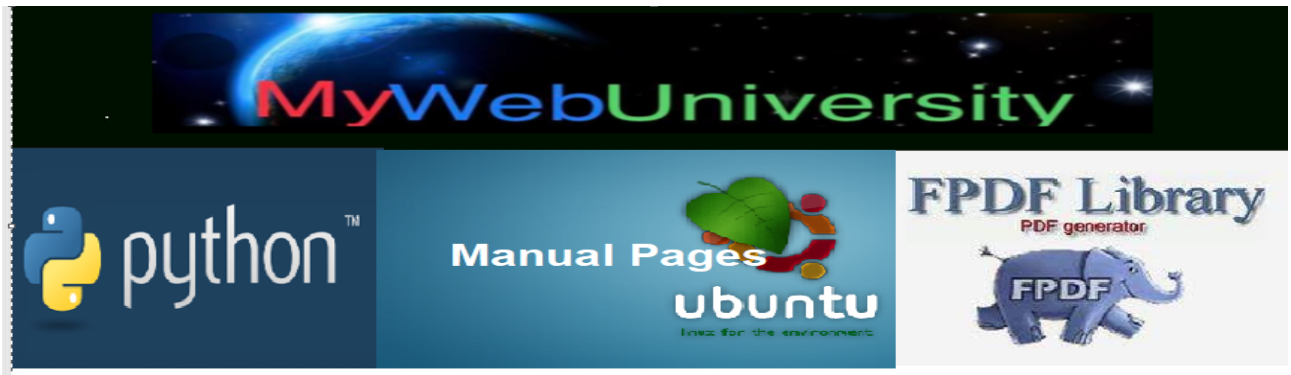

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man trace-cmd-report.1

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

NAME

trace-cmd-report - show in ASCII a trace created by trace-cmd record

SYNOPSIS

trace-cmd report [OPTIONS] [input-file [input-file ...]]

DESCRIPTION

The trace-cmd(1) report command will output a human readable report of a trace created by trace-cmd record.

OPTIONS

-i input-file

By default, trace-cmd report will read the file trace.dat. But the -i option open up the given input-file instead. Note, the input file may also be specified as the last item on the command line.

-e

This outputs the endianess of the file. trace-cmd report is smart enough to be able to read big endian files on little endian machines, and vise versa.

-f

This outputs the list of all functions that have been mapped in the trace.dat file. Note, this list may contain functions that may not appear in the trace, as it is the list of mappings to translate function addresses into function names.

-P

This outputs the list of "trace_printk()" data. The raw trace data points to static pointers in the kernel. This must be stored in the trace.dat file.

-E

This lists the possible events in the file (but this list is not necessarily the list of events in the file).

--events

This will list the event formats that are stored in the trace.dat file.

--event regex

This will print events that match the given regex. If a colon is specified, then the characters before the colon will be used to match the system and the characters after the colon will match the event.

trace-cmd report --event sys:read

The above will only match events where the system name contains "sys" and the event name contains "read".

trace-cmd report --event read

The above will match all events that contain "read" in its name. Also it may list all events of a system that contains "read" as well.

--check-events

This will parse the event format strings that are stored in the trace.dat file and return whether the formats can be parsed correctly. It will load plugins unless -N is specified.

-t

Print the full timestamp. The timestamps in the data file are usually recorded to the nanosecond. But the default display of the timestamp is only to the microsecond. To see the full timestamp, add the -t option.

-F filter

Add a filter to limit what events are displayed. Filters defined after an input file (specified with -i) only apply to that input file. Filters provided before any input file is given are considered global and apply to all input files.

The format of the filter is:

<events> ':' <filter>

<events> = SYSTEM'/'EVENT | SYSTEM | EVENT | <events> ',' <events>

<filter> = EVENT_FIELD <op> <value> | <filter> '&&' <filter> |

<filter> '||' <filter> | '(' <filter> ')' | '!' <filter>

<op> = '==' | '!=' | '>=' | '<=' | '>' | '<' | '&' | '|' | '^' |

'+' | '-' | '*' | '/' | '%'

<value> = NUM | STRING | EVENT_FIELD

SYSTEM is the name of the system to filter on. If the EVENT is left out, then it applies to all events under the SYSTEM. If only one string is used without the '/' to deliminate between SYSTEM and EVENT, then the filter will be applied to all systems and events that match the given string. Whitespace is ignored, such that "sched:next_pid==123" is equivalent to "sched : next_pid == 123".

