

Introduction to Realtime Linux

Bryan Che



Southern California Linux Expo



What is Realtime Performance?

- Realtime is not about faster performance
- Realtime is not about higher throughput
- Realtime may even be slower than non-realtime!

So, then what is realtime performance, and why would anyone want it?



Realtime is About Deterministic Performance





- •Non-Realtime
 - Average times highly variable
 - No prioritization of traffic
 - Favors throughput

- •Realtime
 - Highly deterministic time
 - Prioritization of traffic
 - Sacrifices throughput for low, deterministic latency

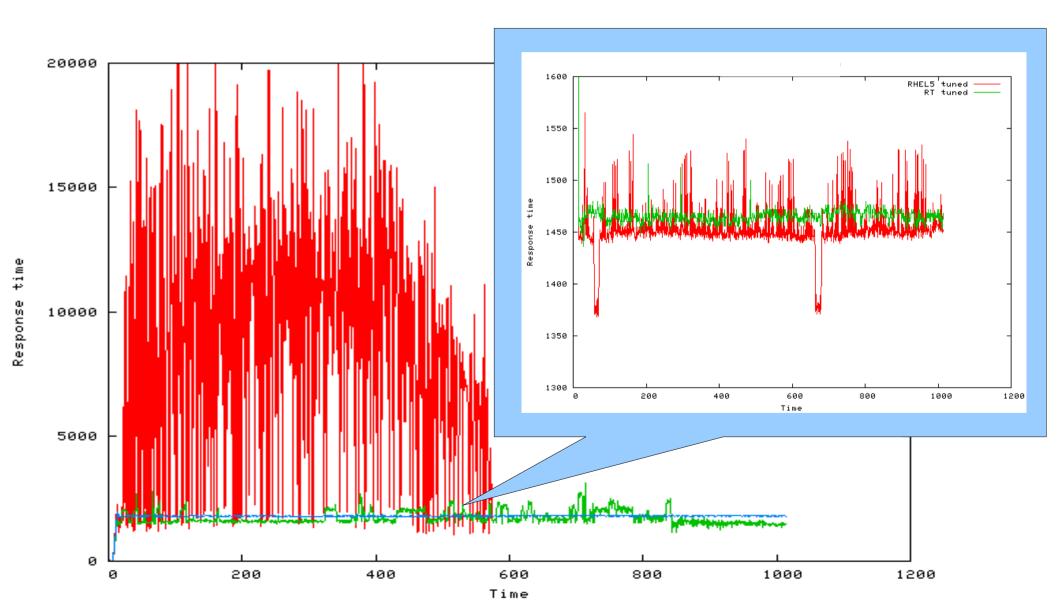


Why Realtime Linux?

- Enables applications and transactions to run predictably, with guaranteed response times
 - Provides microsecond accuracy
- Provides competitive advantage & meets Service Level Agreements
 - Multimedia: precise timing and synchronization
 - Travel web site: missed booking
 - Program trading: missed trades
 - Command & Control: life & death
- Industries particularly interested in Realtime Linux include Government, Defense, Financial Services, Telecommunications, Manufacturing, etc.



What Does Realtime Linux Provide?





What is Realtime Linux?

- Patchset developed at kernel.org community which adds to the standard Linux kernel:
 - Full preemption
 - Threaded IRQs
- Breaks down long-running, un-preemptable code paths to provide responsive behavior
- Large patchset:
 - Diffstat of patch set 2.6.26.6-rt11 shows:
 - 664 files changed, 37806 insertions(+), 4217 deletions(-)

Key Developers

 Ingo Molnar (Maintainer, Red Hat), Thomas Gleixner (Red Hat contractor), Steven Rostedt (Red Hat), Paul McKenney (IBM), John Stultz (IBM), Gregory Haskins (Novell), Peter Zijlstra (Red Hat), etc



Key Changes in the Realtime Kernel

Preemption

- Most locks converted to rt_mutex
- priority inheritance for mutexes
- threaded interrupt handlers (both hard and soft)
- Spinlocks can sleep
- Interrupts not turned off for almost all operations
- high-resolution timers
- Completely Fair Scheduler (CFS) *
- Read-Copy-Update (RCU) *
- Ftrace tracing logic

