Demystifying the Real-Time Linux Scheduling Latency (Artifact)

Daniel Bristot de Oliveira
Red Hat, Inc, Italy
bristot@redhat.com

Daniel Casini
Scuola Superiore Sant'Anna, Italy
daniel.casini@santannapisa.it

Rômulo Silva de Oliveira
Universidade Federal de Santa Catarina, Brazil
romulo.deoliveira@ufsc.br

Tommaso Cucinotta
Scuola Superiore Sant'Anna, Italy
tommaso.cucinotta@santannapisa.it

Abstract

The "Demystifying the Real-Time Linux Scheduling Latency" paper defines a safe bound for the real-time Linux scheduling latency. It also presents a tool kit that enables the measurements and analysis of the variables that compose the bond. The tool kit is used in the experimental section, performing the scheduling latency analyses on real platforms. This artifact provides the means to evaluate the tool kit and to reproduce the results of the experimental section.

2012 ACM Subject Classification Computer systems organization → Real-time operating systems

Keywords and phrases Real-time operating systems, Linux kernel, PREEMPT_RT, Scheduling latency

Digital Object Identifier 10.4230/DARTS.6.1.2

Funding This work has been partially supported by CAPES, The Brazilian Agency for Higher Education, project PrInt CAPES-UFSC "Automation 4.0."

Related Article Daniel Bristot de Oliveira, Daniel Casini, Rômulo Silva de Oliveira, and Tommaso Cucinotta, "Demystifying the Real-Time Linux Scheduling Latency", in 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), LIPIcs, Vol. 165, pp. 9:1–9:23, 2020.
`https://doi.org/10.4230/LIPIcs.ECRTS.2020.9`

Related Conference 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), July 7–10, 2020, Virtual Conference

1 Scope

The "Demystifying the Real-Time Linux Scheduling Latency" paper defines a safe bound for the real-time Linux scheduling latency. It also proposes a tool kit that enables the measurement of the variables by tracing the events of the PREEMPT_RT thread model [1] using an efficient approach for high-frequency trace analysis [2]. The tool kit is then used to perform the trace and analysis of the scheduling latency on real systems, demonstrating the practical benefits of the analysis. The results of the experiments are presented in Section 6 of the paper. This artifact provides the means to evaluate the tool kit and reproduce the results of the experimental section. It is composed of a self bootable operating system image, configured with the PREEMPT_RT kernel, that includes the mentioned tool kit. It also delivers a set of scripts that replicates the execution of all experiments presented in the paper, which can be used to validate the results.

2 Content

The artifact is delivered via a self-bootable operating system disk image. The image can be written on any HD or SSD with more than 16 GiB of space. By booting the system with the disk containing the image, a Fedora 31 will start with all set for the experiments.

On Linux, the image can be writing using the `dd` command as `root`. For example, given that a 16GB USB memory stick is presented as the `/dev/sdb` device, the following command will write the image `ecrts2020.img` to it:

```
# dd if=ecrts2020.img of=/dev/sdb
```

On Windows, it is possible to write the image using the `Win32 disk imager` tool.

Inside the home directory, there are some scripts that automate the execution of the experiments. The script `run_experiment.sh` is the principal of these scripts. Running it without any arguments will result in a five minutes trace and analysis. It accepts a different duration as its first argument (time in minutes). For instance, the command: `run_experiments.sh 15` will run the trace for 15 minutes. The script code has useful comments and can be used to learn how to use the tool. It is important to notice that, the longer the tracing section, the larger are the memory and disk requirements. Still, the experiments can run on any regular modern PC in 2020.

Other scripts then use the above-mentioned script to setup and start experiments presented in Section 6 of the paper. They are the following:

- `run_experiment_1a.sh`: the experiment 1.a in Figure 26.
- `run_experiment_1b.sh`: the experiment 1.b in Figure 26.
- `run_experiment_1c.sh`: the experiment 1.c in Figure 26.
- `run_experiment_2a.sh`: the experiment 2.a in Figure 26.
- `run_experiment_2b.sh`: the experiment 2.b in Figure 26.
- `run_experiment_2c.sh`: the experiment 2.c in Figure 26.
- `run_experiment_3a.sh`: the experiment 3.a in Figure 27.
- `run_experiment_3b.sh`: the experiment 3.b in Figure 27.
- `run_experiment_3c.sh`: the experiment 3.c in Figure 27.
- `run_experiment_4a.sh`: the experiment 4.a in Figure 27.

It is possible to find the kernel source and binary, the latency parser module source and binary, and the perf tool code that we extended and/or developed for the paper inside the `demo` folder.

The default user (ecrts) asks no password, and the root password is *ecrts2020*. Sudo asks no password for the ecrts user to run a command as root.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). Additional material can be found at: `https://bristot.me/demystifying-the-real-time-linux-latency/`.

4 Tested platforms

The self-bootable operating system image runs on any system with the *Intel 64 bits processors*. Before starting the experiments, to achieve similar results, the hardware must be tuned for low latency. As a general rule, the authors suggest to:

- disable hyperthreading;
- disable power management;
- disable frequency scaling;
- disable sources of SMI;
- disable deep idle states.
- disable turbo boost;
- setup all the BIOS for performance.

Beware that, because laptop processors are optimized for power savings, and often face thermal throttling, they are less deterministic, and this can influence the results (of any tool). So for the artifact evaluation, we suggest using desktops, workstations or servers.

It is also important to keep in mind that the results presented in the experimental section are only valid for that specific environment. Every and each hardware will present its own interrupt task set, and will force the threads to cross by different code sections that influence in the values. Moreover, all the results are based on observed values: it is not a goal of the paper to demonstrate finding the worst values. The main objective of this experimental study is to corroborate the practical applicability of the analysis tool.

5 License

The artifact is available under GPL v2 license.

6 MD5 sum of the artifact

3454c0893db32779b83f01ab8c65bac4

7 Size of the artifact

6.67 GB (compressed)

References

1 Daniel B. de Oliveira, Rômulo S. de Oliveira, and Tommaso Cucinotta. A thread synchronization model for the preempt_rt linux kernel. *Journal of Systems Architecture*, page 101729, 2020. `doi:10.1016/j.sysarc.2020.101729`.

2 Daniel Bristot de Oliveira, Tommaso Cucinotta, and Rômulo Silva de Oliveira. Efficient formal verification for the linux kernel. In *International Conference on Software Engineering and Formal Methods*, pages 315–332. Springer, 2019.