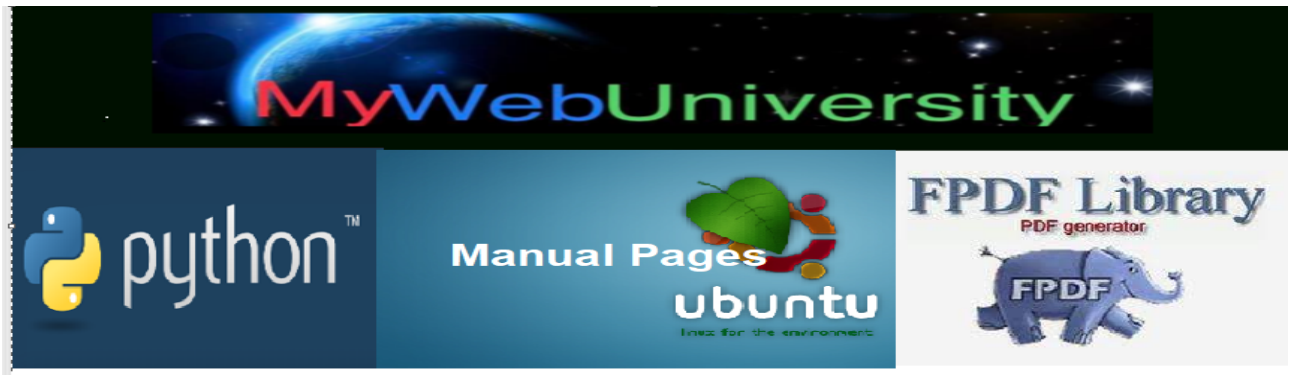

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man perf-record.1

PERF-RECORD(1) perf Manual PERF-RECORD(1)

NAME

perf-record - Run a command and record its profile into perf.data

SYNOPSIS

perf record [-e <EVENT> | --event=EVENT] [-a] <command>

perf record [-e <EVENT> | --event=EVENT] [-a] -- <command> [<options>]

DESCRIPTION

This command runs a command and gathers a performance counter profile from it, into perf.data - without displaying anything.

This file can then be inspected later on, using perf report.

OPTIONS

<command>...

Any command you can specify in a shell.

-e, --event=

Select the PMU event. Selection can be:

? a symbolic event name (use perf list to list all events)

? a raw PMU event in the form of rN where N is a hexadecimal value that represents the raw register encoding with the layout of the event control registers as described by entries in /sys/bus/event_source/devices/cpu/format/*.

? a symbolic or raw PMU event followed by an optional colon and a list of event modifiers, e.g., cpu-cycles:p. See the perf-list(1) man page for details on event modifiers.

? a symbolically formed PMU event like pmu/param1=0x3,param2/ where param1, param2,

etc are defined as formats for the PMU in
/sys/bus/event_source/devices/<pmu>/format/*.

? a symbolically formed event like pmu/config=M,config1=N,config3=K/
where M, N, K are numbers (in decimal, hex, octal format). Acceptable
values for each of 'config', 'config1' and 'config2' are defined by
corresponding entries in /sys/bus/event_source/devices/<pmu>/format/*
param1 and param2 are defined as formats for the PMU in:
/sys/bus/event_source/devices/<pmu>/format/*
There are also some parameters which are not defined in .../<pmu>/format/*.
These params can be used to overload default config values per event.
Here are some common parameters:
- 'period': Set event sampling period
- 'freq': Set event sampling frequency
- 'time': Disable/enable time stamping. Acceptable values are 1 for
enabling time stamping. 0 for disabling time stamping.
The default is 1.
- 'call-graph': Disable/enable callgraph. Acceptable str are "fp" for
FP mode, "dwarf" for DWARF mode, "lbr" for LBR mode and
"no" for disable callgraph.
- 'stack-size': user stack size for dwarf mode
- 'name' : User defined event name. Single quotes (') may be used to
escape symbols in the name from parsing by shell and tool
like this: name=\'CPU_CLK_UNHALTED.THREAD:cmask=0x1\'.
- 'aux-output': Generate AUX records instead of events. This requires
that an AUX area event is also provided.
- 'aux-action': "pause" or "resume" to pause or resume an AUX
area event (the group leader) when this event occurs.
"start-paused" on an AUX area event itself, will
start in a paused state.
- 'aux-sample-size': Set sample size for AUX area sampling. If the
'--aux-sample' option has been used, set aux-sample-size=0 to disable
AUX area sampling for the event.
See the linkperf:perf-list[1] man page for more parameters.

