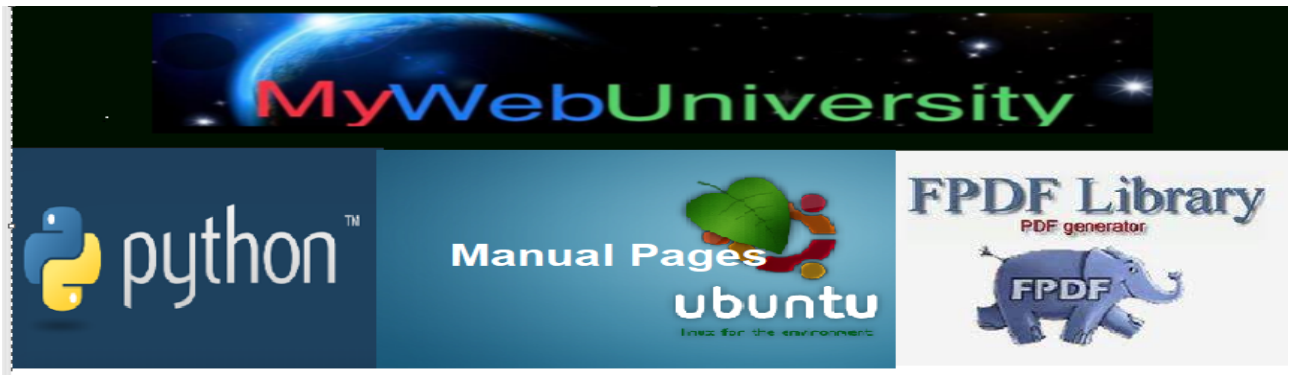

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'trace-cmd-profile.1'

$ man trace-cmd-profile.1

TRACE-CMD-PROFILE(1) libtracefs Manual TRACE-CMD-PROFILE(1)

NAME

trace-cmd-profile - profile tasks running live

SYNOPSIS

trace-cmd profile [OPTIONS] [command]

DESCRIPTION

The trace-cmd(1) profile will start tracing just like trace-cmd-record(1), with the --profile option, except that it does not write to a file, but instead, it will read the events as they happen and will update the accounting of the events. When the trace is finished, it will report the results just like trace-cmd-report(1) would do with its --profile option. In other words, the profile command does the work of trace-cmd record --profile, and trace-cmd report --profile without having to record the data to disk, in between.

The advantage of using the profile command is that the profiling can be done over a long period of time where recording all events would take up too much disk space.

This will enable several events as well as the function graph tracer with a depth of one (if the kernel supports it). This is to show where tasks enter and exit the kernel and how long they were in the kernel.

To disable calling function graph, use the -p option to enable another tracer. To not enable any tracer, use -p nop.

All timings are currently in nanoseconds.

OPTIONS

These are the same as trace-cmd-record(1) with the --profile option.

-p tracer

Set a tracer plugin to run instead of function graph tracing set to depth of 1. To not run any tracer, use -p nop.

-S

Only enable the tracer or events speficied on the command line. With this option, the function_graph tracer is not enabled, nor are any events (like sched_switch), unless they are specifically specified on the command line (i.e. -p function -e sched_switch -e sched_wakeup)

-G

Set interrupt (soft and hard) events as global (associated to CPU instead of tasks).

-o file

Write the output of the profile to file. This supersedes --stderr

-H event-hooks

Add custom event matching to connect any two events together. Format is:

[<start_system>:]<start_event>,<start_match>[,<start_pid>]/

[<end_system>:]<end_event>,<end_match>[,<flags>]

The start_system:start_event (start_system is optional), is the event that starts the timing.

start_match is the field in the start event that is to match with the end_match in the end event.

start_pid is optional, as matches are attached to the tasks that run the events, if another field should be used to find that task, then it is specified with start_pid.

end_system:end_event is the event that ends the timing (end_system is optional).

end_match is the field in end_match that wil match the start event field start_match.

flags are optional and can be the following (case insensitive):

p : The two events are pinned to the same CPU (start and end happen on the same CPU always).

s : The event should have a stack traced with it (enable stack tracing for the start event).

g : The event is global (not associated to a task). start_pid is

not applicable with this flag.

--stderr

Redirect the output to stderr. The output of the command being executed is not changed.

This allows watching the command execute and saving the output of the profile to another file.

--verbose[=level]

Set the log level. Supported log levels are "none", "critical", "error", "warning", "info", "debug", "all" or their identifiers "0", "1", "2", "3", "4", "5", "6". Setting the log level to specific value enables all logs from that and all previous levels. The level will default to "info" if one is not specified.

