第二届 eBPF开发者大会

www.ebpftravel.com

基于eBPF的函数调用栈观测工具

西安邮电大学--刘冰

中国·西安

个人简介

刘冰，就读于西安邮电大学计算机学院，研究生二年级，曾获得2019全国大学生数学建模竞赛省奖，2023年全国大学计算机系统能力大赛国奖，2023年CCF开源夏令营优秀项目奖

0、背景

1、高效的数据组织方式

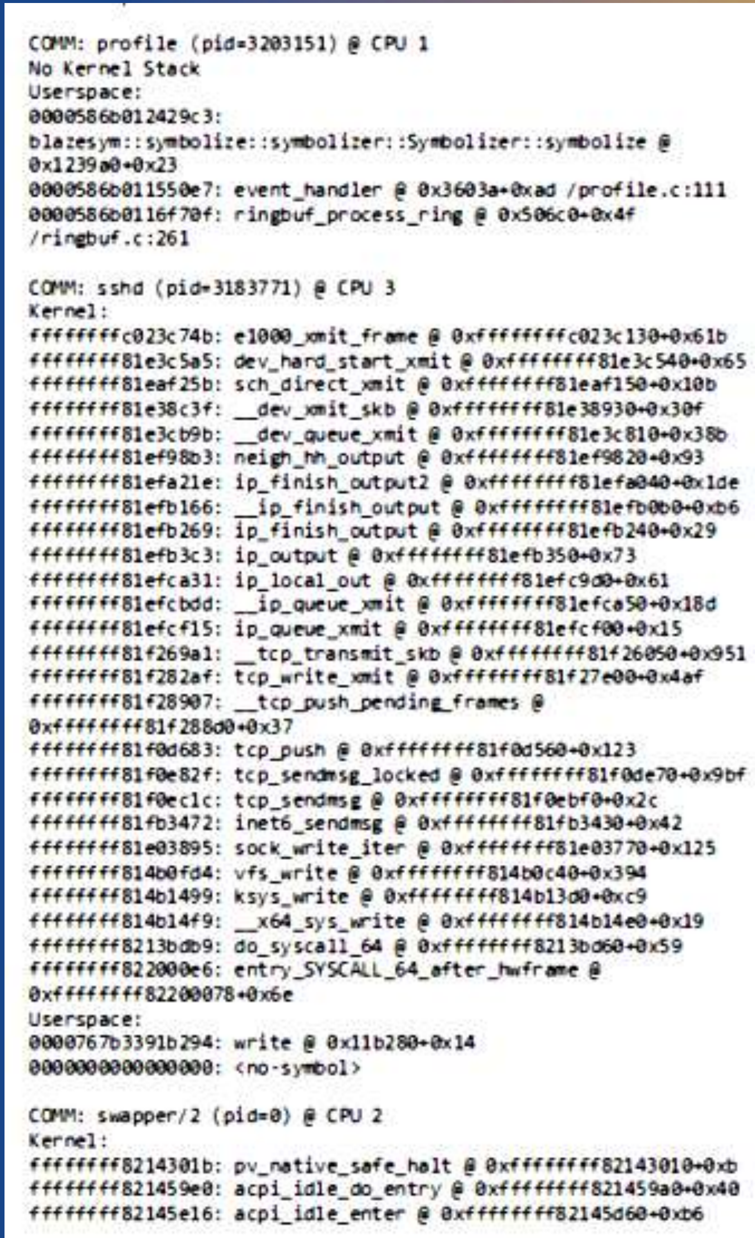
2、PSI触发及及时采集

3、多指标关联

4、持续性观测和长期存储

0、使用不便

- 相关工具繁杂，问题不一
  - profile用于on_cpu调用栈采样，只能设置采样频率，功能单一
  - offcputime用于采集off_cpu调用栈，无法实时显示数据
  - memleak用于采集内存泄露调用栈，无法正常显示用户态调用栈，无法有效执行子进程
  - ...
- 不便于分析
  - 上述等大多数工具都会将采集的数据按调用栈进行输出，篇幅大，不便于直接查阅和分析



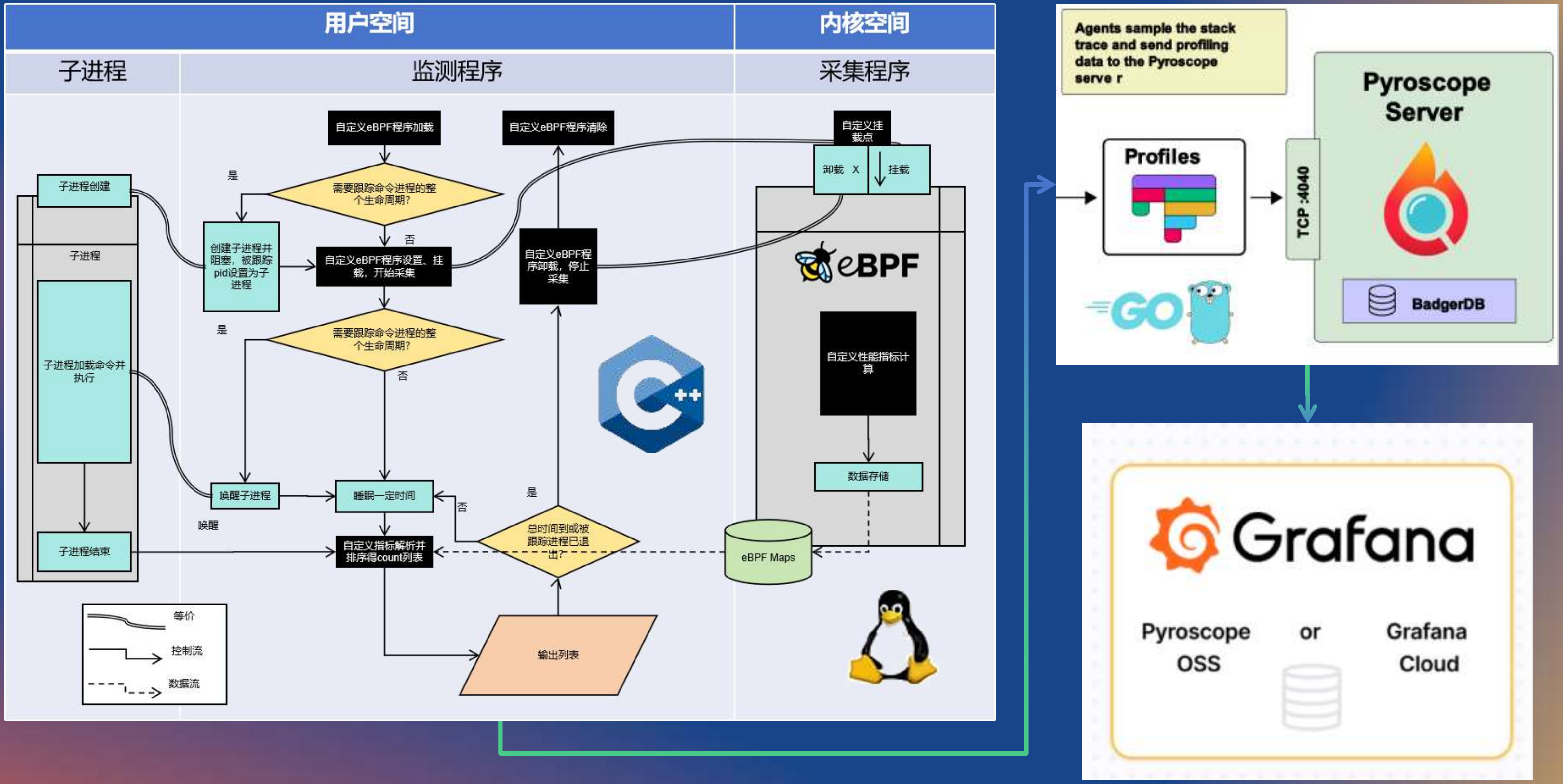
```
COMM: profile (pid=3203151) @ CPU 1
No Kernel Stack
Userspace:
0000586b012429c3:
blazesym::symbolize::symbolizer::Symbolizer::symbolize @
0x1239a0+0x23
0000586b011550e7: event_handler @ 0x3603a+0xad /profile.c:111
0000586b0116f70f: ringbuf_process_ring @ 0x506c0+0x4f
/ringbuf.c:261

COMM: sshd (pid=3183771) @ CPU 3
Kernel:
ffffffffc023c74b: e1000_xmit_frame @ 0xffffffffc023c130+0x61b
ffffffff81e3c5a5: dev_hard_start_xmit @ 0xffffffff81e3c540+0x65
ffffffff81eaf25b: sch_direct_xmit @ 0xffffffff81eaf150+0x10b
ffffffff81e38c3f: __dev_xmit_skb @ 0xffffffff81e38930+0x30f
ffffffff81e3cb9b: __dev_queue_xmit @ 0xffffffff81e3c810+0x38b
ffffffff81ef98b3: neigh_hh_output @ 0xffffffff81ef9820+0x93
ffffffff81efa21e: ip_finish_output2 @ 0xffffffff81efa040+0x1de
ffffffff81efb166: __ip_finish_output @ 0xffffffff81efb0b0+0xb6
ffffffff81efb269: ip_finish_output @ 0xffffffff81efb240+0x29
ffffffff81efb3c3: ip_output @ 0xffffffff81efb350+0x73
ffffffff81efca31: ip_local_out @ 0xffffffff81efc9d0+0x61
ffffffff81efcbdd: __ip_queue_xmit @ 0xffffffff81efca50+0x18d
ffffffff81efcf15: ip_queue_xmit @ 0xffffffff81efcf00+0x15
ffffffff81f269a1: __tcp_transmit_skb @ 0xffffffff81f26050+0x951
ffffffff81f282af: tcp_write_xmit @ 0xffffffff81f27e00+0x4af
ffffffff81f28907: __tcp_push_pending_frames @
0xffffffff81f288d0+0x37
ffffffff81f0d683: tcp_push @ 0xffffffff81f0d560+0x123
ffffffff81f0e82f: tcp_sendmsg_locked @ 0xffffffff81f0de70+0x9bf
ffffffff81f0ec1c: tcp_sendmsg @ 0xffffffff81f0ebf0+0x2c
ffffffff81fb3472: inet6_sendmsg @ 0xffffffff81fb3430+0x42
ffffffff81e03895: sock_write_iter @ 0xffffffff81e03770+0x125
ffffffff814b0fd4: vfs_write @ 0xffffffff814b0c40+0x394
ffffffff814b1499: ksys_write @ 0xffffffff814b13d0+0xc9
ffffffff814b14f9: __x64_sys_write @ 0xffffffff814b14e0+0x19
ffffffff8213bdb9: do_syscall_64 @ 0xffffffff8213bd60+0x59
ffffffff822000e6: entry_SYSCALL_64_after_hwframe @
0xffffffff82200078+0x6e
Userspace:
0000767b3391b294: write @ 0x11b280+0x14
0000000000000000: <no-symbol>

COMM: swapper/2 (pid=0) @ CPU 2
Kernel:
ffffffff8214301b: pv_native_safe_halt @ 0xffffffff82143010+0xb
ffffffff821459e0: acpi_idle_do_entry @ 0xffffffff821459a0+0x40
ffffffff82145e16: acpi_idle_enter @ 0xffffffff82145d60+0xb6
```

0、性能消耗大

- ! 数据量大
- ! IO损耗高

例如perf工具或者基于eBPF的profile工具，其会输出每个调用栈样本，会造成频繁输出，并产生大量数据。

本项目设计了一个统一的框架，主要针对程序性能瓶颈问题，来进行基于eBPF的调用栈采集


用户空间
内核空间
子进程
监测程序
采集程序
自定义eBPF程序加载
自定义eBPF程序清除
自定义挂载点
卸载 X
挂载
是
需要跟踪命令进程的整个生命周期?
子进程创建
子进程
创建子进程并阻塞，被跟踪pid设置为子进程
否
自定义eBPF程序设置、挂载，开始采集
自定义eBPF程序卸载，停止采集
eBPF
是
需要跟踪命令进程的整个生命周期?
否
子进程加载命令并执行
自定义性能指标计算
数据存储
唤醒子进程
睡眠一定时间
否
是
唤醒
总时间到或被跟踪进程已退出?
自定义指标解析并排序得count列表
eBPF Maps
子进程结束
等价
控制流
数据流
输出列表
Agents sample the stack trace and send profiling data to the Pyroscope serve r
Profiles
TCP :4040
Pyroscope Server
GO
BadgerDB
Grafana
Pyroscope OSS
or
Grafana Cloud