*Now in Upstream Kernel

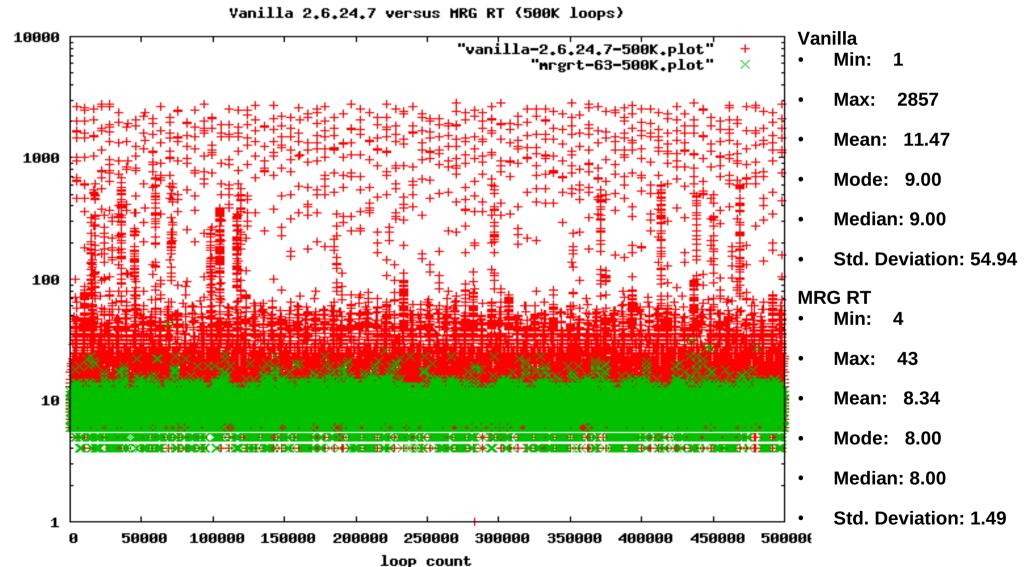


Key Changes in the C Library for Realtime

- pthread_mutex_t has kernel support for PRIO_INHERIT
 - Priority Inheritance is a mechanism used to avoid the deadlock condition known as Priority Inversion
 - The realtime kernels implement priority inheritance (PI) in *futexes* (fast userspace mutexes) used by pthreads
- Fast user-space mutexes (futexes) used for pthread mutexes
- POSIX interfaces to scheduler APIs
 - sched_*
- Timer interfaces
- Note that you don't have to have a realtime kernel for most of these APIs to work



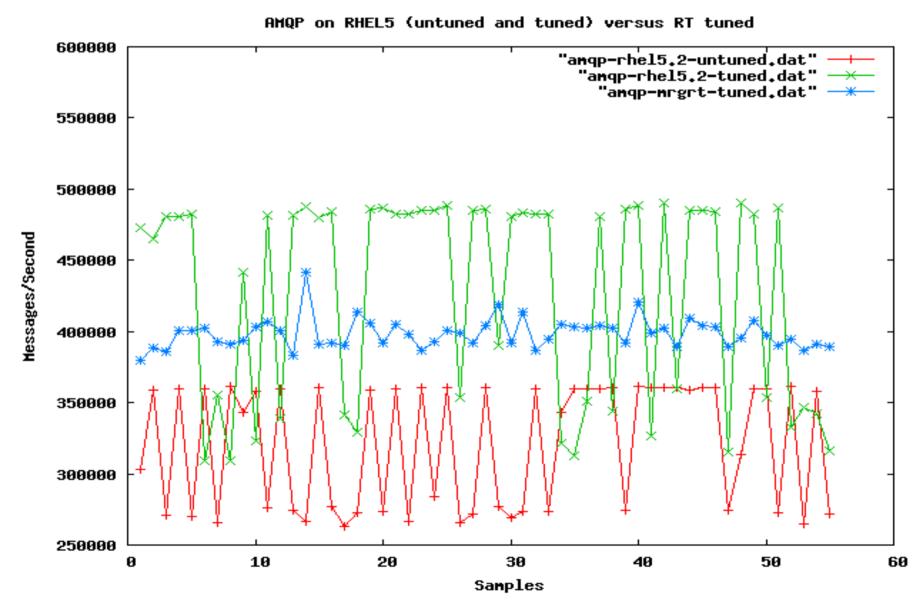
Reducing Scheduling Latency



event response (microseconds)



Realtime Throughput





Realtime Performance Tools

FTrace

- Runtime trace capture of longest latency codepaths both kernel and application. Peak detector
- Selectable triggers for threshold tracing
- Detailed kernel profiles based on latency triggers

TUNA

 Dynamically control tuning parameters like process affinity, parent & threads, scheduling policy, device IRQ priorities, etc.

