perf Counting

03 Jul 2014

"I just want to know how many times this is called."... This is a question I'm often asking for kernel, device, and application events, and there is an efficient way to answer them: using Linux `perf stat`.

The `perf` tool (aka perf_events) has different modes of operation. My previous examples on CPU sampling, static tracepoints, and heat maps, used a mode perf_events calls "sampling", where a binary perf.data file is written containing event data. Another mode, "counting", summarizes events in-kernel and passes the summary to user space. This is more efficient, costing less overhead in terms of CPU and storage.

Process Counts

How many processes are being created and destroyed? Using `perf stat` to *count* the sched_process tracepoints for 5 seconds:

```
# perf stat -e 'sched:sched_process_*' -a sleep 5

 Performance counter stats for 'system wide':

                20      sched:sched_process_free                [100.00%]
                21      sched:sched_process_exit                [100.00%]
                37      sched:sched_process_wait                [100.00%]
                20      sched:sched_process_fork                [100.00%]
                41      sched:sched_process_exec                [100.00%]
                 0      sched:sched_process_hang

       5.001391328 seconds time elapsed
```

Neat. So there were 20 fork()s and 21 exit()s during 5 seconds. I used -a to match on all CPUs, and `sleep 5` as a dummy command to set the counting duration. You can use `-p PID` instead, to match a process, and skip the sleep command so that `perf` runs until Ctrl-C.

This works for any event (see `perf list`) including all static and dynamic tracepoints.

Syscall Counts

Here's syscalls:

```
# perf stat -e 'syscalls:sys_enter_*' -a sleep 5 | awk '$1'
# started on Wed Jul  2 23:39:50 2014
 Performance counter stats for 'system wide':
                60      syscalls:sys_enter_socket               [99.95%]
                68      syscalls:sys_enter_connect               [99.95%]
               148      syscalls:sys_enter_epoll_wait               [99.97%]
                 8      syscalls:sys_enter_statfs               [99.97%]
                18      syscalls:sys_enter_dup2                 [99.98%]
                18      syscalls:sys_enter_getcwd               [99.98%]
               155      syscalls:sys_enter_select               [99.98%]
               226      syscalls:sys_enter_poll                 [99.98%]
[...]
```

I used awk to strip out syscalls that had zero counts. If that doesn't work for you, add "-o /dev/stdout" to the `perf` command, which can be needed for older versions (eg, Linux 3.2). Another issue I've had with older versions is the need to increase the file descriptor limit (ulimit -n), when matching this many probes.

This one-liner can be a quick way to determine which syscalls are in use, before moving to using `perf record` for a closer look, eg, with arguments and stack traces.

Interval Summary

`perf stat` can also print an interval summary in more recent versions, using `-I` and a duration in milliseconds. For example, showing the per-second rate of context switches:

```
# perf stat -I 1000 -e sched:sched_switch -a sleep 5
#           time             counts unit events
     1.000205453                314      sched:sched_switch
     2.000456051                290      sched:sched_switch
     3.000644420                322      sched:sched_switch
     4.000836009                305      sched:sched_switch
     5.001022382                198      sched:sched_switch
     5.001442140                  7      sched:sched_switch
```

Nice. Again, this works for any event.

By CPU-ID

You can also decompose by CPU id:

```
# perf stat -A -e sched:sched_switch -a sleep 5

 Performance counter stats for 'system wide':

CPU0                    495      sched:sched_switch
CPU1                    632      sched:sched_switch

       5.001377216 seconds time elapsed
```

Filtering

Finally, --filter can be added to only increment the counter based on a boolean test. For example, counting read() syscalls where the requested size is greater than 4 Kbytes:

```
# perf stat -e syscalls:sys_enter_read --filter 'count > 4096' -a sleep 5

 Performance counter stats for 'system wide':

                 1      syscalls:sys_enter_read

       5.001407932 seconds time elapsed
```

Notice that I used a "count" variable. Where did that come from, and what else is there?

You can figure out many of the variables by reading the static tracepoint definitions in the kernel source (see the example in my static tracepoints post). Another way is to print their format files:

```
# cat /sys/kernel/debug/tracing/events/syscalls/sys_enter_read/format
name: sys_enter_read
ID: 509
format:
    field:unsigned short common_type;   offset:0;   size:2; signed:0;
    field:unsigned char common_flags;   offset:2;   size:1; signed:0;
    field:unsigned char common_preempt_count;   offset:3;   size:1; signed:0;
    field:int common_pid;   offset:4;   size:4; signed:1;

    field:int nr;   offset:8;   size:4; signed:1;
    field:unsigned int fd;  offset:16;  size:8; signed:0;
    field:char * buf;   offset:24;  size:8; signed:0;
    field:size_t count; offset:32;  size:8; signed:0;

print fmt: "fd: 0x%08lx, buf: 0x%08lx, count: 0x%08lx", ((unsigned long)(REC->fd)), ((unsigned lo
```

If it's a dynamic tracepoint, you can use `perf probe -V` to list available variables. I wish there was a similar option for static tracepoints, as well.

So that's a quick tour of basic perf_events counting capabilities. More is possible; see my perf_events page and the perf_events wiki.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).