0、背景

1、友好的数据组织方式

2、PSI触发及及时采集

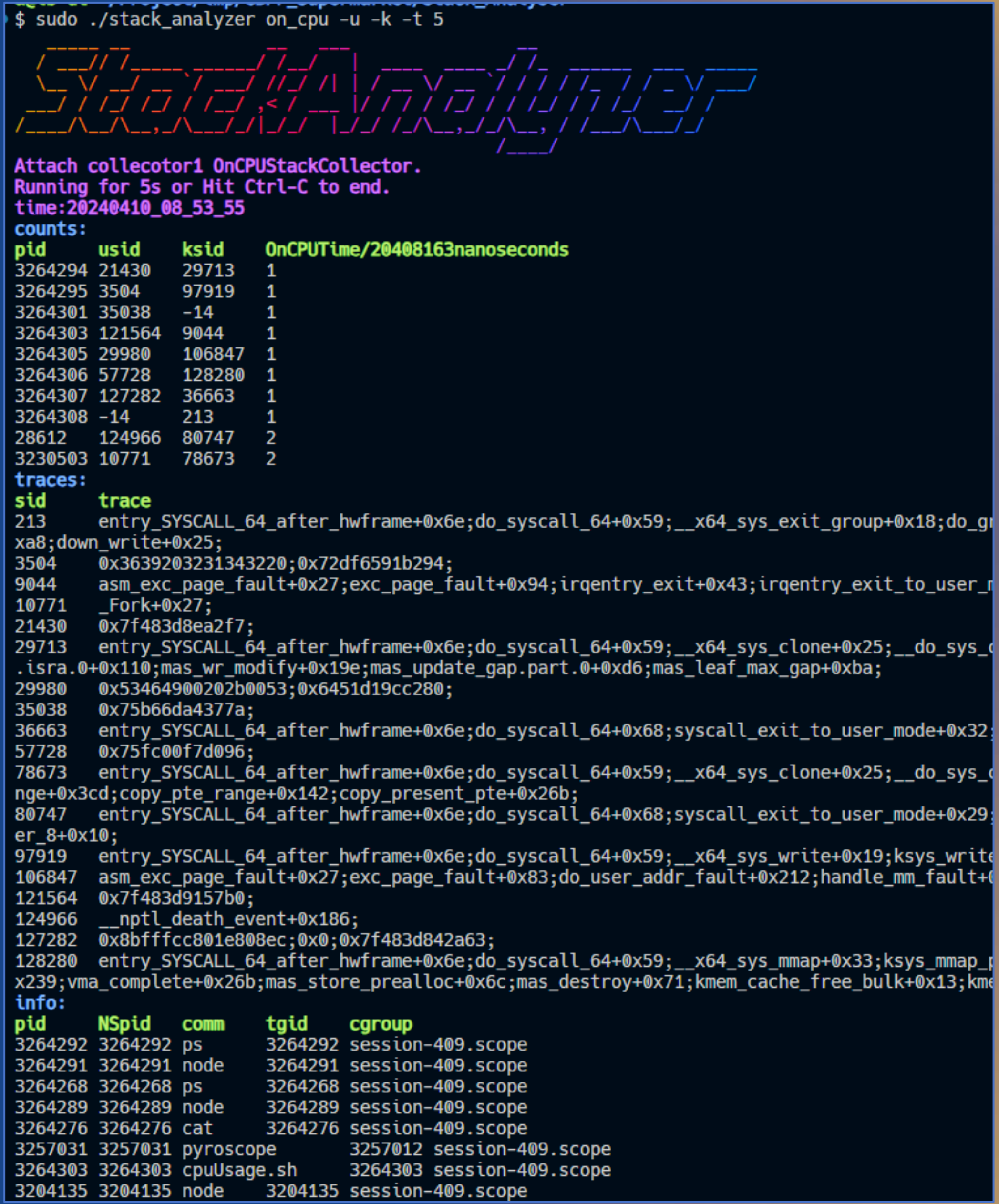
3、多指标关联

4、持续性观测和长期存储

1、本地拆分展示

默认49hz采集样本 输出Top10调用栈

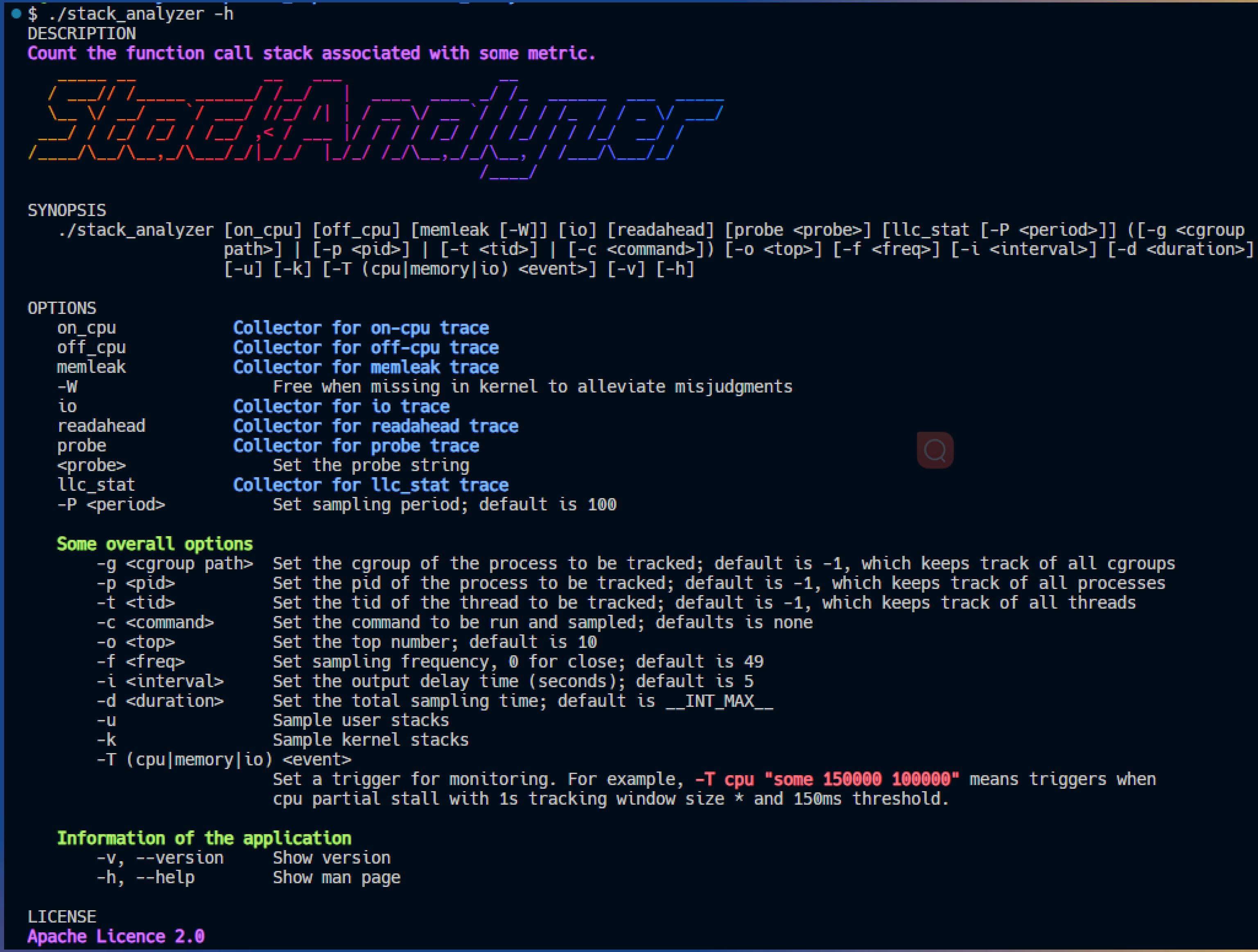
- ✓ 第一个表为pid，用户栈id，内核栈id及其关联指标
  按指标值进行排序，便于找出高消耗进程和其瓶颈
  调用栈的id
- ✓ 第二个表为栈id和栈，可根据栈id找到调用栈，
  查看函数调用关系
- ✓ 第三个表为pid、命名空间pid、命令名、
  命名空间tgid和cgroup id，由此可看出pid所属的
  组，容器等信息，有利于进一步回溯



```
$ sudo ./stack_analyzer on_cpu -u -k -t 5

Attach collecotor1 OnCPUStackCollector.
Running for 5s or Hit Ctrl-C to end.
time:20240410_08_53_55
counts:
pid     usid    ksid    OnCPUTime/20408163nanoseconds
3264294 21430   29713   1
3264295 3504    97919   1
3264301 35038   -14     1
3264303 121564  9044    1
3264305 29980   106847  1
3264306 57728   128280  1
3264307 127282  36663   1
3264308 -14     213     1
28612   124966  80747   2
3230503 10771   78673   2
traces:
sid     trace
213     entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_exit_group+0x18;do_gr
xa8;down_write+0x25;
3504    0x3639203231343220;0x72df6591b294;
9044    asm_exc_page_fault+0x27;exc_page_fault+0x94;irqentry_exit+0x43;irqentry_exit_to_user_m
10771   _Fork+0x27;
21430   0x7f483d8ea2f7;
29713   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_clone+0x25;__do_sys_c
.isra.0+0x110;mas_wr_modify+0x19e;mas_update_gap.part.0+0xd6;mas_leaf_max_gap+0xba;
29980   0x53464900202b0053;0x6451d19cc280;
35038   0x75b66da4377a;
36663   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x68;syscall_exit_to_user_mode+0x32;
57728   0x75fc00f7d096;
78673   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_clone+0x25;__do_sys_c
nge+0x3cd;copy_pte_range+0x142;copy_present_pte+0x26b;
80747   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x68;syscall_exit_to_user_mode+0x29;
er_8+0x10;
97919   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_write+0x19;ksys_write
106847  asm_exc_page_fault+0x27;exc_page_fault+0x83;do_user_addr_fault+0x212;handle_mm_fault+0
121564  0x7f483d9157b0;
124966  __nptl_death_event+0x186;
127282  0x8bfffcc801e808ec;0x0;0x7f483d842a63;
128280  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_mmap+0x33;ksys_mmap_p
x239;vma_complete+0x26b;mas_store_prealloc+0x6c;mas_destroy+0x71;kmem_cache_free_bulk+0x13;kme
info:
pid     NSpid   comm    tgid    cgroup
3264292 3264292 ps      3264292 session-409.scope
3264291 3264291 node    3264291 session-409.scope
3264268 3264268 ps      3264268 session-409.scope
3264289 3264289 node    3264289 session-409.scope
3264276 3264276 cat     3264276 session-409.scope
3257031 3257031 pyroscope       3257012 session-409.scope
3264303 3264303 cpuUsage.sh     3264303 session-409.scope
3204135 3204135 node    3204135 session-409.scope
```

