



Max
Planck
Institute
for
Software Systems

A Comparison of **Scheduling Latency** in Linux, PREEMPT_RT, and LITMUS^{RT}

Felipe Cerqueira and Björn Brandenburg

July 9th, 2013

Linux as a Real-Time OS

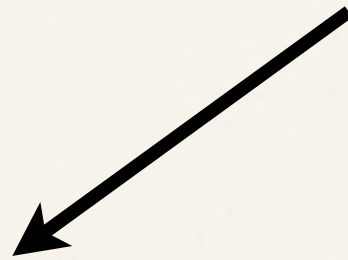


Linux as a Real-Time OS



Optimizing
system
responsiveness

Linux as a Real-Time OS



Optimizing
system
responsiveness



PREEMPT_RT
(Linux)

Linux as a Real-Time OS



Optimizing
system
responsiveness



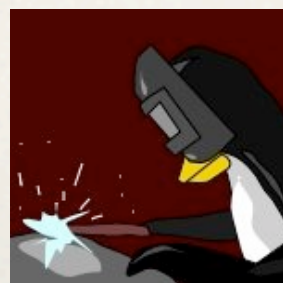
PREEMPT_RT
(Linux)

Algorithmic changes
based on real-time
systems research

Linux as a Real-Time OS



Optimizing
system
responsiveness



PREEMPT_RT
(Linux)

Algorithmic changes
based on real-time
systems research

LITMUS^{RT}

Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

PREEMPT_RT



- ❖ Main real-time branch of Linux
- ❖ **Goal:** decrease **scheduling latency** through the use of low-level hacks
 - ❖ Convert in-kernel spinlocks into (preemptable) mutexes
 - ❖ Limit the extent of non-preemptable sections
- ❖ Commonly evaluated with **cyclicttest**
 - ❖ **Single**, easy-to-compare measure of scheduling latency as output

LITMUS^{RT}

Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

- ❖ Testbed for applied real-time systems research
- ❖ **Goal**
 - ❖ Allow implementation and evaluation of novel multiprocessor schedulers and synchronization protocols
 - ❖ **NOT** to reduce scheduling latency
- ❖ Evaluated with **Feather-Trace**
 - ❖ Flexible, fine-grained measurement of **different** overheads

- ❖ Testbed for multiprocessor scheduling in real-time systems

- ❖ Goal

How do LITMUS^{RT} and
PREEMPT_RT compare?

processor

- ❖ **NOT** to reduce scheduling latency

- ❖ Evaluated with **Feather-Trace**

- ❖ Flexible, fine-grained measurement of **different** overheads

- ❖ Testbed for multiprocessor scheduling in real-time systems

- ❖ Goal

How do LITMUS^{RT} and
PREEMPT_RT compare?

processor

- ❖ Eval

It is not straightforward to
compare them!

ads

Objective



VS.



PREEMPT_RT
(Linux)

Direct comparison of
scheduling latency between
LITMUS^{RT} and PREEMPT_RT

Background

How is LITMUS^{RT} evaluated?

- ✧ Evaluated with *feathertrace*
 - ✧ Lightweight tracing framework for measuring fine-grained overheads (e.g., IPI latency, context-switching overhead, etc.)
 - ✧ Extensively used (20+ publications)
 - ✧ Suitable for schedulability analysis
 - ✧ Check if a task is going to miss a deadline

How is PREEMPT_RT evaluated?

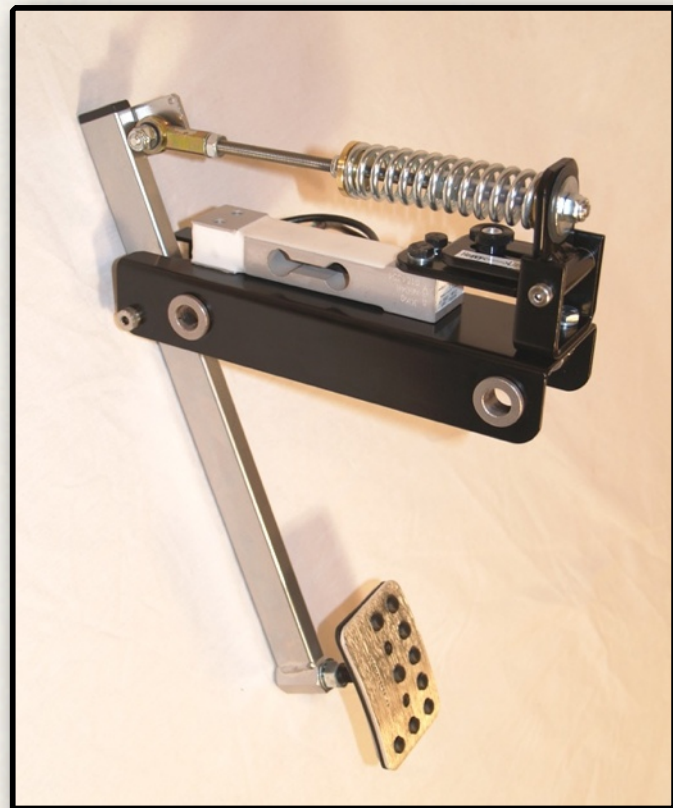
- ❖ Evaluated with **cyclictest**
- ❖ Standard benchmark for assessing real-time responsiveness
- ❖ Creator: Thomas Gleixner
Current maintainer: Clark Williams
- ❖ Reports **scheduling latency** as a single measure
 - ❖ Treats hardware and OS as a black-box

```
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.00 0.01 0.05 1/178 4320

T: 0 ( 4305) P:99 I:1000 C: 13092 Min:      2 Act:    3 Avg:    3 Max:    9
T: 1 ( 4306) P:99 I:1500 C:  8728 Min:      2 Act:    3 Avg:    2 Max:    9
T: 2 ( 4307) P:99 I:2000 C:  6546 Min:      2 Act:    2 Avg:    2 Max:    5
T: 3 ( 4308) P:99 I:2500 C:  5236 Min:      2 Act:    2 Avg:    2 Max:    4
T: 4 ( 4309) P:99 I:3000 C:  4364 Min:      2 Act:    2 Avg:    2 Max:    3
T: 5 ( 4310) P:99 I:3500 C:  3740 Min:      2 Act:    2 Avg:    2 Max:    3
T: 6 ( 4311) P:99 I:4000 C:  3273 Min:      2 Act:    3 Avg:    2 Max:    4
T: 7 ( 4312) P:99 I:4500 C:  2909 Min:      2 Act:    3 Avg:    2 Max:    4
T: 8 ( 4313) P:99 I:5000 C:  2618 Min:      3 Act:    3 Avg:    3 Max:    4
T: 9 ( 4314) P:99 I:5500 C:  2380 Min:      2 Act:    2 Avg:    2 Max:    6
T:10 ( 4315) P:99 I:6000 C:  2182 Min:      2 Act:    4 Avg:    3 Max:    4
T:11 ( 4316) P:99 I:6500 C:  2014 Min:      2 Act:    3 Avg:    2 Max:    4
T:12 ( 4317) P:99 I:7000 C:  1870 Min:      2 Act:    3 Avg:    3 Max:    6
T:13 ( 4318) P:99 I:7500 C:  1745 Min:      2 Act:    3 Avg:    2 Max:    3
T:14 ( 4319) P:99 I:8000 C:  1636 Min:      2 Act:    4 Avg:    3 Max:    4
T:15 ( 4320) P:99 I:8500 C:  1540 Min:      2 Act:    3 Avg:    2 Max:    4
```

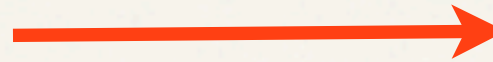

Scheduling Latency

Time until the highest-priority task is scheduled



brake sensor
HP task

interrupt!



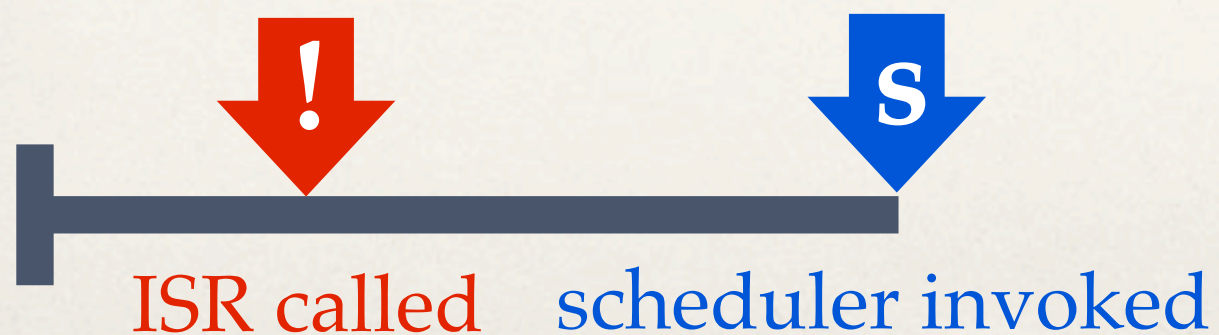
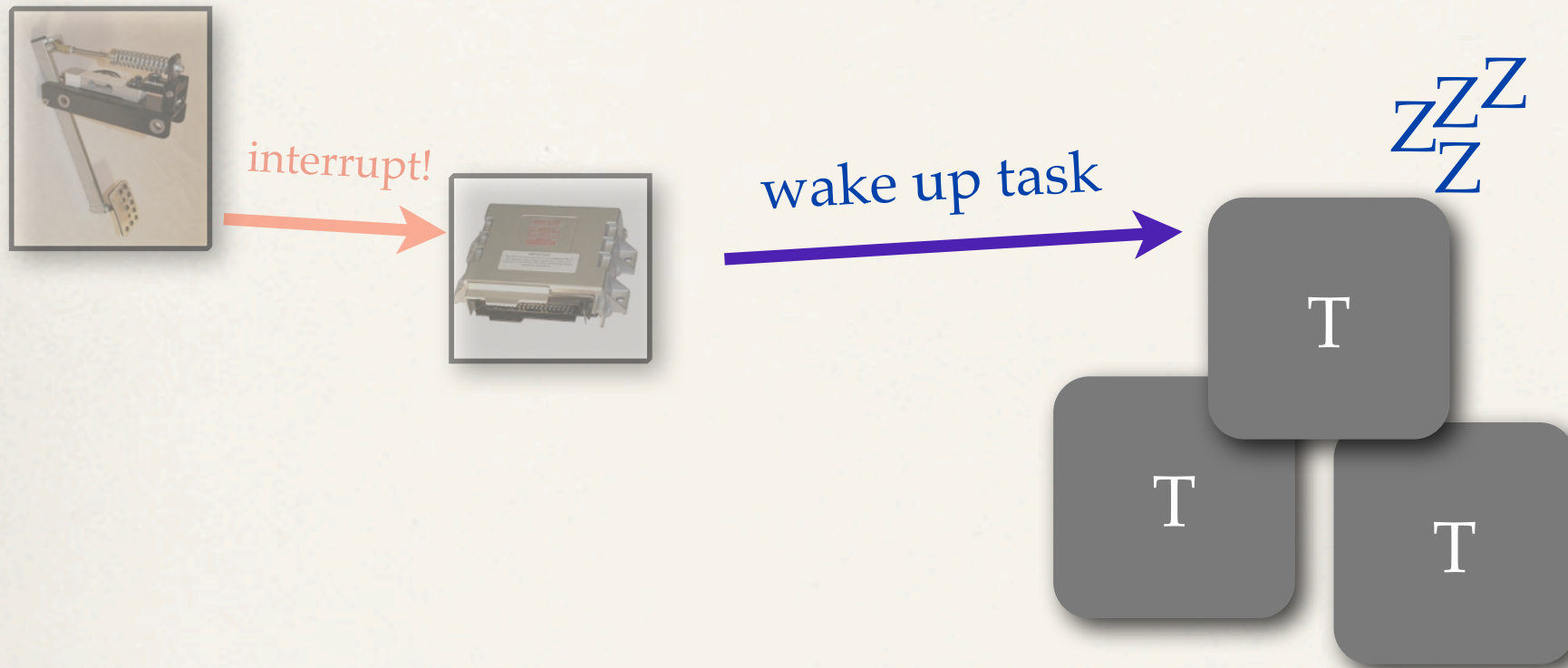
ECU



ISR called

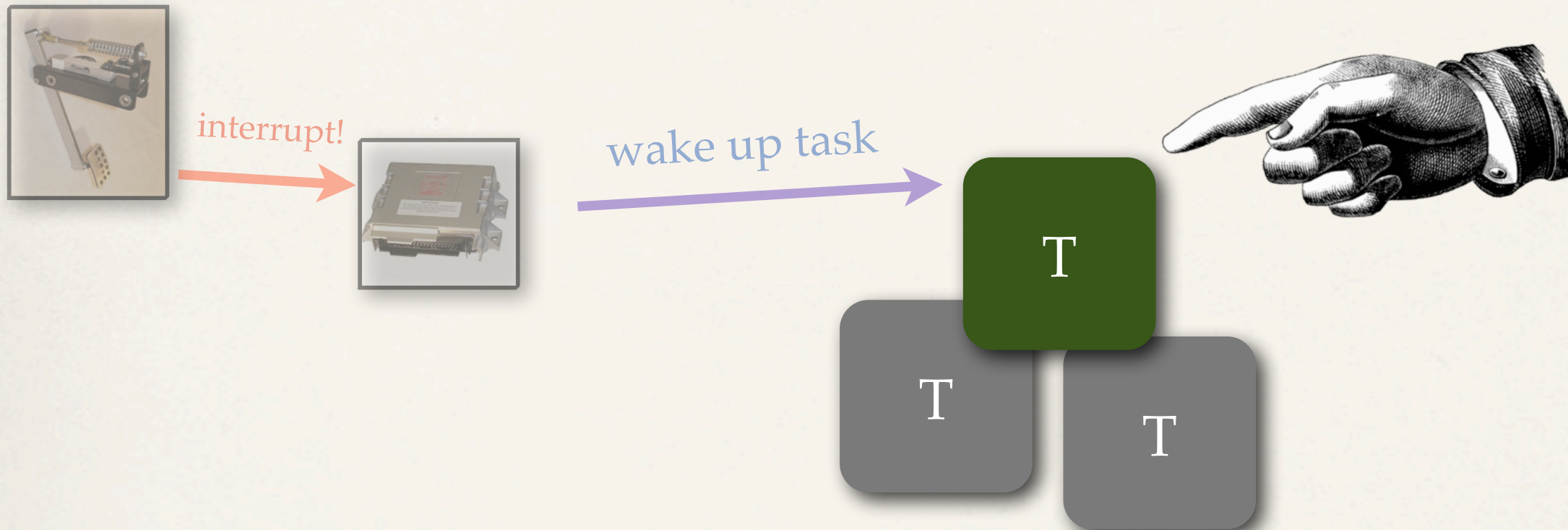
Scheduling Latency

Time until the highest-priority task is scheduled



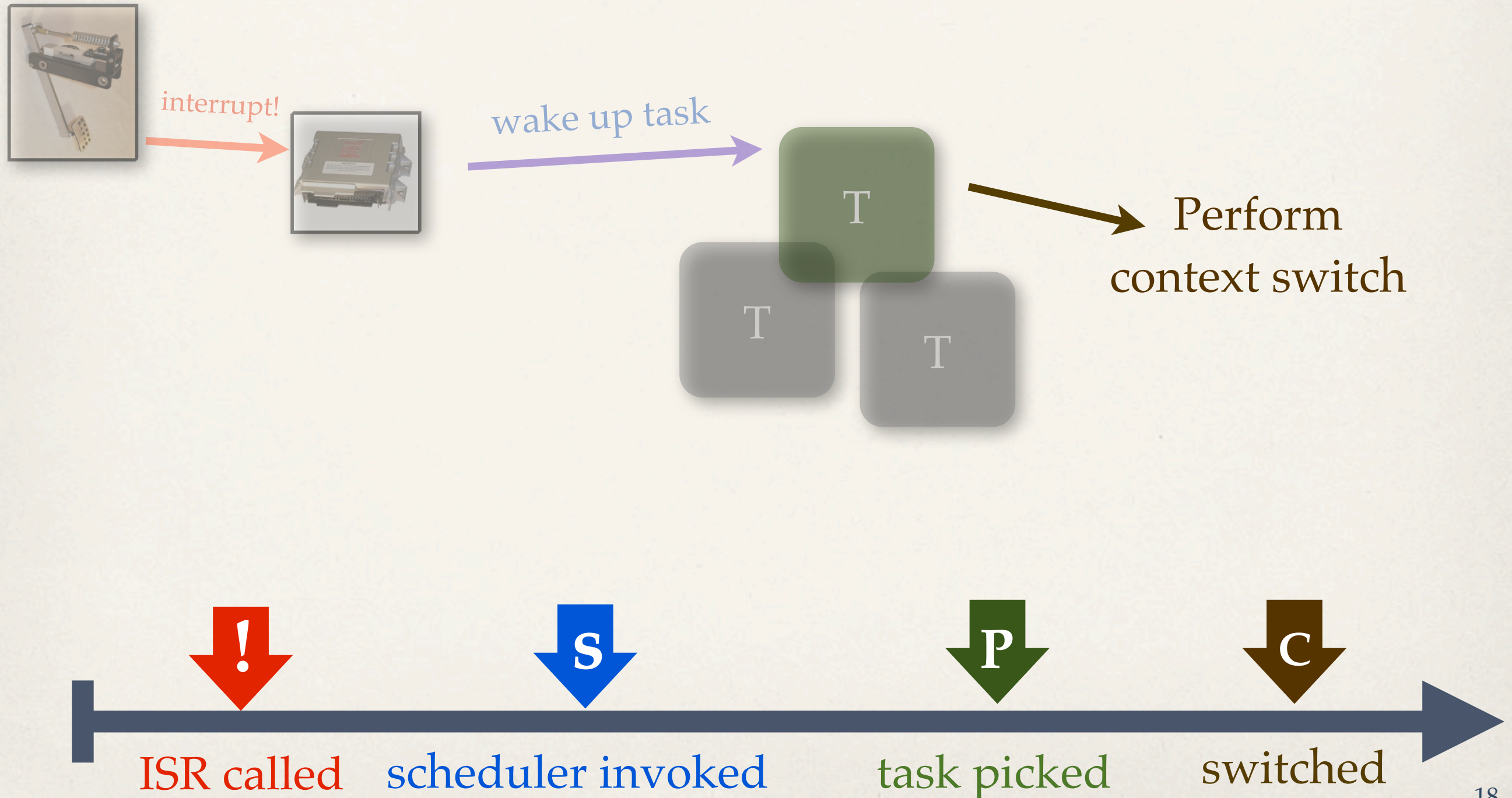
Scheduling Latency

Time until the highest-priority task is scheduled



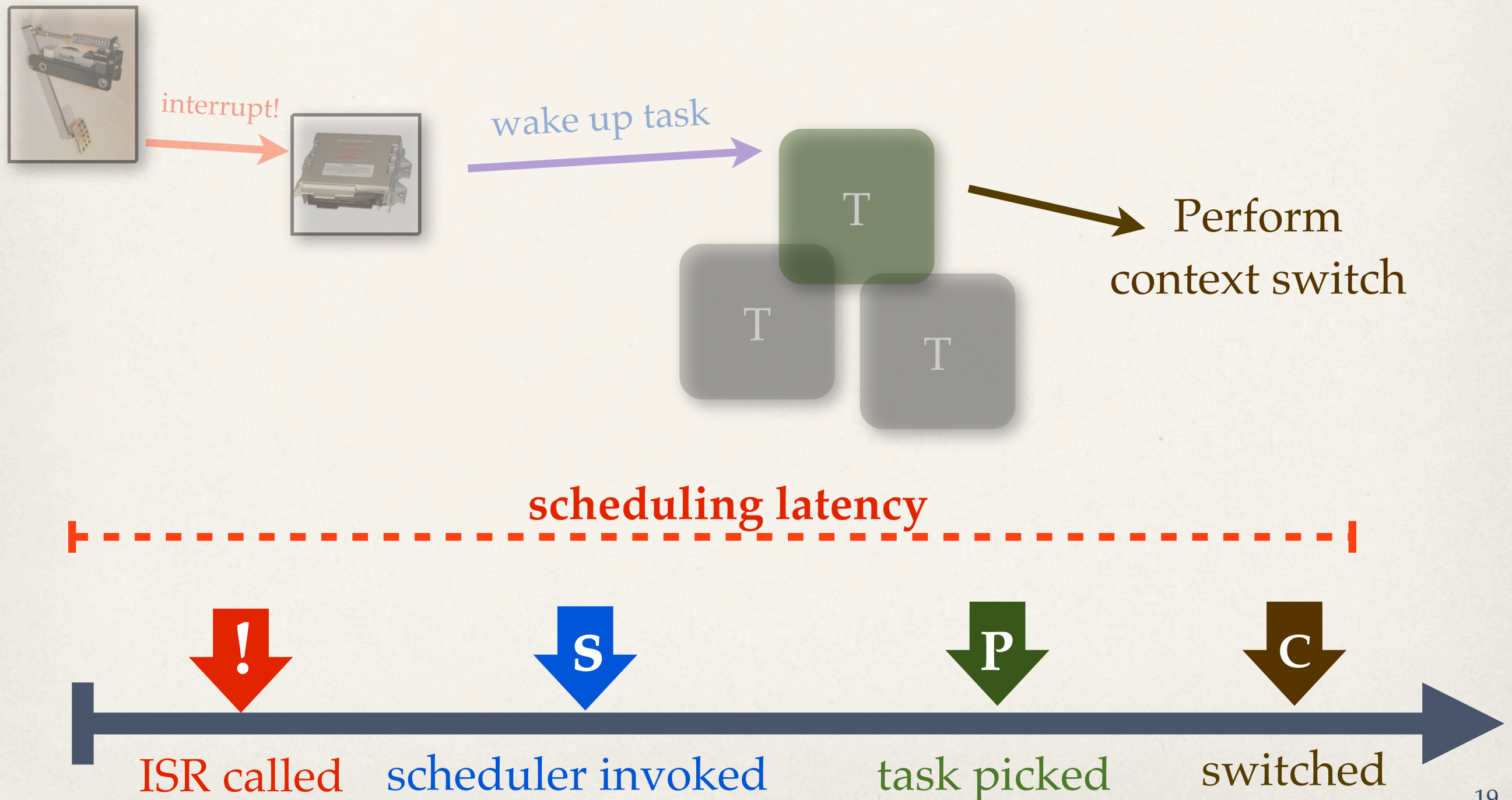
Scheduling Latency

Time until the highest-priority task is scheduled

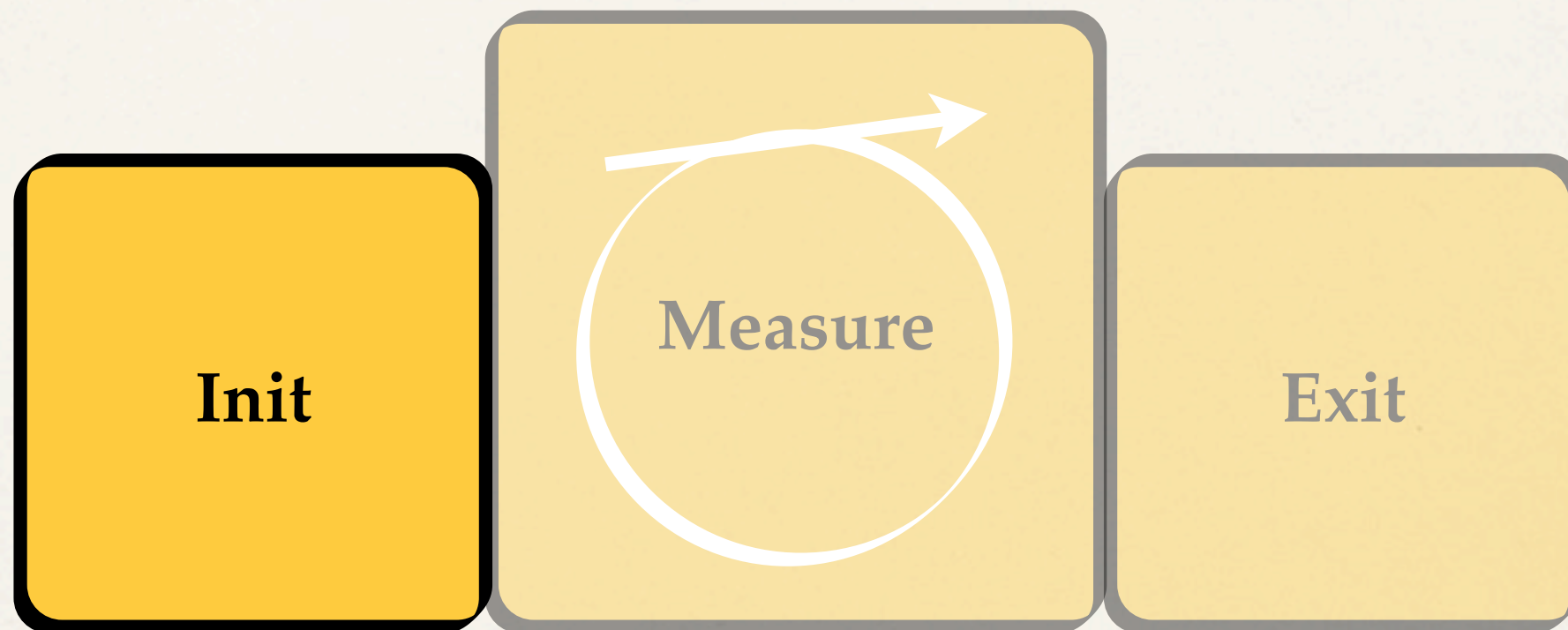


Scheduling Latency

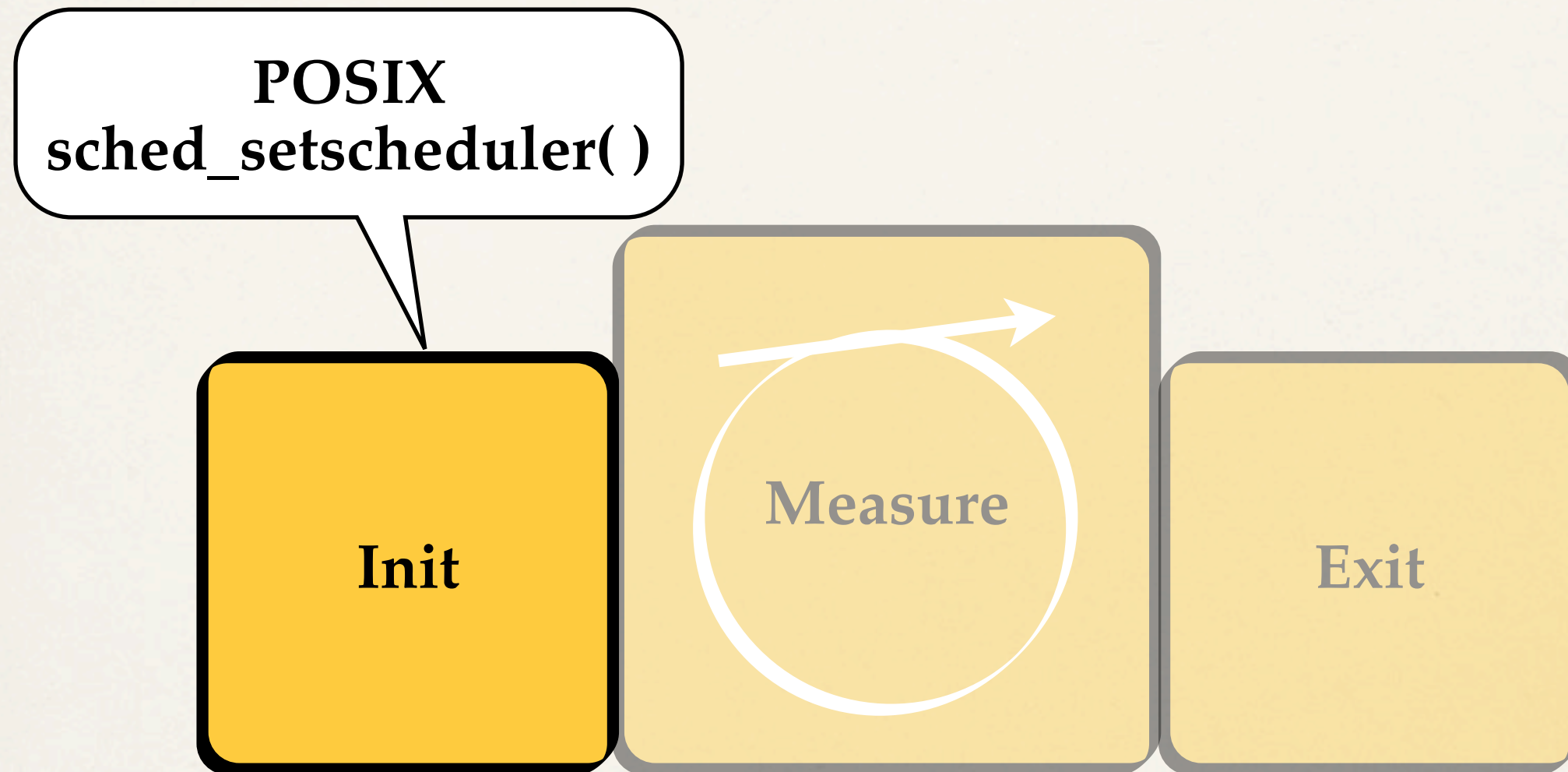
Time until the highest-priority task is scheduled



How does cyclicttest measure Scheduling Latency?

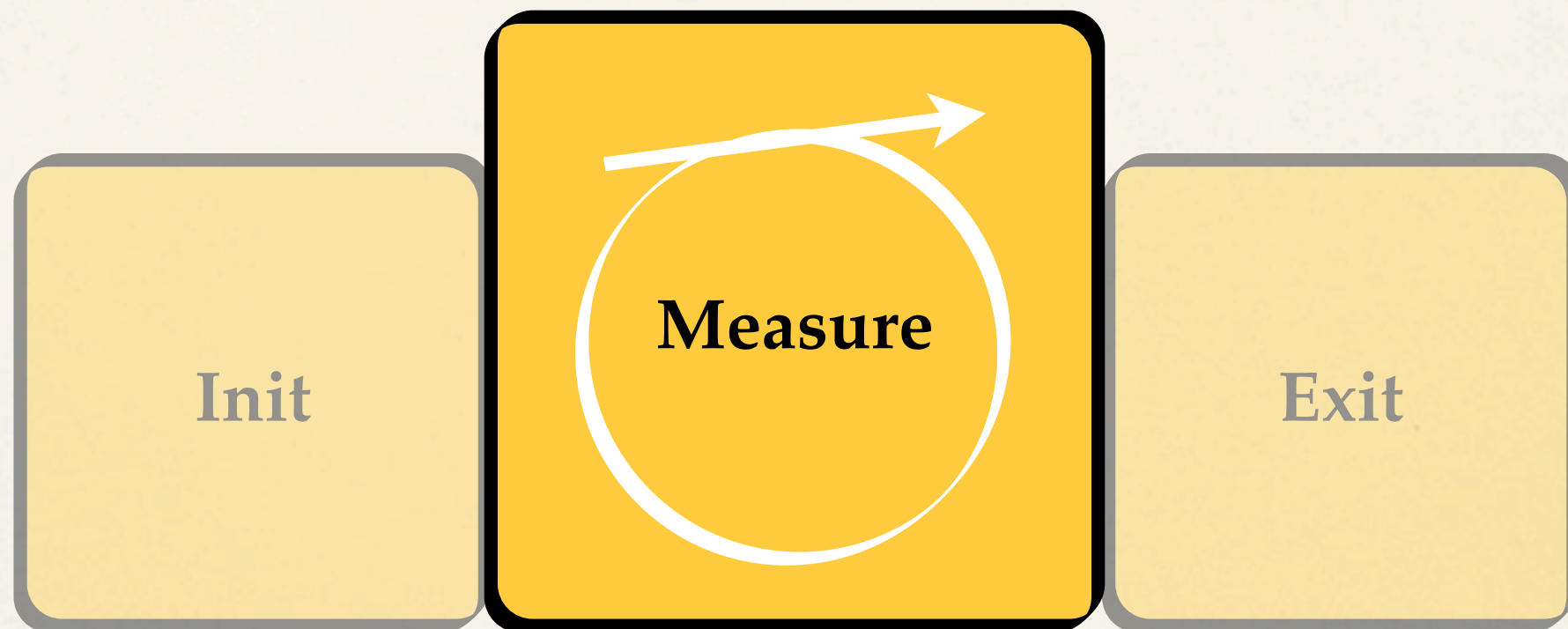


How does cyclicttest measure Scheduling Latency?

