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
- Algorithmic changes based on real-time systems research

PREEMPT_RT (Linux)
Linux as a Real-Time OS

- Optimizing system responsiveness
- Algorithmic changes based on real-time systems research
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 **cyclic_test**
  - **Single**, easy-to-compare measure of scheduling latency as output
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 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

How do LITMUS$^\text{RT}$ and PREEMPT\_RT compare?
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

How do LITMUS\textsuperscript{RT} and PREEMPT\_RT compare?

It is not straightforward to compare them!
Objective

Direct comparison of scheduling latency between LITMUS\textsuperscript{RT} and PREEMPT\_RT (Linux)
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
Scheduling Latency

Time until the highest-priority task is scheduled

 ISR called

brake sensor
HP task

interrupt!

ECU
Scheduling Latency

Time until the highest-priority task is scheduled

 ISR called  scheduler invoked
Scheduling Latency

Time until the highest-priority task is scheduled
Scheduling Latency

Time until the highest-priority task is scheduled

ISR called → scheduler invoked → task picked → switched

perform context switch
Scheduling Latency

Time until the highest-priority task is scheduled

ISR called → scheduler invoked → task picked → switched → perform context switch

Interrupt!

Wake up task

Scheduling latency
How does cyclic test measure Scheduling Latency?
How does cyclicitest measure Scheduling Latency?

Measuring thread is granted real-time status.

POSIX sched_setscheduler()
How does cyclic test measure scheduling latency?
How does cyclic test 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 cyclic test measure Scheduling Latency?
How does cyclic test measure scheduling latency?

Measuring thread returns to best-effort status.

POSIX sched_setscheduler()
cyclic test on LITMUS$^{RT}$

LITMUS$^{RT}$ does not use POSIX API to setup real-time tasks!
cyclic test on \textsc{LITMUS}^{RT}

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

cyclic test works, but does not measure what we expect...
Porting cyclictest to LITMUS\textsuperscript{RT}

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

cyclictest works, but does not measure what we expect...
Porting cyclicTest to LITMUS$^\text{RT}$

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

cyclicTest works, but does not measure what we expect...
Porting cyclictest to LITMUS\textsuperscript{RT}

LITMUS\textsuperscript{RT} API \texttt{task\_mode()}

LITMUS\textsuperscript{RT} API \texttt{task\_mode()}

LITMUS\textsuperscript{RT} Init

Measure

LITMUS\textsuperscript{RT} Exit

No changes in the measurement phase, no bias.
Study
Questions that We Address

Stock Linux

- Userspace
- Scheduler + Dispatcher
- Linux (core)
Questions that We Address

Stock Linux

- Userspace
- Scheduler + Dispatcher
- Linux (core)
The Cost of LITMUS$^\text{RT}$

**Stock Linux**
- Userspace
- Scheduler + Dispatcher
- Linux (core)

**LITMUS$^\text{RT}$**
- Userspace
- Scheduling Policy Plugins
- Dispatcher
- Linux (core)
The Cost of LITMUS$^\text{RT}$

Question 1
How much latency does the scheduling policy interface add to the system?
LITMUSR^T vs. PREEMPT_RT

Stock Linux

Userspace
Scheduler + Dispatcher
Linux (core)

LITMUSR^T

Userspace
Scheduling Policy Plugins
Dispatcher
Linux (core)

PREEMPT_RT

Userspace
Scheduler + Dispatcher
Linux (core)
LITMUS$RT$ vs. PREEMPT$RT$

LITMUS$RT$
- Userspace
- Scheduling Policy Plugins
- Dispatcher
- Linux (core)

PREEMPT$RT$
- Userspace
- Scheduler + Dispatcher
- Linux (core)
LITMUS$^RT$ vs. PREEMPT$^RT$

Question 2
What is the penalty for LITMUS$^RT$ not being based on PREEMPT$^RT$?
Evaluation

- Userspace
- Scheduler + Dispatcher
- Linux (core)
Evaluation
Evaluation

cyclic test + background workload

Scheduler  +  Dispatcher

Linux (core)
Background Workloads

CPU-bound background tasks

I/O-bound background tasks

NO background tasks
Experimental Setup

Different kernels:

1. **LITMUS\textsuperscript{RT} (Linux 3.0)**
   - Partitioned Fixed Priority (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**

\} \text{SCHED\_FIFO}
Experimental Setup

- 16-core Intel Xeon platform
- cyclictest’s standard setup:
  - one real-time task per processor
  - periods: \{1000, 1500, 2000, \ldots\} \mu s

