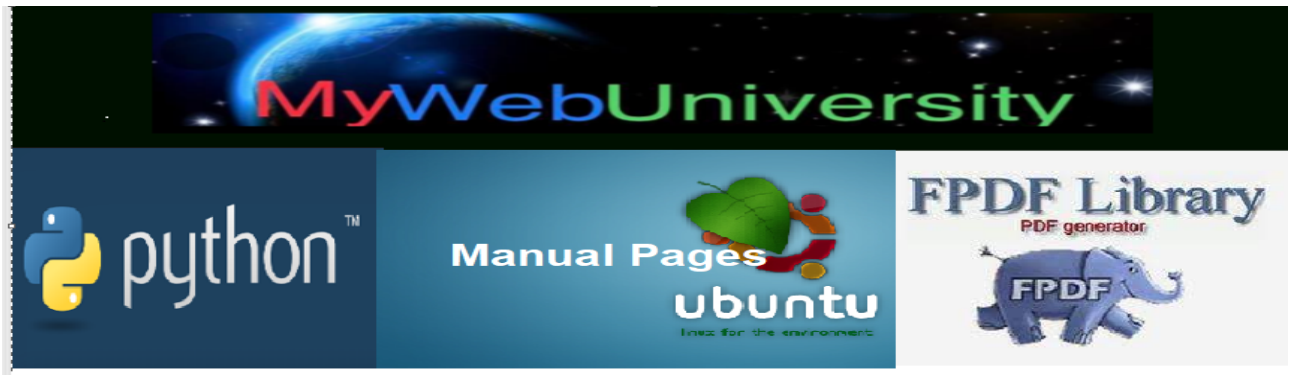

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man perf-sched.1

PERF-SCHED(1) perf Manual PERF-SCHED(1)

NAME

perf-sched - Tool to trace/measure scheduler properties (latencies)

SYNOPSIS

perf sched {record|latency|map|replay|script|timehist|stats}

DESCRIPTION

There are several variants of perf sched:

'perf sched record <command>' to record the scheduling events

of an arbitrary workload.

'perf sched latency' to report the per task scheduling latencies

and other scheduling properties of the workload.

Example usage:

perf sched record -- sleep 1

perf sched latency

Task | Runtime ms | Count | Avg delay ms | Max delay ms | Max delay start | Max delay end

|

---

perf:(2) | 2.804 ms | 66 | avg: 0.524 ms | max: 1.069 ms | max start: 254752.314960 s | max end: 254752.316029 s

NetworkManager:1343 | 0.372 ms | 13 | avg: 0.008 ms | max: 0.013 ms | max start: 254751.551153 s | max end: 254751.551166 s

kworker/1:2-xfs:4649 | 0.012 ms | 1 | avg: 0.008 ms | max: 0.008 ms | max start: 254751.519807 s | max

end: 254751.519815 s

kworker/3:1-xfs:388 | 0.011 ms | 1 | avg: 0.006 ms | max: 0.006 ms | max start: 254751.519809 s | max end: 254751.519815 s

sleep:147736 | 0.938 ms | 3 | avg: 0.006 ms | max: 0.007 ms | max start: 254751.313817 s | max end: 254751.313824 s

It shows Runtime(time that a task spent actually running on the CPU),
Count(number of times a delay was calculated) and delay(time that a
task was ready to run but was kept waiting).
Tasks with the same command name are merged and the merge count is
given within (), However if -p option is used, pid is mentioned.
'perf sched script' to see a detailed trace of the workload that
was recorded (aliased to 'perf script' for now).
'perf sched replay' to simulate the workload that was recorded
via perf sched record. (this is done by starting up mockup threads
that mimic the workload based on the events in the trace. These
threads can then replay the timings (CPU runtime and sleep patterns)
of the workload as it occurred when it was recorded - and can repeat
it a number of times, measuring its performance.)
'perf sched map' to print a textual context-switching outline of
workload captured via perf sched record. Columns stand for
individual CPUs, and the two-letter shortcuts stand for tasks that
are running on a CPU. A '*' denotes the CPU that had the event, and
a dot signals an idle CPU.
'perf sched timehist' provides an analysis of scheduling events.
Example usage:

```
perf sched record -- sleep 1
perf sched timehist
```

