A Preliminary Assessment of the real-time capabilities of Real-Time Linux on Raspberry Pi 5

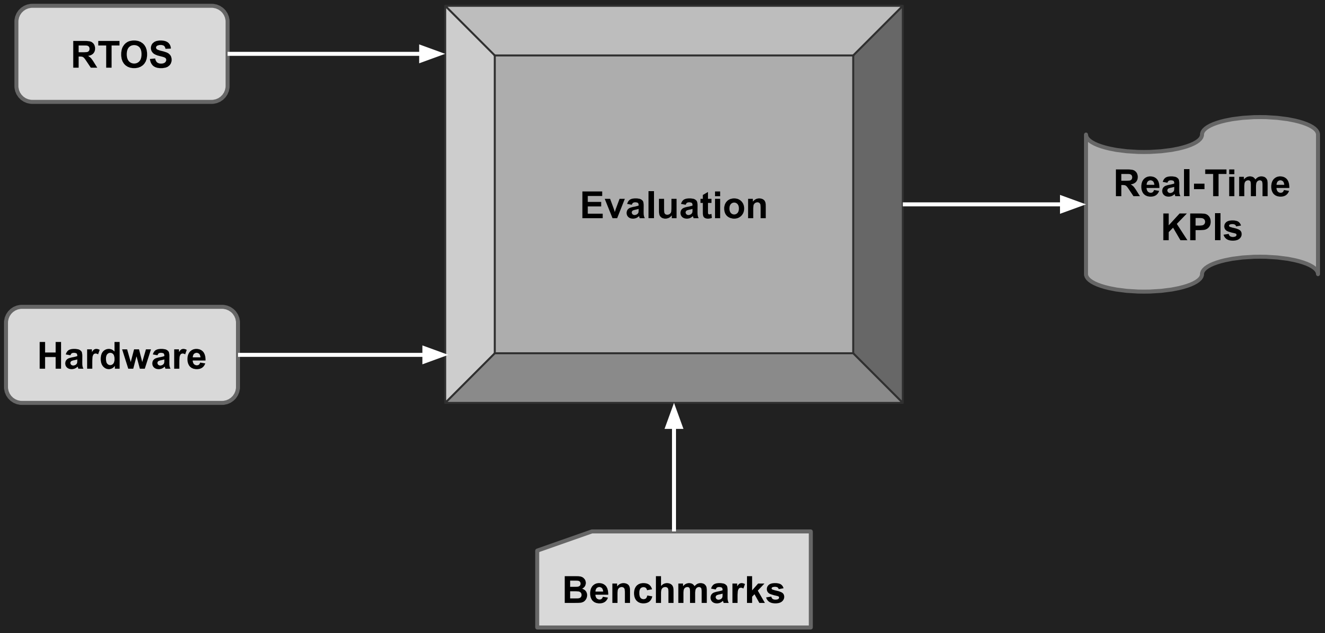
OSPERT 2024

Wannes Dewit, **Antonio Paolillo**, Joël Goossens

VUB SOFTWARE LANGUAGES LAB

ULB UNIVERSITÉ LIBRE DE BRUXELLES

Overall goal: design a methodology

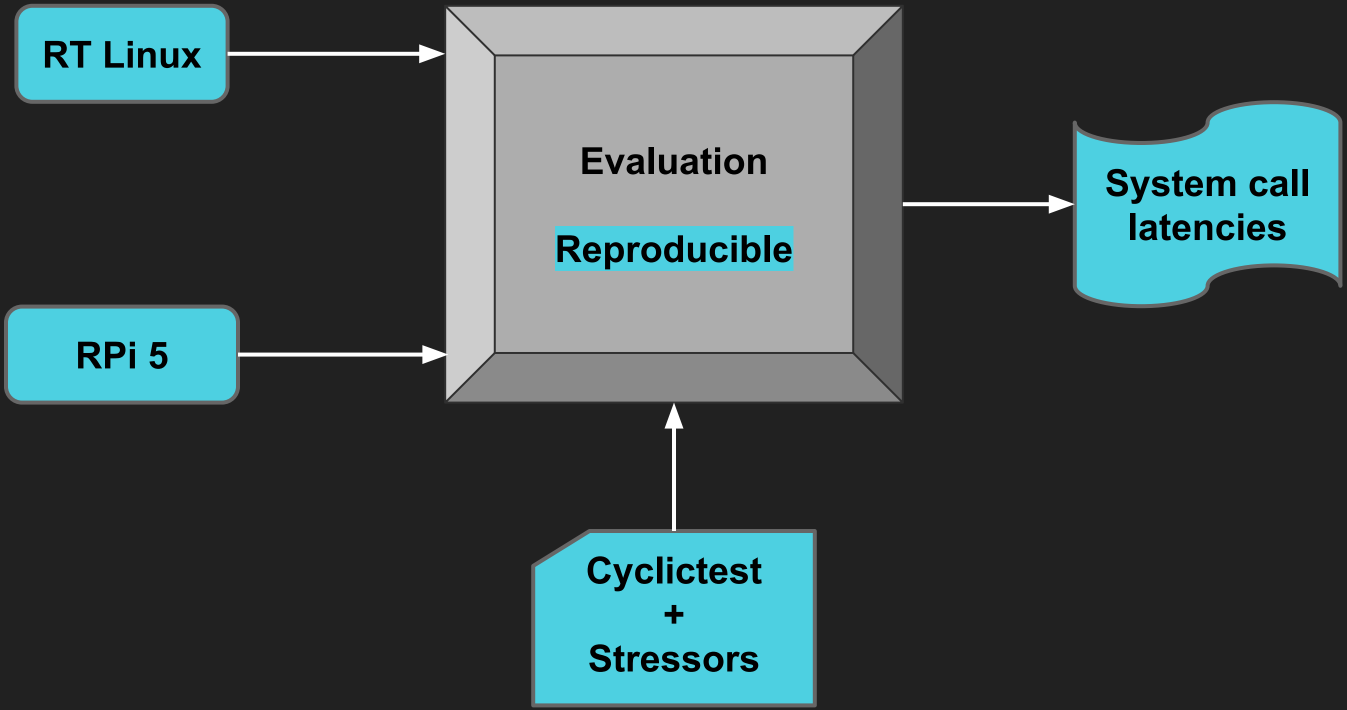


Open questions

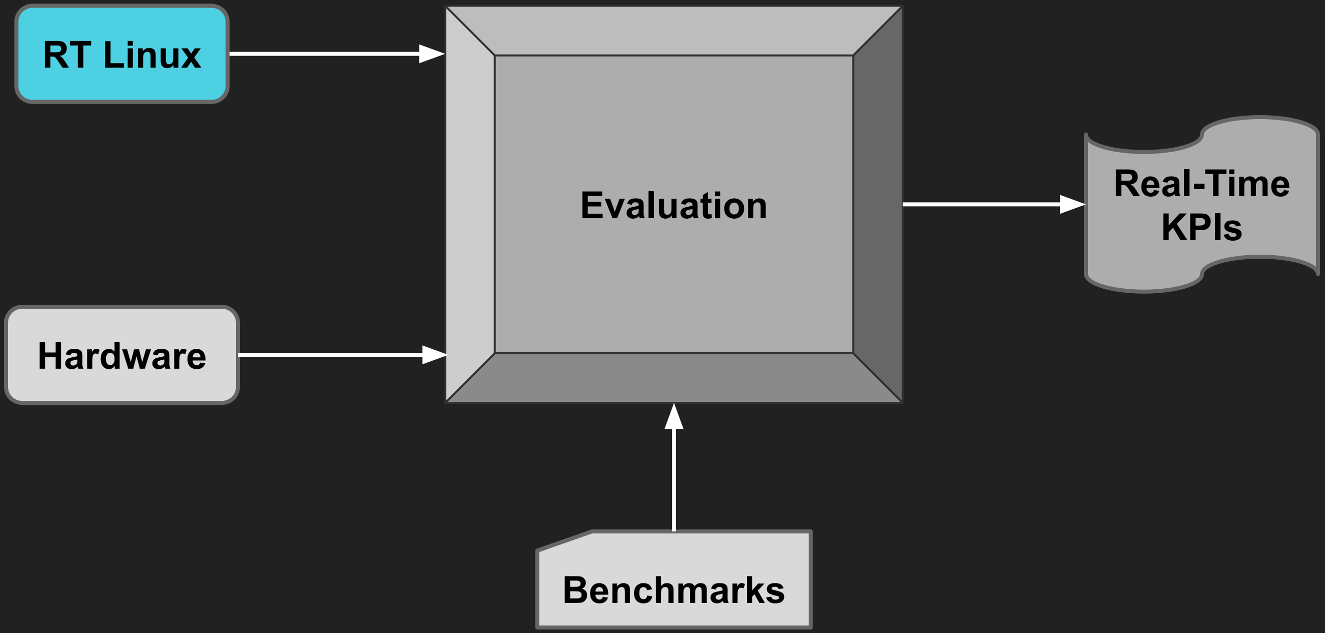
- **How to evaluate a given RTOS / HW couple?**
- **How to choose KPIs?**

  → that represent real-time behavior / capabilities
- **What benchmarks to choose / design for those?**