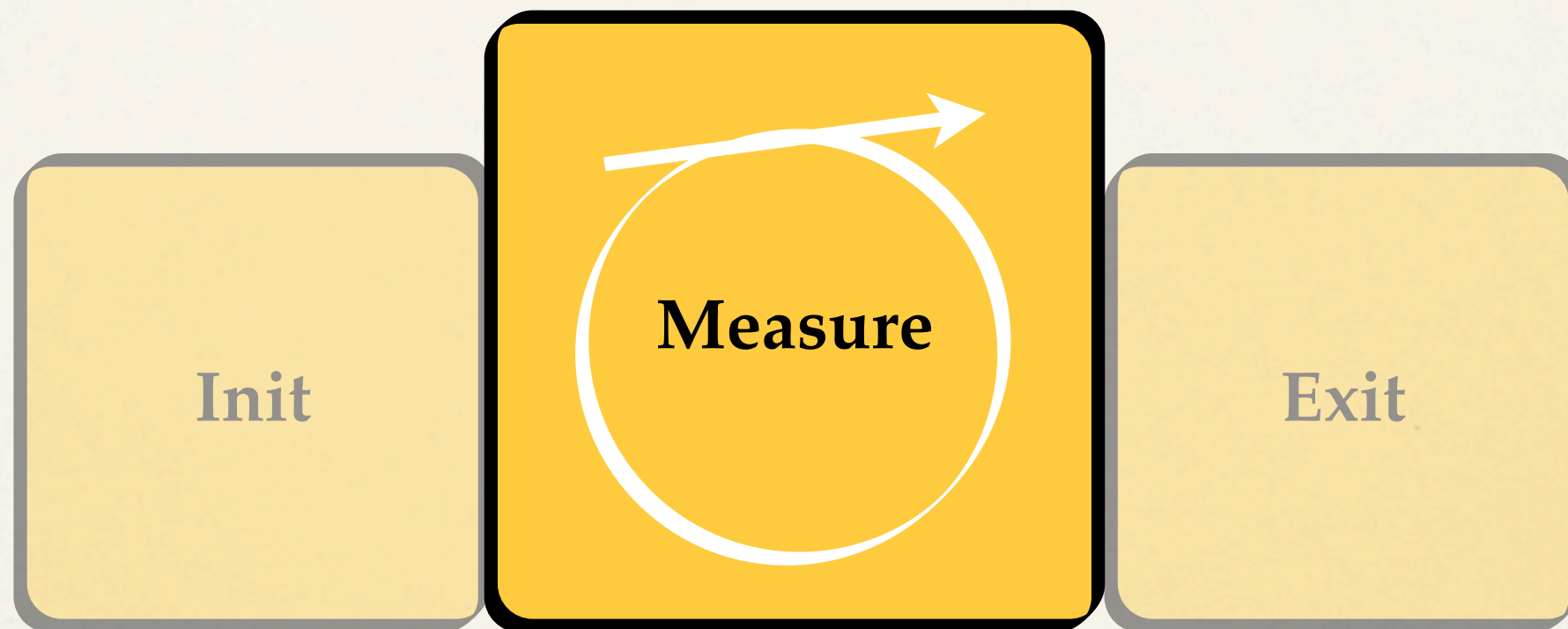


Measuring thread is granted real-time status.

How does cyclicttest measure Scheduling Latency?

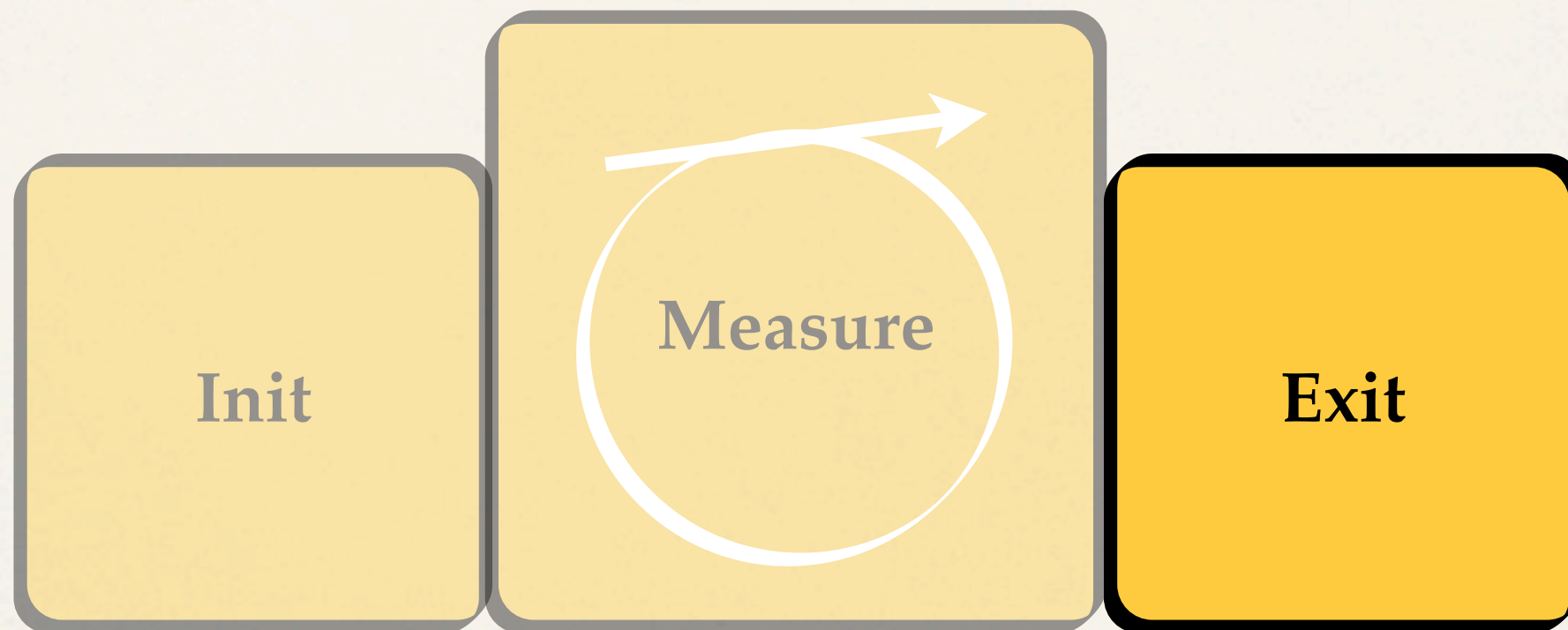


How does cyclicttest measure Scheduling Latency?

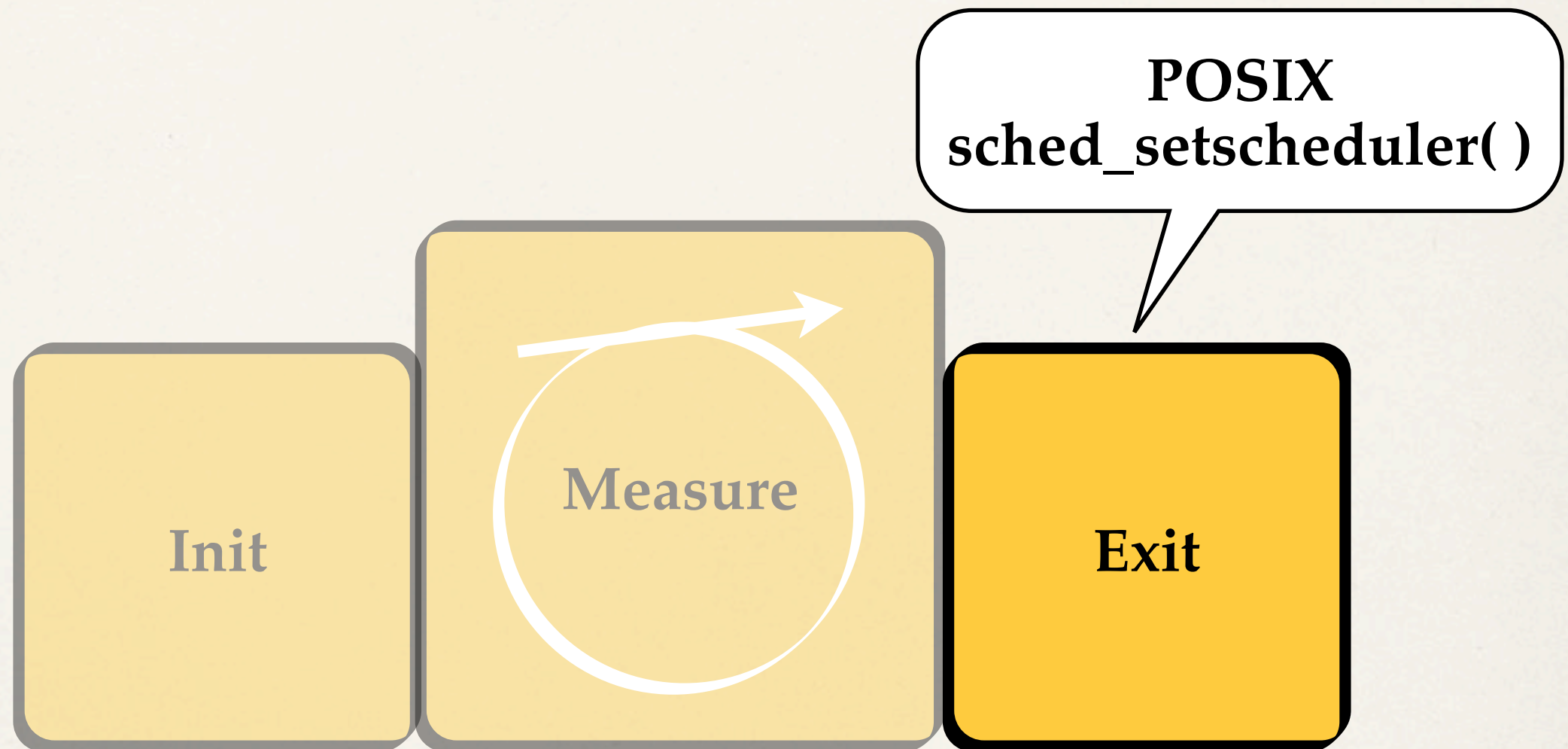


Periodically setup one-shot timers with *nanosleep*.
Calculate delta between the instant the task starts
executing and the instant the timer should have fired.

How does cyclicttest measure Scheduling Latency?

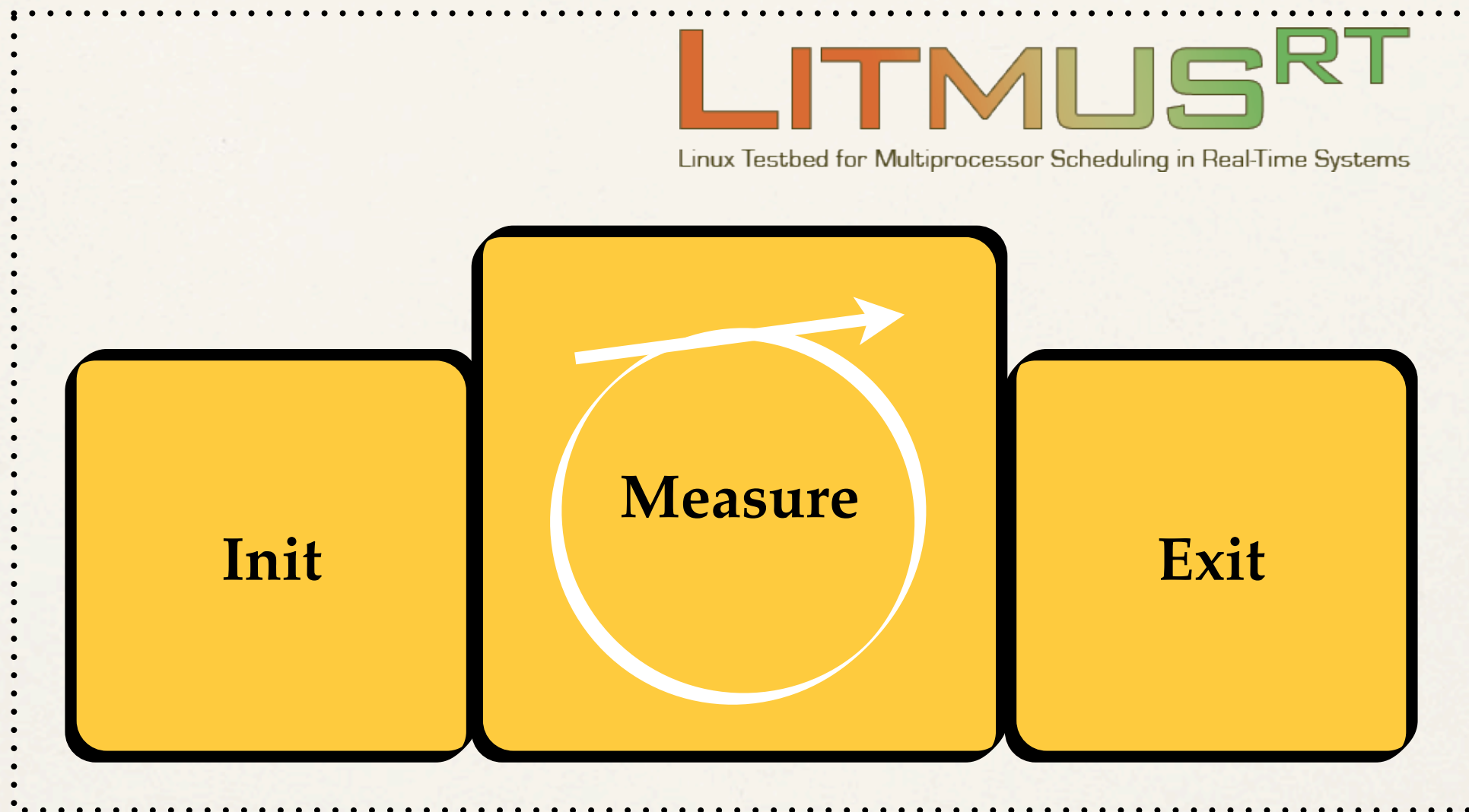


How does cyclicttest measure Scheduling Latency?



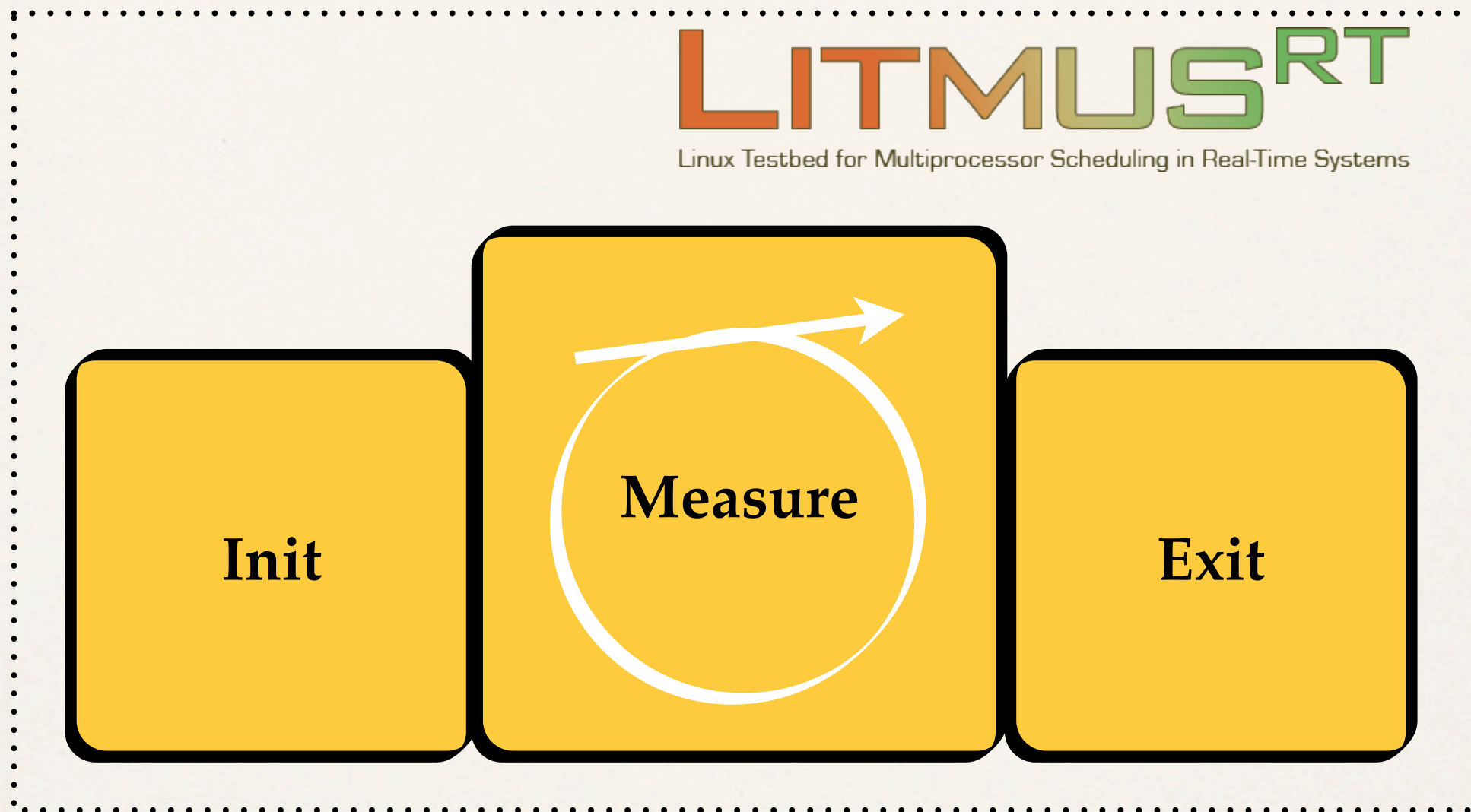
Measuring thread returns to best-effort status.

cyclictest on LITMUS^{RT}



LITMUS^{RT} does not use POSIX API to setup real-time tasks!

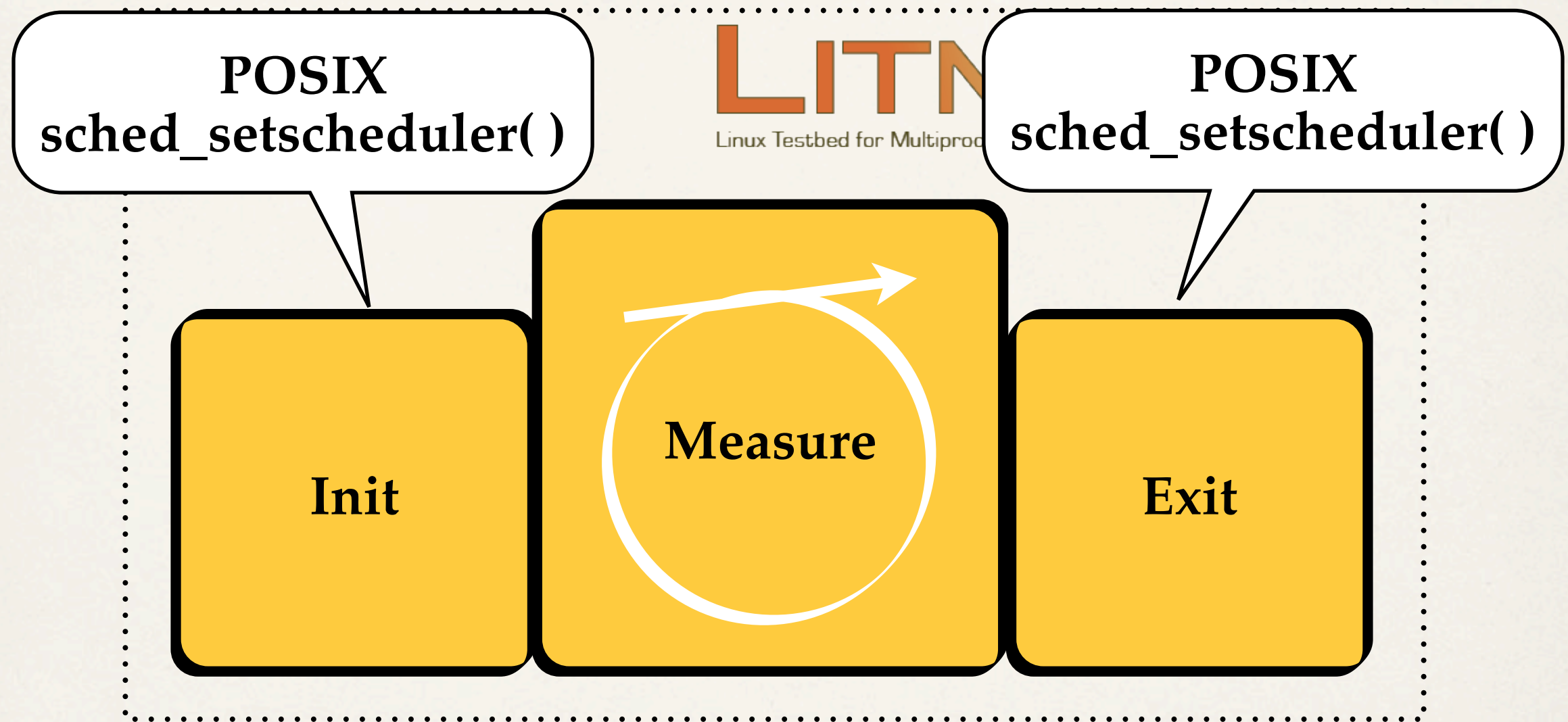
cyclictest on LITMUS^{RT}



LITMUS^{RT} does not use POSIX API to setup real-time tasks!

cyclictest works, but does not measure what we expect...

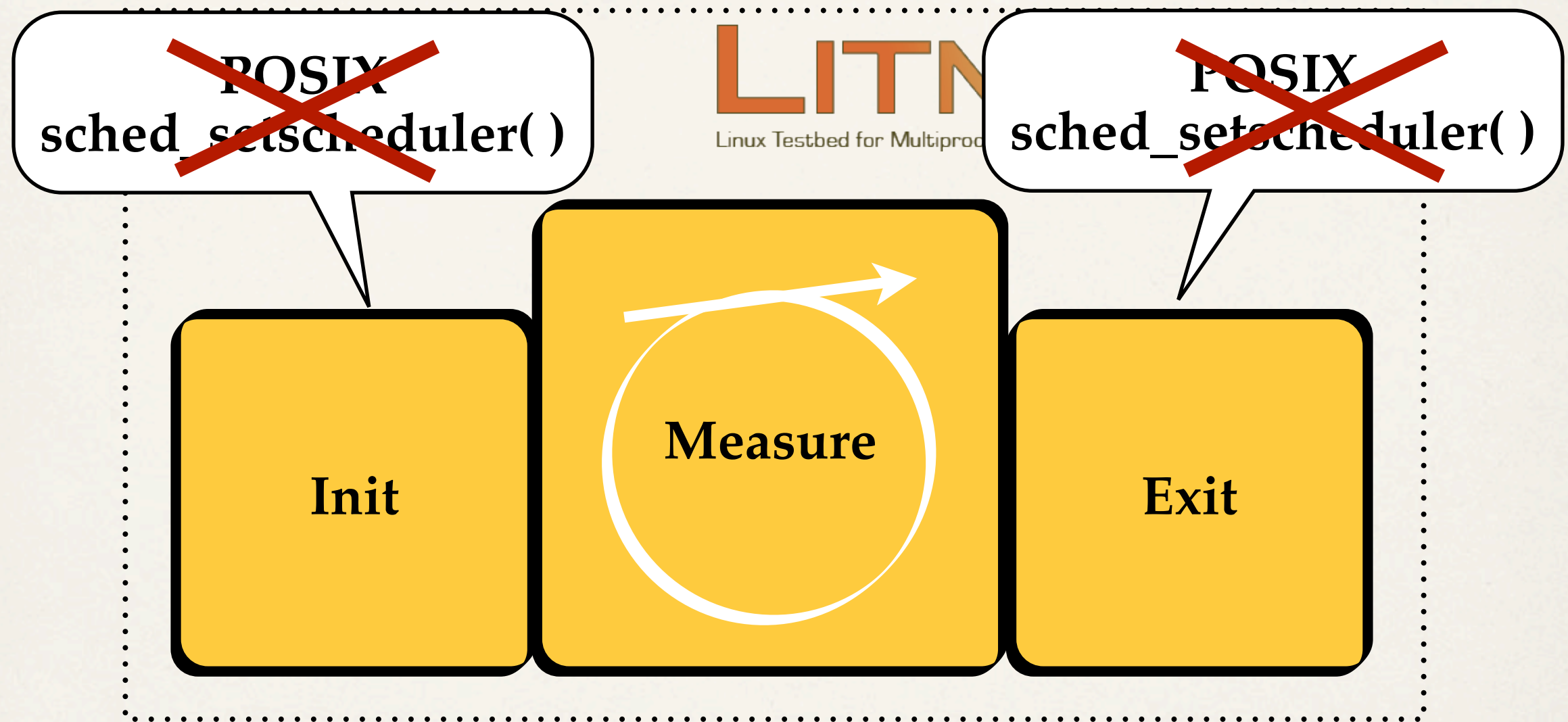
Porting cyclicttest to LITMUS^{RT}



LITMUS^{RT} does not use POSIX API to setup real-time tasks!

cyclicttest works, but does not measure what we expect...

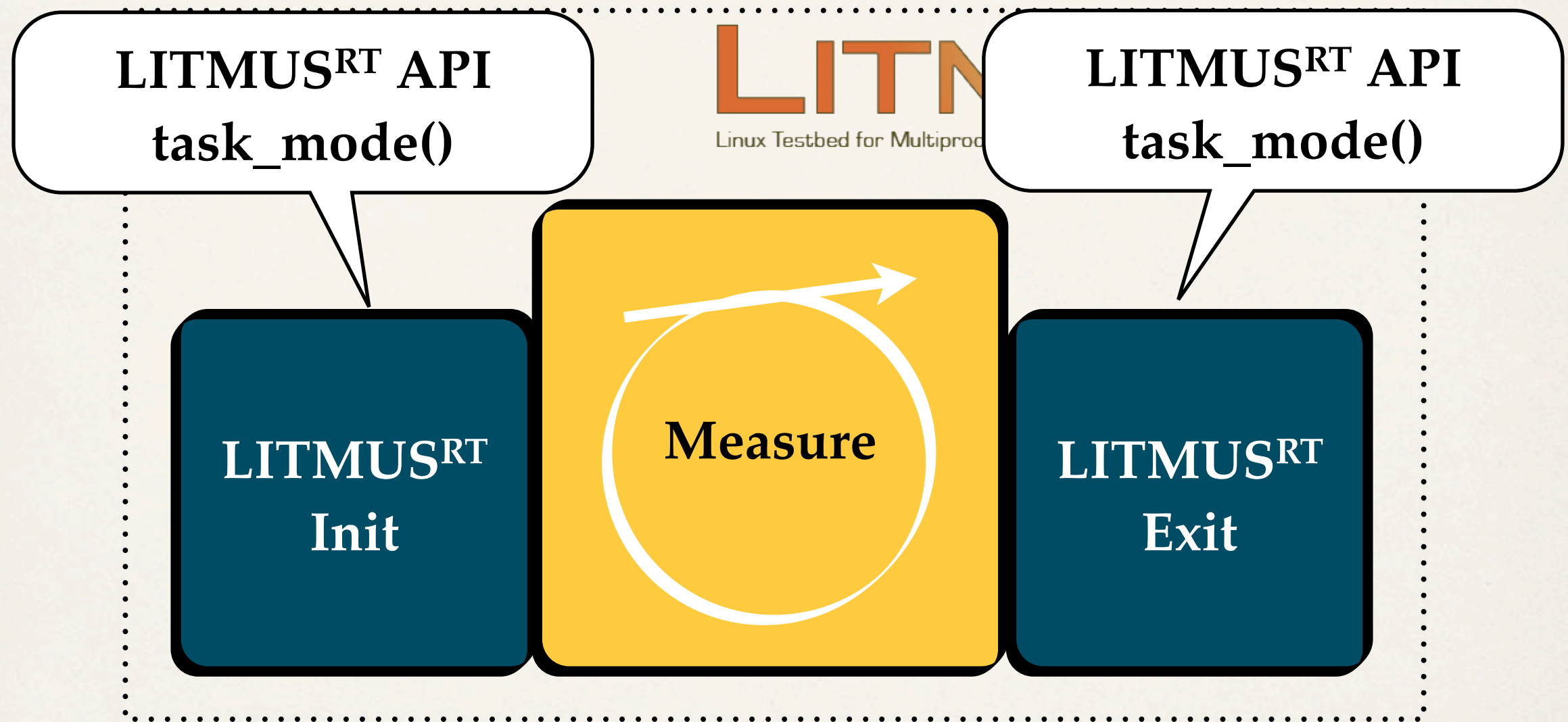
Porting cyclicttest to LITMUS^{RT}



LITMUS^{RT} does not use POSIX API to setup real-time tasks!

cyclicttest works, but does not measure what we expect...

Porting cyclicttest to LITMUS^{RT}

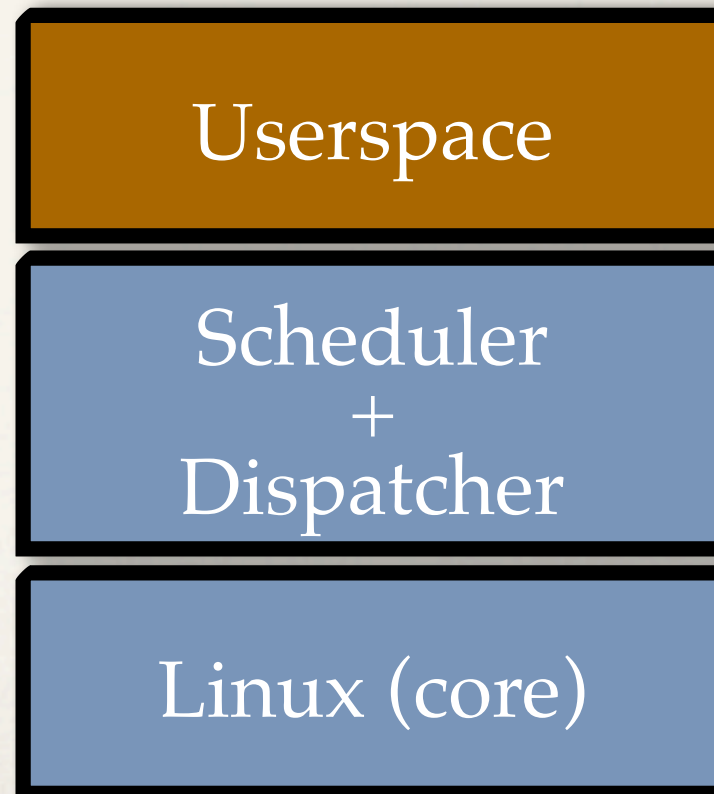


No changes in the measurement phase, no bias.