By default it shows the individual schedule events, including the wait
time (time between sched-out and next sched-in events for the task), the
task scheduling delay (time between runnable and actually running) and
run time for the task:

```
time    cpu  task name                wait time  sch delay   run time
                [tid/pid]                (msec)     (msec)     (msec)
```

```
 -------------- ------  --------------------  ---------  ---------  ---------
   79371.874569 [0011]  gcc[31949]              0.014      0.000      1.148
   79371.874591 [0010]  gcc[31951]              0.000      0.000      0.024
   79371.874603 [0010]  migration/10[59]        3.350      0.004      0.011
   79371.874604 [0011]  <idle>                  1.148      0.000      0.035
   79371.874723 [0005]  <idle>                  0.016      0.000      1.383
   79371.874746 [0005]  gcc[31949]              0.153      0.078      0.022
...
Times are in msec.usec.
'perf sched stats {record | report | diff} <command>' to capture, report the diff
in schedstat counters and show the difference between perf sched stats report
respectively. schedstat counters which are present in the linux kernel and are
exposed through the file ``/proc/schedstat``. These counters are enabled or disabled
via the sysctl governed by the file ``/proc/sys/kernel/sched_schedstats``. These
counters accounts for many scheduler events such as ``schedule()`` calls, load-balancing
events, ``try_to_wakeup()`` call among others. This is useful in understanding the
scheduler behavior for the workload.
Note: The tool will not give correct results if there is topological reordering or
      online/offline of cpus in between capturing snapshots of `/proc/schedstat`.
Example usage:
   perf sched stats record -- sleep 1
   perf sched stats report
   perf sched stats diff
A detailed description of the schedstats can be found in the Kernel Documentation:
https://www.kernel.org/doc/html/latest/scheduler/sched-stats.html
The result can be interpreted as follows:
The `perf sched stats report` starts with description of the columns present in
the report. These column names are given before cpu and domain stats to improve
the readability of the report.
----------------------------------------------------------------------------------------------------
DESC                    -> Description of the field
COUNT                   -> Value of the field
PCT_CHANGE              -> Percent change with corresponding base value
```

AVG_JIFFIES -> Avg time in jiffies between two consecutive occurrence of event

---

Next is the total profiling time in terms of jiffies:

Time elapsed (in jiffies) : 2323

Next is CPU scheduling statistics. These are simple diffs of /proc/schedstat CPU lines
along with description. The report also prints % relative to base stat.
In the example below, schedule() left the CPU0 idle 36.58% of the time. 0.45% of total
try_to_wake_up() was to wakeup local CPU. And, the total waittime by tasks on CPU0 is
48.70% of the total runtime by tasks on the same CPU.

CPU 0

DESC COUNT PCT_CHANGE

---

yld_count : 0

array_exp : 0

sched_count : 402267

sched_goidle : 147161 (36.58%)

ttwu_count : 236309

ttwu_local : 1062 (0.45%)

rq_cpu_time : 7083791148

run_delay : 3449973971 (48.70%)

pcount : 255035

Next is load balancing statistics. For each of the sched domains
(eg: `SMT`, `MC`, `DIE`...), the scheduler computes statistics under
the following three categories:

1) Idle Load Balance: Load balancing performed on behalf of a long
idling CPU by some other CPU.

2) Busy Load Balance: Load balancing performed when the CPU was busy.

3) New Idle Balance : Load balancing performed when a CPU just became

idle.

Under each of these three categories, sched stats report provides

different load balancing statistics. Along with direct stats, the

report also contains derived metrics prefixed with *. Example:

CPU 0, DOMAIN SMT CPUS 0,64

DESC COUNT AVG_JIFFIES

---------- <Category busy> ----------

busy_lb_count : 136 $ 17.08 $

busy_lb_balanced : 131 $ 17.73 $

busy_lb_failed : 0 $ 0.00 $

busy_lb_imbalance_load : 58

busy_lb_imbalance_util : 0

busy_lb_imbalance_task : 0

busy_lb_imbalance_misfit : 0

busy_lb_gained : 7

busy_lb_hot_gained : 0

busy_lb_nobusyq : 2 $ 1161.50 $

busy_lb_nobusyg : 129 $ 18.01 $

*busy_lb_success_count : 5

*busy_lb_avg_pulled : 1.40

---------- <Category idle> ----------

idle_lb_count : 449 $ 5.17 $

idle_lb_balanced : 382 $ 6.08 $

idle_lb_failed : 3 $ 774.33 $

idle_lb_imbalance_load : 0

idle_lb_imbalance_util : 0

idle_lb_imbalance_task : 71

idle_lb_imbalance_misfit : 0

idle_lb_gained : 67

idle_lb_hot_gained : 0

idle_lb_nobusyq : 0 $ 0.00 $

idle_lb_nobusyg : 382 $ 6.08 $

*idle_lb_success_count : 64

*idle_lb_avg_pulled : 1.05

---------- <Category newidle> ----------

newidle_lb_count : 30471 $ 0.08 $

newidle_lb_balanced : 28490 $ 0.08 $

newidle_lb_failed : 633 $ 3.67 $

newidle_lb_imbalance_load : 0

newidle_lb_imbalance_util : 0

newidle_lb_imbalance_task : 2040

newidle_lb_imbalance_misfit : 0

newidle_lb_gained : 1348

newidle_lb_hot_gained : 0

newidle_lb_nobusyq : 6 $ 387.17 $

newidle_lb_nobusyg : 26634 $ 0.09 $

*newidle_lb_success_count : 1348

*newidle_lb_avg_pulled : 1.00

Consider following line:

newidle_lb_balanced : 28490 $ 0.08 $

While profiling was active, the load-balancer found 28490 times the load
needs to be balanced on a newly idle CPU 0. Following value encapsulated
inside $ is average jiffies between two events (2323 / 28490 = 0.08).