1、精准数据采集

- ✓ 按cgroup过滤
- ✓ 按pid过滤
- ✓ 按tid过滤
- ✓ 创建指定命令进程并采集

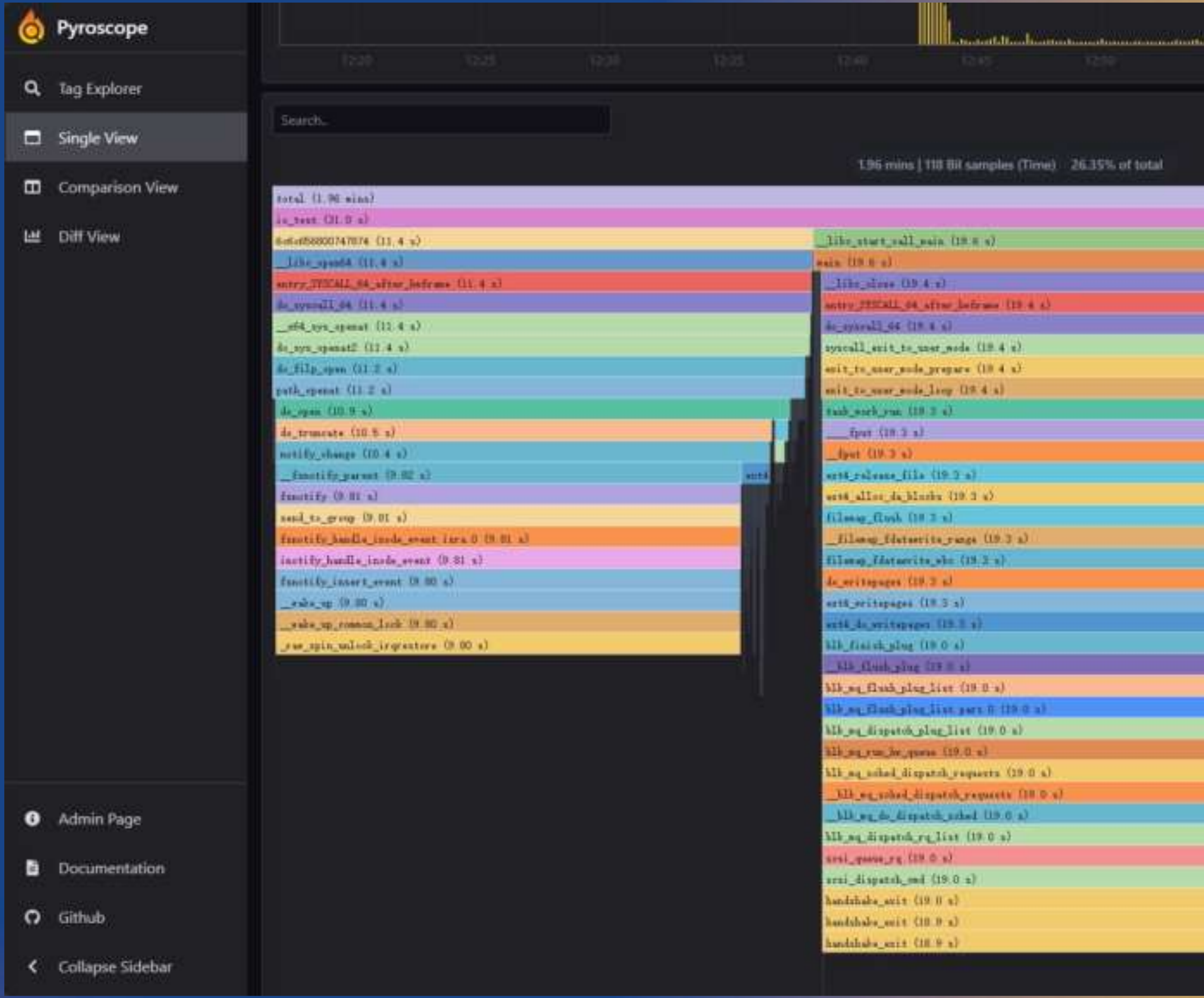


```
$ ./stack_analyzer -h
DESCRIPTION
Count the function call stack associated with some metric.
    _____ __             __   ___                __
   / ___// /_____ ______/ /__/   |  ____  ____ _/ /_  ______  ___  _____
   \__ \/ __/ __ `/ ___/ //_/ /| | / __ \/ __ `/ / / / /_  / / _ \/ ___/
  ___/ / /_/ /_/ / /__/ ,< / ___ |/ / / / /_/ / / /_/ / / /_/  __/ /
 /____/\__/\__,_/\___/_/|_/_/  |_/_/ /_/\__,_/_/\__, / /___/\___/_/
                                               /____/

SYNOPSIS
   ./stack_analyzer [on_cpu] [off_cpu] [memleak [-W]] [io] [readahead] [probe <probe>] [llc_stat [-P <period>]] ([-g <cgroup
                    path>] | [-p <pid>] | [-t <tid>] | [-c <command>]) [-o <top>] [-f <freq>] [-i <interval>] [-d <duration>]
                    [-u] [-k] [-T (cpu|memory|io) <event>] [-v] [-h]

OPTIONS
   on_cpu                Collector for on-cpu trace
   off_cpu               Collector for off-cpu trace
   memleak               Collector for memleak trace
   -W                        Free when missing in kernel to alleviate misjudgments
   io                    Collector for io trace
   readahead             Collector for readahead trace
   probe                 Collector for probe trace
   <probe>                   Set the probe string
   llc_stat              Collector for llc_stat trace
   -P <period>               Set sampling period; default is 100

   Some overall options
       -g <cgroup path>      Set the cgroup of the process to be tracked; default is -1, which keeps track of all cgroups
       -p <pid>              Set the pid of the process to be tracked; default is -1, which keeps track of all processes
       -t <tid>              Set the tid of the thread to be tracked; default is -1, which keeps track of all threads
       -c <command>          Set the command to be run and sampled; defaults is none
       -o <top>              Set the top number; default is 10
       -f <freq>             Set sampling frequency, 0 for close; default is 49
       -i <interval>         Set the output delay time (seconds); default is 5
       -d <duration>         Set the total sampling time; default is __INT_MAX__
       -u                    Sample user stacks
       -k                    Sample kernel stacks
       -T (cpu|memory|io) <event>
                             Set a trigger for monitoring. For example, -T cpu "some 150000 100000" means triggers when
                             cpu partial stall with 1s tracking window size * and 150ms threshold.

   Information of the application
       -v, --version         Show version
       -h, --help            Show man page

LICENSE
Apache Licence 2.0
```

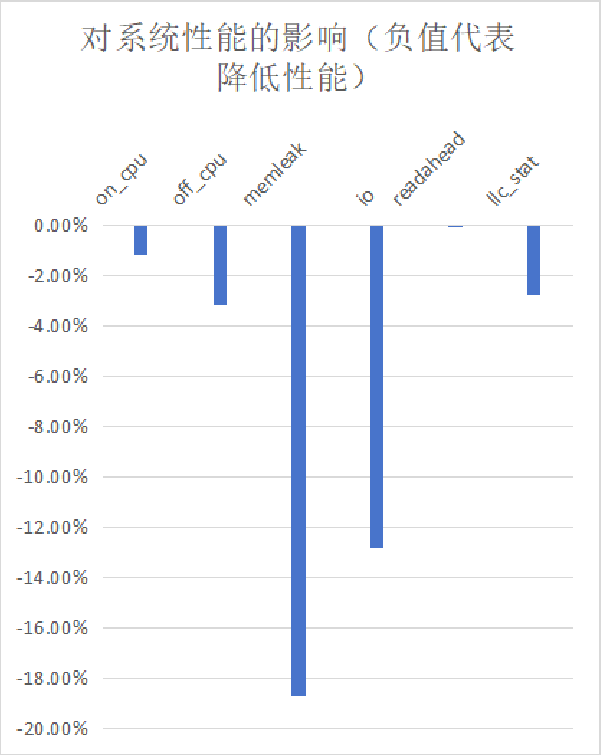
1、服务端可视化

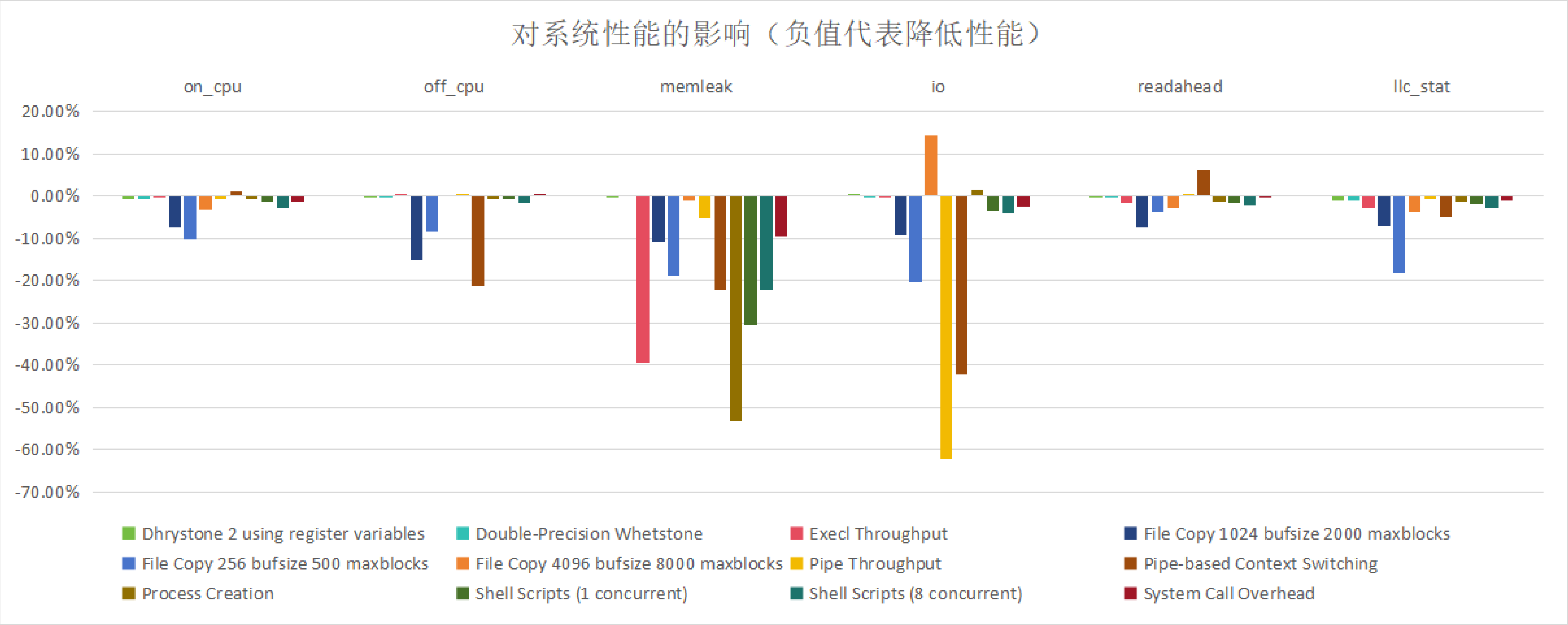
兼容grafana pyroscope

- ✓ 本项目的发送器将采集数据发送到pyroscope服务端
- ✓ 有指标时序图、调用栈火焰图和函数占比列表三种可视化方案
- ✓ 支持对比视图和差分视图


Pyroscope
Tag Explorer
Single View
Comparison View
Diff View
Admin Page
Documentation
Github
Collapse Sidebar
12:20
12:25
12:30
12:35
12:40
12:45
12:50
Search...
1.96 mins | 118 Bil samples (Time)
26.35% of total
total (1.96 mins)
io_test (31.0 s)
0x6e6f58800747874 (11.4 s)
__libc_open64 (11.4 s)
entry_SYSCALL_64_after_hwframe (11.4 s)
do_syscall_64 (11.4 s)
__x64_sys_openat (11.4 s)
do_sys_openat2 (11.4 s)
do_filp_open (11.2 s)
path_openat (11.2 s)
do_open (10.9 s)
do_truncate (10.5 s)
notify_change (10.4 s)
__fsnotify_parent (9.82 s)
ext4
fsnotify (9.81 s)
send_to_group (9.81 s)
fsnotify_handle_inode_event.isra.0 (9.81 s)
inotify_handle_inode_event (9.81 s)
fsnotify_insert_event (9.80 s)
__wake_up (9.80 s)
__wake_up_common_lock (9.80 s)
_raw_spin_unlock_irqrestore (9.80 s)
__libc_start_call_main (19.6 s)
main (19.6 s)
__libc_close (19.4 s)
entry_SYSCALL_64_after_hwframe (19.4 s)
do_syscall_64 (19.4 s)
syscall_exit_to_user_mode (19.4 s)
exit_to_user_mode_prepare (19.4 s)
exit_to_user_mode_loop (19.4 s)
task_work_run (19.3 s)
____fput (19.3 s)
__fput (19.3 s)
ext4_release_file (19.3 s)
ext4_alloc_da_blocks (19.3 s)
filemap_flush (19.3 s)
__filemap_fdatawrite_range (19.3 s)
filemap_fdatawrite_wbc (19.3 s)
do_writepages (19.3 s)
ext4_writepages (19.3 s)
ext4_do_writepages (19.3 s)
blk_finish_plug (19.0 s)
__blk_flush_plug (19.0 s)
blk_mq_flush_plug_list (19.0 s)
blk_mq_flush_plug_list.part.0 (19.0 s)
blk_mq_dispatch_plug_list (19.0 s)
blk_mq_run_hw_queue (19.0 s)
blk_mq_sched_dispatch_requests (19.0 s)
__blk_mq_sched_dispatch_requests (19.0 s)
__blk_mq_do_dispatch_sched (19.0 s)
blk_mq_dispatch_rq_list (19.0 s)
scsi_queue_rq (19.0 s)
scsi_dispatch_cmd (19.0 s)
handshake_exit (19.0 s)
handshake_exit (18.9 s)
handshake_exit (18.9 s)

1、性能测试


对系统性能的影响（负值代表降低性能）
on_cpu
off_cpu
memleak
io
readahead
llc_stat
0.00%
-2.00%
-4.00%
-6.00%
-8.00%
-10.00%
-12.00%
-14.00%
-16.00%
-18.00%
-20.00%


对系统性能的影响（负值代表降低性能）
on_cpu
off_cpu
memleak
io
readahead
llc_stat
20.00%
10.00%
0.00%
-10.00%
-20.00%
-30.00%
-40.00%
-50.00%
-60.00%
-70.00%
Dhrystone 2 using register variables
Double-Precision Whetstone
Execl Throughput
File Copy 1024 bufsize 2000 maxblocks
File Copy 256 bufsize 500 maxblocks
File Copy 4096 bufsize 8000 maxblocks
Pipe Throughput
Pipe-based Context Switching
Process Creation
Shell Scripts (1 concurrent)
Shell Scripts (8 concurrent)
System Call Overhead

