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

PREEMPT_RT (Linux)

Algorithmic changes based on real-time systems research

LITMUS_RT
Linux Testbed for Multiprocessor Scheduling in Real-Time Systems
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 **cyclicstest**

- **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 LITMUSRT 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$^{RT}$ and PREEMPT_RT compare?

It is not straightforward to compare them!
Objective

Direct comparison of scheduling latency between LITMUS$^\text{RT}$ 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

ISR called
Scheduling Latency

Time until the highest-priority task is scheduled

ISR called
Scheduler invoked
Scheduling Latency

Time until the highest-priority task is scheduled

ISR called
scheduler invoked
task picked
Scheduling Latency

Time until the highest-priority task is scheduled

 ISR called
 scheduler invoked
 task picked
 switched

interrupt!

wake up task

Perform context switch
Scheduling Latency

Time until the highest-priority task is scheduled

ISR called ➔ scheduler invoked ➔ task picked ➔ switched

interrupt! ➔ wake up task ➔ Perform context switch
How does cyclic test measure Scheduling Latency?
How does cyclic test measure Scheduling Latency?

Measuring thread is granted real-time status.

POSIX sched_setscheduler()
How does cyclic test measure Scheduling Latency?
How does cyclicertest 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 works, but does not measure what we expect...

LITMUS\textsuperscript{RT} does not use POSIX 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$^{RT}$

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

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

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

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<sup>RT</sup>
The Cost of LITMUS\textsuperscript{RT}

Question 1
How much latency does the scheduling policy interface add to the system?
**LITMUS$RT$ vs. PREEMPT$RT$**

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

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

**PREEMPT$RT$**
- Userspace
- Scheduler + Dispatcher
- Linux (core)
LITMUS<sup>RT</sup> vs. PREEMPT<sub>RT</sub>

**Stock Linux**

- Userspace
- Scheduler + Dispatcher
- Linux (core)

**LITMUS<sup>RT</sup>**

- Userspace
- Scheduling Policy Plugins
- Dispatcher
- Linux (core)

**PREEMPT<sub>RT</sub>**

- Userspace
- Scheduler + Dispatcher
- Linux (core)
LITMUS\textsuperscript{RT} vs. PREEMPT\_RT

\begin{itemize}
\item LITMUS\textsuperscript{RT}
  \begin{itemize}
  \item Userspace
  \item Scheduling Policy Plugins
  \item Dispatcher
  \item Linux (core)
  \end{itemize}
\item PREEMPT\_RT
  \begin{itemize}
  \item Userspace
  \item Scheduler + Dispatcher
  \item Linux (core)
  \end{itemize}
\end{itemize}

Tuesday, July 9, 13
Question 2
What is the penalty for LITMUS^{RT} not being based on PREEMPT^{RT}?
Evaluation

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

Userspace

Scheduler
+
Dispatcher

Linux (core)
Evaluation

cyclic

+ 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 \textbf{Fixed Priority} (P-FP),
   - Partitioned \textbf{EDF} with synchronization support (PSN-EDF),
   - Global \textbf{EDF} with synchronization support (GSN-EDF)

2. **PREEMPT_RT** (Linux 3.8.13)

3. Unpatched Linux 3.0 and Linux 3.8.13
Experimental Setup

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

Scheduling Latency ($\mu$s)

- **P-FP (LITMUS^RT)**: Average (99% conf.) = 3.5, Maximum = 15.3
- **Linux 3.0**: Average (99% conf.) = 2.9, Maximum = 13.9
- **PREEMPT_RT**: Average (99% conf.) = 2.7, Maximum = 11.2
Similar max. and avg. latency for Linux 3.0 and LITMUS$^{RT}$. 

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></th>
<th>Average (99% conf.)</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-FP (LITMUS$^{RT}$)</td>
<td>15.3</td>
<td></td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>13.9</td>
<td></td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>11.2</td>
<td></td>
</tr>
</tbody>
</table>
Similar max. and avg. latency for Linux 3.0 and LITMUS\textsuperscript{RT}.