Note: If user explicitly sets options which conflict with the params,
the value set by the parameters will be overridden.
Also not defined in .../<pmu>/format/* are PMU driver specific
configuration parameters. Any configuration parameter preceded by
the letter '@' is not interpreted in user space and sent down directly
to the PMU driver. For example:
perf record -e some_event/@cfg1,@cfg2=config/ ...
will see 'cfg1' and 'cfg2=config' pushed to the PMU driver associated
with the event for further processing. There is no restriction on
what the configuration parameters are, as long as their semantic is
understood and supported by the PMU driver.

? a hardware breakpoint event in the form of \mem:addr[/len][:access] where addr is the address in memory you want to break in. Access is the memory access type (read, write, execute) it can be passed as follows: \mem:addr[:[r][w][x]]. len is the range, number of bytes from specified addr, which the breakpoint will cover. If you want to profile read-write accesses in 0x1000, just set mem:0x1000:rw. If you want to profile write accesses in [0x1000~1008), just set mem:0x1000/8:w.

? a group of events surrounded by a pair of brace ("{event1,event2,...}"). Each event is separated by commas and the group should be quoted to prevent the shell interpretation. You also need to use --group on "perf report" to view group events together.

--filter=<filter>

Event filter. This option should follow an event selector (-e). If the event is a tracepoint, the filter string will be parsed by the kernel. If the event is a hardware trace PMU (e.g. Intel PT or CoreSight), it?ll be processed as an address filter. Otherwise it means a general filter using BPF which can be applied for any kind of event.

? tracepoint filters

In the case of tracepoints, multiple '--filter' options are combined
using '&&'.

? address filters

A hardware trace PMU advertises its ability to accept a number of
address filters by specifying a non-zero value in

/sys/bus/event_source/devices/<pmu>/nr_addr_filters.

Address filters have the format:

filter|start|stop|tracestop <start> [/ <size>] [@<file name>]

Where:

- 'filter': defines a region that will be traced.

- 'start': defines an address at which tracing will begin.

- 'stop': defines an address at which tracing will stop.

- 'tracestop': defines a region in which tracing will stop.

<file name> is the name of the object file, <start> is the offset to the code to trace in that file, and <size> is the size of the region to trace. 'start' and 'stop' filters need not specify a <size>.

If no object file is specified then the kernel is assumed, in which case the start address must be a current kernel memory address.

<start> can also be specified by providing the name of a symbol. If the symbol name is not unique, it can be disambiguated by inserting #n where 'n' selects the n'th symbol in address order. Alternately #0, #g or #G select only a global symbol. <size> can also be specified by providing the name of a symbol, in which case the size is calculated to the end of that symbol. For 'filter' and 'tracestop' filters, if <size> is omitted and <start> is a symbol, then the size is calculated to the end of that symbol.

If <size> is omitted and <start> is '*', then the start and size will be calculated from the first and last symbols, i.e. to trace the whole file.

If symbol names (or '*') are provided, they must be surrounded by white space.

The filter passed to the kernel is not necessarily the same as entered. To see the filter that is passed, use the -v option.

The kernel may not be able to configure a trace region if it is not within a single mapping. MMAP events (or /proc/<pid>/maps) can be examined to determine if that is a possibility.

Multiple filters can be separated with space or comma.

A BPF filter can access the sample data and make a decision based on the data. Users need to set an appropriate sample type to use the BPF filter. BPF filters need root privilege.

The sample data field can be specified in lower case letter. Multiple filters can be separated with comma. For example,

```
  --filter 'period > 1000, cpu == 1'
```

or

```
  --filter 'mem_op == load || mem_op == store, mem_lvl > l1'
```

The former filter only accept samples with period greater than 1000 AND CPU number is 1. The latter one accepts either load and store memory operations but it should have memory level above the L1. Since the mem_op and mem_lvl fields come from the (memory) data_source, it'd only work with some events which set the data_source field.

Also user should request to collect that information (with -d option in the above case). Otherwise, the following message will be shown.

```
$ sudo perf record -e cycles --filter 'mem_op == load'
Error: cycles event does not have PERF_SAMPLE_DATA_SRC
 Hint: please add -d option to perf record.
failed to set filter "BPF" on event cycles with 22 (Invalid argument)
```

Essentially the BPF filter expression is:

```
<term> <operator> <value> ("," | "||") <term> <operator> <value>)*
```

The <term> can be one of:

```
  ip, id, tid, pid, cpu, time, addr, period, txn, weight, phys_addr,
  code_pgsz, data_pgsz, weight1, weight2, weight3, ins_lat, retire_lat,
  p_stage_cyc, mem_op, mem_lvl, mem_snoop, mem_remote, mem_lock,
  mem_dtlb, mem_blk, mem_hops, uid, gid
```

