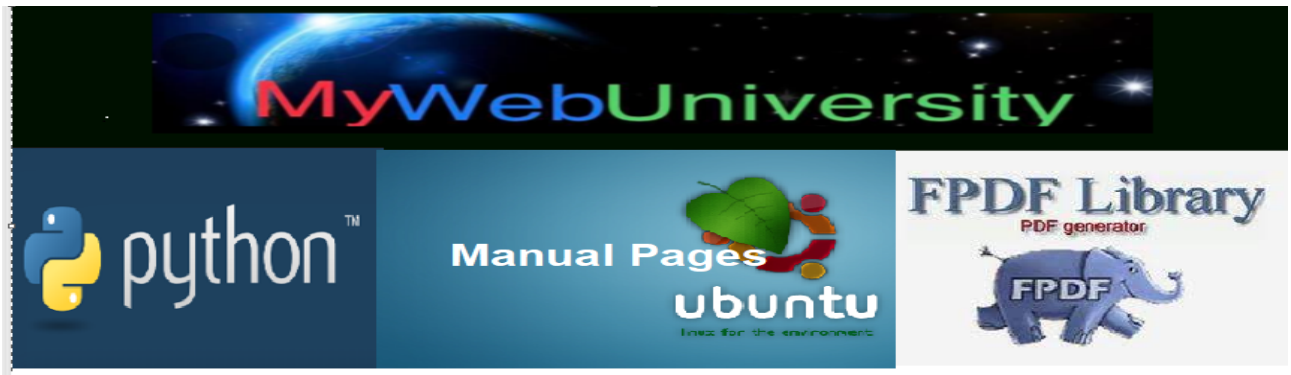

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man perf-trace.1

PERF-TRACE(1) perf Manual PERF-TRACE(1)

NAME

perf-trace - strace inspired tool

SYNOPSIS

perf trace

perf trace record

DESCRIPTION

This command will show the events associated with the target, initially syscalls, but other system events like pagefaults, task lifetime events, scheduling events, etc.

This is a live mode tool in addition to working with perf.data files like the other perf tools. Files can be generated using the perf record command but the session needs to include the raw_syscalls events (-e raw_syscalls:*). Alternatively, perf trace record can be used as a shortcut to automatically include the raw_syscalls events when writing events to a file.

The following options apply to perf trace; options to perf trace record are found in the perf record man page.

OPTIONS

-a, --all-cpus

System-wide collection from all CPUs.

-e, --expr, --event

List of syscalls and other perf events (tracepoints, HW cache events, etc) to show. Globbing is supported, e.g.: "epoll_*", "msg", etc. See perf list for a complete list of events. Prefixing with ! shows all syscalls but the ones specified. You may need to escape it.

--filter=<filter>

Event filter. This option should follow an event selector (-e) which selects tracepoint event(s).

-D msecs, --delay msecs

After starting the program, wait msecs before measuring. This is useful to filter out the startup phase of the program, which is often very different.

-o, --output=

Output file name.

-p, --pid=

Record events on existing process ID (comma separated list).

-t, --tid=

Record events on existing thread ID (comma separated list).

-u, --uid=

Record events in threads owned by uid. Name or number.

-G, --cgroup

Record events in threads in a cgroup.

Look for cgroups to set at the /sys/fs/cgroup/perf_event directory, then remove the /sys/fs/cgroup/perf_event/ part and try:

perf trace -G A -e sched:*switch

Will set all raw_syscalls:sys_{enter,exit}, pgfault, vfs_getname, etc _and_ sched:sched_switch to the 'A' cgroup, while:

perf trace -e sched:*switch -G A

will only set the sched:sched_switch event to the 'A' cgroup, all the other events (raw_syscalls:sys_{enter,exit}, etc are left "without" a cgroup (on the root cgroup, sys wide, etc).

Multiple cgroups:

perf trace -G A -e sched:*switch -G B

the syscall ones go to the 'A' cgroup, the sched:sched_switch goes to the 'B' cgroup.

--filter-pids=

Filter out events for these pids and for trace itself (comma separated list).

-v, --verbose

Increase the verbosity level.

--no-inherit

Child tasks do not inherit counters.

-m, --mmap-pages=

Number of mmap data pages (must be a power of two) or size specification in bytes with appended unit character - B/K/M/G. The size is rounded up to the nearest power-of-two page value.

-C, --cpu

Collect samples only on the list of CPUs provided. Multiple CPUs can be provided as a comma-separated list with no space: 0,1. Ranges of CPUs are specified with -: 0-2. In per-thread mode with inheritance mode on (default), Events are captured only when the thread executes on the designated CPUs. Default is to monitor all CPUs.

--duration

Show only events that had a duration greater than N.M ms.

--sched

Accrue thread runtime and provide a summary at the end of the session.

--failure

Show only syscalls that failed, i.e. that returned < 0.

-i, --input

Process events from a given perf data file.

-T, --time

Print full timestamp rather time relative to first sample.

--comm

Show process COMM right beside its ID, on by default, disable with --no-comm.