Example: enable all critical, error and warning logs

trace-cmd profile --verbose=warning

EXAMPLES

```
# trace-cmd profile -F sleep 1
 [..]
task: sleep-1121
  Event: sched_switch:R (2) Total: 234559 Avg: 117279 Max: 129886 Min:104673
        |
        + ftrace_raw_event_sched_switch (0xffffffff8109f310)
           100% (2) time:234559 max:129886 min:104673 avg:117279
            __schedule (0xffffffff816c1e81)
            preempt_schedule (0xffffffff816c236e)
            ___preempt_schedule (0xffffffff81351a59)
             |
             + unmap_single_vma (0xffffffff81198c05)
             |   55% (1) time:129886 max:129886 min:0 avg:129886
             |    stop_one_cpu (0xffffffff8110909a)
             |    sched_exec (0xffffffff810a119b)
             |    do_execveat_common.isra.31 (0xffffffff811de528)
             |    do_execve (0xffffffff811dea8c)
             |    SyS_execve (0xffffffff811ded1e)
             |    return_to_handler (0xffffffff816c8458)
```

```
            |    stub_execve (0xffffffff816c6929)
            |
            + unmap_single_vma (0xffffffff81198c05)
               45% (1) time:104673 max:104673 min:0 avg:104673
                unmap_vmas (0xffffffff81199174)
                exit_mmap (0xffffffff811a1f5b)
                mmput (0xffffffff8107699a)
                flush_old_exec (0xffffffff811ddb75)
                load_elf_binary (0xffffffff812287df)
                search_binary_handler (0xffffffff811dd3e0)
                do_execveat_common.isra.31 (0xffffffff811de8bd)
                do_execve (0xffffffff811dea8c)
                SyS_execve (0xffffffff811ded1e)
                return_to_handler (0xffffffff816c8458)
                stub_execve (0xffffffff816c6929)
Event: sched_switch:S (1) Total: 1000513242 Avg: 1000513242 Max: 1000513242 Min:1000513242
      |
      + ftrace_raw_event_sched_switch (0xffffffff8109f310)
         100% (1) time:1000513242 max:1000513242 min:0 avg:1000513242
          __schedule (0xffffffff816c1e81)
          schedule (0xffffffff816c23b9)
          do_nanosleep (0xffffffff816c4f1c)
          hrtimer_nanosleep (0xffffffff810dcd86)
          SyS_nanosleep (0xffffffff810dcea6)
          return_to_handler (0xffffffff816c8458)
          tracesys_phase2 (0xffffffff816c65b0)
Event: sched_wakeup:1121 (1) Total: 43405 Avg: 43405 Max: 43405 Min:43405
      |
      + ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)
         100% (1) time:43405 max:43405 min:0 avg:43405
          ttwu_do_wakeup (0xffffffff810a01a2)
          ttwu_do_activate.constprop.122 (0xffffffff810a0236)
          try_to_wake_up (0xffffffff810a3ec3)
```

wake_up_process (0xffffffff810a4057)

hrtimer_wakeup (0xffffffff810db772)

__run_hrtimer (0xffffffff810dbd91)

hrtimer_interrupt (0xffffffff810dc6b7)

local_apic_timer_interrupt (0xffffffff810363e7)

smp_trace_apic_timer_interrupt (0xffffffff816c8c6a)

trace_apic_timer_interrupt (0xffffffff816c725a)

finish_task_switch (0xffffffff8109c3a4)

__schedule (0xffffffff816c1e01)

schedule (0xffffffff816c23b9)

ring_buffer_wait (0xffffffff811323a3)

wait_on_pipe (0xffffffff81133d93)

tracing_buffers_splice_read (0xffffffff811350b0)

do_splice_to (0xffffffff8120476f)

SyS_splice (0xffffffff81206c1f)

tracesys_phase2 (0xffffffff816c65b0)

Event: func: sys_nanosleep() (1) Total: 1000598016 Avg: 1000598016 Max: 1000598016 Min:1000598016