This paper

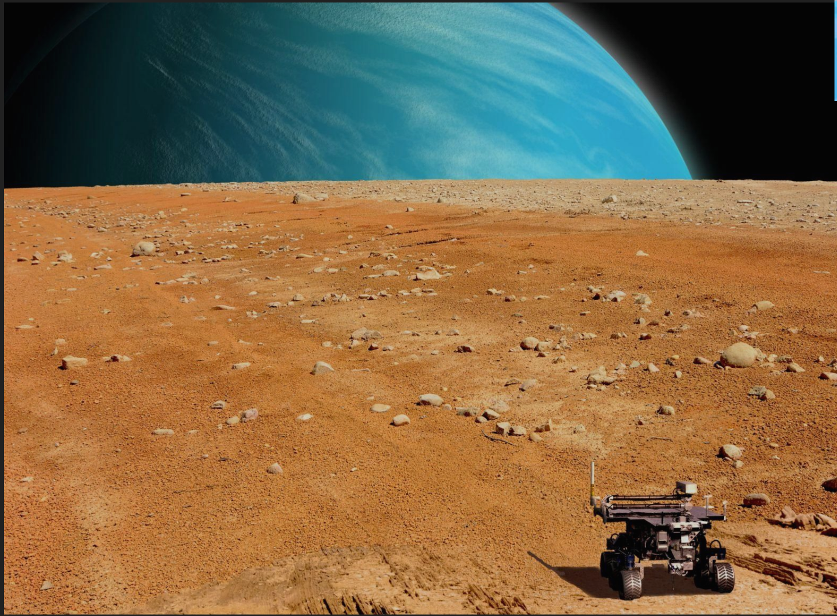

RT Linux
RPi 5
Evaluation
Reproducible
System call latencies
Cyclictest
+
Stressors

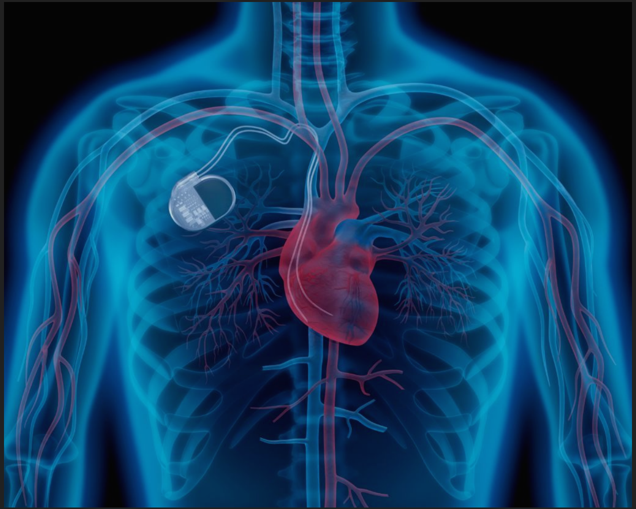
Real-Time Linux?



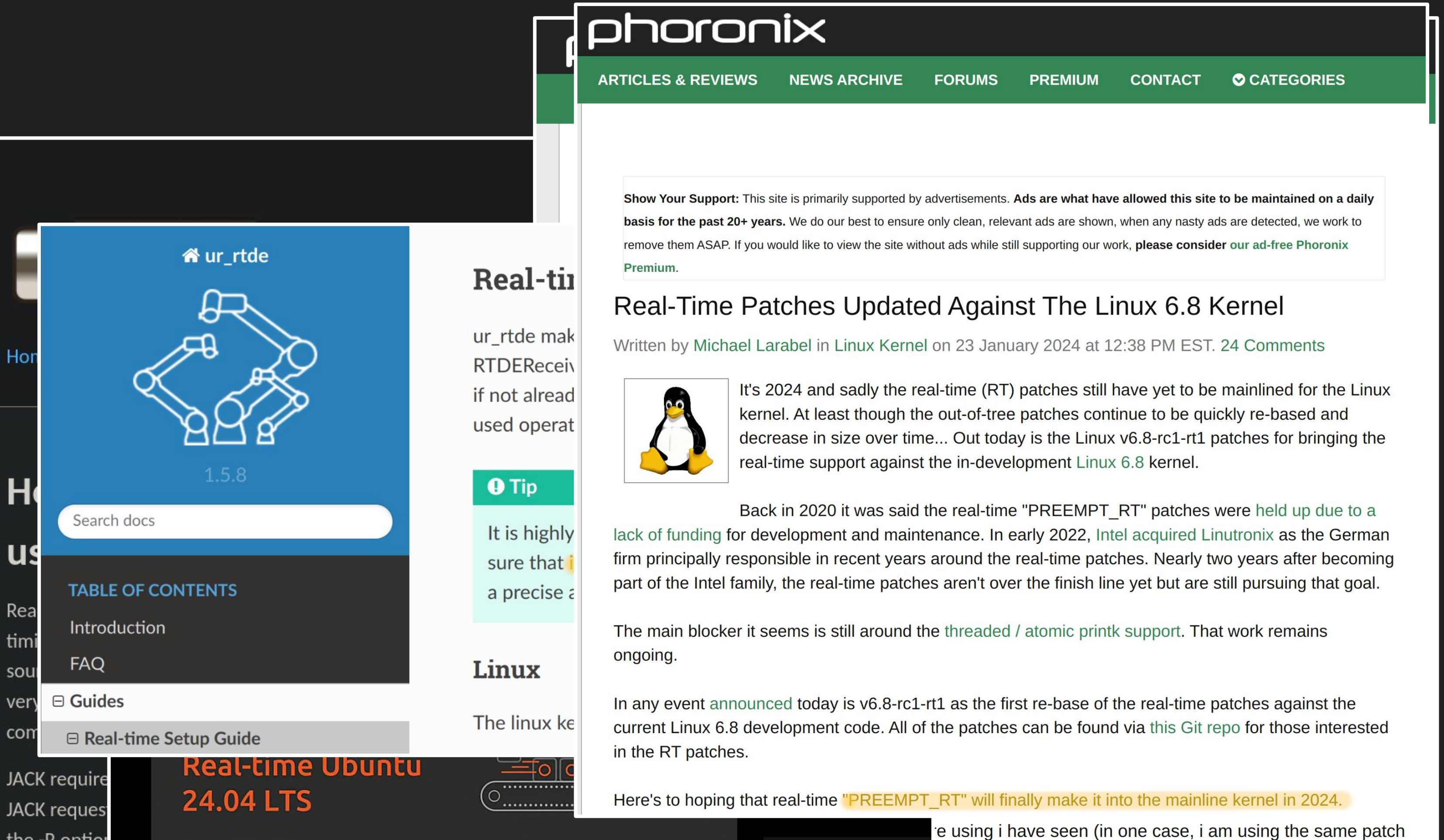
Context

→ Safety-critical environments





Context: besides safety-critical



ur_rtde

1.5.8

Search docs

TABLE OF CONTENTS

Introduction

FAQ

⊟ Guides

⊟ Real-time Setup Guide

Real-ti

ur_rtde mak
RTDEReceiv
if not alread
used operat

Tip

It is highly
sure that i
a precise a

Linux

The linux ke

phoronix

ARTICLES & REVIEWS NEWS ARCHIVE FORUMS PREMIUM CONTACT CATEGORIES

Show Your Support: This site is primarily supported by advertisements. **Ads are what have allowed this site to be maintained on a daily basis for the past 20+ years.** We do our best to ensure only clean, relevant ads are shown, when any nasty ads are detected, we work to remove them ASAP. If you would like to view the site without ads while still supporting our work, **please consider** our ad-free Phoronix Premium.

Real-Time Patches Updated Against The Linux 6.8 Kernel

Written by Michael Larabel in Linux Kernel on 23 January 2024 at 12:38 PM EST. 24 Comments

It's 2024 and sadly the real-time (RT) patches still have yet to be mainlined for the Linux kernel. At least though the out-of-tree patches continue to be quickly re-based and decrease in size over time... Out today is the Linux v6.8-rc1-rt1 patches for bringing the real-time support against the in-development Linux 6.8 kernel.

Back in 2020 it was said the real-time "PREEMPT_RT" patches were held up due to a lack of funding for development and maintenance. In early 2022, Intel acquired Linutronix as the German firm principally responsible in recent years around the real-time patches. Nearly two years after becoming part of the Intel family, the real-time patches aren't over the finish line yet but are still pursuing that goal.

The main blocker it seems is still around the threaded / atomic printk support. That work remains ongoing.

In any event announced today is v6.8-rc1-rt1 as the first re-base of the real-time patches against the current Linux 6.8 development code. All of the patches can be found via this Git repo for those interested in the RT patches.

Here's to hoping that real-time "PREEMPT_RT" will finally make it into the mainline kernel in 2024.

[redacted] e using i have seen (in one case, i am using the same patch

Implementation of Real-Time Linux?

* Deadlines

→ RT scheduling classes
→ SCHED_DEADLINE

* Determinism

→ not necessarily fast; worst-case scenario

→ RTlocks & critical sections

→ ISR, system calls & jitter

→ **PREEMPT_RT: to make the kernel “more” preemptive**

PREEMPT_RT improved the kernel also for non-RT users

- Generic Timekeeping
- High resolution timers
- Mutex infrastructure
- Generic interrupt handling infrastructure
- Priority inheritance for user space mutexes
- Preemptible and hierarchical RCU
- Threaded interrupt handlers
- Tracing infrastructure
- Lock dependency validator
- Rewrite of the CPU hotplug infrastructure
- Refactoring of the timer wheel
- Refactoring of high resolution timers
- ...

→ lots of upstreamed components already

Why Real-Time Linux?