Study

Questions that We Address

Stock Linux



Questions that We Address

Stock Linux

Userspace

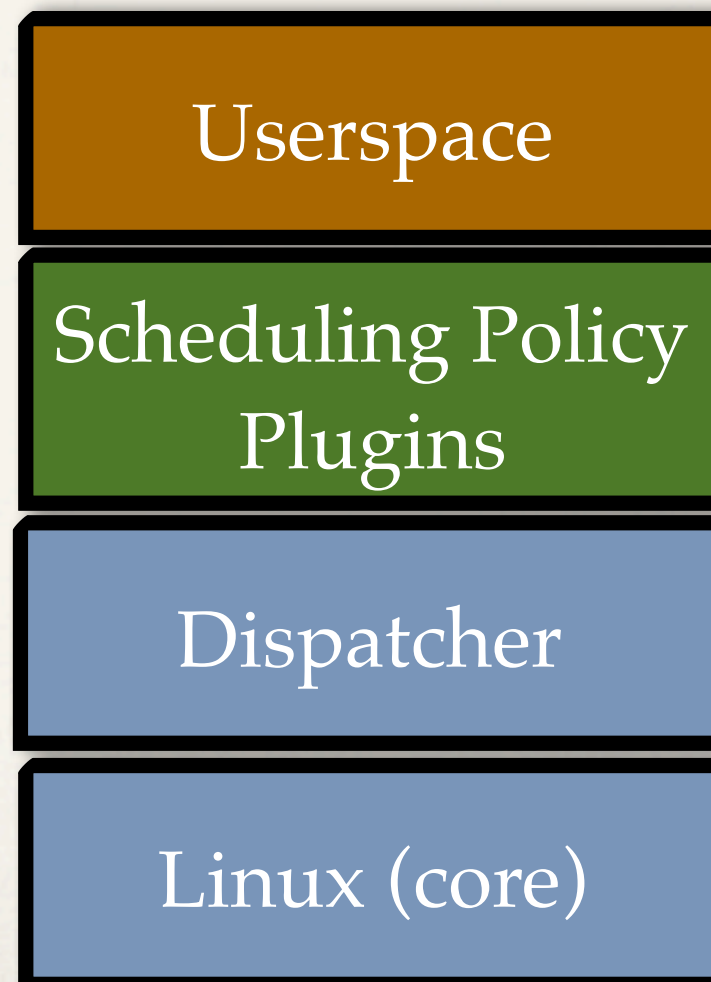
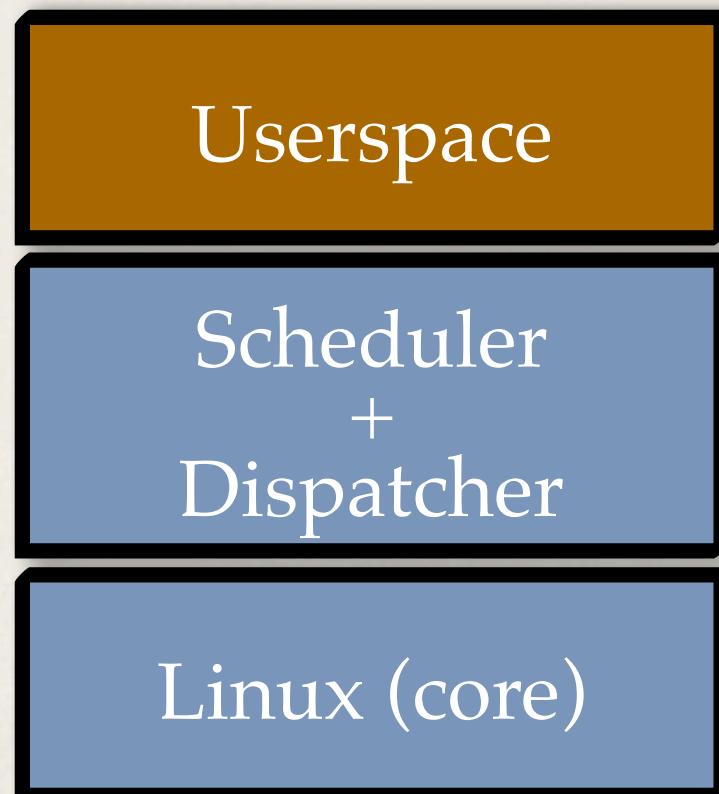
Scheduler
+
Dispatcher

Linux (core)

The Cost of LITMUS^{RT}

LITMUS^{RT}

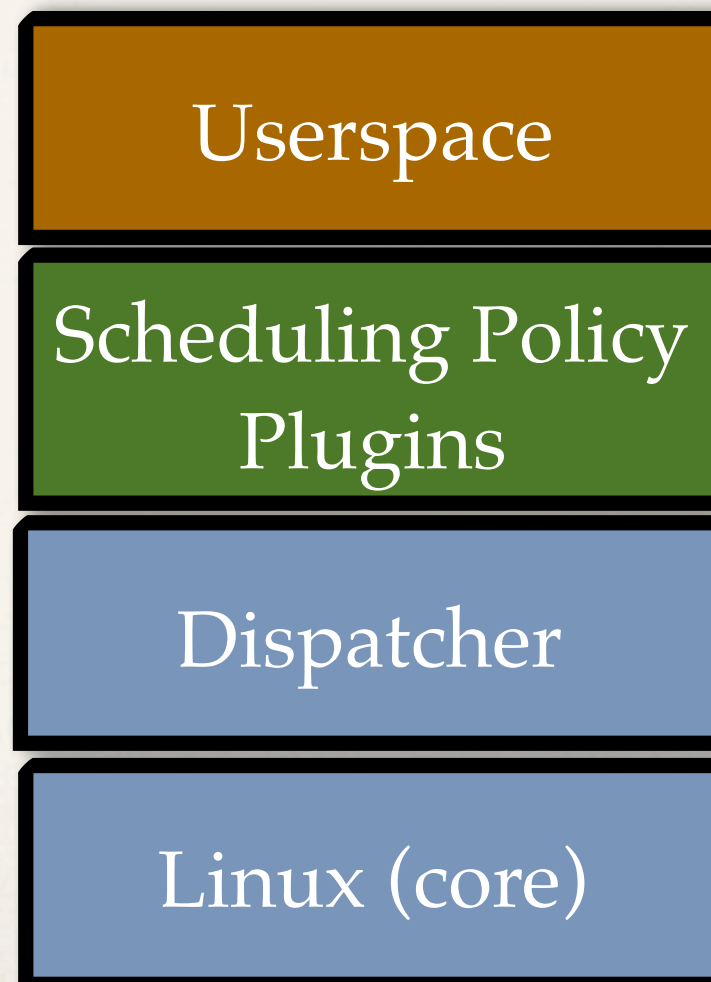
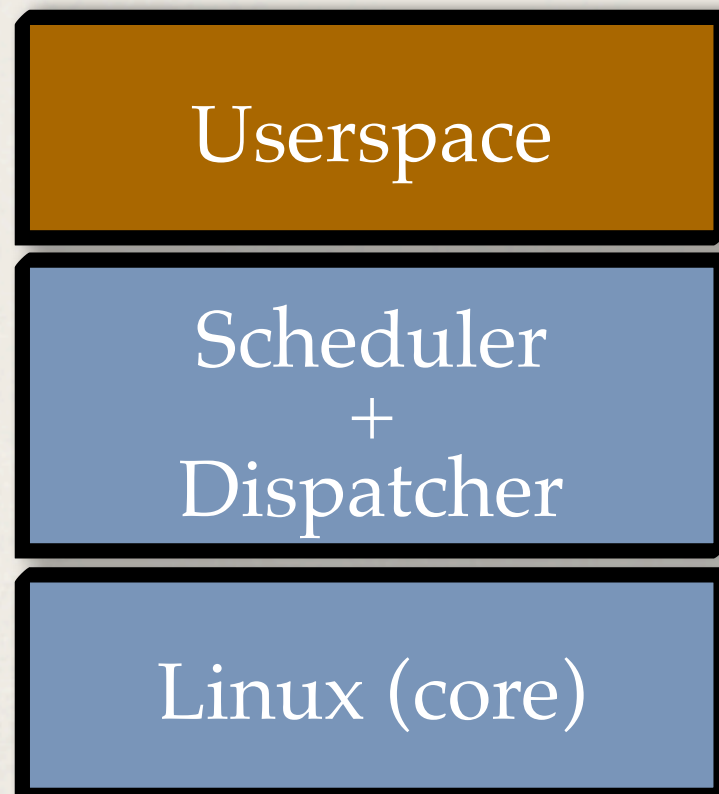
Stock Linux



The Cost of LITMUS^{RT}

LITMUS^{RT}

Stock Linux

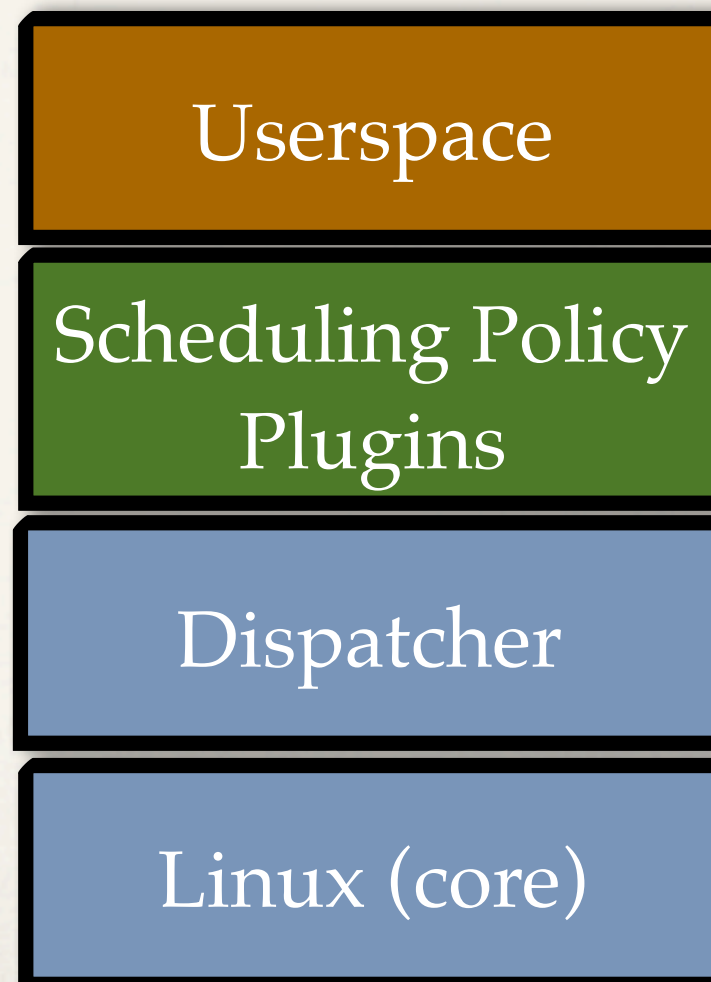
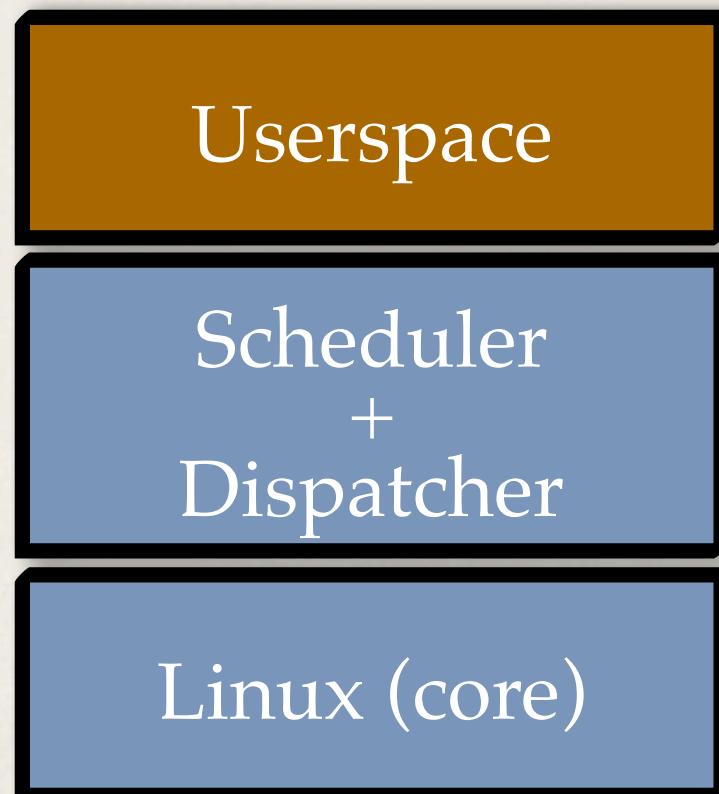


Question 1
How much latency does the scheduling policy interface add to the system?

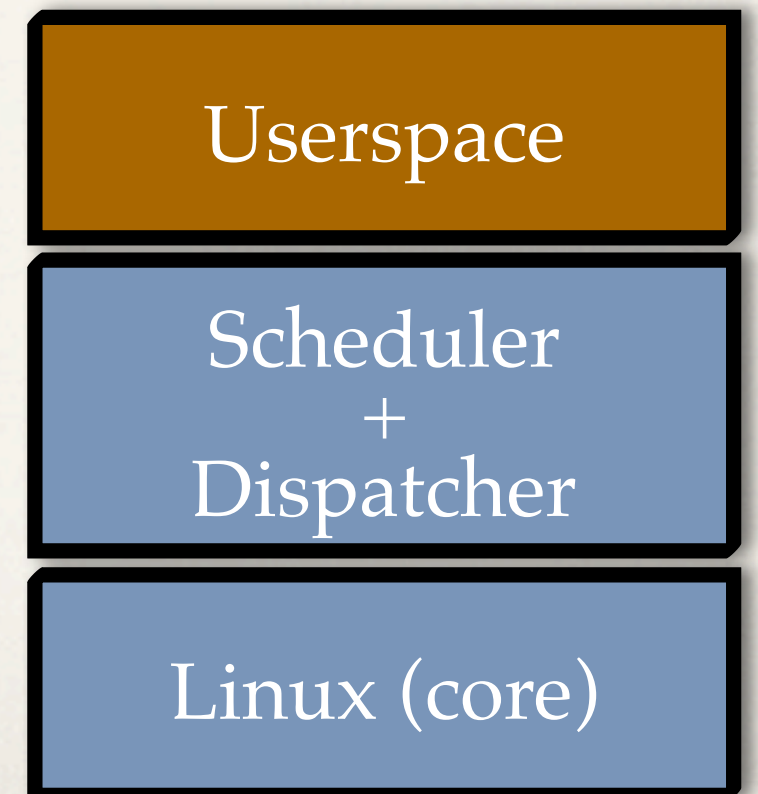
LITMUS^{RT} vs. PREEMPT_RT

LITMUS^{RT}

Stock Linux



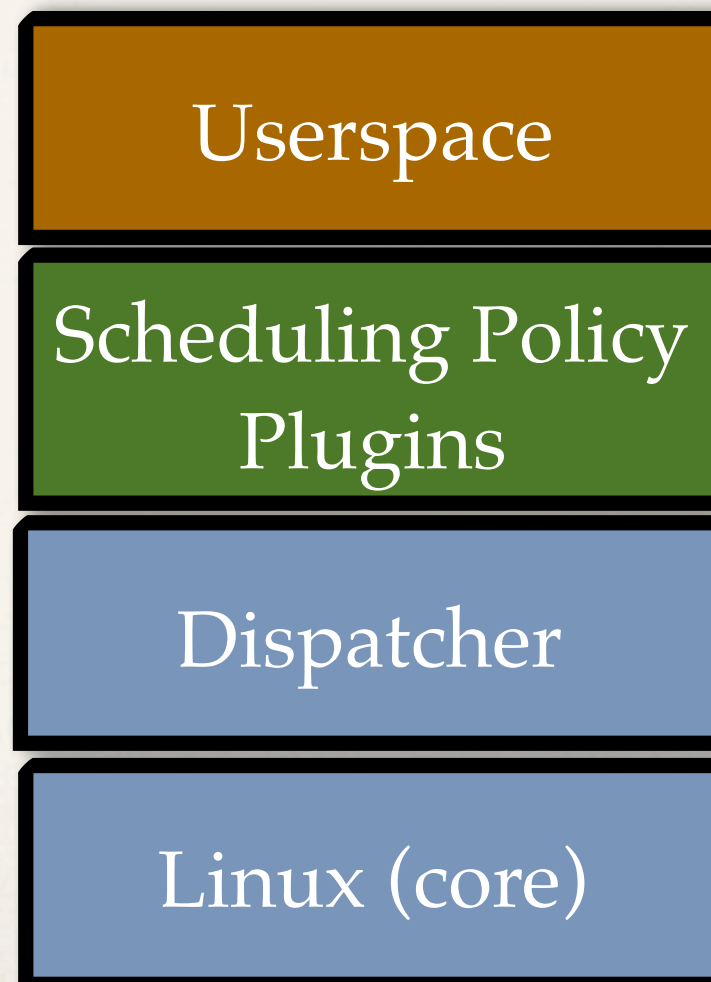
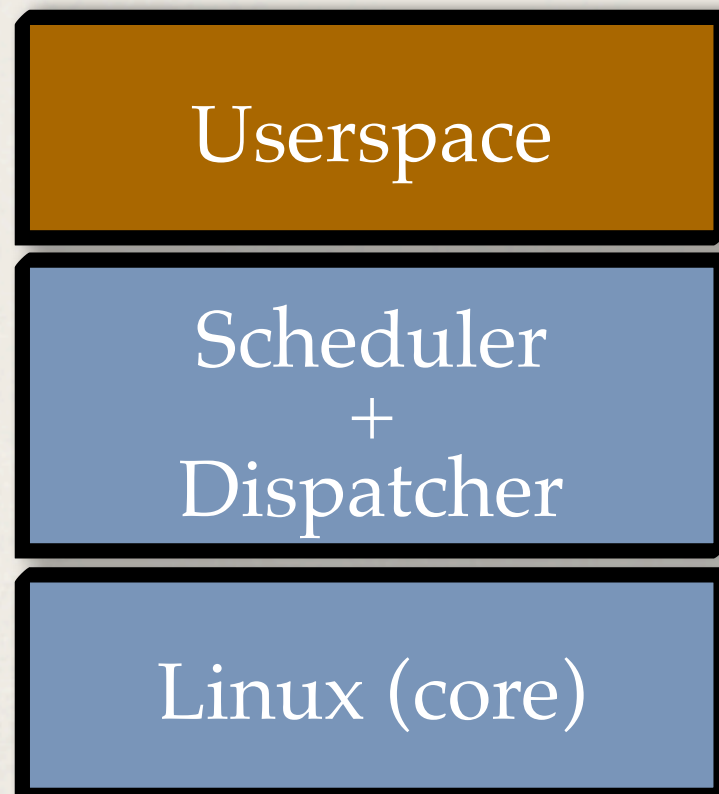
PREEMPT_RT



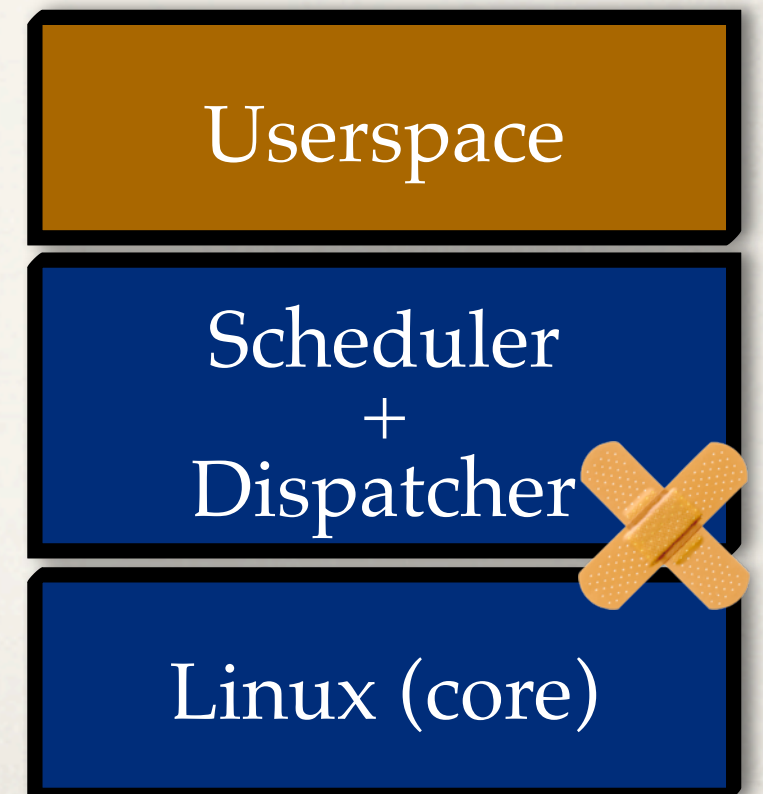
LITMUS^{RT} vs. PREEMPT_RT

LITMUS^{RT}

Stock Linux

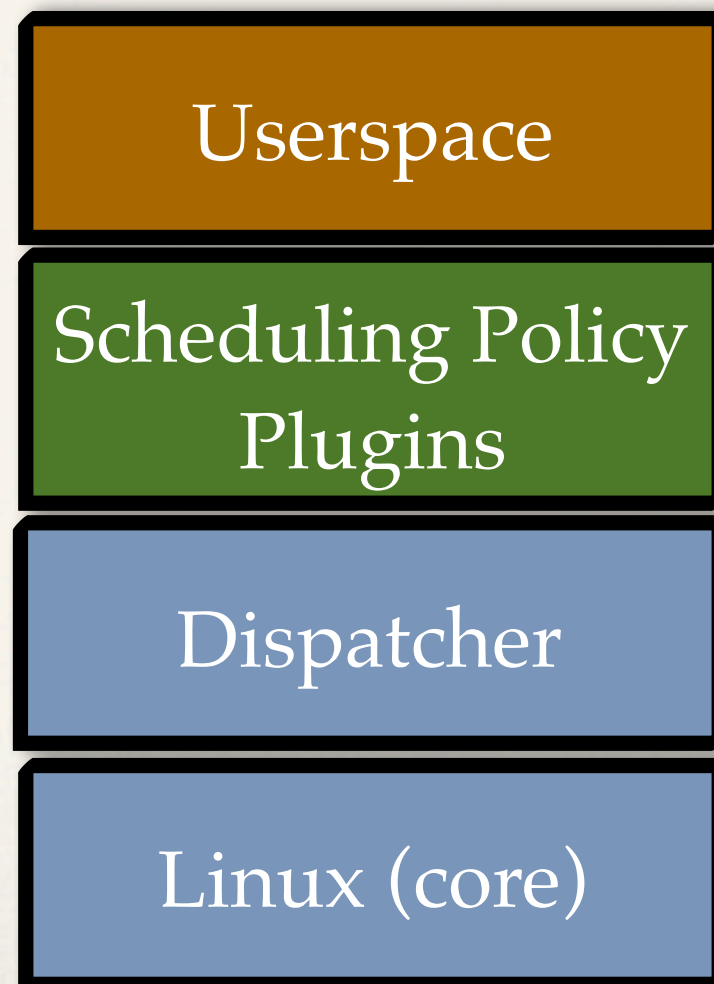


PREEMPT_RT

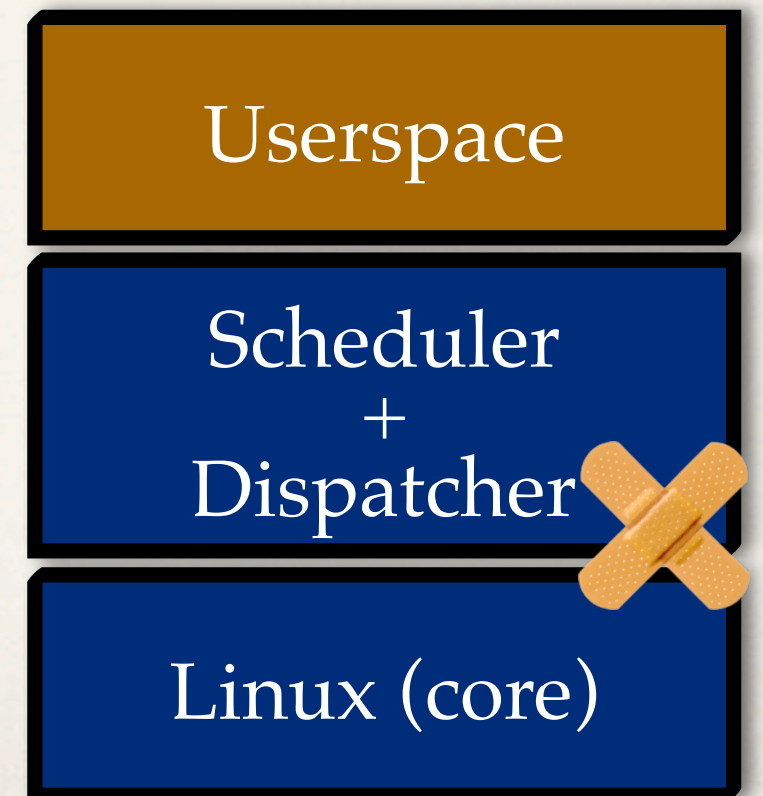


LITMUS^{RT} vs. PREEMPT_RT

LITMUS^{RT}



PREEMPT_RT

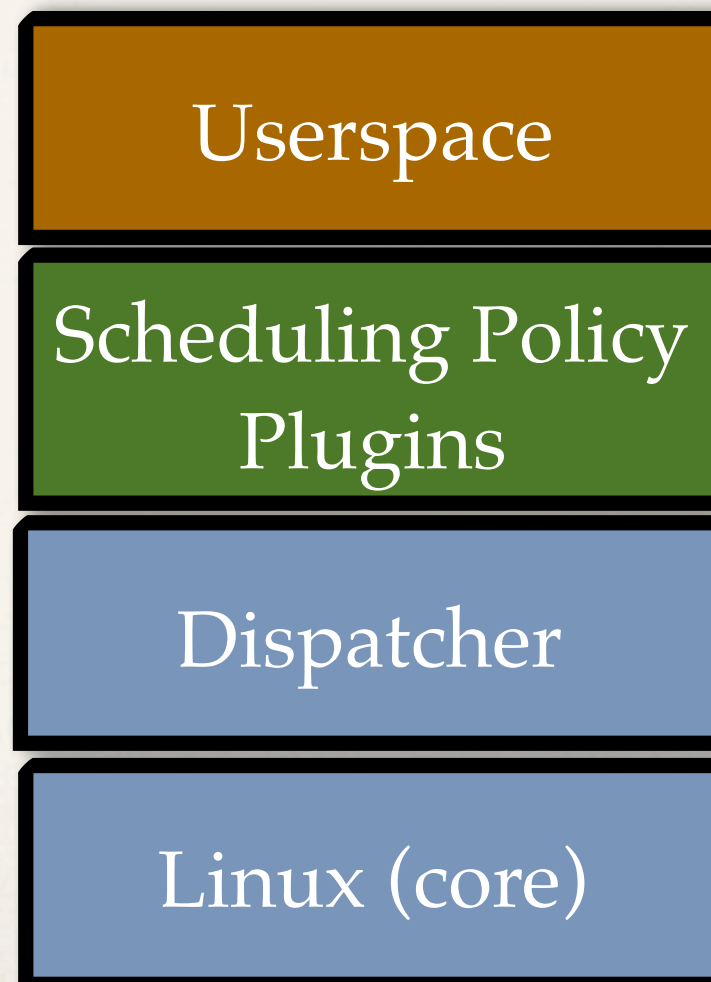


LITMUS^{RT} vs. PREEMPT_RT

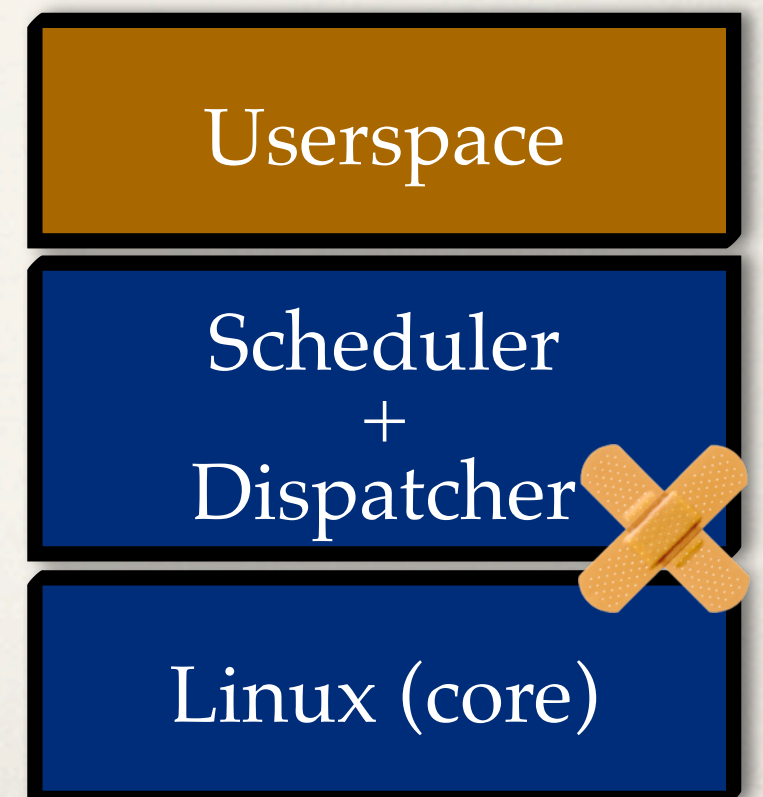
Question 2

What is the penalty for
LITMUS^{RT} not being
based on
PREEMPT_RT?

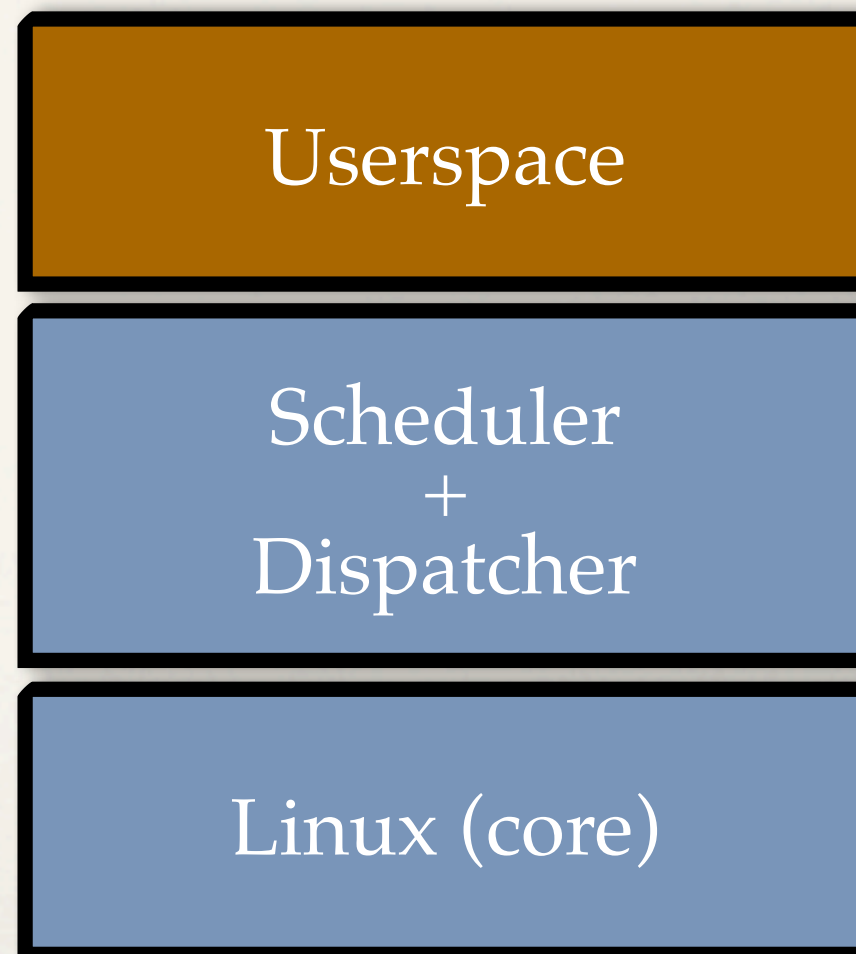
LITMUS^{RT}



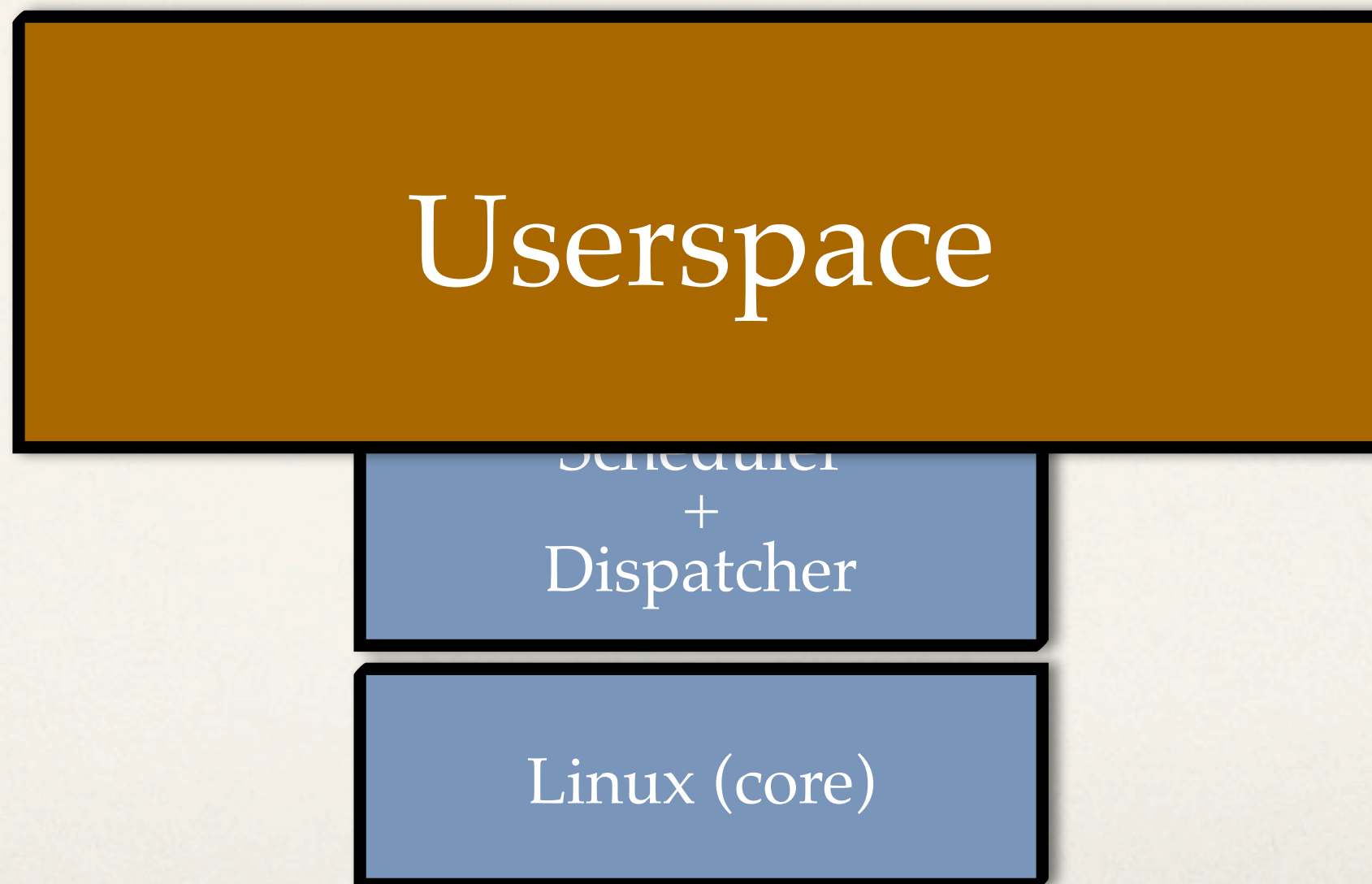
PREEMPT_RT



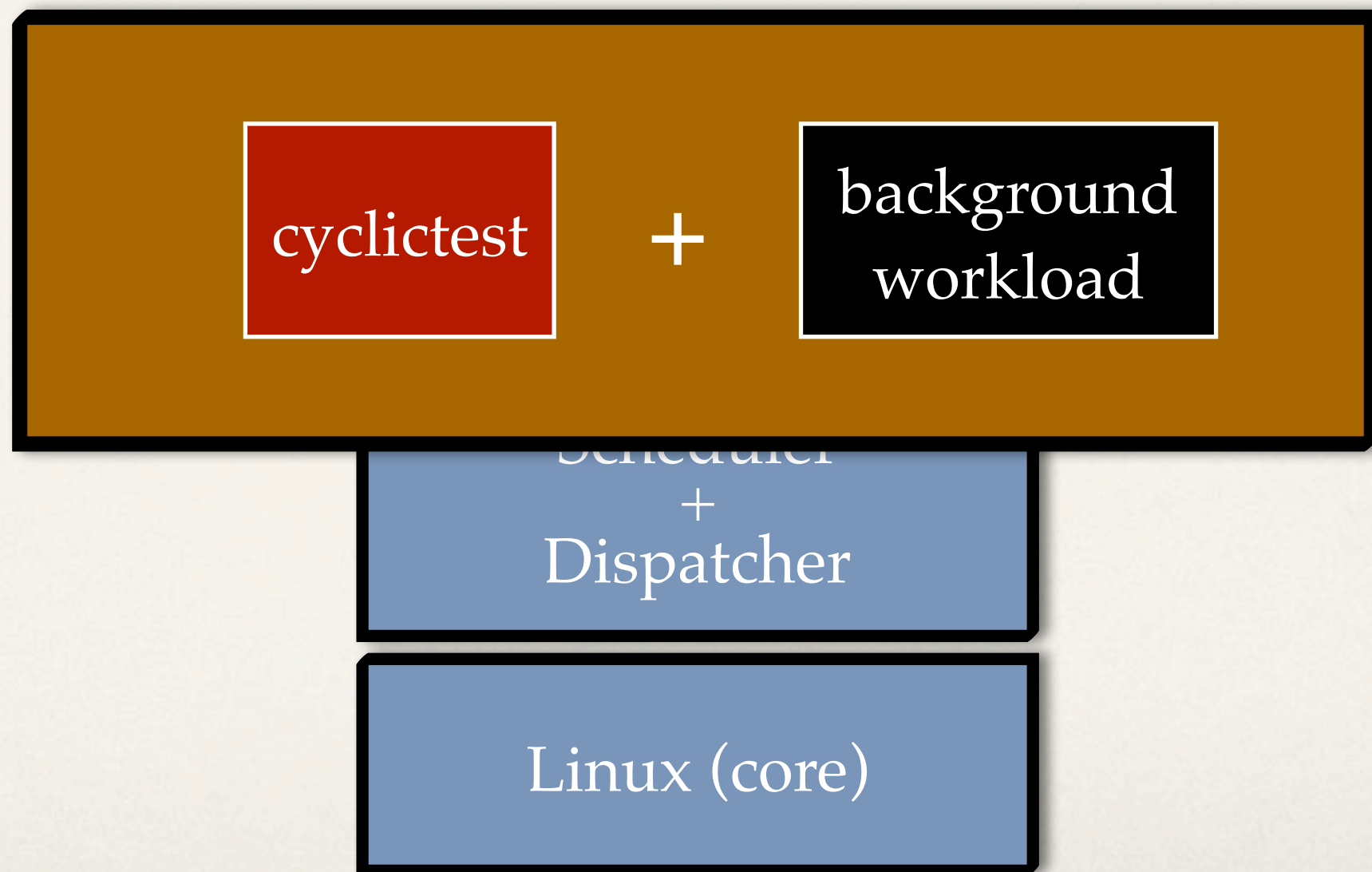
Evaluation



Evaluation



Evaluation



Background Workloads

NO

background tasks



**CPU-bound
background tasks**



**I/O-bound
background tasks**

Experimental Setup

Different kernels:

