openEuler

bpftrace内核调测

目录

CONTENT

01 简介

02 打印

03 调用者

04 onCPU

05 offCPU

06 统计延时

openEuler

简介（1）

- 涉及
  - bpftrace的简单介绍
  - bpftrace在内核调测的各种应用

- 不涉及
  - bpftrace的详细介绍
  - bpftrace的实现
  - 用户态的调测

简介（2）

- 基于 eBPF 的调测工具，使用LLVM将脚本编译为eBPF指令
- 实现了高级的脚本语言，类似于awk和c语法

```
bpftrace -e 'tracepoint:syscalls:sys_enter_openat /comm=="cat"/ { printf("%s %s\n", comm, str(args->filename)); }  ‘
```

probe /predicates/ { action; }

openEuler

打印全局变量（1）

```
$ bpftrace -e 'i:s:1 { printf("%lu\n", *((uint64 *)kaddr("jiffies")));}
Attaching 1 probe...
4458065444
4458065694
4458065944
```

```
$ cat read_init_task.bt
#include <linux/sched.h>
BEGIN
{
        $task = (struct task_struct *)kaddr("init_task");
        printf("init task name %s\n", $task->comm);
        exit();
}
$ bpftrace ./read_init_task.bt
```

```
$ cat read_init_task_v2.bt
struct task_struct
{
        char pad[2872];
        char comm[16];
}
BEGIN
{
        $task = (struct task_struct *)kaddr("init_task");
        printf("init task name %s\n", $task->comm);
        exit();
}

$ bpftrace ./read_init_task.bt
Attaching 1 probe...
init task name swapper/0
```

openEuler

打印函数参数（2）

```
$ cat openat.bt
/* do_sys_open(int dfd, const char __user *filename, int flags, umode_t mode) */
k:do_sys_open
{
        @name[tid] = uptr(arg1);
}
kr:do_sys_open /@name[tid]/
{
        if ((int32)retval < 0) {
                printf("%s open %s err %d\n", comm, str(@name[tid]), retval);
        }
        delete(@name[tid]);
}
$ bpftrace ./openat.bt
cat open /proc/1/maps err -13
cat open /proc/1/mapss err -2
```

openEuler

打印分配的对象（3）

```
#include <linux/tcp.h>
k:tcp_write_xmit /@tcp == 0/
{
        $sk = (struct sock *)arg0;
        $family = $sk->__sk_common.skc_family;
        if ($family == AF_INET) {
                $sport = $sk->__sk_common.skc_num;
                $dport = $sk->__sk_common.skc_dport;
                $dport = ($dport >> 8) | (($dport << 8) & 0xff00);

                if ($sport == 22 && $dport == 42666) {
                        @tcp = (struct tcp_sock *)arg0;
                        printf("got sk from %s/%d\n", comm, pid);
                }
        }
}
```

```
i:s:1 /@tcp/
{
        printf("data_segs_out %u una 0x%x wnd 0x%x\n",
                @tcp->data_segs_out, @tcp->snd_una,
                @tcp->snd_wnd);
}

$ bpftrace ./sock_read.bt
Attaching 2 probes...

got sk from sshd/1122026
data_segs_out 899 una 0x2bc6473e wnd 0xfa80
data_segs_out 900 una 0x2bc64792 wnd 0xfa80
data_segs_out 901 una 0x2bc647e6 wnd 0xfa80
data_segs_out 902 una 0x2bc6483a wnd 0xfa80
data_segs_out 903 una 0x2bc6488e wnd 0xfa80
```

openEuler

调用者的堆栈

```
k:__kmalloc,k:__kmalloc_node /2048 < arg0 && arg0 <= 4096/
{
        @detail[comm, kstack] = count();
        @size = lhist(arg0, 2048, 4096, 1024);
}
```

```
$ bpftrace ./kmalloc.bt
@detail[pidstat,
    __kmalloc_node+1
    seq_read_iter+889
    seq_read+262
    vfs_read+149
    ksys_read+95
    do_syscall_64+56
    entry_SYSCALL_64_after_hwframe+68
]: 6696
```

```
k:blkdev_direct_IO
{
        @when[tid] = nsecs;
        @how[tid] = kstack;
}
kr:blkdev_direct_IO /@when[tid]/
{
        $delay = nsecs - @when[tid];
        @usecs[@how[tid]] = hist($delay);
        delete(@when[tid]);
        delete(@how[tid]);
}
```

```
@size:
[3K, 4K)          1 |                                                    |
[4K, ...)      6701 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
```

