The realtime preemption patch (PREEMPT_RT)

Concepts and mainline integration

Thomas Gleixner

RealTime Linux WorkShop 2006

Goals

- Fully preemptive kernel
- Realtime guarantees suitable for the vast majority of applications
- POSIX compliance (single API)

Components

- Priority inheritance
- High resolution timers
- Threaded interrupt handlers
- Sleeping spinlocks

History

- Autumn 2004 MontaVista, Timesys, Lynuxworks post realtime related patch fragments
- Ingo Molnar reimplements parts from scratch and posts the realtime preemption patch
- A core team forms
- Kernel Summit 2006 accepts a plan to merge all components into mainline over time

Priority Inheritance

- Avoid unbounded latencies caused by priority inversion
- New concurrency control primitive “rt-mutex”
- Merged into mainline in 2.6.18 six month after Linus stated, that priority inheritance is evil and never will be merged into mainline
- glibc support available

High resolution timers

- hrtimer base infrastructure
- Generic time of day framework
- clockevents framework
- high resolution timer implementation with support of dynamic ticks

Threaded Interrupt Handlers

- Run interrupt handling code in thread context
- short low level handlers to wake up the handler thread
- handlers in thread context are preemptible
- can integrate softirq (bottom half) handlers into the top half handler
- depends on the enhanced generic interrupt handling framework

Sleeping spinlocks

- extend rt-mutex functionality to match spinlock semantics
- priority inheritance functionality of rt-mutex
- spinlock protected regions become preemptible
- raw_spinlock primitive to protect critical data structures (only used in audited core code pathes)

Full preemptible kernel

- Enable sleeping spinlocks
- Use the rtmutex primitive for all kinds of in kernel locks
- Drastically reduce interrupt off regions
- Tweaks VM and scheduler core code
- Dynamic timer softirq priorities
-

Debugging facilities

- latency tracer
- lock dependency validator
- timer statistics

Mainline improvements

- about 1200 cleanup and bugfix patches related to race conditions and locking problems
- enhanced debugging facilities
- more awareness about locking semantics and constraints

Performance

- User space latencies <50us from hardware interrupt to user space handler execution. Measured on a 800MHZ PIII system in a 4 weeks 24/7 test.

Real world usage

- Laser welding control
- Woodworking machines
- Motion controllers
- Soft PLCs
- Enterprise Realtime (Distributed Realtime Java)

Merge timeline

- Oct 2004: (2.6.9) first preempt-rt patch
- Jan 2006: (2.6.16) hrtimer base infrastructure
- April 2006: (2.6.17) robust futexes
- July 2007: (2.6.18) pi-futexes, generic interrupt layer (ARM, PPC), lock dependency validator
- October 2007: (2.6.19) generic interrupt layer (i386, x86_64, MSI)

Merge timeline plan

- 2.6.20: high resolution timer / dynamic ticks
- 2.6.21: threaded interrupt handlers
- 2.6.22: sleeping spinlocks core
- 2.6.23: realtime core code (i386, x64_86)
- 2.6.24: more architectures
- ....

Resources

- rt.wiki.kernel.org
- www.osadl.org - Live-CD
- LKML (developers)