0、背景

1、高效的数据组织方式

2、PSI触发及及时采集

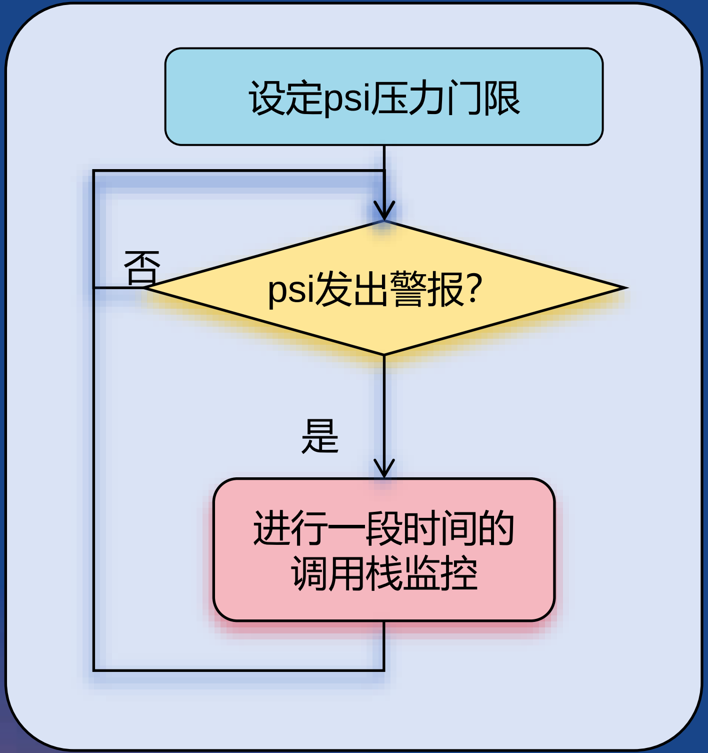
3、多指标关联

4、持续性观测和长期存储

2、PSI触发

✓ PSI报警后监控

本系统可基于psi实现动态的负载监控，着眼与高阻塞环境，节省不必要的监控开销。



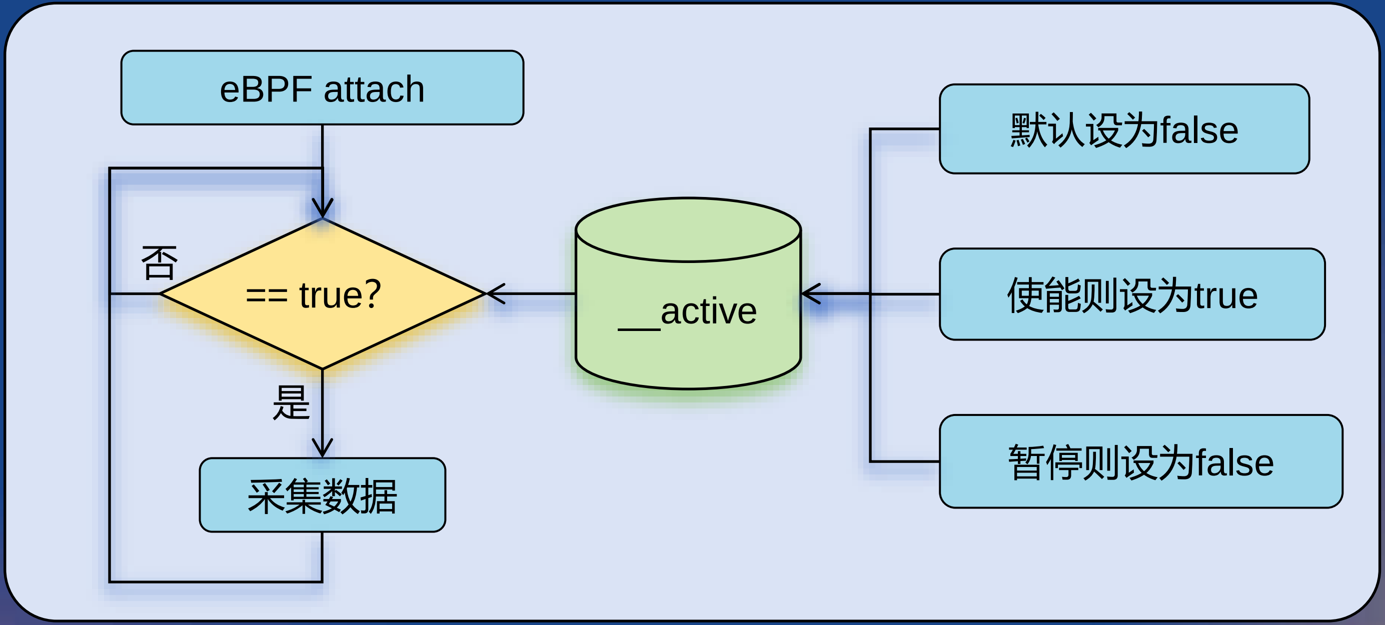
2、高阻塞环境下及时触发

✓ 核心思想：

eBPF attach需要进行系统调用，在高阻塞环境可能会使数据采集不及时，这里使用eBPF 全局变量进行采集控制

✓ 优点：

减小性能损耗
及时地数据采集



1、 背景

2、 PSI触发及及时采集

3、 多指标关联

4、 持续性观测及长期存储

3、增加数据复用性

✓关联多指标的监测功能

按进程调用栈，统计IO操作的次数和数据量

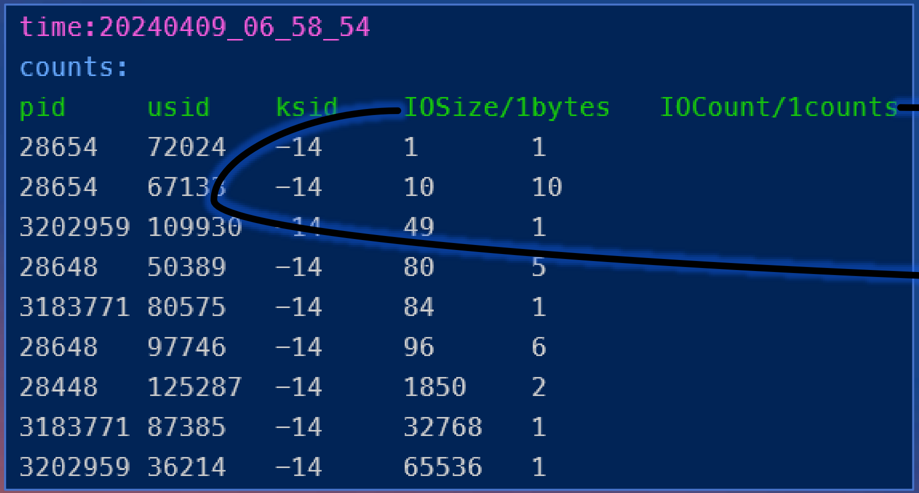
按进程调用栈，统计缓存失配次数、缓存命中次数以及缓存命中率

按进程调用栈，统计内存泄露大小、内存分配未释放次数

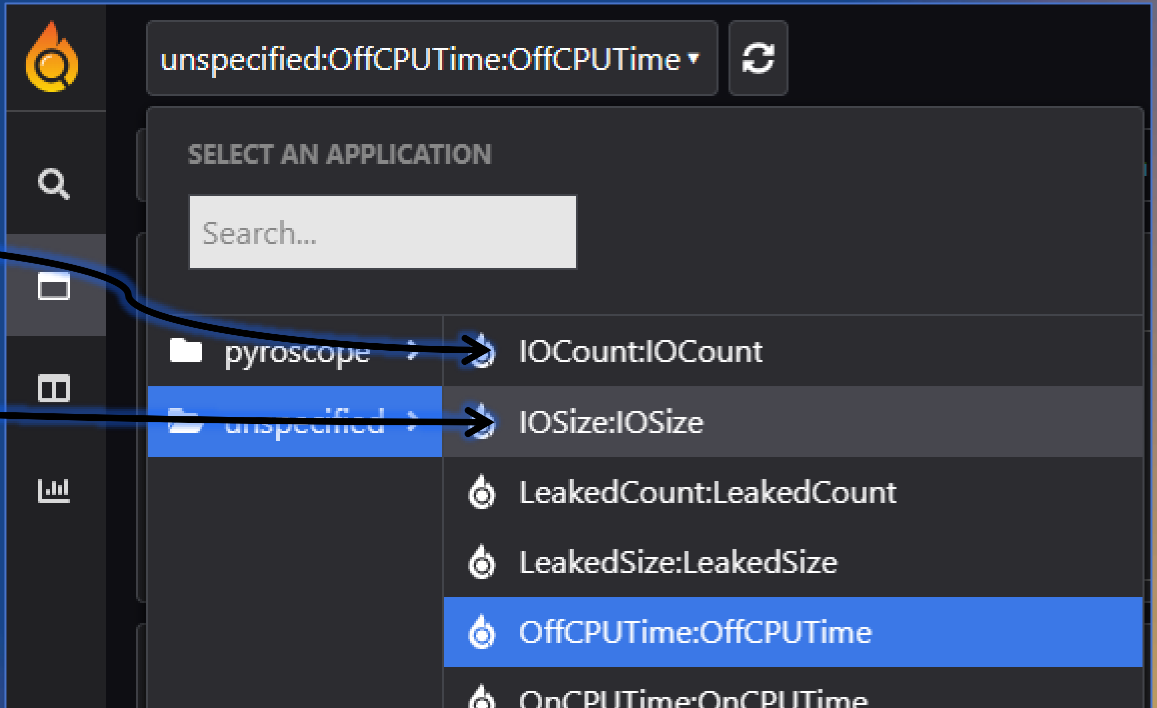
。。。

✓多视图观察

为调用栈数据提供了更多的视图，便于多维度剖析调用栈性能瓶颈



```
time:20240409_06_58_54
counts:
pid     usid    ksid    IOSize/1bytes   IOCount/1counts
28654   72024   -14     1       1
28654   67136   -14     10      10
3202959 109930  -14     49      1
28648   50389   -14     80      5
3183771 80575   -14     84      1
28648   97746   -14     96      6
28448   125287  -14     1850    2
3183771 87385   -14     32768   1
3202959 36214   -14     65536   1
```



