Inside The RT Patch

Talk:
Steven Rostedt
(Red Hat)

Benchmarks:
Darren V Hart
(IBM)
Inside The RT Patch

Talk:
Steven Rostedt
(Red Hat)

Benchmarks:
Darren V Hart
(IBM)
Understanding PREEMPT_RT

Talk:
Steven Rostedt
(Red Hat)

Benchmarks:
Darren V Hart
(IBM)
Understanding PREEMPT_RT

Talk:
Steven Rostedt
(Red Hat)

Benchmarks:
Darren V Hart
(Intel)
Understanding PREEMPT_RT

Talk:
Steven Rostedt
(Red Hat)

Benchmarks:
Darren V Hart
(Intel)
ELC-EU

So what should I talk about?
So what should I talk about?
Trebuchet

Wikimedia Commons
Trebuchet

Wikimedia Commons
Trebuchet
Trebuchet

$L = \text{Kinetic Energy} - \text{Potential Energy}$

\[
L = \frac{1}{2} m \left( v_1^2 + v_2^2 \right) + \frac{1}{2} I \left( \dot{\theta}_1^2 + \dot{\theta}_2^2 \right) - mg (y_1 + y_2)
\]

\[
= \frac{1}{2} m \left( x_1^2 + y_1^2 + x_2^2 + y_2^2 \right) + \frac{1}{2} I \left( \dot{\theta}_1^2 + \dot{\theta}_2^2 \right) - mg (y_1 + y_2)
\]

\[
L = \frac{1}{6} m l^2 \left[ \dot{\theta}_2^2 + 4 \dot{\theta}_1^2 + 3 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \right] + \frac{1}{2} m g l \left( 3 \cos \theta_1 + \cos \theta_2 \right).
\]

\[
\dot{\theta}_1 = \frac{6}{m l^2} \frac{2 p_{\theta_1} - 3 \cos(\theta_1 - \theta_2) p_{\theta_2}}{16 - 9 \cos^2(\theta_1 - \theta_2)}
\]

\[
p_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = \frac{1}{6} m l^2 \left[ 8 \dot{\theta}_1 + 3 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \right]
\]

\[
\dot{\theta}_2 = \frac{6}{m l^2} \frac{8 p_{\theta_2} - 3 \cos(\theta_1 - \theta_2) p_{\theta_1}}{16 - 9 \cos^2(\theta_1 - \theta_2)}.
\]

\[
p_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = \frac{1}{6} m l^2 \left[ 2 \dot{\theta}_2 + 3 \dot{\theta}_1 \cos(\theta_1 - \theta_2) \right].
\]

\[
\dot{p}_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = -\frac{1}{2} m l^2 \left[ \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + 3 \frac{g}{l} \sin \theta_1 \right]
\]

\[
\dot{p}_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = -\frac{1}{2} m l^2 \left[ -\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + \frac{g}{l} \sin \theta_2 \right].
\]
\[ L = \text{Kinetic Energy} - \text{Potential Energy} \]
\[ = \frac{1}{2} m (v_1^2 + v_2^2) + \frac{1}{2} I \left( \dot{\theta}_1^2 + \dot{\theta}_2^2 \right) - mg (y_1 + y_2) \]
\[ = \frac{1}{2} m \left( \dot{x}_1^2 + \dot{y}_1^2 + \dot{x}_2^2 + \dot{y}_2^2 \right) + \frac{1}{2} I \left( \dot{\theta}_1^2 + \dot{\theta}_2^2 \right) - mg (y_1 + y_2) \]

\[ L = \frac{1}{6} m \ell^2 \left[ \dot{\theta}_2^2 + 4 \dot{\theta}_1^2 + 3 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \right] + \frac{1}{2} m g \ell \left( 3 \cos \theta_1 + \cos \theta_2 \right). \]

\[ \dot{\theta}_1 = \frac{6}{m \ell^2} \frac{2 p_{\theta_1} - 3 \cos(\theta_1 - \theta_2) p_{\theta_2}}{16 - 9 \cos^2(\theta_1 - \theta_2)} \]

\[ p_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = \frac{1}{6} m \ell^2 \left[ 8 \dot{\theta}_1 + 3 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \right] \]