Free and Open-Source

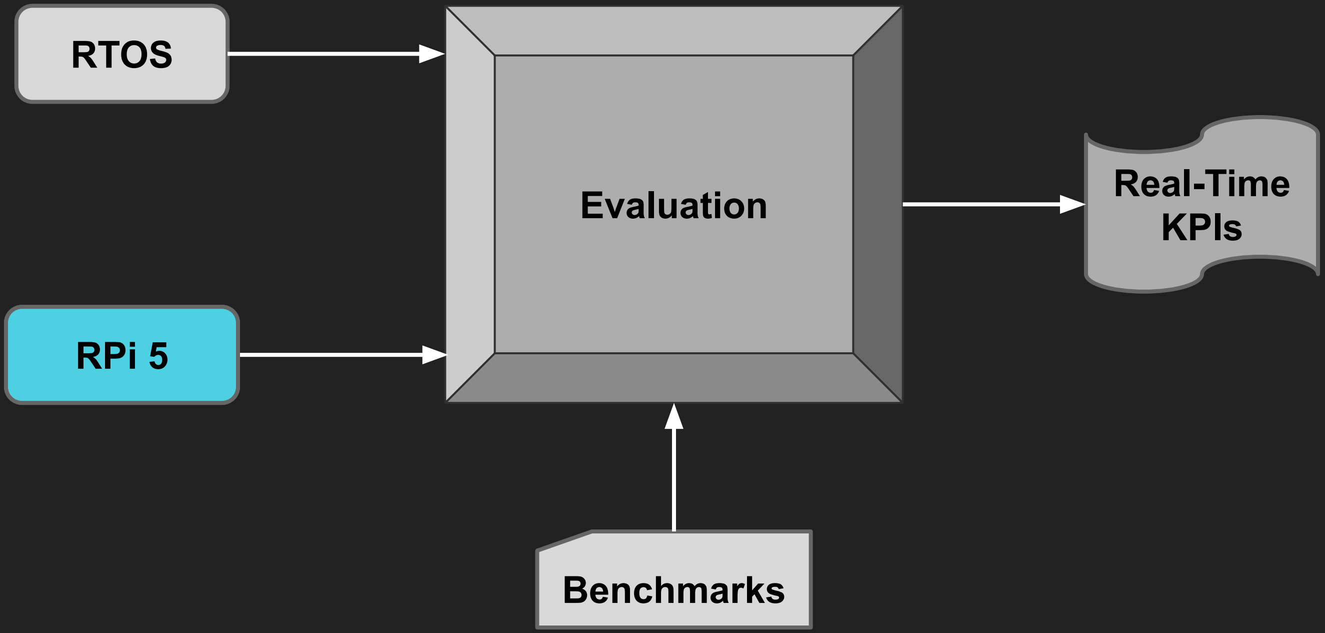
Device drivers

-> plug-and-play

Ease of use, very active community

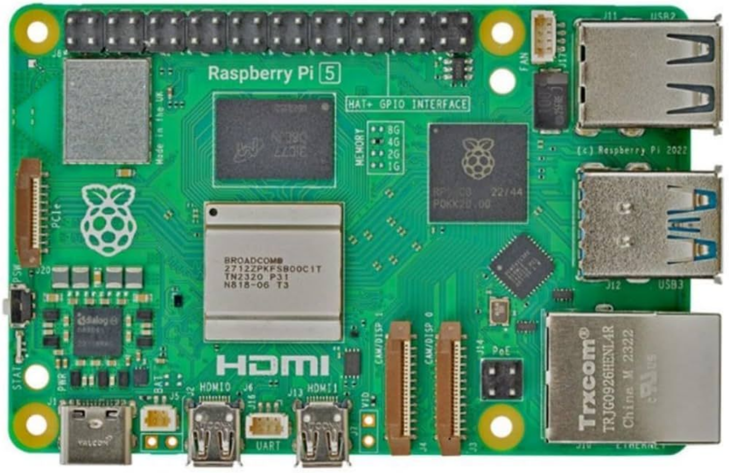
-> Easy adoption

Real-Time Linux?

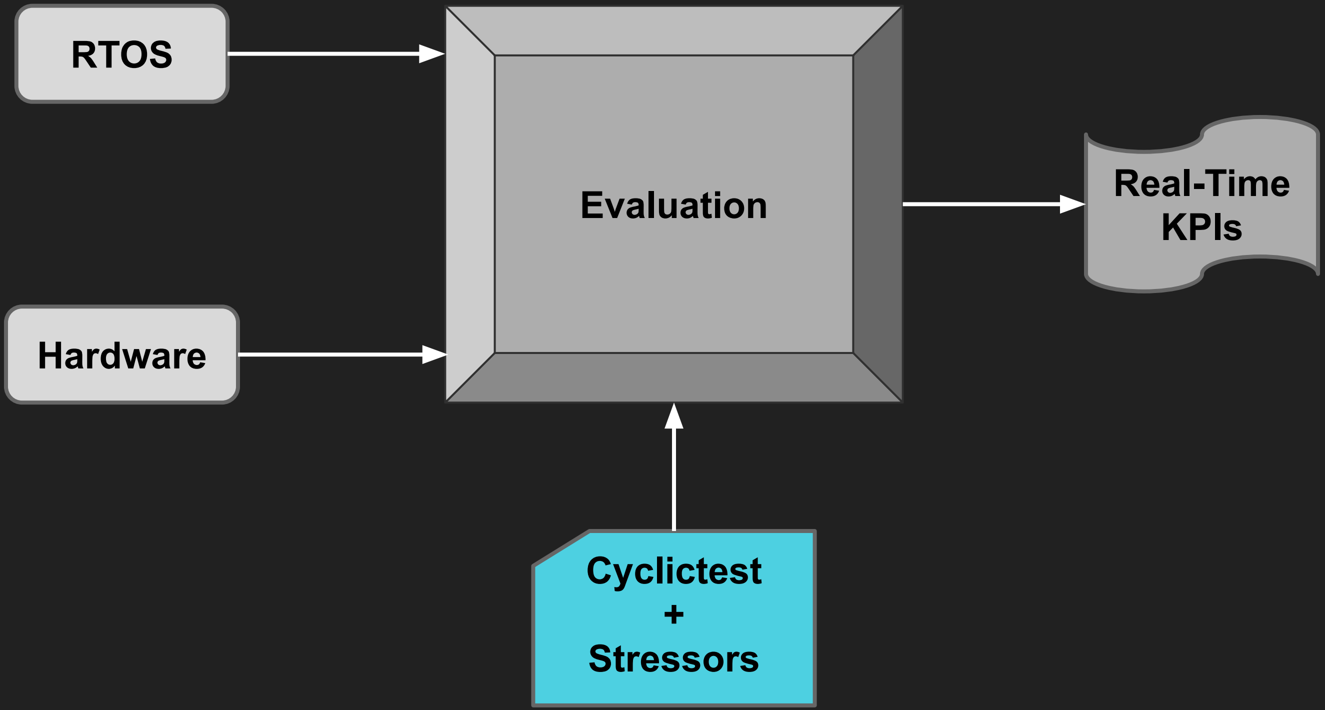


Raspberry Pi 5 Model B Rev 1.0

2.4GHz quad-core 64-bit Arm Cortex-A76 CPU



Real-Time Linux?



RTOS

Hardware

Evaluation

Real-Time KPIs

Cyclictest + Stressors

Why benchmarking?

* Size and complexity of Linux

* “Realistic” scenario

* Bridge between theory and practice

Evaluation methodology

#1: Stressing

-> CPU/IO load: *stress-ng*

-> Networking load: *iperf3*

#2: Measuring

-> Scheduling latency: *cyclictest*

#1.1: Stressing with *stress-ng*

```
sudo docker run --rm colinianking/stress-ng --all 1 -t1h 1> /dev/null &
```

#1.1: Stressing with *stress-ng*

```
sudo docker run --rm colinianking/stress-ng --all 1 -t1h 1> /dev/null &
```

#1.2: Stressing with *iperf3*

```
iperf3 -c <IP> -w 64K -P 100 -t 3800
```

#1.2: Stressing with *iperf3*

```
iperf3 -c <IP> -w 64K -P 100 -t 3800
```

#1.2: Stressing with *iperf3*

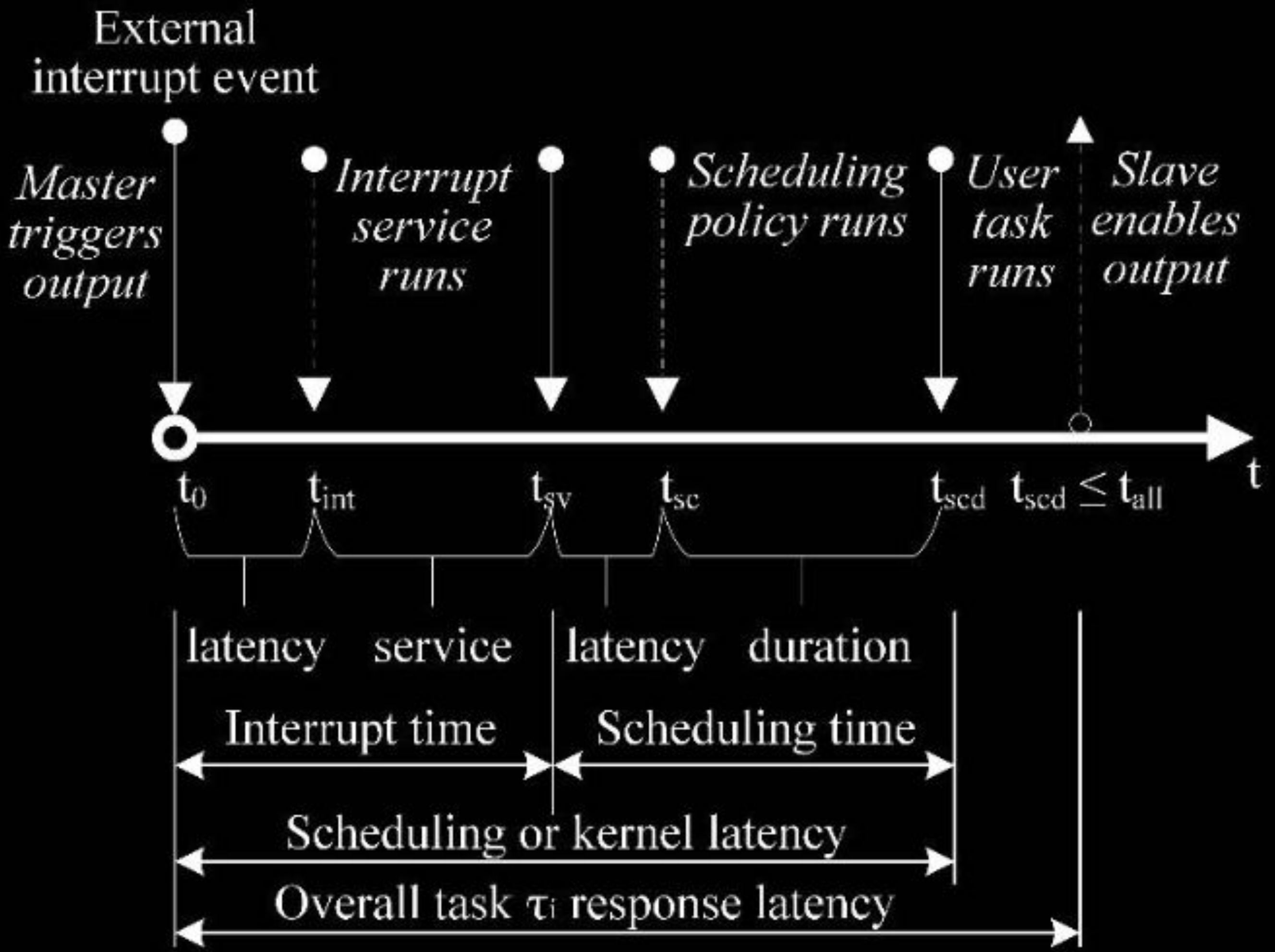
```
iperf3 -c <IP> -w 64K -P 100 -t 3800
```