The <operator> can be one of:

```
  ==, !=, >, >=, <, <=, &
```

The <value> can be one of:

```
  <number> (for any term)
  na, load, store, pfetch, exec (for mem_op)
  l1, l2, l3, l4, cxl, io, any_cache, lfb, ram, pmem (for mem_lvl)
  na, none, hit, miss, hitm, fwd, peer (for mem_snoop)
```

remote (for mem_remote)

na, locked (for mem_locked)

na, l1_hit, l1_miss, l2_hit, l2_miss, any_hit, any_miss, walk, fault (for mem_dtlb)

na, by_data, by_addr (for mem_blk)

hops0, hops1, hops2, hops3 (for mem_hops)

--exclude-perf

Don?t record events issued by perf itself. This option should follow an event selector (-e) which selects tracepoint event(s). It adds a filter expression common_pid != $PERFPID to filters. If other --filter exists, the new filter expression will be combined with them by &&.

--latency

Enable data collection for latency profiling. Use perf report --latency for latency-centric profile.

-a, --all-cpus

System-wide collection from all CPUs (default if no target is specified).

-p, --pid=

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

-t, --tid=

Record events on existing thread ID (comma separated list). This option also disables inheritance by default. Enable it by adding --inherit.

-u, --uid=

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

-r, --realtime=

Collect data with this RT SCHED_FIFO priority.

--no-buffering

Collect data without buffering.

-c, --count=

Event period to sample.

-o, --output=

Output file name.

-i, --no-inherit

Child tasks do not inherit counters.

-F, --freq=

Profile at this frequency. Use max to use the currently maximum allowed frequency, i.e. the value in the kernel.perf_event_max_sample_rate sysctl. Will throttle down to the currently maximum allowed frequency. See --strict-freq.

--strict-freq

Fail if the specified frequency can?t be used.

-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. By adding a comma, an additional parameter with the same semantics used for the normal mmap areas can be specified for AUX tracing area.

-g

Enables call-graph (stack chain/backtrace) recording for both kernel space and user space.

--call-graph

Setup and enable call-graph (stack chain/backtrace) recording, implies -g. Default is "fp" (for user space).

The unwinding method used for kernel space is dependent on the unwinder used by the active kernel configuration, i.e CONFIG_UNWINDER_FRAME_POINTER (fp) or CONFIG_UNWINDER_ORC (orc)

Any option specified here controls the method used for user space.

Valid options are "fp" (frame pointer), "dwarf" (DWARF's CFI - Call Frame Information) or "lbr" (Hardware Last Branch Record facility).

In some systems, where binaries are build with gcc --fomit-frame-pointer, using the "fp" method will produce bogus call graphs, using "dwarf", if available (perf tools linked to the libunwind or libdw library) should be used instead.

Using the "lbr" method doesn't require any compiler options. It will produce call graphs from the hardware LBR registers. The main limitation is that it is only available on new Intel platforms, such as Haswell. It can only get user call chain. It doesn't work with branch stack sampling at the same time.

When "dwarf" recording is used, perf also records (user) stack dump

when sampled. Default size of the stack dump is 8192 (bytes).

User can change the size by passing the size after comma like

"--call-graph dwarf,4096".

When "fp" recording is used, perf tries to save stack entries

up to the number specified in sysctl.kernel.perf_event_max_stack

by default. User can change the number by passing it after comma

like "--call-graph fp,32".

Also "defer" can be used with "fp" (like "--call-graph fp,defer") to

enable deferred user callchain which will collect user-space callchains

when the thread returns to the user space.

-q, --quiet

Don?t print any warnings or messages, useful for scripting.

-v, --verbose

Be more verbose (show counter open errors, etc).

-s, --stat

Record per-thread event counts. Use it with perf report -T to see the values.

-d, --data

Record the sample virtual addresses. Implies --sample-mem-info and --data-mmap.

--phys-data

Record the sample physical addresses.

--data-page-size

Record the sampled data address data page size.

--code-page-size

Record the sampled code address (ip) page size

-T, --timestamp

Record the sample timestamps. Use it with perf report -D to see the timestamps, for

instance.

-P, --period

Record the sample period.

--sample-cpu

Record the sample cpu.

--sample-identifier

Record the sample identifier i.e. PERF_SAMPLE_IDENTIFIER bit set in the sample_type

member of the struct perf_event_attr argument to the perf_event_open system call.

--sample-mem-info

Record the sample data source information for memory operations. It requires hardware supports and may work on specific events only. Please consider using perf mem record instead if you?re not sure.

-n, --no-samples

Don?t sample.

