Evaluation of Real-time Performance in Embedded Linux

LinuxCon Europe 2014

Hiraku Toyooka, Hitachi
<hiraku.toyooka.gu@hitachi.com>
• Hiraku Toyooka
  • Software engineer at Hitachi

• Working on operating systems
  • Linux (mainly) for industrial control systems
    • Server, Embedded
    • Tracing
    • Real-time virtualization (partitioning)
Agenda

• What is a real-time system?
• Real-time features in Linux
• Tuning points for evaluation
• Performance measurement
What is a real-time system?
What is a real-time system?

- Real-time system must always handle events within deadlines
  - Real-time doesn’t mean fast
  - Worst case performance is important
  - Deterministic behavior is required

Cycle #1
- Non-RT system
  - Done
- RT system
  - Done

Cycle #2
- Non-RT system
  - Done
- RT system
  - Done

Cycle #3
- Non-RT system
  - Done
- RT system
  - Done
- Deadline Miss!
Real-time features in Linux
Linux and real-time

- Linux is originally designed to be a general purpose OS (GPOS)
  - High functionality
  - Average or peak performance is usually important for GPOS
  - Latency varies widely
Current mainline Linux has a variety of real-time features

- Certain level of deterministic behavior by appropriate configurations and tuning

- PREEMPT_RT patchset reduce maximum latency much more
  - Many real-time features in mainline are derived from PREEMPT_RT
(Some)Real-time features

• Mainline
  • Fixed-priority scheduling
  • Kernel preemption
  • IRQ thread
  • Others

• Out of tree
  • PREEMPT_RT patchset
Fixed-priority scheduling

- Linux has three types of schedulers
  - Completely Fair Scheduler (CFS)
    - Policy: SCHED_OTHER (Default), _BATCH, _IDLE
    - Task has dynamic-priority based on time slice (non-deterministic)
  - Real-time scheduler
    - Policy: SCHED_FIFO, _RR
    - Task has fixed-priority (deterministic)
  - Deadline scheduler
    - Policy: SCHED_DEADLINE
    - Merged in 3.14
Fixed-priority scheduling (cont.)

- There are per-cpu runqueues in Linux
- RT tasks are always dispatched prior to CFS tasks
Kernel preemption

- Without preemption support: when higher priority tasks wake up, they are delayed until current task exits from a syscall or yields explicitly

- CONFIG_PREEMPT
  - Enable forcible context-switch (preemption) even in kernel mode when a higher priority task wakes up
  - Improve response latency of high priority tasks
Interrupt handling discussion:

- Move a part of the interrupt handler code to a kernel thread
  - Prioritize interrupt handlers and reduce interrupt-disabled section
  - thread_fn is specified when registering IRQ handler

- CONFIG_IRQ_FORCED_THREADING
  - Force threading of all interrupt handlers
    - except handlers marked IRQF_NO_THREAD
  - Enabled by boot parameter “threadirqs”
# ps lgrep irq

<table>
<thead>
<tr>
<th>PID</th>
<th>Username</th>
<th>Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>root</td>
<td>0:00</td>
<td>[ksoftirqd/0]</td>
</tr>
<tr>
<td>64</td>
<td>root</td>
<td>0:00</td>
<td>[irq/12-edma]</td>
</tr>
<tr>
<td>65</td>
<td>root</td>
<td>0:00</td>
<td>[irq/14-edma_err]</td>
</tr>
<tr>
<td>203</td>
<td>root</td>
<td>0:00</td>
<td>[irq/70-omap_i2c]</td>
</tr>
<tr>
<td>921</td>
<td>root</td>
<td>0:00</td>
<td>[irq/30-omap_i2c]</td>
</tr>
<tr>
<td>956</td>
<td>root</td>
<td>0:00</td>
<td>[irq/18-musb-hdr]</td>
</tr>
<tr>
<td>959</td>
<td>root</td>
<td>0:00</td>
<td>[irq/19-musb-hdr]</td>
</tr>
<tr>
<td>967</td>
<td>root</td>
<td>0:00</td>
<td>[irq/75-rtc0]</td>
</tr>
<tr>
<td>968</td>
<td>root</td>
<td>0:00</td>
<td>[irq/76-rtc0]</td>
</tr>
<tr>
<td>977</td>
<td>root</td>
<td>0:00</td>
<td>[irq/64-mmc0]</td>
</tr>
<tr>
<td>978</td>
<td>root</td>
<td>0:00</td>
<td>[irq/166-mmc0]</td>
</tr>
<tr>
<td>988</td>
<td>root</td>
<td>0:00</td>
<td>[irq/77-wkup_m3]</td>
</tr>
<tr>
<td>989</td>
<td>root</td>
<td>0:00</td>
<td>[irq/78-wkup_m3_]</td>
</tr>
<tr>
<td>990</td>
<td>root</td>
<td>0:00</td>
<td>[irq/120-smartre]</td>
</tr>
<tr>
<td>991</td>
<td>root</td>
<td>0:00</td>
<td>[irq/121-smartre]</td>
</tr>
<tr>
<td>1006</td>
<td>root</td>
<td>0:00</td>
<td>[irq/72-OMAP UAR]</td>
</tr>
</tbody>
</table>
Others

- High resolution timer support
- PI (Priority Inheritance) mutex
- Preemptive RCU
- RT group scheduling
- RT throttling
- Full tickless (no_hz)
PREEMPT_RT patchset

• provides more deterministic behavior and quick response
  • It consists of lots of patches lowering maximum latency
    • There are 318 patches in 3.14.2-rt3

• CONFIG_PREEMPT_RT_FULL
  • allows “full preemption”
    • nearly all of the kernel can be preempted
    • replaces spin-locks with mutexes (Sleeping spin-lock)
    • many other improvements
Tuning points for evaluation
Tuning points for evaluation