Next are active_load_balance() stats. alb did not trigger while the
profiling was active, hence it's all 0s.

---------- <Category active_load_balance()> ----------

alb_count : 0

alb_failed : 0

alb_pushed : 0

Next are sched_balance_exec() and sched_balance_fork() stats. They are
not used but we kept it in RFC just for legacy purpose. Unless opposed,
we plan to remove them in next revision.

Next are wakeup statistics. For every domain, the report also shows task-wakeup statistics. Example:

```
---------------------------------------- <Wakeup Info> ----------------------------------------
ttwu_wake_remote                                :       1590
ttwu_move_affine                                :         84
ttwu_move_balance                               :          0
```

Same set of stats are reported for each CPU and each domain level.

How to interpret the diff

~~~~~~~~~~~~~~~~~~~~~~~~~

The `perf sched stats diff` will also start with explaining the columns present in the diff. Then it will show the diff in time in terms of jiffies. The order of the values depends on the order of input data files. It will take `perf.data.old` and `perf.data` respectively as the defaults for comparison. Example:

```
Time elapsed (in jiffies)                       :       2009,       2001
```

Below is the sample representing the difference in cpu and domain stats of two runs. Here third column or the values enclosed in `|...|` shows the percent change between the two. Second and fourth columns shows the side-by-side representions of the corresponding fields from `perf sched stats report`.

CPU <ALL CPUS SUMMARY>

```
DESC                                    COUNT1      COUNT2   PCT_CHANG>
```

```
yld_count                               :          0,          0 |     0.00>
array_exp                               :          0,          0 |     0.00>
sched_count                             :     528533,     412573 |   -21.94>
sched_goidle                            :     193426,     146082 |   -24.48>
ttwu_count                              :     313134,     385975 |    23.26>
```
~~~~~~~~~~~~~~~~~~~~~~~~~

ttwu_local : 1126, 1282 | 13.85>

rq_cpu_time : 8257200244, 8301250047 | 0.53>

run_delay : 4728347053, 3997100703 | -15.47>

pcount : 335031, 266396 | -20.49>

Below is the sample of domain stats diff:

