ftrace框架及指令修改机制

目录

CONTENT

OpenEuler

什么是ftrace

- Linux官方trace工具，用于了解内核运行时行为
- 2.6.31(2008)进入内核
- 不仅仅是function trace
  - trace event: tracepoint、kprobe event、uprobe event
  - tracer：function、function graph、irqsoff……

Steven Rostedt,

VMware,

Maintainer of ftrace,

Maintainer of PREEMPT_RT patch

OpenEuler

如何使用ftrace

- **内核支持ftrace**

  # make menuconfig

  -> Kernel hacking

  -> Tracers

- **挂载debugfs或者tracefs**

  - **mount –t tracefs nodev /sys/kernel/tracing**
  - **mount –t debugfs nodev /sys/kernel/debug**

- **echo写入控制参数、cat读取数据**

- **文档**

  - **/sys/kernel/debug/tracing/README**
  - **Documentation/trace/***

OpenEuler

tracing目录

```
# ls /sys/kernel/debug/tracing

available_events            hwlat_detector          set_event_pid           trace_clock
available_filter_functions  instances               set_ftrace_filter       trace_marker
available_tracers           kprobe_events           set_ftrace_notrace      trace_marker_raw
buffer_percent              kprobe_profile          set_ftrace_notrace_pid  trace_options
buffer_size_kb              max_graph_depth         set_ftrace_pid          trace_pipe
buffer_total_size_kb        options                 set_graph_function      trace_stat
current_tracer              per_cpu                 set_graph_notrace       tracing_cpumask
dynamic_events              printk_formats          snapshot                tracing_max_latency
dyn_ftrace_total_info       README                  stack_max_size          tracing_on
enabled_functions           saved_cmdlines          stack_trace             tracing_thresh
error_log                   saved_cmdlines_size     stack_trace_filter      uprobe_events
events                      saved_tgids             synthetic_events        uprobe_profile
free_buffer                 set_event               timestamp_mode
function_profile_enabled    set_event_notrace_pid   trace
```

OpenEuler

tracing目录

```
# ls /sys/kernel/debug/tracing

available_events            hwlat_detector          set_event_pid           trace_clock
available_filter_functions  instances               set_ftrace_filter       trace_marker
available_tracers           kprobe_events           set_ftrace_notrace      trace_marker_raw
buffer_percent              kprobe_profile          set_ftrace_notrace_pid  trace_options
buffer_size_kb              max_graph_depth         set_ftrace_pid          trace_pipe
buffer_total_size_kb        options                 set_graph_function      trace_stat
current_tracer              per_cpu                 set_graph_notrace       tracing_cpumask
dynamic_events              printk_formats          snapshot                tracing_max_latency
dyn_ftrace_total_info       README                  stack_max_size          tracing_on
enabled_functions           saved_cmdlines          stack_trace             tracing_thresh
error_log                   saved_cmdlines_size     stack_trace_filter      uprobe_events
events                      saved_tgids             synthetic_events        uprobe_profile
free_buffer                 set_event               timestamp_mode
function_profile_enabled    set_event_notrace_pid   trace
```

OpenEuler

function tracer

```
# cd /sys/kernel/debug/tracing
# echo function > current_tracer
# echo 1 > tracing_on
# cat trace

# tracer: function
#
# entries-in-buffer/entries-written: 75285/75285   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
            bash-382     [007] .... 16017.449444: mutex_unlock <-rb_simple_write
            bash-382     [007] .... 16017.449451: __fsnotify_parent <-vfs_write
            bash-382     [007] d... 16017.449453: exit_to_user_mode_prepare <-syscall_exit_to_user_mode
            bash-382     [007] d... 16017.449453: fpregs_assert_state_consistent <-exit_to_user_mode_prepare
            bash-382     [007] d... 16017.449453: switch_fpu_return <-exit_to_user_mode_prepare
            bash-382     [007] .... 16017.449459: __x64_sys_dup2 <-do_syscall_64
            bash-382     [007] .... 16017.449459: ksys_dup3 <-__x64_sys_dup2
            bash-382     [007] .... 16017.449459: _raw_spin_lock <-ksys_dup3
            bash-382     [007] .... 16017.449460: expand_files <-ksys_dup3
            bash-382     [007] .... 16017.449460: do_dup2 <-ksys_dup3
            bash-382     [007] .... 16017.449460: filp_close <-do_dup2
            bash-382     [007] .... 16017.449460: dnotify_flush <-filp_close
            bash-382     [007] .... 16017.449461: locks_remove_posix <-filp_close
            bash-382     [007] .... 16017.449461: fput <-filp_close
```

OpenEuler

function tracer

- **dynamic tracing**
  - set_ftrace_filter
  - set_ftrace_notrace
  - set_ftrace_pid
  - …

```
# cd /sys/kernel/debug/tracing
# echo "*sched*" > set_ftrace_filter
# echo function > current_tracer
# cat trace

# tracer: function
#
# entries-in-buffer/entries-written: 75285/75285   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
            bash-382     [007] d.h. 65616.238919: tick_sched_timer <-__hrtimer_run_queues
            bash-382     [007] d.h. 65616.238921: tick_sched_handle <-tick_sched_timer
            bash-382     [007] d.h. 65616.238922: rcu_sched_clock_irq <-update_process_times
            bash-382     [007] d.h. 65616.238923: scheduler_tick <-update_process_times
            bash-382     [007] .... 65616.238949: _cond_resched <-ldsem_down_read
            bash-382     [007] .... 65616.238950: _cond_resched <-__flush_work.isra.0
            bash-382     [007] .... 65616.238951: _cond_resched <-do_select
            bash-382     [007] .... 65616.238951: schedule_hrtimeout_range <-do_select
            bash-382     [007] .... 65616.238951: schedule_hrtimeout_range_clock <-schedule_hrtimeout_range
            bash-382     [007] .... 65616.238952: schedule <-schedule_hrtimeout_range_clock
          <idle>-0       [001] d.h. 65616.238982: sched_ttwu_pending <-flush_smp_call_function_queue
          <idle>-0       [001] d.h. 65616.238986: resched_curr <-check_preempt_curr
          <idle>-0       [001] .N.. 65616.238996: schedule_idle <-do_idle
       rcu_sched-11      [001] .... 65616.239000: _cond_resched <-rcu_gp_kthread
       rcu_sched-11      [001] .... 65616.239001: schedule_timeout <-rcu_gp_kthread
```