- Duration: 20 minutes per experiment
  - Almost 6 million samples for each case

- Results shown in microseconds
First Scenario

NO

background tasks
No Background Tasks

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th>System</th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS^RT)</td>
<td>3.5</td>
<td>15.3</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>2.9</td>
<td>13.9</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>2.7</td>
<td>11.2</td>
</tr>
</tbody>
</table>
No Background Tasks

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

Scheduling Latency (µs)

<table>
<thead>
<tr>
<th></th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS$^\text{RT}$)</td>
<td>3.5</td>
<td>15.3</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>2.9</td>
<td>13.9</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>2.7</td>
<td>11.2</td>
</tr>
</tbody>
</table>
No Background Tasks

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

Improved max. latency for PREEMPT$_\text{RT}$.

Scheduling Latency (µs)

<table>
<thead>
<tr>
<th></th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS$^\text{RT}$)</td>
<td>3.5</td>
<td>15.3</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>2.9</td>
<td>13.9</td>
</tr>
<tr>
<td>PREEMPT$_\text{RT}$</td>
<td>2.7</td>
<td>11.2</td>
</tr>
</tbody>
</table>
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

Scheduling Latency (μs)

- P-FP (LITMUS^RT): 5.2 μs
- Linux 3.0: 47.6 μs
- PREEMPT_RT: 3.4 μs

Average (99% conf.)

Maximum

49
CPU-bound Background Tasks

Scheduling Latency (μs)  

Latency under PREEMPT_RT is significantly lower.

- P-FP (LITMUS^RT): 5.2 μs
- Linux 3.0: 72.7 μs
- PREEMPT_RT: 17.4 μs

Average (99% confidence)
CPU-bound Background Tasks

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

Latency under PREEMPT_RT is significantly lower.
LITMUS\textsuperscript{RT} vs. Linux 3.0: CPU-bound Background Tasks

![Graph showing scheduling latency comparison]

- **Linux 3.0**
  - avg = 4.22\mu s
  - max = 72.73\mu s

- **P-FP (LITMUS\textsuperscript{RT})**
  - avg = 5.17\mu s
  - max = 47.59\mu s

Samples: total = 5854711

Log scale!
LITMUS$^\text{RT}$ vs. Linux 3.0: CPU-bound Background Tasks

![Graph showing scheduling latency comparison between Linux 3.0 and P-FP(LITMUS$^\text{RT}$).]

- **Linux 3.0**
  - Avg: 4.22µs
  - Max: 72.73µs

- **P-FP(LITMUS$^\text{RT}$)**
  - Avg: 5.17µs
  - Max: 47.59µs

**Samples**: Total = 5854711

3 samples out of ~6 million
LITMUS$^\text{RT}$ vs. Linux 3.0: CPU-bound Background Tasks

- **Linux 3.0**
  - avg = 4.22µs
  - max = 72.73µs

- **P-FP(LITMUS$^\text{RT}$)**
  - avg = 5.17µs
  - max = 47.59µs

Samples: total = 5854711

Slightly worse latencies on average.
LITMUS\textsuperscript{RT} vs. Linux 3.0: CPU-bound Background Tasks

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Scheduling Latency (\(\mu s\)) (Bin size = 1\(\mu s\))}
\end{figure}

- Linux 3.0:
  \begin{itemize}
  \item avg = 4.22\(\mu s\)
  \item max = 72.73\(\mu s\)
  \end{itemize}

- P-FP (LITMUS\textsuperscript{RT}):
  \begin{itemize}
  \item avg = 5.17\(\mu s\)
  \item max = 47.59\(\mu s\)
  \end{itemize}

Samples: total = 5854711

- Extra spinlock
- Lack of low-level optimizations
CPU-bound Background Tasks

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

Latency under PREEMPT_RT is significantly lower.

<table>
<thead>
<tr>
<th></th>
<th>Average (99%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS$^{RT}$)</td>
<td>5.2</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>4.2</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>3.4</td>
</tr>
</tbody>
</table>
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

**Scheduling Latency (μs)**

<table>
<thead>
<tr>
<th>System</th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS^RT)</td>
<td>6.6</td>
<td>3956.5</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>6.4</td>
<td>4300.4</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>4.1</td>
<td>44.2</td>
</tr>
</tbody>
</table>

The chart uses a log scale for the y-axis to represent the scheduling latencies.
I/O-bound Background

Huge impact on scheduling latency under standard Linux.

Huge impact on
scheduling latency
under standard 
Linux.