CPU <ALL CPUS SUMMARY>, DOMAIN SMT

DESC COUNT1 COUNT2 PCT_CHANG>

---------- <Category busy> ----------

busy_lb_count : 122, 80 | -34.43>

busy_lb_balanced : 115, 76 | -33.91>

busy_lb_failed : 1, 3 | 200.00>

busy_lb_imbalance_load : 35, 49 | 40.00>

busy_lb_imbalance_util : 0, 0 | 0.00>

busy_lb_imbalance_task : 0, 0 | 0.00>

busy_lb_imbalance_misfit : 0, 0 | 0.00>

busy_lb_gained : 7, 2 | -71.43>

busy_lb_hot_gained : 0, 0 | 0.00>

busy_lb_nobusyq : 0, 0 | 0.00>

busy_lb_nobusyg : 115, 76 | -33.91>

*busy_lb_success_count : 6, 1 | -83.33>

*busy_lb_avg_pulled : 1.17, 2.00 | 71.43>

---------- <Category idle> ----------

idle_lb_count : 568, 620 | 9.15>

idle_lb_balanced : 462, 449 | -2.81>

idle_lb_failed : 11, 21 | 90.91>

idle_lb_imbalance_load : 0, 0 | 0.00>

idle_lb_imbalance_util : 0, 0 | 0.00>

idle_lb_imbalance_task : 115, 189 | 64.35>

idle_lb_imbalance_misfit : 0, 0 | 0.00>

idle_lb_gained : 103, 169 | 64.08>

```
idle_lb_hot_gained                         :          0,          0 |     0.00>
idle_lb_nobusyq                            :          0,          0 |     0.00>
idle_lb_nobusyg                            :        462,        449 |    -2.81>
*idle_lb_success_count                     :         95,        150 |    57.89>
*idle_lb_avg_pulled                        :       1.08,       1.13 |     3.92>
----------------------------------------- <Category newidle> -----------------------------------------
newidle_lb_count                           :      16961,       3155 |   -81.40>
newidle_lb_balanced                        :      15646,       2556 |   -83.66>
newidle_lb_failed                          :        397,        142 |   -64.23>
newidle_lb_imbalance_load                  :          0,          0 |     0.00>
newidle_lb_imbalance_util                  :          0,          0 |     0.00>
newidle_lb_imbalance_task                  :       1376,        655 |   -52.40>
newidle_lb_imbalance_misfit                :          0,          0 |     0.00>
newidle_lb_gained                          :        917,        457 |   -50.16>
newidle_lb_hot_gained                      :          0,          0 |     0.00>
newidle_lb_nobusyq                         :          3,          1 |   -66.67>
newidle_lb_nobusyg                         :      14480,       2103 |   -85.48>
*newidle_lb_success_count                  :        918,        457 |   -50.22>
*newidle_lb_avg_pulled                     :       1.00,       1.00 |     0.11>
---------------------------------- <Category active_load_balance()> ----------------------------------
alb_count                                  :          0,          1 |     0.00>
alb_failed                                 :          0,          0 |     0.00>
alb_pushed                                 :          0,          1 |     0.00>
---------------------------------- <Category sched_balance_exec()> ----------------------------------
sbe_count                                  :          0,          0 |     0.00>
sbe_balanced                               :          0,          0 |     0.00>
sbe_pushed                                 :          0,          0 |     0.00>
---------------------------------- <Category sched_balance_fork()> ----------------------------------
sbf_count                                  :          0,          0 |     0.00>
sbf_balanced                               :          0,          0 |     0.00>
sbf_pushed                                 :          0,          0 |     0.00>
----------------------------------------- <Wakeup Info> -----------------------------------------
ttwu_wake_remote                           :       2031,       2914 |    43.48>
```

ttwu_move_affine : 73, 124 | 69.86>

ttwu_move_balance : 0, 0 | 0.00>

OPTIONS

Applicable to {record|latency|map|replay|script}

-i, --input=<file>

Input file name. (default: perf.data unless stdin is a fifo)

-v, --verbose

Be more verbose. (show symbol address, etc)

-D, --dump-raw-trace=

Display verbose dump of the sched data.

-f, --force

Don?t complain, do it.

OPTIONS FOR PERF SCHED LATENCY

-C, --CPU <n>

CPU to profile on.

-p, --pids

latency stats per pid instead of per command name.

-s, --sort <key[,key2...]>

sort by key(s): runtime, switch, avg, max by default it?s sorted by "avg ,max ,switch ,runtime".

OPTIONS FOR PERF SCHED MAP

--compact

Show only CPUs with activity. Helps visualizing on high core count systems.

--cpus

Show just entries with activities for the given CPUs.

--color-cpus

Highlight the given cpus.

--color-pids

Highlight the given pids.

--task-name <task>

Map output only for the given task name(s). Separate the task names with a comma (without whitespace). The sched-out time is printed and is represented by *- for the

given task name(s). (- indicates other tasks while . is idle).

--fuzzy-name

Given task name(s) can be partially matched (fuzzy matching).

OPTIONS FOR PERF SCHED TIMEHIST

-k, --vmlinux=<file>

vmlinux pathname

--kallsyms=<file>

kallsyms pathname

-g, --call-graph

Display call chains if present (default on).

--max-stack

Maximum number of functions to display in backtrace, default 5.

-C=, --cpu=

Only show events for the given CPU(s) (comma separated list).

-p=, --pid=

Only show events for given process ID (comma separated list).

-t=, --tid=

Only show events for given thread ID (comma separated list).

-s, --summary

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

-S, --with-summary

Show all scheduling events followed by a summary by thread with min, max, and average run times (in sec) and relative stddev.

--symfs=<directory>

Look for files with symbols relative to this directory.

-V, --cpu-visual

Show visual aid for sched switches by CPU: i marks idle time, s are scheduler events.

-w, --wakeups

Show wakeup events.

-M, --migrations

Show migration events.

-n, --next

Show next task.

-I, --idle-hist

Show idle-related events only.

--time

Only analyze samples within given time window: <start>,<stop>. Times have the format seconds.microseconds. If start is not given (i.e., time string is ,x.y) then analysis starts at the beginning of the file. If stop time is not given (i.e, time string is x.y,) then analysis goes to end of file.

--state

Show task state when it switched out.

--show-prio

Show task priority.

--prio

Only show events for given task priority(ies). Multiple priorities can be provided as a comma-separated list with no spaces: 0,120. Ranges of priorities are specified with -: 120-129. A combination of both can also be provided: 0,120-129.

-P, --pre-migrations

Show pre-migration wait time. pre-migration wait time is the time spent by a task waiting on a runqueue but not getting the chance to run there and is migrated to a different runqueue where it is finally run. This time between sched_wakeup and migrate_task is the pre-migration wait time.

OPTIONS FOR PERF SCHED REPLAY

-r, --repeat <n>

repeat the workload n times (0: infinite). Default is 10.

SEE ALSO

perf-record(1)