-R, --raw-samples

Collect raw sample records from all opened counters (default for tracepoint counters).

-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), samples are captured only when the thread executes on the designated CPUs. Default is to monitor all CPUs.

User space tasks can migrate between CPUs, so when tracing selected CPUs, a dummy event is created to track sideband for all CPUs.

-B, --no-buildid

Do not save the build ids of binaries in the perf.data files. This skips post processing after recording, which sometimes makes the final step in the recording process to take a long time, as it needs to process all events looking for mmap records. The downside is that it can misresolve symbols if the workload binaries used when recording get locally rebuilt or upgraded, because the only key available in this case is the pathname. You can also set the "record.build-id" config variable to 'skip to have this behaviour permanently.

-N, --no-buildid-cache

Do not update the buildid cache. This saves some overhead in situations where the information in the perf.data file (which includes buildids) is sufficient. You can also set the "record.build-id" config variable to no-cache to have the same effect.

-G name,..., --cgroup name,...

monitor only in the container (cgroup) called "name". This option is available only in per-cpu mode. The cgroup filesystem must be mounted. All threads belonging to container "name" are monitored when they run on the monitored CPUs. Multiple cgroups can be provided. Each cgroup is applied to the corresponding event, i.e., first cgroup to first

event, second cgroup to second event and so on. It is possible to provide an empty cgroup (monitor all the time) using, e.g., -G foo,,bar. Cgroups must have corresponding events, i.e., they always refer to events defined earlier on the command line. If the user wants to track multiple events for a specific cgroup, the user can use -e e1 -e e2 -G foo,foo or just use -e e1 -e e2 -G foo.

If wanting to monitor, say, cycles for a cgroup and also for system wide, this command line can be used: perf stat -e cycles -G cgroup_name -a -e cycles.

-b, --branch-any

Enable taken branch stack sampling. Any type of taken branch may be sampled. This is a shortcut for --branch-filter any. See --branch-filter for more infos.

-j, --branch-filter

Enable taken branch stack sampling. Each sample captures a series of consecutive taken branches. The number of branches captured with each sample depends on the underlying hardware, the type of branches of interest, and the executed code. It is possible to select the types of branches captured by enabling filters. The following filters are defined:

- any: any type of branches
- any_call: any function call or system call
- any_ret: any function return or system call return
- ind_call: any indirect branch
- ind_jmp: any indirect jump
- call: direct calls, including far (to/from kernel) calls
- u: only when the branch target is at the user level
- k: only when the branch target is in the kernel
- hv: only when the target is at the hypervisor level
- in_tx: only when the target is in a hardware transaction
- no_tx: only when the target is not in a hardware transaction
- abort_tx: only when the target is a hardware transaction abort
- cond: conditional branches
- stack: save call stack
- no_flags: don?t save branch flags e.g prediction, misprediction etc
- no_cycles: don?t save branch cycles
- hw_index: save branch hardware index

? save_type: save branch type during sampling in case binary is not available later

For the platforms with Intel Arch LBR support (12th-Gen+ client or 4th-Gen Xeon+ server), the save branch type is unconditionally enabled when the taken branch stack sampling is enabled.

? priv: save privilege state during sampling in case binary is not available later

? counter: save occurrences of the event since the last branch entry. Currently, the feature is only supported by a newer CPU, e.g., Intel Sierra Forest and later platforms. An error out is expected if it?s used on the unsupported kernel or CPUs.

The option requires at least one branch type among any, any_call, any_ret, ind_call, cond. The privilege levels may be omitted, in which case, the privilege levels of the associated event are applied to the branch filter. Both kernel (k) and hypervisor (hv) privilege levels are subject to permissions. When sampling on multiple events, branch stack sampling is enabled for all the sampling events. The sampled branch type is the same for all events. The various filters must be specified as a comma separated list: --branch-filter any_ret,u,k Note that this feature may not be available on all processors.

-W, --weight

Enable weightened sampling. An additional weight is recorded per sample and can be displayed with the weight and local_weight sort keys. This currently works for TSX abort events and some memory events in precise mode on modern Intel CPUs.

--namespaces

Record events of type PERF_RECORD_NAMESPACES. This enables cgroup_id sort key.

--all-cgroups

Record events of type PERF_RECORD_CGROUP. This enables cgroup sort key.

--transaction

Record transaction flags for transaction related events.

--per-thread

Use per-thread mmaps. By default per-cpu mmaps are created. This option overrides that and uses per-thread mmaps. A side-effect of that is that inheritance is automatically disabled. --per-thread is ignored with a warning if combined with -a or -C options.