#2: Measuring with *cyclictest*

Scheduling latency

“measures the difference between a thread's intended wake-up time and the time at which it actually wakes up”
-The Linux Foundation wiki

“provides an easy-to-interpret metric that reflects various sources of unpredictability as a single, opaque measure.”
-Cerqueira and Brandenburg


External interrupt event
Master triggers output
Interrupt service runs
Scheduling policy runs
User task runs
Slave enables output
t_0
t_{int}
t_{sv}
t_{sc}
t_{scd}
$t_{scd} \leq t_{all}$
t
latency
service
latency
duration
Interrupt time
Scheduling time
Scheduling or kernel latency
Overall task τ_i response latency

cf. Adam et al.

#2: Measuring with *cyclictest*

```
sudo cyclictest -vmn -i100 -p99 -t --duration=1h > cyclictest_<X>.txt
```

#2: Measuring with *cyclictest*

```
sudo cyclictest -vmn -i100 -p99 -t --duration=1h > cyclictest_<X>.txt
```

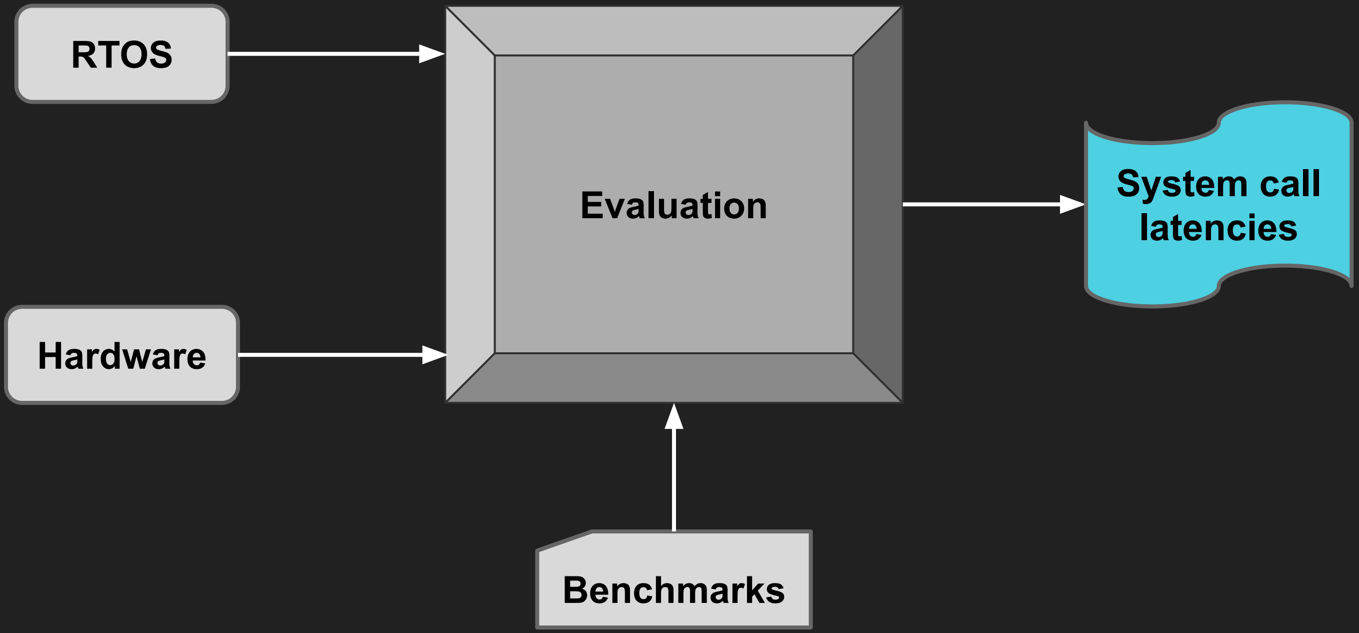
#2: Measuring with *cyclictest*

```
sudo cyclictest -vmn -i100 -p99 -t --duration=1h > cyclictest_<X>.txt
```

#2: Measuring with *cyclictest*

```
sudo cyclictest -vmn -i100 -p99 -t --duration=1h > cyclictest_<X>.txt
```

Real-Time Linux?



RTOS

Hardware

Evaluation

Benchmarks

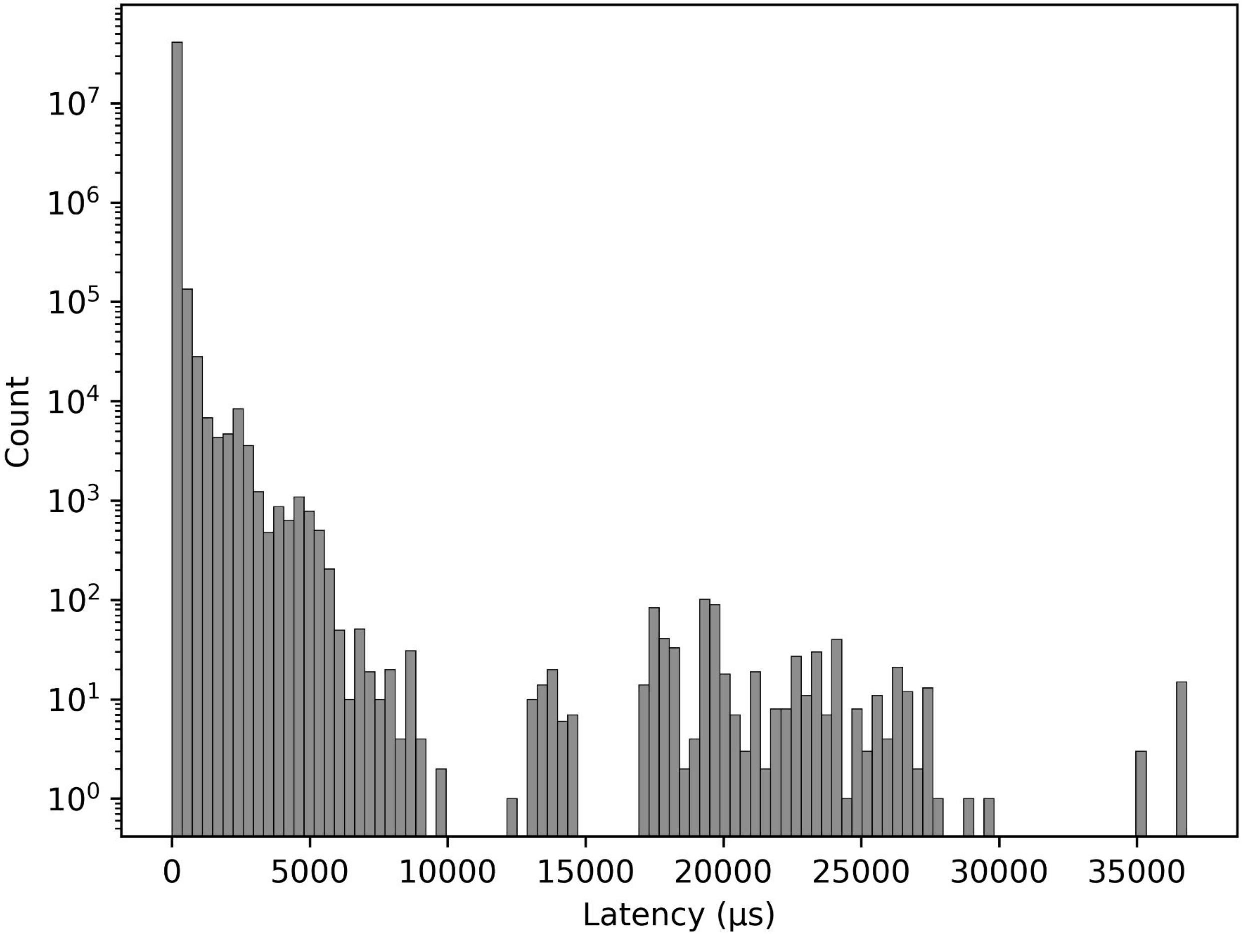
System call latencies

System metrics under the load

100% CPU load

~70% memory load

→ 4 RT tasks measuring the scheduling latency (*cyclictest*)