3、便于扩展的指标

- ✓自定义指标列表
- ✓自定义指标解析方法

利于框架可扩展性

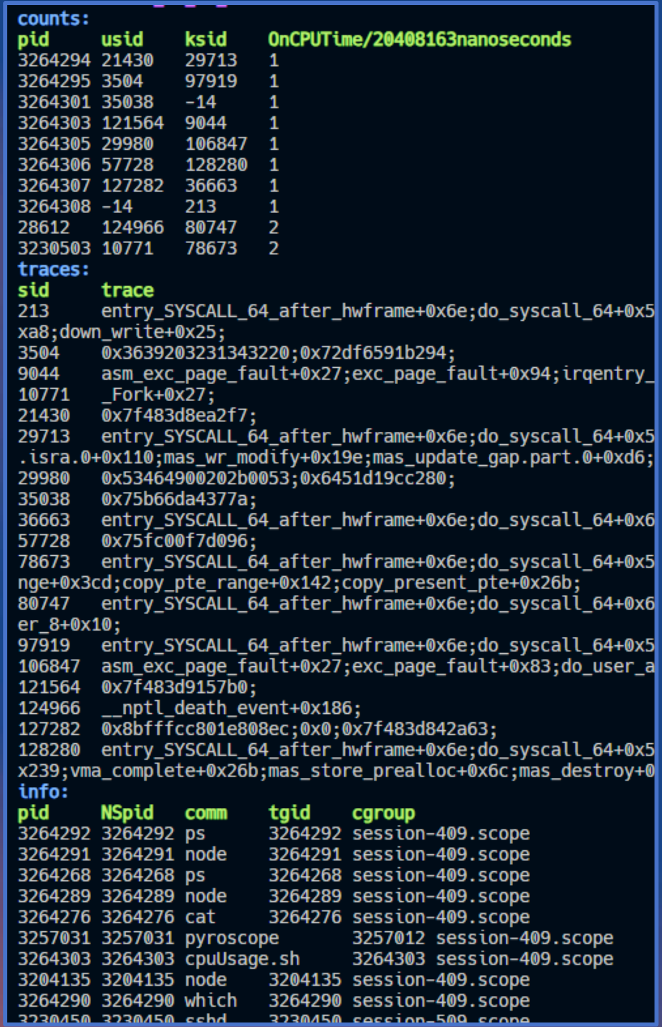
1、 背景

2、 PSI触发及及时采集

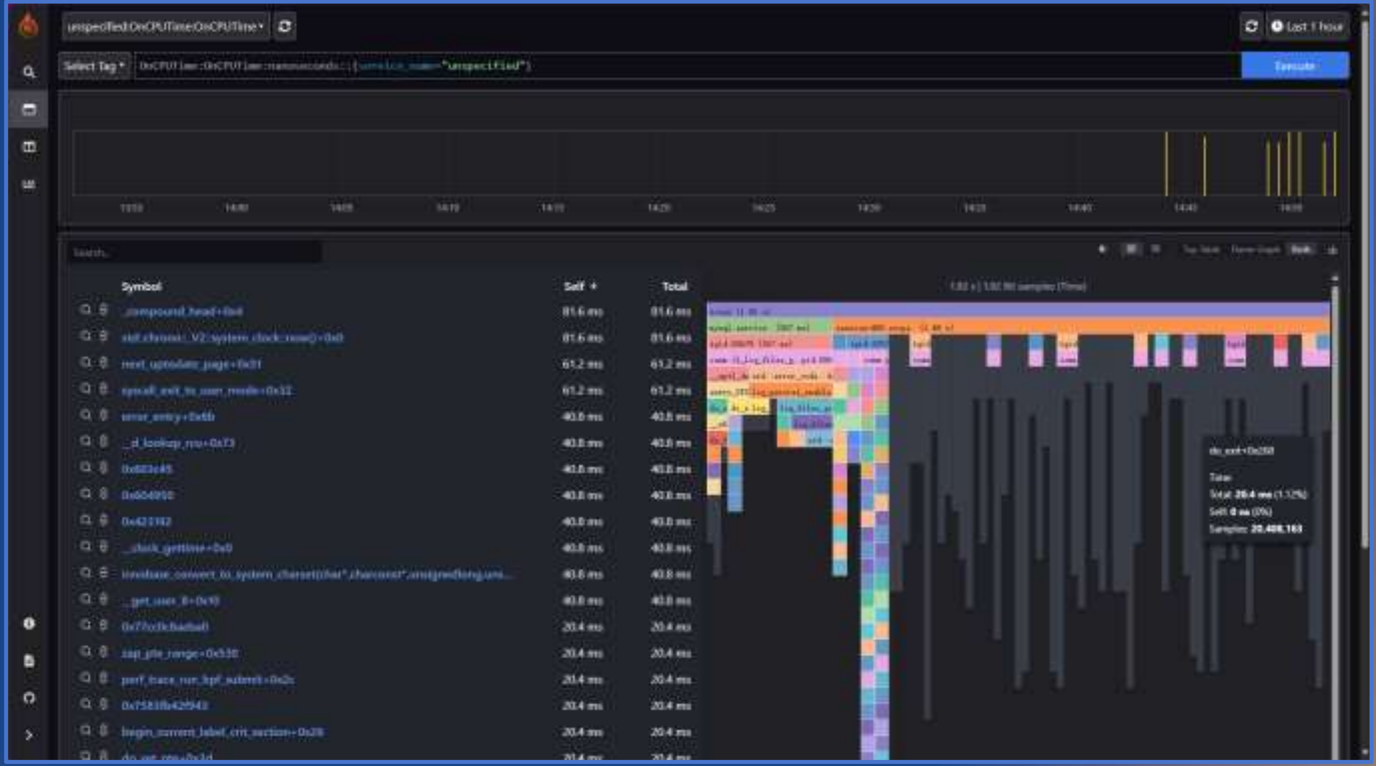
3、 多指标关联

4、 持续性观测及长期存储

4、eBPF应用案例 -- cpu占用



```
counts:
pid     usid    ksid    OnCPUTime/20408163nanoseconds
3264294 21430   29713   1
3264295 3504    97919   1
3264301 35038   -14     1
3264303 121564  9044    1
3264305 29980   106847  1
3264306 57728   128280  1
3264307 127282  36663   1
3264308 -14     213     1
28612   124966  80747   2
3230503 10771   78673   2
traces:
sid     trace
213     entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x5
xa8;down_write+0x25;
3504    0x3639203231343220;0x72df6591b294;
9044    asm_exc_page_fault+0x27;exc_page_fault+0x94;irqentry_
10771   _Fork+0x27;
21430   0x7f483d8ea2f7;
29713   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x5
.isra.0+0x110;mas_wr_modify+0x19e;mas_update_gap.part.0+0xd6;
29980   0x53464900202b0053;0x6451d19cc280;
35038   0x75b66da4377a;
36663   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x6
57728   0x75fc00f7d096;
78673   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x5
nge+0x3cd;copy_pte_range+0x142;copy_present_pte+0x26b;
80747   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x6
er_8+0x10;
97919   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x5
106847  asm_exc_page_fault+0x27;exc_page_fault+0x83;do_user_a
121564  0x7f483d9157b0;
124966  __nptl_death_event+0x186;
127282  0x8bfffcc801e808ec;0x0;0x7f483d842a63;
128280  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x5
x239;vma_complete+0x26b;mas_store_prealloc+0x6c;mas_destroy+0
info:
pid     NSpid   comm    tgid    cgroup
3264292 3264292 ps      3264292 session-409.scope
3264291 3264291 node    3264291 session-409.scope
3264268 3264268 ps      3264268 session-409.scope
3264289 3264289 node    3264289 session-409.scope
3264276 3264276 cat     3264276 session-409.scope
3257031 3257031 pyroscope       3257012 session-409.scope
3264303 3264303 cpuUsage.sh     3264303 session-409.scope
3204135 3204135 node    3204135 session-409.scope
3264290 3264290 which   3264290 session-409.scope
3230450 3230450 sshd    3230450 session-509.scope
```

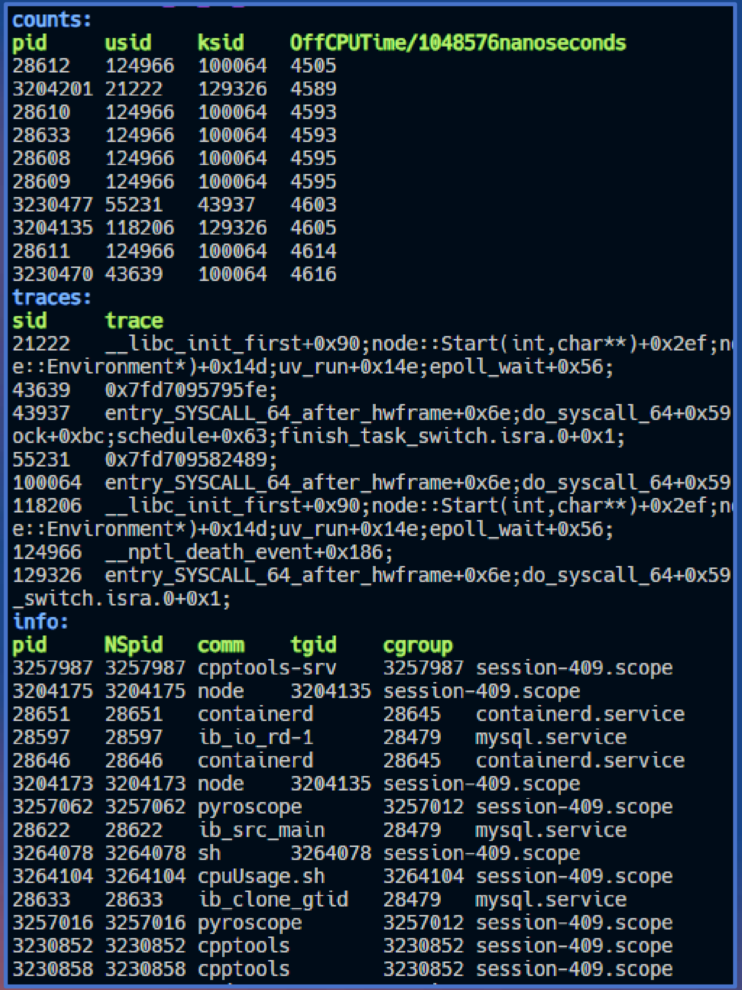


OnCPUTime:OnCPUTime:nanoseconds::{service_name="unspecified"}

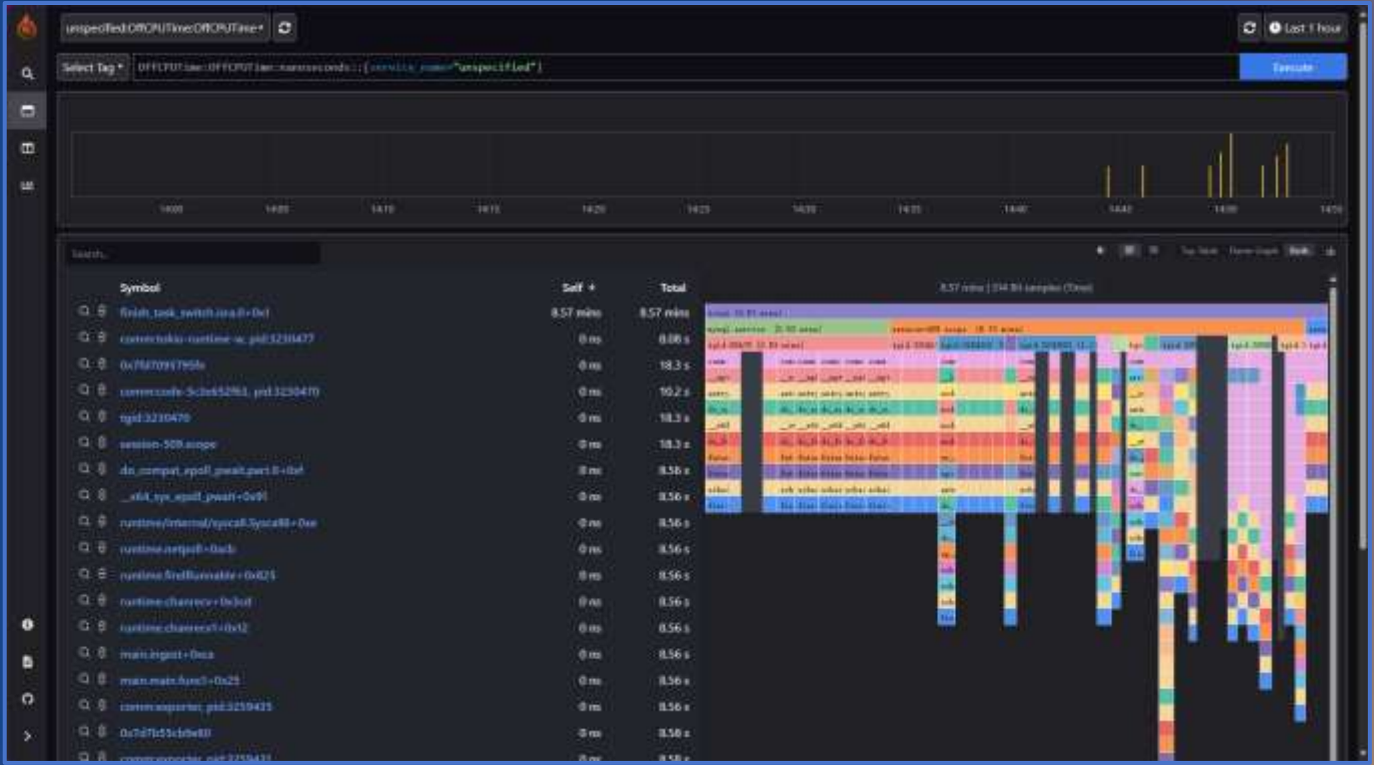
Symbol	Self	Total
_compound_head+0x4	81.6 ms	81.6 ms
std::chrono::_V2::system_clock::now()+0x0	81.6 ms	81.6 ms
next_uptodate_page+0x31	61.2 ms	61.2 ms
syscall_exit_to_user_mode+0x32	61.2 ms	61.2 ms
error_entry+0xbb	40.8 ms	40.8 ms
__d_lookup_rcu+0x73	40.8 ms	40.8 ms
[illegible]	40.8 ms	40.8 ms
[illegible]	40.8 ms	40.8 ms
[illegible]	40.8 ms	40.8 ms
__clock_gettime+0x0	40.8 ms	40.8 ms
innobase_convert_to_system_charset(char*,charconst*,unsignedlong,uns...	40.8 ms	40.8 ms
__get_user_8+0x10	40.8 ms	40.8 ms
[illegible]	20.4 ms	20.4 ms
zap_pte_range+0x530	20.4 ms	20.4 ms
perf_trace_run_bpf_submit+0x2c	20.4 ms	20.4 ms
[illegible]	20.4 ms	20.4 ms
begin_current_label_crit_section+0x28	20.4 ms	20.4 ms
do_set_pte+0x1d	20.4 ms	20.4 ms