OpenEuler

function tracer

- filter commands
- format: <function>:<command>:<parameter>
  - traceon/traceoff

    当__schedule_bug被调用5次时关闭tracing：echo '__schedule_bug:traceoff:5' > set_ftrace_filter

  - stacktrace

    某个函数被跟踪时，打印调用栈：echo '__schedule_bug:traceoff:5' > set_ftrace_filter

OpenEuler

trace event

```
# ls /sys/kernel/debug/tracing/events

alarmtimer        exceptions    initcall      mce             page_pool     rtc        thermal
avc               ext4          intel_iommu   mdio            percpu        sched      timer
block             fib           iomap         mei             power         scsi       tlb
bpf_test_run      fib6          iommu         migrate         printk        sctp       ucsi
bpf_trace         filelock      io_uring      mmap            pwm           signal     udp
bridge            filemap       irq           module          qdisc         skb        vmscan
cgroup            fs_dax        irq_matrix    msr             random        smbus      vsyscall
clk               ftrace        irq_vectors   napi            ras           smmu       wbt
compaction        gpio          iscsi         neigh           raw_syscalls  sock       workqueue
context_tracking  header_event  jbd2          net             rbdetonate    spi        writeback
cpuhp             header_page   kmem          nfsd            rcu           sunrpc     x86_fpu
devlink           huge_memory   kvm           nmi             regmap        swiotlb    xdp
dma_fence         hwmon         kvmmmu        oom             rpcgss        syscalls   xen
enable            hyperv        kyber         page_isolation  rpm           task       xfs
error_report      i2c           libata        pagemap         rseq          tcp        xhci-hcd
```

OpenEuler

trace event

```
# ls /sys/kernel/debug/tracing/events/sched

enable                  sched_process_exec  sched_stat_iowait   sched_wait_task
filter                  sched_process_exit  sched_stat_runtime  sched_wake_idle_without_ipi
sched_kthread_stop      sched_process_fork  sched_stat_sleep    sched_wakeup
sched_kthread_stop_ret  sched_process_free  sched_stat_wait     sched_wakeup_new
sched_migrate_task      sched_process_hang  sched_stick_numa    sched_waking
sched_move_numa         sched_process_wait  sched_swap_numa
sched_pi_setprio        sched_stat_blocked  sched_switch
```

```
# ls /sys/kernel/debug/tracing/events/sched/sched_switch

enable  filter  format  hist  id  trigger
```

