An Overview of Realtime Linux
Clark Williams
Senior Architect, Red Hat, Inc.
Realtime Linux

Who, What, When, Where and Why?
(and How)
Realtime Linux

- A Modified Linux kernel that targets
  - Fast response to events
  - Consistent response times
  - Highly precise timers
- C library Interfaces to the kernel
  - POSIX threads
  - System calls for scheduler manipulations
- Measurement Tools
- Tuning Tools
Who does this stuff?

- Ingo Molnar (Red Hat)
- Thomas Gleixner (Red Hat contractor)
- Steven Rostedt (Red Hat)
- Paul McKenney (IBM)
- John Stultz (IBM)
- Gregory Haskins (Novell)
- Peter Zijlstra (Red Hat)
- many others...
What does it do?
What does it do? (continued)

- The next slide shows data from a program named cyclicTest
  - cyclicTest measures the delta from when it's scheduled to wake up from when it actually does wake up.
  - Test runs on two kernels:
    - vanilla 2.6.24.7 kernel
    - RT kernel based on the same source
  - 500,000 loops
  - hackbench load running in the background
What does it do? (continued)
What does that mean statistically?

Vanilla (500K samples)
- Min: 1
- Max: 2857
- Mean: 11.47
- Mode: 9.00
- Median: 9.00
- Std. Deviation: 54.94

MRG RT (500K samples)
- Min: 4
- Max: 43
- Mean: 8.34
- Mode: 8.00
- Median: 8.00
- Std. Deviation: 1.49
**What** changed in the RT Kernel?

- **Preemption**
  - Most locks converted to rt_mutex
  - priority inheritance
  - threaded interrupt handlers
  - 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
What changed in the C library?

- `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 RT kernels implement priority inheritance (PI) in *futexes* (fast user-space mutexes) used by pthreads
- POSIX interfaces to scheduler APIs
  - `sched_*`
- Timer interfaces
- Note that you don't have to have an RT kernel for most of these APIs to work
What Measurement tools Are available?

- rt-tests from Thomas Gleixner
  - cyclicetest
  - signaltest
  - pi_stress
- LTP realtime tests
  - sched_latency
  - sched_football
  - pi-tests
**What** Tuning tools are available?

- **ftrace**
  - rt kernel built-in mechanism to trace events
- **oprofile**
  - system-wide sampling profiler
- **systemtap**
  - custom data probes for kernel space
- **tuna**
  - isolate processor cores
  - adjust thread priorities
  - change interrupt affinity
  - save/restore tunings
**When** did this happen?

- **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
  - Latest is 2.6.25.4-rt4
    - HRT, RCU and CFS merged upstream
    - new latency tracer (ftrace)
**When** does RT help?

When you need fast, consistent response times

The following slide shows the difference between running the QPID messaging application on a stock Linux kernel and an RT Linux kernel

- Y-axis is messages-per-second
- X-axis is sample point (55 samples)
When does RT help? (continued)
*When* does RT help? (continued)

- Note that tuned RHEL5.2 kernel actually processed more messages than tuned RT kernel
- Note also that the tuned RT kernel was MUCH more consistent in its processing times
  - Standard Deviation of messages/sec on RT kernel is much smaller than on either of the RHEL runs
  - This consistency is the biggest strength of RT
**Where is it used?**

- Financial systems
  - Market data applications
  - Analysis applications
  - Message-passing platforms
- Command, Control and Communications systems
  - Shipboard data systems interfaces
- Any application that cares about responding promptly to events
Why the drastic kernel changes?

- Running with interrupts off for the majority of times allows preemption (i.e. quicker reschedule to high priority threads)
- Each interrupt event is a potential reschedule point
- If you can be interrupted while holding a lock, you need to be able to come back to that context, so need to be in a thread
- Threaded interrupts allow control of interrupt processing order via thread priorities
**Why** should I care?

- You should care if you need to meet deadlines in your application(s)
- Realtime Linux is an effort to make the Linux kernel **deterministically responsive** to events
  - The amount of time which elapses from event occurrence to event handling is bounded.
- Bounded by *what*?
  - dependent on the hardware you're running
  - Given the same load, a 3 GHz processor will respond to an event faster than a 700 Mhz processor
  - Better to say that we're trying to make the response time more consistent (i.e. reduce latency standard deviation)
**How** can I use it?