do_exit+0x2b0
Total: 20.4 ms (1.12%)
Self: 0 ms (0%)
Samples: 20,408,163

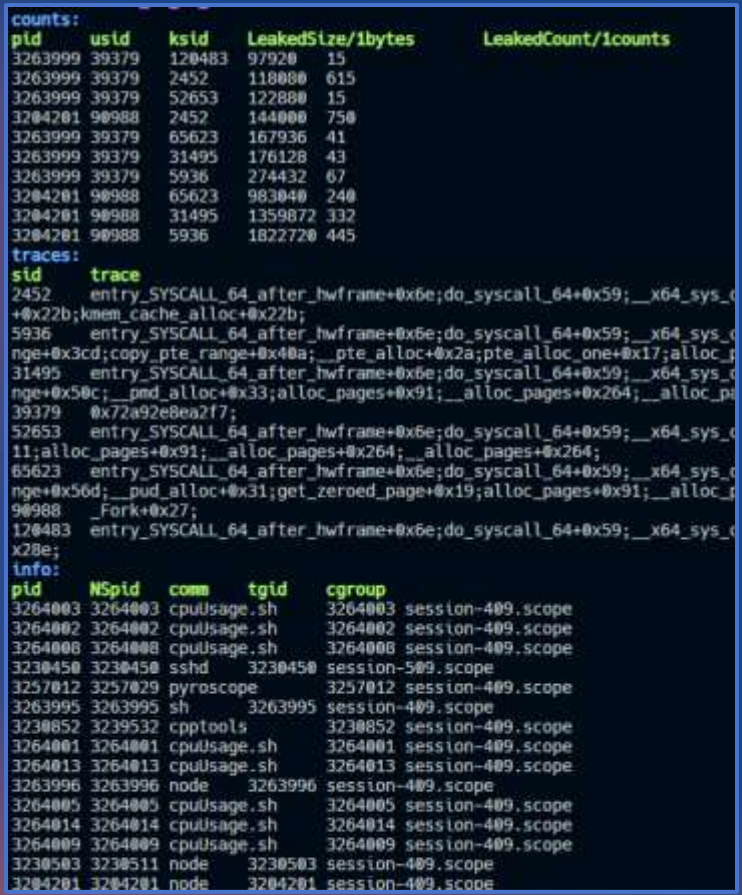
4、eBPF应用案例 -- 阻塞



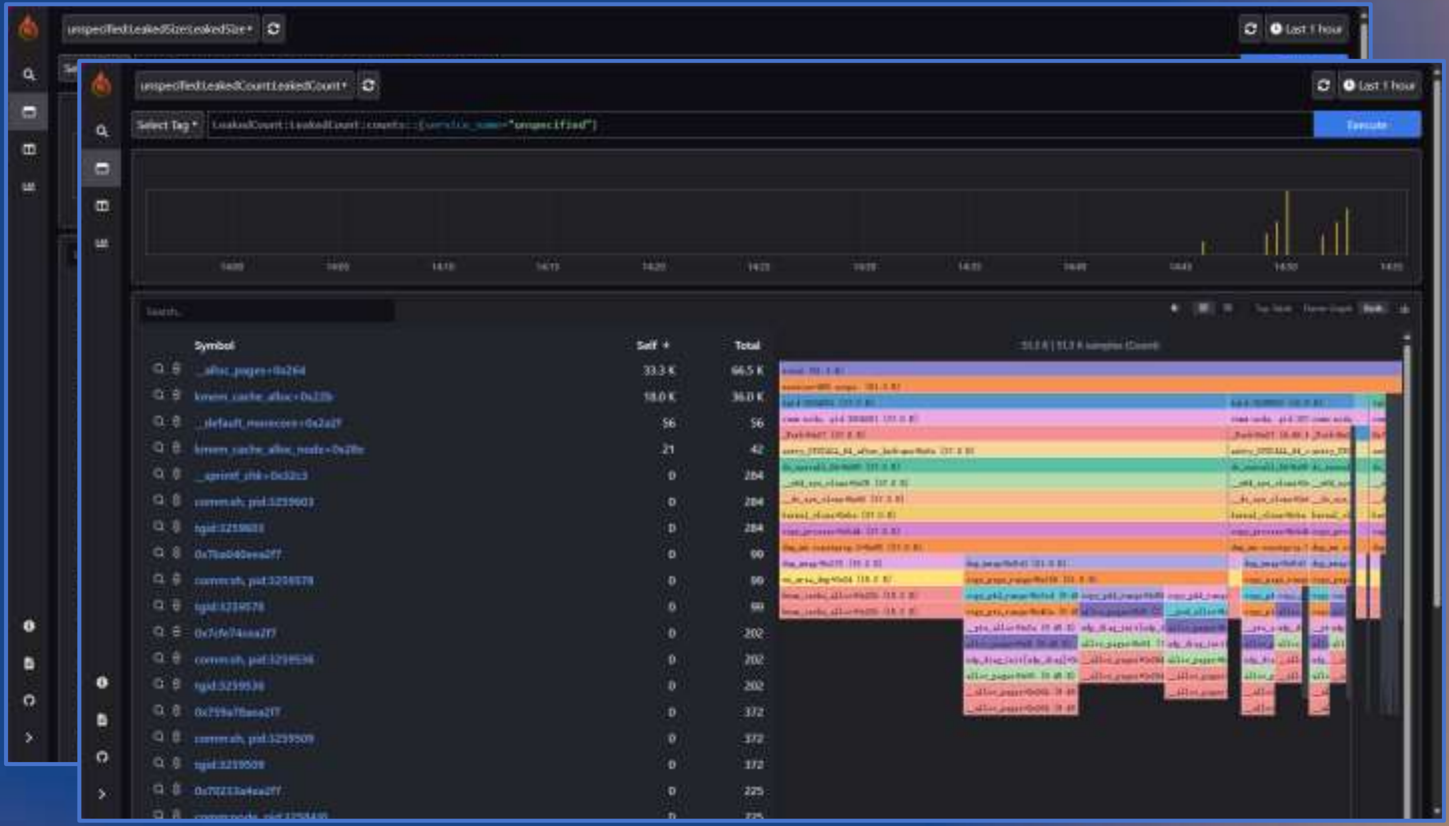
```
counts:
pid     usid    ksid    OffCPUTime/1048576nanoseconds
28612   124966  100064  4505
3204201 21222   129326  4589
28610   124966  100064  4593
28633   124966  100064  4593
28608   124966  100064  4595
28609   124966  100064  4595
3230477 55231   43937   4603
3204135 118206  129326  4605
28611   124966  100064  4614
3230470 43639   100064  4616
traces:
sid     trace
21222   __libc_init_first+0x90;node::Start(int,char**)+0x2ef;n
e::Environment*)+0x14d;uv_run+0x14e;epoll_wait+0x56;
43639   0x7fd7095795fe;
43937   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59
ock+0xbc;schedule+0x63;finish_task_switch.isra.0+0x1;
55231   0x7fd709582489;
100064  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59
118206  __libc_init_first+0x90;node::Start(int,char**)+0x2ef;n
e::Environment*)+0x14d;uv_run+0x14e;epoll_wait+0x56;
124966  __nptl_death_event+0x186;
129326  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59
_switch.isra.0+0x1;
info:
pid     NSpid   comm    tgid    cgroup
3257987 3257987 cpptools-srv    3257987 session-409.scope
3204175 3204175 node    3204135 session-409.scope
28651   28651   containerd      28645   containerd.service
28597   28597   ib_io_rd-1      28479   mysql.service
28646   28646   containerd      28645   containerd.service
3204173 3204173 node    3204135 session-409.scope
3257062 3257062 pyroscope       3257012 session-409.scope
28622   28622   ib_src_main     28479   mysql.service
3264078 3264078 sh      3264078 session-409.scope
3264104 3264104 cpuUsage.sh     3264104 session-409.scope
28633   28633   ib_clone_gtid   28479   mysql.service
3257016 3257016 pyroscope       3257012 session-409.scope
3230852 3230852 cpptools        3230852 session-409.scope
3230858 3230858 cpptools        3230852 session-409.scope
```


unspecified:OffCPUTime:OffCPUTime
Select Tag
OFFCPUTime:OFFCPUTime:nanoseconds::{service_name="unspecified"}
Last 1 hour
Execute
Search...
Symbol
Self
Total
finish_task_switch.isra.0+0x1
8.57 mins
8.57 mins

4、eBPF应用案例 --内存泄露



```
counts:
pid     usid    ksid    LeakedSize/1bytes       LeakedCount/1counts
3263999 39379   120483  97920   15
3263999 39379   2452    118080  615
3263999 39379   52653   122880  15
3204201 90988   2452    144000  750
3263999 39379   65623   167936  41
3263999 39379   31495   176128  43
3263999 39379   5936    274432  67
3204201 90988   65623   983040  240
3204201 90988   31495   1359872 332
3204201 90988   5936    1822720 445
traces:
sid     trace
2452    entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
+0x22b;kmem_cache_alloc+0x22b;
5936    entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x3cd;copy_pte_range+0x40a;__pte_alloc+0x2a;pte_alloc_one+0x17;alloc_p
31495   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x50c;__pmd_alloc+0x33;alloc_pages+0x91;__alloc_pages+0x264;__alloc_pa
39379   0x72a92e8ea2f7;
52653   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
11;alloc_pages+0x91;__alloc_pages+0x264;__alloc_pages+0x264;
65623   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x56d;__pud_alloc+0x31;get_zeroed_page+0x19;alloc_pages+0x91;__alloc_p
90988   _Fork+0x27;
120483  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
x28e;
info:
pid     NSpid   comm    tgid    cgroup
3264003 3264003 cpuUsage.sh     3264003 session-409.scope
3264002 3264002 cpuUsage.sh     3264002 session-409.scope
3264008 3264008 cpuUsage.sh     3264008 session-409.scope
3230450 3230450 sshd    3230450 session-509.scope
3257012 3257029 pyroscope       3257012 session-409.scope
3263995 3263995 sh      3263995 session-409.scope
3230852 3239532 cpptools        3230852 session-409.scope
3264001 3264001 cpuUsage.sh     3264001 session-409.scope
3264013 3264013 cpuUsage.sh     3264013 session-409.scope
3263996 3263996 node    3263996 session-409.scope
3264005 3264005 cpuUsage.sh     3264005 session-409.scope
3264014 3264014 cpuUsage.sh     3264014 session-409.scope
3264009 3264009 cpuUsage.sh     3264009 session-409.scope
3230503 3230511 node    3230503 session-409.scope
3204201 3204201 node    3204201 session-409.scope
```

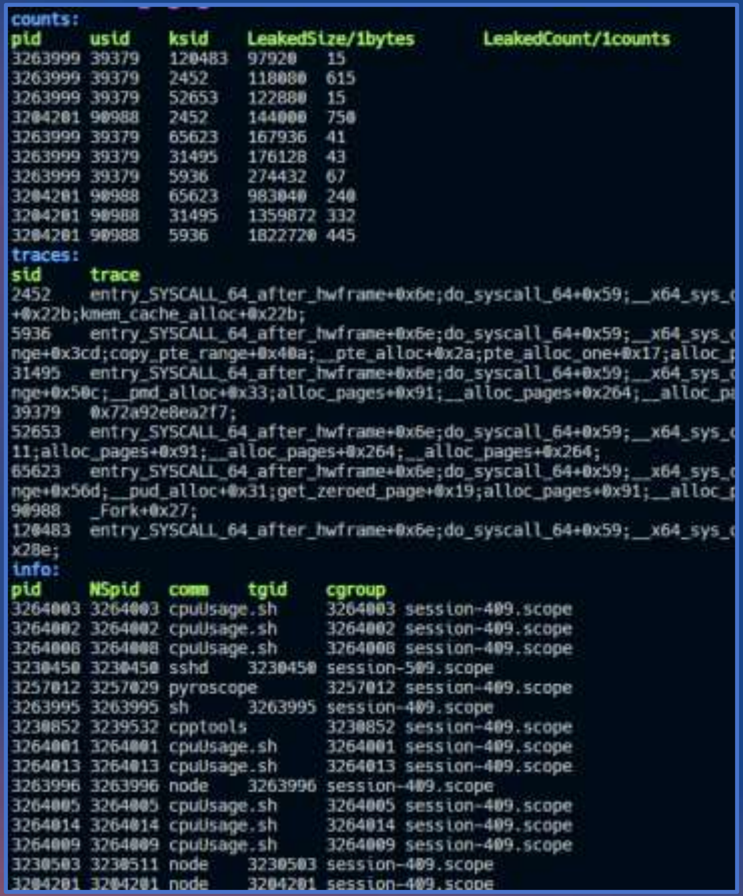


unspecified:LeakedSize:LeakedSize
Last 1 hour
unspecified:LeakedCount:LeakedCount
Select Tag
Execute
Search...