-D, --delay=

After starting the program, wait msecs before measuring (-1: start with events disabled), or enable events only for specified ranges of msecs (e.g. -D 10-20,30-40

means wait 10 msecs, enable for 10 msecs, wait 10 msecs, enable for 10 msecs, then stop). Note, delaying enabling of events is useful to filter out the startup phase of the program, which is often very different.

-I, --intr-regs

Capture machine state (registers) at interrupt, i.e., on counter overflows for each sample. List of captured registers depends on the architecture. This option is off by default. It is possible to select the registers to sample using their symbolic names, e.g. on x86, ax, si. To list the available registers use --intr-regs=\?. To name registers, pass a comma separated list such as --intr-regs=ax,bx. The list of register is architecture dependent.

--user-regs

Similar to -I, but capture user registers at sample time. To list the available user registers use --user-regs=\?.

--running-time

Record running and enabled time for read events (:S)

-k, --clockid

Sets the clock id to use for the various time fields in the perf_event_type records. See clock_gettime(). In particular CLOCK_MONOTONIC and CLOCK_MONOTONIC_RAW are supported, some events might also allow CLOCK_BOOTTIME, CLOCK_REALTIME and CLOCK_TAI.

-S, --snapshot

Select AUX area tracing Snapshot Mode. This option is valid only with an AUX area tracing event. Optionally, certain snapshot capturing parameters can be specified in a string that follows this option:

? e: take one last snapshot on exit; guarantees that there is at least one snapshot in the output file;

? <size>: if the PMU supports this, specify the desired snapshot size.

In Snapshot Mode trace data is captured only when signal SIGUSR2 is received and on exit if the above e option is given.

--aux-sample[=OPTIONS]

Select AUX area sampling. At least one of the events selected by the -e option must be an AUX area event. Samples on other events will be created containing data from the AUX area. Optionally sample size may be specified, otherwise it defaults to 4KiB.

--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.

--switch-events

Record context switch events i.e. events of type PERF_RECORD_SWITCH or PERF_RECORD_SWITCH_CPU_WIDE. In some cases (e.g. Intel PT, CoreSight or Arm SPE) switch events will be enabled automatically, which can be suppressed by by the option --no-switch-events.

--vmlinux=PATH

Specify vmlinux path which has debuginfo. (enabled when BPF prologue is on)

--buildid-all

Record build-id of all DSOs regardless whether it?s actually hit or not.

--buildid-mmap

Legacy record build-id in map events option which is now the default. Behaves indentically to --no-buildid. Disable with --no-buildid-mmap.

--aio[=n]

Use <n> control blocks in asynchronous (Posix AIO) trace writing mode (default: 1, max: 4). Asynchronous mode is supported only when linking Perf tool with libc library providing implementation for Posix AIO API.

--affinity=mode

Set affinity mask of trace reading thread according to the policy defined by mode value:

? node - thread affinity mask is set to NUMA node cpu mask of the processed mmap buffer

? cpu - thread affinity mask is set to cpu of the processed mmap buffer

--mmap-flush=number

Specify minimal number of bytes that is extracted from mmap data pages and processed for output. One can specify the number using B/K/M/G suffixes.

The maximal allowed value is a quarter of the size of mmaped data pages.

The default option value is 1 byte which means that every time that the output writing thread finds some new data in the mmaped buffer the data is extracted, possibly compressed (-z) and written to the output, perf.data or pipe.

Larger data chunks are compressed more effectively in comparison to smaller chunks so extraction of larger chunks from the mmap data pages is preferable from the perspective of

output size reduction.

Also at some cases executing less output write syscalls with bigger data size can take less time than executing more output write syscalls with smaller data size thus lowering runtime profiling overhead.

-z, --compression-level[=n]

Produce compressed trace using specified level n (default: 1 - fastest compression, 22 - smallest trace)

--all-kernel

Configure all used events to run in kernel space.

--all-user

Configure all used events to run in user space.

--kernel-callchains

Collect callchains only from kernel space. I.e. this option sets perf_event_attr.exclude_callchain_user to 1.

--user-callchains

Collect callchains only from user space. I.e. this option sets perf_event_attr.exclude_callchain_kernel to 1.

Don?t use both --kernel-callchains and --user-callchains at the same time or no callchains will be collected.

--timestamp-filename Append timestamp to output file name.

--timestamp-boundary

Record timestamp boundary (time of first/last samples).

--switch-output[=mode]

Generate multiple perf.data files, timestamp prefixed, switching to a new one based on mode value:

? "signal" - when receiving a SIGUSR2 (default value) or

? <size> - when reaching the size threshold, size is expected to be a number with appended unit character - B/K/M/G

? <time> - when reaching the time threshold, size is expected to be a number with appended unit character - s/m/h/d

Note: the precision of the size threshold hugely depends on your configuration - the number and size of your ring buffers (-m). It is generally more precise for higher sizes

(like >5M), for lower values expect different sizes.