Event: func: sys_munmap() (1) Total: 14300 Avg: 14300 Max: 14300 Min:14300

Event: func: sys_arch_prctl() (1) Total: 571 Avg: 571 Max: 571 Min:571

Event: func: sys_mprotect() (4) Total: 14382 Avg: 3595 Max: 7196 Min:2190

Event: func: SyS_read() (1) Total: 2640 Avg: 2640 Max: 2640 Min:2640

Event: func: sys_close() (5) Total: 4001 Avg: 800 Max: 1252 Min:414

Event: func: sys_newfstat() (3) Total: 11684 Avg: 3894 Max: 10206 Min:636

Event: func: SyS_open() (3) Total: 23615 Avg: 7871 Max: 10535 Min:4743

Event: func: sys_access() (1) Total: 5924 Avg: 5924 Max: 5924 Min:5924

Event: func: SyS_mmap() (8) Total: 39153 Avg: 4894 Max: 12354 Min:1518

Event: func: smp_trace_apic_timer_interrupt() (1) Total: 10298 Avg: 10298 Max: 10298 Min:10298

Event: func: SyS_brk() (4) Total: 2407 Avg: 601 Max: 1564 Min:206

Event: func: do_notify_resume() (2) Total: 4095 Avg: 2047 Max: 2521 Min:1574

Event: func: sys_execve() (5) Total: 1625251 Avg: 325050 Max: 1605698 Min:3570

|

+ ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)

100% (1) time:1605698 max:1605698 min:0 avg:1605698

ttwu_do_wakeup (0xffffffff810a01a2)

ttwu_do_activate.constprop.122 (0xffffffff810a0236)

try_to_wake_up (0xffffffff810a3ec3)

wake_up_process (0xffffffff810a4057)

cpu_stop_queue_work (0xffffffff81108df8)

stop_one_cpu (0xffffffff8110909a)

sched_exec (0xffffffff810a119b)

do_execveat_common.isra.31 (0xffffffff811de528)

do_execve (0xffffffff811dea8c)

SyS_execve (0xffffffff811ded1e)

return_to_handler (0xffffffff816c8458)

stub_execve (0xffffffff816c6929)

stub_execve (0xffffffff816c6929)

Event: func: syscall_trace_enter_phase2() (38) Total: 21544 Avg: 566 Max: 1066 Min:329

Event: func: syscall_trace_enter_phase1() (38) Total: 9202 Avg: 242 Max: 376 Min:150

Event: func: __do_page_fault() (53) Total: 257672 Avg: 4861 Max: 27745 Min:458

|

+ ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)

100% (1) time:27745 max:27745 min:0 avg:27745

ttwu_do_wakeup (0xffffffff810a01a2)

ttwu_do_activate.constprop.122 (0xffffffff810a0236)

try_to_wake_up (0xffffffff810a3ec3)

default_wake_function (0xffffffff810a4002)

autoremove_wake_function (0xffffffff810b50fd)

__wake_up_common (0xffffffff810b4958)

__wake_up (0xffffffff810b4cb8)

rb_wake_up_waiters (0xffffffff8112f126)

irq_work_run_list (0xffffffff81157d0f)

irq_work_run (0xffffffff81157d5e)

smp_trace_irq_work_interrupt (0xffffffff810082fc)

trace_irq_work_interrupt (0xffffffff816c7aaa)

return_to_handler (0xffffffff816c8458)

trace_do_page_fault (0xffffffff810478b2)

```
            trace_page_fault (0xffffffff816c7dd2)
Event: func: syscall_trace_leave() (38) Total: 26145 Avg: 688 Max: 1264 Min:381
Event: func: __sb_end_write() (1) Total: 373 Avg: 373 Max: 373 Min:373
Event: func: fsnotify() (1) Total: 598 Avg: 598 Max: 598 Min:598
Event: func: __fsnotify_parent() (1) Total: 286 Avg: 286 Max: 286 Min:286
Event: func: mutex_unlock() (2) Total: 39636 Avg: 19818 Max: 39413 Min:223
Event: func: smp_trace_irq_work_interrupt() (6) Total: 236459 Avg: 39409 Max: 100671 Min:634
        |
        + ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)
            100% (4) time:234348 max:100671 min:38745 avg:58587
             ttwu_do_wakeup (0xffffffff810a01a2)
             ttwu_do_activate.constprop.122 (0xffffffff810a0236)
             try_to_wake_up (0xffffffff810a3ec3)
             default_wake_function (0xffffffff810a4002)
             autoremove_wake_function (0xffffffff810b50fd)
             __wake_up_common (0xffffffff810b4958)
             __wake_up (0xffffffff810b4cb8)
             rb_wake_up_waiters (0xffffffff8112f126)
             irq_work_run_list (0xffffffff81157d0f)
             irq_work_run (0xffffffff81157d5e)
             smp_trace_irq_work_interrupt (0xffffffff810082fc)
             return_to_handler (0xffffffff816c8458)
             trace_irq_work_interrupt (0xffffffff816c7aaa)
              |
              + ftrace_return_to_handler (0xffffffff81140840)
              |   84% (3) time:197396 max:100671 min:38745 avg:65798
              |    return_to_handler (0xffffffff816c846d)
              |    trace_page_fault (0xffffffff816c7dd2)
              |
              + ftrace_return_to_handler (0xffffffff81140840)
                  16% (1) time:36952 max:36952 min:0 avg:36952
                   ftrace_graph_caller (0xffffffff816c8428)
                   mutex_unlock (0xffffffff816c3f75)
```

rb_simple_write (0xffffffff81133142)

vfs_write (0xffffffff811d7727)