Standard Linux performance tools

- Gdb, OProfile Frysk source level debuggers & profiler
- SystemTap, kprobe kernel event tracing and dynamic data collection
- kexec/kdump standard kernel dump/save core capabilities



Realtime Java With Realtime Linux

- Standard Java deployments typically have highly undeterministic performance —especially because of garbage collection
- JSR 1 provides a realtime specification for Java and realtime JVMs
 - Requires an underlying realtime operating system to provide priority inheritance and preemption —like realtime Linux!
 - Provides deterministic garbage collection, realtime threads, and deterministic performance



Red Hat and IBM have partnered to deliver Realtime Java on Realtime Linux for the US Navy DDG 1000 Zumwalt Class Destroyer Program



How to Develop for Realtime Linux

Use POSIX threads

 finer grained applications mean more parallelism, so can take advantage of multiple cores

Use POSIX threads synchronization mechanisms

- Mutexes
- Barriers
- Condition variables

Set appropriate priorities for your threads

- Any SCHED_FIFO thread is higher priority than any SCHED_OTHER thread
- ensure that your high priority threads don't hog the processor



How to Deploy Realtime Linux

Tune your system!

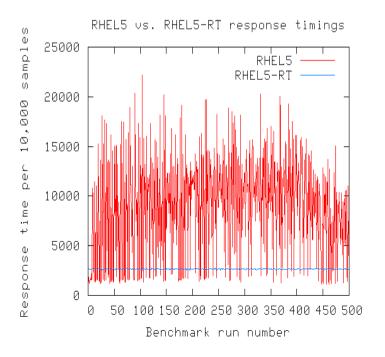
- No two applications behave the same
- use *tuna* to tweak priorities and affinities
- use oprofile to find application hotspots
- use *ftrace* to find long latency areas

Dedicate processors to your application threads

- Use tuna or taskset to bind threads to specific processors and move other threads off
- 4-way and 8-way processors getting cheaper

Use cpu affinity field in /proc/irqs/<n>/smp_affinity to bind interrupts to specific processors

