How Linux RT_PREEMPT Works

A common observation about real time systems is that the cost of the increased determinism of real time is decreased throughput and increased average latency. Does this hold true for Linux PREEMPT_RT_FULL?

This presentation enumerates some of the design choices and implementation that enable Linux PREEMPT_RT_FULL real time and the resulting performance implications.

Frank Rowand, Sony Network Entertainment  
October 28, 2011
The Question

The cost of the increased determinism of real time

- Increased average latency
- Decreased throughput

Is this true for Linux PREEMPT_RT_FULL?
Compare the latency of an application on kernels built with:

1) CONFIG_PREEMPT_NONE

2) CONFIG_PREEMPT_RT_FULL
Some Random Data

Test System:
- ARM11 MPCore development system
- 4 cpus
- 210 Mhz processor clock
- L1 cache 64 Kbyte per processor
- L2 cache 1 Mbyte unified
- Linux 3.0.6-rt17
Cyclic test wakeup latency, no load
blue: PREEMPT_NONE  magenta: PREEMPT_RT_FULL
Latency (Response Time)

Kernel without RT patchset:

+ smaller average
+ smaller minimum
- larger maximum

PREEMPT RT enabled:

- larger average
- larger minimum
+ smaller maximum
+ more consistent
### Statistics

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREEMPT_NONE</td>
<td>29</td>
<td>38</td>
<td>9186</td>
</tr>
<tr>
<td>PREEMPT_RT_FULL</td>
<td>35</td>
<td>41</td>
<td>95</td>
</tr>
</tbody>
</table>
Latency (Response Time)

Next graph shows an old kernel, circa 2009

Hardware configuration: unknown
Red Hat Enterprise Linux
Red Hat MRG tuned

Messaging Workload

Response time

source: Red Hat
The Answer

The cost of the increased determinism of real time
- Increased average latency

Is this true for Linux PREEMPT_RT_FULL?

YES
Some Random Data

Compare the throughput of an application on kernels built with:

1) CONFIG_PREEMPT_NONE
2) CONFIG_PREEMPT_RT_FULL
Some Random Data

The workload I used for the throughput results
- is not realistic
- is not reasonable
- violates real time application design rules
- is stupid!
- but was easy to implement...
Some Random Data

Test System (same as first test system):
- ARM11 MPCore development system
- 4 cpus
- 210 Mhz processor clock
- L1 cache 64 Kbyte per processor
- L2 cache 1 Mbyte unified
- Linux 3.0.6-rt17
Test Variables

UP vs SMP

SMP, maxcpus=4 vs SMP maxcpus=1

workload: SCHED_FIFO vs SCHED_NORMAL

1, 2, or 4 instances of the workload

Permutations of variables results in 10 tests
PREEMPT_RT_FULL without debug
red: user green: sys blue: user + sys magenta: real

 combustible

 test #

 seconds
PREEMPT_RT_FULL without debug
red: user green: sys blue: user + sys magenta: real

seconds

PREEMPT_NONE without debug
red: user green: sys blue: user + sys magenta: real

seconds
PREEMPT_RT_FULL without debug
red: user green: sys blue: user + sys magenta: real

.seconds

PREEMPT_NONE without debug
red: user green: sys blue: user + sys magenta: real

.seconds

test #
test #
PREEMPT_RT_FULL vs. PREEMPT_NONE (without debug)
red: user
PREEMPT_RT_FULL vs. PREEMPT_NONE (without debug)
green: sys
PREEMPT_RT_FULL vs. PREEMPT_NONE (without debug)
blue: user + sys
PREEMPT_RT_FULL vs. PREEMPT_NONE (without debug)
magenta: real
The Answer

The cost of the increased determinism of real time

- Increased average latency
- Decreased throughput

Is this true for Linux PREEMPT_RT_FULL?

YES
Part 2

This presentation enumerates some of the design choices and implementation that enable Linux PREEMPT_RT_FULL real time and the resulting performance implications.
Enabling real-time Linux

- preemptible kernel
- locking
- threaded interrupt handlers
- threaded softirq
Non-Preemptible Kernel

When a task invokes a system call, the system call must complete (or sleep due to blocking on a resource) before another task can be scheduled.

Preemption can not occur during the execution of the system call.
Non-Preemptible Kernel

Preemption can not occur during the execution of the system call.

Scheduling may occur on:
- completion of system call
- system call sleeping
Non-Preemptible Kernel

Problems of typical non-preemptible kernel:

- kernel path lengths non-deterministic
- longest kernel path has long duration
- large variance in kernel path length
Non-Preemptible Kernel

Next slide illustrates non-preemptible kernel.
Non-Preemptible Kernel

Next slide illustrates non-preemptible kernel.