\[ \dot{\theta}_2 = \frac{6}{m \ell^2} \frac{8 p_{\theta_2} - 3 \cos(\theta_1 - \theta_2) p_{\theta_1}}{16 - 9 \cos^2(\theta_1 - \theta_2)} \]

\[ p_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = \frac{1}{6} m \ell^2 \left[ 2 \dot{\theta}_2 + 3 \dot{\theta}_1 \cos(\theta_1 - \theta_2) \right]. \]

\[ p_{\theta_1} = \frac{\partial L}{\partial \theta_1} = -\frac{1}{2} m \ell^2 \left[ \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + 3 \frac{g}{\ell} \sin \theta_1 \right] \]

\[ \dot{p}_{\theta_2} = \frac{\partial L}{\partial \theta_2} = -\frac{1}{2} m \ell^2 \left[ -\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + \frac{g}{\ell} \sin \theta_2 \right]. \]
Where to get the RT patch

- **Stable Repository**

- **Patches**

- **Wiki**
What is a Real-time OS?

• Deterministic
  - Does what you expect to do
  - When you expect it will do it

• Does not mean fast
  - Would be nice to have throughput
  - Guarantying determinism adds overhead
  - Provides fast “worst case” times

• Can meet your deadlines
  - If you have done your homework
What is a Real-time OS?

- Dependent on the system
  - SMI
  - Cache
  - Bus contention
- hwlat detector
  - New enhancements coming
The Goal of PREEMPT_RT

• 100% Preemptible kernel
  – Not actually possible, but let’s try regardless
  – Remove disabling of interrupts
  – Removal of disabling other forms of preemption

• Quick reaction times!
  – bring latencies down to a minimum
Menuconfig

Preemption Model
Use the arrow keys to navigate this window or press the hotkey of the item you wish to select followed by the <SPACE BAR>. Press <?> for additional information about this option.

( ) No Forced Preemption (Server)
( ) Voluntary Kernel Preemption (Desktop)
( ) Preemptible Kernel (Low-Latency Desktop)
( ) Preemptible Kernel (Basic RT)
(X) Fully Preemptible Kernel (RT)
No Preemption

- Server
  - Do as most possible with as little scheduling overhead
- Never schedule unless a function explicitly calls schedule()
- Long latency system calls.
- Back in the days of 2.4 and before.
Voluntary Preemption

- `might_sleep();`
  - Calls `might_resched();` calls `_cond_resched()`
  - Used as a debugging aid to catch functions that might schedule called from atomic operations.
  - `need_resched` – why not schedule?
  - Schedule only at “preemption points”. 
Preemptible Kernel

- Robert Love's CONFIG_PREEMPT
- SMP machines must protect the same critical sections as a preemptible kernel.
- Preempt anywhere except within spin_locks and some minor other areas (preempt_disable).
- Every spin_lock acts like a single “global lock” WRT preemption.
Preemptible Kernel (Basic RT)

- Mostly to help out debugging PREEMPT_RT_FULL
- Enables parts of the PREEMPT_RT options, without sleeping spin_locks
- Don't worry about it (It will probably go away)
Fully Preemptible Kernel
The RT Patch

- PREEMPT_RT_FULL
- Preempt everywhere! (except from preempt_disable and interrupts disabled).
- spin_locks are now mutexes.
- Interrupts as threads
  - interrupt handlers can schedule
- Priority inheritance inside the kernel (not just for user mutexes)
Sleeping spin_lock

- CONFIG_PREEMPT is a global lock (like the BKL but for the CPU)
- sleeping spin_locks contains critical sections that are localized to tasks
- Must have threaded interrupts
- Must not be in atomic paths (preempt_disable or local_irq_save)
- Uses priority inheritance
  - Not just for futexes
PREEMPT_LAZY

- RT can preempt almost anywhere
- Spinlocks that are now mutexes can be preempted
  - Much more likely to cause contention
- Do not preempt on migrate_disable()
  - used by sleepable spinlocks
- Increases throughput on non-RT tasks
Priority Inheritance

- Prevents unbounded priority inversion
  - Can't stop bounded priority inversion
- Is a bit complex
  - One owner per lock
  - Why we hate rwlocks
    - will explain more later
Unbounded Priority Inversion