1. **LITMUS^{RT}** (Linux 3.0)

Partitioned **F**ixed **P**riority (P-FP),
Partitioned **EDF** with synchronization support (PSN-EDF),
Global **EDF** with synchronization support (GSN-EDF)

2. **PREEMPT_RT** (Linux 3.8.13)

3. **Unpatched Linux 3.0 and Linux 3.8.13**

} SCHED_FIFO

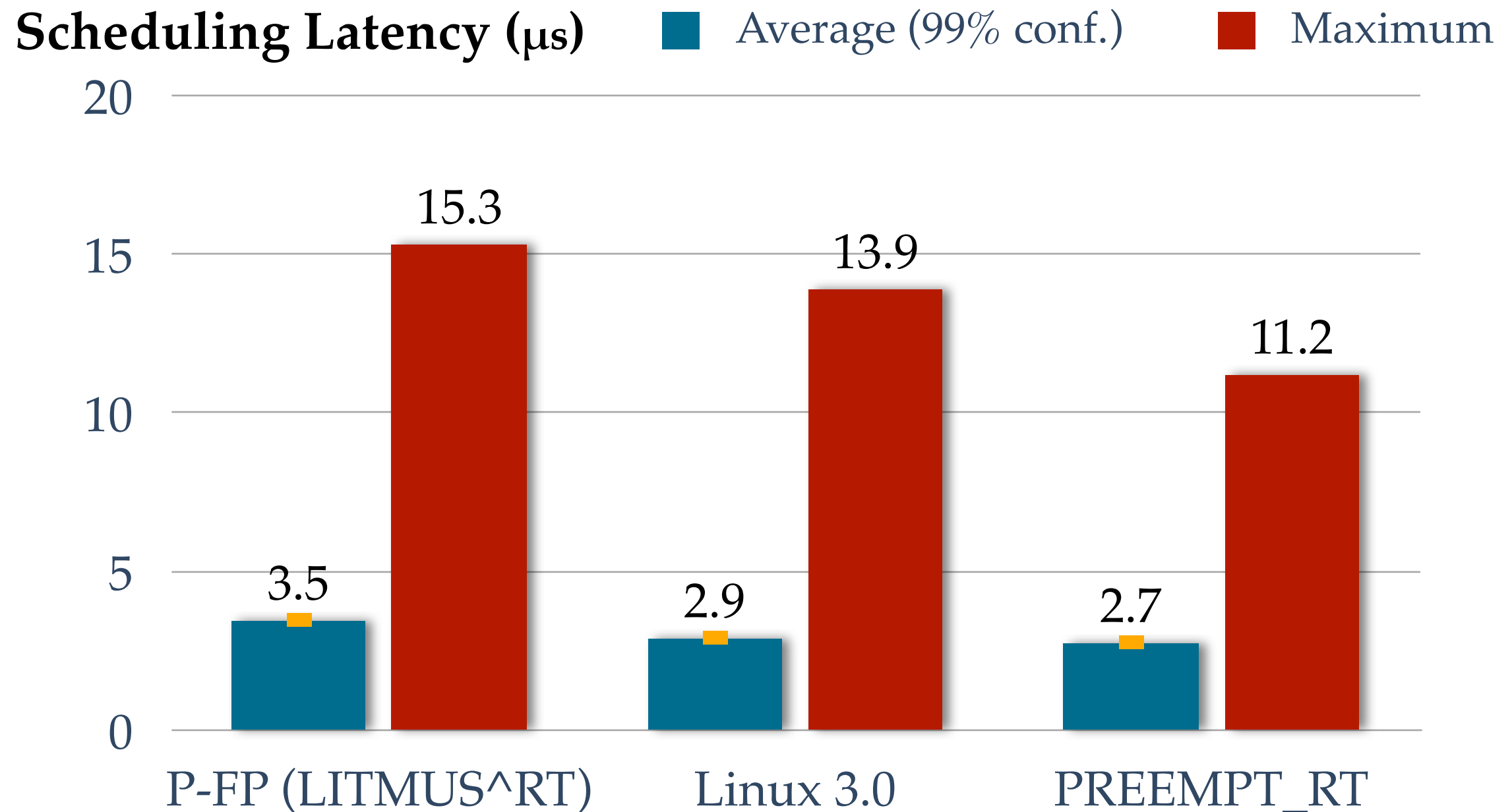
Experimental Setup

- ❖ **16-core Intel Xeon platform**
- ❖ **cyclicttest's standard setup:**
 - ❖ one real-time task per processor
 - ❖ periods: {1000, 1500, 2000, ...} μs
- ❖ **Duration: 20 minutes per experiment**
 - ❖ Almost 6 million samples for each case
- ❖ **Results shown in microseconds**

First Scenario

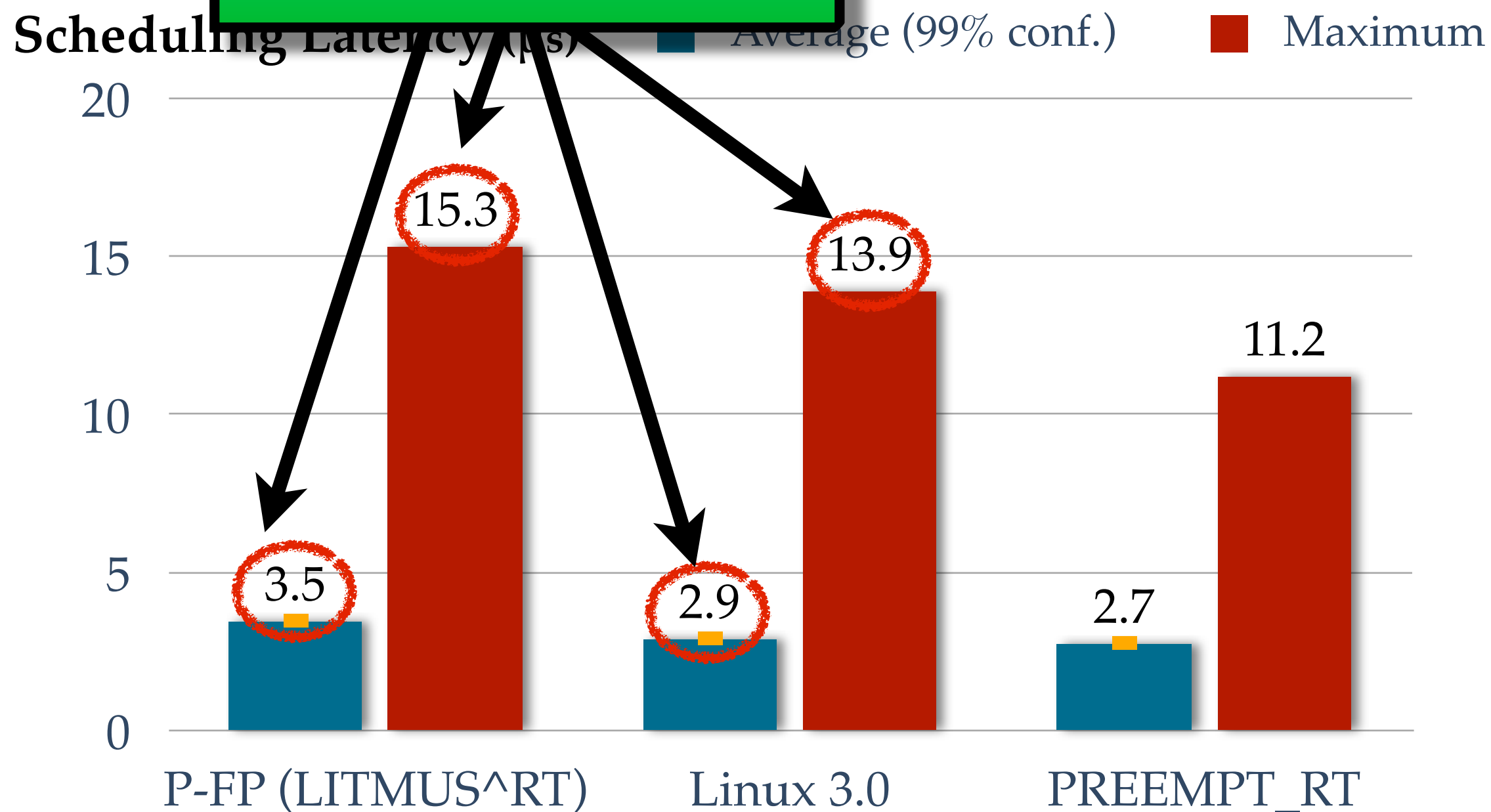
NO
background tasks

No Background Tasks



No Backlogs

Similar max. and avg. latency for Linux 3.0 and LITMUS^{RT}.

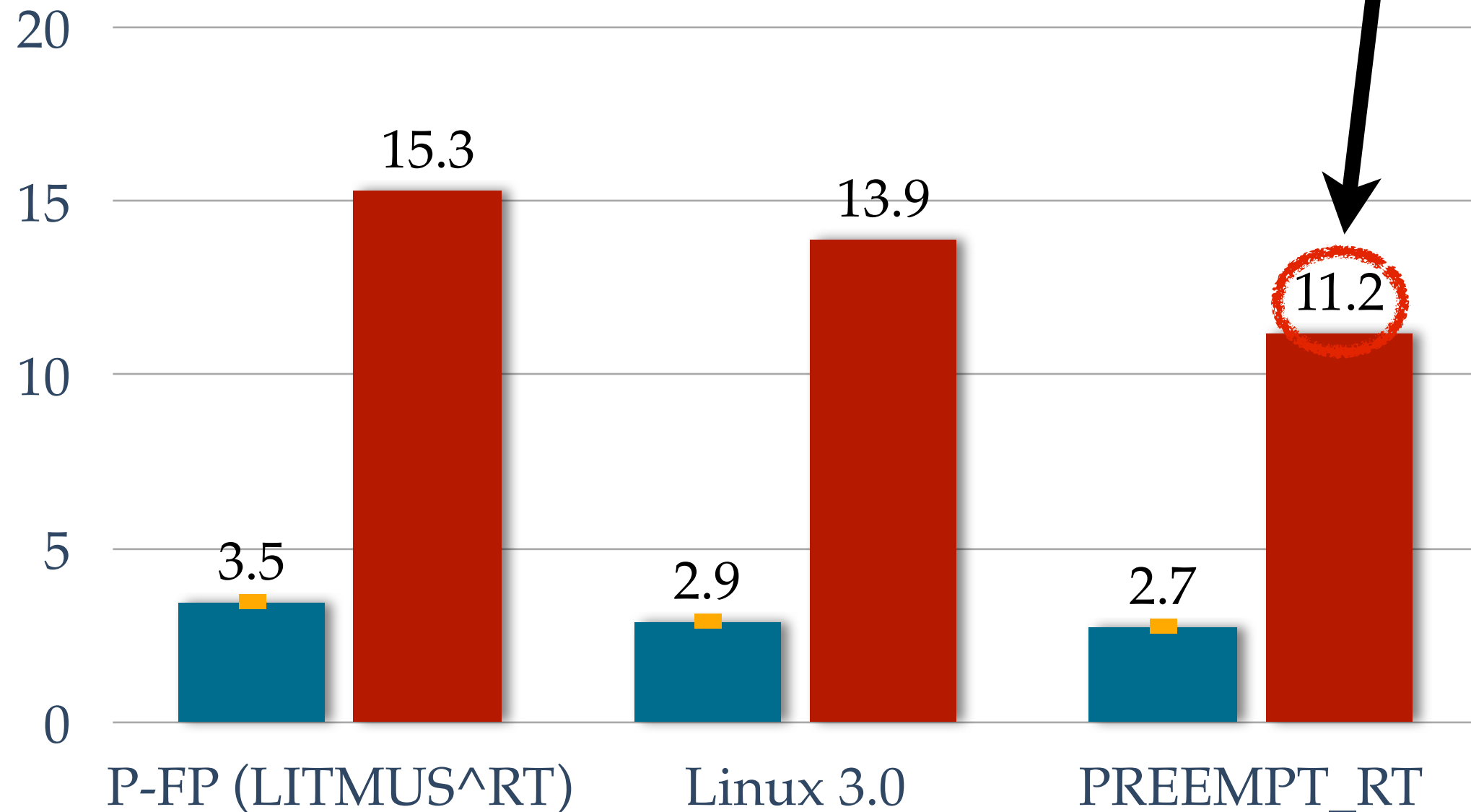


No Backlogs

Similar max. and avg. latency for Linux 3.0 and LITMUS^{RT}.

Improved max. latency for PREEMPT_RT

Scheduling Latency (μ s) ■ Average (99% conf.) ■ Maximum



Second Scenario



CPU-bound background tasks

- ❖ Tasks running an infinite loop accessing memory (read/write)
- ❖ Working set larger than L2 cache size

Second Scenario



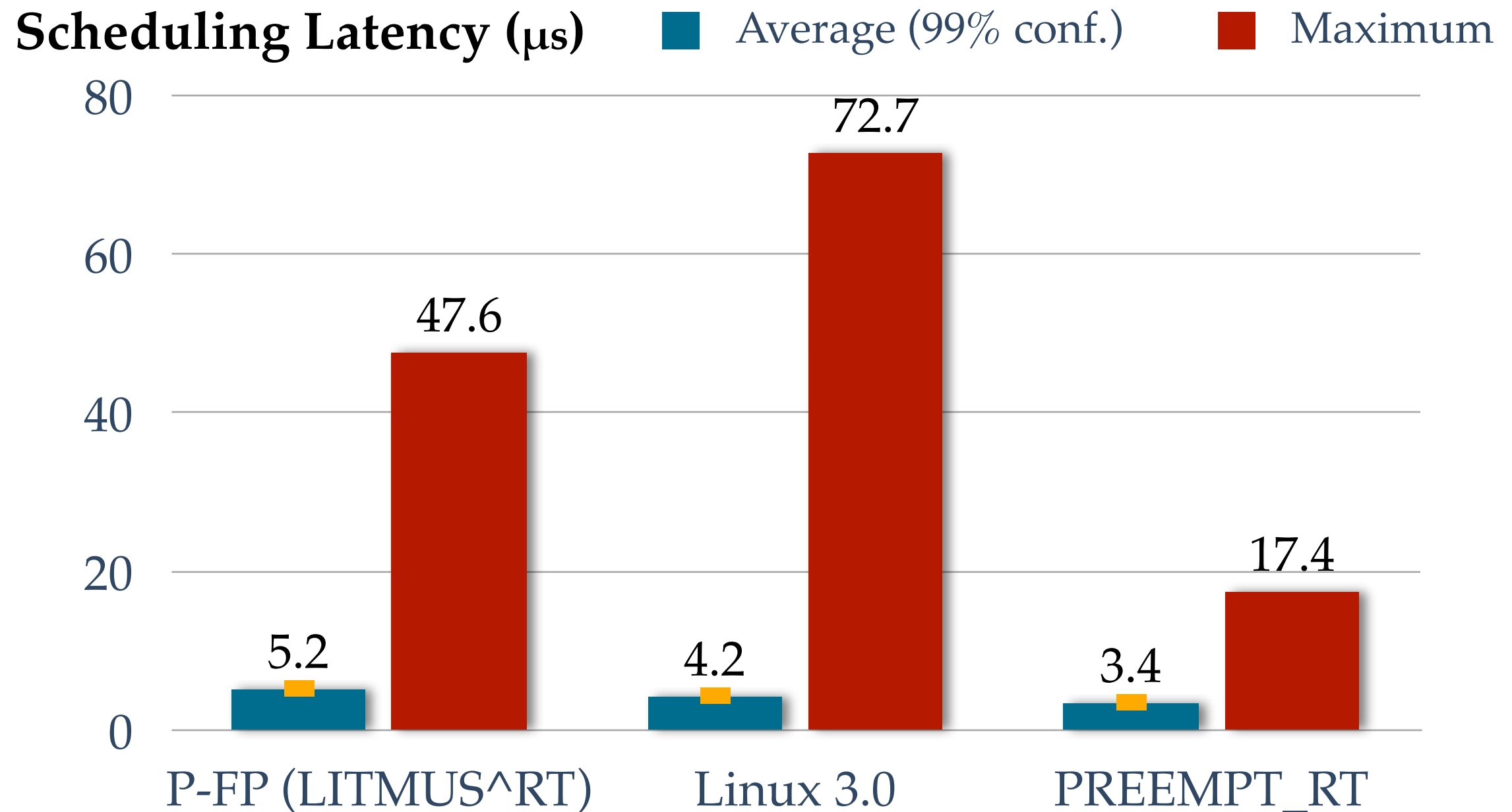
**CPU-bound
background tasks**

- ❖ Tasks running an infinite loop accessing memory (read/write)
- ❖ Working set larger than L2 cache size



**Generates memory traffic
and cache contention!**

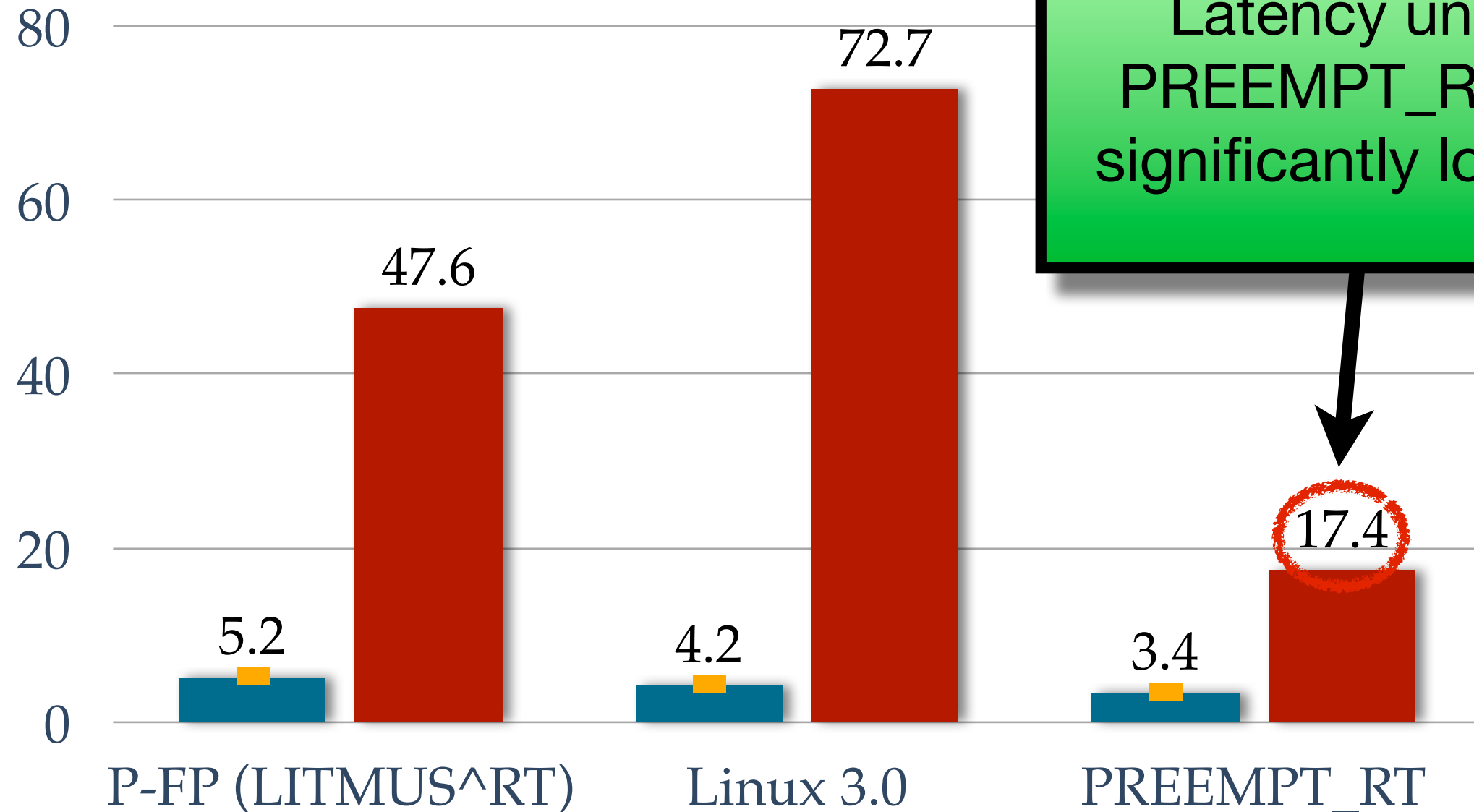
CPU-bound Background Tasks



CPU-bound Background Tasks

Scheduling Latency (μs)

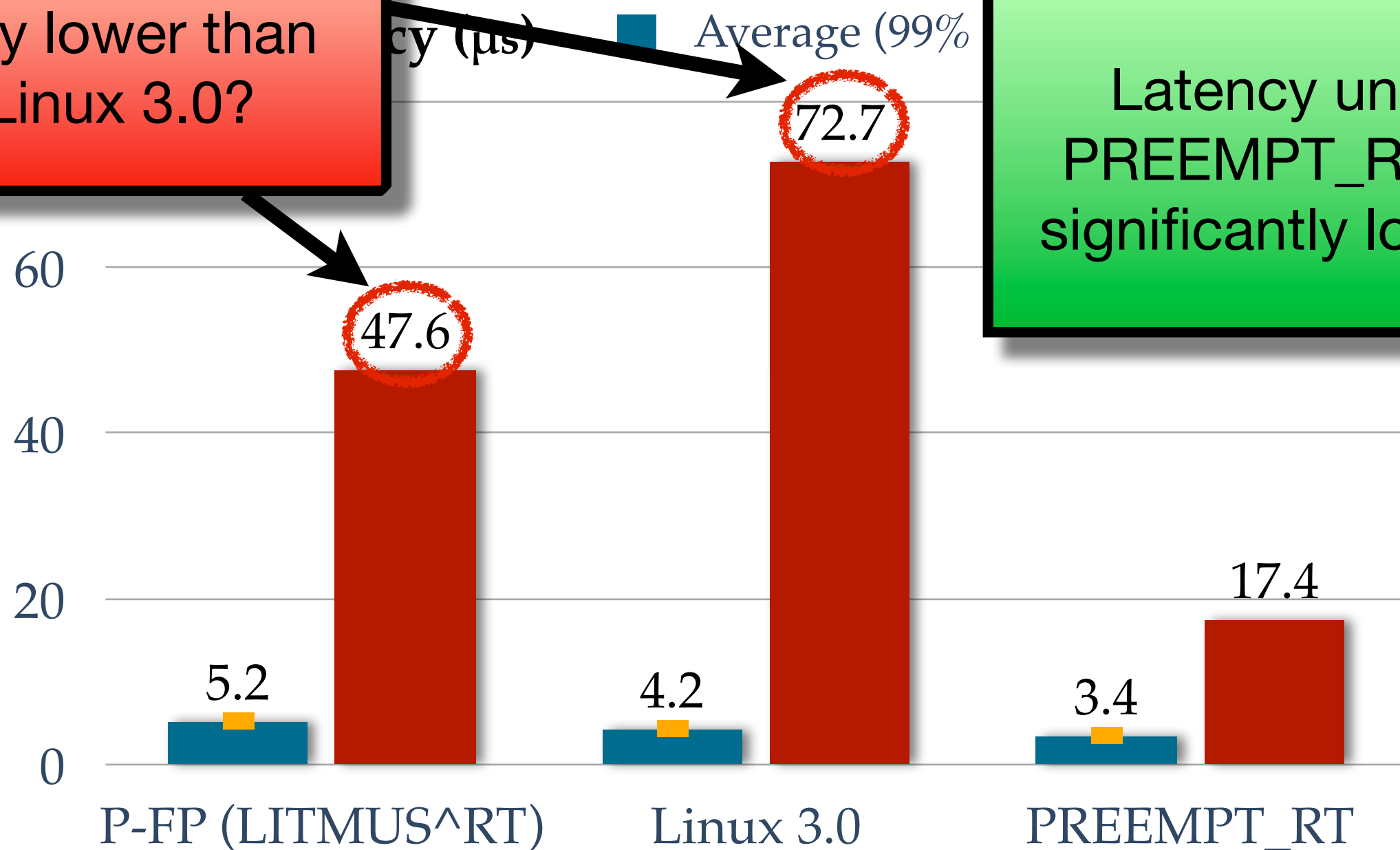
■ Average (99%)



Latency under
PREEMPT_RT is
significantly lower.

CPU-bound Background Tasks

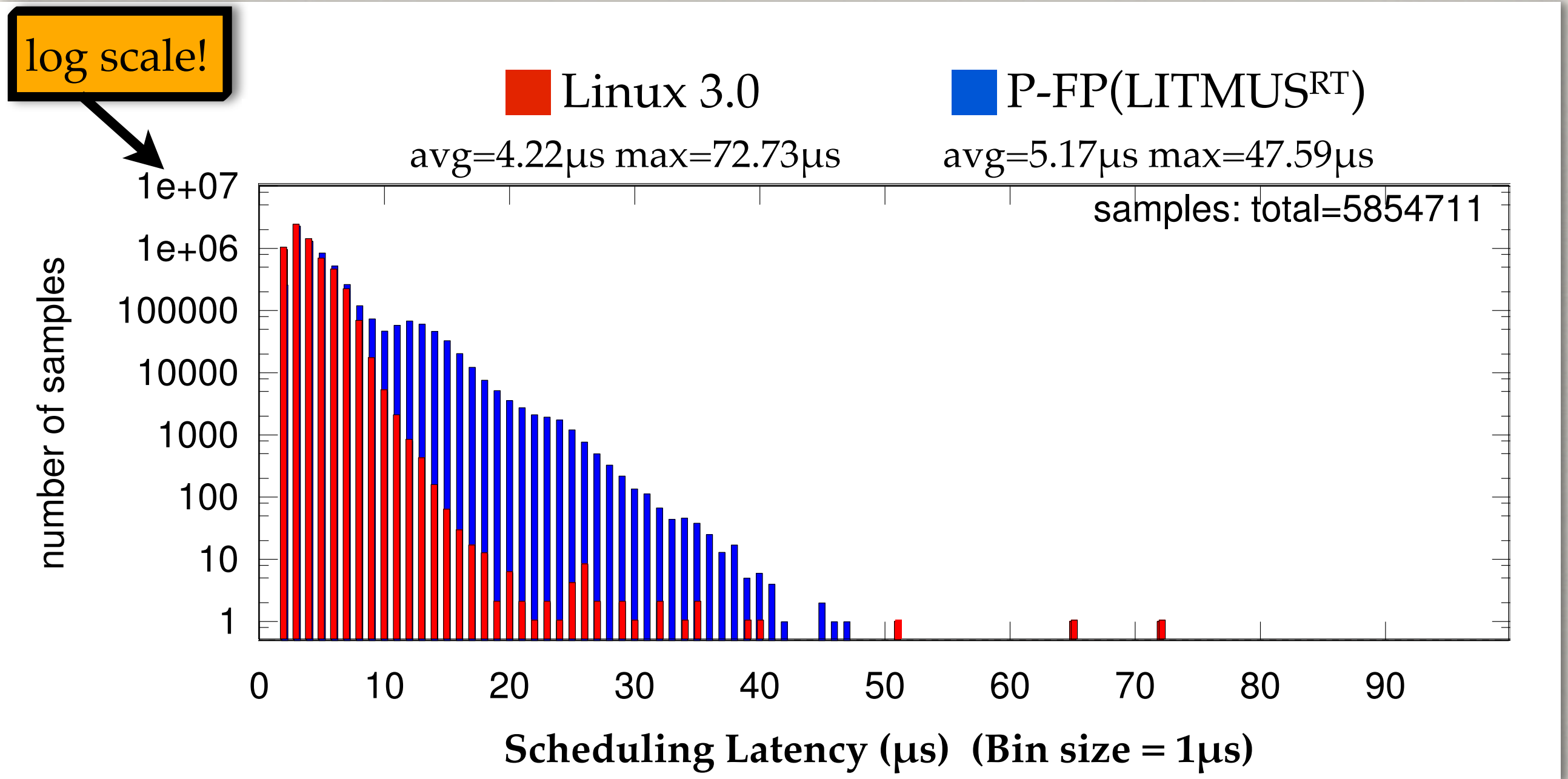
LITMUS^{RT}'s
latency lower than
on Linux 3.0?



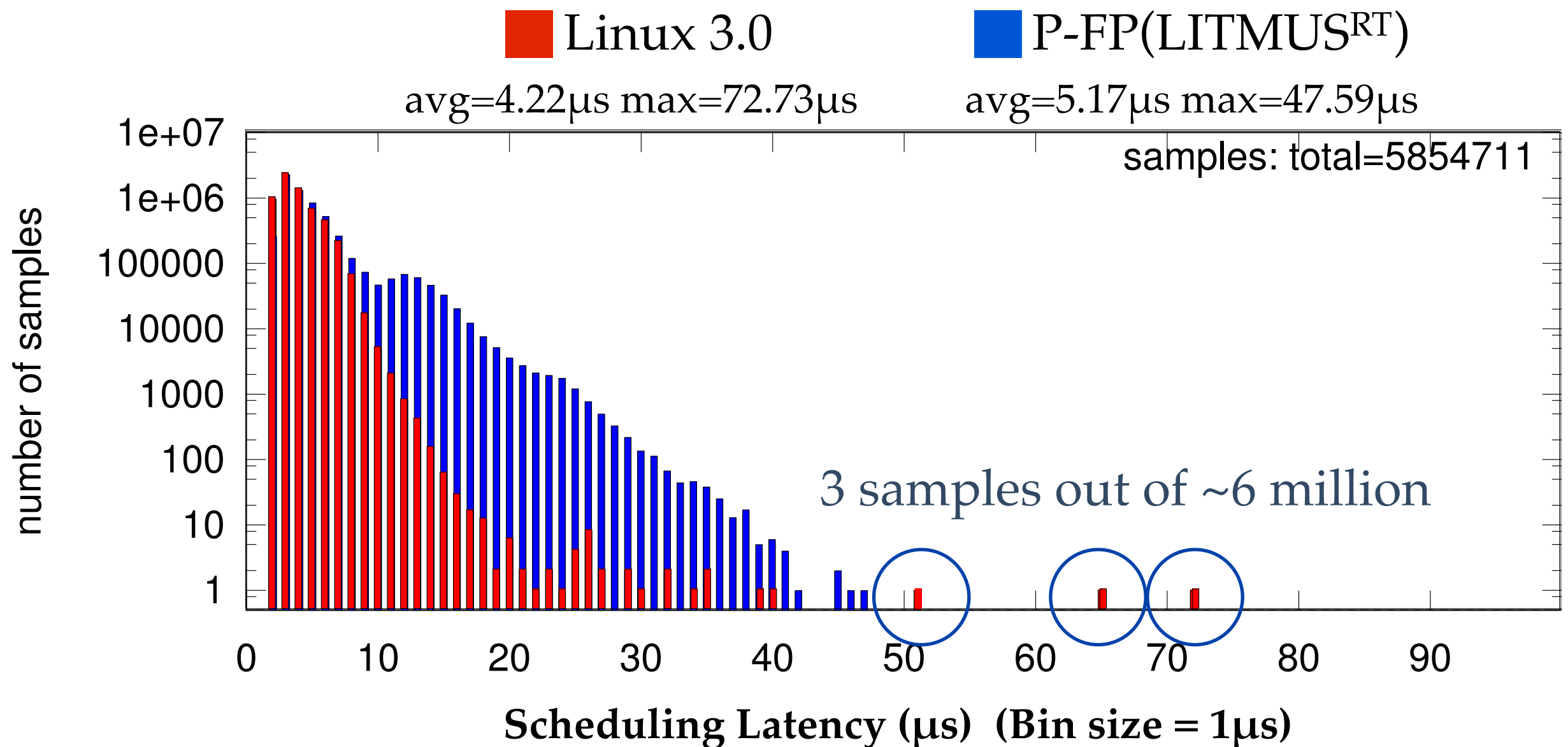
Latency under
PREEMPT_RT is
significantly lower.

