第三届 eBPF开发者大会

www.ebpftravel.com

eBPF-交流研讨

中国·西安

第三届 eBPF开发者大会

www.ebpftravel.com

eBPF技术在滴滴自动驾驶场景下的相关实践

陈涛·滴滴

中国·西安

1 项目整体概况

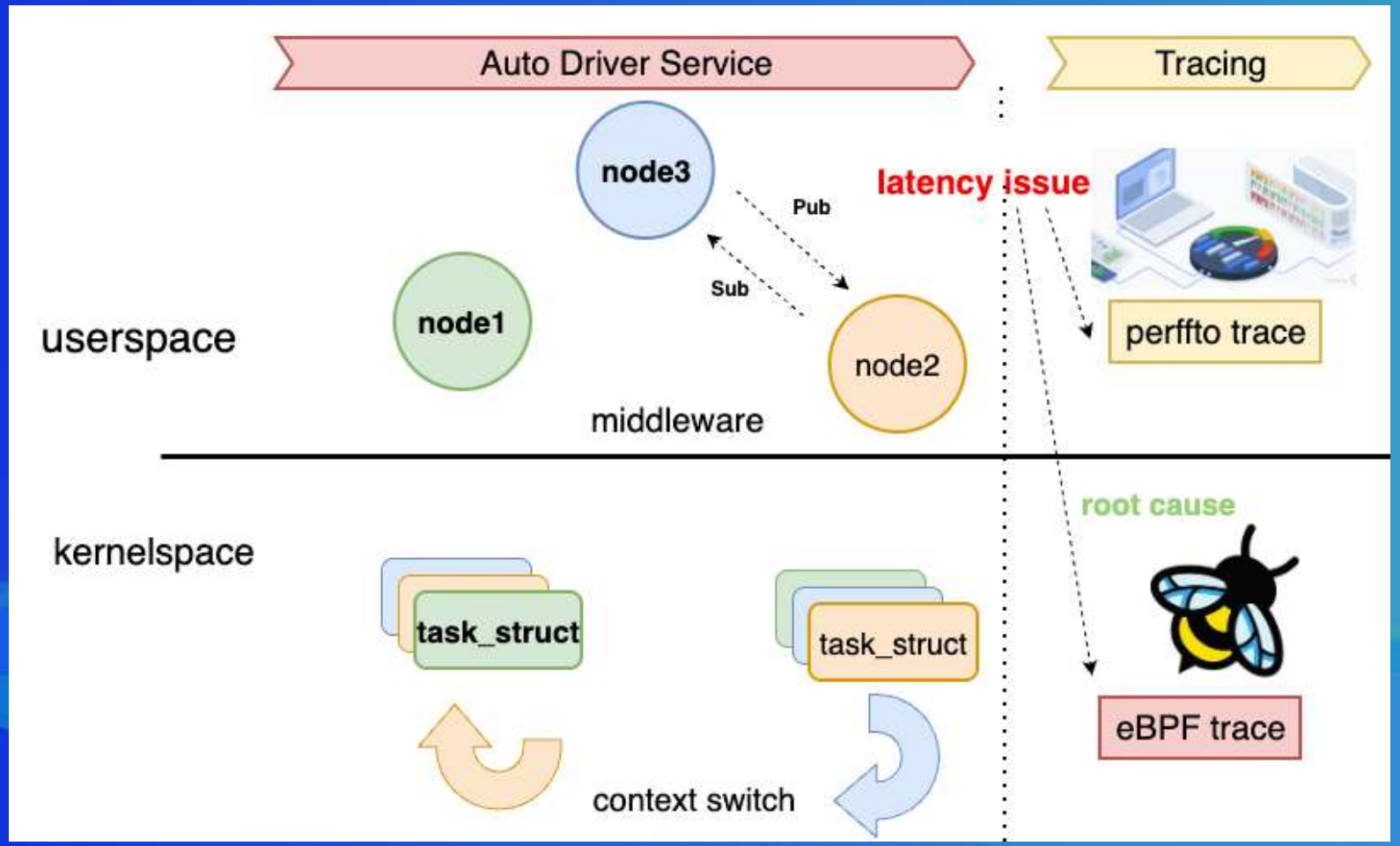
2 如何做到低频问题的定位/定界

3 问题案例

4 未来展望

1 项目整体概况

1.eBPF使用场景



以调度场景为列

perffto

- 用户态函数trace，能看到时延不知道底层原因

eBPF

- 内核级trace，善于定位根因
- 安全、高效、随时可上可下，不需要修改内核代码

2.覆盖的子系统

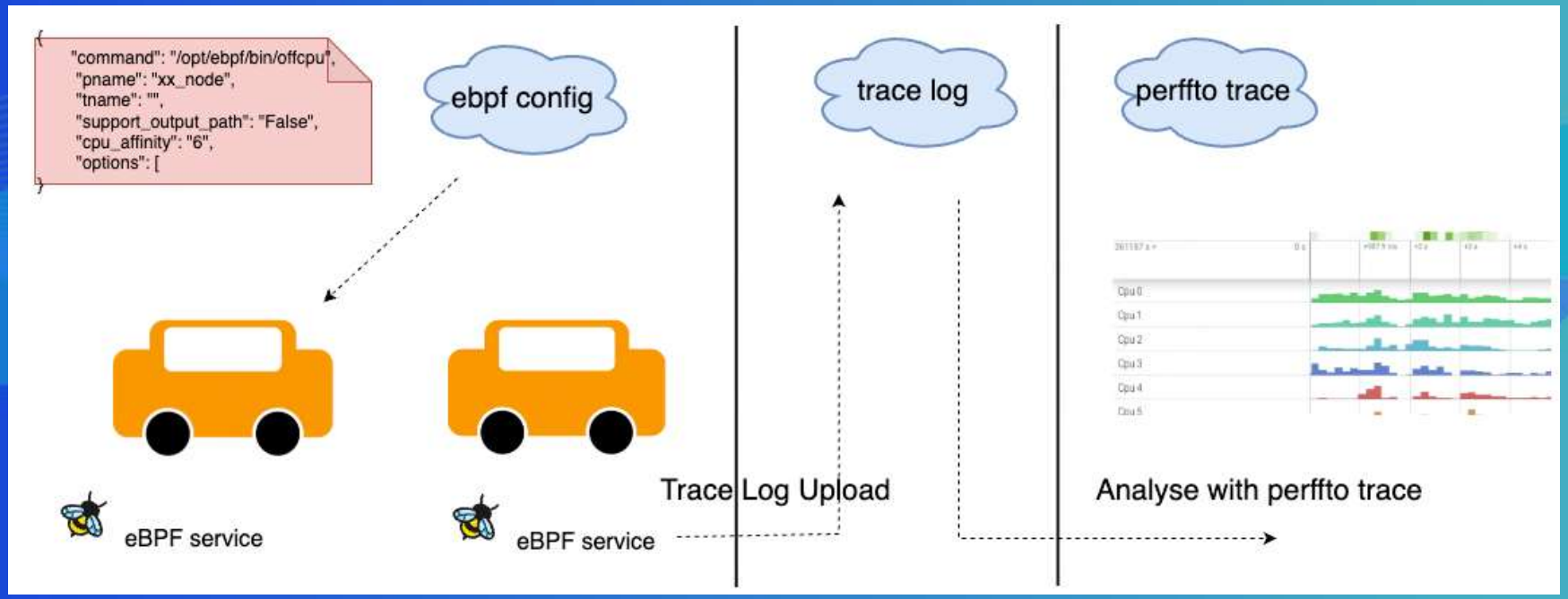
调度子系统	net子系统	others
offcpu 任务执行时延	netrack 网络时延	top_cache 查看cache占用
syscall 陷入内核时延	tcp_allloc skb分配时延	direct_reclaim 直接内存回收造成的系统hang
oncpu 实时 profiling	traffic_stat 进程级流量统计	rcetrace
irq_delay 查看中断时延	tcprtt 网络健康度衡量	systctl_watch 查看配置更改

3.车端工具部署

常态化部署 → 抓捕偶现问题

云端配置 → 不同问题配置不同工具；抓捕对应节点

日志分析 → eBFP日志结合perffto数据



2 如何做到低频问题的定位/定界

工具常态化部署遇到的挑战

- 常态化部署工具，实时追踪

应对车端毫秒级低频问题

遇到的挑战

- 业务节点对时延非常敏感(ms)
- 热点路径工具自身资源消耗较大

- 优化自身代码
- 增加部署策略、过滤条件(黑名单)等

工具优化

eBPF编程范式

用户态项目框架

libbpf-bootstrap

- 增加静态编译

libbpf

- 增加自定义help func
  bpf_preempt_cnount等

bazelsym 栈解析库

- cpu消耗较大替换成bcc库

内核程序

- 1.hook目标函数
- 2.将数据存在map
- 3.抓取栈信息(root cause)
- 4.数据输出

```
SEC(kprob/tracepoint/raw_tp/fentry xx)                    1
int prog_handler()
{
        ...
        bpf_map_update()
        bpf_map_lookup()                                  2
        ....
        bpf_get_stack                                     3

        bpf_perf_event_output                             4
}
```

