过程，不存在包依赖和CPU/内存等资源消耗问题；
3、自适应不同版本差异：更适合在集群多内核版本共存的场景；

LCC框架

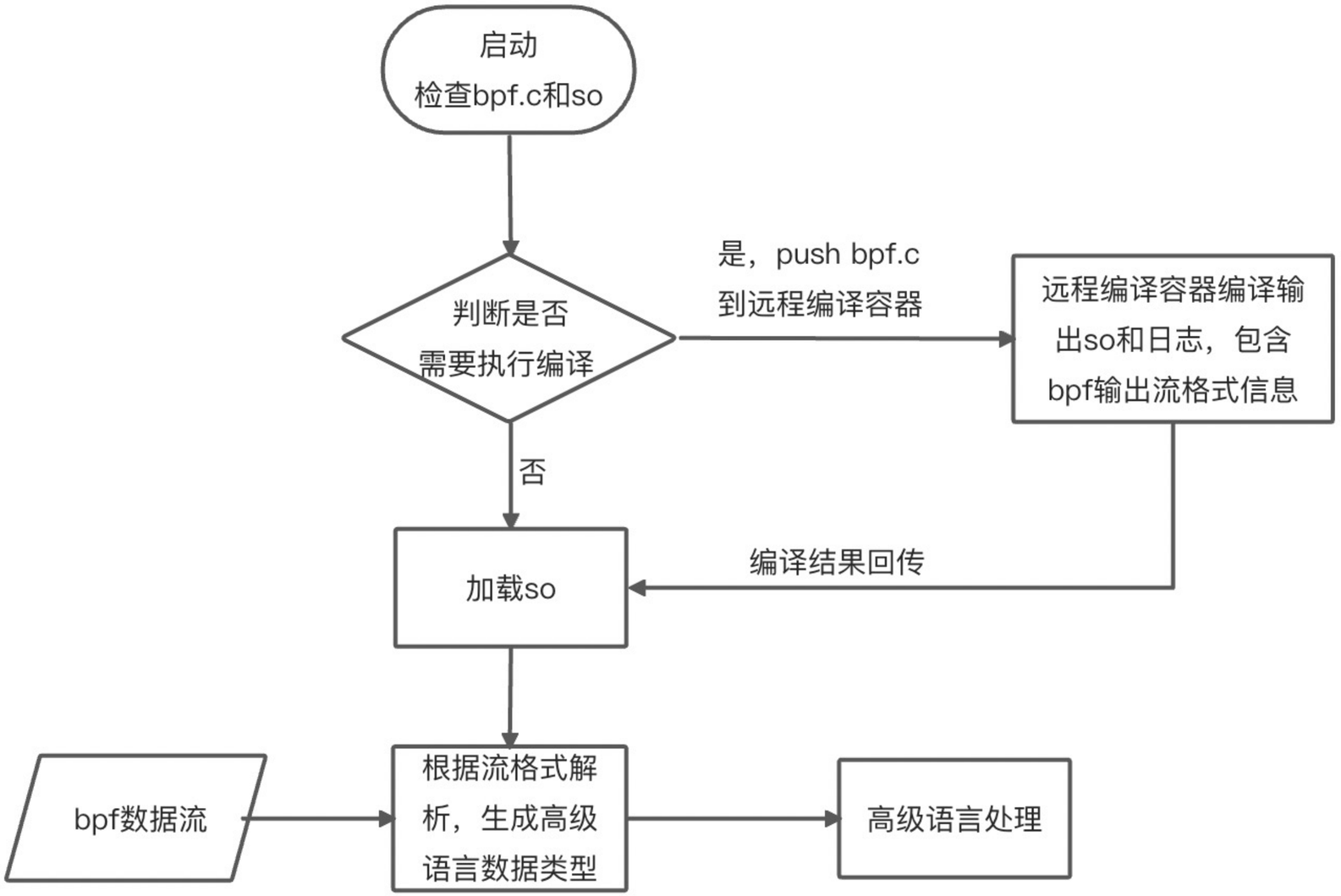
CORE实现



3.10支持eBPF



LCC 实现流程


启动
检查bpf.c和so
判断是否
需要执行编译
是，push bpf.c
到远程编译容器
远程编译容器编译输
出so和日志，包含
bpf输出流格式信息
否
加载so
编译结果回传
bpf数据流
根据流格式解
析，生成高级
语言数据类型
高级语言处理

LCC典型应用场景

以netinfo在生产集群部署为例，监控内存OOM事件，同样oom_kill_process符号，3.10和4.19 入参并不一致。

```
/kernel-3.10.0-327 /mm/
D       oom_kill.c      412  void oom_kill_process(struct task_struct *p, gfp_t gfp_mask
/linux-4.19.91-19.1. x86_64/mm/
D       oom_kill.c      960  static void oom_kill_process(struct oom_control *oc, const
```

如果要在一个发行包中可以兼容上述版本，那就需要包含两个二进制来兼容上述不同内核。

采用LCC方案：在发行包中只提供bpf.c，LCC发起远程编译后，编译服务器会根据不同内核版本返回对应的编译结果，方便在生产环境中批量部署；

同样的，对网络性能影响较大的RTO重传及窗口信息、零窗口探测等信息获取，均可采用同一套代码来进行管理，无需针对不同版本发布不同的二进制。

06 开源计划

开源计划

Netinfo和LCC计划在龙蜥社区开源，扫码可以加入系统运维SIG，参与代码贡献和技术讨论

WWW.ALIYUN.COM