LITMUS^{RT} vs. Linux 3.0:

CPU-bound Background Tasks

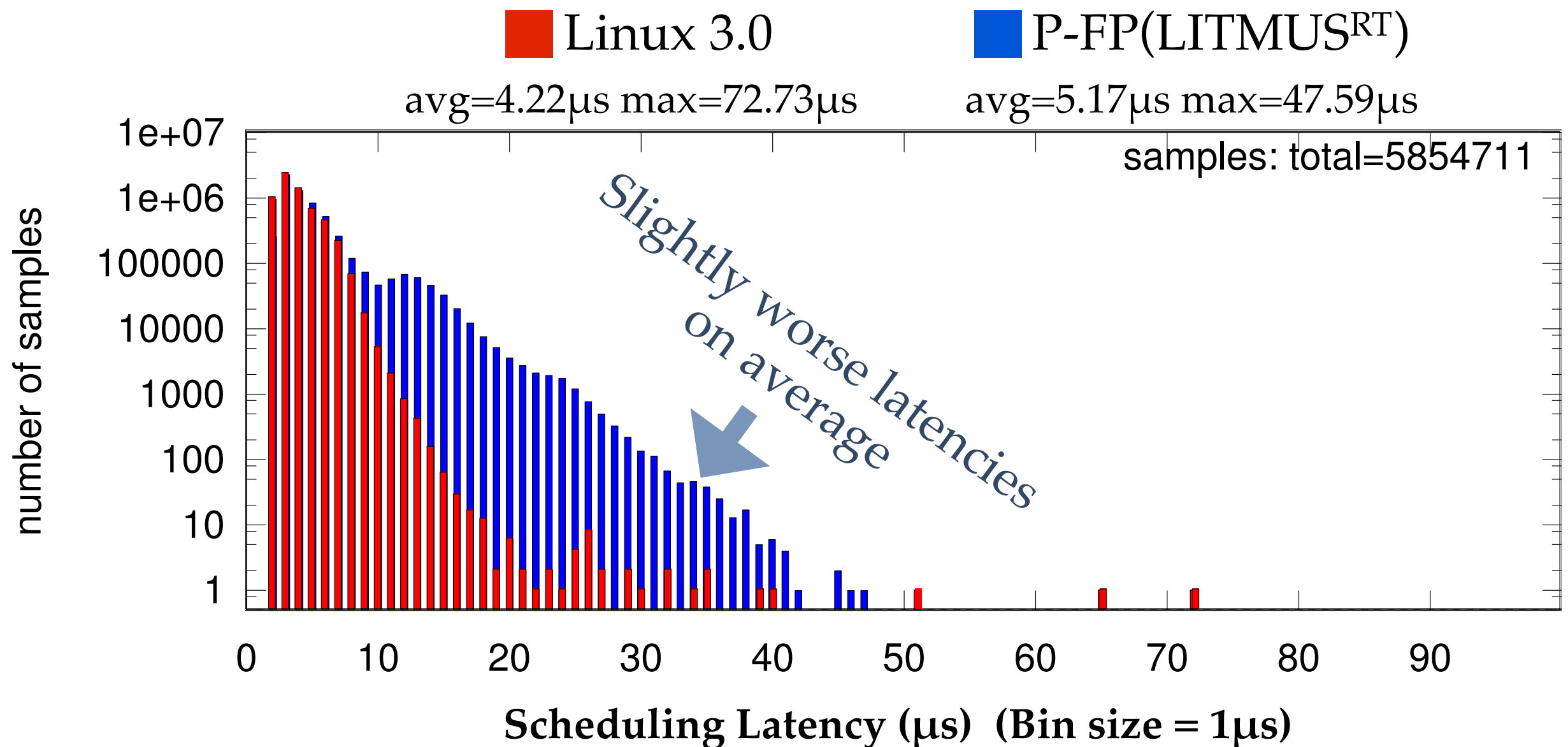


LITMUS^{RT} vs. Linux 3.0: CPU-bound Background Tasks

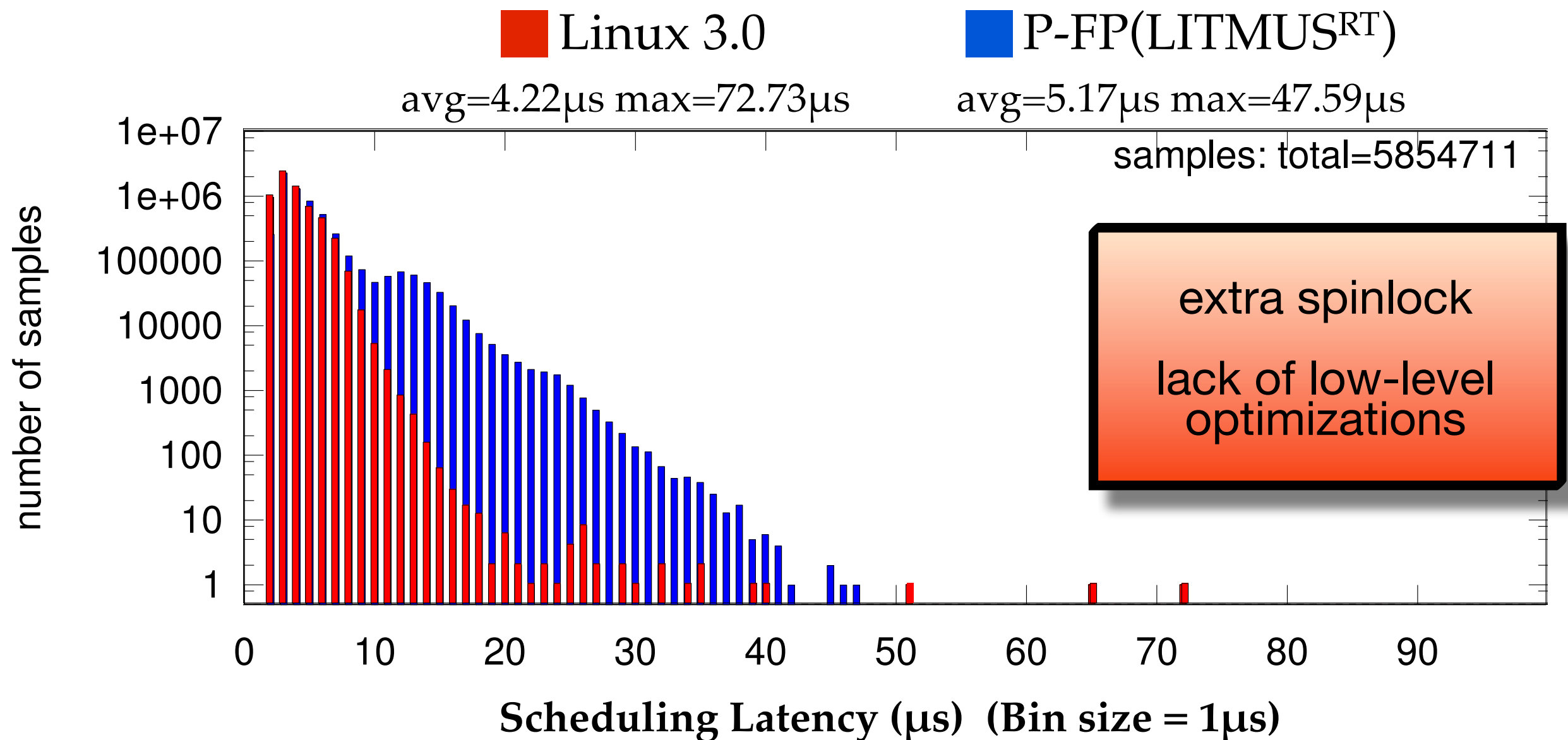


LITMUS^{RT} vs. Linux 3.0:

CPU-bound Background Tasks



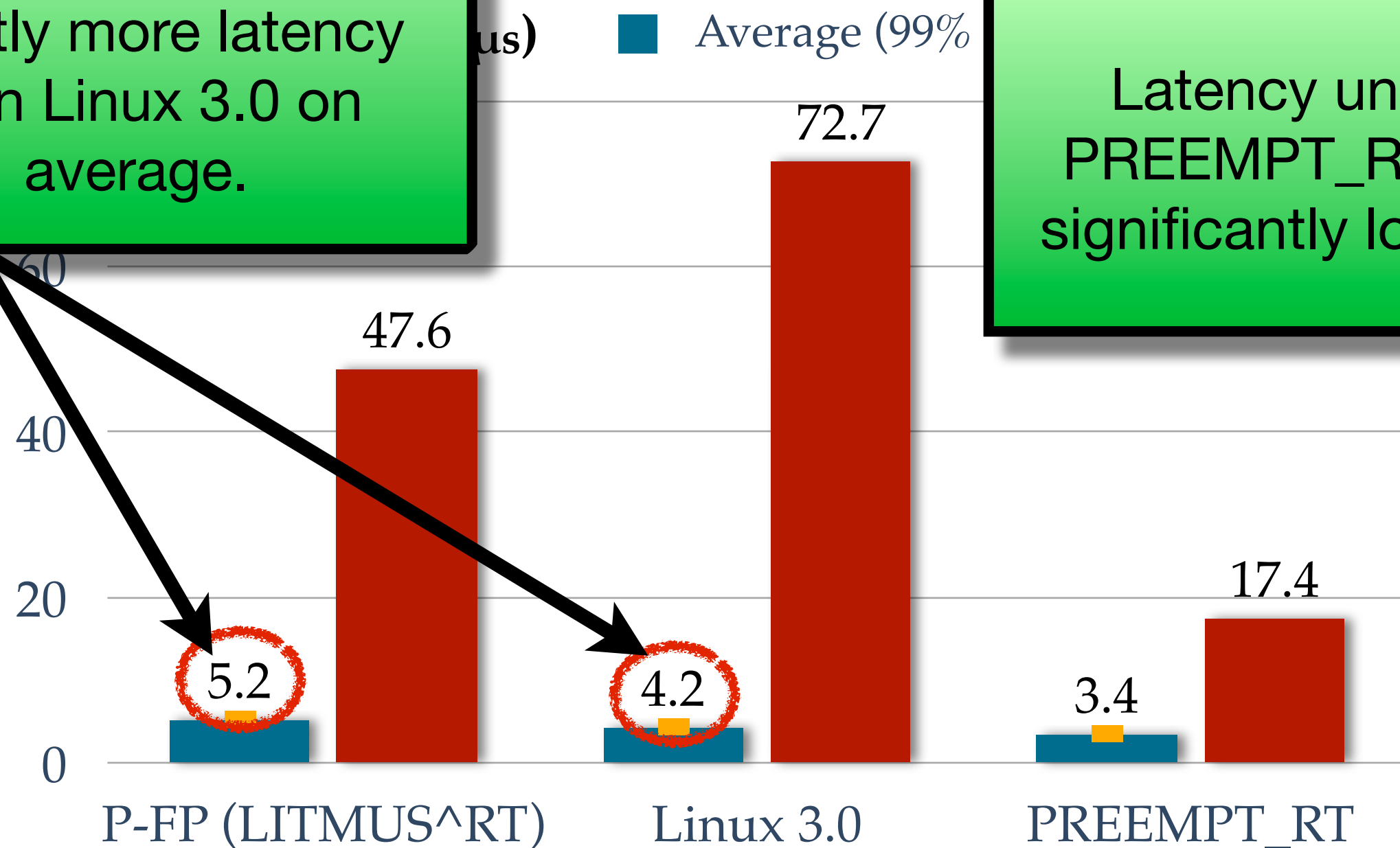
LITMUS^{RT} vs. Linux 3.0: CPU-bound Background Tasks



CPU-bound Background Tasks

LITMUS^{RT} incurs slightly more latency than Linux 3.0 on average.

Latency under PREEMPT_RT is significantly lower.



Third Scenario



**I/O-bound
background tasks**

- ❖ **hackbench:** Linux scheduler stress tool
- bonnie++:** Disk and file system benchmark
- wget:** Network activity

Third Scenario



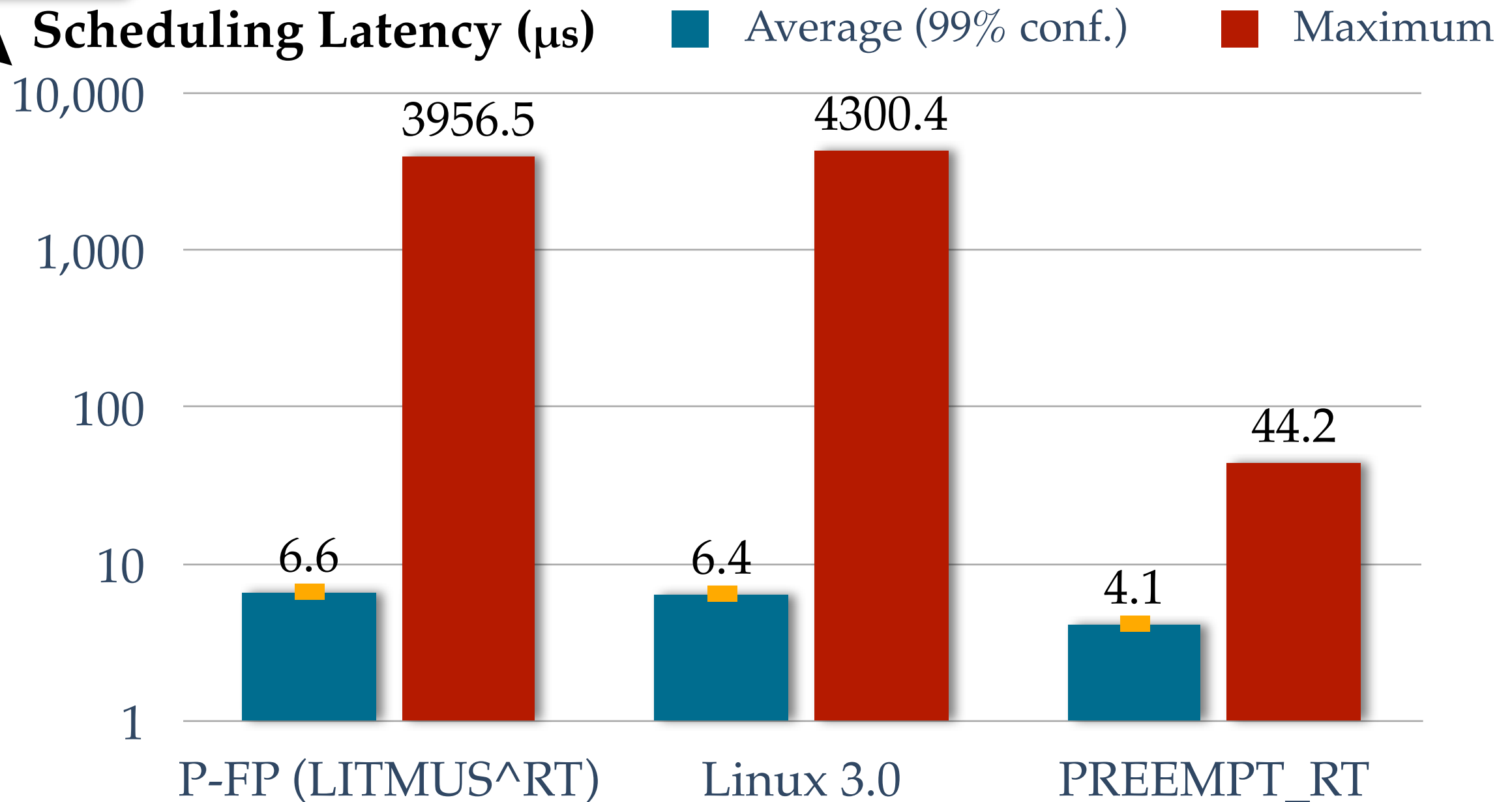
**I/O-bound
background tasks**

→ Causes a lot of system
calls and interrupts

- ❖ **hackbench**: Linux scheduler stress tool
- bonnie++**: Disk and file system benchmark
- wget**: Network activity

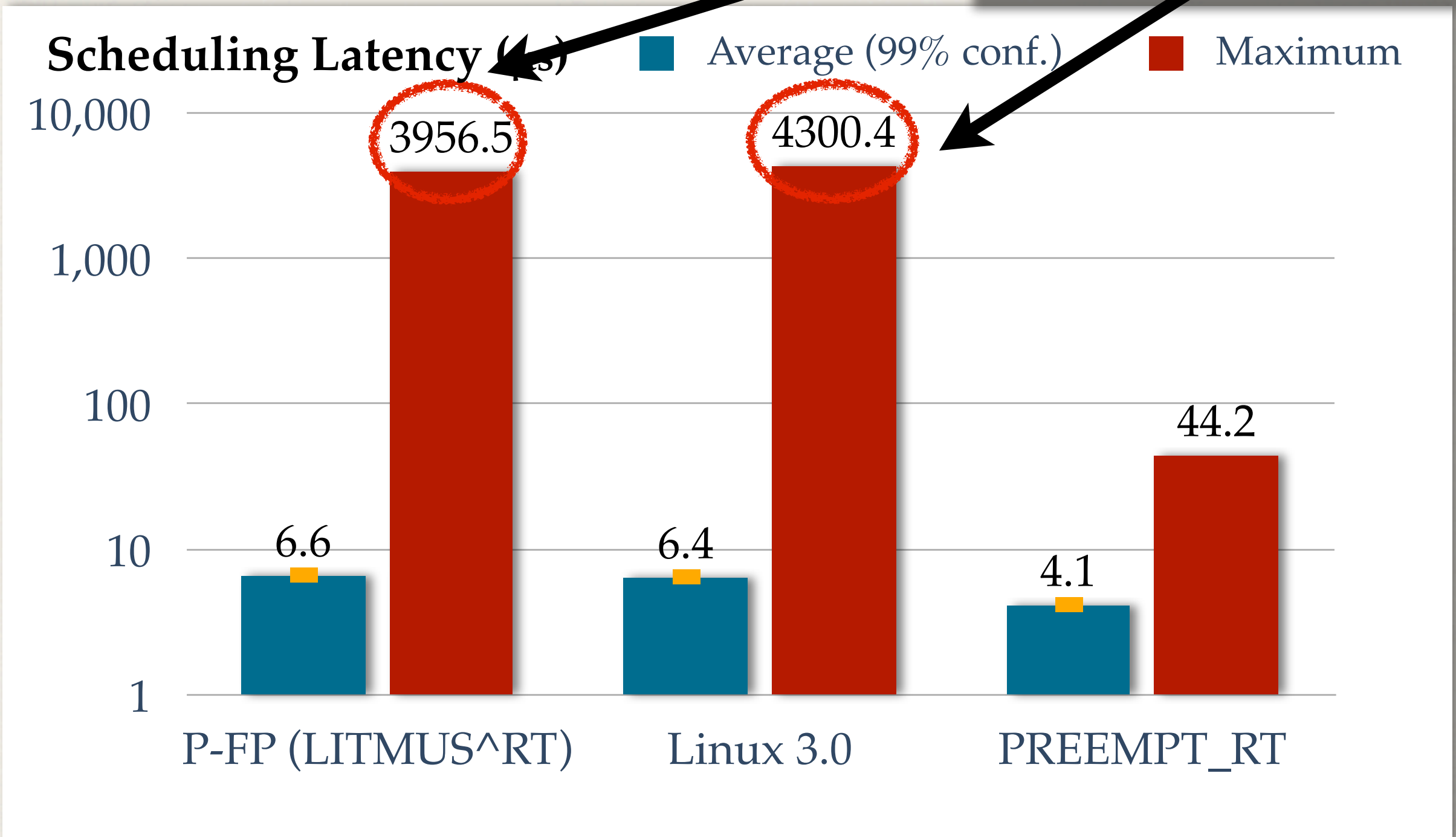
I/O-bound Background Tasks

log scale!



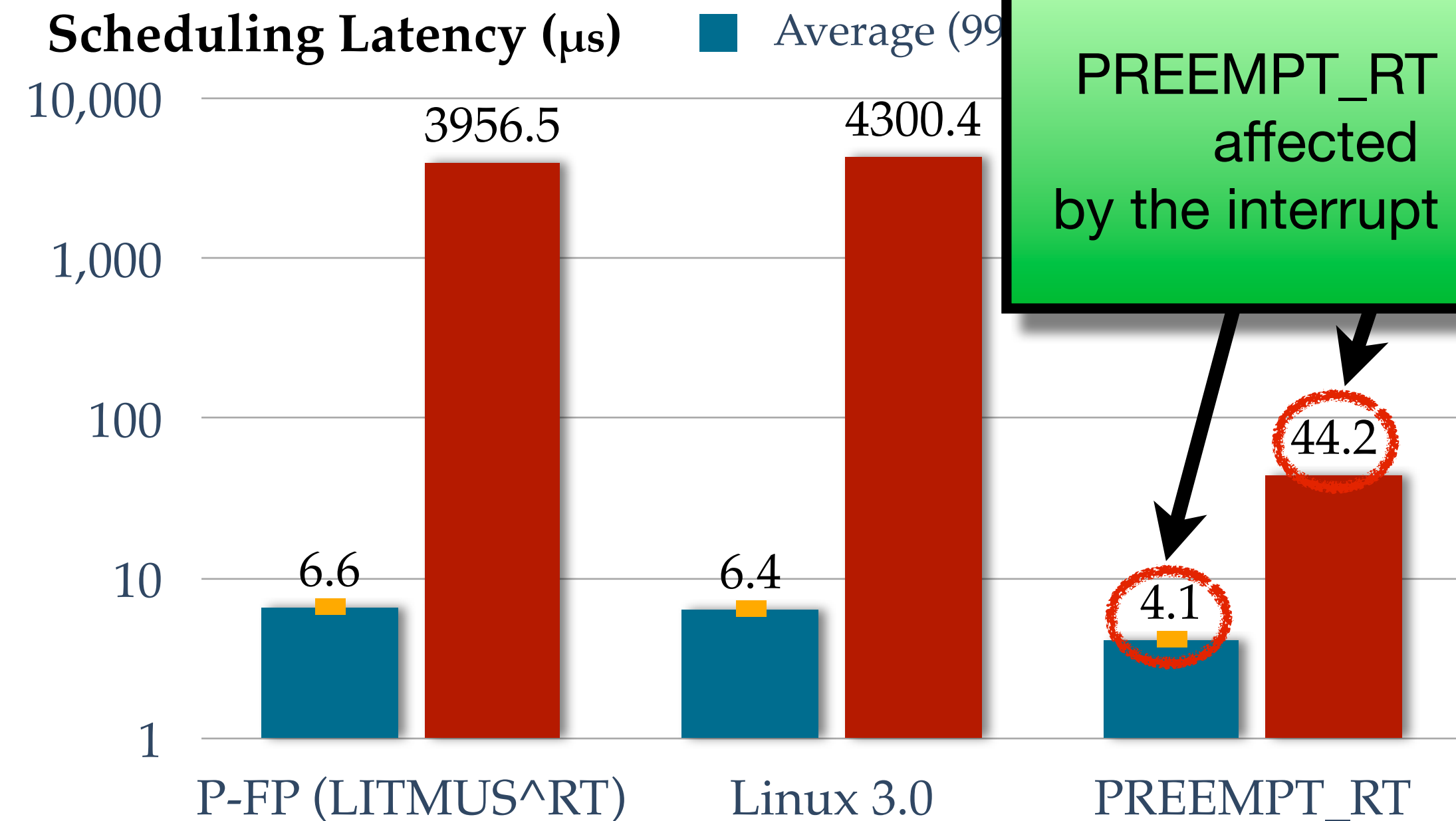
I/O-bound Backgrou

Huge impact on scheduling latency under standard Linux.



I/O-bound Backgrou

Huge impact on scheduling latency under standard Linux.



PREEMPT_RT is not affected by the interrupt load!

Summary

1. Cost of the scheduling plugin layer

2. LITMUS^{RT} vs. PREEMPT_RT

Summary

1. Cost of the scheduling plugin layer

The overhead introduced by LITMUS^{RT} is **small** ✓

2. LITMUS^{RT} vs. PREEMPT_RT

Summary

1. Cost of the scheduling plugin layer

The overhead introduced by LITMUS^{RT} is **small**



2. LITMUS^{RT} vs. PREEMPT_RT

PREEMPT_RT **significantly** decreases scheduling latency.



Importance of Feather-Trace

- ❖ cyclicttest was ported to LITMUS^{RT}.
- ❖ Should it become the standard tool for evaluating LITMUS^{RT}?

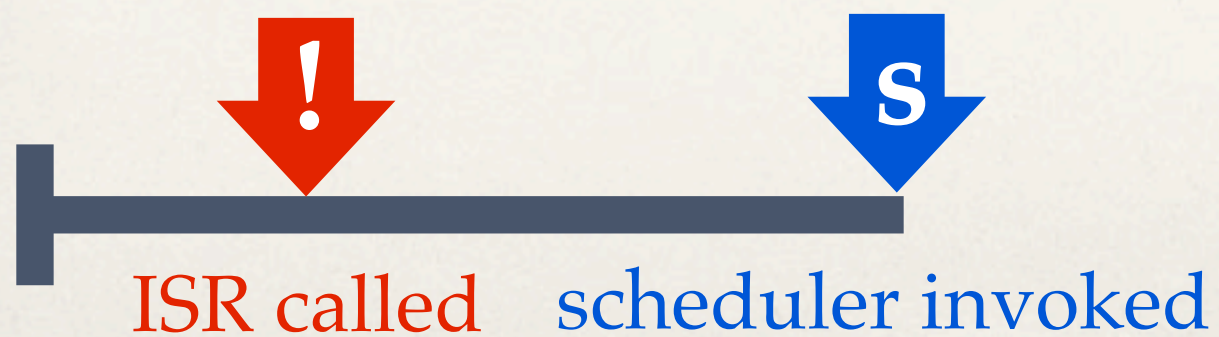
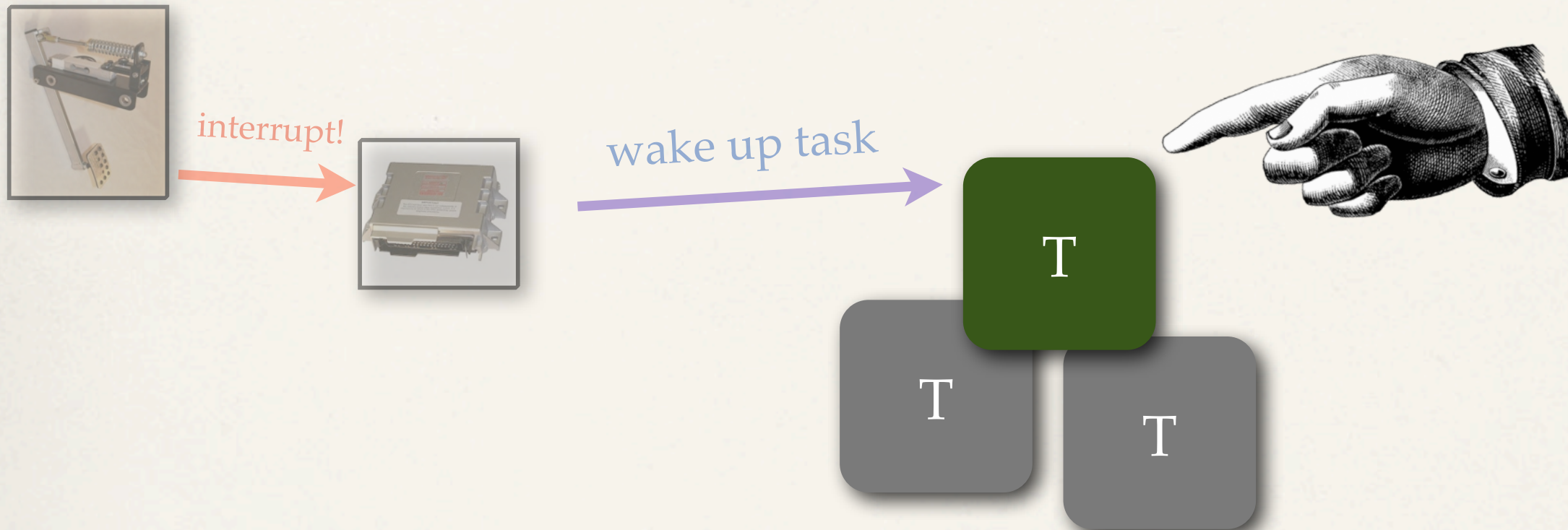
Importance of Feather-Trace

- ❖ cyclicttest was ported to LITMUS^{RT}.
- ❖ Should it become the standard tool for evaluating LITMUS^{RT}?

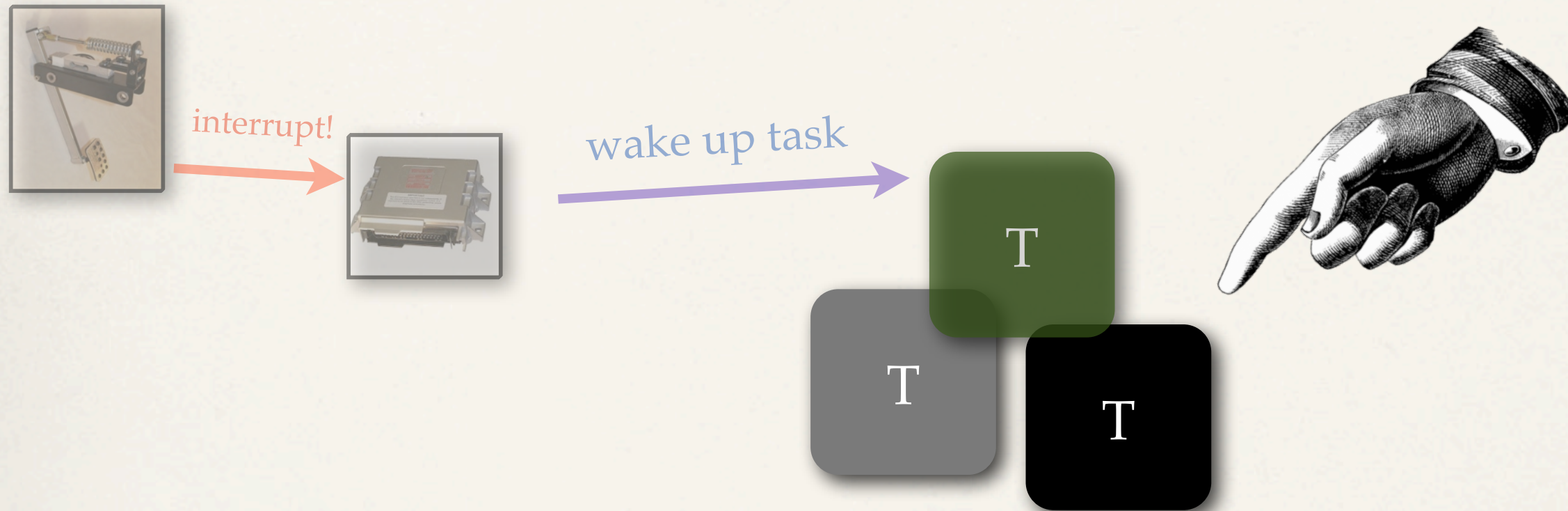
NO!

Interference?