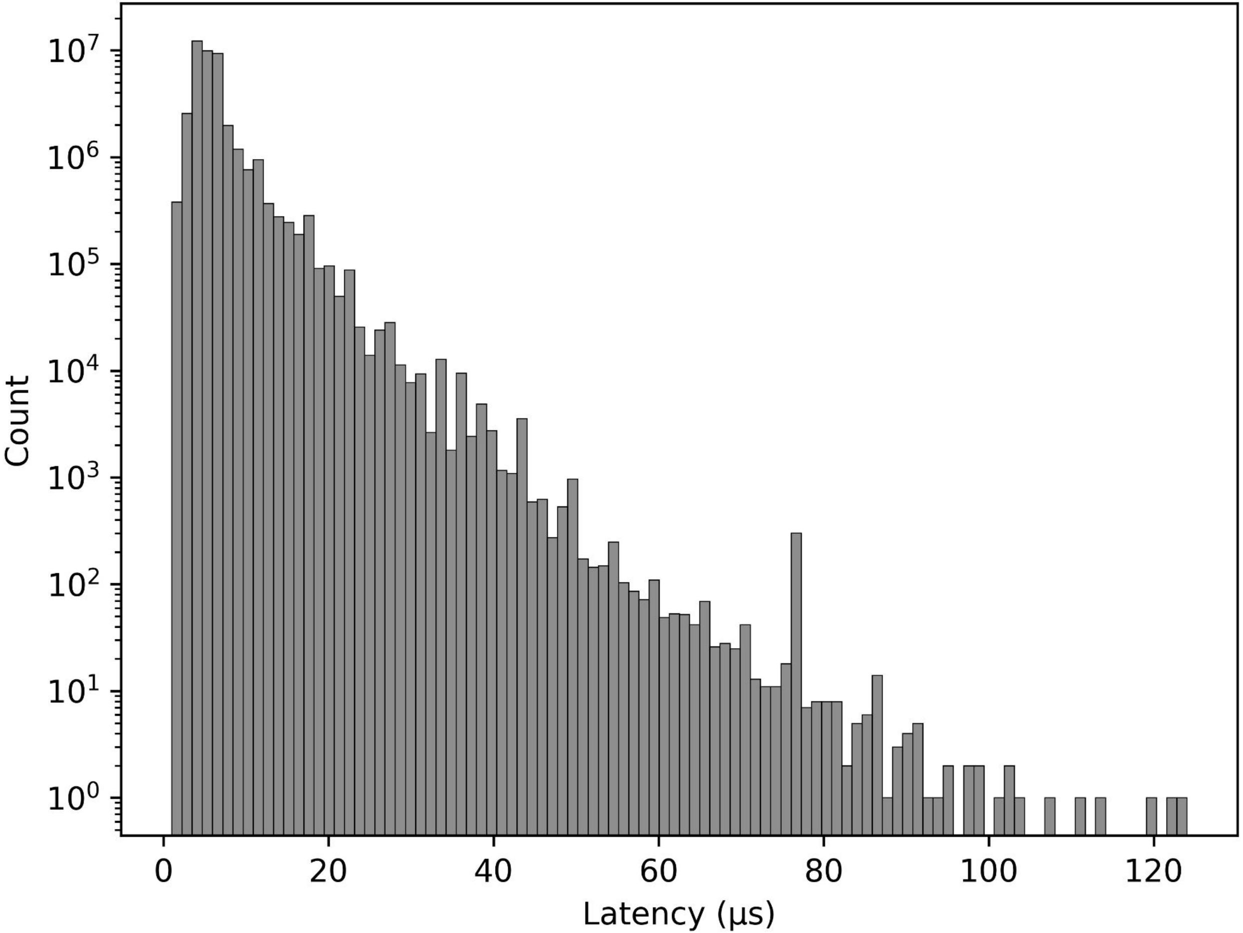


Cyclictest results

Raspberry Pi 5
Model B

Debian Linux kernel
6.6.21-v8-16k+

Max. latency:
36802µs



Cyclictest results

Raspberry Pi 5
Model B

Debian Linux kernel
6.6.21-rt25-v8-16k+

With PREEMPT_RT

Max. latency:
124μs

Benchmarking results (in µs)

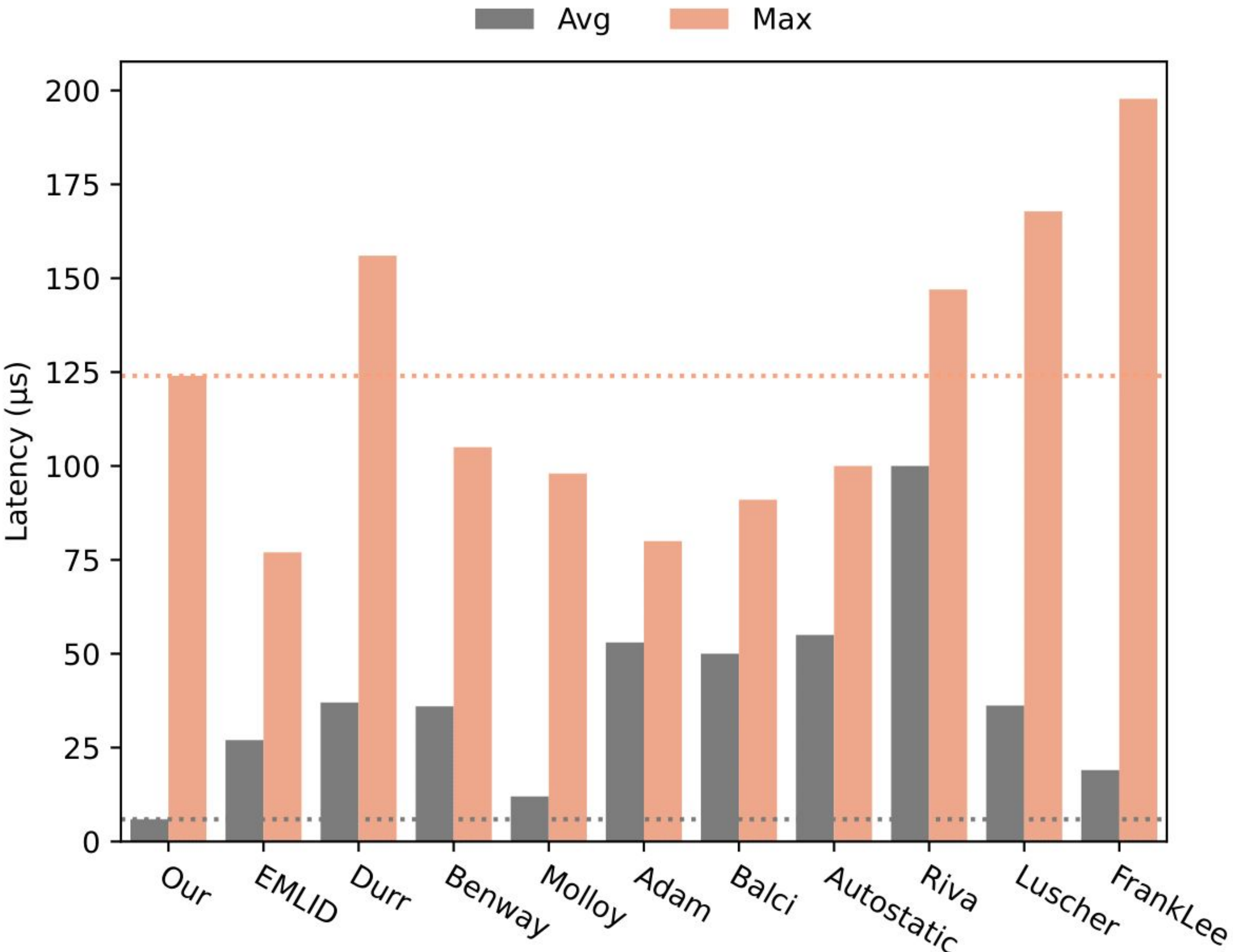
	Max	Min	Mean	St. Dev.
Custom kernel	36802	1	14.6942	122.08
RT kernel	124	1	5.9126	3.2484

Benchmarking results (in µs)

	Max	Min	Mean	St. Dev.
Custom kernel	36802	1	14.6942	122.08
RT kernel	124	1	5.9126	3.2484

Benchmarking results (in µs)

	Max	Min	Mean	St. Dev.
Custom kernel	36802	1	14.6942	122.08
RT kernel	124	1	5.9126	3.2484



Results from other studies

RT kernel (Min, Avg, Max)
1, 5.9, 124

Benchmarking results (in µs)