STRING is defined with single or double quotes (single quote must end with single quote, and double with double). Whitespace within quotes are not ignored.

The representation of a SYSTEM or EVENT may also be a regular expression as defined by 'regcomp(3)'.

The EVENT_FIELD is the name of the field of an event that is being filtered. If the event does not contain the EVENT_FIELD, that part of the equation will be considered false.

-F 'sched : bogus == 1 || common_pid == 2'

The "bogus == 1" will always evaluate to FALSE because no event has a field called "bogus", but the "common_pid == 2" will still be evaluated since all events have the field "common_pid". Any "sched" event that was traced by the process with the PID of 2 will be shown.

Note, the EVENT_FIELD is the field name as shown by an events format (as displayed with *--events*), and not what is found in the output.

If the output shows "ID:foo" but the field that "foo" belongs to was called "name" in the event format, then "name" must be used in the filter.

The same is true about values. If the value that is displayed is converted by to a string symbol, the filter checks the original value and not the value displayed. For example, to filter on all tasks that were in the running state at a context switch:

-F 'sched/sched_switch : prev_state==0'

Although the output displays 'R', having 'prev_stat=="R"' will not work.

Note: You can also specify 'COMM' as an EVENT_FIELD. This will use the task name (or comm) of the record to compare. For example, to filter out

all of the "trace-cmd" tasks:

-F '.*:COMM != "trace-cmd"'

-I

Do not print events where the HARDIRQ latency flag is set. This will filter out most events that are from interrupt context. Note, it may not filter out function traced functions that are in interrupt context but were called before the kernel "in interrupt" flag was set.

-S

Do not print events where the SOFTIRQ latency flag is set. This will filter out most events that are from soft interrupt context.

-v

This causes the following filters of -F to filter out the matching events.

-v -F 'sched/sched_switch : prev_state == 0'

Will not display any sched_switch events that have a prev_state of 0.

Removing the *-v* will only print out those events.

-T

Test the filters of -F. After processing a filter string, the resulting filter will be displayed for each event. This is useful for using a filter for more than one event where a field may not exist in all events. Also it can be used to make sure there are no misspelled event field names, as they will simply be ignored. -T is ignored if -F is not specified.

-V

Show verbose messages (see --verbose but only for the numbers)

-L

This will not load system wide plugins. It loads "local only". That is what it finds in the ~/.trace-cmd/plugins directory.

-N

This will not load any plugins.

-n event-re

This will cause all events that match the option to ignore any registered handler (by the plugins) to print the event. The normal event will be printed instead. The event-re is a regular expression as defined by regcomp(3).

--profile

With the --profile option, "trace-cmd report" will process all the events first, and then output a format showing where tasks have spent their time in the kernel, as well as where they are blocked the most, and where wake up latencies are.

See trace-cmd-profile(1) for more details and examples.

-G

Set interrupt (soft and hard) events as global (associated to CPU instead of tasks). Only works for --profile.

-H event-hooks

Add custom event matching to connect any two events together.

See trace-cmd-profile(1) for format.

-R

This will show the events in "raw" format. That is, it will ignore the event?s print formatting and just print the contents of each field.

-r event-re

This will cause all events that match the option to print its raw fields. The event-re is a regular expression as defined by regcomp(3).

-l

This adds a "latency output" format. Information about interrupts being disabled, soft irq being disabled, the "need_resched" flag being set, preempt count, and big kernel lock are all being recorded with every event. But the default display does not show this information. This option will set display this information with 6 characters. When one of the fields is zero or N/A a '.\' is shown.

<idle>-0 0d.h1. 106467.859747: function: ktime_get <-- tick_check_idle

The 0d.h1. denotes this information.

It starts with a number. This represents the CPU number that the event occurred on.

The second character is one of the following:

'd' - Interrupts are disabled

'.' - Interrupts are enabled

'X' - Has flags that are not yet known by trace-cmd

The third character is the "need rescheduling" flag.

'N' - A schedule is set to take place

'.' - No scheduling is set

The fourth character represents the context the event was in when it triggered

'h' - Hard interrupt context

's' - Soft interrupt context

'H' - Hard interrupt context that interrupted a soft interrupt

'.' - Normal context

The next is a number (should be less than 10), that represents the preemption depth (the number of times preempt_disable() is called without preempt_enable()).

'.' means preemption is enabled.