SyS_write (0xffffffff811d7acf)

tracesys_phase2 (0xffffffff816c65b0)

Event: sys_enter:35 (1) Total: 1000599765 Avg: 1000599765 Max: 1000599765 Min:1000599765

Event: sys_enter:11 (1) Total: 55025 Avg: 55025 Max: 55025 Min:55025

Event: sys_enter:158 (1) Total: 1584 Avg: 1584 Max: 1584 Min:1584

Event: sys_enter:10 (4) Total: 18359 Avg: 4589 Max: 8764 Min:2933

Event: sys_enter:0 (1) Total: 4223 Avg: 4223 Max: 4223 Min:4223

Event: sys_enter:3 (5) Total: 9948 Avg: 1989 Max: 2606 Min:1203

Event: sys_enter:5 (3) Total: 15530 Avg: 5176 Max: 11840 Min:1405

Event: sys_enter:2 (3) Total: 28002 Avg: 9334 Max: 12035 Min:5656

Event: sys_enter:21 (1) Total: 7814 Avg: 7814 Max: 7814 Min:7814

Event: sys_enter:9 (8) Total: 49583 Avg: 6197 Max: 14137 Min:2362

Event: sys_enter:12 (4) Total: 108493 Avg: 27123 Max: 104079 Min:922

Event: sys_enter:59 (5) Total: 1631608 Avg: 326321 Max: 1607529 Min:4563

Event: page_fault_user:0x398d86b630 (1)

Event: page_fault_user:0x398d844de0 (1)

Event: page_fault_user:0x398d8d9020 (1)

Event: page_fault_user:0x1d37008 (1)

Event: page_fault_user:0x7f0b89e91074 (1)

Event: page_fault_user:0x7f0b89d98ed0 (1)

Event: page_fault_user:0x7f0b89ec8950 (1)

Event: page_fault_user:0x7f0b89d83644 (1)

Event: page_fault_user:0x7f0b89d622a8 (1)

Event: page_fault_user:0x7f0b89d5a560 (1)

Event: page_fault_user:0x7f0b89d34010 (1)

Event: page_fault_user:0x1d36008 (1)

Event: page_fault_user:0x398d900510 (1)

Event: page_fault_user:0x398dbb3ae8 (1)

Event: page_fault_user:0x398d87f490 (1)

Event: page_fault_user:0x398d8eb660 (1)

Event: page_fault_user:0x398d8bd730 (1)

Event: page_fault_user:0x398d9625d9 (1)

Event: page_fault_user:0x398d931810 (1)

Event: page_fault_user:0x398dbb7114 (1)

Event: page_fault_user:0x398d837610 (1)

Event: page_fault_user:0x398d89e860 (1)

Event: page_fault_user:0x398d8f23b0 (1)

Event: page_fault_user:0x398dbb4510 (1)

Event: page_fault_user:0x398dbad6f0 (1)

Event: page_fault_user:0x398dbb1018 (1)

Event: page_fault_user:0x398d977b37 (1)

Event: page_fault_user:0x398d92eb60 (1)

Event: page_fault_user:0x398d8abff0 (1)

Event: page_fault_user:0x398dbb0d30 (1)

Event: page_fault_user:0x398dbb6c24 (1)

Event: page_fault_user:0x398d821c50 (1)

Event: page_fault_user:0x398dbb6c20 (1)

Event: page_fault_user:0x398d886350 (1)

Event: page_fault_user:0x7f0b90125000 (1)

Event: page_fault_user:0x7f0b90124740 (1)

Event: page_fault_user:0x7f0b90126000 (1)

Event: page_fault_user:0x398d816230 (1)

Event: page_fault_user:0x398d8002b8 (1)

Event: page_fault_user:0x398dbb0b40 (1)

Event: page_fault_user:0x398dbb2880 (1)

Event: page_fault_user:0x7f0b90141cc6 (1)

Event: page_fault_user:0x7f0b9013b85c (1)

Event: page_fault_user:0x7f0b90127000 (1)

Event: page_fault_user:0x606e70 (1)