调用者的CPU

```
#include <linux/irq.h>
#include <linux/interrupt.h>
#include <linux/irqdesc.h>
k:handle_irq_event_percpu
{
        $desc = (struct irq_desc *)arg0;
        @irq[cpu, $desc->action->irq,
            str($desc->action->name)] = count();
}
i:s:2
{
        time("%H:%M:%S\n");
        print(@irq, 5);
}
```

```
$ bpftrace ./irq.bt
Attaching 2 probes...
10:40:54
@irq[59, 274, mpt3sas0-msix23]: 251
@irq[23, 256, mpt3sas0-msix5]: 252
@irq[55, 270, mpt3sas0-msix19]: 253
@irq[58, 273, mpt3sas0-msix22]: 254
@irq[61, 276, mpt3sas0-msix25]: 297
10:40:56
@irq[59, 274, mpt3sas0-msix23]: 502
@irq[23, 256, mpt3sas0-msix5]: 507
@irq[55, 270, mpt3sas0-msix19]: 510
@irq[58, 273, mpt3sas0-msix22]: 513
@irq[61, 276, mpt3sas0-msix25]: 601
```

onCPU

```
profile:hz:9999 /pid == cpid/ {
        @[kstack] = count();
}

$bpftrace ./on_cpu.bt –c    ‘pidstat 1 10’
```

```
@[
    do_task_stat+848
    proc_single_show+74
    seq_read_iter+285
    seq_read+262
    vfs_read+149
    ksys_read+95
    do_syscall_64+56
    entry_SYSCALL_64_after_hwframe+68
]: 63
@[
    syscall_enter_from_user_mode+23
    do_syscall_64+22
    entry_SYSCALL_64_after_hwframe+68
]: 77
```

openEuler

offCPU

```
#include <linux/sched.h>

k:finish_task_switch
{
        $prev = (struct task_struct *)arg0;
        if ($prev->pid == (int32)cpid) {
                @start[$prev->pid] = nsecs;
        }

        $last = @start[tid];
        if ($last != 0) {
                        @off[kstack, comm] = sum(nsecs - $last);
                        delete(@start[tid]);
        }
}
```

```
$ bpftrace ./offcpu.bt -c 'rmmod percpu_rwsem  '
Attaching 2 probes...
@off[
    finish_task_switch+1
    __sched_text_start+1229
    schedule+68
    schedule_timeout+523
    wait_for_completion+140
    __wait_rcu_gp+297
    synchronize_rcu+104
    free_module+201
    __x64_sys_delete_module+436
    do_syscall_64+56
    entry_SYSCALL_64_after_hwframe+68
, rmmod]: 11419102
```

openEuler

offwakeCPU

```
#include <linux/sched.h>
k:try_to_wake_up
{
        $t = (struct task_struct *)arg0;
        if ($t->pid == cpid) {
                @waker[cpid] = kstack(7);
        }
}
k:finish_task_switch
{
        $prev = (struct task_struct *)arg0;
        if ($prev->pid == cpid) {
                @ns[cpid] = nsecs;
        }
        if (tid == cpid && @ns[tid]) {
                $delay = (nsecs - @ns[tid]) / 1000;
                @us[comm, kstack(7), @waker[tid]] = sum($delay);
                delete(@ns[tid]);
                delete(@waker[tid]);
        }
}
```

```
$ bpftrace ./offwakecpu.bt -c 'sleep 1'
@us[sleep,
    finish_task_switch+1
    __sched_text_start+1229
    schedule+68
    do_nanosleep+107
    hrtimer_nanosleep+154
    __x64_sys_nanosleep+164
    do_syscall_64+56
,
    try_to_wake_up+1
    hrtimer_wakeup+30
    __hrtimer_run_queues+244
    hrtimer_interrupt+270
    __sysvec_apic_timer_interrupt+89
    sysvec_apic_timer_interrupt+109
    asm_sysvec_apic_timer_interrupt+18
]: 1000057
```