On some systems, "migrate disable" may exist, in which case a number will be shown for that, or '.' meaning migration is enabled.

If lockdep in enabled on the system, then the number represents the depth of locks that are held when the event triggered. '.' means no locks are held.

-w

If both the sched_switch and sched_wakeup events are enabled, then this option will report the latency between the time the task was first woken, and the time it was scheduled in.

-q

Quiet non critical warnings.

-O

Pass options to the trace-cmd plugins that are loaded.

-O plugin:var=value

The 'plugin:' and '=value' are optional. Value may be left off for options that are boolean. If the 'plugin:' is left off, then any variable that matches in all plugins will be set.

Example: -O fgraph:tailprint

--cpu <cpu list>

List of CPUs, separated by "," or ":", used for filtering the events. A range of CPUs can be specified using "cpuX-cpuY" notation, where all CPUs in the range between cpuX and cpuY will be included in the list. The order of CPUs in the list must be from lower to greater.

Example: "--cpu 0,3" - show events from CPUs 0 and 3

"--cpu 2-4" - show events from CPUs 2, 3 and 4

--cpus

List the CPUs that have data in the trace file then exit.

--first-event

Show the timestamp of the first event of all CPUs that have data.

--last-event

Show the timestamp of the last event of all CPUs that have data.

--stat

If the trace.dat file recorded the final stats (outputed at the end of record) the

--stat option can be used to retrieve them.

--uname

If the trace.dat file recorded uname during the run, this will retrieve that

information.

--version

If the trace.dat file recorded the version of the executable used to create it, report

that version.

--ts-offset offset

Add (or subtract if negative) an offset for all timestamps of the previous data file

specified with -i. This is useful to merge sort multiple trace.dat files where the

difference in the timestamp is known. For example if a trace is done on a virtual guest,

and another trace is done on the host. If the host timestamp is 1000 units ahead of the

guest, the following can be done:

trace-cmd report -i host.dat --ts-offset -1000 -i guest.dat

This will subtract 1000 timestamp units from all the host events as it merges

with the guest.dat events. Note, the units is for the raw units recorded in

the trace. If the units are nanoseconds, the addition (or subtraction) from

the offset will be nanoseconds even if the displayed units are microseconds.

--ts2secs HZ

Convert the current clock source into a second (nanosecond resolution) output. When

using clocks like x86-tsc, if the frequency is known, by passing in the clock frequency,

this will convert the time to seconds.

This option affects any trace.dat file given with *-i* proceeding it.

If this option comes before any *-i* option, then that value becomes

the default conversion for all other trace.dat files. If another

--ts2secs option appears after a *-i* trace.dat file, than that option

will override the default value.

Example: On a 3.4 GHz machine

trace-cmd record -p function -C x86-tsc

trace-cmd report --ts2ns 3400000000

The report will convert the cycles timestamps into a readable second display. The default display resolution is microseconds, unless *-t* is used.

The value of --ts-offset must still be in the raw timestamp units, even with this option. The offset will be converted as well.

--ts-diff

Show the time differences between events. The difference will appear in parenthesis just after the timestamp.

--ts-check

Make sure no timestamp goes backwards, and if it does, print out a warning message of the fact.

--nodate

Ignore converting the timestamps to the date set by trace-cmd record(3) --date option.

--raw-ts

Display raw timestamps, without any corrections.

--align-ts

Display timestamps aligned to the first event.

--verbose[=level]

Set the log level. Supported log levels are "none", "crit", "err", "warn", "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 report --verbose=warning

EXAMPLES

Using a trace.dat file that was created with:

trace-cmd record -p function -e all sleep 5

The default report shows:

trace-cmd report

```
cpus=8
          sleep-89142 [001] ...1. 162573.215752: function:             mutex_unlock
          sleep-89142 [001] ...1. 162573.215754: function:             __mutex_unlock_slowpath
          sleep-89142 [001] ..... 162573.215755: lock_release:         0xffffffff855e7448 trace_types_lock
          sleep-89142 [001] ..... 162573.215757: lock_release:         0xffff892a01b54420 sb_writers
          sleep-89142 [001] ...1. 162573.215757: function:             preempt_count_add
          sleep-89142 [001] ...1. 162573.215758: preempt_disable:      caller=vfs_write+0x147 parent=vfs_write+0x147
          sleep-89142 [001] ...2. 162573.215758: function:             rcu_read_lock_any_held
          sleep-89142 [001] ...2. 162573.215759: function:               rcu_lockdep_current_cpu_online
          sleep-89142 [001] ...2. 162573.215759: function:             preempt_count_sub
          sleep-89142 [001] ...1. 162573.215760: preempt_enable:       caller=vfs_write+0x176 parent=vfs_write+0x176
          sleep-89142 [001] ...1. 162573.215761: function:             __f_unlock_pos
          sleep-89142 [001] ...1. 162573.215761: function:             mutex_unlock
[...]
```

