How to use Perf in UNIX (Linux)

Mateusz Ferenc

2023
May

Introduction

Perf is a tool that uses the hardware PMU (Performance Measurement Unit) to count system statistics.
Using Perf, the user is able to obtain performance counters, tracepoints, kprobes and uprobes.

Performance counters are CPU hardware registers that count hardware events such as instructions executed, cache misses, or branch failures.
They provide a basis for profiling applications to trace dynamic control flow and identify hotspots.
Perf provides rich generalized abstractions over hardware-specific capabilities.
Among other things, it provides per-task, per-CPU, and per-workload counters, sampling on top of these, and source code event annotation.

A few preliminary clarifications:
Almost all of the commands below require administrative privileges [i.e., superuser, sudo].
Text enclosed in ']['' means it is optional, and the '|' character means the first or second option is possible.

1   How to setup

First of all Perf needs to be installed. (also you can check if it not already installed using: **which perf** or by simply typing **perf** in your command line to get perf help)

You can install Perf using your package menager.
Package name is ***linux-tools-*your kernel version****.

On Debian based systems:
apt install linux-tools-'uname -r'

When Perf is finally installed you should enable system profiling in kernel configuration.
To do this you need to edit in **/etc/sysctl.conf** file, value after **kernel.perf_event_paranoid** to **-1** .

Now you are able to use Perf properly.

2 Preparing the program for testing

You can measure system statistics using pre-existing trace points, but using custom trace points gives you more flexibility with Perf, resulting in better results.

To be able to use tracepoints you need to prepare you program (written in c/c++).
The preparation consists of compiling program using gcc with debug flag (like below).
gcc -g -o *<output-file-name> <file-name>*
"-g" flag used to produce debugging information in the operating system's native format

3 Creating and using tracepoints

Now that we have compiled program, we can create tracepoints according to functions or specified location in the code.

With the help of the following command, you can get functions that can be probed.
perf probe –funcs –exec *<executable-name>*

Also it is possible to list probe-able code lines of any function <function-name>
perf probe –line *<function-name>* –exec *<executable-name>*

With the information from the above commands, you can create your own trace points. To do this, you can use the following commands.

To create a trace point at the entry of the specified function (in this case probe-definition), use:
perf probe –exec *<executable-name> <probe-definition>*
Format of probing 'probe-definition' can be different, what affects where probe is placed:

$$\textbf{[EVENT=]FUNC[\%return | :RL]}$$

- **EVENT=** - Specify event name, optional. If specified, overrides the Perf naming convention.

- **FUNC** - function name to probe, moderatory.

- **%return** - place probe at exit of the function, optional.

- **:RL** - Relative line number from function input, obtained by using the –line switch with the above command, optional.

If '**EVENT=**' is not specified, Perf will create tracepoint (name) by using following convention:

$$\textbf{probe\_\textit{executable-name}:\textit{function-name}[\_\_\_return | :\textit{line-number}]}$$

otherwise:

$$\textbf{probe\_\textit{executable-name}:\textit{event-name}}$$

i.e.: probe_test:TestFunction___return, probe_test:TestEvent, probe_test:TestFunction:15

To test if newly created tracepoints, you can simply run Perf stat like below:
perf stat –event *<your-event-name>*
Output of Perf stat command should contain number how many times your event/events occured.

4   Recording and obtaining results

When you have finally prepared your tracepoints, it is time to get the results.

My example will show measurements using two self-defined tracepoints.
Because I need to show the time difference between tracepoints pointing to the entrance and exit of the function being measured, i.e. I was measuring how much time it takes to execute code in my function.

To do this, you must have at least two tracepoints and use the following command:
perf record –event <*your-event-entrance*> –event <*your-event-exit*> <*your-executable*>
i.e.: perf record -e probe_test:TestFunction -e probe_test:TestFunction___return ./test_executable

Now using Perf script, you can get data from Perf record:
perf script –ns –deltatime
The '–ns' flag is used to display time in nanosecond resolution, the '–deltatime' flag is used to calculate the difference (in time) between the occurrence of events.

5   How to interpret the output

Analyze file generated by Perf script

6   More about the tracepoints

adding leader event:
perf record –event <*your-event-entrance*>
redirect to file:
perf script –ns –deltatime > <*file-name*>

7   Adjusting hardware te get better results

Single core run (no context-switching)
Disable SpeedStep technology
Disable HyperThreading technology
Disable TurboBoost technology
Change CPU governor to performance
cpupower frequency-set –governor performance