A possible use case is to, given an external event, slice the perf.data file that gets then processed, possibly via a perf script, to decide if that particular perf.data snapshot should be kept or not.

Implies --timestamp-filename, --no-buildid and --no-buildid-cache. The reason for the latter two is to reduce the data file switching overhead. You can still switch them on with:

--switch-output --no-no-buildid --no-no-buildid-cache

--switch-output-event

Events that will cause the switch of the perf.data file, auto-selecting --switch-output=signal, the results are similar as internally the side band thread will also send a SIGUSR2 to the main one.

Uses the same syntax as --event, it will just not be recorded, serving only to switch the perf.data file as soon as the --switch-output event is processed by a separate sideband thread.

This sideband thread is also used to other purposes, like processing the PERF_RECORD_BPF_EVENT records as they happen, asking the kernel for extra BPF information, etc.

--switch-max-files=N

When rotating perf.data with --switch-output, only keep N files.

--dry-run

Parse options then exit. --dry-run can be used to detect errors in cmdline options.

perf record --dry-run -e can act as a BPF script compiler if llvm.dump-obj in config file is set to true.

--synth=TYPE

Collect and synthesize given type of events (comma separated). Note that this option controls the synthesis from the /proc filesystem which represent task status for pre-existing threads.

Kernel (and some other) events are recorded regardless of the choice in this option. For example, --synth=no would have MMAP events for kernel and modules.

Available types are:

? task - synthesize FORK and COMM events for each task

? mmap - synthesize MMAP events for each process (implies task)

? cgroup - synthesize CGROUP events for each cgroup

? all - synthesize all events (default)

? no - do not synthesize any of the above events

--tail-synthesize

Instead of collecting non-sample events (for example, fork, comm, mmap) at the beginning of record, collect them during finalizing an output file. The collected non-sample events reflects the status of the system when record is finished.

--overwrite

Makes all events use an overwritable ring buffer. An overwritable ring buffer works like a flight recorder: when it gets full, the kernel will overwrite the oldest records, that thus will never make it to the perf.data file.

When --overwrite and --switch-output are used perf records and drops events until it receives a signal, meaning that something unusual was detected that warrants taking a snapshot of the most current events, those fitting in the ring buffer at that moment.

overwrite attribute can also be set or canceled for an event using config terms. For example: cycles/overwrite/ and instructions/no-overwrite/.

Implies --tail-synthesize.

--kcore

Make a copy of /proc/kcore and place it into a directory with the perf data file.

--max-size=<size>

Limit the sample data max size, <size> is expected to be a number with appended unit character - B/K/M/G

--num-thread-synthesize

The number of threads to run when synthesizing events for existing processes. By default, the number of threads equals 1.

--control=fifo:ctl-fifo[,ack-fifo], --control=fd:ctl-fd[,ack-fd]

ctl-fifo / ack-fifo are opened and used as ctl-fd / ack-fd as follows. Listen on ctl-fd descriptor for command to control measurement.

Available commands:

? enable : enable events

? disable : disable events

? enable name : enable event name

? disable name : disable event name

? snapshot : AUX area tracing snapshot).

? stop : stop perf record

? ping : ping

? 'evlist [-v|-g|-F] : display all events

-F Show just the sample frequency used for each event.

-v Show all fields.

-g Show event group information.

Measurements can be started with events disabled using --delay=-1 option. Optionally send control command completion (ack\n) to ack-fd descriptor to synchronize with the controlling process. Example of bash shell script to enable and disable events during measurements:

```
#!/bin/bash
ctl_dir=/tmp/
ctl_fifo=${ctl_dir}perf_ctl.fifo
test -p ${ctl_fifo} && unlink ${ctl_fifo}
mkfifo ${ctl_fifo}
exec {ctl_fd}<>${ctl_fifo}
ctl_ack_fifo=${ctl_dir}perf_ctl_ack.fifo
test -p ${ctl_ack_fifo} && unlink ${ctl_ack_fifo}
mkfifo ${ctl_ack_fifo}
exec {ctl_fd_ack}<>${ctl_ack_fifo}
perf record -D -1 -e cpu-cycles -a              \
            --control fd:${ctl_fd},${ctl_fd_ack} \
            -- sleep 30 &
perf_pid=$!
sleep 5  && echo 'enable' >&${ctl_fd} && read -u ${ctl_fd_ack} e1 && echo "enabled(${e1})"
sleep 10 && echo 'disable' >&${ctl_fd} && read -u ${ctl_fd_ack} d1 && echo "disabled(${d1})"
exec {ctl_fd_ack}>&-
unlink ${ctl_ack_fifo}
exec {ctl_fd}>&-
unlink ${ctl_fifo}
wait -n ${perf_pid}
exit $?
```

--threads=<spec>

Write collected trace data into several data files using parallel threads. <spec> value

can be user defined list of masks. Masks separated by colon define CPUs to be monitored by a thread and affinity mask of that thread is separated by slash:

<cpus mask 1>/<affinity mask 1>:<cpus mask 2>/<affinity mask 2>:...

