Realtime BoF Session

RealTime Testing Best Practice of RealTime WG

YungJoon Jung
Contents

- Introduction
- Current RealTime Testing Methods
- Plan
Introduction
Introduction

- CELF RTWG has been collecting realtime testing methods
- What has been collected?
  - Test presentations and documents
  - Test programs
  - Test results
  - Benchmark programs
  - Stress programs and actions
  - Issues and Techniques
  - Etc
Introduction

- You can see our this works at:
- See http://elinux.org/Realtime_Testing_Best_Practices
Current RealTime Testing Methods
lpptest

- Included in the RT-preempt patch
- Use the driver in the linux kernel, to toggle a bit on the parallel port, and watch for a response toggle back
- Rt-preempt patch applied
  - drivers/char/lpptest.c
  - scripts/testlpp.c
- Update of lpptest
  - Removed dependency on TSC
- This requires a separate machine to send the signal on the parallel port and receive the response

April 15th, 2008
CE Linux Forum
RealFeel (1/2)

- Simple test program that how well a periodic interrupt is processed
- This program let periodic interrupt(/dev/rtc) have a fixed interval
- The program measures the time duration from interrupt to interrupt, and compares this to the expected value for the duration
- Depends on TSC in user space for timestamps
- See [http://brain.mcmaster.ca/~hahn/realfeel.c](http://brain.mcmaster.ca/~hahn/realfeel.c)
RealFeel (2/2)

- Result samples
RealFeel-ETRI

- Measures latency from interrupt occur to application, interrupt should wakeup, invocation
- Extends realfeel in several ways
  - Supports interrupt handler patch file for /dev/rtc
    - Send timestamp information to application through rtc parameter when interrupt occurs
  - Adds a histogram feature to dump the results to a histogram
  - Locks the process pages in memory
  - Changes the scheduling priority to SCHED_FIFO, at highest priority
- Still, depends on TSC.
  - But, we have ported to mainstone board(Xscale)
RealFeel-ETRI Results (1/2)

- Platform: Via EPIA (Nehemiah) 1GHz, 256Mbyte memory
- Kernel version: Linux 2.6.24.4
- Stress: ping (per 100 nano sec) from other machine, hackbench 20 (per 50sec)
- Test time: 10 hours

<table>
<thead>
<tr>
<th></th>
<th>Max latency time (usec)</th>
<th>Min latency time (usec)</th>
<th>Ave latency time (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>3888.57</td>
<td>3.96148</td>
<td>13.8279</td>
</tr>
<tr>
<td>Voluntary</td>
<td>3904.31</td>
<td>3.88947</td>
<td>11.2108</td>
</tr>
<tr>
<td>Preemptible</td>
<td>6792.74</td>
<td>4.75657</td>
<td>11.6637</td>
</tr>
<tr>
<td>Realtime-reempt</td>
<td>65.8249</td>
<td>9.36812</td>
<td>13.0913</td>
</tr>
</tbody>
</table>
RealFeel-ETRI Results (2/2)

- Platform: Mainston Board (Xscale) PXA270 520MHz, 64Mbyte Memory
- Kernel version: Linux 2.6.24.4
- Stress: ping (per 100 nano sec) from other machine, hackbench 10 (per 50sec)
- Test time: 10 hours

<table>
<thead>
<tr>
<th></th>
<th>Max latency time (usec)</th>
<th>Min latency time (usec)</th>
<th>Ave latency time (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td>7195.3</td>
<td>1.46731</td>
<td>223.65</td>
</tr>
<tr>
<td>Realtime-reempt</td>
<td>1451.32</td>
<td>52.1442</td>
<td>118.653</td>
</tr>
</tbody>
</table>
Implement a SM501 UART driver with UIO
Measure the schedule latency (when kernel handler invokes user handler)
The Result of an experiment with ULDD

- The effectiveness of PREEMPT_RT can be particularly observed when the scheduler policy for ULDD is set to SCHED_FIFO under high system load.
Cyclic test

- Wants to measure from periodic event to task wake up time (measuring worst case response time of real-time task)
  - Periodic event can be software timer or nanosleep
  - Creates a number of tasks with different priority
Cyclic test Results

- Platform: Pentium III 400MHz based PC (tglx’s reference machine)
- Kernel version: Linux 2.6.16
- Test case 1: \texttt{clock_nanosleep}, interval 10000 microseconds, 10000 loops, 100\% load
  - Commandline: \texttt{cyclictest --t 1 --p 80 --n --i 10000 --l 10000}

