LINUX PLUMBERS CONFERENCE September 20-24, 2021

Function tracing with arguments

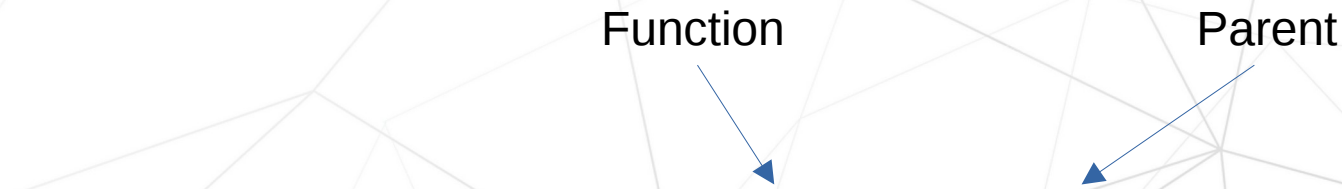
Steven Rostedt

VMware Inc.

Function tracing

```
# trace-cmd start -p function
# trace-cmd show
# tracer: function
#
# entries-in-buffer/entries-written: 268213/530953   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
          <idle>-0       [002] dN.1 197690.658887: rcu_idle_exit <-cpuidle_enter_state
          <idle>-0       [002] dN.1 197690.658887: sched_idle_set_state <-cpuidle_enter_state
          <idle>-0       [002] .N.1 197690.658888: cpuidle_reflect <-do_idle
          <idle>-0       [002] .N.1 197690.658888: menu_reflect <-do_idle
          <idle>-0       [002] .N.1 197690.658888: tick_nohz_idle_got_tick <-menu_reflect
          <idle>-0       [002] .N.1 197690.658888: arch_cpu_idle_exit <-do_idle
          <idle>-0       [002] .N.1 197690.658888: tick_nohz_idle_exit <-do_idle
          <idle>-0       [002] dN.1 197690.658889: ktime_get <-tick_nohz_idle_exit
          <idle>-0       [002] dN.1 197690.658889: nr_iowait_cpu <-tick_nohz_idle_exit
          <idle>-0       [002] dN.1 197690.658889: tick_nohz_restart_sched_tick <-
```

Function and Parent



What about Arguments?

- Would be nice to see the arguments

What about Arguments?

- Would be nice to see the arguments
- How hard would that be?

What about Arguments?

- Would be nice to see the arguments
- How hard would that be?
- Need way to extract arguments (regs and stack pointer)

What about Arguments?

- Would be nice to see the arguments
- How hard would that be?
- Need way to extract arguments (regs and stack pointer)
- Need way to map to arguments (What regs are what arg)
  - Quick look up table

What about Arguments?

- Would be nice to see the arguments
- How hard would that be?
- Need way to extract arguments (regs and stack pointer)
- Need way to map to arguments (What regs are what arg)
  - Quick look up table
- Need way to store it into the ring buffer

What about Arguments?

- Would be nice to see the arguments
- How hard would that be?
- Need way to extract arguments (regs and stack pointer)
- Need way to map to arguments (What regs are what arg)
  - Quick look up table
- Need way to store it into the ring buffer
- Need way to read it from the ring buffer

Function hook

```
<schedule>:
        call    ftrace_caller
        push    %rbx
        mov     %gs:0x16100,%rbx
        mov     0x10(%rbx),%rax
        test    %rax,%rax
```

```
ftrace_caller:
        save_args
        mov     IP(%rsp), %rdi
        mov     PIP(%rsp), %rsi
        mov     ftrace_ops, %rdx
        mov     $0, %rcx
        call    callback
```

Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct pt_regs *regs)
```

Function regs hook

```
<schedule>:
        call    ftrace_regs_caller
        push    %rbx
        mov     %gs:0x16100,%rbx
        mov     0x10(%rbx),%rax
        test    %rax,%rax
```

```
ftrace_regs_caller:
        save_regs
        mov     IP(%rsp), %rdi
        mov     PIP(%rsp), %rsi
        mov     ftrace_ops, %rdx
        mov     %rsp, %rcx
        call    callback
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct pt_regs *regs)
{
        if (!regs)
                return;
```

Function hook

```
<schedule>:
        call    ftrace_caller
        push    %rbx
        mov     %gs:0x16100,%rbx
        mov     0x10(%rbx),%rax
        test    %rax,%rax
```

```
ftrace_caller:
        save_args
        mov     IP(%rsp), %rdi
        mov     PIP(%rsp), %rsi
        mov     ftrace_ops, %rdx
        mov     $0, %rcx
        call    callback
```

Function hook

```
<schedule>:
        call    ftrace_caller
        push    %rbx
        mov     %gs:0x16100,%rbx
        mov     0x10(%rbx),%rax
        test    %rax,%rax
```

```
ftrace_caller:
        save_args
        mov     IP(%rsp), %rdi
        mov     PIP(%rsp), %rsi
        mov     ftrace_ops, %rdx
        mov     $0, %rcx
        call    callback
```

Function hook

```
<schedule>:
        call    ftrace_caller
        push    %rbx
        mov     %gs:0x16100,%rbx
        mov     0x10(%rbx),%rax
        test    %rax,%rax
```

```
ftrace_caller:
    save_args
    mov     IP(%rsp), %rdi
    mov     PIP(%rsp), %rsi
    mov     ftrace_ops, %rdx
    mov     %rsp, %rcx
    call    callback
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct pt_regs *regs)
{
        if (!regs)
                return;
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct pt_regs *regs)
{
        if (!regs)
                return;
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct pt_regs *regs)
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct ftrace_regs *fregs)
```

Regs Callback hook

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct ftrace_regs *fregs)
{
        struct pt_regs *regs = ftrace_get_regs(fregs);

        if (!regs)
                return;
```

Mapping regs to args

- We have the regs

Mapping regs to args

- We have the regs
- Need to map to args (each function is different)

Mapping regs to args

- We have the regs
- Need to map to args (each function is different)
- BTF can define the arguments for every function

Mapping regs to args

- We have the regs
- Need to map to args (each function is different)
- BTF can define the arguments for every function
- But can it do it quickly?

Mapping regs to args

- Use ftrace_ops mapping?
  - Can increase size of kernel footprint

Mapping regs to args

- Use ftrace_ops mapping?
  - Can increase size of kernel footprint
- External module to load hash mapping
  - Can map function address to BTF table

Mapping regs to args

- Use ftrace_ops mapping?
  - Can increase size of kernel footprint
- External module to load hash mapping
  - Can map function address to BTF table
- Still need BTF processing

BTF argument processing?

```
void callback(unsigned long ip, unsigned long pip,
              struct ftrace_ops *ops, struct ftrace_regs *fregs)
{
        struct btf *btf;
        char *tmp_buf = get_temp_buffer();
        int size;

        btf = btf_hash_lookup(ip);

        size = btf_write_args(tmp_buf, btf, fregs);

        ring_buffer_write(tmp_buf, size);
}
```

BTF argument processing?

```
btf_write_args(tmp_buf, btf, fregs);

tmp_buf :

    char name[]          : string ending in '\0'

    type                 : defining the argument data type (0 for last one)
                           Examples: u8, s8, u16, s16, u32, s32, ...
                           Also defines the size of data

    data                 : The data defined by type
```