OpenEuler

trace event

- **event trigger**
  - **kmalloc()申请超过一定大小的内存时，打印调用栈**

    echo 'stacktrace:5 if bytes_req >= 512' > events/kmem/kmalloc/trigger

OpenEuler

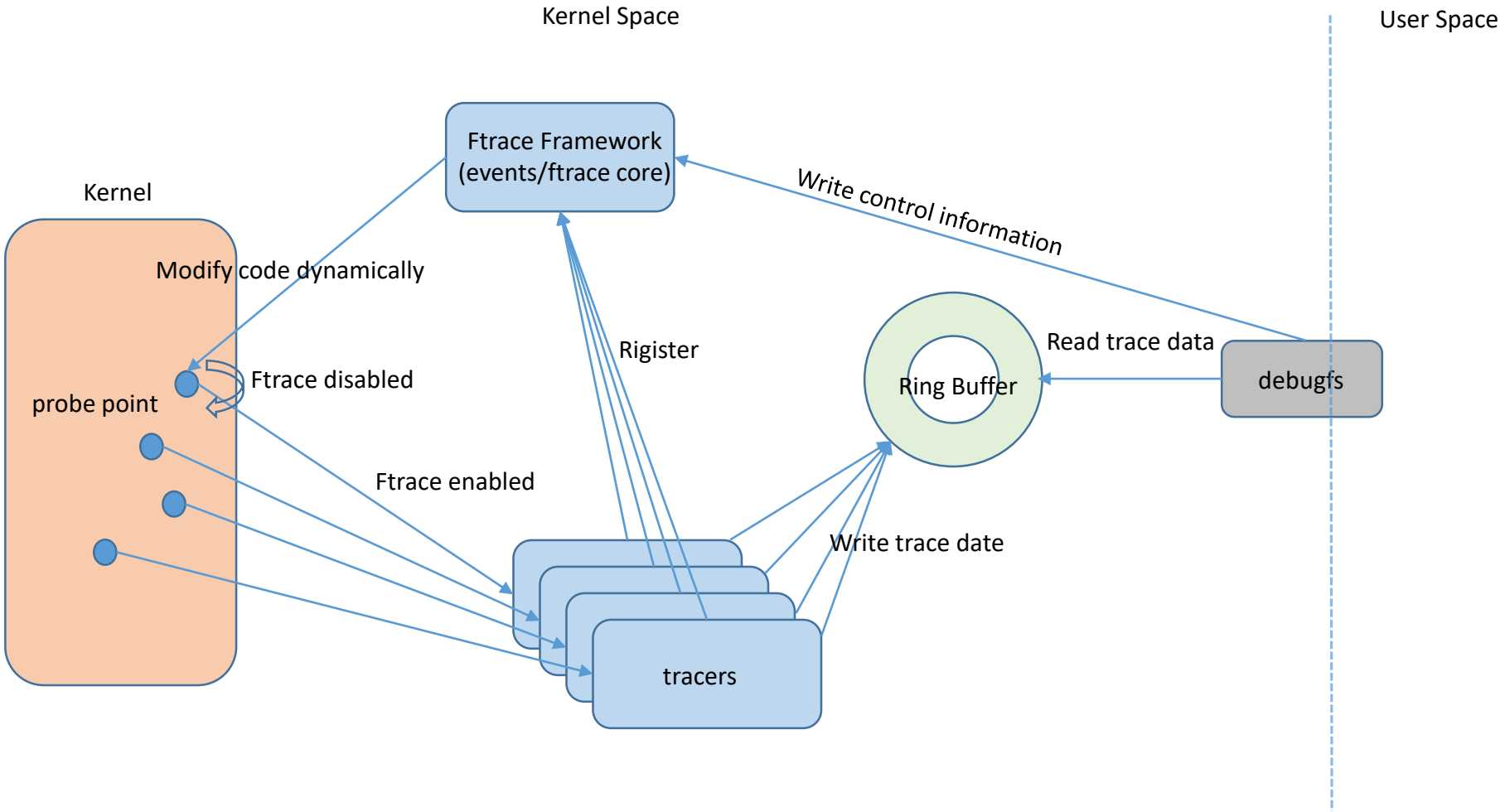
trace event

```
# cd /sys/kernel/debug/tracing
# echo 'stacktrace:5 if bytes_req >= 512' > events/kmem/kmalloc/trigger
# echo 1 > events/kmem/kmalloc/enable
# cat trace

# tracer: nop
#
# entries-in-buffer/entries-written: 128/128   #P:8
#
#                _-----=> irqs-off
#               / _----=> need-resched
#              | / _---=> hardirq/softirq
#              || / _--=> preempt-depth
#              ||| /    delay
#      TASK-PID   CPU# ||||  TIMESTAMP  FUNCTION
#       ||     |  ||||   |     |
 systemd-timesyn-282     [004] .... 17578.503729: kmalloc: call_site=selinux_sk_alloc_security+0x47/0x90 ptr=00000000c1a6e507
bytes_req=32 bytes_alloc=32 gfp_flags=GFP_KERNEL|__GFP_ZERO
          systemd-1      [004] .... 17578.503859: kmalloc: call_site=single_open+0x2f/0xa0 ptr=00000000867b7607 bytes_req=32
bytes_alloc=32 gfp_flags=GFP_KERNEL_ACCOUNT
          systemd-1      [004] .... 17578.503873: kmalloc: call_site=proc_cgroup_show+0x32/0x2b0 ptr=00000000b62a1493 bytes_req=4096
bytes_alloc=4096 gfp_flags=GFP_KERNEL
          systemd-1      [004] .... 17578.503881: <stack trace>
 => kmem_cache_alloc_trace
 => proc_cgroup_show
 => proc_single_show
 => seq_read_iter
 => seq_read
 => vfs_read
 => ksys_read
 => __x64_sys_read
 => do_syscall_64
 => entry_SYSCALL_64_after_hwframe
```

OpenEuler

ftrace框架



OpenEuler

function trace实现

- 函数插桩
- 插桩点动态管理
- 注册ftrace回调函数
- 如何trace

OpenEuler

函数插桩

- 利用gcc –pg选项，在每个函数入口附近插入mcount调用
- x86
  - pg选项插入“callq mcount”
  - gcc 4.6支持-mfentry，在函数序言前插入”__fentry__”调用
  - gcc 5 支持-mrecord-count，-mnop-mcount
- arm64
  - pg选项插入“bl _mcount”
  - gcc 8.1支持-fpatchable-function-entry=N，在函数入口处插入N条nop指令

OpenEuler

函数插桩

- 利用gcc –pg选项，在每个函数入口附近插入mcount调用
- x86
  - pg选项插入“call mcount”
  - gcc 4.6支持-mfentry，在函数序言前插入”__fentry__”调用
  - gcc 5 支持-mrecord-count，-mnop-mcount
- arm64
  - pg选项插入“bl _mcount”
  - gcc 8.1支持-fpatchable-function-entry=N，在函数入口处插入N条nop指令

OpenEuler

内核schedule函数

```
asmlinkage __visible void __sched schedule(void)
{
        struct task_struct *tsk = current;

        sched_submit_work(tsk);
        do {
                preempt_disable();
                __schedule(false);
                sched_preempt_enable_no_resched();
        } while (need_resched());
        sched_update_worker(tsk);
}
```

OpenEuler

内核schedule函数

```
0000000000000670 <schedule>:
 670:   d503233f        paciasp
 674:   a9be7bfd        stp     x29, x30, [sp, #-32]!
 678:   910003fd        mov     x29, sp
 67c:   a90153f3        stp     x19, x20, [sp, #16]
 680:   d5384114        mrs     x20, sp_el0
 684:   f9400e80        ldr     x0, [x20, #24]
 688:   b4000200        cbz     x0, 6c8 <schedule+0x58>
 68c:   b9402e80        ldr     w0, [x20, #44]
 690:   721c041f        tst     w0, #0x30
 694:   54000160        b.eq    6c0 <schedule+0x50>  // b.none
 698:   b9401281        ldr     w1, [x20, #16]
 69c:   11000421        add     w1, w1, #0x1
 6a0:   b9001281        str     w1, [x20, #16]
 6a4:   37280580        tbnz    w0, #5, 754 <schedule+0xe4>
 6a8:   aa1403e0        mov     x0, x20
 ...
```