tuna can do this easily





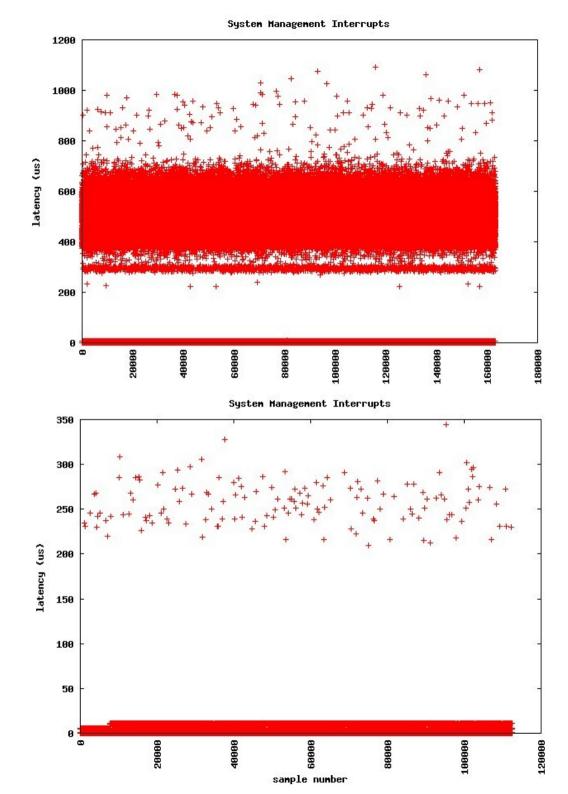
Use TUNA for Tuning

| Image: Second | 80 | 🔞 Tuna (on perf20.lab.bos.redhat.com) 🔤 🕂 🗙 | | | | | | | | | | | | | |
|---|--|---|---------|----------|--------|---------|----------|----------|------|-----------|-------------|---------------|---------|--|----------|
| Image: Description of the second s | Socke | 1 1 | | 4 | | | | IRQ | PID | Policy | Priority | Affinity | Events | Users | |
| 1 0 2 132 132 110 22 132 110 | | CPU | Usage | | | | | 17 | 1473 | FIFO | 50 | 1,13 | 51525 | megasas | |
| 2 0 23 127 FFO 50 2,14 30 enc_ncd.usb1 23 12 0 15 0 23 629 FFO 50 46098 eth.1eth.000 21 12 0 15 0 230 6320 FFO 50 0.23 1a eth.0sc 21 14 0 17 0 221 6148 FFO 50 0.23 1a eth.0sc 2233 6146 FFO 50 13 56938 eth.0v18-Rx 2233 6145 FFO 50 12 55448 eth.0v18-Rx 2235 6145 FFO 50 12 55406 eth.0v18-Rx 2235 6144 FFO 50 14 5603 eth.0v18-Rx 2235 6144 FFO 50 14 58014 eth.0v10-Rx 2235 6144 FFO 50 14 58014 eth.0v10-Rx 2235 6143 FFO 50 14 58014 eth.0v10-Rx 2240 6139 FFO 50 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>22</td> <td>1321</td> <td>FIFO</td> <td>50</td> <td>1,13</td> <td>858</td> <td>uhci_hcd:usb2,uhci_hcd:usb3,uhci_hcd:usb4,uhci_h</td> <td>i </td> | | | | | | | | 22 | 1321 | FIFO | 50 | 1,13 | 858 | uhci_hcd:usb2,uhci_hcd:usb3,uhci_hcd:usb4,uhci_h | i |
| 12 0 2 0 50 0 46038 eth3(e1000) 2 13 0 16 0 223 6320 FiFO 50 0 -23 1 eth0.452(e1000) 2 14 0 17 0 223 6148 FiFO 50 13 56938 eth0.452(e1000) 2 223 6147 FiFO 50 13 56938 eth0.413-Rx 2 233 6146 FiFO 50 14 56700 eth0.413-Rx 2 23 6144 FiFO 50 14 56700 eth0.413-Rx 2 10 0 223 6144 FiFO 50 14 56700 eth0.413-Rx 2 23 6144 FiFO 50 14 58816 eth0.4010-Rx 2 23 0 223 6140 FiFO 50 14 58816 eth0.401-Rx 2 23 0 223 0 13 16407 10 10 10 | | | | | | | | 23 | 1270 | FIFO | 50 | 2,14 | 30 | ehci_hcd:usb1 | |
| I 13 I 16 I IIC 200 Socker 1 IIC 200 Socker 2 Socker 2 Socker 3 Filter (PU Usage IIC 200 | | | | | | | | 2229 | 6529 | FIFO | 50 | 0 | 46098 | eth3(e1000) | |
| 2 14 0 17 0 50 50 0.23 1 eth0:sic socket 2 socket 3 socke 3 | | | | | | | | 2230 | 6320 | FIFO | 50 | 13 | 1624017 | eth2(e1000) | |
| 2/2/2 0.147 FiFO 5/0 1.3 59338 eth0/15-KX Socket 2 Socket 3 2/232 614/2 FiFO 5/0 2 55448 eth0/15-KX Socket 2 Filter CPU Usage Filter CPU Usage 5/14/2 FiFO 5/0 1/2 55448 eth0/15-KX 2/23 614/3 FiFO 5/0 1/2 55448 eth0/13-RX 2/23 6/14/2 FiFO 5/0 1/2 55406 eth0/11-RX 2/23 614/3 FiFO 5/0 1/4 56700 eth0/11-RX 2/23 6/14/2 FiFO 5/0 1/4 56014 eth0/11-RX 2/23 614/3 FiFO 5/0 1/4 58014 eth0/10-RX 2/23 6/14/2 FiFO 5/0 6/0573 eth0/10/2-RX 2 | | | | | | | | 2231 | 6148 | FIFO | 50 | 0-23 | 1 | eth0:lsc | |
| Filter CPU Usage Filter So 14 S600 etho:V1-Rx Usage Usage Filter So 14 S6014 etho:V0-Rx Usage Usage So 14 S6014 etho:V0-Rx Usage Usage So 14 S6014 etho:V0-Rx Usage | | 14 | 0 | | 17 | 0 | | 2232 | 6147 | FIFO | 50 | 13 | 56938 | eth0:v15-Rx | |
| Image: constraint of the constraint | Socke | et 2 | , | Sock | et 3 - | , | | 2233 | 6146 | FIFO | 50 | 2 | 55448 | eth0:v14-Rx | |
| Image: state of the state | Filter | CPU | Usage | Filter | CPU | Usage | | 2234 | 6145 | FIFO | 50 | 12 | 55406 | eth0:v13-Rx | |
| Image: Second | | 6 | 0 | | 9 | 0 | | 2235 | 6144 | FIFO | 50 | 14 | 56700 | eth0:v12-Rx | |
| Image: Second | | 7 | 0 | | 10 | 0 | | 2236 | 6143 | FIFO | 50 | 1 | 56803 | eth0:v11-Rx | |
| □ 19 0 □ 22 0 223 6140 FIFO 50 14 58816 eth0:v8-Rx □ 20 0 0 23 0 10 6139 FIFO 50 0 60573 eth0:v7-Rx • PID Policy Priority Affnity VolC±tSwitch NonVolCtxtSwitch Command Line •< | | 8 | 0 | | 11 | 0 | | 2237 | 6142 | FIFO | 50 | 14 | 58014 | eth0:v10-Rx | |
| Image: Second | | 18 | 0 | | 21 | 0 | | 2238 | 6141 | FIFO | 50 | 1 | 57371 | eth0:v9-Rx | |
| PID Policy Priority Affinity VolCxtswitch NonVolCxtswitch Command Line 1 OTHER 0 0-23 20259 2744 init [3] 2 OTHER 0 0-23 530 1320 kthreadd 3 FIFO 99 0 702 0 migration/0 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-block/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 5677 0 sirq-tasklet/0 | | 19 | 0 | | 22 | 0 | | 2239 | 6140 | FIFO | 50 | 14 | 58816 | eth0:v8-Rx | |