A

B

C

blocked

preempted

preempted
Priority Inheritance

A

- blocked
- wakes up
- preempted
- releases lock

B

- sleeps
- wakes up

C

- blocked

- releases lock
raw_spin_lock

- Some spin_locks should never be converted to a mutex
- Same as current mainline spin_locks
- Should only be used for scheduler, rtmutex implementation, debugging/tracing infrastructure and for timer interrupts.
- Timer drivers for clock events (HPET, PM timer, TSC)
- Exists today in current mainline, with no other purpose as to annotate what locks are special (Thank you Linus!)
Threaded Interrupts

- Lowers Interrupt Latency
- Prioritize interrupts even when the hardware does not support it.
- Less noise from things like “updatedb”
Interrupt Latency

Task

interrupt

device handler
Interrupt Thread

Task

interrupt

wake up device thread

device handler

sleep
Non-Thread IRQs

- Timer interrupt
  - Manages the system (sends signals to others about time management)
- IRQF_TIMER
  - Denotes that an interrupt handler is a timer
- IRQF_NO_THREAD
  - When the interrupt must not be a thread
  - Don't use unless you know what you are doing
  - Must not call spin_locks
Threaded Interrupts

- Now in mainline
  - Per device interrupts
  - One big switch (all irqs as threads)
- Per device is still preferred
  - except for non shared interrupts
  - Shared devices can have different priorities
- One big switch
  - Handlers the same, but just threaded
Threaded Interrupts

- request_threaded_irq()
  - Tells system driver wants handler as thread

- Driver registers two functions
  - handler
    - If NULL must have thread_fn
      - Disables irq lin
      - handler assigned by system
    - non-NULL is called by hard irq
  - thread_fn (optional)
    - When set makes irq threaded
    - non-NULL to disable device only
Threaded Interrupts

- The kernel command line parameter
  - threadirqs
- threadirqs forces all IRQS to have a “special” handler and uses the handler as thread_fn
  - except IRQF_NOTHREAD, IRQF_PER_CPU and IRQF_ONESHOT
local_irq_disable

- EVIL!!!
- This includes local_irq_save
- No inclination to what it's protecting
- SMP unsafe
- High latency
spin_lock_irqsave

- The Angel
- PREEMP_RT does **NOT** disable interrupts
  - Remember, in PREEMPT_RT spin_locks are really mutexes
  - low latency
- Tight coupling between critical sections and disabling interrupts
- Gives a hint to what it's protecting
  - (spin_lock name)
preempt_disable

- local_irq_disable's younger sibling
- Also does not give a hint to what it protects
- preempt_enable_no_resched
  - only should be used within preempt_disabled locations
  - __preempt_enable_no_resched
    - Only use before directly calling schedule()
per_cpu

- Avoid using:
  - local_irq_save
  - preempt_disable
  - get_cpu_var (well, you can, but be nice – it calls preempt_disable)

- Do:
  - pinned CPU threads
  - get_cpu_light()
  - get_local_var(var)
  - local_lock[_irq[save]](var)
get_cpu_light()

- Non PREEMPT_RT is same as get_cpu()
- On PREEMPT_RT disables migration
get_local_var(var)

- Non PREEMPT_RT is same as get_cpu_var(var)
- On PREEMPT_RT disabled migration
local_lock[_irq[save]](var)

- Non PREEMPT_RT is just preempt_disable()
- On PREEMPT_RT grabs a lock based on var
  - disables migration
- Use local_unlock[_irq[restore]](var)
- Labels what you are protecting
rwlocks

- Death of Determinism
- Writes must wait for unknown amount of readers
- Recursive locking
- Possible strange deadlock due to writers
  - Yes, affects mainline too!
NOHZ

- idle nohz best for power management
- Not nice for responses from idle
- Process nohz coming soon (nothing to do with idle nohz, but uses same ideas and in some cases, same code)
Real-Time User Space

• Don't use priority 99
• Don't implement spin locks
  – Use priority inheritance futexes
  – PTHREAD_PRIO_INHERIT
• Avoid slow I/O
• mmap passing data
• mlock_all()
  – at least the stuff you know you need
Questions?