OpenEuler

内核schedule函数

- **加-pg选项**

```
0000000000000708 <schedule>:
 708:   a9be7bfd        stp     x29, x30, [sp, #-32]!
 70c:   910003fd        mov     x29, sp
 710:   a90153f3        stp     x19, x20, [sp, #16]
 714:   aa1e03e0        mov     x0, x30
 718:   94000000        bl      0 <_mcount>
 71c:   d5384114        mrs     x20, sp_el0
 720:   f9400e80        ldr     x0, [x20, #24]
 724:   b4000200        cbz     x0, 764 <schedule+0x5c>
 728:   b9402e80        ldr     w0, [x20, #44]
 72c:   721c041f        tst     w0, #0x30
 730:   54000160        b.eq    75c <schedule+0x54>  // b.none
 734:   b9401281        ldr     w1, [x20, #16]
 738:   11000421        add     w1, w1, #0x1
 73c:   b9001281        str     w1, [x20, #16]
 740:   37280540        tbnz    w0, #5, 7e8 <schedule+0xe0>
 744:   aa1403e0        mov     x0, x20
 ...
```

OpenEuler

直接替换mcount实现?

- **static ftrace**

```
_mcount:
            mcount_enter

            ldr_l x2, ftrace_trace_function
            adr x0, ftrace_stub
            cmp x0, x2              // if (ftrace_trace_function
            b.eq skip_ftrace_call //          != ftrace_stub) {
            mcount_get_pc x0                  //            function's pc
            mcount_get_lr x1                  //            function's lr (= parent's pc)
            blr x2                            //          (*ftrace_trace_function)(pc, lr);

skip_ftrace_call:                             // }
#ifdef CONFIG_FUNCTION_GRAPH_TRACER
            ldr_l x2, ftrace_graph_return
            cmp x0, x2              // if ((ftrace_graph_return
            b.ne ftrace_graph_caller          //          != ftrace_stub)

            ldr_l x2, ftrace_graph_entry      //    || (ftrace_graph_entry
            adr_l x0, ftrace_graph_entry_stub // != ftrace_graph_entry_stub))
            cmp x0, x2
            b.ne ftrace_graph_caller          // ftrace_graph_caller();
#endif
            mcount_exit
```

OpenEuler

直接替换mcount实现?

- 一旦使能function trace，会对kernel中所有函数(有notrace修饰、inline函数)进行trace，性能开销大，并且trace日志太多，会冲刷掉感兴趣的日志
- 不使能时，也有mcount函数调用开销

OpenEuler

dynamic ftrace

- 不使能时，插桩点替换为nop指令
- 允许只对部分函数进行trace
- 需要对插桩点进行收集和动态管理

OpenEuler

recordmcount

- scripts/recordmcount.c
- 内核编译时，每编译完成一个.c文件，调用recordmcount
- 读取每个.o文件的重定位段
  - 找到mcount调用的地址，先放在临时缓冲区
  - 新增段”__mcount_loc”，并将mcount调用地址写到这个段

OpenEuler

recordmcount

kernel/sched/core.o

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...
```

temporary buffer

```
__mcount_loc:
 &schedu + 0x10
 &preempt_schedule_irq + 0x10
 &schedule_idle + 0xc
 &yield + 0xc
```

OpenEuler

recordmcount

kernel/sched/core.o

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...

__mcount_loc:
 &schedule + 0x10
 &preempt_schedule_irq + 0x10
 &schedule_idle + 0xc
 &yield + 0xc
```

```
__mcount_loc:
 &schedule + 0x10
 &preempt_schedule_irq + 0x10
 &schedule_idle + 0xc
 &yield + 0xc
```

OpenEuler

链接脚本

- include/asm-generic/vmlinux.lds.h
- 在链接脚本中定义变量
  - __start_mcount_loc
  - __stop_mcount_loc

```
#define MCOUNT_REC()     . = ALIGN(8);                                  \
                         __start_mcount_loc = .;                        \
                         KEEP(*(__mcount_loc))                          \
                         KEEP(*(__patchable_function_entries))          \
                         __stop_mcount_loc = .;                         \
                         ftrace_stub_graph = ftrace_stub;
#else
# ifdef CONFIG_FUNCTION_TRACER
#  define MCOUNT_REC()  ftrace_stub_graph = ftrace_stub;
# else
#  define MCOUNT_REC()
# endif
#endif
```

OpenEuler

recordmcount

vmlinux

kernel/sched/core.o

```
__mcount_loc:
 &schedule + 0x10
 &preempt_schedule_irq + 0x10
 &schedule_idle + 0xc
 &yield + 0xc
 ...
```