Event: page_fault_user:0x7f0b90144010 (1)

Event: page_fault_user:0x7fffcb31b038 (1)

Event: page_fault_user:0x606da8 (1)

Event: page_fault_user:0x400040 (1)

Event: page_fault_user:0x398d222218 (1)

Event: page_fault_user:0x398d015120 (1)

Event: page_fault_user:0x398d220ce8 (1)

Event: page_fault_user:0x398d220b80 (1)

Event: page_fault_user:0x7fffcb2fcff8 (1)

Event: page_fault_user:0x398d001590 (1)

Event: page_fault_user:0x398d838490 (1)

Event: softirq_raise:RCU (3) Total: 252931 Avg: 84310 Max: 243288 Min:4639

Event: softirq_raise:SCHED (2) Total: 241249 Avg: 120624 Max: 239076 Min:2173

```
      |
      + ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)
         100% (1) time:239076 max:239076 min:0 avg:239076
          ttwu_do_wakeup (0xffffffff810a01a2)
          ttwu_do_activate.constprop.122 (0xffffffff810a0236)
          try_to_wake_up (0xffffffff810a3ec3)
          default_wake_function (0xffffffff810a4002)
          autoremove_wake_function (0xffffffff810b50fd)
          __wake_up_common (0xffffffff810b4958)
          __wake_up (0xffffffff810b4cb8)
          rb_wake_up_waiters (0xffffffff8112f126)
          irq_work_run_list (0xffffffff81157d0f)
          irq_work_run (0xffffffff81157d5e)
          smp_trace_irq_work_interrupt (0xffffffff810082fc)
          trace_irq_work_interrupt (0xffffffff816c7aaa)
          irq_exit (0xffffffff8107dd66)
          smp_trace_apic_timer_interrupt (0xffffffff816c8c7a)
          trace_apic_timer_interrupt (0xffffffff816c725a)
          prepare_ftrace_return (0xffffffff8103d4fd)
          ftrace_graph_caller (0xffffffff816c8428)
          mem_cgroup_begin_page_stat (0xffffffff811cfd25)
          page_remove_rmap (0xffffffff811a4fc5)
          stub_execve (0xffffffff816c6929)
          unmap_single_vma (0xffffffff81198b1c)
          unmap_vmas (0xffffffff81199174)
```

exit_mmap (0xffffffff811a1f5b)

mmput (0xffffffff8107699a)

flush_old_exec (0xffffffff811ddb75)

load_elf_binary (0xffffffff812287df)

search_binary_handler (0xffffffff811dd3e0)

do_execveat_common.isra.31 (0xffffffff811de8bd)

do_execve (0xffffffff811dea8c)

SyS_execve (0xffffffff811ded1e)

return_to_handler (0xffffffff816c8458)

Event: softirq_raise:HI (3) Total: 72472 Avg: 24157 Max: 64186 Min:3430

Event: softirq_entry:RCU (2) Total: 3191 Avg: 1595 Max: 1788 Min:1403

|

+ ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)

100% (1) time:1788 max:1788 min:0 avg:1788

ttwu_do_wakeup (0xffffffff810a01a2)

ttwu_do_activate.constprop.122 (0xffffffff810a0236)

try_to_wake_up (0xffffffff810a3ec3)

default_wake_function (0xffffffff810a4002)

autoremove_wake_function (0xffffffff810b50fd)

__wake_up_common (0xffffffff810b4958)

__wake_up (0xffffffff810b4cb8)

rb_wake_up_waiters (0xffffffff8112f126)

irq_work_run_list (0xffffffff81157d0f)

irq_work_run (0xffffffff81157d5e)

smp_trace_irq_work_interrupt (0xffffffff810082fc)

trace_irq_work_interrupt (0xffffffff816c7aaa)

irq_work_queue (0xffffffff81157e95)

ring_buffer_unlock_commit (0xffffffff8113039f)