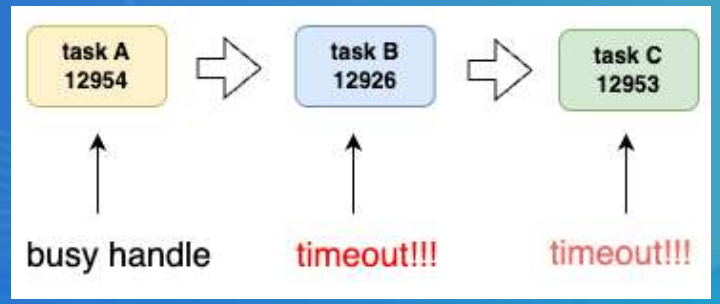
栈解析的重要性

利

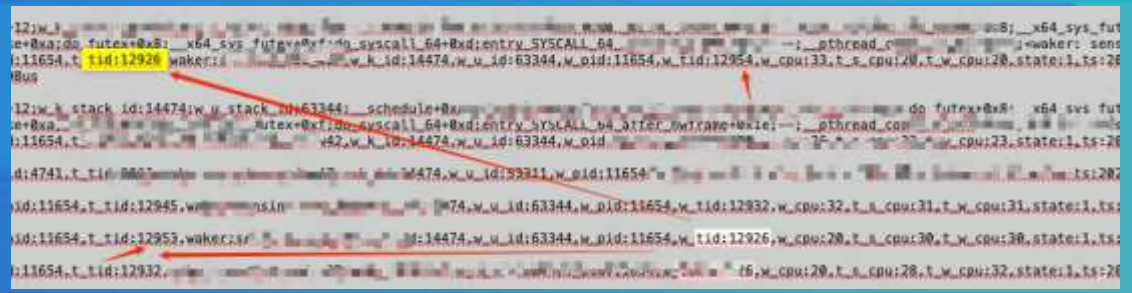
- 直观呈现问题第一现场，利于问题分析

弊

- 在bpf_prog中耗时占大头



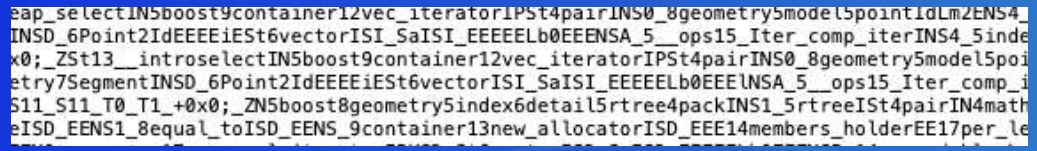
通过栈信息分析用户态程序等锁唤醒

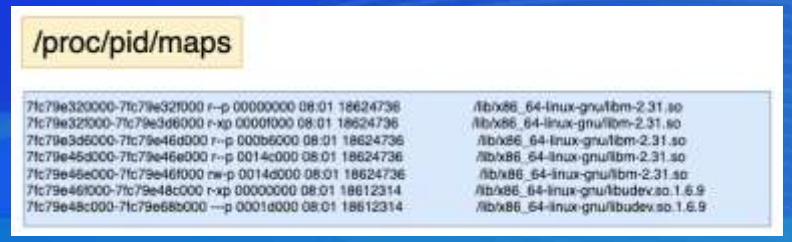


1.采栈优化-栈解析

blazesym 解析库存在的问题

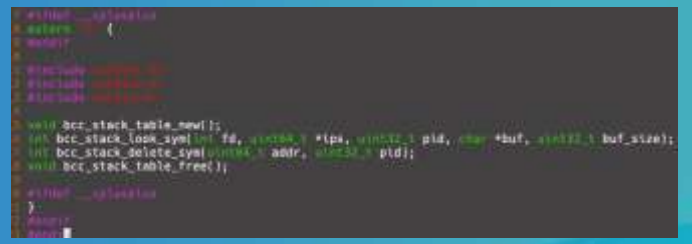
- C++栈信息需要符号后处理
- cpu消耗较大，读取符号信息会频繁持有mmap_sem锁

