

# How to profile your applications using the Linux perf tools

Baptiste Wicht — 2011-07-18 09:30 — 14 Comments

When an application encounters some performance issues, we have to find the code that causes the problem to optimize only what really matters.

To find the code we have to optimize, the profilers are really useful. In this post, we'll use the Linux perf tools to profile a simple C++ application.

The perf tools are integrated in the Linux kernel since the 2.6 version. The perf tools are based on the perf events subsystem. The perf profiler uses hardware counters to profile the application. The result of this profiler are really precise and because it is not doing instrumentation of the code, it is really fast.

## Installation

First of all, if it is not already done, you have to install the perf tools on your computer. On Ubuntu, you can just use apt-get to install it :

```
sudo apt-get install linux-tools
```

On the other systems, just use your favorite package manager to install the perf tools.

## Usage

To profile an application, you have to record information about an execution, for that, you just have to use perf record :

```
perf record program [program_options]
```

For example :

```
perf record eddic assembly.eddi
```

Once the execution is over, perf will gives you some information about the record, like:

```
[ perf record: Woken up 44 times to write data ]  
[ perf record: Captured and wrote 11.483 MB perf.data (-5017
```

If you see that the size of the perf.data is really small, generally for small execution time, you can configure the event period in order to have more information with the `--count=period` option using a small period. I usually uses 1000, but it can be useful to use a smaller number when the application has a short execution time.

Then, to see the list of the most costly functions, you just have to use perf report :

```
perf report
```

It will display a list of the most costly functions ordered by cost. Here is an example taken from one of my C++ applications (the length of the function names is reduced to fit the screen) here :

```

# Events: 374K cycles
#
# Overhead      Command      Shared Object
# .....
#
 87.58%      inlining  [vesafb]      [k] 0x
 83.27%      readelf  [vesafb]      [k] 0x
 41.40%      sh      [vesafb]      [k] 0x
 37.74%      inlining  libstdc++.so.6.0.14  [.] 0x
 12.49%      readelf  libc-2.13.so  [.] vf
  5.37%      inlining  inlining      [.] pa
  5.20%      readelf  libc-2.13.so  [.] _I
  4.50%      readelf  readelf       [.] 0x
  4.10%      inlining  libc-2.13.so  [.] _i
  4.01%      inlining  libc-2.13.so  [.] me
  3.80%      readelf  libc-2.13.so  [.] _
  2.58%      inlining  libc-2.13.so  [.] _
  1.86%      inlining  libc-2.13.so  [.] _I
  1.84%      inlining  inlining      [.] pa
  1.83%      readelf  libc-2.13.so  [.] _
  1.83%      inlining  libc-2.13.so  [.] _i
  1.77%      inlining  libc-2.13.so  [.] _
  1.50%      inlining  libc-2.13.so  [.] cf
  1.48%      inlining  libc-2.13.so  [.] _
  1.23%      inlining  inlining      [.] pa
  1.19%      inlining  libboost_graph.so.1.46.1  [.] ch
 1.17%      readelf  libc-2.13.so  [.] __dcgettext
 1.15%      inlining  libc-2.13.so  [.] _IO_getline_info_internal

```


For every functions, you have the information about the cost of the function, the command used to launch it and the shared object in which the function is located. You can navigate through the list like in more utility.

This tool can be really useful to see which functions is interesting to optimize in order to increase the overall performance of the application.

For more information about the perf tools, you can read the perf wiki ([https://perf.wiki.kernel.org/index.php/Main\\_Page](https://perf.wiki.kernel.org/index.php/Main_Page))

In a future article, I will talk about another profiler, Callgrind.

C++ Linux Performances Tools

Contents © 2017 Baptiste Wicht (<mailto:baptistewicht@gmail.com>) - Powered by Nikola (<http://getnikola.com>) - License:  (<http://creativecommons.org/licenses/by/4.0/>)

How to profile your applications using the Linux p...

<https://baptiste-wicht.com/posts/2011/07/profile-a...>

Source (profile-applications-linux-perf-tools.wp)