Adding some complexity:

- 2 external events occur
- lock (critical section) during syscall
syscall  external                              syscall

events                                completes

RT task                              
Normal task                           
syscall                               
IRQ 1 handler                         
IRQ 2 handler                         
critical section                      

RT task runs                          
wake RT task                          
RT task runs                           

Preemptible Kernel

Mainline 2.6 and 3.0 kernel

CONFIG_PREEMPT_NONE
  No forced kernel preemption

CONFIG_PREEMPT_VOLUNTARY
  Explicit preemption points in kernel

CONFIG_PREEMPT
  All kernel code (not in critical section) preemptible
Preemptible Kernel

RT_PREEMPT patch renames config option:

Vanilla 2.6 kernel

CONFIG_PREEMPT

RT_PREEMPT 2.6 kernel

CONFIG_PREEMPT_DESKTOP
Preemptible Kernel

RT_PREEMPT patch renames config option:

Vanilla 3.0 kernel

CONFIG_PREEMPT

RT_PREEMPT 3.0 kernel

CONFIG_PREEMPT_LL
Preemptible Kernel

Mainline 2.6 and 3.0 kernel

CONFIG_PREEMPT “fully preemptible”
- except when preemption is explicitly disabled
- except when interrupts are explicitly disabled
- except when a lock is held
  (“in a critical section”)

Preemptible Kernel

Next slide illustrates preemptible kernel.
Score

- added 0 schedule with context switch

+ shorter wakeup latency
Preemptible Kernel

RT_PREEMPT 2.6 kernel

CONFIG_PREEMPT_RT

“fully preemptible”
- except when preemption is explicitly disabled
- except when interrupts are explicitly disabled
- except when a raw spinlock is held
Preemptible Kernel

RT_PREEMPT 3.0 kernel

CONFIG_PREEMPT_RT_FULL

“fully preemptible”
- except when preemption is explicitly disabled
- except when interrupts are explicitly disabled
- except when a raw spinlock is held
Preemptible Kernel

CONFIG_PREEMPT_RT
CONFIG_PREEMPT_RT_FULL

Most kernel locks are converted to preemptible priority inheritance mutex.

Some kernel locks are converted to non-preemptible raw spinlock.
Preemptible Kernel

Next slide illustrates preemptible kernel with spinlocks converted to mutexes.
RT task
Normal task
syscall
IRQ 1 handler
IRQ 2 handler

critical section (mutex)

syscall     external
devices
completes

RT task
runs

wake
RT task

RT task runs
Score

- added 0 schedule with context switch
+ shorter wakeup latency
Priority Inheritance Mutex

- May result in more schedule events.

- Avoids priority inversion.

- Reader-Writer lock limited to one concurrent reader to minimize PI complexity.
  - Limits scalability of multiple readers.
non-PI Mutex – Priority Inversion

higher number is higher priority

A gets lock
B wakes

A

B

C

A gets lock
B wakes
C wakes
C blocks on lock
PI Mutex

higher number is higher priority

A
B
C

A gets lock
B wakes
C wakes
C blocks on lock, A boosted to 3
A releases lock, unboosted back to 1, C gets lock
Threaded Interrupt Handler
Overview Of
Interrupt handling algorithm

- Save context
- Handle “highest priority” interrupt
  Interrupt handler executes in interrupt mode
  irq_exit() may process softirq or wake softirqd
- Iterate over active interrupts (arch dependent)
- Schedule
- Restore context
  returning either to previous process or
to newly scheduled process
Overview Of Threaded Interrupt handling algorithm

- Save context
- Handle “highest priority” interrupt
  Wake Interrupt handler thread.
  irq_exit() may process softirq or wake softirqd
- Iterate over active interrupts (arch dependent)
- Schedule
- Restore context
  returning either to previous process or to newly scheduled process

Interrupt handler thread executes when scheduled.
Threaded Interrupt Handler

RT_PREEMPT patchset converts almost all drivers to threaded model. (Timer handler executes in interrupt context.)

2.6.xx mainline does not convert drivers to threaded model. Each driver must be explicitly converted.
Threaded Interrupt Handler

Priorities must be properly set for:

- interrupt handler threads
- softirq threads
- other kernel threads
- real time application processes / threads

Do not expect default priorities to be proper.
Preemptible Kernel

Next slide illustrates preemptible kernel with interrupt threads.
RT task
Normal task
syscall
IRQ 1 handler
IRQ 2 handler
syscall
external events
syscall completes

critical section (mutex)

RT task runs
wake RT task
RT task runs
Score

- added 2 schedule with context switch
+ shorter wakeup latency
Other Interrupt Overhead
Other Interrupt Overhead

CONFIG_PREEMPT_RT and
CONFIG_PREEMPT_RT_FULL changes:

irq_exit() may process softirq or wake ksoftirqd thread

to:

irq_exit() may wake ksoftirqd thread
Score

- added 1 schedule with context switch

+ shorter wakeup latency
Other Interrupt Overhead

raise_softirq(), (softirq trigger) typically called from:

- irq context (timer softirq)
- interrupt thread

but can be called from anywhere in the kernel.

Previous slides show trigger from irq context.
Other Locking Overhead

CONFIG_TREE_PREEMPT_RCU

Evolving in the early 3.0 RT patches...

Not analyzed in this presentation.
Other Locking Overhead

local_lock()

Uses migrate_disable() instead of preempt_disable().

Evolving in the early 3.0 RT patches...

Not analyzed in this presentation.
Real Life

Real systems are much more complicated than the previous diagrams.

Other scenarios can generate different performance improvements or penalties.
Recap: Enabling real-time Linux

- preemptible kernel
- locking
- interrupt handlers
- threaded softirq
Impact of Real-Time Features

+ Variance of real-time task latency decreased
+ Maximum real-time task latency decreased
- Average real-time task latency may be increased
- Throughput decreased
Recap: The Answer

The cost of the increased determinism of real time
- Increased average latency
- Decreased throughput

Is this true for Linux PREEMPT_RT_FULL?

YES
Questions?
How to get a copy of the slides

1) leave a business card with me

2) frank.rowand@am.sony.com