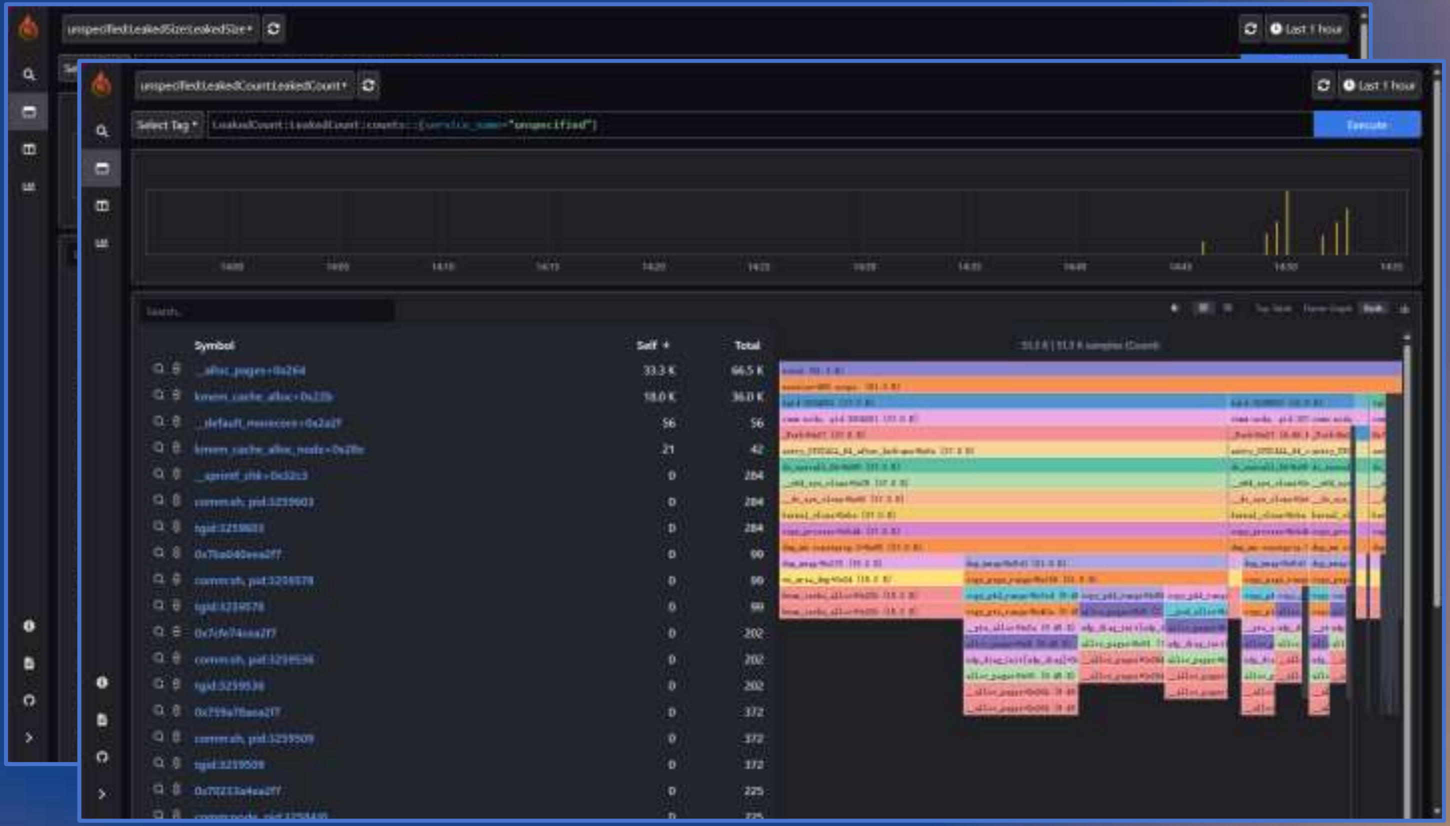
LeakedCount:LeakedCount:counts::{service_name="unspecified"}

Symbol	Self	Total
__alloc_pages+0x264	33.3 K	66.5 K
kmem_cache_alloc+0x22b	18.0 K	36.0 K
__default_morecore+[illegible]	56	56
kmem_cache_alloc_node+[illegible]	21	42
[illegible]	0	284
comm:sh, pid:[illegible]	0	284
tgid:[illegible]	0	284
[illegible]	0	99
comm:sh, pid:[illegible]	0	99
tgid:[illegible]	0	99
[illegible]	0	202
comm:sh, pid:[illegible]	0	202
tgid:[illegible]	0	202
[illegible]	0	372
comm:sh, pid:[illegible]	0	372
tgid:[illegible]	0	372
[illegible]	0	225
comm:node, pid:[illegible]	0	225

4、eBPF应用案例 --内存泄露



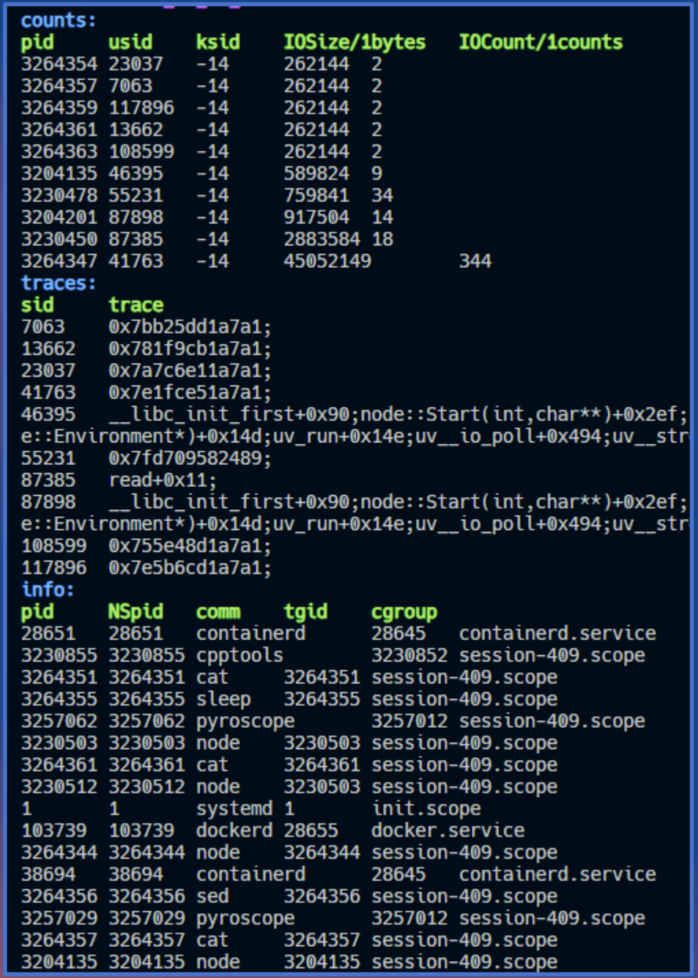
```
counts:
pid     usid    ksid    LeakedSize/1bytes       LeakedCount/1counts
3263999 39379   120483  97920   15
3263999 39379   2452    118080  615
3263999 39379   52653   122880  15
3204201 90988   2452    144000  750
3263999 39379   65623   167936  41
3263999 39379   31495   176128  43
3263999 39379   5936    274432  67
3204201 90988   65623   983040  240
3204201 90988   31495   1359872 332
3204201 90988   5936    1822720 445
traces:
sid     trace
2452    entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
+0x22b;kmem_cache_alloc+0x22b;
5936    entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x3cd;copy_pte_range+0x40a;__pte_alloc+0x2a;pte_alloc_one+0x17;alloc_p
31495   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x50c;__pmd_alloc+0x33;alloc_pages+0x91;__alloc_pages+0x264;__alloc_pa
39379   0x72a92e8ea2f7;
52653   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
11;alloc_pages+0x91;__alloc_pages+0x264;__alloc_pages+0x264;
65623   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
nge+0x56d;__pud_alloc+0x31;get_zeroed_page+0x19;alloc_pages+0x91;__alloc_p
90988   _Fork+0x27;
120483  entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0x59;__x64_sys_c
x28e;
info:
pid     NSpid   comm    tgid    cgroup
3264003 3264003 cpuUsage.sh     3264003 session-409.scope
3264002 3264002 cpuUsage.sh     3264002 session-409.scope
3264008 3264008 cpuUsage.sh     3264008 session-409.scope
3230450 3230450 sshd    3230450 session-509.scope
3257012 3257029 pyroscope       3257012 session-409.scope
3263995 3263995 sh      3263995 session-409.scope
3230852 3239532 cpptools        3230852 session-409.scope
3264001 3264001 cpuUsage.sh     3264001 session-409.scope
3264013 3264013 cpuUsage.sh     3264013 session-409.scope
3263996 3263996 node    3263996 session-409.scope
3264005 3264005 cpuUsage.sh     3264005 session-409.scope
3264014 3264014 cpuUsage.sh     3264014 session-409.scope
3264009 3264009 cpuUsage.sh     3264009 session-409.scope
3230503 3230511 node    3230503 session-409.scope
3204201 3204201 node    3204201 session-409.scope
```


unspecified:LeakedSize:LeakedSize
Last 1 hour
unspecified:LeakedCount:LeakedCount
Last 1 hour
Select Tag
Execute
Search...

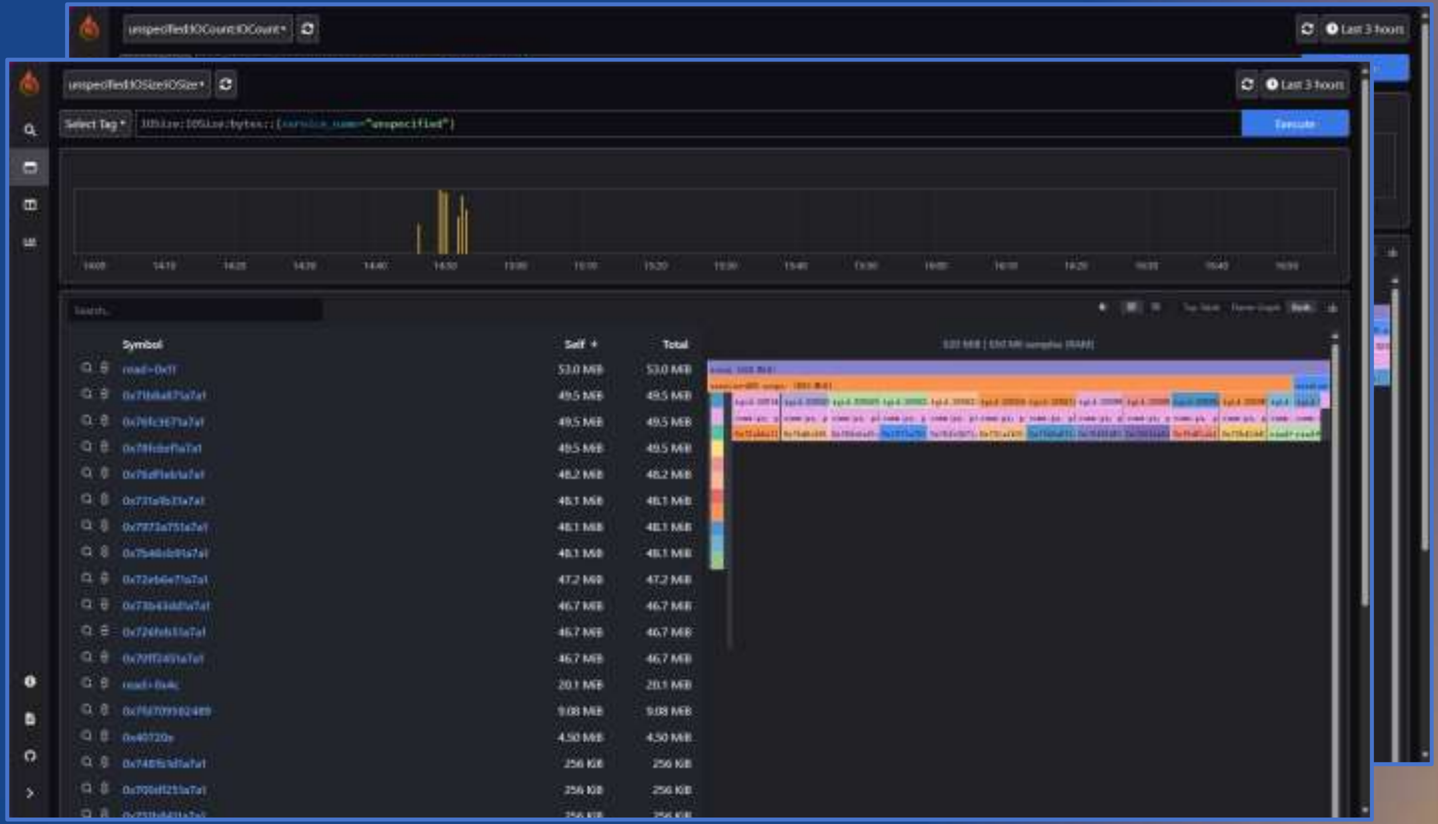
```
LeakedCount:LeakedCount:counts::{service_name="unspecified"}
```