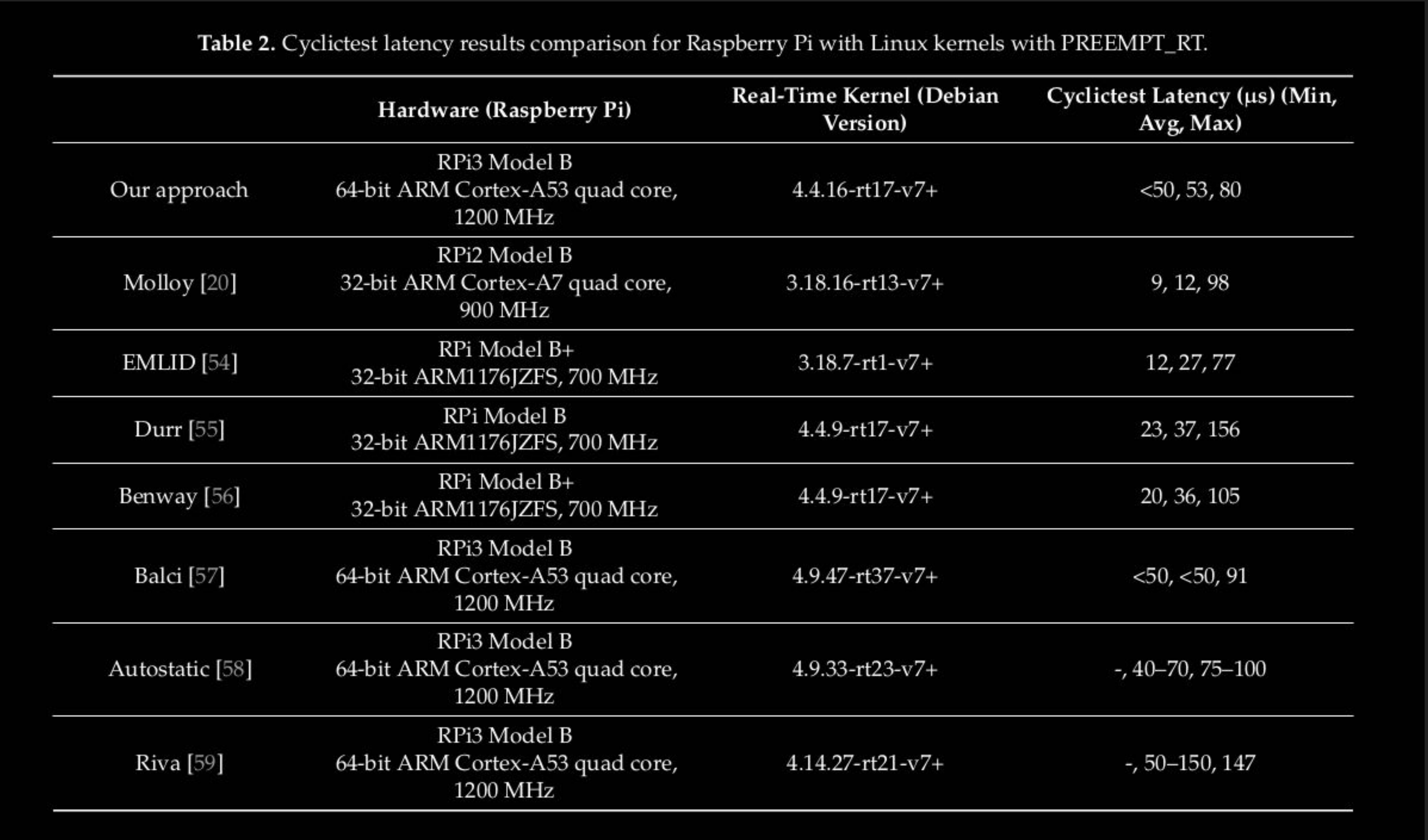


Table 2. Cyclictest latency results comparison for Raspberry Pi with Linux kernels with PREEMPT_RT.

	Hardware (Raspberry Pi)	Real-Time Kernel (Debian Version)	Cyclictest Latency (µs) (Min, Avg, Max)
Our approach	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.4.16-rt17-v7+	<50, 53, 80
Molloy [20]	RPi2 Model B 32-bit ARM Cortex-A7 quad core, 900 MHz	3.18.16-rt13-v7+	9, 12, 98
EMLID [54]	RPi Model B+ 32-bit ARM1176JZFS, 700 MHz	3.18.7-rt1-v7+	12, 27, 77
Durr [55]	RPi Model B 32-bit ARM1176JZFS, 700 MHz	4.4.9-rt17-v7+	23, 37, 156
Benway [56]	RPi Model B+ 32-bit ARM1176JZFS, 700 MHz	4.4.9-rt17-v7+	20, 36, 105
Balci [57]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.9.47-rt37-v7+	<50, <50, 91
Autostatic [58]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.9.33-rt23-v7+	-, 40–70, 75–100
Riva [59]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.14.27-rt21-v7+	-, 50–150, 147

RT kernel (Min, Avg, Max)
1, 5.9, 124

cf. Adam et al.

Benchmarking results (in µs)

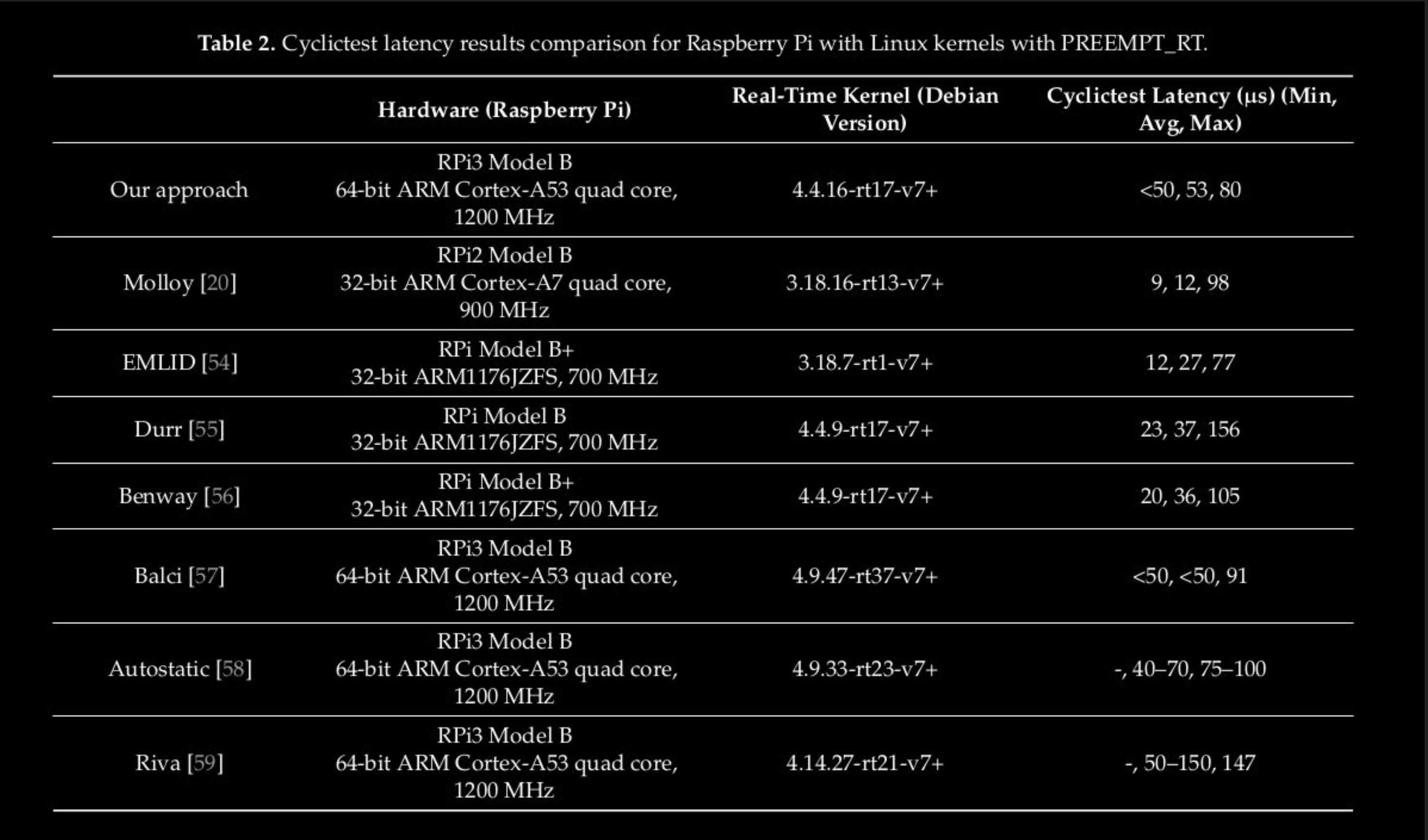


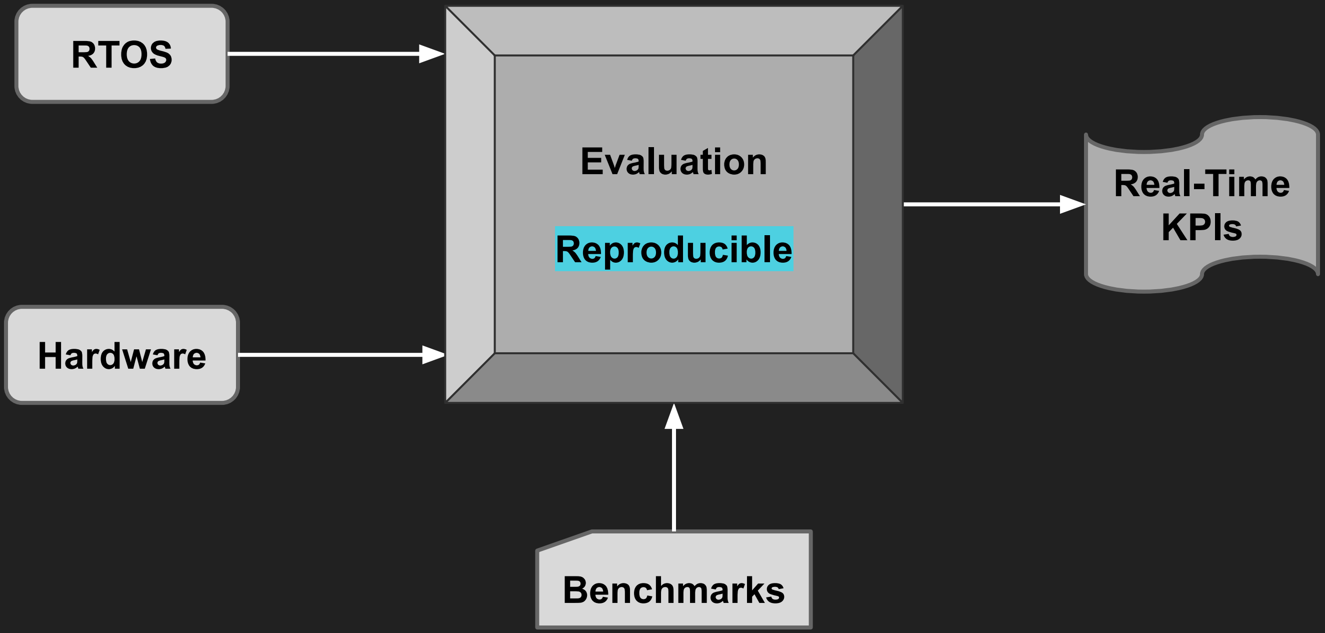
Table 2. Cyclictest latency results comparison for Raspberry Pi with Linux kernels with PREEMPT_RT.