openEuler

统计延时（1）

```
t:workqueue:workqueue_execute_start
{
        @start[tid] = nsecs;
        @pending[tid] = ksym(args->function);
}
t:workqueue:workqueue_execute_end
/@start[tid]/
{
        $dur = (nsecs - @start[tid]) / 1000;
        @us[@pending[tid]] = stats($dur);
        delete(@start[tid]);
        delete(@pending[tid]);
}
i:s:5
{
        print(@pending);
        print(@us);
}
```

```
$ bpftrace ./wq.bt
@pending[1006577]: vmstat_update
@us[wb_update_bandwidth_workfn]: count 2, average 0, total 1
@us[blk_mq_run_work_fn]: count 11, average 0, total 8
@us[_base_fault_reset_work]: count 5, average 0, total 3
@us[wq_barrier_func]: count 3, average 1, total 3
@us[vmstat_update]: count 11, average 1, total 21
@us[lru_add_drain_per_cpu]: count 7, average 2, total 20
@us[flush_to_ldisc]: count 1, average 4, total 4
@us[blk_mq_requeue_work]: count 1, average 4, total 4
@us[flush_memcg_stats_work]: count 271, average 5, total 1415
@us[gc_worker]: count 2, average 9, total 18
@us[acpi_os_execute_deferred]: count 20, average 43, total 866
@us[flush_memcg_stats_dwork]: count 3, average 45, total 136
@us[wb_workfn]: count 5, average 128, total 641
```

openEuler

统计延时（2）

```
#include <linux/genhd.h>
#include <linux/blkdev.h>

k:blk_mq_start_request
{
	@req[arg0] = nsecs;
}
k:blk_account_io_done /@req[arg0]/
{
	$req = (struct request *)arg0;
	$name = $req->rq_disk->disk_name;
	$delay = (nsecs - @req[arg0]) / 1000;
	@us_dist[$name] = hist($delay);
	@us_stat[$name] = stats($delay);
	delete(@req[arg0]);
}
```

```
@us_dist[sdb]:
[128, 256)     2 |@                                                   |
[256, 512)     2 |@                                                   |
[512, 1K)      4 |@@                                                  |
[1K, 2K)       9 |@@@@@@                                              |
[2K, 4K)      20 |@@@@@@@@@@@@@@                                      |
[4K, 8K)      23 |@@@@@@@@@@@@@@@@                                    |
[8K, 16K)     48 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@                  |
[16K, 32K)    72 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|

@us_dist[nvme0n1]:
[8, 16)     1992 |@                                                   |
[16, 32)       0 |                                                    |
[32, 64)       0 |                                                    |
[64, 128)  57200 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[128, 256)    16 |                                                    |
[256, 512)     7 |                                                    |

@us_stat[nvme0n1]: count 59215, average 74, total 4425093
@us_stat[sdb]: count 180, average 13705, total 2467014
```

参考

https://github.com/iovisor/bpftrace

https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md

<<BPF Performance Tools: Linux System and Application Observability>>

https://github.com/brendangregg/bpf-perf-tools-book.git

openEuler

Thank you

openEuler

开销

```
k:write_null /pid == cpid/
{
        @v1 = count();
}
```

```
k:write_null /pid == cpid/
{
        @start = nsecs;
}
```

```
k:write_null /pid == cpid/
{
        @v2[kstack] = count();
}
```

```
kr:write_null /pid == cpid &&
@start/
{
        $delay = nsecs - @start;
        @v3 = hist($delay);
        @start = 0;
}
```

base	v1	v2	v3	
221.7	282.9	636.5	680.6	nsecs per call

openEuler