kernel/module.o

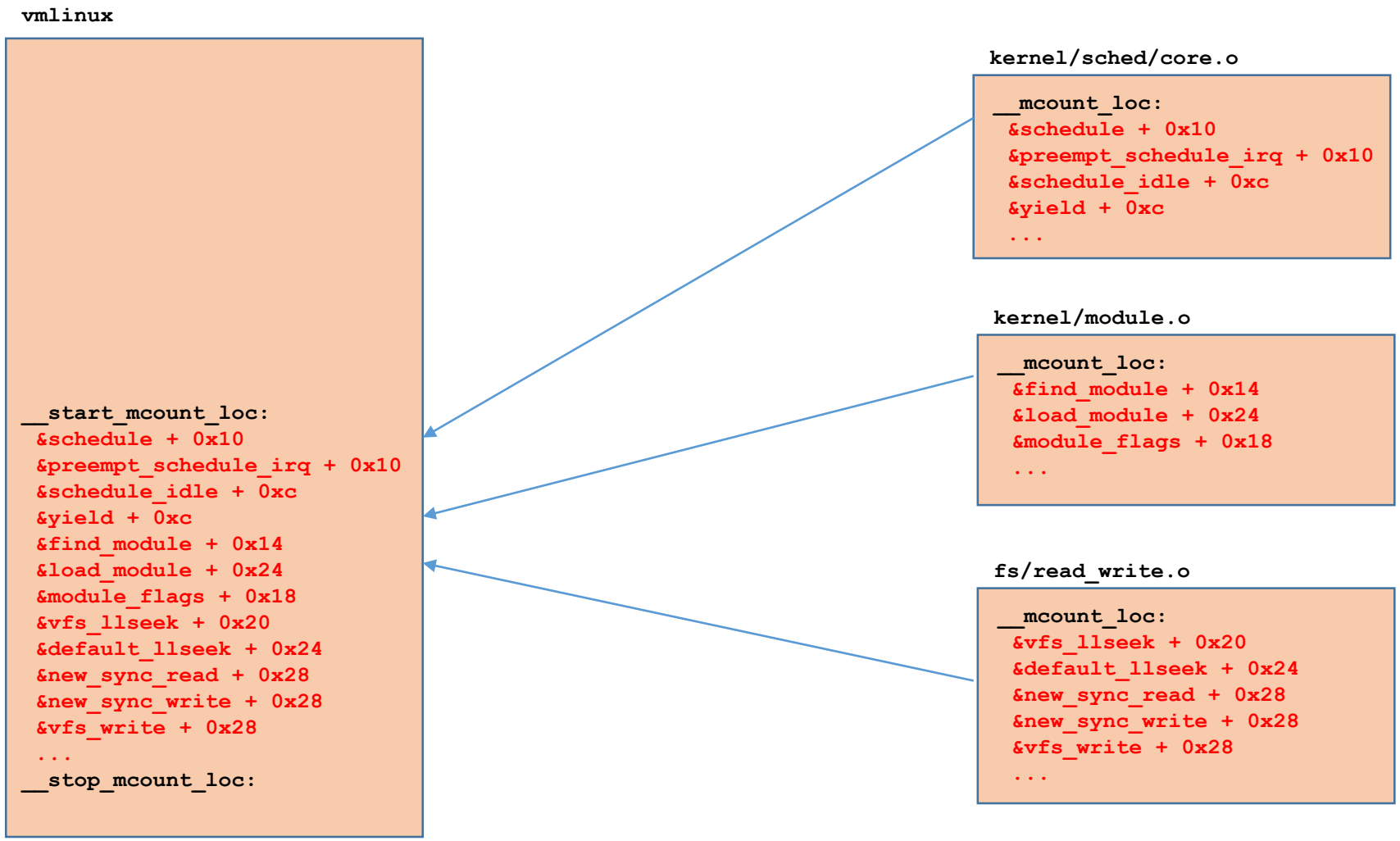
```
__mcount_loc:
 &find_module + 0x14
 &load_module + 0x24
 &module_flags + 0x18
 ...
```

fs/read_write.o

```
__mcount_loc:
 &vfs_llseek + 0x20
 &default_llseek + 0x24
 &new_sync_read + 0x28
 &new_sync_write + 0x28
 &vfs_write + 0x28
 ...
```