Scheduling Latency (s)

<table>
<thead>
<tr>
<th></th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS^RT)</td>
<td>3956.5</td>
<td></td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>4300.4</td>
<td>44.2</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>6.6</td>
<td>6.4</td>
</tr>
</tbody>
</table>
I/O-bound Background

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></th>
<th>Average (99%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS^RT)</td>
<td>3956.5</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>4300.4</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>44.2</td>
</tr>
</tbody>
</table>

**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

- cyclicetest was ported to LITMUS\textsuperscript{RT}.
- Should it become the standard tool for evaluating LITMUS\textsuperscript{RT}?
Importance of Feather–Trace

- cyclic tests was ported to LITMUS\textsuperscript{RT}.
- Should it become the standard tool for evaluating LITMUS\textsuperscript{RT}? \textbf{NO!}
Interference?

- ISR called
- Scheduler invoked

OOPS... a higher priority task
Interference?

ISR called

scheduler invoked

another task picked
Interference?

ISA called
scheduler invoked
another task picked

interrupt!
wake up task

ISR called scheduler invoked another task picked

task picked
Interference?

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

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

Overhead-aware schedulability analysis is required!
cyclic test or Feather-Trace?

BOTH!

**cyclic test**
- Practical, easy-to-understand measure
- Can easily compare responsiveness between kernels.

**LITMUS$^\text{RT}$/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\textsuperscript{RT}: small overheads in comparison with stock Linux

PREEMPT\_RT is highly necessary for Linux as a RTOS
LITMUS\textsuperscript{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

Scheduling Latency (μs)

- Average
- Maximum

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 3.0</td>
<td>2.9</td>
<td>13.9</td>
</tr>
<tr>
<td>Linux 3.8.13</td>
<td>2.9</td>
<td>19.7</td>
</tr>
</tbody>
</table>
No Background Tasks

<table>
<thead>
<tr>
<th>Scheduling Latency (μs)</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Similar averages**

- **Linux 3.0**
  - Average: 2.9
  - Maximum: 13.9

- **Linux 3.8.13**
  - Average: 2.9
  - Maximum: 19.7
No Background Tasks

**Linux 3.0**
- avg = 2.89µs
- max = 13.89µs

**Linux 3.8.13**
- avg = 2.89µs
- max = 19.73µs

Samples: Total = 5854779

Samples: Total = 5854801

Similar shapes
CPU-bound Background Tasks

**Scheduling Latency (μs)**

- **Average:**
  - Linux 3.0: 4.2 μs
  - Linux 3.8.13: 4.0 μs
- **Maximum:**
  - Linux 3.0: 72.7 μs
  - Linux 3.8.13: 64.5 μs
CPU-bound Background Tasks

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></th>
<th>Linux 3.0</th>
<th>Linux 3.8.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>4.2</td>
<td>4.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>72.7</td>
<td>64.5</td>
</tr>
</tbody>
</table>

Similar averages
CPU-bound Background Tasks

![Histograms showing scheduling latency for Linux 3.0 and Linux 3.8.13]

- **Linux 3.0**
  - Average: 4.22μs
  - Maximum: 72.73μs

- **Linux 3.8.13**
  - Average: 4.02μs
  - Maximum: 64.47μs

Both histograms show similar shapes with a peak around 1μs and a long tail to the right.
I/O-bound Background Tasks

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 3.0</td>
<td>6.4</td>
<td>4300.4</td>
</tr>
<tr>
<td>Linux 3.8.13</td>
<td>6.2</td>
<td>5464.1</td>
</tr>
</tbody>
</table>

Log scale!
I/O-bound Background Tasks

Scheduling Latency (μs)

- **Average**
- **Maximum**

<table>
<thead>
<tr>
<th></th>
<th>Linux 3.0</th>
<th>Linux 3.8.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar</td>
<td>6.4</td>
<td>6.2</td>
</tr>
<tr>
<td>Averages</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Linux 3.0: 4300.4 μs
- Linux 3.8.13: 5464.1 μs
I/O-bound Background Tasks

- **Linux 3.0**
  - avg = 6.39μs
  - max = 4300.43μs
  - samples: total = 5854674

- **Linux 3.8.13**
  - avg = 6.23μs
  - max = 5464.07μs
  - samples: total = 5854773