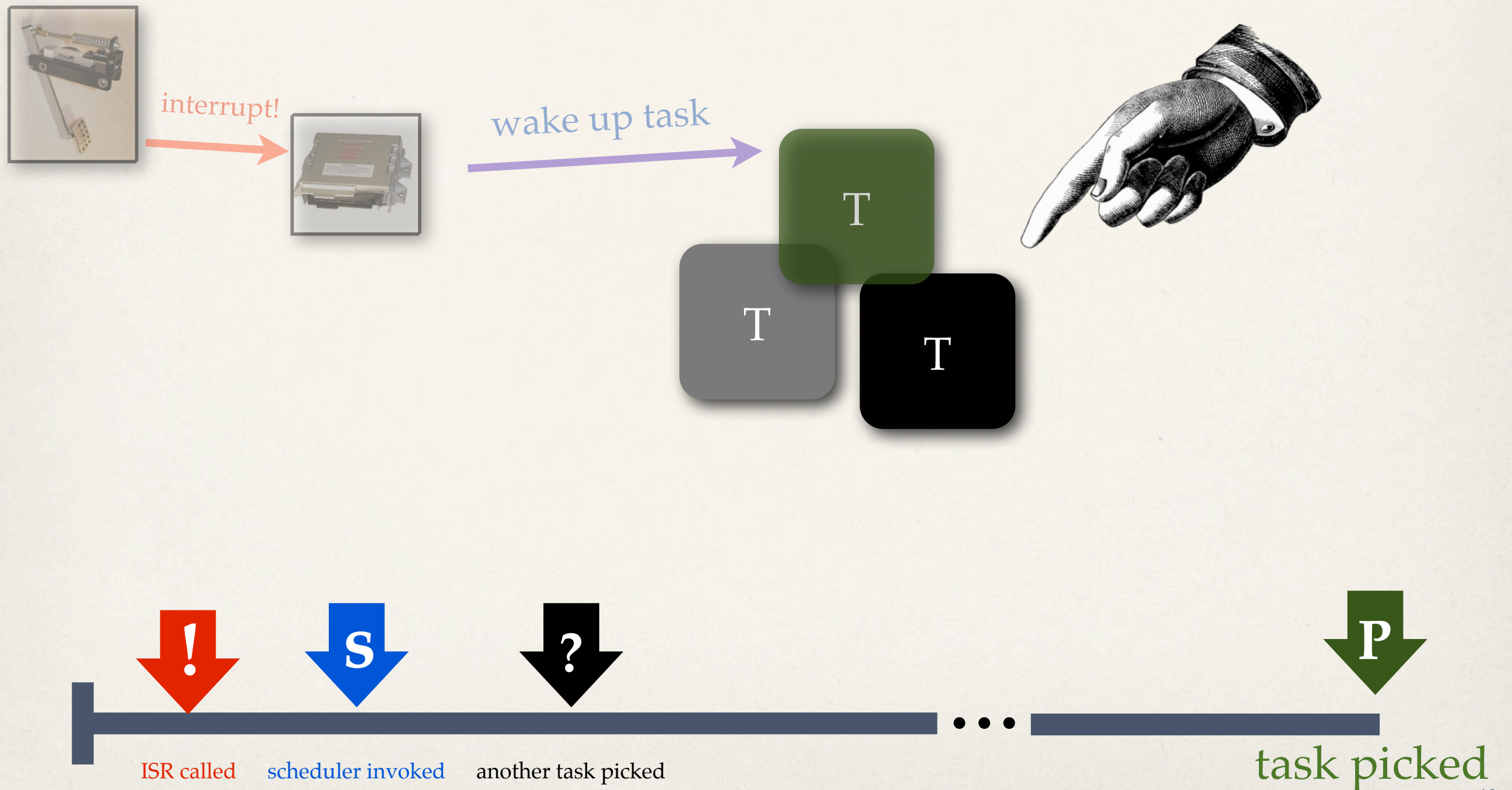
OOPS...
a higher priority
task



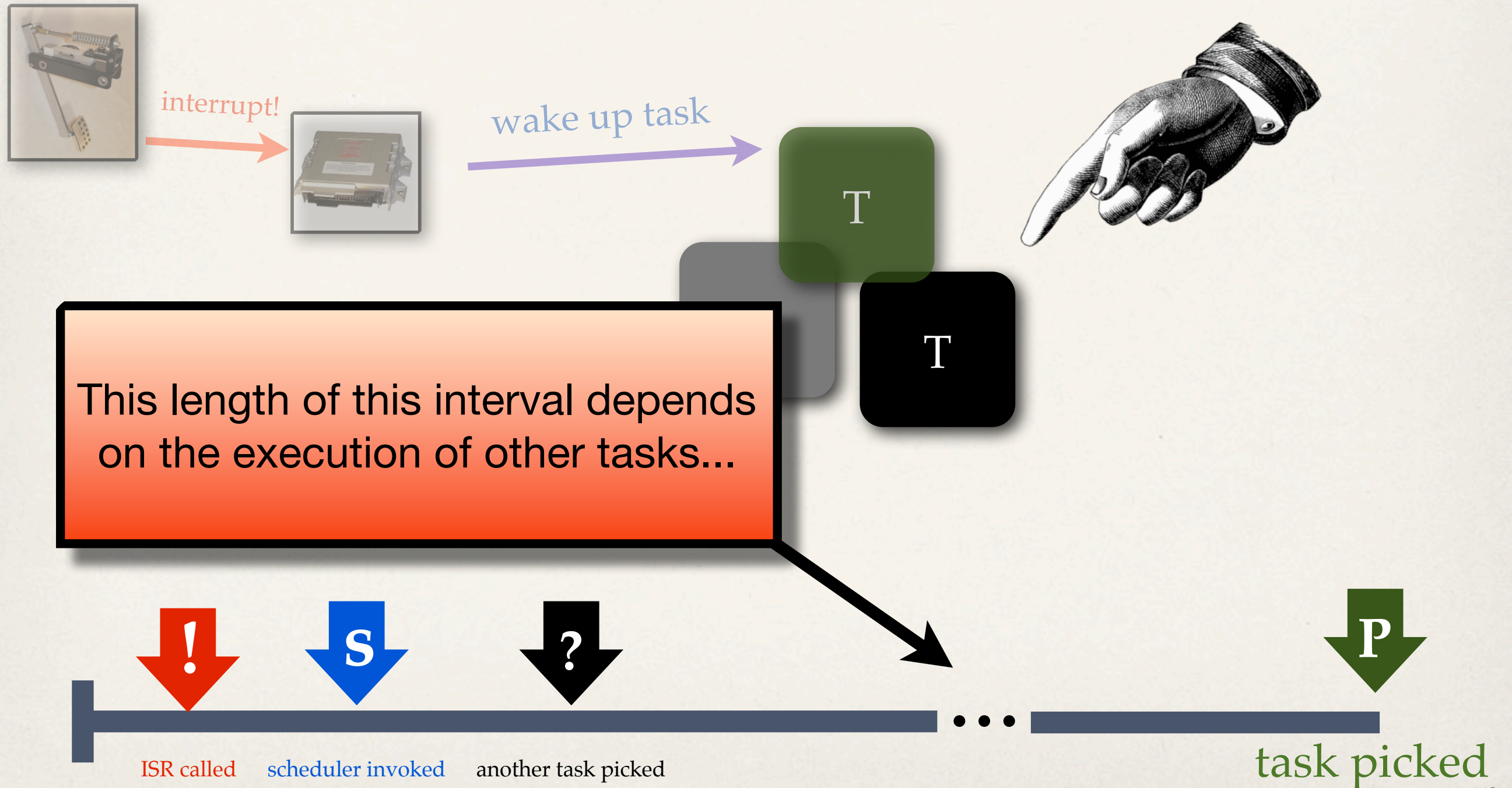
Interference?



Interference?



Interference?



Interference?

..., which depends on other kinds of overhead, preemptions, context switches, etc.



This length of this interval depends on the execution of other tasks...



ISR called



scheduler invoked



another task picked



task picked

Interference?

..., which depends on other kinds of overhead, preemptions, context switches, etc.

Overhead-aware schedulability analysis is required!

This length of this interval depends on the execution of other tasks...



cyclictest or Feather-Trace?

BOTH!

cyclictest

Practical, easy-to-understand measure

Can easily compare responsiveness between kernels.

LITMUS^{RT}/Feather-Trace

For tasks other than the highest-priority ones, schedulability analysis is necessary.

Only with Feather-Trace we obtain the data required for the analysis.

Conclusion

LITMUS^{RT}: small overheads in comparison with stock Linux

PREEMPT_RT is highly necessary for Linux as a RTOS
LITMUS^{RT} will be ported to PREEMPT_RT soon

Scheduling latency should not be used as the sole metric for quantifying real-time guarantees

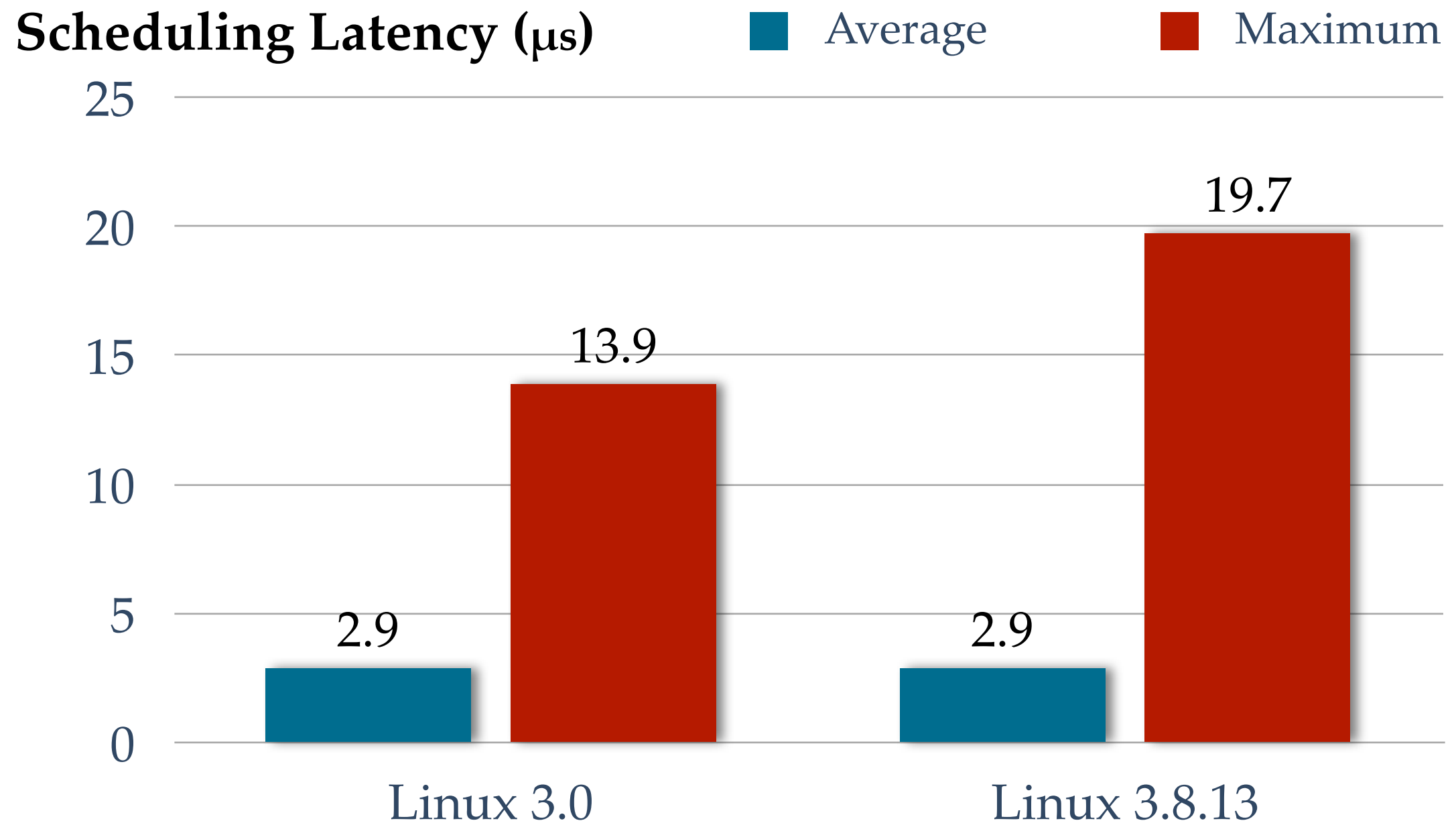
Thank You!

We also have a patch that implements Feather-Trace on top of standard Linux, enabling fine-grained measurements.

Appendix

Linux 3.0 vs. Linux 3.8.13

No Background Tasks

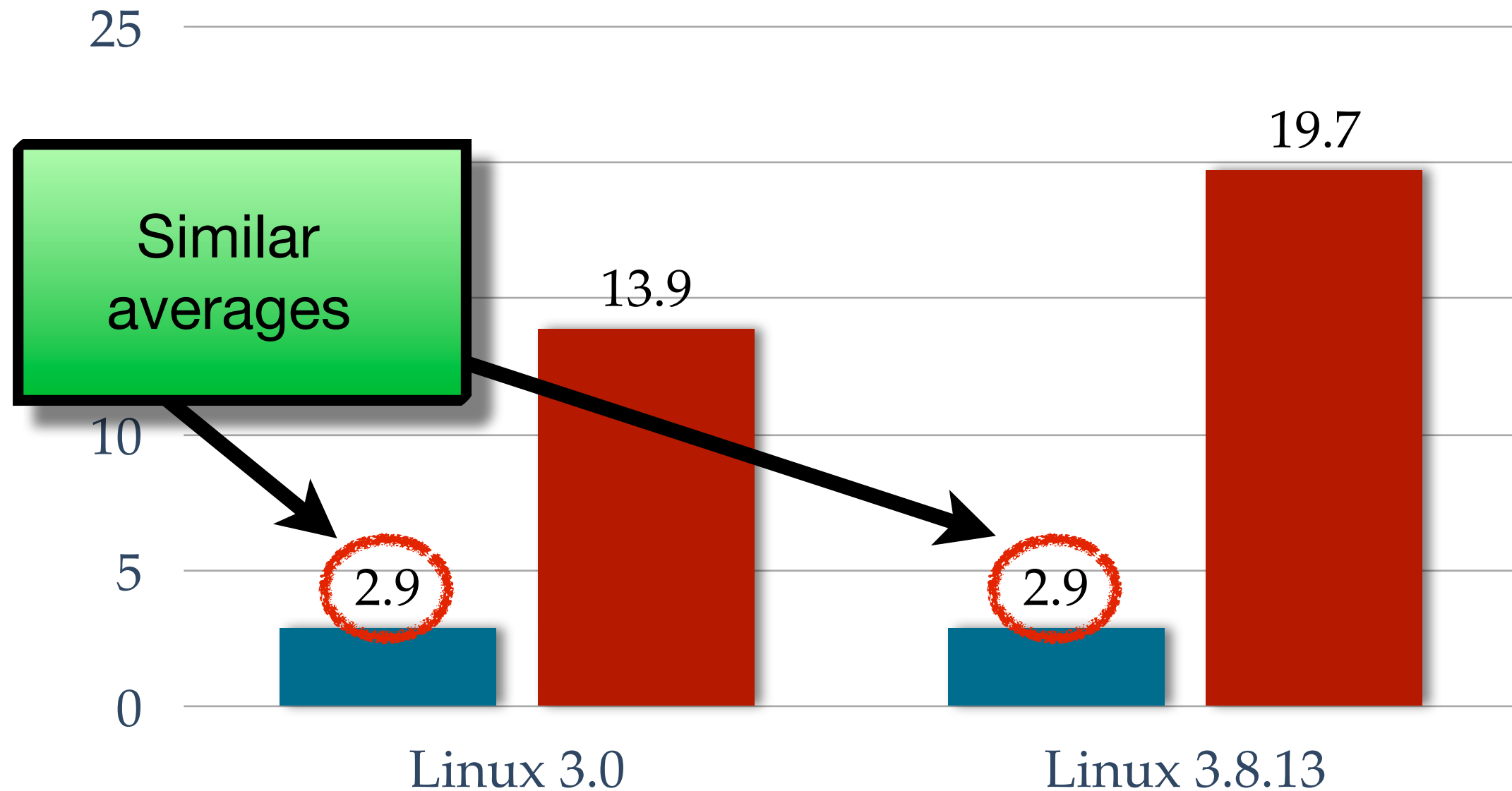


No Background Tasks

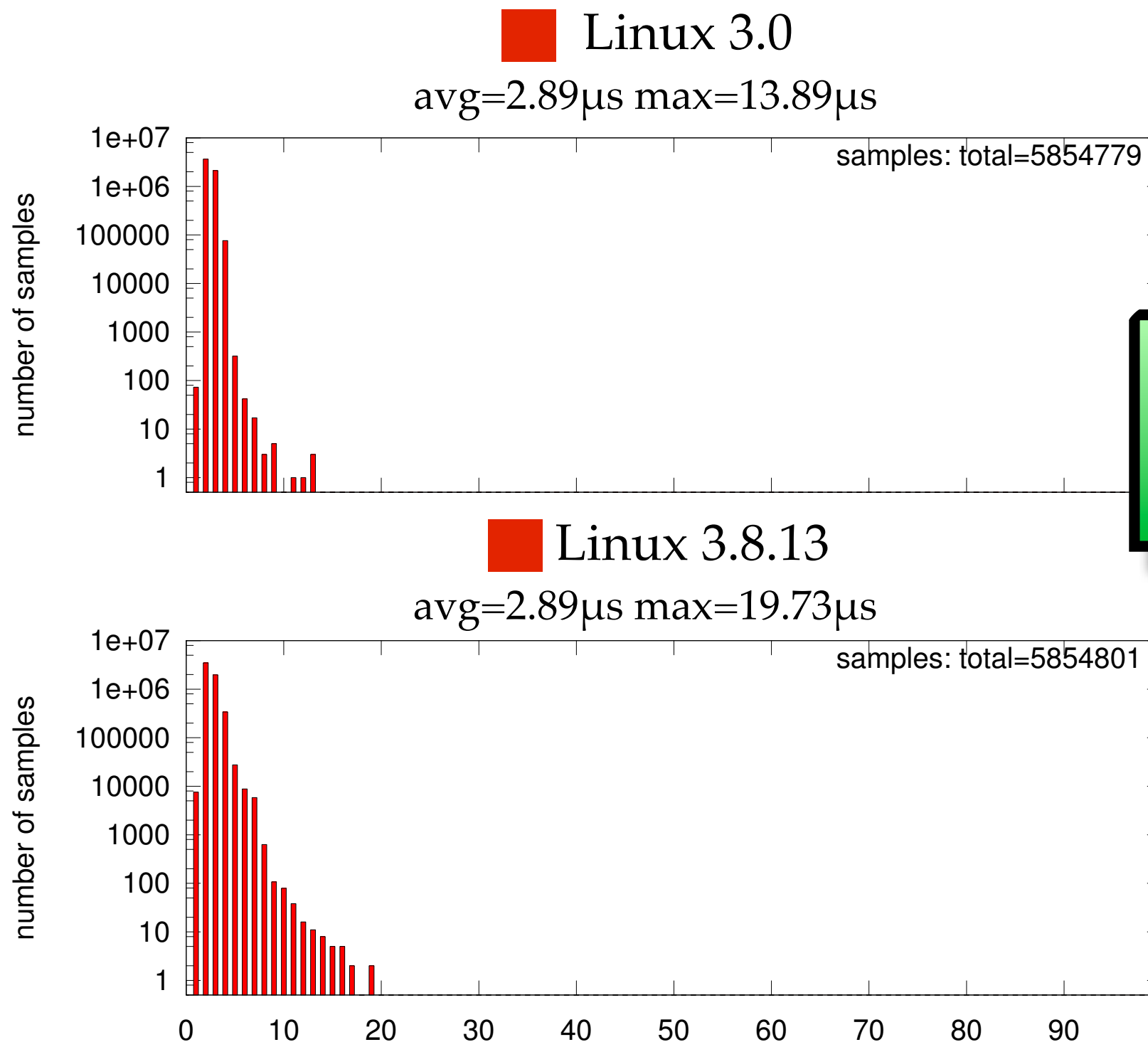
Scheduling Latency (μs)