CPUs or affinity masks must not overlap with other corresponding masks. Invalid CPUs are ignored, but masks containing only invalid CPUs are not allowed.

For example user specification like the following:

0,2-4/2-4:1,5-7/5-7

specifies parallel threads layout that consists of two threads, the first thread monitors CPUs 0 and 2-4 with the affinity mask 2-4, the second monitors CPUs 1 and 5-7 with the affinity mask 5-7.

<spec> value can also be a string meaning predefined parallel threads layout:

? cpu - create new data streaming thread for every monitored cpu

? core - create new thread to monitor CPUs grouped by a core

? package - create new thread to monitor CPUs grouped by a package

? numa - create new threed to monitor CPUs grouped by a NUMA domain

Predefined layouts can be used on systems with large number of CPUs in order not to spawn multiple per-cpu streaming threads but still avoid LOST events in data directory files.

Option specified with no or empty value defaults to CPU layout. Masks defined or provided by the option value are filtered through the mask provided by -C option.

--debuginfod[=URLs]

Specify debuginfod URL to be used when cacheing perf.data binaries, it follows the same syntax as the DEBUGINFOD_URLS variable, like:

http://192.168.122.174:8002

If the URLs is not specified, the value of DEBUGINFOD_URLS system environment variable is used.

--off-cpu

Enable off-cpu profiling with BPF. The BPF program will collect task scheduling information with (user) stacktrace and save them as sample data of a software event named "offcpu-time". The sample period will have the time the task slept in nanoseconds.

Note that BPF can collect stack traces using frame pointer ("fp") only, as of now. So the applications built without the frame pointer might see bogus addresses.

off-cpu profiling consists two types of samples: direct samples, which

share the same behavior as regular samples, and the accumulated samples, stored in BPF stack trace map, presented after all the regular samples.

--off-cpu-thresh

Once a task?s off-cpu time reaches this threshold (in milliseconds), it generates a direct off-cpu sample. The default is 500ms.

--setup-filter=<action>

Prepare BPF filter to be used by regular users. The action should be either "pin" or "unpin". The filter can be used after it?s pinned.

--data-mmap

Enable recording MMAP events for non-executable mappings. Basically perf only records executable mappings but data mmaping can be useful when you analyze data access with sample addresses. So using -d option would enable this unless you specify --no-data-mmap manually.

INTEL HYBRID SUPPORT

Support for Intel hybrid events within perf tools.

For some Intel platforms, such as AlderLake, which is hybrid platform and it consists of atom cpu and core cpu. Each cpu has dedicated event list. Part of events are available on core cpu, part of events are available on atom cpu and even part of events are available on both.

Kernel exports two new cpu pmus via sysfs: /sys/bus/event_source/devices/cpu_core /sys/bus/event_source/devices/cpu_atom

The cpus files are created under the directories. For example,

cat /sys/bus/event_source/devices/cpu_core/cpus 0-15

cat /sys/bus/event_source/devices/cpu_atom/cpus 16-23

It indicates cpu0-cpu15 are core cpus and cpu16-cpu23 are atom cpus.

As before, use perf-list to list the symbolic event.

perf list

inst_retired.any [Fixed Counter: Counts the number of instructions retired. Unit: cpu_atom]

inst_retired.any [Number of instructions retired. Fixed Counter - architectural event. Unit: cpu_core]

The Unit: xxx is added to brief description to indicate which pmu the event is belong to.

Same event name but with different pmu can be supported.

Enable hybrid event with a specific pmu

To enable a core only event or atom only event, following syntax is supported:

```
        cpu_core/<event name>/
  or
        cpu_atom/<event name>/
```

For example, count the cycles event on core cpus.

```
  perf stat -e cpu_core/cycles/
```

Create two events for one hardware event automatically

When creating one event and the event is available on both atom and core, two events are created automatically. One is for atom, the other is for core. Most of hardware events and cache events are available on both cpu_core and cpu_atom.