- **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
<table>
<thead>
<tr>
<th>Filter</th>
<th>CPU</th>
<th>Usage</th>
<th>IRQ</th>
<th>PID</th>
<th>Policy</th>
<th>Priority</th>
<th>Affinity</th>
<th>Events</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0.1</td>
<td>809</td>
<td>timer</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>391</td>
<td>FIFO</td>
<td>0.1</td>
<td>8</td>
<td>18042</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>3038</td>
<td>FIFO</td>
<td>0.1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>409</td>
<td>FIFO</td>
<td>0.1</td>
<td>465</td>
<td>serial</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>1751</td>
<td>FIFO</td>
<td>0.1</td>
<td>3</td>
<td>floppy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PID</th>
<th>Policy</th>
<th>Priority</th>
<th>Affinity</th>
<th>VoICtxSwitch</th>
<th>NonVoICtxSwitch</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OTHER</td>
<td>0</td>
<td>0.1</td>
<td>19612</td>
<td>1609</td>
<td>int[3]</td>
</tr>
<tr>
<td>2</td>
<td>OTHER</td>
<td>0</td>
<td>0.1</td>
<td>174</td>
<td>210</td>
<td>kthreadd</td>
</tr>
<tr>
<td>3</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>507</td>
<td>0</td>
<td>migration/0</td>
</tr>
<tr>
<td>4</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>2</td>
<td>0</td>
<td>posix_cpu_timer</td>
</tr>
<tr>
<td>5</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>2</td>
<td>0</td>
<td>sirq-high/0</td>
</tr>
<tr>
<td>6</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>93509037</td>
<td>0</td>
<td>sirq-timer/0</td>
</tr>
<tr>
<td>7</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>400075</td>
<td>1</td>
<td>sirq-net-tx/0</td>
</tr>
<tr>
<td>8</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>65038</td>
<td>0</td>
<td>sirq-block/0</td>
</tr>
<tr>
<td>9</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>338</td>
<td>0</td>
<td>sirq-tasklet/0</td>
</tr>
<tr>
<td>10</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>22278917</td>
<td>0</td>
<td>sirq-sched/0</td>
</tr>
<tr>
<td>11</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>2058939</td>
<td>0</td>
<td>sirq-hc/0</td>
</tr>
<tr>
<td>12</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>65</td>
<td>0</td>
<td>sirq-hc/0</td>
</tr>
<tr>
<td>13</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>20007</td>
<td>235</td>
<td>desched/0</td>
</tr>
<tr>
<td>14</td>
<td>FIFO</td>
<td>0</td>
<td>0.1</td>
<td>431</td>
<td>0</td>
<td>migration/1</td>
</tr>
<tr>
<td>15</td>
<td>OTHER</td>
<td>0</td>
<td>0.1</td>
<td>40007</td>
<td>235</td>
<td>desched/0</td>
</tr>
<tr>
<td>16</td>
<td>FIFO</td>
<td>99</td>
<td>1</td>
<td>431</td>
<td>0</td>
<td>migration/1</td>
</tr>
<tr>
<td>17</td>
<td>FIFO</td>
<td>99</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>posix_cpu_timer</td>
</tr>
<tr>
<td>18</td>
<td>FIFO</td>
<td>99</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>sirq-high/1</td>
</tr>
<tr>
<td>19</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>93512225</td>
<td>0</td>
<td>sirq-timer/1</td>
</tr>
<tr>
<td>20</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>sirq-net-tx/1</td>
</tr>
<tr>
<td>21</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>39260</td>
<td>0</td>
<td>sirq-net-tx/1</td>
</tr>
<tr>
<td>22</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>38679</td>
<td>1</td>
<td>sirq-block/1</td>
</tr>
<tr>
<td>23</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>113</td>
<td>0</td>
<td>sirq-tasklet/1</td>
</tr>
<tr>
<td>24</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>22511725</td>
<td>0</td>
<td>sirq-sched/1</td>
</tr>
<tr>
<td>25</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>3120</td>
<td>0</td>
<td>sirq-hc/0</td>
</tr>
<tr>
<td>26</td>
<td>FIFO</td>
<td>50</td>
<td>1</td>
<td>2027617</td>
<td>0</td>
<td>sirq-hc/0</td>
</tr>
<tr>
<td>27</td>
<td>FIFO</td>
<td>99</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>watchdog/0</td>
</tr>
<tr>
<td>28</td>
<td>OTHER</td>
<td>0</td>
<td>0.1</td>
<td>33935</td>
<td>229</td>
<td>desched/1</td>
</tr>
</tbody>
</table>
How can I use it? (continued)

- 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 you high priority threads don't hog the processor
How can I get it?

- Red Hat Messaging, Realtime, Grid (MRG)
  - Separate product
  - Layered on Red Hat Enterprise Linux 5.2
  - Kernel is 2.6.24.7-rt11 based
  - Supports i686 and x86_64 architectures only
  - Comes with tuna and other support packages
  - No application changes required for RHEL5 applications
    - No recompiles needed
Questions?

Clark Williams
williams@redhat.com