■ Average

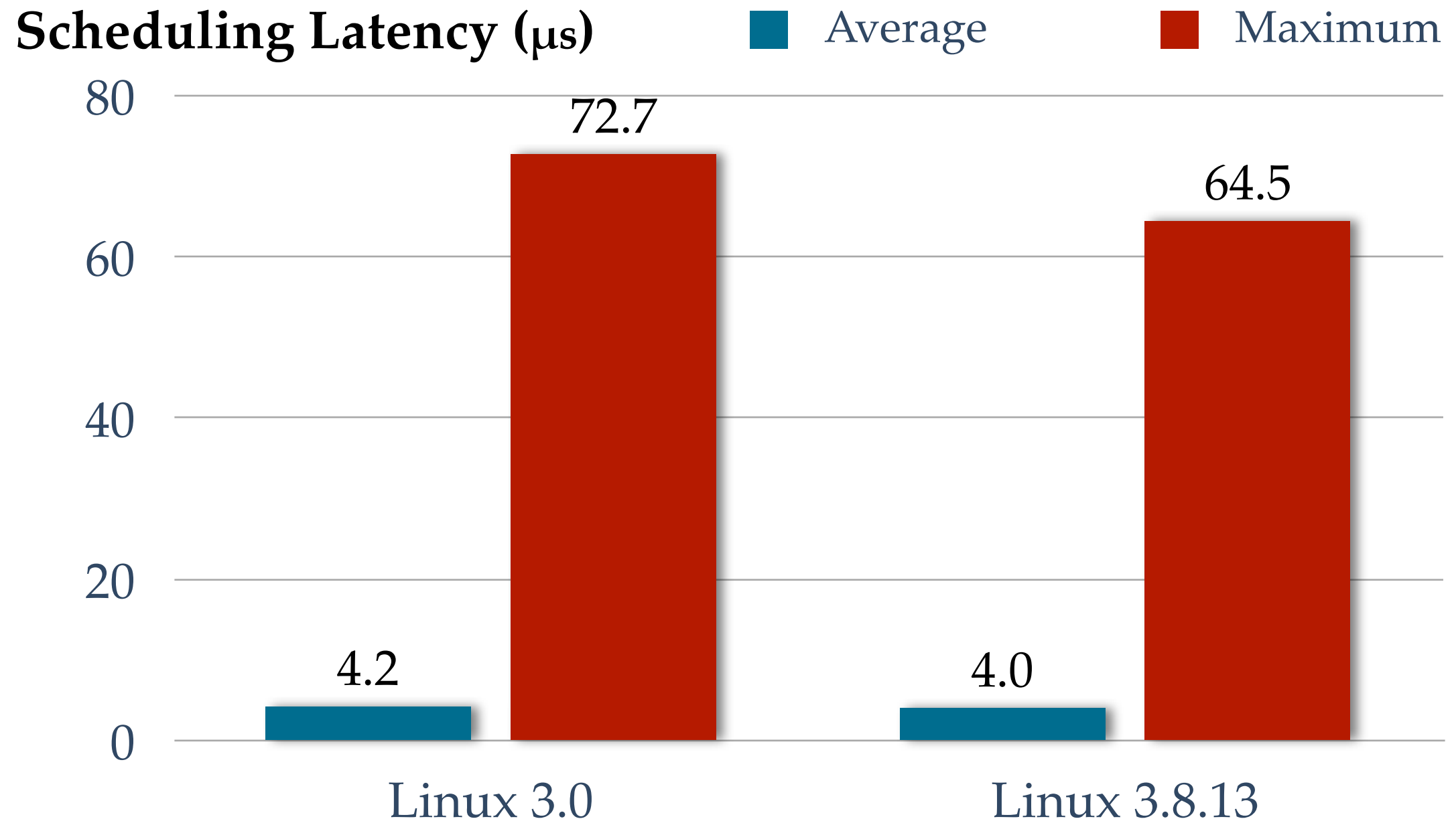
■ Maximum



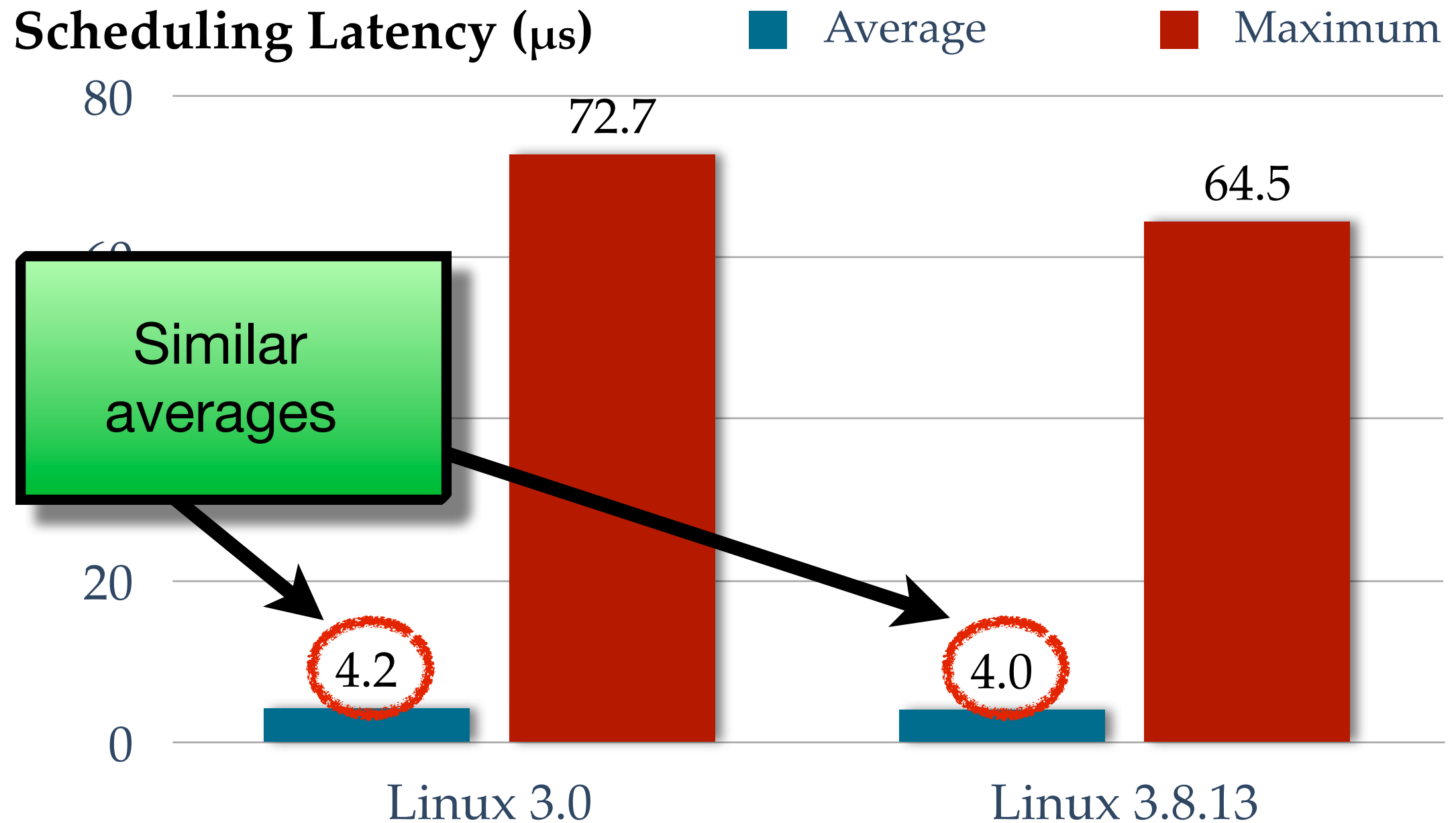
No Background Tasks



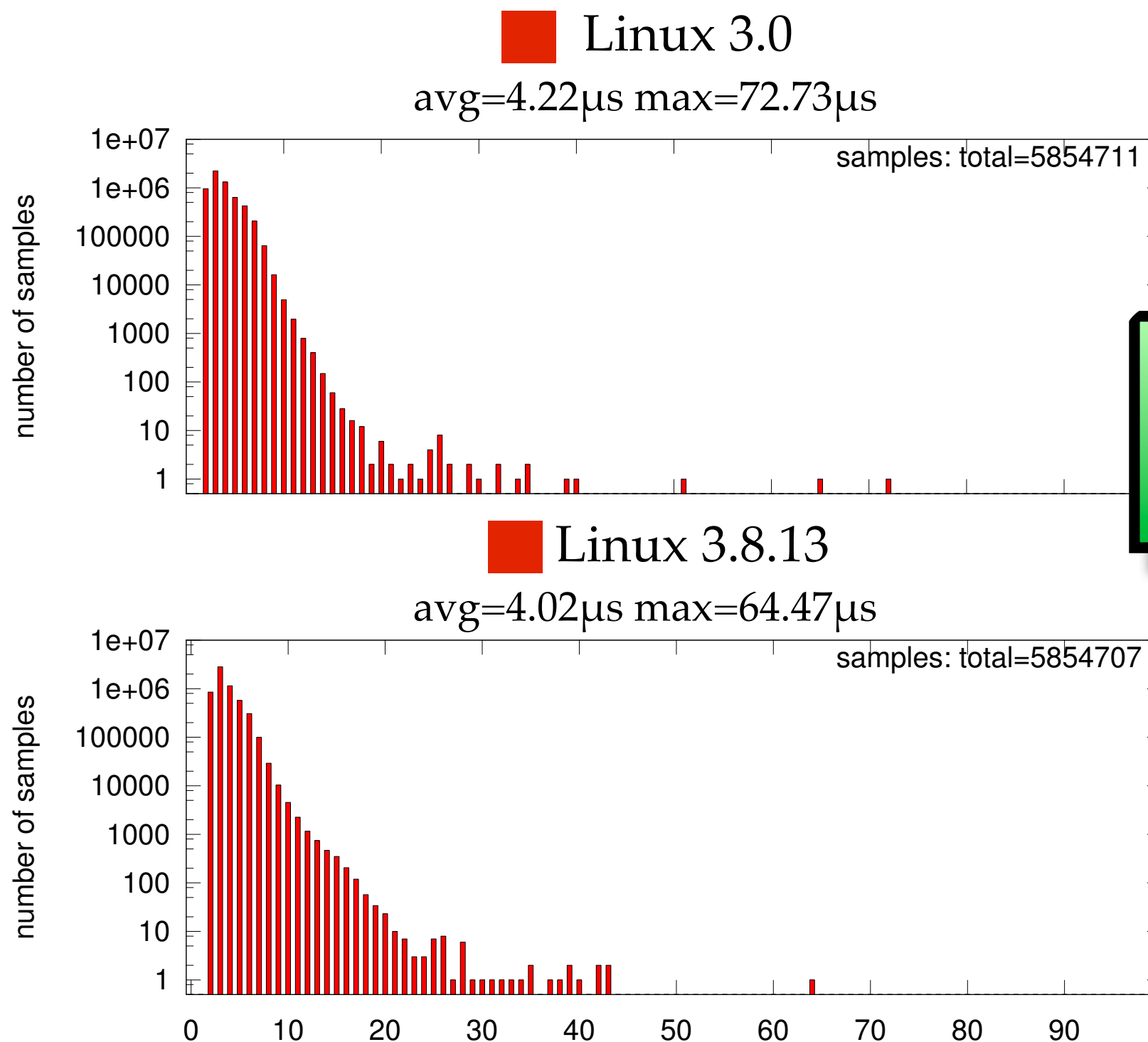
CPU-bound Background Tasks



CPU-bound Background Tasks



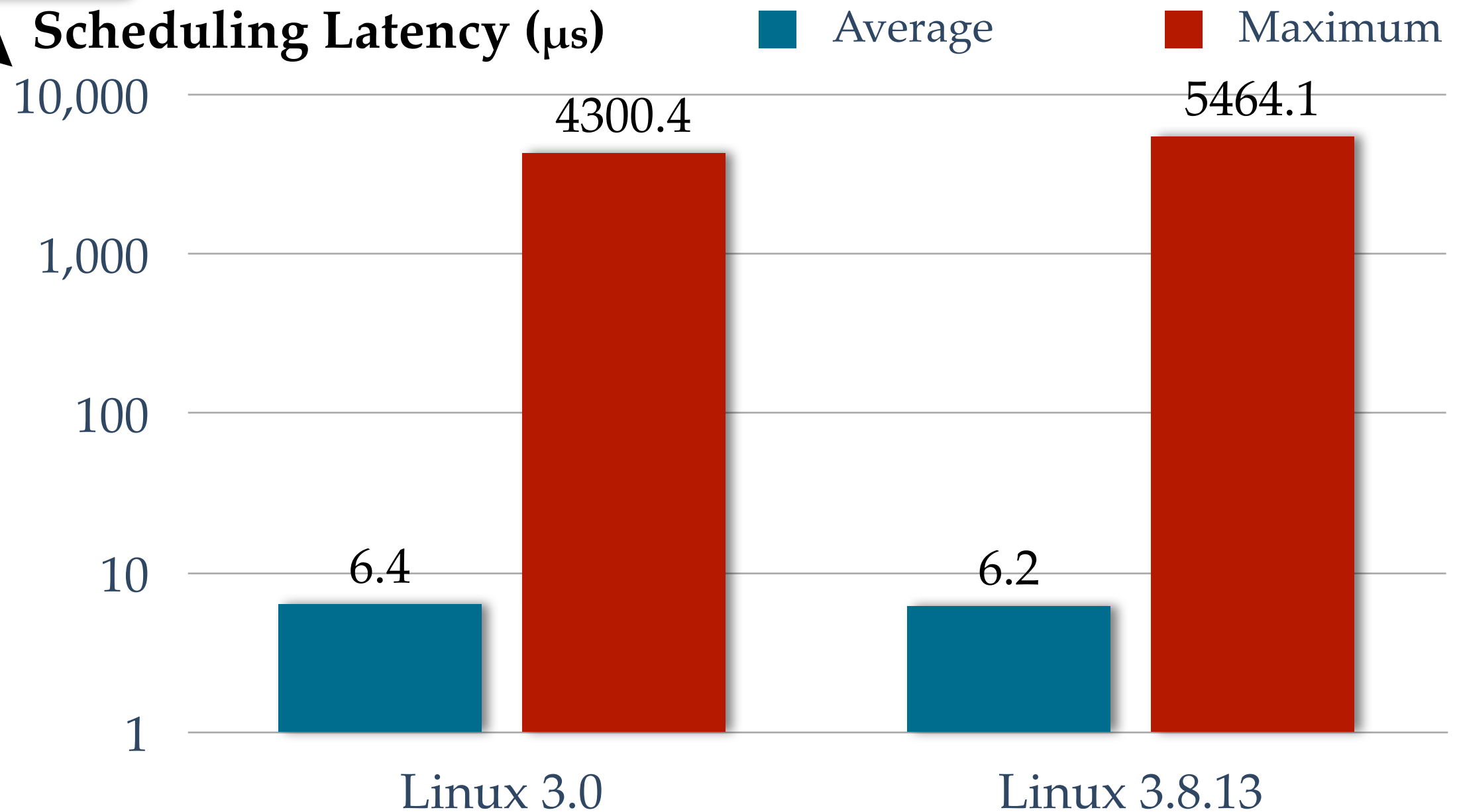
CPU-bound Background Tasks



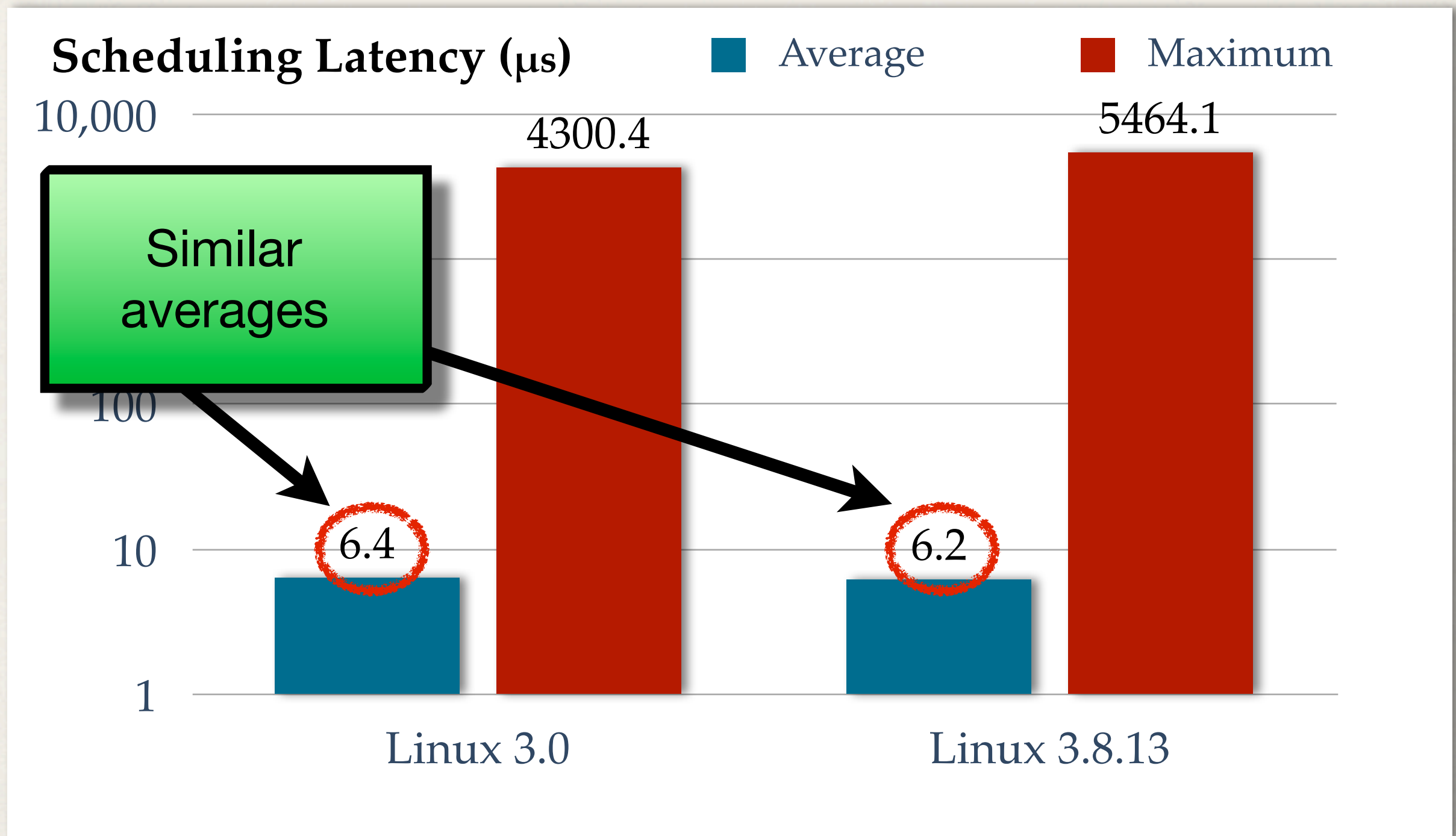
Similar
shapes

I/O-bound Background Tasks

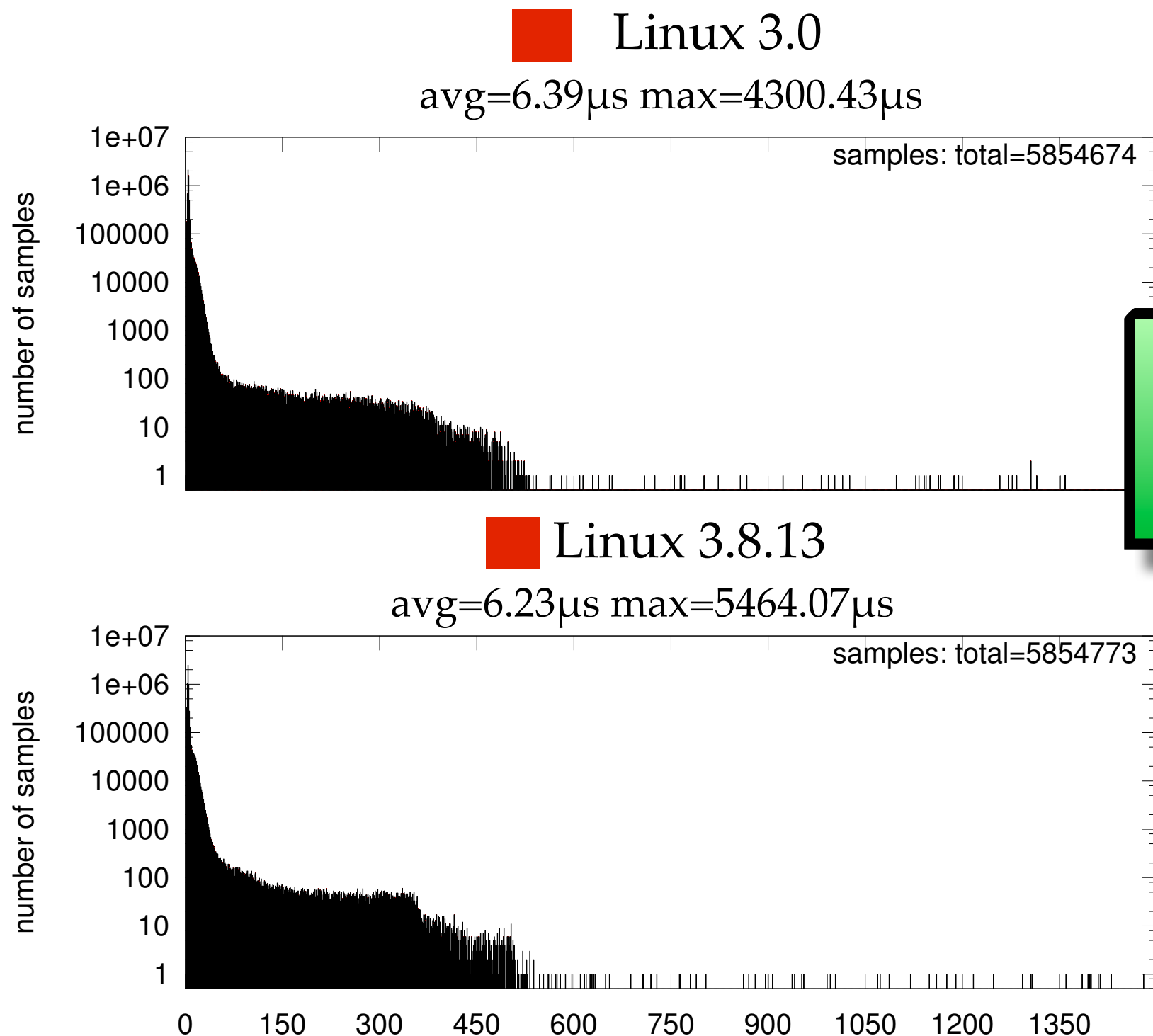
log scale!