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Min (usec)</th>
<th>Max (usec)</th>
<th>Avg (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.16</td>
<td>55</td>
<td>4280</td>
<td>2198</td>
</tr>
<tr>
<td>2.6.16-hrt5</td>
<td>11</td>
<td>458</td>
<td>55</td>
</tr>
<tr>
<td>2.6.16-rt12</td>
<td>6</td>
<td>67</td>
<td>29</td>
</tr>
</tbody>
</table>
LRTB

- Benchmark framework for realtime responsiveness of Linux
  - See http://www.opsys.com/lrtbf/
  - For this test, 3 computers are needed (target, host and logger)
    - Target: test system
    - Host: control system and collects the data
    - Logger: cause to interrupts on the target and record the time it takes for the target to respond

- Latest results from opersysinc.
  - Kernel version: 2.6.12
  - Max_preempt_delay: 55us
  - Average_preempt_delay: 14us
Hourglass

- is a synthetic real-time application that can be used to learn how CPU scheduling in a general-purpose operating system works at microsecond and millisecond granularities
  - creates very detailed map of when each Hourglass thread has access to the CPU
  - supports multiple thread execution models; e.g. periodic and CPU-bound
  - acts as an abstraction layer for threading, timing, and CPU scheduling functionality on Unix- and Win32-based systems
Others

- **Woerner test**
  - received an interrupt on the serial port, and pushed data through several processes, before sending back out the serial port
  - requires an external machine for triggering the test and measuring the results.

- **Senoner test**
  - This is a latency test that simulates and audio workload
  - See [http://www.gardena.net/benno/linux/audio/](http://www.gardena.net/benno/linux/audio/)
## Test Program Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Ipptest</th>
<th>RealFeel-ETRI</th>
<th>Cyclicertest</th>
<th>LRTB</th>
<th>Woerner test</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform specific (for target)?</td>
<td>No. But requires a separated machine</td>
<td>Yes – i386. But ported to Xscale.</td>
<td>Probably no. hrt should be ported</td>
<td>No. but requires parallel port on target</td>
<td>No. but requires serial port on target</td>
</tr>
<tr>
<td>How to generate interrupt?</td>
<td>Data on parallel port</td>
<td>Periodic timer programmed i386 /dev/rtc</td>
<td>Clock and Timer(clock_nanosleep, POSIX interval)</td>
<td>Data on parallel port</td>
<td>Data on serial port</td>
</tr>
<tr>
<td>What does test measure?</td>
<td>End-to-end response latency</td>
<td>From interrupt handler to task invocation</td>
<td>Interrupt and scheduling latency</td>
<td>End-to-end response latency</td>
<td>End-to-end response latency</td>
</tr>
</tbody>
</table>

April 15th, 2008

CE Linux Forum
Stress Method

- Usual Stress programs
  - Find, du, ping, several benchmark programs (i.e. hackbench, lmbench, unixbench)
  - Ingo Molnar’s stress
    - Big file copy & hackbench & ping

- Stress actions
  - Things to spoil your RT performance test
    - have a bus-master device do a long DMA on the bus
    - get a page fault on your RT process (can be prevented with mlockall)
    - get multiple TLB flushes on your RT code path (how to cause this??)
    - get lots of instruction and data cache misses on your RT code path

April 15th, 2008
Test results summary
Platforms Tested and in Use with CONFIG_PREEMPT_RT