__buffer_unlock_commit (0xffffffff811367d5)

trace_buffer_unlock_commit (0xffffffff811376a2)

ftrace_event_buffer_commit (0xffffffff81146d5f)

ftrace_raw_event_sched_process_exec (0xffffffff8109c511)

do_execveat_common.isra.31 (0xffffffff811de9a3)

```
            do_execve (0xffffffff811dea8c)
            SyS_execve (0xffffffff811ded1e)
            return_to_handler (0xffffffff816c8458)
            stub_execve (0xffffffff816c6929)
Event: softirq_entry:SCHED (2) Total: 2289 Avg: 1144 Max: 1350 Min:939
Event: softirq_entry:HI (3) Total: 180146 Avg: 60048 Max: 178969 Min:499
          |
          + ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)
             100% (1) time:178969 max:178969 min:0 avg:178969
              ttwu_do_wakeup (0xffffffff810a01a2)
              ttwu_do_activate.constprop.122 (0xffffffff810a0236)
              try_to_wake_up (0xffffffff810a3ec3)
              wake_up_process (0xffffffff810a4057)
              wake_up_worker (0xffffffff8108de74)
              insert_work (0xffffffff8108fca6)
              __queue_work (0xffffffff8108fe12)
              delayed_work_timer_fn (0xffffffff81090088)
              call_timer_fn (0xffffffff810d8f89)
              run_timer_softirq (0xffffffff810da8a1)
              __do_softirq (0xffffffff8107d8fa)
              irq_exit (0xffffffff8107dd66)
              smp_trace_apic_timer_interrupt (0xffffffff816c8c7a)
              trace_apic_timer_interrupt (0xffffffff816c725a)
              prepare_ftrace_return (0xffffffff8103d4fd)
              ftrace_graph_caller (0xffffffff816c8428)
              mem_cgroup_begin_page_stat (0xffffffff811cfd25)
              page_remove_rmap (0xffffffff811a4fc5)
              stub_execve (0xffffffff816c6929)
              unmap_single_vma (0xffffffff81198b1c)
              unmap_vmas (0xffffffff81199174)
              exit_mmap (0xffffffff811a1f5b)
              mmput (0xffffffff8107699a)
              flush_old_exec (0xffffffff811ddb75)
```

```
            load_elf_binary (0xffffffff812287df)
            search_binary_handler (0xffffffff811dd3e0)
            do_execveat_common.isra.31 (0xffffffff811de8bd)
            do_execve (0xffffffff811dea8c)
            SyS_execve (0xffffffff811ded1e)
            return_to_handler (0xffffffff816c8458)
   ---
```

The above uses -F to follow the sleep task. It filters only on events that pertain to sleep.

Note, in order to follow forks, you need to also include the -c flag.

Other tasks will appear in the profile as well if events reference more than one task (like sched_switch and sched_wakeup do. The "prev_pid" and "next_pid" of sched_switch, and the "common_pid" and "pid" of sched_wakeup).

Stack traces are attached to events that are related to them.

Taking a look at the above output:

```
   Event: sched_switch:R (2) Total: 234559 Avg: 117279 Max: 129886 Min:104673
```

This shows that task was preempted (it?s in the running R state). It was preempted twice (2) for a total of 234,559 nanoseconds, with a average preempt time of 117,279 ns, and maximum of 128,886 ns and minimum of 104,673 ns.

The tree shows where it was preempted:

```
   |
   + ftrace_raw_event_sched_switch (0xffffffff8109f310)
       100% (2) time:234559 max:129886 min:104673 avg:117279
       __schedule (0xffffffff816c1e81)
       preempt_schedule (0xffffffff816c236e)
       ___preempt_schedule (0xffffffff81351a59)
        |
        + unmap_single_vma (0xffffffff81198c05)
        |   55% (1) time:129886 max:129886 min:0 avg:129886
        |    stop_one_cpu (0xffffffff8110909a)
        |    sched_exec (0xffffffff810a119b)
        |    do_execveat_common.isra.31 (0xffffffff811de528)
        |    do_execve (0xffffffff811dea8c)
        |    SyS_execve (0xffffffff811ded1e)
```

```
        |    return_to_handler (0xffffffff816c8458)
        |    stub_execve (0xffffffff816c6929)
        |
        + unmap_single_vma (0xffffffff81198c05)
           45% (1) time:104673 max:104673 min:0 avg:104673
            unmap_vmas (0xffffffff81199174)
            exit_mmap (0xffffffff811a1f5b)
            mmput (0xffffffff8107699a)
            flush_old_exec (0xffffffff811ddb75)
            load_elf_binary (0xffffffff812287df)
            search_binary_handler (0xffffffff811dd3e0)
            do_execveat_common.isra.31 (0xffffffff811de8bd)
            do_execve (0xffffffff811dea8c)
            SyS_execve (0xffffffff811ded1e)
            return_to_handler (0xffffffff816c8458)
            stub_execve (0xffffffff816c6929)
    Event: sched_switch:S (1) Total: 1000513242 Avg: 1000513242 Max: 1000513242 Min:10005132
```

This shows that the task was scheduled out in the INTERRUPTIBLE state once for a total of 1,000,513,242 ns (~1s), which makes sense as the task was a "sleep 1".