-s, --summary

Show only a summary of syscalls by thread with min, max, and average times (in msec) and relative stddev.

-S, --with-summary

Show all syscalls followed by a summary by thread with min, max, and average times (in msec) and relative stddev.

--errno-summary

To be used with -s or -S, to show stats for the errnos experienced by syscalls, using only this option will trigger --summary.

--summary-mode=mode

To be used with -s or -S, to select how to show summary. By default it?ll show the syscall summary by thread. Possible values are: thread, total, cgroup.

--tool_stats

Show tool stats such as number of times fd?pathname was discovered thru hooking the open syscall return + vfs_getname or via reading /proc/pid/fd, etc.

-f, --force

Don?t complain, do it.

-F=[all|min|maj], --pf=[all|min|maj]

Trace pagefaults. Optionally, you can specify whether you want minor, major or all pagefaults. Default value is maj.

--syscalls

Trace system calls. This options is enabled by default, disable with --no-syscalls.

--call-graph [mode,type,min[,limit],order[,key][,branch]]

Setup and enable call-graph (stack chain/backtrace) recording. See --call-graph section in perf-record and perf-report man pages for details. The ones that are most useful in perf trace are dwarf and lbr, where available, try: perf trace --call-graph dwarf.

Using this will, for the root user, bump the value of --mmap-pages to 4 times the maximum for non-root users, based on the kernel.perf_event_mlock_kb sysctl. This is done only if the user doesn't specify a --mmap-pages value.

--kernel-syscall-graph

Show the kernel callchains on the syscall exit path.

--max-events=N

Stop after processing N events. Note that strace-like events are considered only at exit time or when a syscall is interrupted, i.e. in those cases this option is equivalent to the number of lines printed.

--switch-on EVENT_NAME

Only consider events after this event is found.

--switch-off EVENT_NAME

Stop considering events after this event is found.

--show-on-off-events

Show the --switch-on/off events too.

--max-stack

Set the stack depth limit when parsing the callchain, anything beyond the specified

depth will be ignored. Note that at this point this is just about the presentation part, i.e. the kernel is still not limiting, the overhead of callchains needs to be set via the knobs in --call-graph dwarf.

Implies '--call-graph dwarf' when --call-graph not present on the command line, on systems where DWARF unwinding was built in.

Default: /proc/sys/kernel/perf_event_max_stack when present for live sessions (without --input/-i), 127 otherwise.

--min-stack

Set the stack depth limit when parsing the callchain, anything below the specified depth will be ignored. Disabled by default.

Implies '--call-graph dwarf' when --call-graph not present on the command line, on systems where DWARF unwinding was built in.

--print-sample

Print the PERF_RECORD_SAMPLE PERF_SAMPLE_ info for the raw_syscalls:sys_{enter,exit} tracepoints, for debugging.

--proc-map-timeout

When processing pre-existing threads /proc/XXX/mmap, it may take a long time, because the file may be huge. A time out is needed in such cases. This option sets the time out limit. The default value is 500 ms.

--sort-events

Do sorting on batches of events, use when noticing out of order events that may happen, for instance, when a thread gets migrated to a different CPU while processing a syscall.

--libtraceevent_print

Use libtraceevent to print tracepoint arguments. By default perf trace uses the same beautifiers used in the strace-like enter+exit lines to augment the tracepoint arguments.

--force-btf

Use btf_dump to pretty print syscall argument data, instead of using hand-crafted pretty printers. This option is intended for testing BTF integration in perf trace.

btf_dump-based pretty-printing serves as a fallback to hand-crafted pretty printers, as the latter can better pretty-print integer flags and struct pointers.

--bpf-summary

Collect system call statistics in BPF. This is only for live mode and works well with

-s/--summary option where no argument information is required.

--max-summary=N

Maximum number of lines in the summary mode. Note that this applies to each entry (thread or cgroup).

PAGEFAULTS

When tracing pagefaults, the format of the trace is as follows:

<min|maj>fault [<ip.symbol>+<ip.offset>] ? <addr.dso@addr.offset[1]> (<map type><addr level>).

? min/maj indicates whether fault event is minor or major;

? ip.symbol shows symbol for instruction pointer (the code that generated the fault); if no debug symbols available, perf trace will print raw IP;

? addr.dso shows DSO for the faulted address;

? map type is either d for non-executable maps or x for executable maps;

? addr level is either k for kernel dso or . for user dso.

For symbols resolution you may need to install debugging symbols.

Please be aware that duration is currently always 0 and doesn?t reflect actual time it took for fault to be handled!

When --verbose specified, perf trace tries to print all available information for both IP and fault address in the form of dso@symbol[2]+offset.

EXAMPLES

Trace only major pagefaults:

$ perf trace --no-syscalls -F

Trace syscalls, major and minor pagefaults:

$ perf trace -F all

1416.547 (0.000 ms): python/20235 majfault [CRYPTO_push_info_+0x0] => /lib/x86_64-linux-gnu/libcrypto.so.1.0.0@0x61be0 (x.)