The note on the third column:

```
          sleep-89998 [002] ...1. 223087.004011: lock_acquire:         0xffff892b7cf32c20 lock
          sleep-89998 [002] ...1. 223087.004011: lock_acquire:         0xffffffff85517f00 read rcu_read_lock
          <idle>-0     [005] dNh2. 223087.004012: sched_wakeup:         trace-cmd:89992 [120] CPU:005
          sleep-89998 [002] ...1. 223087.004012: lock_acquire:         0xffffffff85517f00 read rcu_read_lock
          sleep-89998 [002] ...1. 223087.004013: lock_release:         0xffffffff85517f00 rcu_read_lock
```

It follows the same as shown in the Linux kernel /sys/kernel/tracing/trace file.

```
# cat /sys/kernel/tracing/trace
# tracer: nop
#
# entries-in-buffer/entries-written: 0/0   #P:8
#
#                              _-----=> irqs-off/BH-disabled
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| / _-=> migrate-disable
#                            |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
```

```
#              | |         |   |||||     |         |
```

Is the same as explained in the -l option. Where the first position is:

'.' - means interrupts and bottom halves enabled

'd' - means interrupts and bottom halves are disabled

The second position:

'N' - means that the "NEED_RESCHED" flag is set and the kernel should try to

schedule as soon as possible.

The third position:

'.' - In normal/schedulable context

's' - In soft interrupt context

'h' - In hard interrupt context

'H' - in hard interrupt context that interrupted a soft interrupt

The forth position is the preempt count depth:

'pass:[.]' - preemption is enabled

'#' - the depth of preemption disabled (nested)

The fifth column is the migration disabled counter:

'.' - migration is enabled

'#' - the depth of migration being disabled (nested)

To see everything but the function traces:

```
 # trace-cmd report -v -F 'function'
cpus=8
          sleep-89142 [001] ..... 162573.215755: lock_release:         0xffffffff855e7448 trace_types_lock
          sleep-89142 [001] ..... 162573.215757: lock_release:         0xffff892a01b54420 sb_writers
          sleep-89142 [001] ...1. 162573.215758: preempt_disable:      caller=vfs_write+0x147 parent=vfs_write+0x147
          sleep-89142 [001] ...1. 162573.215760: preempt_enable:       caller=vfs_write+0x176 parent=vfs_write+0x176
          sleep-89142 [001] ..... 162573.215762: lock_release:         0xffff892a19601ac8 &f->f_pos_lock
          sleep-89142 [001] ..... 162573.215764: sys_exit:             NR 1 = 1
          sleep-89142 [001] ..... 162573.215766: sys_exit_write:       0x1
          sleep-89142 [001] d.... 162573.215767: irq_disable:          caller=syscall_exit_to_user_mode+0x15 parent=0x0
          sleep-89142 [001] d.... 162573.215768: irq_enable:           caller=syscall_exit_to_user_mode+0xed parent=0x0
          sleep-89142 [001] ..... 162573.215773: lock_acquire:         0xffff892a4ad29318 read &mm->mmap_lock
          sleep-89142 [001] ..... 162573.215775: lock_release:         0xffff892a4ad29318 &mm->mmap_lock
          sleep-89142 [001] ..... 162573.215778: lock_acquire:         0xffff892a4ad29318 read &mm->mmap_lock
```

[...]