	Hardware (Raspberry Pi)	Real-Time Kernel (Debian Version)	Cyclictest Latency (µs) (Min, Avg, Max)
Our approach	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.4.16-rt17-v7+	<50, 53, 80
Molloy [20]	RPi2 Model B 32-bit ARM Cortex-A7 quad core, 900 MHz	3.18.16-rt13-v7+	9, 12, 98
EMLID [54]	RPi Model B+ 32-bit ARM1176JZFS, 700 MHz	3.18.7-rt1-v7+	12, 27, 77
Durr [55]	RPi Model B 32-bit ARM1176JZFS, 700 MHz	4.4.9-rt17-v7+	23, 37, 156
Benway [56]	RPi Model B+ 32-bit ARM1176JZFS, 700 MHz	4.4.9-rt17-v7+	20, 36, 105
Balci [57]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.9.47-rt37-v7+	<50, <50, 91
Autostatic [58]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.9.33-rt23-v7+	-, 40–70, 75–100
Riva [59]	RPi3 Model B 64-bit ARM Cortex-A53 quad core, 1200 MHz	4.14.27-rt21-v7+	-, 50–150, 147

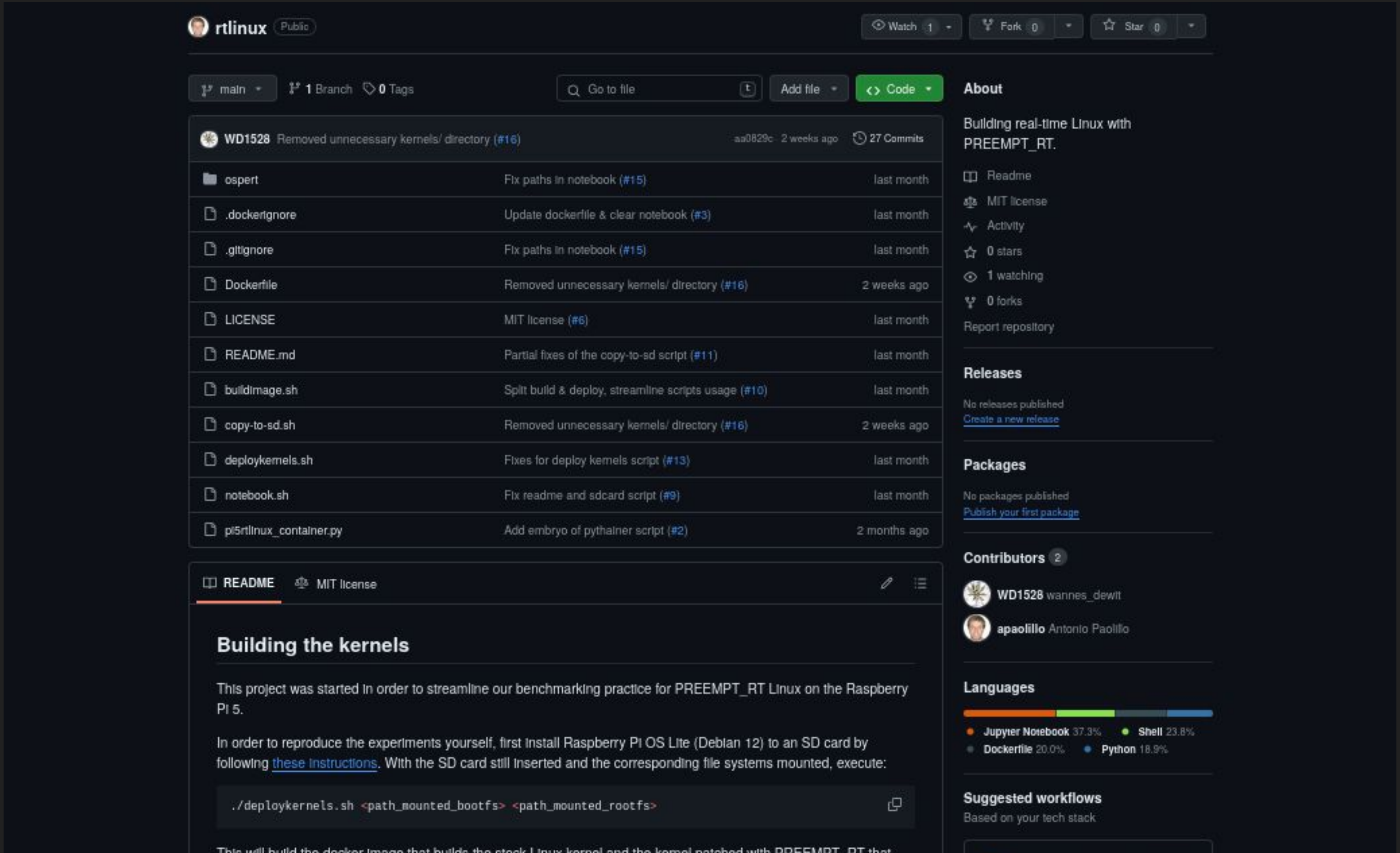
RT kernel (Min, Avg, Max)
1, 5.9, 124

cf. Adam et al.

Real-Time Linux?



https://github.com/apaolillo/rtlinux



rtlinux Public

WD1528 Removed unnecessary kernels/ directory (#16) aa0829c 2 weeks ago 27 Commits

ospert	Fix paths in notebook (#15)	last month
.dockerignore	Update dockerfile & clear notebook (#3)	last month
.gitignore	Fix paths in notebook (#15)	last month
Dockerfile	Removed unnecessary kernels/ directory (#16)	2 weeks ago
LICENSE	MIT license (#6)	last month
README.md	Partial fixes of the copy-to-sd script (#11)	last month
buildimage.sh	Split build & deploy, streamline scripts usage (#10)	last month
copy-to-sd.sh	Removed unnecessary kernels/ directory (#16)	2 weeks ago
deploykernels.sh	Fixes for deploy kernels script (#13)	last month
notebook.sh	Fix readme and sdcard script (#9)	last month
pi5rtlinux_container.py	Add embryo of pythainer script (#2)	2 months ago

Building the kernels

This project was started in order to streamline our benchmarking practice for PREEMPT_RT Linux on the Raspberry Pi 5.

In order to reproduce the experiments yourself, first install Raspberry Pi OS Lite (Debian 12) to an SD card by following these instructions. With the SD card still inserted and the corresponding file systems mounted, execute:

```
./deploykernels.sh <path_mounted_bootfs> <path_mounted_rootfs>
```

This will build the docker image that builds the stock Linux kernel and the kernel patched with PREEMPT_RT that

About

Building real-time Linux with PREEMPT_RT.