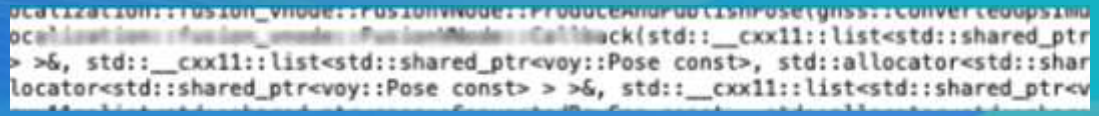


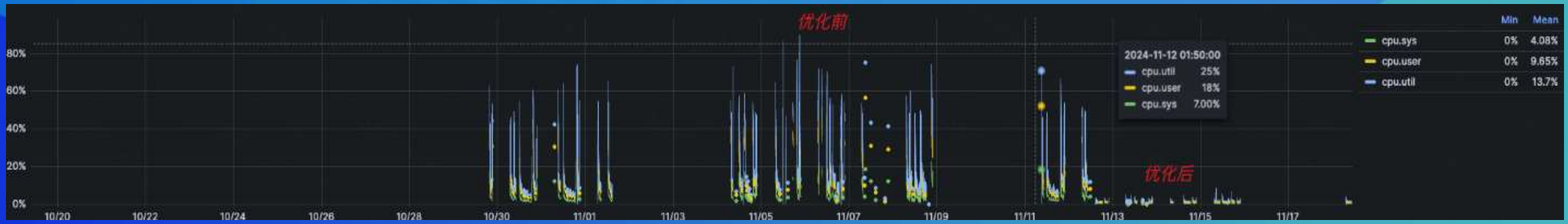


封装栈解析API，使用BCC解析库

- cpu使用降低70%
- 栈更直观无需后处理

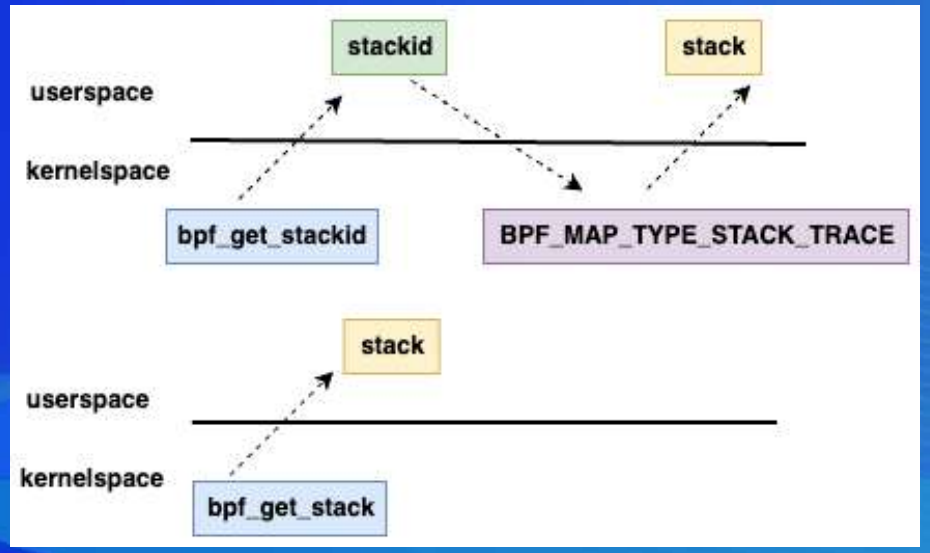






1.采栈优化-采栈时机

bpf_get_stack_id vs bpf_get_stack



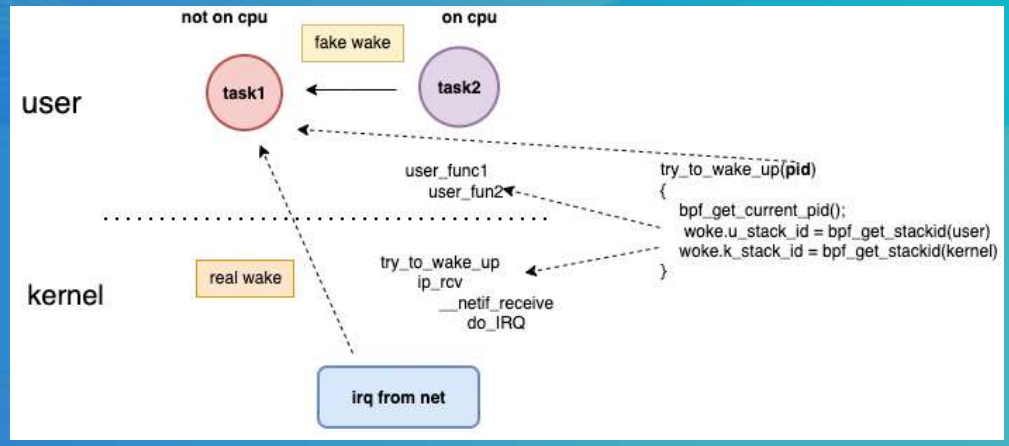
id hash conflict

USER_STACK vs KERNEL_STACK

1.内核线程上下文不采集用户态栈

```
,waker:kworker/u256:2,w_k_id:4363,w_u_id:-14,w_pid:64889,w
_len:0
ack_id:4363;w_u_stack_id:-14;finish_task_switch+0x1;schedu
```

2.中断上下文不采集用户态栈　30%占比！！！



2.函数hook方式优化

kprobe	tracepoint	raw_tracepoint	fentry
1.787M/s	1.873M/s	3.217M/s	3.795M/s

以实际内核版本实测为准

test enviroment

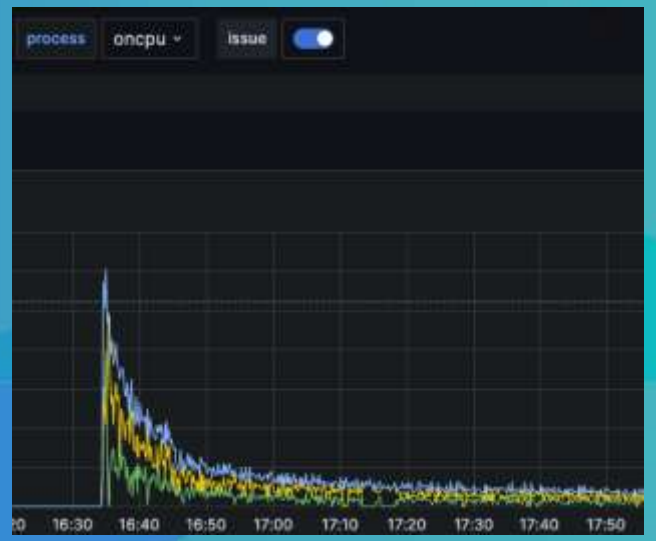
```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         40 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               AuthenticAMD
  BIOS Vendor ID:        QEMU
  Model name:            QEMU Virtual CPU version 2.5+
```

3.引入trigger机制

由频繁系统调用引起的sys打高

只在监控指标burst的情况下触发工具使用

- 节省CPU资源消耗