To see only the kmalloc calls that were greater than 1000 bytes:

```
 # trace-cmd report -F 'kmalloc: bytes_req > 1000'
cpus=8
                sleep-89142  [001] .....  162573.219401:  kmalloc:                (tomoyo_find_next_domain+0x84)
call_site=tomoyo_find_next_domain+0x84  ptr=0xffff892a176c0000  bytes_req=4096  bytes_alloc=4096  gfp_flags=0xd40
node=-1 accounted=false
                sleep-89142  [001] .....  162573.219511:  kmalloc:                (tomoyo_realpath_from_path+0x42)
call_site=tomoyo_realpath_from_path+0x42  ptr=0xffff892a176c6000  bytes_req=4096  bytes_alloc=4096  gfp_flags=0xc40
node=-1 accounted=false
           trace-cmd-89135  [000] .....  162573.244301:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a63f84000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89135  [000] .....  162573.244471:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a63f84000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89134  [007] .....  162573.267148:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a628d4000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89134  [007] .....  162573.267403:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a628d4000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89141  [002] .....  162573.290583:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a12d3c000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89141  [002] .....  162573.290754:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a12d3c000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
           trace-cmd-89139  [004] .....  162573.784636:  kmalloc:                (kvmalloc_node_noprof+0x43)
call_site=kvmalloc_node_noprof+0x43  ptr=0xffff892a63d70000  bytes_req=8193  bytes_alloc=16384  gfp_flags=0x12dc0
node=-1 accounted=false
```

[...]

To see wakeups and sched switches that left the previous task in the running state:

```
 # trace-cmd report -F 'sched: prev_state == 0' -F 'sched_waking'
cpus=8
            sleep-89142 [001] d.h6. 162573.215941: sched_waking:         comm=trace-cmd pid=89135 prio=120
target_cpu=000
           <idle>-0     [000] dNh7. 162573.216219: sched_waking:         comm=trace-cmd pid=89134 prio=120
target_cpu=007
      <idle>-0     [000] d..2. 162573.216423: sched_switch:       swapper/0:0 [120] R ==> trace-cmd:89135 [120]
           <idle>-0     [007] dNh7. 162573.216511: sched_waking:         comm=trace-cmd pid=89141 prio=120
target_cpu=002
      <idle>-0     [007] d..2. 162573.216698: sched_switch:       swapper/7:0 [120] R ==> trace-cmd:89134 [120]
           <idle>-0     [002] dNh7. 162573.216776: sched_waking:         comm=trace-cmd pid=89136 prio=120
target_cpu=001
      <idle>-0     [002] d..2. 162573.216951: sched_switch:       swapper/2:0 [120] R ==> trace-cmd:89141 [120]
            sleep-89142 [001] d.s3. 162573.231260: sched_waking:         comm=rcu_preempt pid=17 prio=120
target_cpu=002
      <idle>-0     [002] d..2. 162573.231568: sched_switch:       swapper/2:0 [120] R ==> rcu_preempt:17 [120]
            sleep-89142 [001] d.s2. 162573.240425: sched_waking:         comm=rcu_preempt pid=17 prio=120
target_cpu=002
      <idle>-0     [002] d..2. 162573.240719: sched_switch:       swapper/2:0 [120] R ==> rcu_preempt:17 [120]
           sleep-89142 [001] d.h7. 162573.241983: sched_waking:         comm=trace-cmd pid=89135 prio=120
target_cpu=000
```

The above needs a little explanation. The filter specifies the "sched" subsystem, which includes all scheduling events. Any event that does not have the format field "prev_state", will evaluate those expressions as FALSE, and will not produce a match. Only the sched_switch event will match that. The second "-F" will include the sched_waking event.

```
  # trace-cmd report -w -F 'sched_switch, sched_wakeup.*'
 [...]
      trace-cmd-89141 [007] d..2. 162583.263060: sched_switch:       trace-cmd:89141 [120] R ==> trace-cmd:89135
[120]
        kworker/u36:1-51219 [000] d..2. 162583.266957: sched_switch:         kworker/u36:1:51219 [120] R ==>
kworker/u33:2:49692 [120] Latency: 4024.977 usecs
      trace-cmd-89135 [007] d..2. 162583.267109: sched_switch:       trace-cmd:89135 [120] R ==> trace-cmd:89141
[120]
```

trace-cmd-89139 [001] d..2. 162583.267147: sched_switch: trace-cmd:89139 [120] D ==> swapper/1:0 [120]

kworker/u36:2-88857 [002] d..2. 162583.267913: sched_switch: kworker/u36:2:88857 [120] R ==> trace-cmd:89136 [120]

kworker/u33:2-49692 [000] d..2. 162583.268334: sched_switch: kworker/u33:2:49692 [120] I ==> kworker/u36:1:51219 [120]