- Readme
- MIT license
- Activity
- 0 stars
- 1 watching
- 0 forks

Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 2

WD1528 wannes_dewit
apaolillo Antonio Paolillo

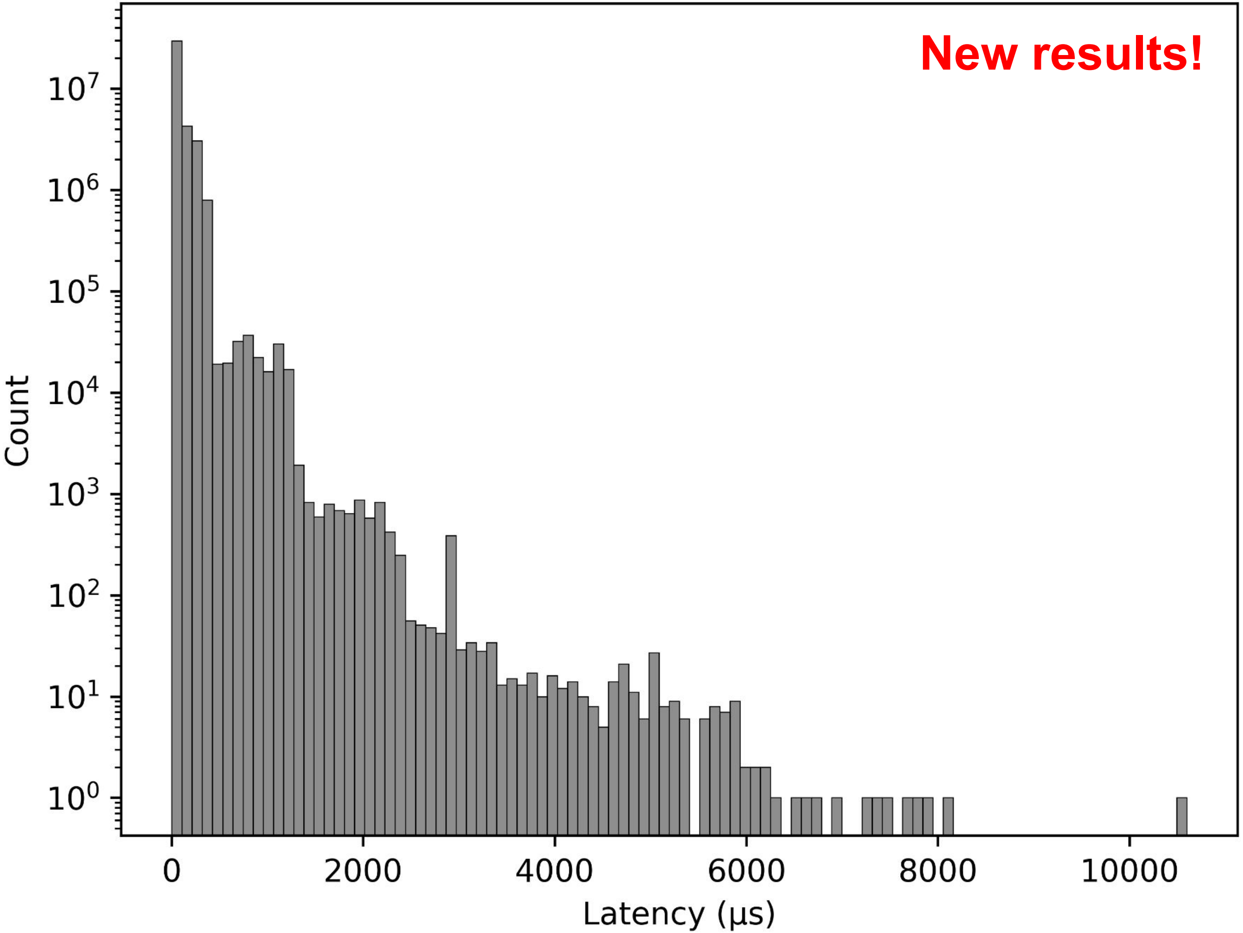
Languages

● Jupyter Notebook 37.3%	● Shell 23.8%
● Dockerfile 20.0%	● Python 18.9%

Suggested workflows

Based on your tech stack

Cyclictest latencies on Raspberry Pi 4
with stock Linux v6.6.21 (unpatched)



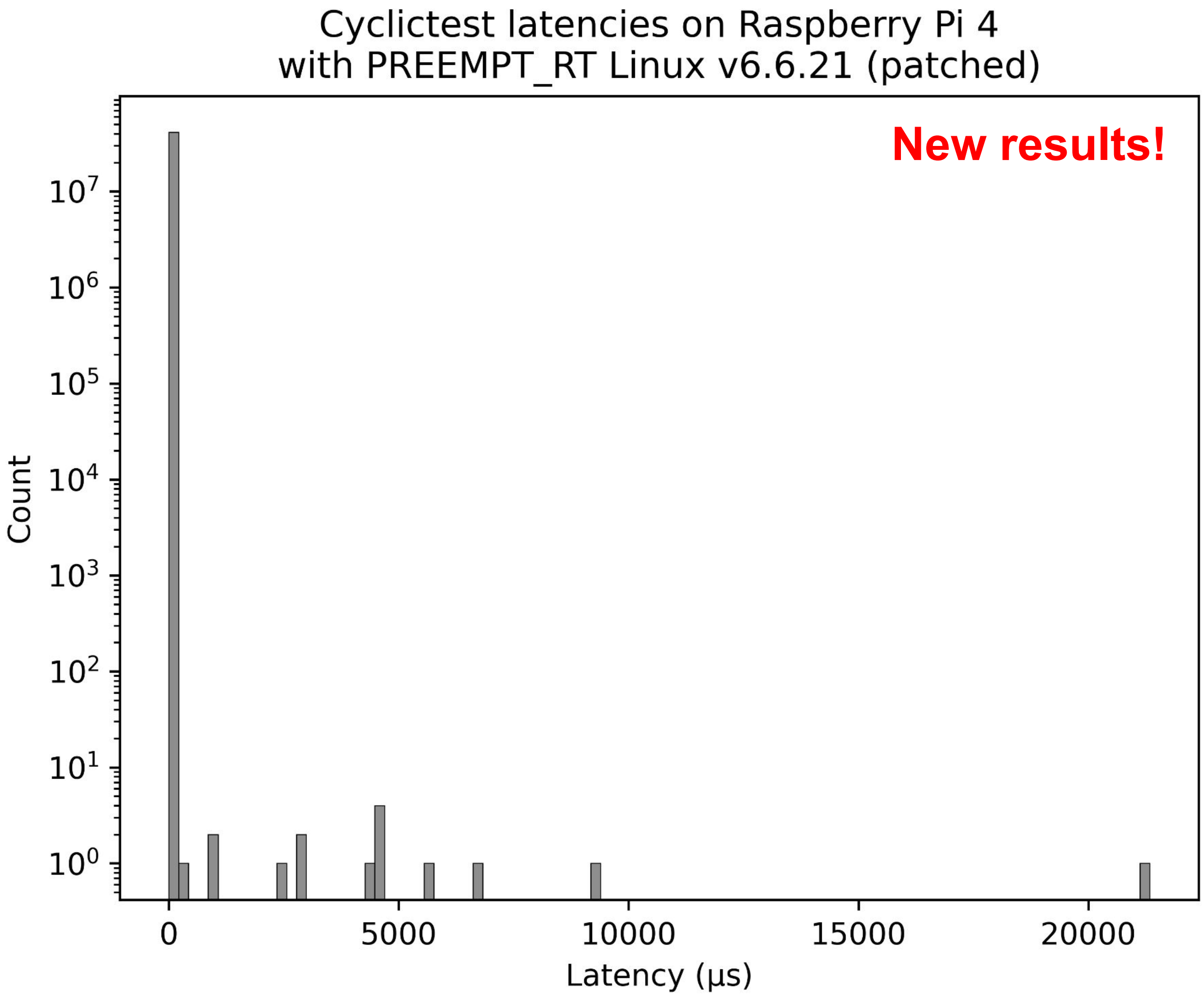
New results!

Cyclictest results

Raspberry Pi 4
Model B

Debian Linux kernel
6.6.21-v8-16k+

Max. latency:
10598μs



New results!

Cyclictest results

Raspberry Pi 4
Model B

Debian Linux kernel
6.6.21-rt26-v8-16k+

With PREEMPT_RT

Max. latency:
21332µs

Future work

* RT Linux experiments: tune the kernel for RT performance

→ RT throttling

→ CPU freq

→ priority of interrupts

* Benchmarking tools

→ rt-bench, RTEval

→ timerlat

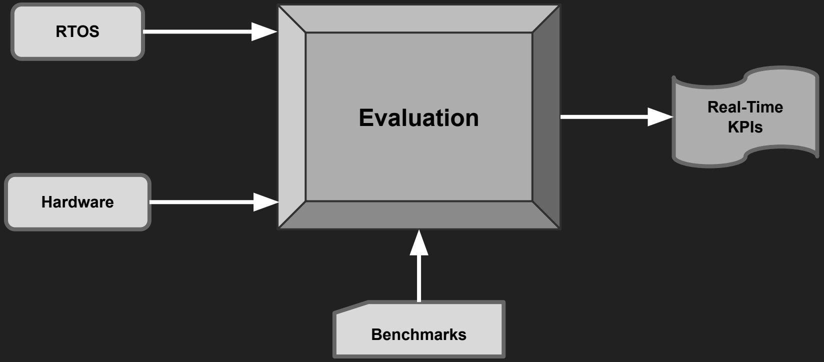
→ benchkit [1]

* Metrics

→ end-to-end response latency

→ RTOS jitter

→ raw performance comparison (e.g. throughput) for rt *drawbacks*



⇒ Suggestions are welcome!

[1] https://github.com/open-s4c/benchkit

Alternative RTOS

freeRTOS

LITMUS^{RT}

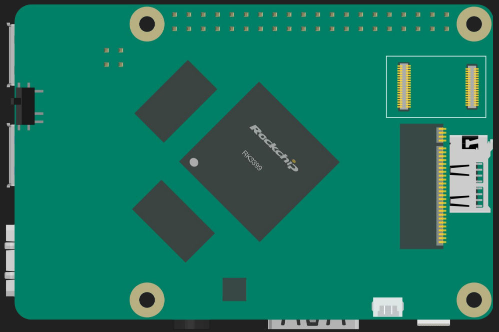
Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

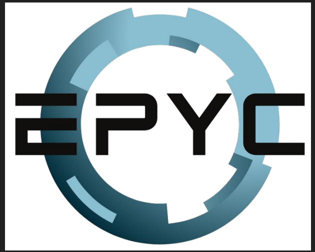
XENOMAI

Zephyr™

VxWorks

Alternative Hardware





Used images / References:

- https://geral.com/etude-fabrication-serie/systemes-embarques-mesure/software/xenomai-logo/
- https://intellinium.io/zephyr-os-is-ready-for-connecter-worker-devices/
- http://www.advantech.com/products/bac450df-e534-4d08-be67-8af5cdb692e9/vxworks/mod_e10fd9bc-7a00-4338-90fe-f1e6df160b4b
- https://jackaudio.org/faq/linux_rt_config.html
- https://www.lifehacker.com.au/2020/07/how-to-watch-nasas-mars-rover-launch-live/
- https://www.pulse-cardiology.com/blog/types-of-pacemakers/
- https://www.youtube.com/watch?v=DUeRp3RCubo
- https://wiki.linuxfoundation.org/realtime/rtl/blog#preempt-rt-history
- Paraskevas Karachatzis, Jan Ruh, and Silviu S. Craciunas. 2023. An Evaluation of Time-triggered Scheduling in the Linux Kernel. In Proceedings of the 31st International Conference on Real-Time Networks and Systems (RTNS '23). Association for Computing Machinery, New York, NY, USA, 119–131. https://doi.org/10.1145/3575A757.3593660
- Adam, George K., Nikos Petrellis, and Lambros T. Doulos. 2021. "Performance Assessment of Linux Kernels with PREEMPT_RT on ARM-Based Embedded Devices" *Electronics* 10, no. 11: 1331. https://doi.org/10.3390/electronics10111331