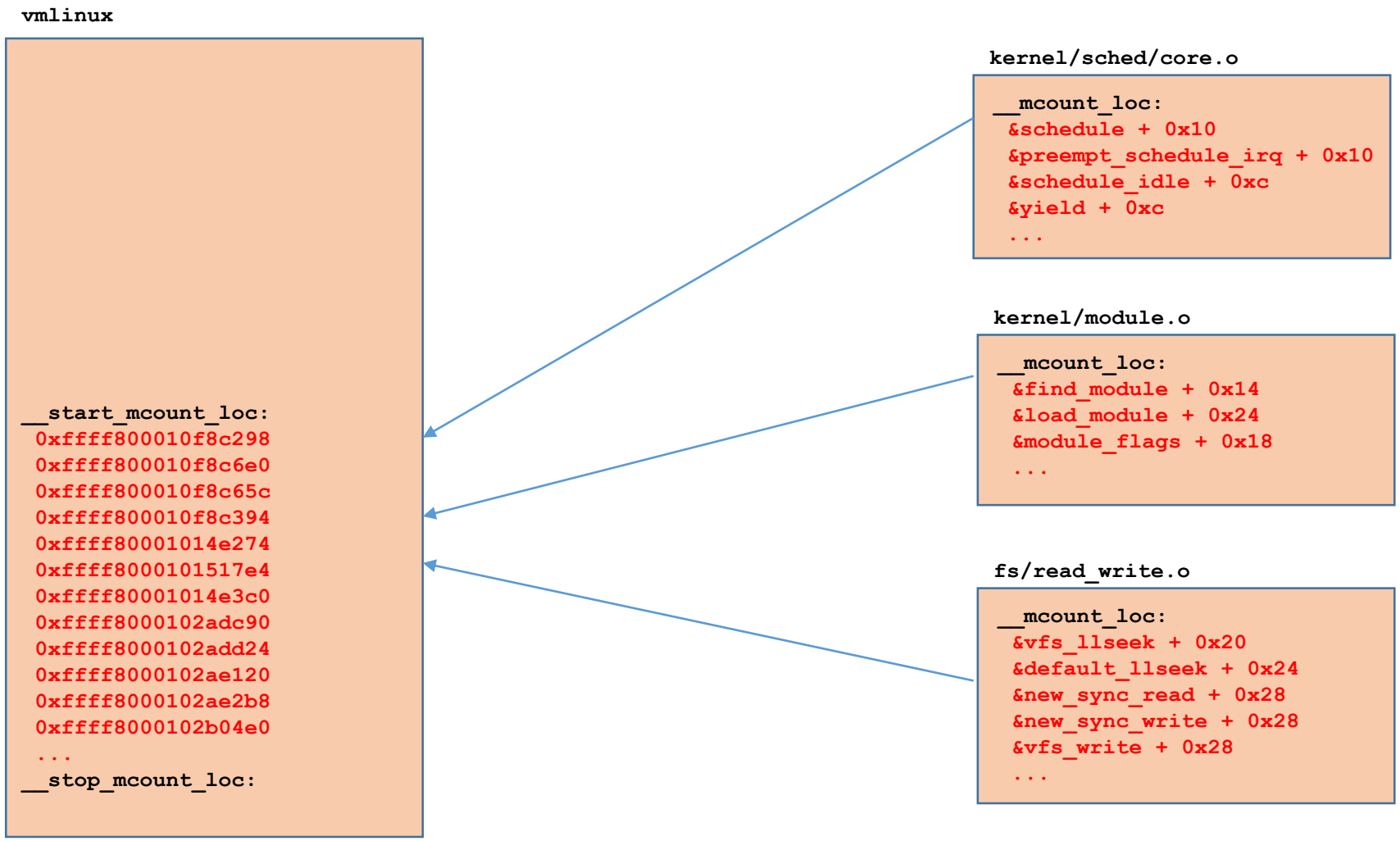
OpenEuler

recordmcount



OpenEuler

recordmcount



OpenEuler

内核启动阶段替换为nop指令

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      0 <_mcount>
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      0 <_mcount>
 ...

__start_mcount_loc:
...
__stop_mcount_loc:
```

OpenEuler

内核启动阶段替换为nop指令

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

__start_mcount_loc:
...
__stop_mcount_loc:
```

OpenEuler

如何trace

- 需要确定想要trace的函数的插桩点位置
- 记录某个trace点的状态
- 建立新的控制结构

OpenEuler

struct dyn_trace

```
struct dyn_ftrace {
        unsigned long ip; /* address of mcount call-site */
        unsigned long flags;
        struct dyn_arch_ftrace arch;
};
```

OpenEuler

struct dyn_trace

```
struct dyn_ftrace {
        unsigned long ip; /* address of mcount call-site */
        unsigned long flags;
        struct dyn_arch_ftrace arch;
};
```

arch/arm/include/asm/ftrace.h

```
#ifdef CONFIG_DYNAMIC_FTRACE
struct dyn_arch_ftrace {
#ifdef CONFIG_ARM_MODULE_PLTS
        struct module *mod;
#endif
};
```

OpenEuler

ftrace_pages

- 按需申请的内存，保存dyn_ftrace数据
- 将__mcount_loc数据迁移到ftrace_pages后，将其释放
- ftrace_pages中数据是有序的，便于查找
- dmesg可看到ftrace_pages占用的内存

```
# dmesg | grep ftrace
[ 0.000000] ftrace: allocating 55680 entries in 218 pages
```

  - Linux 5.10内核
  - 有55680个dyn_ftrace
  - 申请了218个4K pages，即872K内存

OpenEuler

ftrace_pages

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

__start_mcount_loc:
...
__stop_mcount_loc:
```

ftrace_pages

```
...
ip   = 0xffff800010f8c298
flag = 0
ip   = 0xffff800010f8c6e0
flag = 0
ip   = 0xffff800010f8c65c
flag = 0
ip   = 0xffff800010f8c394
flag = 0
ip   = 0xffff80001014e274
flag = 0
ip   = 0xffff8000101517e4
flag = 0
ip   = 0xffff80001014e3c0
flag = 0
ip   = 0xffff8000102adc90
flag = 0
ip   = 0xffff8000102add24
flag = 0
ip   = 0xffff8000102ae120
flag = 0
ip   = 0xffff8000102ae2b8
flag = 0
ip   = 0xffff8000102b04e0
flag = 0
...
```

OpenEuler

ftrace_pages

ftrace_pages

```
...
ip   = 0xffff800010f8c298
flag = 0
ip   = 0xffff800010f8c6e0
flag = 0
ip   = 0xffff800010f8c65c
flag = 0
ip   = 0xffff800010f8c394
flag = 0
ip   = 0xffff80001014e274
flag = 0
ip   = 0xffff8000101517e4
flag = 0
ip   = 0xffff80001014e3c0
flag = 0
ip   = 0xffff8000102adc90
flag = 0
ip   = 0xffff8000102add24
flag = 0
ip   = 0xffff8000102ae120
flag = 0
ip   = 0xffff8000102ae2b8
flag = 0
ip   = 0xffff8000102b04e0
flag = 0
...
```

```
# cat available_filter_functions

schedule
preempt_schedule_irq
schedule_idle
yield
find_module
load_module
module_flags
vfs_llseek
default_llseek
new_sync_read
new_sync_write
vfs_write
```

OpenEuler

dyn_ftrace flag

```
enum {
        FTRACE_FL_ENABLED = (1UL << 31),
        FTRACE_FL_REGS = (1UL << 30),
        FTRACE_FL_REGS_EN = (1UL << 29),
        FTRACE_FL_TRAMP = (1UL << 28),
        FTRACE_FL_TRAMP_EN = (1UL << 27),
        FTRACE_FL_IPMODIFY = (1UL << 26),
        FTRACE_FL_DISABLED = (1UL << 25),
        FTRACE_FL_DIRECT = (1UL << 24),
        FTRACE_FL_DIRECT_EN = (1UL << 23),
};
```

* ENABLED - the function is being traced
* REGS - the record wants the function to save regs
* REGS_EN - the function is set up to save regs.
* IPMODIFY - the record allows for the IP address to be changed.
* DISABLED - the record is not ready to be touched yet
* DIRECT - there is a direct function to call

OpenEuler

dyn_ftrace flag

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...
```

ftrace_pages

```
...
ip   = 0xffff800010f8c298
flag = 0
ip   = 0xffff800010f8c6e0
flag = 0
ip   = 0xffff800010f8c65c
flag = 0
ip   = 0xffff800010f8c394
flag = 0
ip   = 0xffff80001014e274
flag = 0
ip   = 0xffff8000101517e4
flag = 0
ip   = 0xffff80001014e3c0
flag = 0
ip   = 0xffff8000102adc90
flag = 0
ip   = 0xffff8000102add24
flag = 0
ip   = 0xffff8000102ae120
flag = 0
ip   = 0xffff8000102ae2b8
flag = 0
ip   = 0xffff8000102b04e0
flag = 0
...
```