<idle>-0 [001] dNh4. 162583.268747: sched_wakeup: sleep:89142 [120] CPU:001

<idle>-0 [001] d..2. 162583.268833: sched_switch: swapper/1:0 [120] R ==> sleep:89142 [120] Latency: 85.751 usecs

sleep-89142 [001] d.h4. 162583.269022: sched_wakeup: trace-cmd:89139 [120] CPU:001

trace-cmd-89141 [007] d..2. 162583.271009: sched_switch: trace-cmd:89141 [120] R ==> trace-cmd:89135 [120]

trace-cmd-89136 [002] d..2. 162583.271020: sched_switch: trace-cmd:89136 [120] R ==> kworker/u36:2:88857 [120]

kworker/u36:2-88857 [002] d..2. 162583.271079: sched_switch: kworker/u36:2:88857 [120] I ==> trace-cmd:89136 [120]

trace-cmd-89137 [006] d.h2. 162583.273950: sched_wakeup: trace-cmd:89133 [120] CPU:006

sleep-89142 [001] d..2. 162583.274064: sched_switch: sleep:89142 [120] Z ==> trace-cmd:89139 [120] Latency: 5042.285 usecs

trace-cmd-89135 [007] d..2. 162583.275043: sched_switch: trace-cmd:89135 [120] R ==> trace-cmd:89141 [120]

trace-cmd-89137 [006] d..2. 162583.275158: sched_switch: trace-cmd:89137 [120] R ==> trace-cmd:89133 [120] Latency: 1207.327 usecs

trace-cmd-89136 [002] dNh3. 162583.275229: sched_wakeup: rcu_preempt:17 [120] CPU:002

trace-cmd-89136 [002] d..2. 162583.275294: sched_switch: trace-cmd:89136 [120] R ==> rcu_preempt:17 [120] Latency: 65.255 usecs

rcu_preempt-17 [002] d..2. 162583.275399: sched_switch: rcu_preempt:17 [120] I ==> trace-cmd:89136 [120]

Average wakeup latency: 20082.580 usecs

Maximum Latency: 1032049.277 usecs at timestamp: 162574.787022

Minimum Latency: 29.023 usecs at timestamp: 162583.189731

RT task timings:

Average wakeup latency: 139.568 usecs

Maximum Latency: 220.583 usecs at timestamp: 162577.347038

Minimum Latency: 75.902 usecs at timestamp: 162577.719121

The above trace produces the wakeup latencies of the tasks. The "sched_switch" event reports each individual latency after writing the event information. At the end of the report, the average wakeup latency is reported, as well as the maxim and minimum latency and the timestamp they happened at. It does this for both normal tasks as well as real-time tasks.

```
# trace-cmd report -w -F 'sched_switch, sched_wakeup.*: prio < 100 || next_prio < 100'
cpus=8
          <idle>-0     [001] dNh5. 162573.291142: sched_wakeup:         migration/1:23 [0] CPU:001
          <idle>-0     [001] d..2. 162573.291237: sched_switch:         swapper/1:0 [120] R ==> migration/1:23 [0] Latency:
94.643 usecs
      trace-cmd-89141 [002] dNh6. 162573.346785: sched_wakeup:         migration/2:28 [0] CPU:002
      trace-cmd-89141 [002] d..2. 162573.346929: sched_switch:         trace-cmd:89141 [120] R ==> migration/2:28 [0]
Latency: 143.971 usecs
      trace-cmd-89134 [003] dNh4. 162573.410852: sched_wakeup:         migration/3:33 [0] CPU:003
      trace-cmd-89134 [003] d..2. 162573.411039: sched_switch:         trace-cmd:89134 [120] R ==> migration/3:33 [0]
Latency: 187.640 usecs
          <idle>-0     [004] dNh5. 162573.490944: sched_wakeup:         migration/4:38 [0] CPU:004
          <idle>-0     [004] d..2. 162573.491098: sched_switch:         swapper/4:0 [120] R ==> migration/4:38 [0] Latency:
153.913 usecs
          <idle>-0     [005] dNh5. 162573.574955: sched_wakeup:         migration/5:43 [0] CPU:005
          <idle>-0     [005] d..2. 162573.575107: sched_switch:         swapper/5:0 [120] R ==> migration/5:43 [0] Latency:
152.875 usecs
          <idle>-0     [006] dNh5. 162573.646878: sched_wakeup:         migration/6:48 [0] CPU:006
          <idle>-0     [006] d..2. 162573.646992: sched_switch:         swapper/6:0 [120] R ==> migration/6:48 [0] Latency:
113.788 usecs
      trace-cmd-89140 [002] dNh7. 162577.346818: sched_wakeup:         migration/2:28 [0] CPU:002
      trace-cmd-89140 [002] d..2. 162577.347038: sched_switch:         trace-cmd:89140 [120] R ==> migration/2:28 [0]
Latency: 220.583 usecs
      trace-cmd-89134 [003] dNh5. 162577.410869: sched_wakeup:         migration/3:33 [0] CPU:003
      trace-cmd-89141 [005] dNh6. 162577.574792: sched_wakeup:         migration/5:43 [0] CPU:005
      trace-cmd-89141 [005] d..2. 162577.574915: sched_switch:         trace-cmd:89141 [120] R ==> migration/5:43 [0]
Latency: 122.648 usecs
```

```
            trace-cmd-89136 [007] dNh6. 162577.719045: sched_wakeup:         migration/7:53 [0] CPU:007
            trace-cmd-89136 [007] d..2. 162577.719121: sched_switch:         trace-cmd:89136 [120] R ==> migration/7:53 [0]
Latency: 75.902 usecs
            trace-cmd-89140 [005] dNh4. 162581.574827: sched_wakeup:         migration/5:43 [0] CPU:005
            trace-cmd-89140 [005] d..2. 162581.574957: sched_switch:         trace-cmd:89140 [120] R ==> migration/5:43 [0]
Latency: 129.717 usecs
          kworker/u46:1-51211 [006] dNh4. 162581.646809: sched_wakeup:         migration/6:48 [0] CPU:006
      Average wakeup latency: 139.568 usecs
      Maximum Latency: 220.583 usecs at timestamp: 162577.347038
      Minimum Latency: 75.902 usecs at timestamp: 162577.719121
      RT task timings:
      Average wakeup latency: 139.568 usecs
      Maximum Latency: 220.583 usecs at timestamp: 162577.347038
      Minimum Latency: 75.902 usecs at timestamp: 162577.719121
```