Improved max. latency for PREEMPT\_RT.

Scheduling Latency (\textmu s)

- P-FP (LITMUS\textsuperscript{RT})
  - Average (99\% conf.): 3.5
  - Maximum: 15.3

- Linux 3.0
  - Average (99\% conf.): 2.9
  - Maximum: 13.9

- PREEMPT\_RT
  - Average (99\% conf.): 2.7
  - Maximum: 11.2

Similar max. and avg. latency for Linux 3.0 and LITMUS\textsuperscript{RT}. Improved max. latency for PREEMPT\_RT.
Second Scenario

CPU-bound background tasks

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

- 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)

<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>5.2</td>
<td>47.6</td>
</tr>
<tr>
<td>Linux 3.0</td>
<td>4.2</td>
<td>72.7</td>
</tr>
<tr>
<td>PREEMPT_RT</td>
<td>3.4</td>
<td>17.4</td>
</tr>
</tbody>
</table>
CPU-bound Background Tasks

Latency under PREEMPT_RT is significantly lower.

Scheduling Latency ($\mu$s)  Average (99%)

P-FP (LITMUS^RT)  5.2  47.6
Linux 3.0  4.2  72.7
PREEMPT_RT  3.4  17.4
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

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

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

Samples: total = 585471

Number of samples vs. Scheduling Latency (µs) (Bin size = 1 µs)
LITMUS$^{RT}$ vs. Linux 3.0: CPU-bound Background Tasks

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

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

Samples: total = 5854711

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

Scheduling Latency (µs) (Bin size = 1µs)

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

- **P-FP(LITMUS\textsuperscript{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

![Graph showing scheduling latency distribution for Linux 3.0 and P-FP(LITMUS\textsuperscript{RT}) with bin size of 1\,µs.]

- **Linux 3.0**
  - avg=4.22\,µs
  - max=72.73\,µs
- **P-FP(LITMUS\textsuperscript{RT})**
  - avg=5.17\,µs
  - max=47.59\,µs

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.
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**

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

Causes a lot of system calls and interrupts
I/O-bound Background Tasks

Scheduling Latency (µs)

- **P-FP (LITMUS^RT)**
  - Average (99% conf.): 3956.5
  - Maximum: 44.2

- **Linux 3.0**
  - Average (99% conf.): 4300.4
  - Maximum: 6.4

- **PREEMPT_RT**
  - Average (99% conf.): 4.1
  - Maximum: 6.6

The diagram uses a log scale!
I/O-bound Background

Huge impact on scheduling latency under standard Linux.

Scheduling Latency (ms)

- P-FP (LITMUS^RT): 3956.5
- Linux 3.0: 4300.4
- PREEMPT_RT: 44.2

Average (99% conf.)

Maximum
I/O-bound Background

**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$^\text{RT}$ vs. PREEMPT$\_\text{RT}$
Summary

1. Cost of the scheduling plugin layer

The overhead introduced by LITMUS\textsuperscript{RT} is small

2. LITMUS\textsuperscript{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

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

• cyclic test was ported to LITMUS\textsuperscript{RT}.

• Should it become the standard tool for evaluating LITMUS\textsuperscript{RT}? \textbf{NO!}
Interference?

ISR called  scheduler invoked  another task picked
Interference?

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!

ISR called  scheduler invoked  another task picked  task picked
cyclictest or Feather-Trace?

BOTH!