OpenEuler

dyn_ftrace flag

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...
```

ftrace_pages

```
...
ip   = 0xffff800010f8c298
flag = 0x40000001
ip   = 0xffff800010f8c6e0
flag = 0
ip   = 0xffff800010f8c65c
flag = 0
ip   = 0xffff800010f8c394
flag = 0x1
ip   = 0xffff80001014e274
flag = 0
ip   = 0xffff8000101517e4
flag = 0
ip   = 0xffff80001014e3c0
flag = 0
ip   = 0xffff8000102adc90
flag = 0
ip   = 0xffff8000102add24
flag = 0
ip   = 0xffff8000102ae120
flag = 0
ip   = 0xffff8000102ae2b8
flag = 0
ip   = 0xffff8000102b04e0
flag = 0
...
```

bit 30
count 1

count 1

OpenEuler

dyn_ftrace flag

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_reg_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      ftrace_caller
 ...
```

ftrace_pages

```
...
ip   = 0xffff800010f8c298
flag = 0xe0000001
ip   = 0xffff800010f8c6e0
flag = 0
ip   = 0xffff800010f8c65c
flag = 0
ip   = 0xffff800010f8c394
flag = 0x80000001
ip   = 0xffff80001014e274
flag = 0
ip   = 0xffff8000101517e4
flag = 0
ip   = 0xffff80001014e3c0
flag = 0
ip   = 0xffff8000102adc90
flag = 0
ip   = 0xffff8000102add24
flag = 0
ip   = 0xffff8000102ae120
flag = 0
ip   = 0xffff8000102ae2b8
flag = 0
ip   = 0xffff8000102b04e0
flag = 0
...
```

bit 29,30,31
count 1

bit 31
count 1

OpenEuler

ftrace entry

Dynamic ftrace

```
ftrace_caller:
          mcount_enter

          mcount_get_pc x0                  //          function's pc
          mcount_get_lr x1                  //          function's lr

ftrace_call:                                // tracer(pc, lr);
          nop                               // This will be replaced with "bl xxx"
                                            // where xxx can be any kind of tracer.
#ifdef CONFIG_FUNCTION_GRAPH_TRACER
ftrace_graph_call:                          // ftrace_graph_caller();
          nop                               // If enabled, this will be replaced
                                            // "b ftrace_graph_caller"
#endif
          mcount_exit
```

OpenEuler

ftrace entry

Dynamic ftrace with registers

- gcc支持-fpatchable-function-entry
- 设置-fpatchable-function-entry=2

OpenEuler

ftrace entry

Dynamic ftrace with registers

- gcc支持-fpatchable-function-entry
- 设置-fpatchable-function-entry=2

```
| Compiled |   Disabled    |   Enabled    |
+----------+---------------+--------------+
| NOP      | MOV X9, LR    | MOV X9, LR   |
| NOP      | NOP           | BL <entry>   |
```

OpenEuler

ftrace entry

Dynamic ftrace

```
ftrace_caller:
        mcount_enter

        mcount_get_pc x0                // function's pc
        mcount_get_lr x1                // function's lr

ftrace_call:                            // tracer(pc, lr);
        nop                             // This will be replaced with "bl xxx"
                                        // where xxx can be any kind of tracer.
#ifdef CONFIG_FUNCTION_GRAPH_TRACER
ftrace_graph_call:                      // ftrace_graph_caller();
        nop                             // If enabled, this will be replaced
                                        // "b ftrace_graph_caller"
#endif
        mcount_exit
```

OpenEuler

注册ftrace回调函数

- 调用register_ftrace_function()注册
- 提供ftrace_ops
- Static ftrace_ops
  - function
  - function graph
  - blk
  - kprobe
- Dynamic ftrace_ops
  - perf

OpenEuler

struct ftrace_ops

```
struct ftrace_ops {
        ftrace_func_t                   func;
        struct ftrace_ops __rcu         *next;
        unsigned long                   flags;
        void                            *private;
        ftrace_func_t                   saved_func;
#ifdef CONFIG_DYNAMIC_FTRACE
        struct ftrace_ops_hash          local_hash;
        struct ftrace_ops_hash          *func_hash;
        struct ftrace_ops_hash          old_hash;
        unsigned long                   trampoline;
        unsigned long                   trampoline_size;
        struct list_head                list;
#endif
};
```

OpenEuler

ftrace entry

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...
```

```
ftrace_caller:
 save regs
 load regs
ftrace_call:
 nop
 ret
```

OpenEuler

ftrace_ops_list_func

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...
```

```
ftrace_caller:
 save regs
 load regs
ftrace_call:
 bl ftrace_ops_list_func
 ret
```

```
ftrace_ops_list_func()
{
    /* itetate ops */
}
```

OpenEuler

ftrace_ops_list_func

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 nop
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 nop
 ...
```

```
ftrace_caller:
 save regs
 load regs
ftrace_call:
 bl ftrace_ops_list_func
 ret
```