| Pilo Policy Priority Affinity VolCtxtSwitch NonVolCtxtSwitch Command Line 1 OTHER 0 0-23 20259 2744 init [3] 2 OTHER 0 0-23 530 1320 kthreadd 3 FIFO 99 0 702 0 migration/0 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 15 0 sirq-high/0 7 FIFO 50 0 133467 0 sirq-net-tx/0 8 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 sirq-tasklet/0 sirq-tasklet/0 | | 20 | 0 | | 23 | 0 | | 2240 | 6139 | FIFO | 50 | 0 | 60573 | eth0:v7-Rx | - |
| Policy Priority Attinity VolCxtswitch NonvolCxtswitch Command Line 1 OTHER 0 0-23 20259 2744 init [3] 2 OTHER 0 0-23 530 1320 kthreadd 3 FIFO 99 0 702 0 migration/0 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-block/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 5677 o sirq-tasklet/0 */ */ */ */ */ */ */ | | | | | | | | 4 | | | | | | • • • • • • • • • • • • • • • • • • • | 1 |
| 2 OTHER 0 0-23 530 1320 kthreadd 3 FIFO 99 0 702 0 migration/0 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 90298186 0 sirq-net-tx/0 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-block/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 0 sirq-block/0 *********************************** | PID | Po | licy | Priority | / At | ffinity | VolC | txtSwitc | h N | onVolCtxt | Switch | Command L | .ine | | ^ |
| 3 FIFO 99 0 702 0 migration/0 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 90298186 0 sirq-net-tx/0 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-block/0 9 FIFO 50 0 1055 0 sirq-tasklet/0 10 FIFO 50 0 567 0 sirq-tasklet/0 Tot wer-rh.pdf | 1 | от | THER | 0 | 0 | -23 | 202 | 59 | 27 | 744 | | init [3] | | | |
| 4 FIFO 99 0 2 0 posixcputmr/0 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 90298186 0 sirq-timer/0 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-net-rx/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 5677 0 sirq-tasklet/0 | 2 | OTHER | | 0 | | 23 | 530 | | 1320 | | | kthreadd | | | |
| 5 FIFO 50 0 2 0 sirq-high/0 6 FIFO 50 0 90298186 0 sirq-timer/0 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-net-rx/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 sirq-tasklet/0 sirq-tasklet/0 | 3 | FIF | FIFO | | 99 0 | | 702 | 0 | | | migration/0 | | | | |
| 6 FIFO 50 0 90298186 0 sirq-timer/0 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-net-rx/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 o sirq-tasklet/0 | 4 | FIFO | | 99 0 | | | 2 | | 0 | | | posixcputmr/0 | | | |
| 7 FIFO 50 0 15 0 sirq-net-tx/0 8 FIFO 50 0 133467 0 sirq-net-rx/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 0 sirq-tasklet/0 | 5 | FIFO | | 50 0 | | | 2 | 0 | | | sirq-high/0 | | | | |
| 8 FIFO 50 0 133467 0 sirq-net-rx/0 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 0 sirq-tasklet/0 Image: Comparison of the compar | 6 | FIF | FIFO 5 | | 0 0 | | 90298186 | | 0 | | | sirq-timer/0 | | | |
| 9 FIFO 50 0 1055 0 sirq-block/0 10 FIFO 50 0 567 0 sirq-tasklet/0 Image: Comparison of the sirq tasklet/0 Image: Comparison of the sirq task of the sirq t | 7 | FIFO | | 50 0 | | | 15 | 15 | | 0 | | sirq-net-tx/0 | | | |
| 10 FIFO 50 0 567 0 sirq-tasklet/0 Image: Constraint of the second sec | 8 | FIFO | | 50 0 | | | 133 | 133467 | | 0 | | • | | | |
| Image: Second sec | 9 | | | | | | 1055 | 055 | | | | | | | |
| [qpid-producer-rh.pdf] [root@perf20:~ Tuna (on perf20.lab | 10 | FIF | 0 | 50 | 0 | | 567 | | 0 | | | sirq-tasklet/ | 0 | | • |
| | 4 | | | | | | | | | | | | | | Ð |
| 子 Applications Places System 🚕 😔 📫 1 GHz 🗰 🖾 ^ 🛛 😨 🚯 📋 📶 root Tue Nov 4, 2:44 PM 🧔 | 13 : 📑 | [qpi | d-produ | icer-rh | .pdf] | 🔲 roo | ot@pe | erf20:~ | | 💽 Tu | na (on per | f20.lab | | | Ĩ |
| | 🕞 Applications Places System 🚕 🌑 📲 1 GHz | | | | | | | | | | | | | G) 2 | |