**cyclictest**

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

**LITMUS\textsuperscript{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)

<table>
<thead>
<tr>
<th></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

Scheduling Latency (μs)

<table>
<thead>
<tr>
<th></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>

Similar averages
No Background Tasks

Linux 3.0
avg=2.89µs max=13.89µs

samples: total=5854779

Linux 3.8.13
avg=2.89µs max=19.73µs

samples: total=5854801

Similar shapes
CPU-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>4.2</td>
<td>72.7</td>
</tr>
<tr>
<td>Linux 3.8.13</td>
<td>4.0</td>
<td>64.5</td>
</tr>
</tbody>
</table>
CPU-bound Background Tasks

![Bar chart showing scheduling latency for Linux 3.0 and Linux 3.8.13]

- **Linux 3.0**
  - Average: 4.2 µs
  - Maximum: 72.7 µs

- **Linux 3.8.13**
  - Average: 4.0 µs
  - Maximum: 64.5 µs

**Similar averages**
CPU-bound Background Tasks

Similar shapes
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>
I/O-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>6.4</td>
<td>6.2</td>
</tr>
<tr>
<td>Maximum</td>
<td>4300.4</td>
<td>5464.1</td>
</tr>
</tbody>
</table>

Similar averages
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\textsuperscript{RT}'s plugins
No Background Tasks

**Scheduling Latency (μs)**

- **Average**
  - P-FP: 3.5
  - PSN-EDF: 3.5
  - GSN-EDF: 3.1

- **Maximum**
  - P-FP: 15.1
  - PSN-EDF: 26.2
  - GSN-EDF: 14.3
No Background Tasks

- **GSN-EDF**
  - avg = 3.06μs
  - max = 14.34μs

- **PSN-EDF**
  - avg = 3.45μs
  - max = 26.17μs

- **P-FP**
  - avg = 3.45μs
  - max = 15.13μs
No Background Tasks

GSN-EDF
avg=3.06μs max=14.34μs

PSN-EDF
avg=3.45μs max=26.17μs

P-FP
avg=3.45μs max=15.13μs

Similar shapes
CPU-bound Background Tasks

Scheduling Latency (µs)

- **P-FP**: Average - 5.2, Maximum - 47.6
- **PSN-EDF**: Average - 5.1, Maximum - 73.3
- **GSN-EDF**: Average - 5.8, Maximum - 60.2
CPU-bound Background Tasks

- **GSN-EDF**
  - avg = 5.81μs
  - max = 60.20μs
  - samples: total = 5854728

- **PSN-EDF**
  - avg = 5.14μs
  - max = 73.27μs
  - samples: total = 5854739

- **P-FP**
  - avg = 5.17μs
  - max = 47.59μs
  - samples: total = 5854719

Similar shapes
CPU-bound Background Tasks

GSN-EDF
avg=5.81µs max=60.20µs
samples: total=5854728

PSN-EDF
avg=5.14µs max=73.27µs
samples: total=5854739

P-FP
avg=5.17µs max=47.59µs
samples: total=5854719

5 samples

Similar shapes
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>6.6</td>
<td>3956.5</td>
</tr>
<tr>
<td>PSN-EDF</td>
<td>6.6</td>
<td>3875.0</td>
</tr>
<tr>
<td>GSN-EDF</td>
<td>11.0</td>
<td>3905.8</td>
</tr>
</tbody>
</table>

log scale!
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>11.0</td>
</tr>
<tr>
<td>GSN-EDF</td>
<td></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>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td></td>
</tr>
<tr>
<td><strong>P-FP</strong></td>
<td>3956.5</td>
</tr>
<tr>
<td><strong>PSN-EDF</strong></td>
<td>3875.0</td>
</tr>
<tr>
<td><strong>GSN-EDF</strong></td>
<td>11.0</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=10.95µs max=3905.79µs

PSN-EDF
avg=6.56µs max=3874.99µs

P-FP
avg=6.60µs max=3956.48µs

Similar shapes
I/O-bound Background Tasks

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

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

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

**Higher average**

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

**Linux 3.8.13 threadirqs**

- avg = 2.82μs
- max = 25.15μs

**Linux 3.8.13**

- avg = 2.89μs
- max = 19.73μs
CPU-bound Background Tasks

Linux 3.8.13 threadirqs
avg=3.67μs max=40.90μs

Linux 3.8.13
avg=4.02μs max=64.47μs
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