```
ftrace_ops_list_func()
{
    /* itetate ops */
}
```

```
perf_ops_func()
{
    /* do trace */
}
```

```
func_trace_ops_func()
{
    /* do trace */
}
```

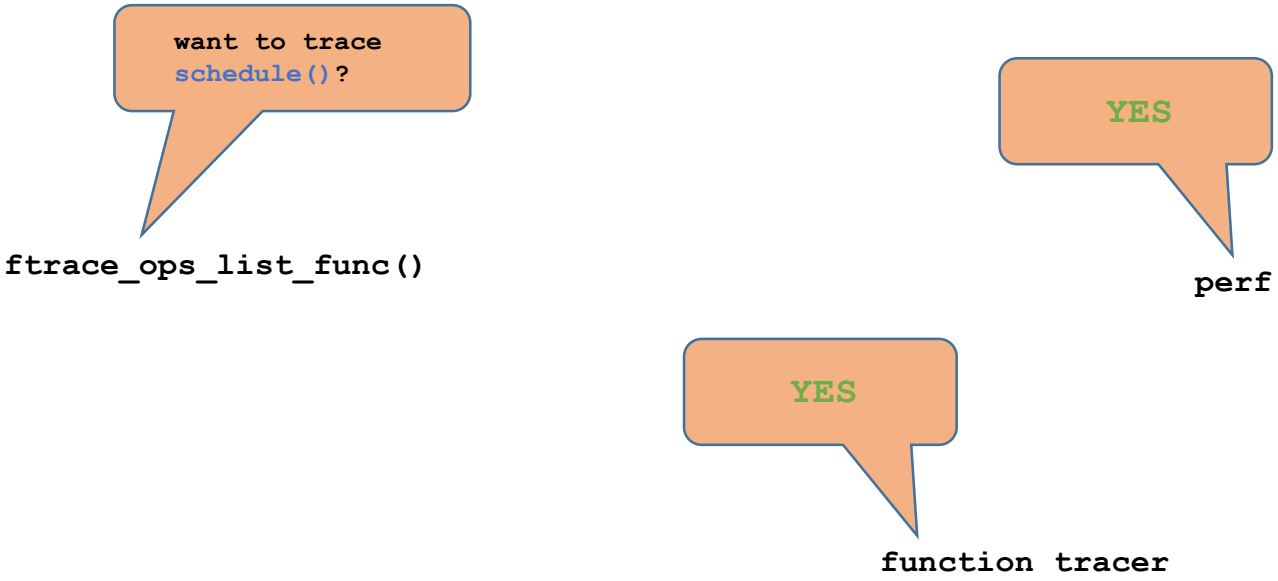
OpenEuler

ftrace回调的一种场景

- function tracer需要trace所有的函数
- perf只想trace schedule()函数

OpenEuler

有什么问题



OpenEuler

有什么问题

want to trace schedule_idle()?

ftrace_ops_list_func()

NO

perf

YES

function tracer

OpenEuler

有什么问题

want to trace yield()?

ftrace_ops_list_func()

NO

perf

YES

function tracer

OpenEuler

ftrace_ops trampoline

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      ftrace_caller
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      ftrace_caller
 ...
```

```
ftrace_caller:
 save regs
 load regs
ftrace_call:
 bl ftrace_ops_list_func
 ret
```

```
ftrace_ops_list_func()
{
    /* itetate ops */
}
```

```
perf_ops_func()
{
    /* do trace */
}
```

```
func_trace_ops_func()
{
    /* do trace */
}
```

OpenEuler

ftrace_ops trampoline

vmlinux

```
<schedule>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      ftrace_caller
 ...

<preempt_schedule_irq>:
 stp     x29, x30, [sp, #-32]!
 mov     x29, sp
 stp     x19, x20, [sp, #16]
 mov     x0, x30
 bl      dynamic_trampoline
 ...

<schedule_idle>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      dynamic_trampoline
 ...

<yield>:
 stp     x29, x30, [sp, #-16]!
 mov     x29, sp
 mov     x0, x30
 bl      dynamic_trampoline
 ...
```

```
ftrace_caller:
 save regs
 load regs
ftrace_call:
 bl ftrace_ops_list_func
 ret
```

```
ftrace_ops_list_func()
{
    /* itetate ops */
}
```

```
perf_ops_func()
{
    /* do trace */
}
```

```
func_trace_ops_func()
{
    /* do trace */
}
```

```
dynamic_trampoline:
 save regs
 load regs
ftrace_call:
 bl func_trace_ops_func
 ret
```

OpenEuler

ftrace hook应用

- **ftrace_reg_caller准备了所有寄存器**
  - kprobe
- **可以修改pc寄存器**
  - livepatch

OpenEuler

基于ftrace的热补丁(x86)

Buggy schedule() function:

```
<schedule>:
 callq      ftrace_caller
 ...
```

```
<ftrace_caller>:
 save regs
 load regs
 callq klp_ftrace_handler
 restore regs
 retq
```

```
klp_ftrace_hanler()
{
  ...
  regs.ip = schedule_fix;
  ...
}
```

Fixed schedule() function:

```
<schedule>:
 nop
 ...
```

OpenEuler

参考资料

[1] Kernel Recipes 2017 - Understanding the Linux Kernel via Ftrace

[2] Kernel Recipes 2019 - ftrace: Where modifying a running kernel all started

[3] Tracing With Ftrace: Critical Tooling For Linux Development

[4] Ftrace原理和代码分析

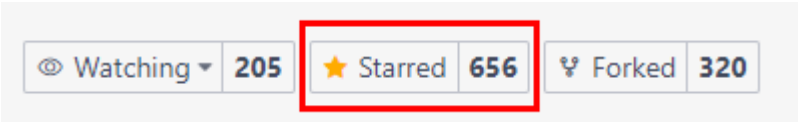
OpenEuler

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

OpenEuler

Thank you

OpenEuler