After the schedule events, the function events are shown. By default the profiler will use the function graph tracer if the depth setting is supported by the kernel. It will set the depth to one which will only trace the first function that enters the kernel. It will also record the amount of time it was in the kernel.

```
    Event: func: sys_nanosleep() (1) Total: 1000598016 Avg: 1000598016 Max: 1000598016 Min:1000598016
    Event: func: sys_munmap() (1) Total: 14300 Avg: 14300 Max: 14300 Min:14300
    Event: func: sys_arch_prctl() (1) Total: 571 Avg: 571 Max: 571 Min:571
    Event: func: sys_mprotect() (4) Total: 14382 Avg: 3595 Max: 7196 Min:2190
    Event: func: SyS_read() (1) Total: 2640 Avg: 2640 Max: 2640 Min:2640
    Event: func: sys_close() (5) Total: 4001 Avg: 800 Max: 1252 Min:414
    Event: func: sys_newfstat() (3) Total: 11684 Avg: 3894 Max: 10206 Min:636
    Event: func: SyS_open() (3) Total: 23615 Avg: 7871 Max: 10535 Min:4743
    Event: func: sys_access() (1) Total: 5924 Avg: 5924 Max: 5924 Min:5924
    Event: func: SyS_mmap() (8) Total: 39153 Avg: 4894 Max: 12354 Min:1518
```

Event: func: smp_trace_apic_timer_interrupt() (1) Total: 10298 Avg: 10298 Max: 10298 Min:10298

Event: func: SyS_brk() (4) Total: 2407 Avg: 601 Max: 1564 Min:206

Event: func: do_notify_resume() (2) Total: 4095 Avg: 2047 Max: 2521 Min:1574

Event: func: sys_execve() (5) Total: 1625251 Avg: 325050 Max: 1605698 Min:3570

Count of times the event was hit is always in parenthesis (5).

The function graph trace may produce too much overhead as it is still triggering (just not tracing) on all functions. To limit functions just to system calls (not interrupts), add the following option:

-l 'sys_*' -l 'SyS_*'

To disable function graph tracing totally, use:

-p nop

To use function tracing instead (note, this will not record timings, but just the count of times a function is hit):

-p function

Following the functions are the events that are recorded.

Event: sys_enter:35 (1) Total: 1000599765 Avg: 1000599765 Max: 1000599765 Min:1000599765

Event: sys_enter:11 (1) Total: 55025 Avg: 55025 Max: 55025 Min:55025

Event: sys_enter:158 (1) Total: 1584 Avg: 1584 Max: 1584 Min:1584

Event: sys_enter:10 (4) Total: 18359 Avg: 4589 Max: 8764 Min:2933

Event: sys_enter:0 (1) Total: 4223 Avg: 4223 Max: 4223 Min:4223

Event: sys_enter:3 (5) Total: 9948 Avg: 1989 Max: 2606 Min:1203

Event: sys_enter:5 (3) Total: 15530 Avg: 5176 Max: 11840 Min:1405

Event: sys_enter:2 (3) Total: 28002 Avg: 9334 Max: 12035 Min:5656

Event: sys_enter:21 (1) Total: 7814 Avg: 7814 Max: 7814 Min:7814

Event: sys_enter:9 (8) Total: 49583 Avg: 6197 Max: 14137 Min:2362

Event: sys_enter:12 (4) Total: 108493 Avg: 27123 Max: 104079 Min:922

Event: sys_enter:59 (5) Total: 1631608 Avg: 326321 Max: 1607529 Min:4563

These are the raw system call events, with the raw system call ID after the "sys_enter:" For example, "59" is execve(2). Why did it execute 5 times? Looking at a strace of this run, we can see:

execve("/usr/lib64/ccache/sleep", ["sleep", "1"], [/* 27 vars */] <unfinished ...>

<... execve resumed>)   = -1 ENOENT (No such file or directory)

execve("/usr/local/sbin/sleep", ["sleep", "1"], [/* 27 vars */] <unfinished ...>

```
<... execve resumed> )      = -1 ENOENT (No such file or directory)
execve("/usr/local/bin/sleep", ["sleep", "1"], [/* 27 vars */] <unfinished ...>
<... execve resumed> )      = -1 ENOENT (No such file or directory)
execve("/usr/sbin/sleep", ["sleep", "1"], [/* 27 vars */] <unfinished ...>
<... execve resumed> )      = -1 ENOENT (No such file or directory)
execve("/usr/bin/sleep", ["sleep", "1"], [/* 27 vars */] <unfinished ...>
<... execve resumed> )      = 0
```

It attempted to execve the "sleep" command for each path in $PATH until it found one.