Hardware Matters

- Hardware can have a big effect on realtime performance
- Hardware drivers may need to be updated to handle threaded interrupts
- Many system BIOS's include Service Management Interrupts (SMIs)
 - Cause non-deterministic latency beneath the operation system by taking CPU cycles for things like power management, administration
 - SMI latencies *cannot* be resolved by realtime linux—they require the hardware OEM to remove SMIs or make them configurable





History of Realtime Linux

- First steps (2000-2004)
 - Ingo Molnar / Andrew Morton low latency patch
 - Robert Love preemption patch

Current State (2004 - today)

- First started on 2.6.9 kernel Ingo Molnar's realtime patch
- Originally called realtime-preempt patch
- Moving From -rt to mainline:
 - BKL preemptable (2.6.8)
 - Mutex patch (2.6.16)
 - Semaphore-to-Mutex conversion (ongoing ~85% done)
 - Hrtimers subsystem (2.6.16)
 - Robust futexes (2.6.17)

- Priority inheritance futexes (PIfutex) (2.6.18)
- Generic IRQ layer (2.6.18)
- Core time re-write (2.6.18)
- Sleepable RCU (2.6.19)
- Latency Tracer (circa 2.6.18)
- High-res+dynticks (2.6.21)
- CFS completely fair scheduler (2.6.23)
- Conversion of spin-locks to mutex (2.6.23+)
- All Interrupt handling in threads (~2.6.23+)
- Full rt-preempt (~2.6.24+)



Realtime Linux Roadmap

Incorporate the realtime patchset into the mainline kernel

- Current realtime kernel is 2.6.26-based
- Convert from patchset to GIT tree for 2.6.28
- Merge threaded IRQs
 - Threaded device handler, allows driver to register as threaded interrupt handler
 - Target for 2.6.30
- Merge preemption
 - Re-work 'macro magic' that implements preemption into lock_t abstraction
 - Depends heavily on acceptance of threaded interrupts
 - Target perhaps for 2.6.31/2.6.32

Improve performance

- Reduce determistic latency vs throughput tradeoff
- Improve performance of surrounding IO systems

Red Hat Enterprise MRG Provides Realtime Linux

- Red Hat Enterprise MRG (Messaging, Realtime, Grid) includes a Realtime kernel and performance tools
- Installs onto standard Red Hat Enterprise Linux 5 and preserves Red Hat Enterprise Linux application certifications
 - No application or code changes necessary
 - Take advantage of Red Hat ecosystem
- Aggressively tracks upstream kernel development for performance

- Red Hat has worked with OEMs to certify realtime hardware
 - Including addressing SMIs
- Red Hat has partnered with IBM and Sun to certify their realtime JVMs for MRG Realtime



Red Hat Enterprise MRG Demo



Additional Information

- RT Wiki: http://rt.wiki.kernel.org/index.php/Main_Page
- Red Hat Enterprise MRG: http://redhat.com/mrg
- Realtime Tuning Guide: http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_MRG/1.1/html/Realtime_Tuning_Guide/