For hardware events, they have pre-defined configs (e.g. 0 for cycles). But on hybrid platform, kernel needs to know where the event comes from (from atom or from core). The original perf event type PERF_TYPE_HARDWARE can?t carry pmu information. So now this type is extended to be PMU aware type. The PMU type ID is stored at attr.config[63:32].

PMU type ID is retrieved from sysfs. /sys/bus/event_source/devices/cpu_atom/type

/sys/bus/event_source/devices/cpu_core/type

The new attr.config layout for PERF_TYPE_HARDWARE:

PERF_TYPE_HARDWARE: 0xEEEEEEEE000000AA AA: hardware event ID EEEEEEEE: PMU type ID

Cache event is similar. The type PERF_TYPE_HW_CACHE is extended to be PMU aware type. The PMU type ID is stored at attr.config[63:32].

The new attr.config layout for PERF_TYPE_HW_CACHE:

PERF_TYPE_HW_CACHE: 0xEEEEEEEE00DDCCBB BB: hardware cache ID CC: hardware cache op ID DD: hardware cache op result ID EEEEEEEE: PMU type ID

When enabling a hardware event without specified pmu, such as, perf stat -e cycles -a (use system-wide in this example), two events are created automatically.

```
  ------------------------------------------------------------
  perf_event_attr:
    size                    120
    config                  0x400000000
    sample_type             IDENTIFIER
    read_format             TOTAL_TIME_ENABLED|TOTAL_TIME_RUNNING
    disabled                1
```

inherit 1

exclude_guest 1

---

and

---

perf_event_attr:

size 120

config 0x800000000

sample_type IDENTIFIER

read_format TOTAL_TIME_ENABLED|TOTAL_TIME_RUNNING

disabled 1

inherit 1

exclude_guest 1

---

type 0 is PERF_TYPE_HARDWARE. 0x4 in 0x400000000 indicates it?s cpu_core pmu. 0x8 in 0x800000000 indicates it?s cpu_atom pmu (atom pmu type id is random).

The kernel creates cycles (0x400000000) on cpu0-cpu15 (core cpus), and create cycles (0x800000000) on cpu16-cpu23 (atom cpus).

For perf-stat result, it displays two events:

Performance counter stats for 'system wide':

6,744,979 cpu_core/cycles/

1,965,552 cpu_atom/cycles/

The first cycles is core event, the second cycles is atom event.

Thread mode example:

perf-stat reports the scaled counts for hybrid event and with a percentage displayed. The percentage is the event?s running time/enabling time.

One example, triad_loop runs on cpu16 (atom core), while we can see the scaled value for core cycles is 160,444,092 and the percentage is 0.47%.

perf stat -e cycles -- taskset -c 16 ./triad_loop

As previous, two events are created.

.ft C

perf_event_attr:

size 120

config 0x400000000

sample_type IDENTIFIER

read_format TOTAL_TIME_ENABLED|TOTAL_TIME_RUNNING

disabled 1

inherit 1

enable_on_exec 1

exclude_guest 1

.ft

and

.ft C

perf_event_attr:

size 120

config 0x800000000

sample_type IDENTIFIER

read_format TOTAL_TIME_ENABLED|TOTAL_TIME_RUNNING

disabled 1

inherit 1

enable_on_exec 1

exclude_guest 1

.ft

Performance counter stats for 'taskset -c 16 ./triad_loop':

233,066,666 cpu_core/cycles/ (0.43%)

604,097,080 cpu_atom/cycles/ (99.57%)

perf-record:

If there is no -e specified in perf record, on hybrid platform, it creates two default cycles and adds them to event list. One is for core, the other is for atom.

perf-stat:

If there is no -e specified in perf stat, on hybrid platform, besides of software events, following events are created and added to event list in order.

cpu_core/cycles/, cpu_atom/cycles/, cpu_core/instructions/, cpu_atom/instructions/, cpu_core/branches/, cpu_atom/branches/, cpu_core/branch-misses/, cpu_atom/branch-misses/

Of course, both perf-stat and perf-record support to enable hybrid event with a specific pmu.

e.g. perf stat -e cpu_core/cycles/ perf stat -e cpu_atom/cycles/ perf stat -e cpu_core/r1a/

perf stat -e cpu_atom/L1-icache-loads/ perf stat -e cpu_core/cycles/,cpu_atom/instructions/

perf stat -e {cpu_core/cycles/,cpu_core/instructions/}

But {cpu_core/cycles/,cpu_atom/instructions/} will return warning and disable grouping,

because the pmus in group are not matched (cpu_core vs. cpu_atom).

SEE ALSO

perf-stat(1), perf-list(1), perf-intel-pt(1)