<table>
<thead>
<tr>
<th>Processor</th>
<th>Model #’s</th>
<th>Kernel version(s)</th>
<th>Contact(s)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opener</td>
<td>IBM LS-20, LS-21, x3455</td>
<td>2.6.16-r22, 2.6.20-r6</td>
<td>Theodore Ts'o, Darren Hart</td>
<td></td>
</tr>
<tr>
<td>Xeon</td>
<td>IBM 686B-P4U</td>
<td>2.6.20-r6, 2.6.21-ro6-ro6</td>
<td>Dave Sperry</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Compaq Evo N610</td>
<td>2.6.16-r22</td>
<td>Dave Sperry</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Celeron 1300</td>
<td>2.6.21.5-rt14</td>
<td>Oleksandr Nataienko</td>
<td></td>
</tr>
<tr>
<td>Prescott</td>
<td>PentiumD 3 GHz</td>
<td>2.6.22.5-rt9</td>
<td>Dmitriy Pisklov</td>
<td></td>
</tr>
<tr>
<td>Intel</td>
<td>Pentium 4 2.4 GHz</td>
<td>2.6.23.9-rt12</td>
<td>Jaswinder Singh</td>
<td></td>
</tr>
<tr>
<td>Intel</td>
<td>Pentium 4 2.8 GHz with HT</td>
<td>2.6.23.9-rt12</td>
<td>Jaswinder Singh</td>
<td></td>
</tr>
<tr>
<td>Intel</td>
<td>FSC D2151S @Celeron D 2.53 GHz</td>
<td>2.6.20-rt6</td>
<td>Remy Bohmer</td>
<td>Avg latency &lt; 10us, worst case latency: &lt; 30us (see Note 1)</td>
</tr>
<tr>
<td>Intel</td>
<td>FSC D2151S @Core 2 Duo E6300</td>
<td>2.6.20-rt6</td>
<td>Remy Bohmer</td>
<td>Avg latency &lt; 10us, worst case latency: &lt; 30us (see Note 1)</td>
</tr>
<tr>
<td>Intel</td>
<td>Asus PSB @Core 2 Duo E6600 2.4GHz</td>
<td>2.6.23-rc4-rt1</td>
<td>Remy Bohmer</td>
<td>Avg latency &lt; 10us, worst case latency: &lt; 30us (see Note 1)</td>
</tr>
<tr>
<td>Intel</td>
<td>IBM Thinkpad T43 @Pentium M CPU 2 GHz</td>
<td>2.6.20-rt3</td>
<td>PhilK</td>
<td>Avg latency &lt; 14us, worst case latency: &lt; 32us</td>
</tr>
<tr>
<td>ARM9 (v4)</td>
<td>Atmel AT91 RM9200-EK @ (See AT91-Notes)</td>
<td>2.6.23.12-rt14</td>
<td>Remy Bohmer</td>
<td>worst case latency: &lt; 300us (see Note 1)</td>
</tr>
<tr>
<td>ARM9 (v4)</td>
<td>FREE_EC8_AT91 @ (See AT91-Notes)</td>
<td>2.6.23.12-rt14</td>
<td>Carlos Camargo</td>
<td></td>
</tr>
<tr>
<td>ARM9 (v5)</td>
<td>Atmel AT91 SAM9281-EK @ (See AT91-Notes)</td>
<td>2.6.23.1-rt5</td>
<td>Remy Bohmer</td>
<td>worst case latency: &lt; 500us (see Note 1)</td>
</tr>
<tr>
<td>ARM9 (v5)</td>
<td>T1 OMAP5912 OSK @</td>
<td>2.6.23-rt1</td>
<td>OMAP ML @</td>
<td>A small additional OMAP specific patch on top of-rt is needed. See ML.</td>
</tr>
<tr>
<td>ARM9 (v5)</td>
<td>T1 OMAP DM6446 DaVinci @</td>
<td>2.6.23-rt1</td>
<td>DaVinci ML @</td>
<td>A small additional DaVinci specific patch on top of-rt is needed. See ML.</td>
</tr>
<tr>
<td>SH4</td>
<td>Renasas R2D/R2D+ SH7761R CPU</td>
<td>2.6.21-rt2 and later</td>
<td>linux-sh ML @</td>
<td></td>
</tr>
</tbody>
</table>
# Test Summary of CELF RTWG

<table>
<thead>
<tr>
<th>Company</th>
<th>Platform</th>
<th>Kernel version</th>
<th>Results with RT-preempt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>Omap 5912(ARM9), 192 MHZ</td>
<td>2.4.20, 2.6.10</td>
<td>Both 30 ~35us worst case(without RT-preempt, interrupt latency)</td>
</tr>
<tr>
<td>IGEL</td>
<td>SH7551R(SH4), 240 MHZ</td>
<td>2.6.21</td>
<td>&lt;50us worst case</td>
</tr>
<tr>
<td>ETRI</td>
<td>Via EPIA(x86), 1 GHZ</td>
<td>2.6.24.4</td>
<td>65 us worst case</td>
</tr>
<tr>
<td>ETRI</td>
<td>Mainstone(Xscale), 520 MHz</td>
<td>2.6.24.4</td>
<td>1451 us worst case</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>SH4, 240 MHZ</td>
<td>2.6.8</td>
<td>1300 us worst case</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>VIA Eden, 600 MHZ</td>
<td>2.6.8</td>
<td>226 us worst case</td>
</tr>
<tr>
<td>Toshiba</td>
<td>Cell BE, 3.2 GHz</td>
<td>2.6.12</td>
<td>less variability</td>
</tr>
</tbody>
</table>
Plan

- RTWG will try to test by using prior mentioned testing methods and collect testing results
- port necessary test program to various architectures
- write stress program and how to give a stress
- collect more related presentations and documents
- would like to write valuable rt testing methods metric and test results taxonomy