The above version will only show the wakeups and context switches of Real Time tasks. The prio used inside the kernel starts at 0 for highest priority. That is prio 0 is equivalent to user space real time priority 99, and priority 98 is equivalent to user space real time priority 1. Prios less than 100 represent Real Time tasks. Notice that the total wake up timings are identical to the RT task timings.

An example of the profile:

```
 # trace-cmd record --profile sleep 1
 # trace-cmd report --profile --comm sleep
task: sleep-21611
  Event: sched_switch:R (1) Total: 99442 Avg: 99442 Max: 99442 Min:99442
    <stack> 1 total:99442 min:99442 max:99442 avg=99442
      => ftrace_raw_event_sched_switch (0xffffffff8105f812)
      => __schedule (0xffffffff8150810a)
      => preempt_schedule (0xffffffff8150842e)
      => ___preempt_schedule (0xffffffff81273354)
      => cpu_stop_queue_work (0xffffffff810b03c5)
      => stop_one_cpu (0xffffffff810b063b)
      => sched_exec (0xffffffff8106136d)
      => do_execve_common.isra.27 (0xffffffff81148c89)
```

```
=> do_execve (0xffffffff811490b0)
=> SyS_execve (0xffffffff811492c4)
=> return_to_handler (0xffffffff8150e3c8)
=> stub_execve (0xffffffff8150c699)
Event: sched_switch:S (1) Total: 1000506680 Avg: 1000506680 Max: 1000506680 Min:1000506680
<stack> 1 total:1000506680 min:1000506680 max:1000506680 avg=1000506680
=> ftrace_raw_event_sched_switch (0xffffffff8105f812)
=> __schedule (0xffffffff8150810a)
=> schedule (0xffffffff815084b8)
=> do_nanosleep (0xffffffff8150b22c)
=> hrtimer_nanosleep (0xffffffff8108d647)
=> SyS_nanosleep (0xffffffff8108d72c)
=> return_to_handler (0xffffffff8150e3c8)
=> tracesys_phase2 (0xffffffff8150c304)
Event: sched_wakeup:21611 (1) Total: 30326 Avg: 30326 Max: 30326 Min:30326
<stack> 1 total:30326 min:30326 max:30326 avg=30326
=> ftrace_raw_event_sched_wakeup_template (0xffffffff8105f653)
=> ttwu_do_wakeup (0xffffffff810606eb)
=> ttwu_do_activate.constprop.124 (0xffffffff810607c8)
=> try_to_wake_up (0xffffffff8106340a)
```

SEE ALSO

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