- 节省存储资源消耗




```
oncpu,k_id:2731,u_id:9938,pid:39784,tid:0,comm:voy_trace_contr,ts:2024-12-04T16:34:16.612890,cpu:3,dur:10.000,prio:120,static_prio:120,normal_prio:120,rt_prio:0,exec_runtime:0,vruntime:5539415789025,nr_running:1
<stack>;path_lookupat+0xa5;filename_lookup+0xb6;kern_path+0x2b;unix_find_other+0x45;unix_stream_connect+0xf2;__sys_connect+0xa3;__x64_sys_connect+0x1a;do_syscall_64+0x5a;entry_SYSCALL_64_after_hwframe+0x44;
oncpu,k_id:2731,u_id:9938,pid:39784,tid:0,comm [redacted]s:2024-12-04T16:34:16.612890,cpu:3,dur:10.000,prio:120,static_prio:120,normal_prio:120,rt_prio:0,exec_runtime:0,vruntime:5539415789025,nr_running:1
<stack>;[redacted]_connect+0x0;0x676e74746c2f6e75;
oncpu,k_id:2731,u_id:9938,pid:39784,tid:0,comm:[redacted],ts:2024-12-04T16:34:16.612890,cpu:3,dur:10.000,prio:120,static_prio:120,normal_prio:120,rt_prio:0,exec_runtime:0,vruntime:5539415789025, nr_running:1
```

3 案例分享

实际问题分享

内核中睡眠锁争抢引起时延(mmap_sem、rtnl_lock)