Similar shapes
LITMUS$^{RT}$'s plugins
No Background Tasks

### Scheduling Latency ($\mu s$)

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP</td>
<td>3.5</td>
<td>15.1</td>
</tr>
<tr>
<td>PSN-EDF</td>
<td>3.5</td>
<td>26.2</td>
</tr>
<tr>
<td>GSN-EDF</td>
<td>3.1</td>
<td>14.3</td>
</tr>
</tbody>
</table>
No Background Tasks

- **GSN-EDF**
  - avg = 3.06μs max = 14.34μs
  - samples: total = 5854797

- **PSN-EDF**
  - avg = 3.45μs max = 26.17μs
  - samples: total = 5854783

- **P-FP**
  - avg = 3.45μs max = 15.13μs
  - samples: total = 5854818

Similar shapes
No Background Tasks

- **GSN-EDF**
  - avg = 3.06μs
  - max = 14.34μs
  - samples: total = 5854797

- **PSN-EDF**
  - avg = 3.45μs
  - max = 26.17μs
  - samples: total = 5854783

- **P-FP**
  - avg = 3.45μs
  - max = 15.13μs
  - samples: total = 5854818

**Similar shapes**

2 samples
CPU-bound Background Tasks

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P-FP</strong></td>
<td>5.2</td>
<td>47.6</td>
</tr>
<tr>
<td><strong>PSN-EDF</strong></td>
<td>5.1</td>
<td>73.3</td>
</tr>
<tr>
<td><strong>GSN-EDF</strong></td>
<td>5.8</td>
<td>60.2</td>
</tr>
</tbody>
</table>
CPU-bound Background Tasks

GSN-EDF
avg=5.81μs max=60.20μs

PSN-EDF
avg=5.14μs max=73.27μs

P-FP
avg=5.17μs max=47.59μs

Similar shapes
CPU-bound Background Tasks

GSN-EDF
avg = 5.81μs max = 60.20μs

PSN-EDF
avg = 5.14μs max = 73.27μs

P-FP
avg = 5.17μs max = 47.59μs

Similar shapes
I/O-bound Background Tasks

Scheduling Latency (μs)

- **P-FP**
  - Average: 6.6
  - Maximum: 3956.5

- **PSN-EDF**
  - Average: 6.6
  - Maximum: 3875.0

- **GSN-EDF**
  - Average: 11.0
  - Maximum: 3905.8

The chart shows the scheduling latency in microseconds (μs) for different background tasks. The y-axis represents the log scale for the latency values.
I/O-bound Background Tasks

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP</td>
<td>3956.5</td>
<td>3905.8</td>
</tr>
<tr>
<td>PSN-EDF</td>
<td>3875.0</td>
<td>3905.8</td>
</tr>
<tr>
<td>GSN-EDF</td>
<td>11.0</td>
<td></td>
</tr>
</tbody>
</table>

Similar results for P-FP and PSN-EDF
I/O-bound Background Tasks

<table>
<thead>
<tr>
<th>Scheduling Latency (μs)</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>3956.5</td>
<td>3875.0</td>
</tr>
</tbody>
</table>

Similar results for P-FP and PSN-EDF.

Higher average for GSN-EDF.
I/O-bound Background Tasks

- **GSN-EDF**
  - avg = 10.95\(\mu\)s
  - max = 3905.79\(\mu\)s

- **PSN-EDF**
  - avg = 6.56\(\mu\)s
  - max = 3874.99\(\mu\)s

- **P-FP**
  - avg = 6.60\(\mu\)s
  - max = 3956.48\(\mu\)s

Samples: total = 5854793

Samples: total = 5854606

Samples: total = 5854660

Similar shapes
I/O-bound Background Tasks

- **GSN-EDF**
  - avg = 10.95μs max = 3905.79μs
  - Higher average

- **PSN-EDF**
  - avg = 6.56μs max = 3874.99μs

- **P-FP**
  - avg = 6.60μs max = 3956.48μs

**Similar shapes**
threadirqs in Linux 3.8.13
No Background Tasks

Linux 3.8.13 threadirqs
avg=2.82μs max=25.15μs

samples: total=5854778

Linux 3.8.13
avg=2.89μs max=19.73μs

samples: total=5854801
CPU-bound Background Tasks

- **Linux 3.8.13 threadirqs**
  - avg=3.67μs max=40.90μs
  - samples: total=5854684

- **Linux 3.8.13**
  - avg=4.02μs max=64.47μs
  - samples: total=5854707
I/O-bound Background Tasks

Linux 3.8.13 threadirqs
avg=5.89μs max=5203.38μs

samples: total=5854724

Linux 3.8.13
avg=6.23μs max=5464.07μs

samples: total=5854773