Realtime Testing for Embedded Platforms

Tim Bird
Senior Software Engineer
Sony Corporation of America
Overview

• Introduction to realtime performance testing
• Recent RT-Preempt testing efforts
• Some RT test programs and methods
  – RealFeel-ETRI
• Testing problems in embedded
• Results from CELF members
• Open discussion
Introduction

• Goal of kernel support for realtime is to guarantee that system meets time deadline constraints

• Testing is critical to ensure that constraints are met

• RT testing is different from functionality testing
RT testing issues

• Instrumentation - why
  – To measure performance
    • Want to measure latency of different parts of response time
  – Need to instrument kernel in order to find cause of failures

• Can measure using:
  – Internal instrumentation
  – External instrumentation (separate machine)
Internal instrumentation

- Consists of event tracing, timestamp gathering, logging
- May add too much overhead
  - Instrumentation can disturb results
    - e.g. amount of time to take measurement may be longer than period being measured
  - Heisenburg uncertainty principle
    - Adding measurement code affects the latencies
Clock issues

• How can you tell the clock is correct?
  – Clock used for timestamp has some deviation from wall time
  – Need clock of high resolution, but low cost to read

• A cycle-counter is often used
  – Not all platforms have one, or can’t be read from user space
  – Other clocks are slower, and higher overhead to read
Instrumentation infrastructure

- Can use existing infrastructure??
  - LTTng found to have too much overhead
    - Samsung test indicated that overhead of LTTng had high variability
    - One log routine took > 4000 us.
  - Latency-trace introduces high overhead
    - Adds additional function call on entry to every kernel function
  - Regular kernel time routines are relatively expensive
- Specialized instrumentation used by most tests
External instrumentation

- Can use specialized hardware (e.g. logic analyzer)
- Can use another Linux machine
  - Very good for end-to-end response time.
  - Usually, no detail of latency areas
    - Does not indicate where delays are occurring
    - Can’t be used to fix latency problems.
Recent RT-Preempt testing efforts

- Many CELF members are working on realtime testing
- Some results published recently:
  - Samsung
  - IGEL
  - ETRI
  - Toshiba
  - Mitsubishi
Some RT test programs and methods

- lpptest
- realfeel-etri
- cyclictest
Ipptest

- Built into RT-preempt patch
- User-space program to read and write parallel port, on host machine
- Kernel parallel port driver to respond to incoming data and send response
- Requires parallel port, which most embedded platforms don’t have
  - Trevor Woerner has a similar test, using serial port
    - Conflict with serial console may be an issue, but haven’t investigated it yet.
Realfeel-ETRI

- Programs a clock on the local system to cause periodic interrupts.
- Uses /dev/rtc
  - /dev/rtc appears to be supported on a number of platforms
  - Requires a clock that can be programmed for periodic interrupts
  - Don’t know if it conflicts with clocksources/clockevents
- ETRI added feature to kernel to return timestamp on read of /dev/rtc
realfeel-ETRI details

- Program sets up for RT performance
  - locks memory
  - sets scheduling priority
  - tells /dev/rtc to deliver periodic interrupts

- Program reads /dev/rtc
  - Call blocks until interrupt occurs
  - read returns timestamp of interrupt start

- Program compares interrupt time with time when user-space signal handler runs.
  - This isolates and measures scheduling latency
cyclicctest

- Program by Thomas Gleixner to measure performance of Linux timer mechanisms
- Uses Linux timer routine (posix timer, itimer or nanosleep), and measures expected time vs. actual time for wakeup
- Needs high-res timers for some tests
- I had problems cross-compiling (but this is likely a bug in my development environment)
Problems testing embedded

- RT-preempt ports are still in progress
- clock source for timestamp (no TSC) varies per platform
- stress programs don’t match final load
Miscellaneous notes

• Many stress programs don’t test worst case
  – Heavy load is not the same as highest latency
  – e.g. ping test keeps system in small set of pages
  • no test of memory-related latency (cache misses
  – Need to test error paths
Samsung Results

• Tester: Sangbae Lee, Samsung

• Test Info:
  – Omap 5912, 192 MHZ??, 2.6.10

• What was measured:
  – interrupt latency and IRQ handler duration

• Results:
  – LTTng had long, variable latency
  – 30 us worst case, AFTER using custom instrumentation
IGEL results

- Tester: Katsuya Matsubara, IGEL
- Test Info:
  - SH7751R (SH4), 2.6.21
- What was measured:
  - UART driver implemented in user space (using UIO)
  - Time to receive
- Results:
  - Don’t have exact numbers, but graphs look good!
IGEL results - graph

![IGEL Results Graph](image-url)
ETRI results

- Tester: YungJoon Jung, ETRI
- Test Info:
  - Via Eden, 800 MHZ, 2.6.20
- What was measured:
  - wakeup time from periodic tick (using /dev/rtc)
- Results:
  - 41 us worst case
ETRI results - graph

vanilla kernel

preemptible kernel

voluntary preemption kernel

real-time preemption kernel
Toshiba results

• Tester: Tsutomu Owa, Toshiba

• Test Info:
  – Cell (PPC64), 2.6.12 running on hypervisor

• What was measured:
  – ping response time
  – response to logical partition switch interrupt

• Results:
  – with RT-preempt, ping response time becomes much less variable
  – under load, response to lpar interrupt was faster with RT-preempt
    • there were still some bad lags
Mitsubishi results - SH4

• Tester: Shinichi Ochai, Mitsubishi

• Test Info:
  – SH4, 240 MHZ, 2.6.8

• What was measured:
  – Interrupt latency, process wakeup latency

• Results:
  – 1300 us worst case
  – worst case depended on load
    • Load with access to compact flash had problems
Mitsubishi results - i386

Test Info:
- Via Eden, 600 MHZ, 2.6.8

What was measured:
- Interrupt latency, process wakeup latency

Results:
- 226 us worst case
- 2.6.14 (with RT-preempt patches) had even worse performance with compact flash access (12 ms)
## Results summary

<table>
<thead>
<tr>
<th>Company</th>
<th>Test Info</th>
<th>Results with RT-preempt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>Omap 5912, 2.6.10</td>
<td>30 us worst case</td>
</tr>
<tr>
<td>IGEL</td>
<td>SH7751R, 2.6.21-rc5</td>
<td>good graph (&lt;50us?)</td>
</tr>
<tr>
<td>ETRI</td>
<td>Via Eden, 800 Mhz, 2.6.20</td>
<td>41 us worst case</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>SH4, 240 Mhz, 2.6.8</td>
<td>1300 us worst case</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>Via Eden, 600 Mhz, 2.6.8</td>
<td>226 us worst case</td>
</tr>
<tr>
<td>Toshiba</td>
<td>Cell, 2.6.12</td>
<td>less variability</td>
</tr>
</tbody>
</table>
Open discussion