The page_fault_user events show what userspace address took a page fault.

```
Event: softirq_raise:RCU (3) Total: 252931 Avg: 84310 Max: 243288 Min:4639
Event: softirq_raise:SCHED (2) Total: 241249 Avg: 120624 Max: 239076 Min:2173
        |
        + ftrace_raw_event_sched_wakeup_template (0xffffffff8109d960)
           100% (1) time:239076 max:239076 min:0 avg:239076
            ttwu_do_wakeup (0xffffffff810a01a2)
            ttwu_do_activate.constprop.122 (0xffffffff810a0236)
            try_to_wake_up (0xffffffff810a3ec3)
            default_wake_function (0xffffffff810a4002)
            autoremove_wake_function (0xffffffff810b50fd)
            __wake_up_common (0xffffffff810b4958)
            __wake_up (0xffffffff810b4cb8)
            rb_wake_up_waiters (0xffffffff8112f126)
            irq_work_run_list (0xffffffff81157d0f)
            irq_work_run (0xffffffff81157d5e)
            smp_trace_irq_work_interrupt (0xffffffff810082fc)
            trace_irq_work_interrupt (0xffffffff816c7aaa)
            irq_exit (0xffffffff8107dd66)
```

The timings for the softirq_raise events measure the time it took from the raised softirq to the time it executed.

The timings for the softirq_entry events measure the time the softirq took to execute.

The stack traces for the softirqs (and possibly other events) are used when an event has a stack attached to it. This can happen if the profile ran more stacks than just the sched events, or when events are dropped and stacks

To have full control of what gets traced, use the -S option that will have trace-cmd not enable any events or the function_graph tracer. Only the events listed on the command line are shown.

If only the time of kmalloc is needed to be seen, and where it was recorded, using the -S option and enabling function_graph and stack tracing for just the function needed will give the profile of only that function.

```
 ---
# trace-cmd profile -S -p function_graph -l '*kmalloc*' -l '*kmalloc*:stacktrace' sleep 1
task: sshd-11786
  Event: func: __kmalloc_reserve.isra.59() (2) Total: 149684 Avg: 74842 Max: 75598 Min:74086
         |
         + __alloc_skb (0xffffffff815a8917)
         |   67% (2) time:149684 max:75598 min:74086 avg:74842
         |    __kmalloc_node_track_caller (0xffffffff811c6635)
         |    __kmalloc_reserve.isra.59 (0xffffffff815a84ac)
         |    return_to_handler (0xffffffff816c8458)
         |    sk_stream_alloc_skb (0xffffffff81604ea1)
         |    tcp_sendmsg (0xffffffff8160592c)
         |    inet_sendmsg (0xffffffff8162fed1)
         |    sock_aio_write (0xffffffff8159f9fc)
         |    do_sync_write (0xffffffff811d694a)
         |    vfs_write (0xffffffff811d7825)
         |    SyS_write (0xffffffff811d7adf)
         |    system_call_fastpath (0xffffffff816c63d2)
         |
         + __alloc_skb (0xffffffff815a8917)
            33% (1) time:74086 max:74086 min:74086 avg:74086
             __alloc_skb (0xffffffff815a8917)
             sk_stream_alloc_skb (0xffffffff81604ea1)
             tcp_sendmsg (0xffffffff8160592c)
             inet_sendmsg (0xffffffff8162fed1)
             sock_aio_write (0xffffffff8159f9fc)
             do_sync_write (0xffffffff811d694a)
```

vfs_write (0xffffffff811d7825)

SyS_write (0xffffffff811d7adf)

system_call_fastpath (0xffffffff816c63d2)

[..]

To watch the command run but save the output of the profile to a file use --stderr, and redirect stderr to a file

trace-cmd profile --stderr cyclictest -p 80 -n -t1 2> profile.out

Or simple use -o

trace-cmd profile -o profile.out cyclictest -p 80 -n -t1

SEE ALSO

trace-cmd(1), trace-cmd-record(1), trace-cmd-report(1), trace-cmd-start(1), trace-cmd-stop(1), trace-cmd-reset(1), trace-cmd-split(1), trace-cmd-list(1), trace-cmd-listen(1)

AUTHOR

Written by Steven Rostedt, <rostedt@goodmis.org[1]>

RESOURCES

https://git.kernel.org/pub/scm/utils/trace-cmd/trace-cmd.git/

COPYING

NOTES

1. rostedt@goodmis.org

   mailto:rostedt@goodmis.org