I/O-bound Background Tasks



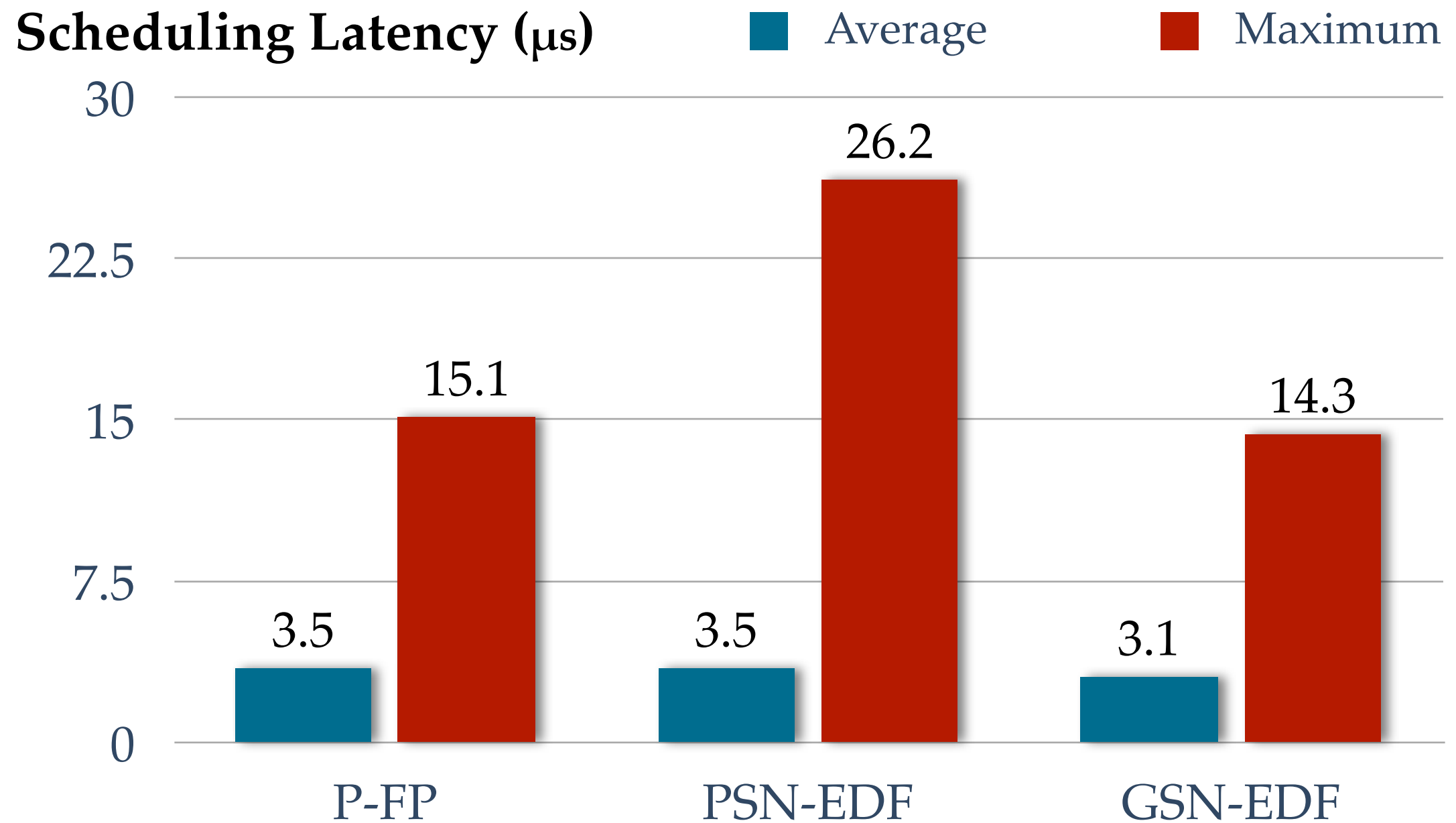
I/O-bound Background Tasks



Similar shapes

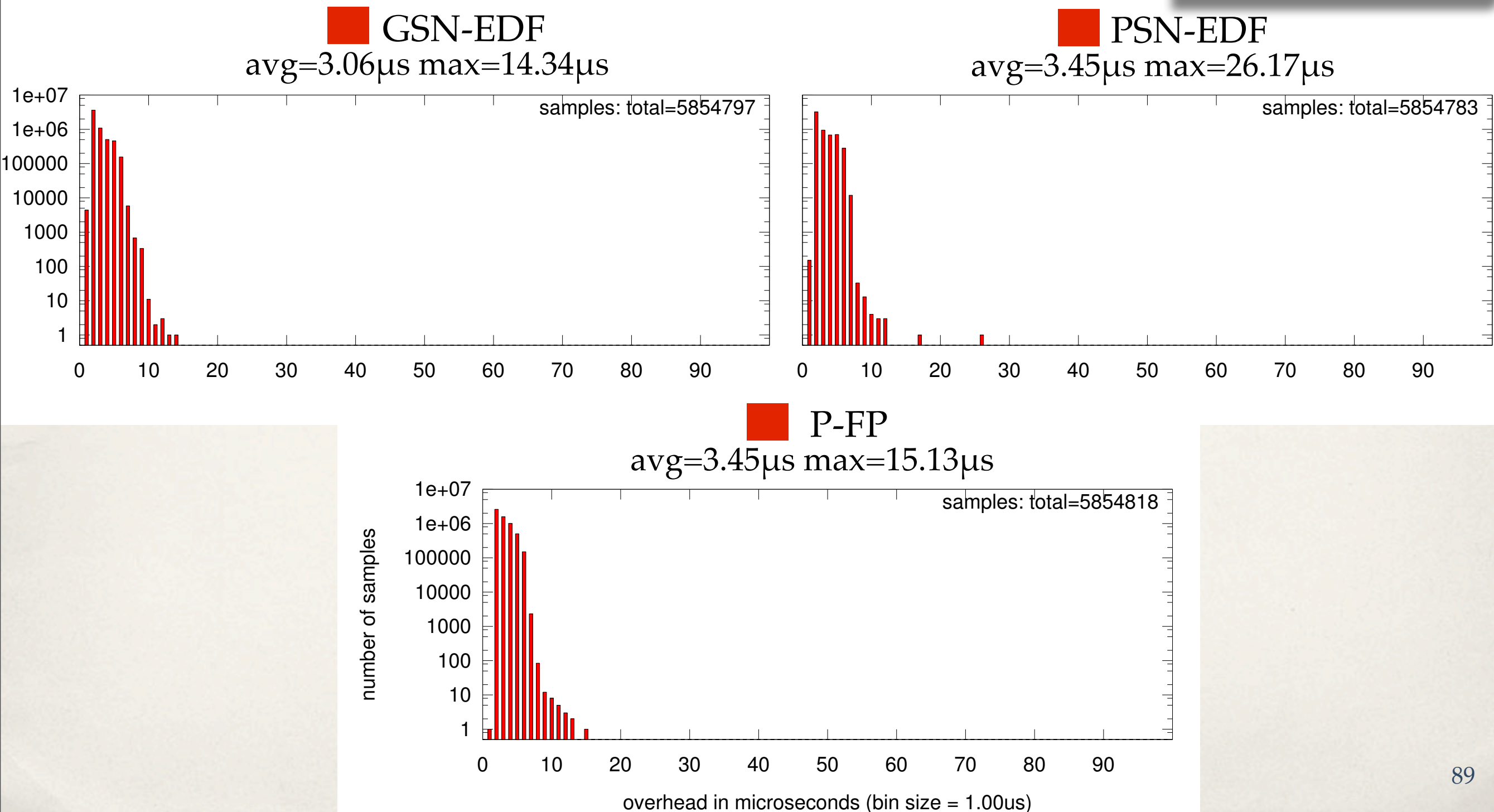
LITMUS^{RT}'s plugins

No Background Tasks



No Background Tasks

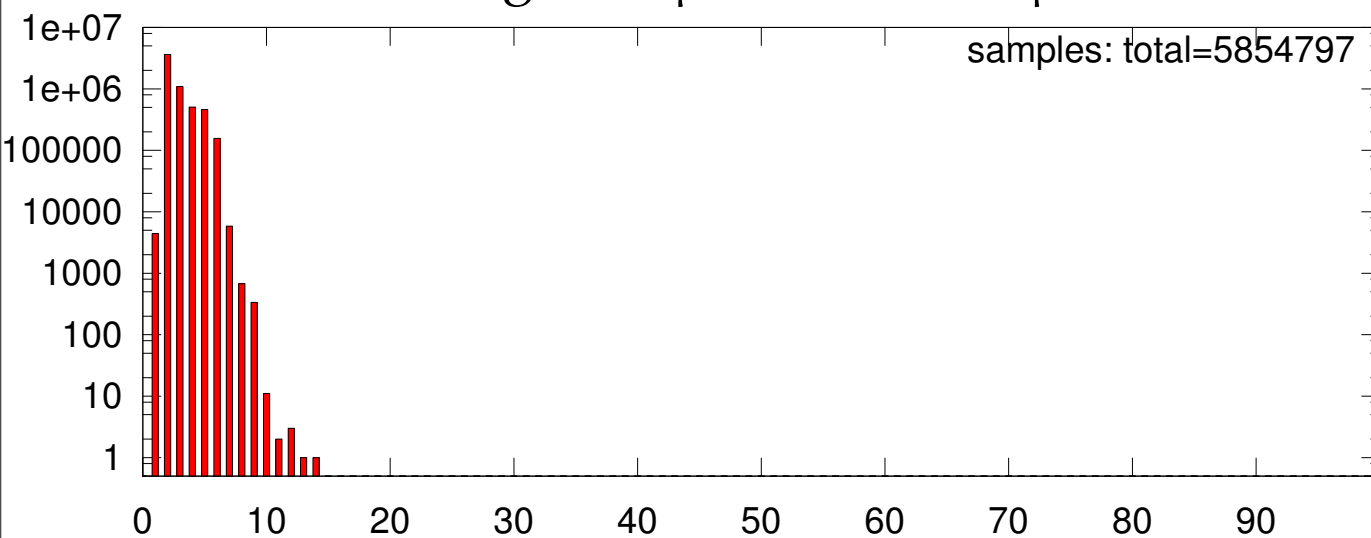
Similar
shapes



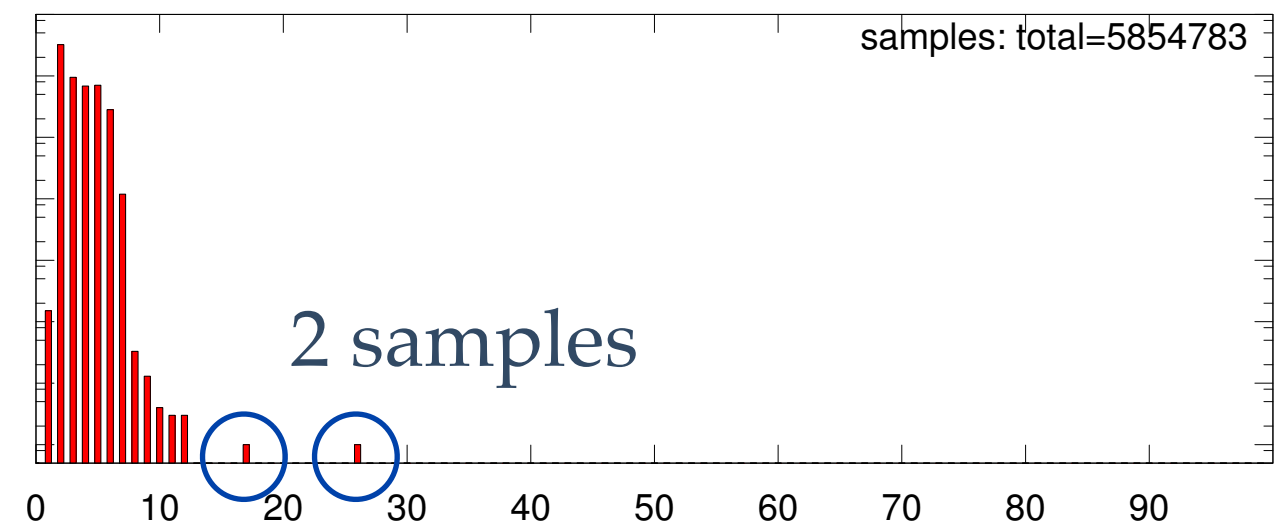
No Background Tasks

Similar
shapes

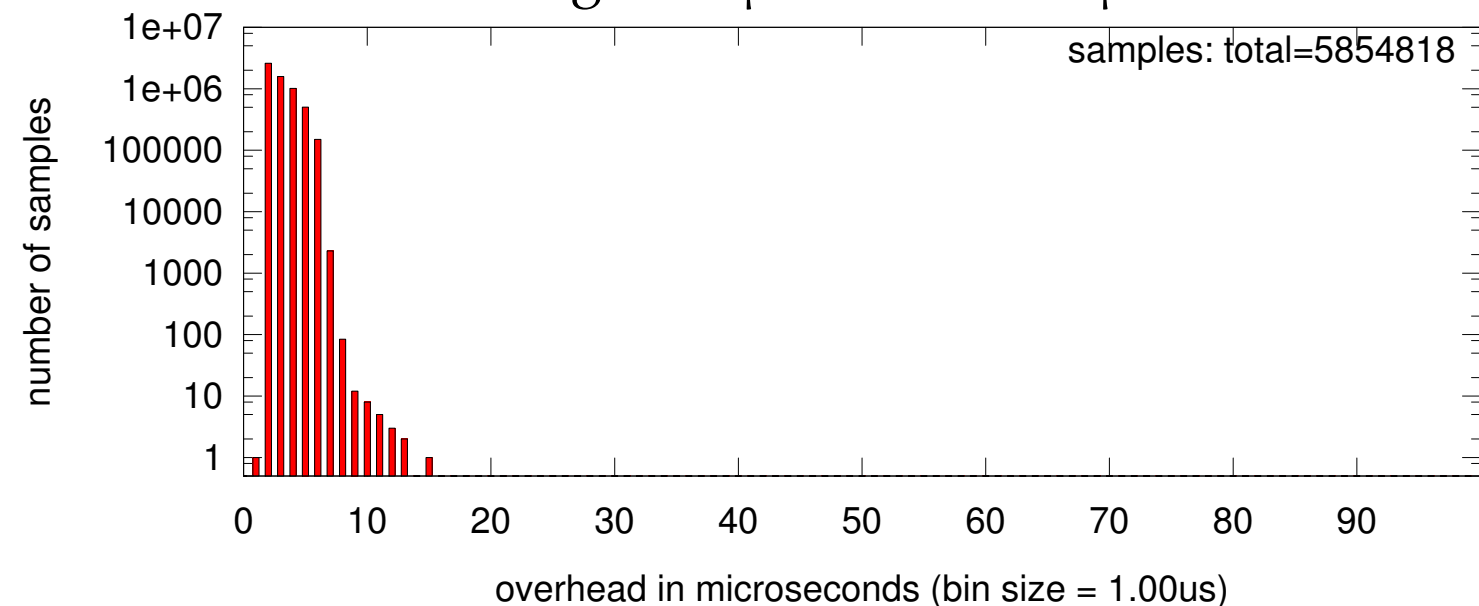
■ GSN-EDF
avg=3.06 μ s max=14.34 μ s



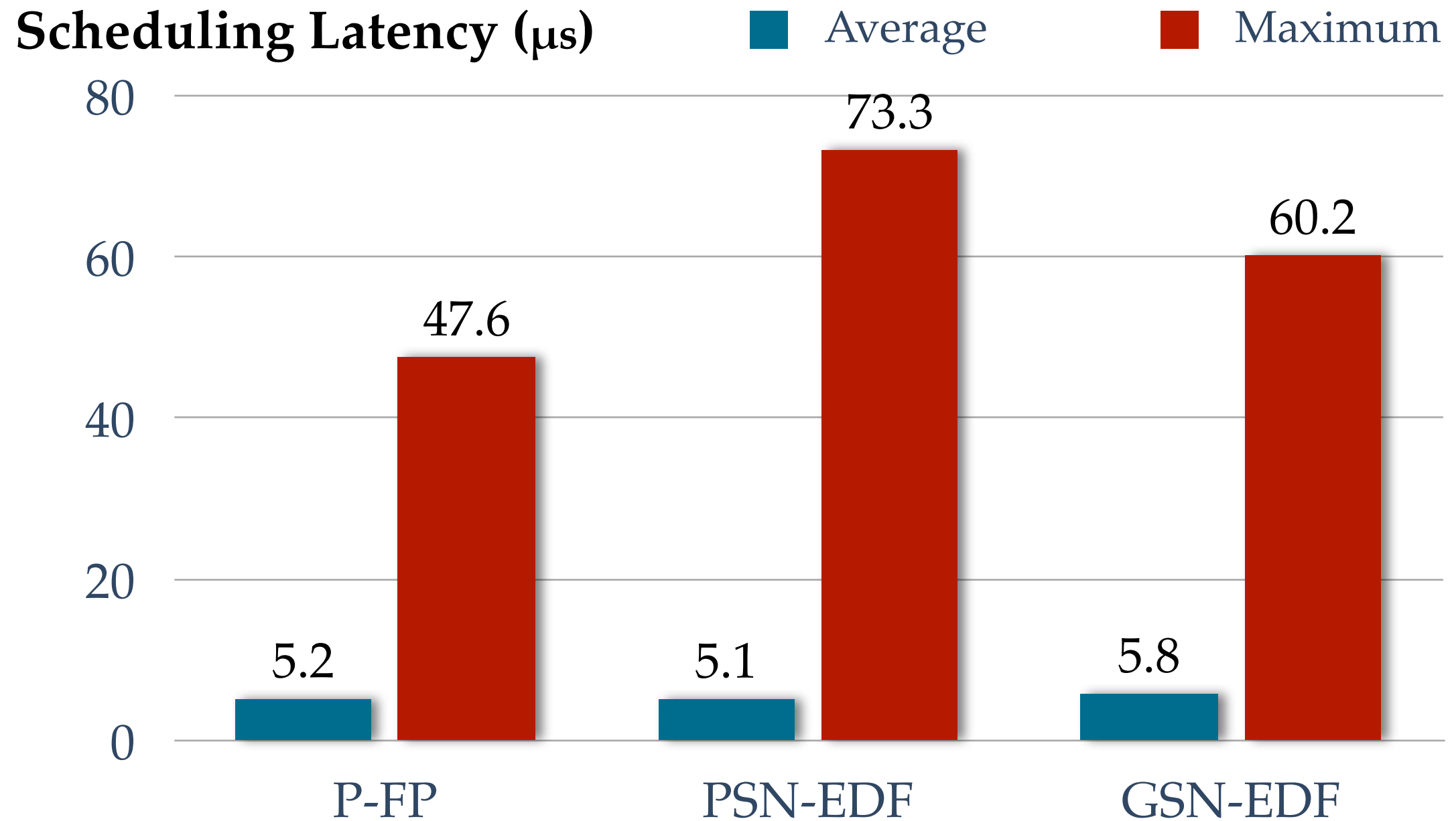
■ PSN-EDF
avg=3.45 μ s max=26.17 μ s



■ P-FP
avg=3.45 μ s max=15.13 μ s

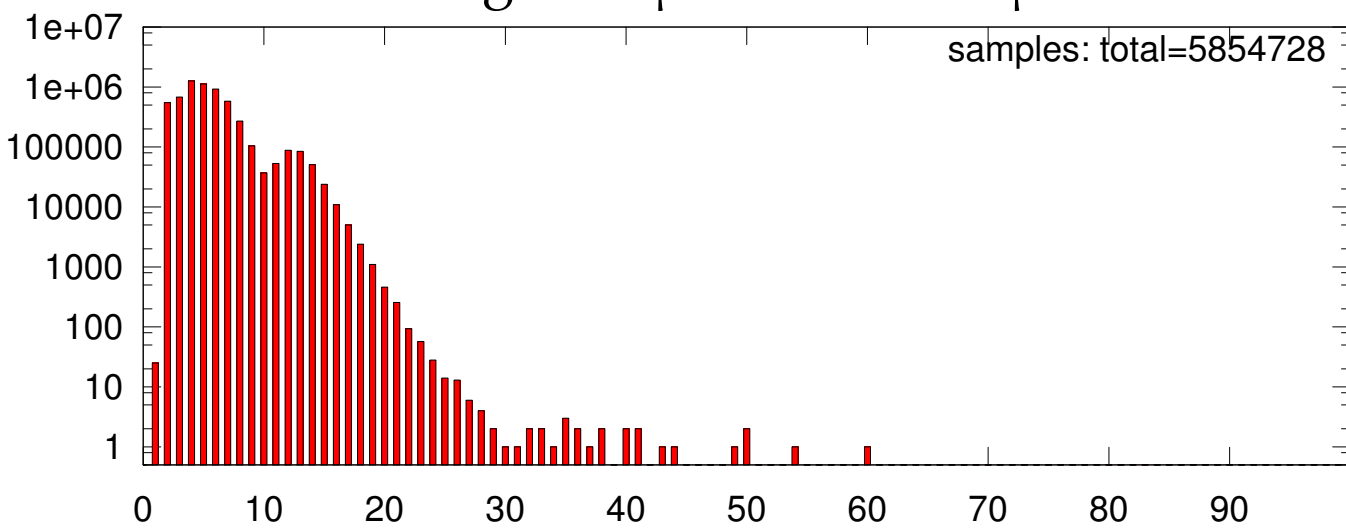


CPU-bound Background Tasks

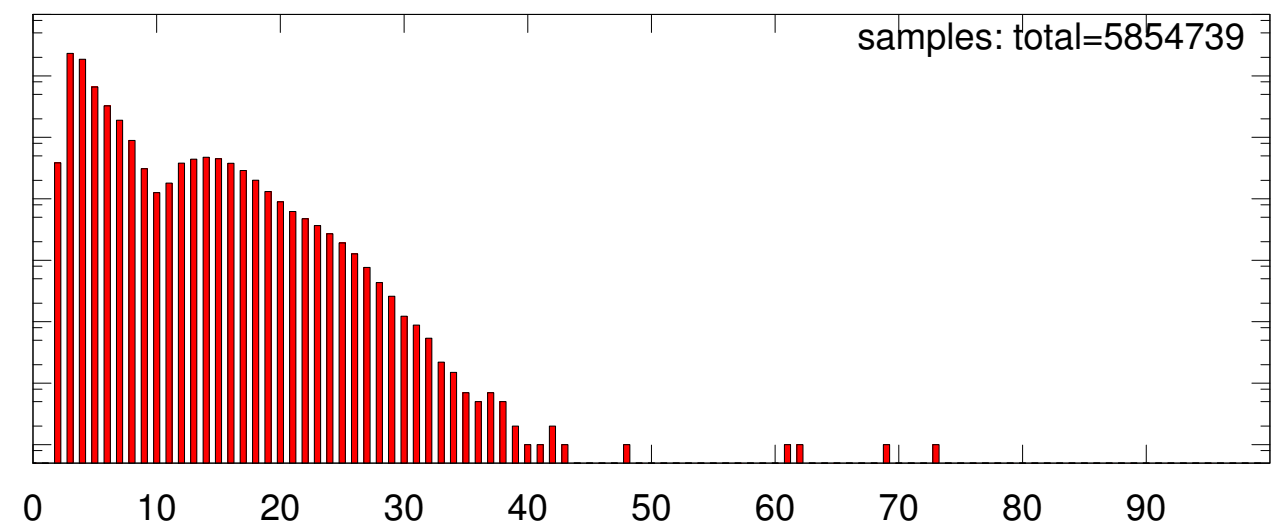


CPU-bound Background Tasks

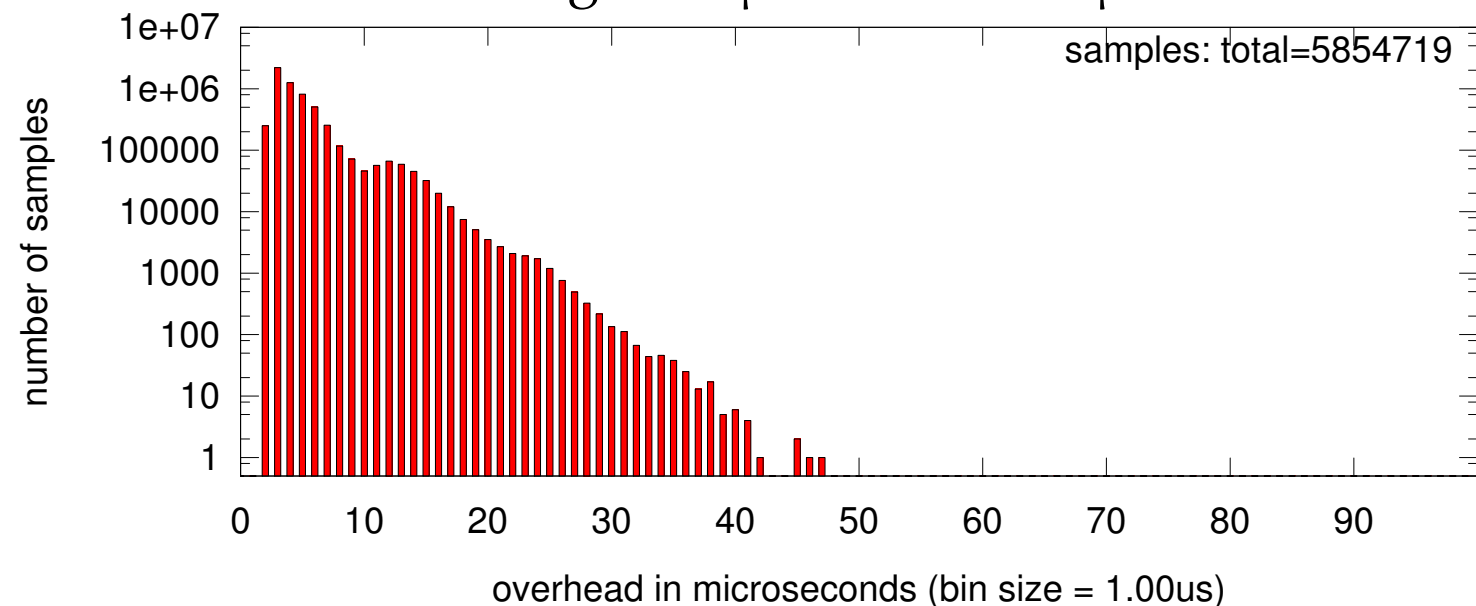
■ GSN-EDF
avg=5.81 μ s max=60.20 μ s



■ PSN-EDF
avg=5.14 μ s max=73.27 μ s



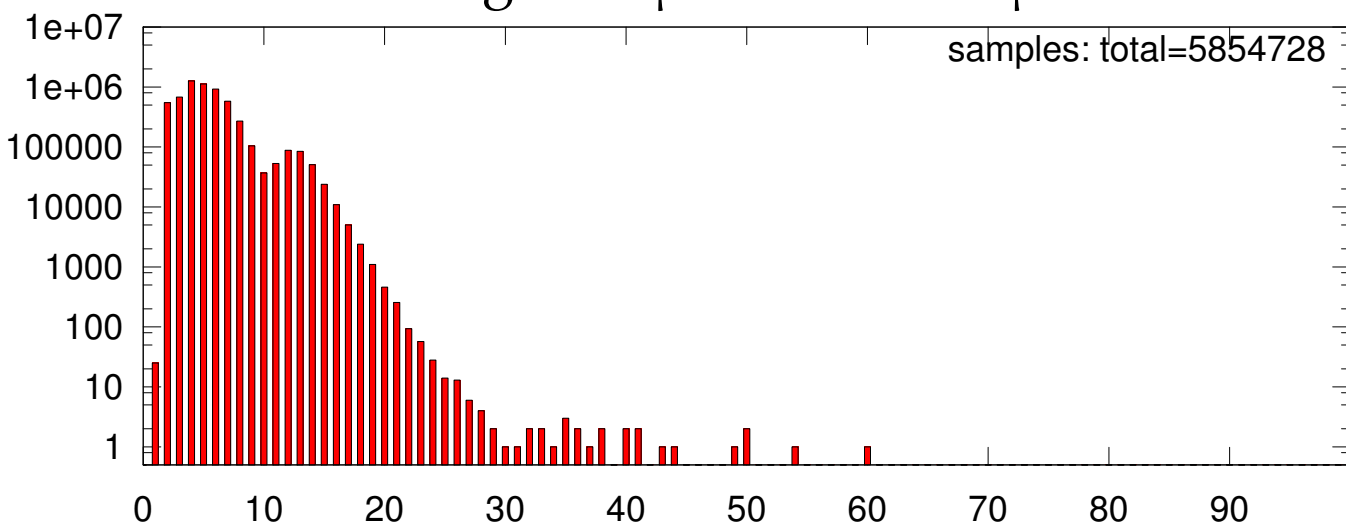
■ P-FP
avg=5.17 μ s max=47.59 μ s



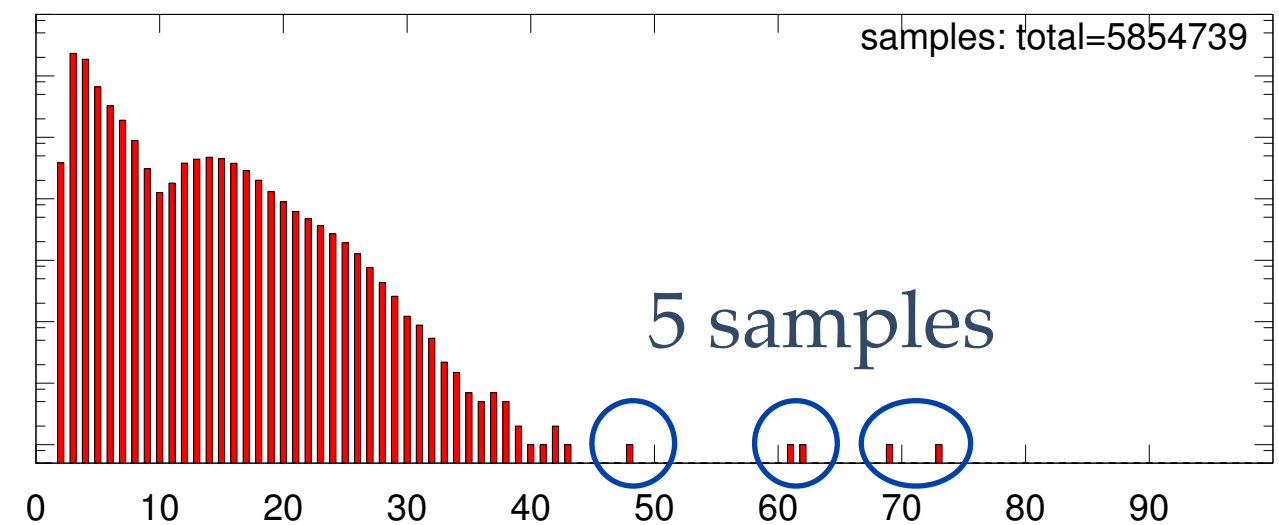
Similar
shapes

CPU-bound Background Tasks

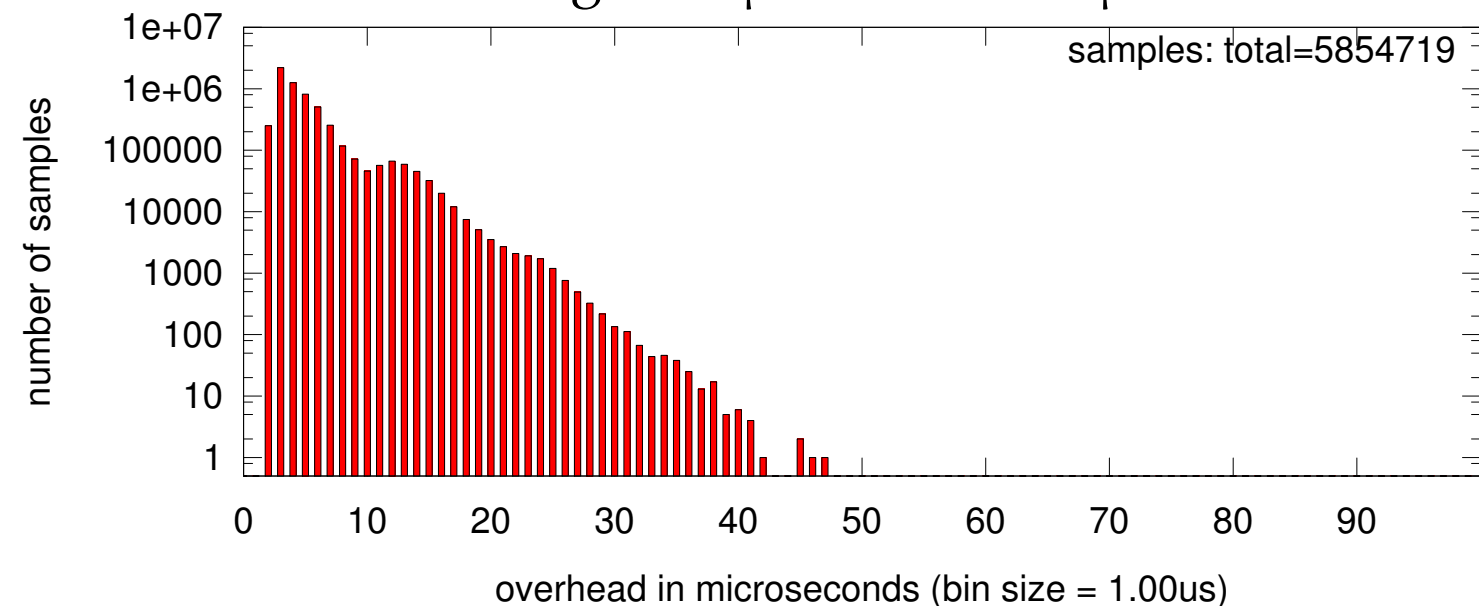
■ GSN-EDF
avg=5.81 μ s max=60.20 μ s



■ PSN-EDF
avg=5.14 μ s max=73.27 μ s



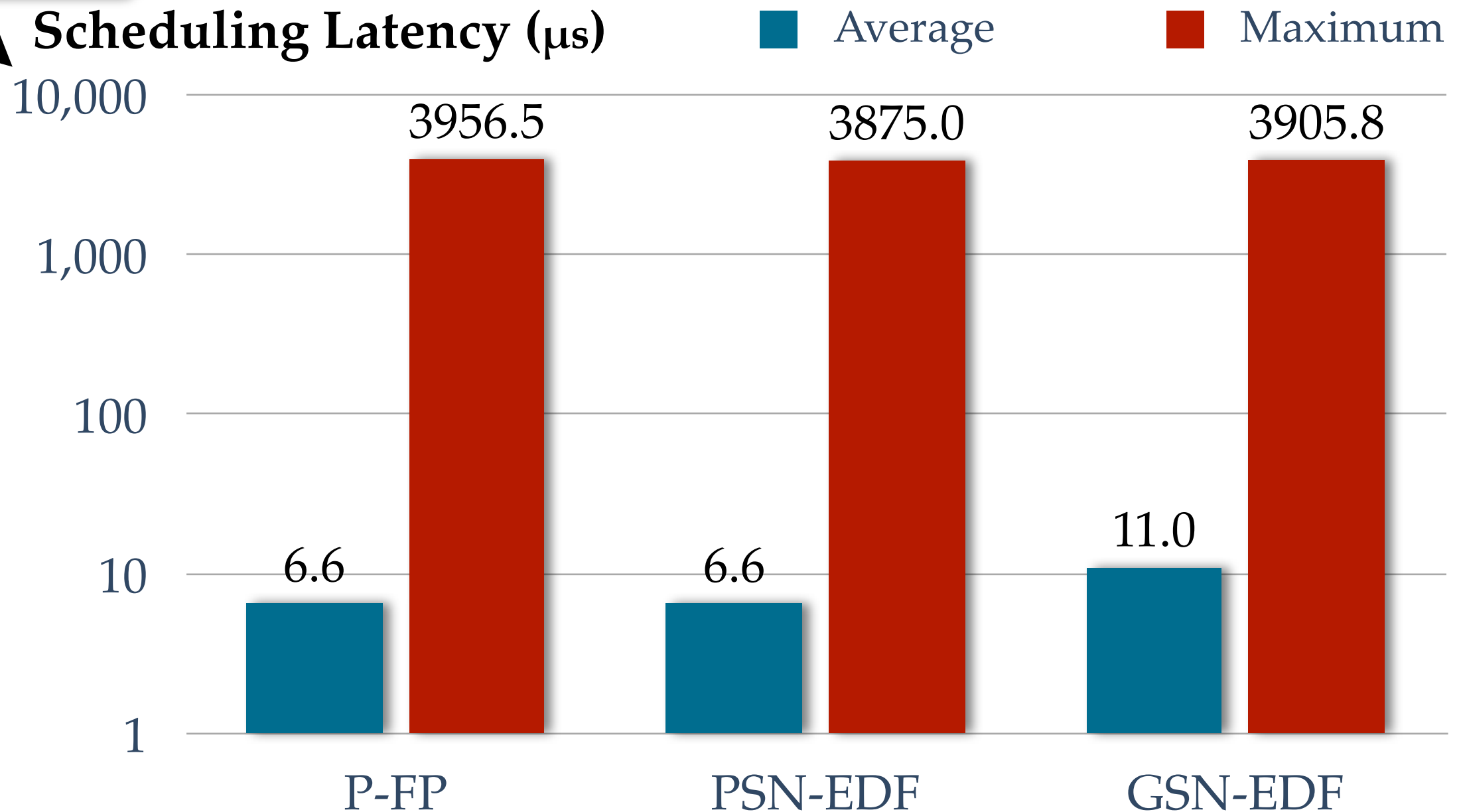
■ P-FP
avg=5.17 μ s max=47.59 μ s



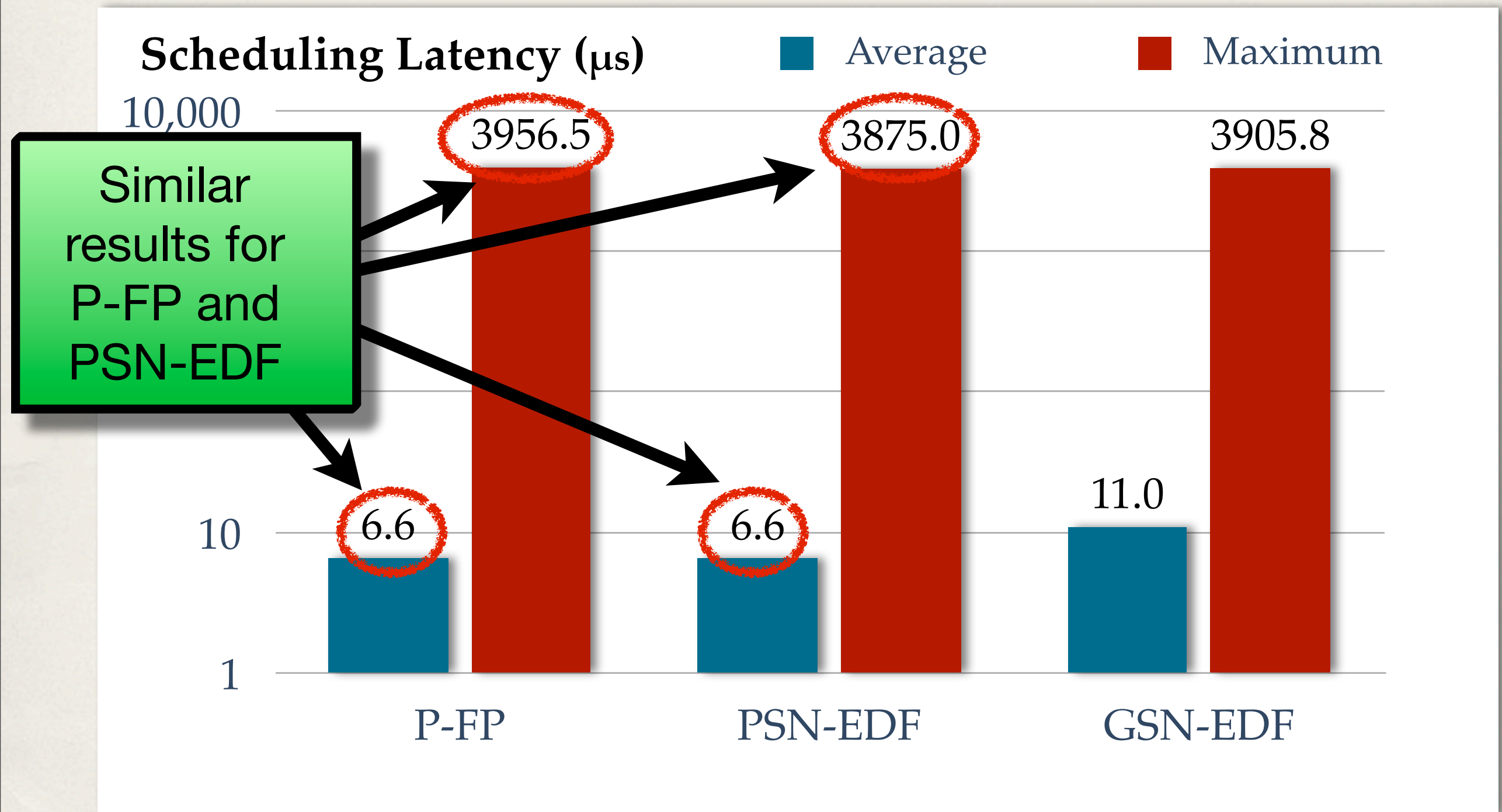
Similar
shapes

I/O-bound Background Tasks

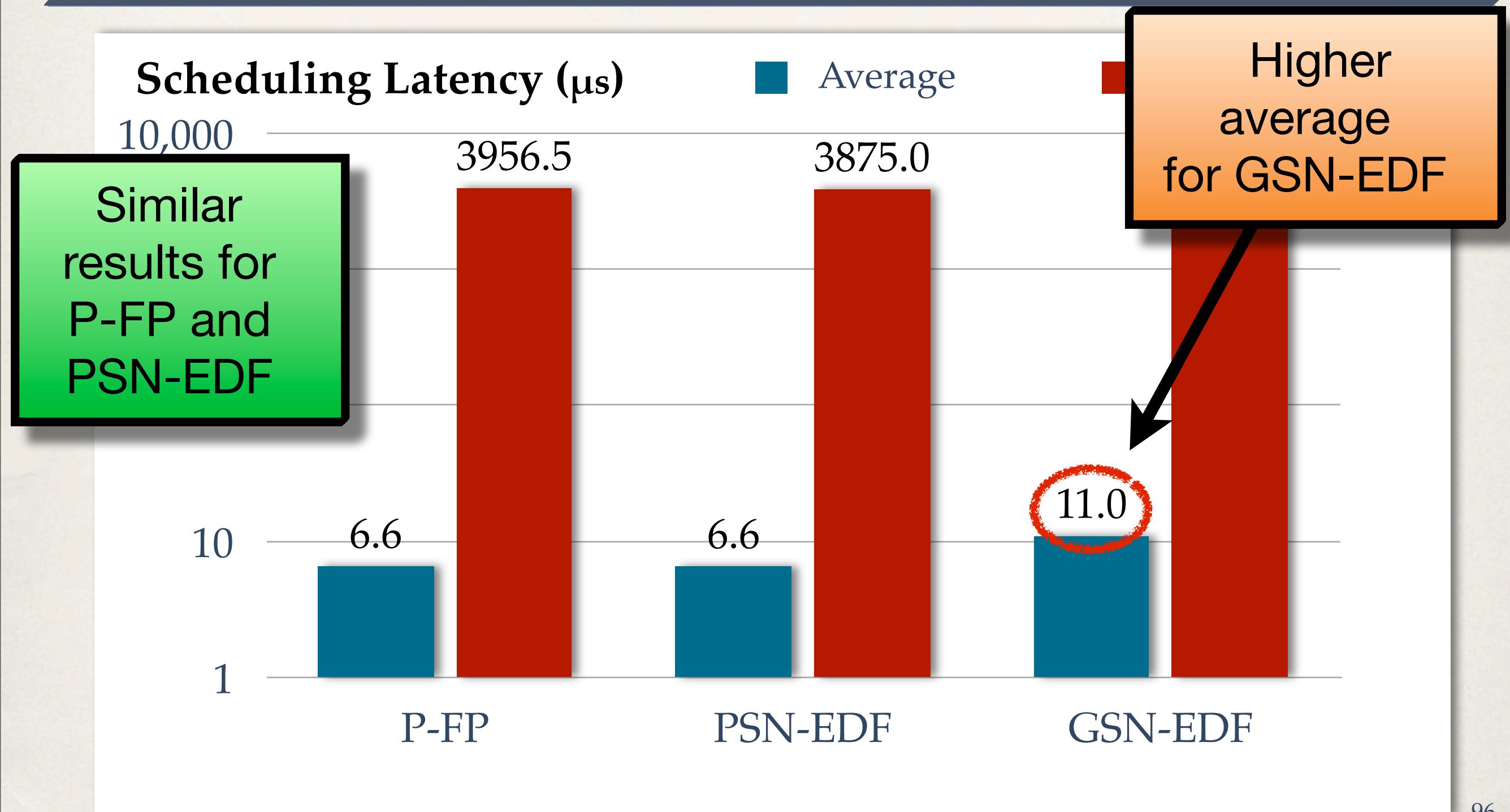
log scale!



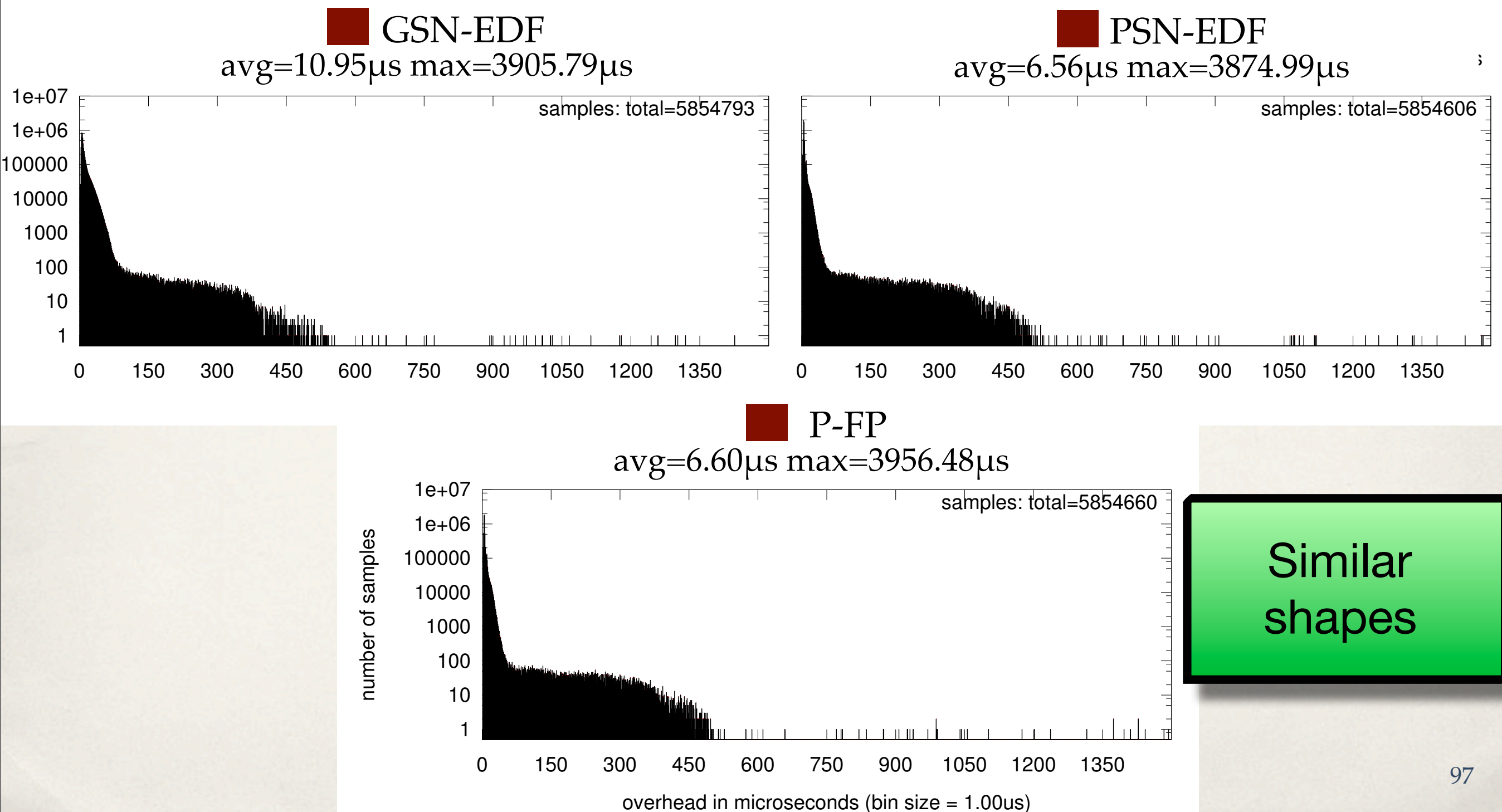
I/O-bound Background Tasks



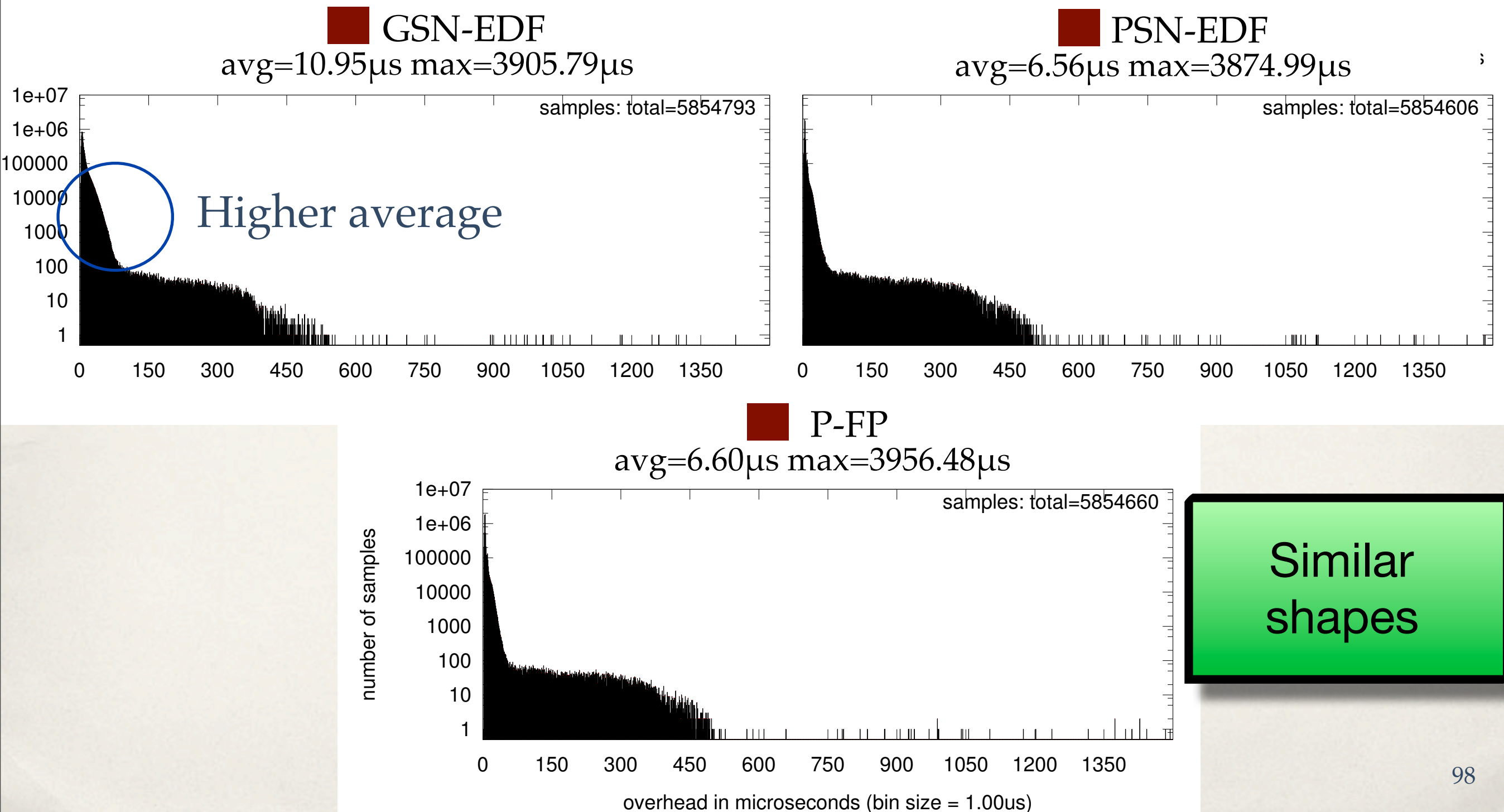
I/O-bound Background Tasks



I/O-bound Background Tasks

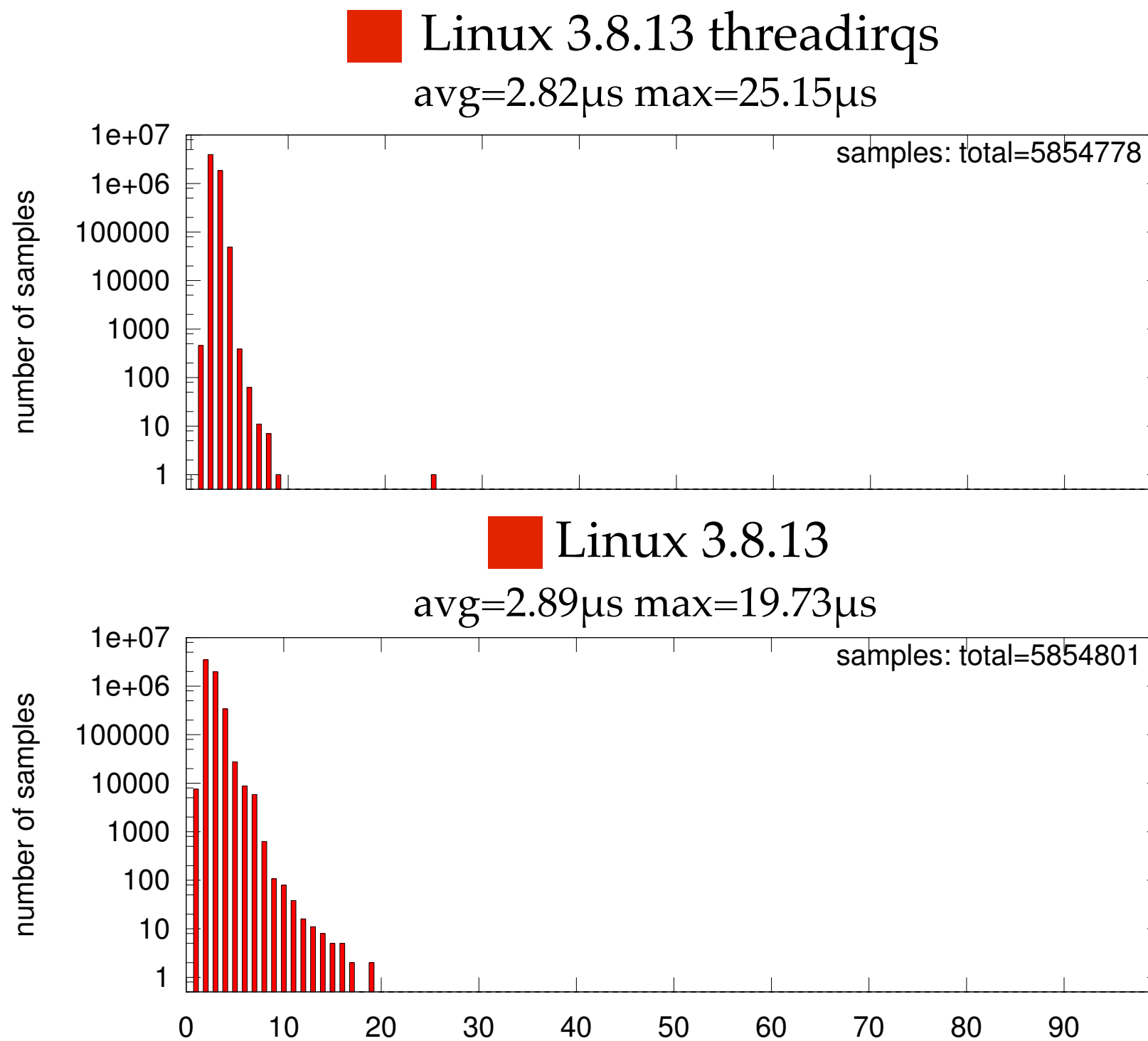


I/O-bound Background Tasks

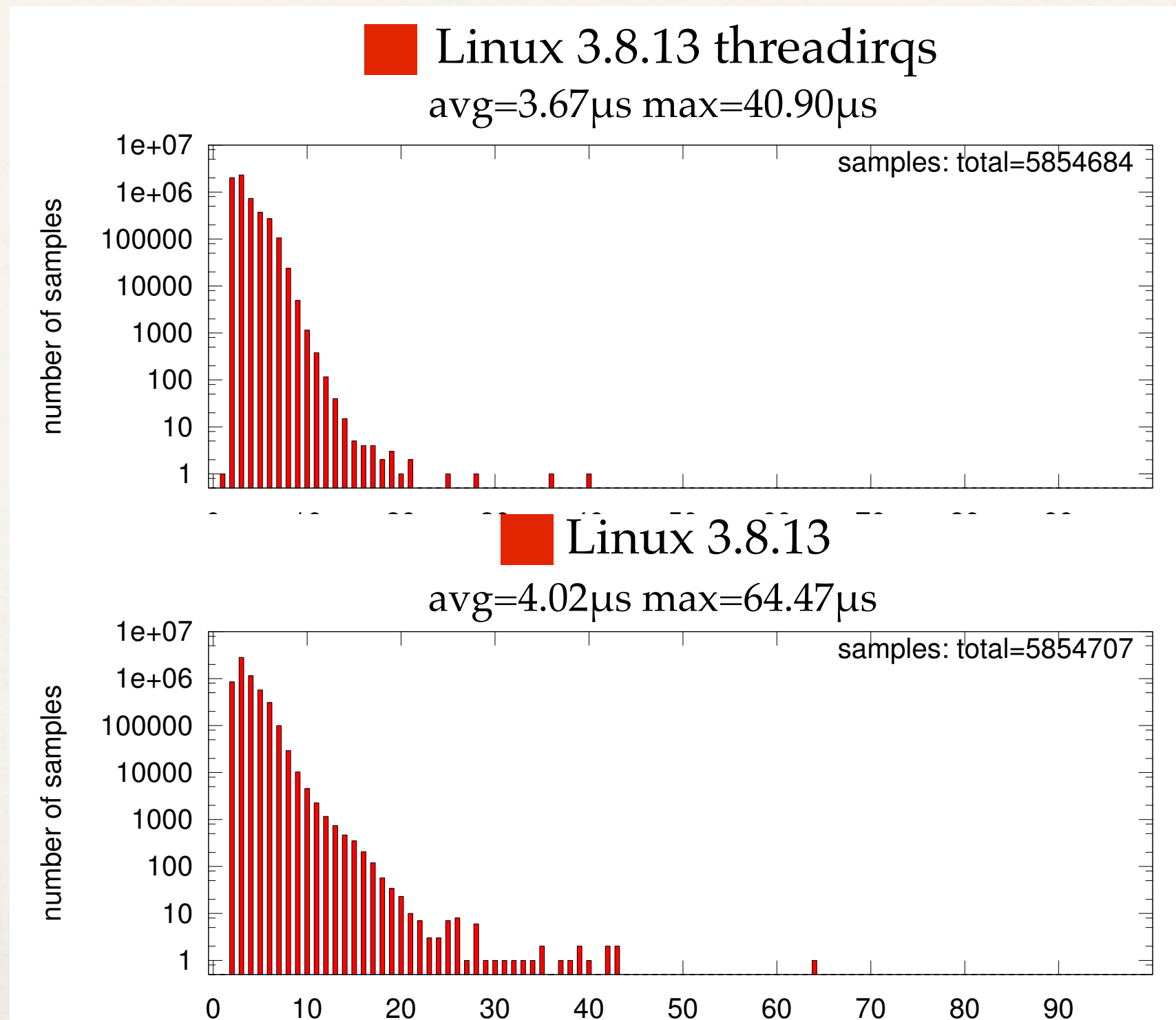


threadirqs in Linux 3.8.13

No Background Tasks



CPU-bound Background Tasks



I/O-bound Background Tasks