CFS公平调度引起的调度时延

高优先级任务抢占低优先级任务(网络包)

内存直接回收(同步等待)

用户程序逻辑问题(老大难！！！)

1.mmap_sem锁争抢

eBPF 工具 1

/proc/pidxx/maps

监控进程 2

/proc/pidxx/schedstat

业务节点 pidxx 3

Libc系统调用进入

```
sys_mprotect
  do_mprotect_pkey
    down_write_killable(&current->
                mm->mmap_sem)
```

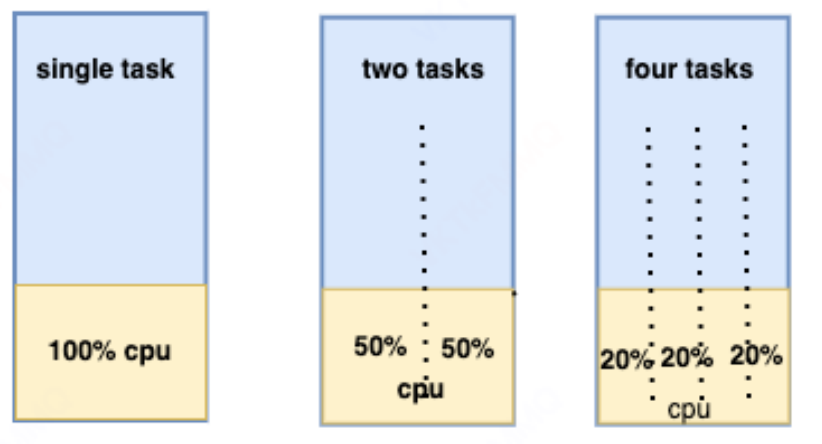
```
static const struct seq_operations proc_pid_maps_op = {
        .start  = m_start,
        .next   = m_next,
        .stop   = m_stop,
        .show   = show_map
};
```

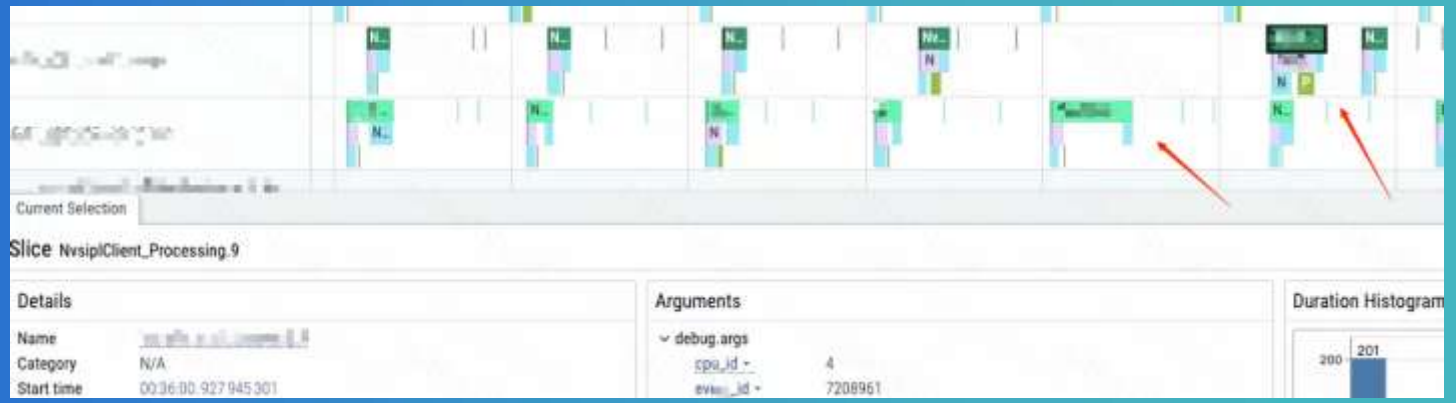
down_read_killable(&mm->mmap_sem);

_mprotect_pkey+0xad;__x64_sys_mprotect+0x1f;do_syscall_64+0x5a;entry_SYSCALL_64_after_hwframe+0x44;--;__mprotect+0x0;----;try_to_wake_up+0x1;rwsem_wake+0x8e;call_rwsem_wake+0x1b;up_read+0x1
+0x35b;__vfs_read+0x1b;vfs_read+0x8e;ksys_read+0x55;__x64_sys_read+0x1a;do_syscall_64+0x5a;entry_SYSCALL_64_after_hwframe+0x44;--;__read+0x0;0x6666372d30303032;<waker: offcpu>