Symbol	Self	Total
__alloc_pages+0x264	33.3 K	66.5 K
kmem_cache_alloc+0x22b	18.0 K	36.0 K
__default_morecore+0x2a2f	56	56
kmem_cache_alloc_node+0x28c	21	42
__sprintf_chk+0x32c3	0	284
comm:sh, pid:3259603	0	284
tgid:3259603	0	284
0x7fba040eea2f7	0	99
comm:sh, pid:3259578	0	99
tgid:3259578	0	99
0x7cfe74eea2f7	0	202
comm:sh, pid:3259536	0	202
tgid:3259536	0	202
0x7f99a7fbea2f7	0	372
comm:sh, pid:3259509	0	372
tgid:3259509	0	372
0x7f0233a4ea2f7	0	225
comm:node, pid:3258430	0	225

4、eBPF应用案例 -- 输入输出

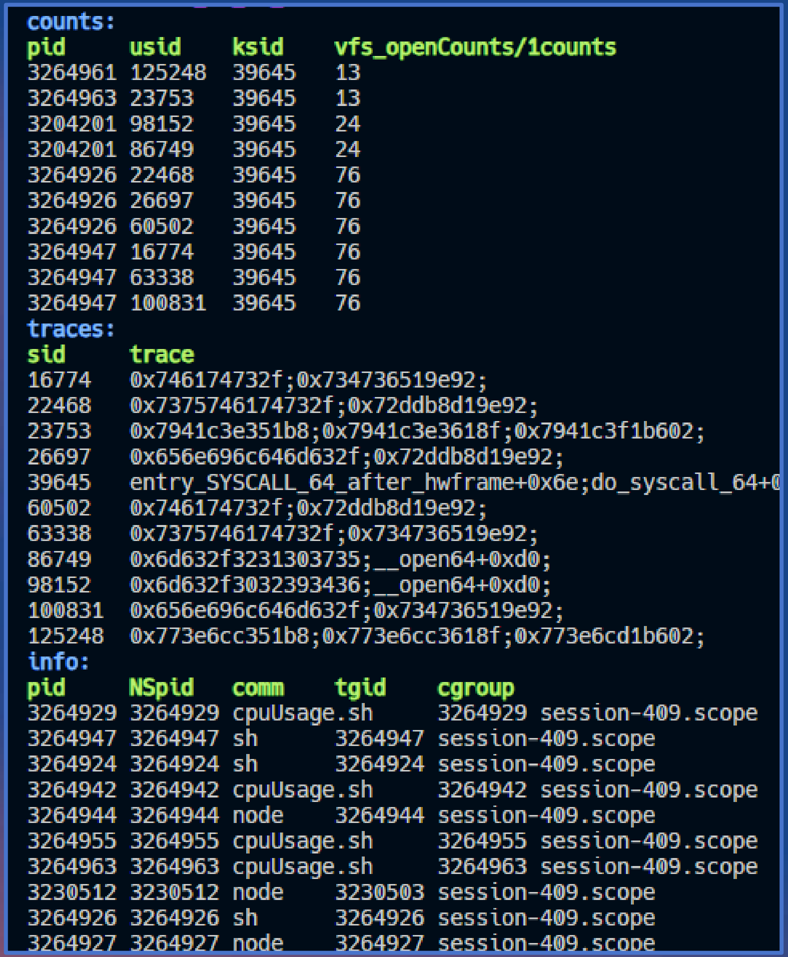


```
counts:
pid     usid    ksid    IOSize/1bytes   IOCount/1counts
3264354 23037   -14     262144  2
3264357 7063    -14     262144  2
3264359 117896  -14     262144  2
3264361 13662   -14     262144  2
3264363 108599  -14     262144  2
3204135 46395   -14     589824  9
3230478 55231   -14     759841  34
3204201 87898   -14     917504  14
3230450 87385   -14     2883584 18
3264347 41763   -14     45052149        344
traces:
sid     trace
7063    0x7bb25dd1a7a1;
13662   0x781f9cb1a7a1;
23037   0x7a7c6e11a7a1;
41763   0x7e1fce51a7a1;
46395   __libc_init_first+0x90;node::Start(int,char**)+0x2ef;
e::Environment*)+0x14d;uv_run+0x14e;uv__io_poll+0x494;uv__str
55231   0x7fd709582489;
87385   read+0x11;
87898   __libc_init_first+0x90;node::Start(int,char**)+0x2ef;
e::Environment*)+0x14d;uv_run+0x14e;uv__io_poll+0x494;uv__str
108599  0x755e48d1a7a1;
117896  0x7e5b6cd1a7a1;
info:
pid     NSpid   comm    tgid    cgroup
28651   28651   containerd      28645   containerd.service
3230855 3230855 cpptools        3230852 session-409.scope
3264351 3264351 cat     3264351 session-409.scope
3264355 3264355 sleep   3264355 session-409.scope
3257062 3257062 pyroscope       3257012 session-409.scope
3230503 3230503 node    3230503 session-409.scope
3264361 3264361 cat     3264361 session-409.scope
3230512 3230512 node    3230503 session-409.scope
1       1       systemd 1       init.scope
103739  103739  dockerd 28655   docker.service
3264344 3264344 node    3264344 session-409.scope
38694   38694   containerd      28645   containerd.service
3264356 3264356 sed     3264356 session-409.scope
3257029 3257029 pyroscope       3257012 session-409.scope
3264357 3264357 cat     3264357 session-409.scope
3204135 3204135 node    3204135 session-409.scope
```

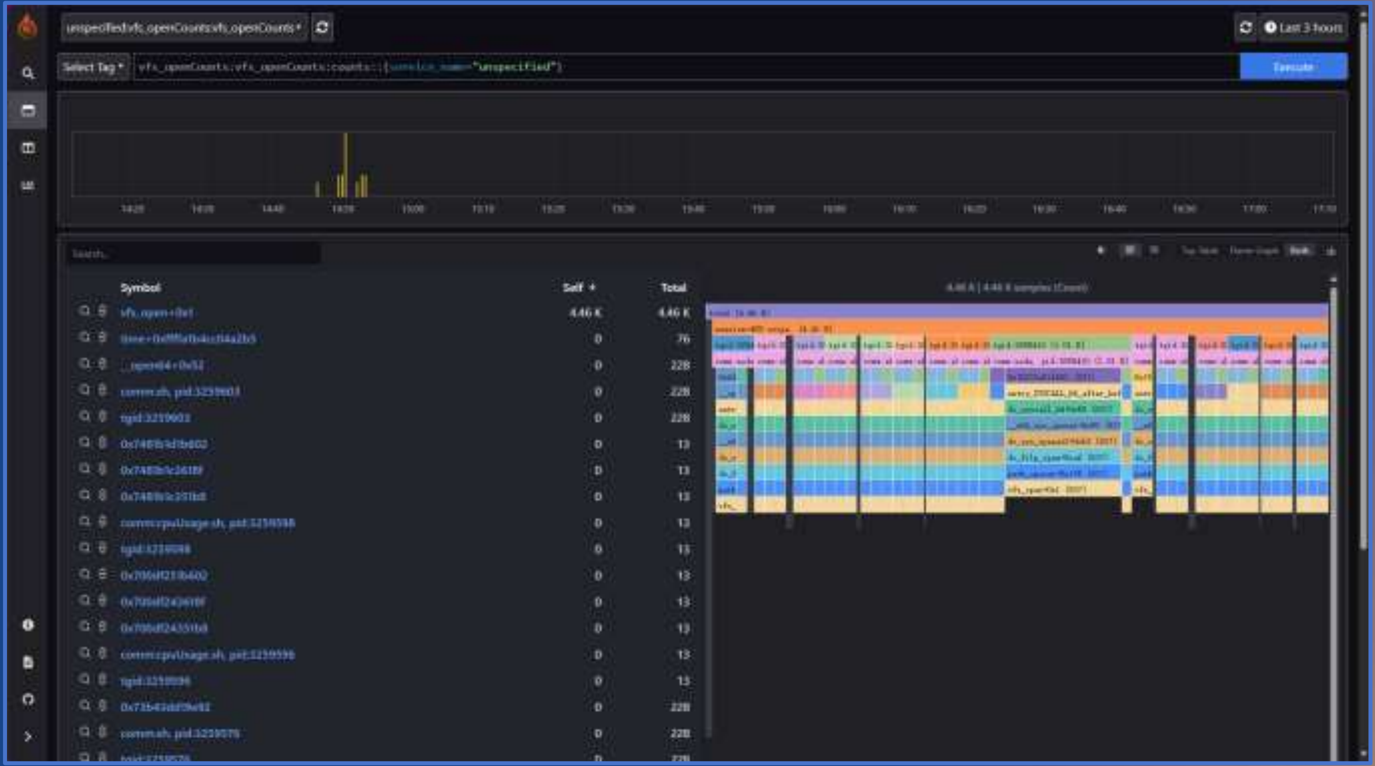

unspecified.IOCount.IOCount
Last 3 hours
unspecified.IOSize.IOSize
Last 3 hours
Select Tag
IOSize:IOSize:bytes::{service_name="unspecified"}
Execute
Search...

Symbol	Self	Total
read+0x11	53.0 MiB	53.0 MiB
[illegible]	49.5 MiB	49.5 MiB
[illegible]	49.5 MiB	49.5 MiB
[illegible]	49.5 MiB	49.5 MiB
[illegible]	48.2 MiB	48.2 MiB
[illegible]	48.1 MiB	48.1 MiB
[illegible]	48.1 MiB	48.1 MiB
[illegible]	48.1 MiB	48.1 MiB
[illegible]	47.2 MiB	47.2 MiB
[illegible]	46.7 MiB	46.7 MiB
[illegible]	46.7 MiB	46.7 MiB
[illegible]	46.7 MiB	46.7 MiB
read+0x4c	20.1 MiB	20.1 MiB
0x7fd709582489	9.08 MiB	9.08 MiB
0x40720e	4.50 MiB	4.50 MiB
[illegible]	256 KiB	256 KiB
[illegible]	256 KiB	256 KiB

4、eBPF应用案例 -- 自定义跟踪点计数：vfs_open



```
counts:
pid     usid    ksid    vfs_openCounts/1counts
3264961 125248  39645   13
3264963 23753   39645   13
3204201 98152   39645   24
3204201 86749   39645   24
3264926 22468   39645   76
3264926 26697   39645   76
3264926 60502   39645   76
3264947 16774   39645   76
3264947 63338   39645   76
3264947 100831  39645   76
traces:
sid     trace
16774   0x746174732f;0x734736519e92;
22468   0x7375746174732f;0x72ddb8d19e92;
23753   0x7941c3e351b8;0x7941c3e3618f;0x7941c3f1b602;
26697   0x656e696c646d632f;0x72ddb8d19e92;
39645   entry_SYSCALL_64_after_hwframe+0x6e;do_syscall_64+0
60502   0x746174732f;0x72ddb8d19e92;
63338   0x7375746174732f;0x734736519e92;
86749   0x6d632f3231303735;__open64+0xd0;
98152   0x6d632f3032393436;__open64+0xd0;
100831  0x656e696c646d632f;0x734736519e92;
125248  0x773e6cc351b8;0x773e6cc3618f;0x773e6cd1b602;
info:
pid     NSpid   comm    tgid    cgroup
3264929 3264929 cpuUsage.sh     3264929 session-409.scope
3264947 3264947 sh      3264947 session-409.scope
3264924 3264924 sh      3264924 session-409.scope
3264942 3264942 cpuUsage.sh     3264942 session-409.scope
3264944 3264944 node    3264944 session-409.scope
3264955 3264955 cpuUsage.sh     3264955 session-409.scope
3264963 3264963 cpuUsage.sh     3264963 session-409.scope
3230512 3230512 node    3230503 session-409.scope
3264926 3264926 sh      3264926 session-409.scope
3264927 3264927 node    3264927 session-409.scope
```



unspecified:vfs_openCounts:vfs_openCounts
Last 3 hours
Select Tag
Execute
Search...
4.46 K | 4.46 K samples (Count)

```
vfs_openCounts:vfs_openCounts:counts::{service_name="unspecified"}
```

Symbol	Self	Total
vfs_open+0x1	4.46 K	4.46 K
[illegible]	0	76
__open64+0x52	0	228
comm:sh, pid:[illegible]	0	228
tgid:[illegible]	0	228
[illegible]	0	13
[illegible]	0	13
[illegible]	0	13
comm:cpuUsage.sh, pid:[illegible]	0	13
tgid:[illegible]	0	13
[illegible]	0	13
[illegible]	0	13
[illegible]	0	13
comm:cpuUsage.sh, pid:[illegible]	0	13
tgid:[illegible]	0	13
[illegible]	0	228
comm:sh, pid:[illegible]	0	228
tgid:[illegible]	0	228

谢谢大家！