As you can see, there was major pagefault in python process, from CRYPTO_push_info_ routine which faulted somewhere in libcrypto.so.

Trace the first 4 open, openat or open_by_handle_at syscalls (in the future more syscalls may match here):

$ perf trace -e open* --max-events 4

[root@jouet perf]# trace -e open* --max-events 4

2272.992 (0.037 ms): gnome-shell/1370 openat(dfd: CWD, filename: /proc/self/stat) = 31

```
    2277.481 ( 0.139 ms): gnome-shell/3039 openat(dfd: CWD, filename: /proc/self/stat) = 65
    3026.398 ( 0.076 ms): gnome-shell/3039 openat(dfd: CWD, filename: /proc/self/stat) = 65
    4294.665 ( 0.015 ms): sed/15879 openat(dfd: CWD, filename: /etc/ld.so.cache, flags: CLOEXEC) = 3
    $
```

Trace the first minor page fault when running a workload:

```
    # perf trace -F min --max-stack=7 --max-events 1 sleep 1
      0.000 ( 0.000 ms): sleep/18006 minfault [__clear_user+0x1a] => 0x5626efa56080 (?k)
                                __clear_user ([kernel.kallsyms])
                                load_elf_binary ([kernel.kallsyms])
                                search_binary_handler ([kernel.kallsyms])
                                __do_execve_file.isra.33 ([kernel.kallsyms])
                                __x64_sys_execve ([kernel.kallsyms])
                                do_syscall_64 ([kernel.kallsyms])
                                entry_SYSCALL_64 ([kernel.kallsyms])
    #
```

Trace the next min page page fault to take place on the first CPU:

```
    # perf trace -F min --call-graph=dwarf --max-events 1 --cpu 0
                0.000 ( 0.000 ms): Web Content/17136 minfault [js::gc::Chunk::fetchNextDecommittedArena+0x4b] =>
0x7fbe6181b000 (?.)
                                js::gc::FreeSpan::initAsEmpty (inlined)
                                js::gc::Arena::setAsNotAllocated (inlined)
                                js::gc::Chunk::fetchNextDecommittedArena (/usr/lib64/firefox/libxul.so)
                                js::gc::Chunk::allocateArena (/usr/lib64/firefox/libxul.so)
                                js::gc::GCRuntime::allocateArena (/usr/lib64/firefox/libxul.so)
                                js::gc::ArenaLists::allocateFromArena (/usr/lib64/firefox/libxul.so)
                                js::gc::GCRuntime::tryNewTenuredThing<JSString, (js::AllowGC)1> (inlined)
                                js::AllocateString<JSString, (js::AllowGC)1> (/usr/lib64/firefox/libxul.so)
                                js::Allocate<JSThinInlineString, (js::AllowGC)1> (inlined)
                                JSThinInlineString::new_<(js::AllowGC)1> (inlined)
                                AllocateInlineString<(js::AllowGC)1, unsigned char> (inlined)
                                js::ConcatStrings<(js::AllowGC)1> (/usr/lib64/firefox/libxul.so)
                                [0x18b26e6bc2bd] (/tmp/perf-17136.map)
    #
```

Trace the next two sched:sched_switch events, four block:*_plug events, the next block:*_unplug and the next three net:*dev_queue events, this last one with a backtrace of at most 16 entries, system wide:

```
# perf trace -e sched:*switch/nr=2/,block:*_plug/nr=4/,block:*_unplug/nr=1/,net:*dev_queue/nr=3,max-stack=16/
  0.000 :0/0 sched:sched_switch:swapper/2:0 [120] S ==> rcu_sched:10 [120]
  0.015 rcu_sched/10 sched:sched_switch:rcu_sched:10 [120] R ==> swapper/2:0 [120]
254.198 irq/50-iwlwifi/680 net:net_dev_queue:dev=wlp3s0 skbaddr=0xffff93498051f600 len=66
                           __dev_queue_xmit ([kernel.kallsyms])
273.977 :0/0 net:net_dev_queue:dev=wlp3s0 skbaddr=0xffff93498051f600 len=78
                           __dev_queue_xmit ([kernel.kallsyms])
274.007 :0/0 net:net_dev_queue:dev=wlp3s0 skbaddr=0xffff93498051ff00 len=78
                           __dev_queue_xmit ([kernel.kallsyms])
2930.140 kworker/u16:58/2722 block:block_plug:[kworker/u16:58]
2930.162 kworker/u16:58/2722 block:block_unplug:[kworker/u16:58] 1
4466.094 jbd2/dm-2-8/748 block:block_plug:[jbd2/dm-2-8]
8050.123 kworker/u16:30/2694 block:block_plug:[kworker/u16:30]
8050.271 kworker/u16:30/2694 block:block_plug:[kworker/u16:30]
#
```

SEE ALSO

perf-record(1), perf-script(1)

NOTES

1. addr.dso@addr.offset

   mailto:addr.dso@addr.offset

2. dso@symbol

   mailto:dso@symbol