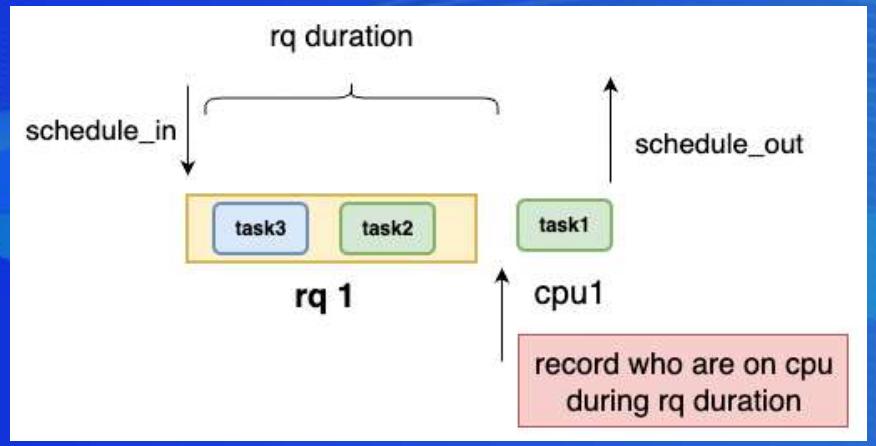
2.CFS公平调度引起的调度时延

CFS虚拟时间补偿机制





在rq长的时刻工具(offcpu)抓取
哪些任务在占用CPU




```
state:1,ts:2024-12-27T08:36:00.785,dur:113.97,q_dur:85500.00us,rq_dur:85478.00us,r
pid:11078,comm:collectd,prio:120,cpu:4,state:1,id:275,ts:2024-12-27T08:36:00.879972791,ser:6179269376,vr:4519552887
pid:11077,comm:collectd,prio:120,cpu:4,state:1,id:280,ts:2024-12-27T08:36:00.880273781,ser:6204805440,vr:4519554258
pid:11076,comm:collectd,prio:120,cpu:4,state:1,id:285,ts:2024-12-27T08:36:00.880472179,ser:6135801760,vr:4519553740
pid:11077,comm:collectd,prio:120,cpu:4,state:1,id:290,ts:2024-12-27T08:36:00.881188718,ser:6204838176,vr:4519554585
pid:52,comm:ksoftirqd/4,prio:120,cpu:4,state:1,id:295,ts:2024-12-27T08:36:00.881605195,ser:24050739616,vr:451953305
pid:11079,comm:collectd,prio:120,cpu:4,state:0,id:300,ts:2024-12-27T08:36:00.881899337,ser:6227900416,vr:4519555936
pid:52,comm:ksoftirqd/4,prio:120,cpu:4,state:1,id:305,ts:2024-12-27T08:36:00.882149447,ser:24050754368,vr:451953319
pid:11079,comm:collectd,prio:120,cpu:4,state:0,id:310,ts:2024-12-27T08:36:00.882404965,ser:6228023200,vr:4519557164
pid:11079,comm:collectd,prio:120,cpu:4,state:1,id:315,ts:2024-12-27T08:36:00.882614084,ser:6228095136,vr:4519557883
pid:817,comm:irq/90-eth0-rx0,prio:49,cpu:4,state:2,id:320,ts:2024-12-27T08:36:00.883001921,ser:582408730080,vr:0
pid:11076,comm:collectd,prio:120,cpu:4,state:1,id:325,ts:2024-12-27T08:36:00.883218527,ser:6135843392,vr:4519554157
```

```
target:                    ,t_k_id:30141,t_u_id:22662,t_pid:15774,t_tid:16418,waker                    ng,w_k_id:51166,w_u_id:25970,w_pid:156
0.01,q_dur:5.00us,rq_dur:0.00us,rq_len:0,ser:2257056959,vr:3700378228307 w_ser:5484873547,w_dur:38.74us,w_ldur:39.47us
```

4 未来展望

未来展望

进一步优化工具，覆盖更多场景

- 使用更多高版本内核特性: fentry task_storage ringbuffer等
- 特定场景下的优化:map perfoutput

平台化、服务化、可视化、自动化

引入大模型做trace分析及性能优化

谢 谢!

- email: chen.dylane@linux.dev