- Clocksource resolution
- Task priority
- Page faults
- Multi-core
- IRQ handling
- Lock debugging
Clocksource resolution

- Getting and updating clock time depend on clocksource
  - gettimeofday(2), clock_gettime(2)
- This means that time measurement is affected by clocksource resolution
  - Confirmation is necessary before measurement

```bash
# dmesg | grep clock
[0.000000] OMAP clocksource: GPTIMER1 at 24000000 Hz
[0.000000] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 178956ms
```
Task priority

- We need to use RT policy and priorities for deterministic scheduling
  - sched_setscheduler(2) can change those scheduling attributes of specified tasks
  - In pthread library, new thread’s scheduling attributes can be set by pthread_attr_setschedparam(3)
    - Do not forget to call pthread_attr_setinheritsched(3) with PTHREAD_EXPLICIT_SCHED
Page faults

• Memory allocation to userspace is usually only virtual address space allocation
  • Initial memory access causes page fault, and this produces more latency
  • Page-out to swap area also causes page faults

• To prevent page faults, mlockall(2) is used
  • All memory-mapped pages remain on physical memory after mlockall(2)
Multi-core

• In multi-core SoC, RT tasks can move from a local core to remote cores
  • This migration causes additional latency

• Tasks can be fixed to a specific core by CPU affinity
  • taskset(1)
  • sched_setaffinity(2)
  • Cgroup cpuset
• IRQ handling (and IRQ thread processing) add latency to RT tasks
  • It is effective to isolate …
    • cores for executing RT tasks
    • cores for handling IRQ unrelated to the RT tasks

• IRQ and IRQ thread can be fixed to a specific cores by IRQ affinity
Lock debugging features

- Lock debugging features add much more latency
- Disabling them is effective
Performance measurement
Example: Interrupt response time

- measured by cyclictest (in rt-tests)
  - Time period between expiry of a hardware timer and scheduling of a userspace task
  - cyclictest -a 0 -p 99 -m -n -l 100000 -q --h 500
    - -a: set affinity to CPU0
    - -p 99: set priority to 99
    - -m: use mlockall
    - -n: use clock_nanosleep to measure time
    - -l 100000: number of trials
    - -q: don’t print anything while testing
    - --h=500: print histogram 0-499us after the test
### Evaluation environment

<table>
<thead>
<tr>
<th>Board</th>
<th>BeagleBone Black</th>
<th>Altima Helio board</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoC</strong></td>
<td><em>TI Sitara AM3359</em></td>
<td><em>Altera Cyclone V</em></td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td><em>ARM Cortex-A8 (1GHz)</em></td>
<td><em>ARM Cortex-A9 (800MHz x2)</em></td>
</tr>
<tr>
<td><strong>L1 cache (inst. / data)</strong></td>
<td>32KB / 32KB</td>
<td>32KB / 32KB</td>
</tr>
<tr>
<td><strong>L2 cache</strong></td>
<td>256KB</td>
<td>512KB</td>
</tr>
<tr>
<td><strong>Main memory</strong></td>
<td>512MB</td>
<td>1GB</td>
</tr>
<tr>
<td><strong>Linux version</strong></td>
<td>3.14.0</td>
<td>3.14.0 [1]</td>
</tr>
<tr>
<td><strong>PREEMPT_RT version</strong></td>
<td>3.14.0-rt1</td>
<td>3.14.0-rt1</td>
</tr>
</tbody>
</table>

Comparison

- Two types of kernels
  - Mainline
  - PREEMPT_RT

<table>
<thead>
<tr>
<th>Configurations</th>
<th>Mainline</th>
<th>PREEMPT_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel version</strong></td>
<td>3.14.0</td>
<td>3.14.0-rt1</td>
</tr>
<tr>
<td><strong>Preemption</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Forced IRQ threading</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Full preemption (RT patch)</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Results (100Mbps load)

### BeagleBone Black (100Mbps)

- **Mainline**
  - Avg: 30us
  - Max: 88us

- **PREEMPT_RT**
  - Avg: 25us
  - Max: 55us

### Helio (100Mbps)

- **Mainline**
  - Avg: 15us
  - Max: 108us

- **PREEMPT_RT**
  - Avg: 11us
  - Max: 30us
Results (1Gbps load)

- Mainline kernel is significantly affected by large amount of I/O
Set IRQ affinity

Cyclic test

CPU 0

eth

CPU 1

echo 2 > /proc/irq/IRQ#/smp_affinity

Cyclic test

CPU 0

eth

CPU 1

Settings

<table>
<thead>
<tr>
<th></th>
<th>Mainline</th>
<th>PREEMPT_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Gbps load</td>
<td>29147us</td>
<td>53us</td>
</tr>
<tr>
<td>Above + IRQ affinity</td>
<td>51us</td>
<td>42us</td>
</tr>
</tbody>
</table>

© Hitachi, Ltd. 2014. All rights reserved.
Add 1000 tasks load

stress –c 1000 –q

<table>
<thead>
<tr>
<th>Settings</th>
<th>Mainline</th>
<th>PREEMPT_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Gbps load</td>
<td>29147us</td>
<td>53us</td>
</tr>
<tr>
<td>Above + IRQ affinity</td>
<td>51us</td>
<td>42us</td>
</tr>
<tr>
<td>Above + 1000 tasks</td>
<td>403us</td>
<td>53us</td>
</tr>
</tbody>
</table>
Set CPU affinity

```
taskset 0x2 \ stress -c 1000 -q
```

<table>
<thead>
<tr>
<th>Settings</th>
<th>Mainline</th>
<th>PREEMPT_RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Gbps load</td>
<td>29147us</td>
<td>53us</td>
</tr>
<tr>
<td>Above + IRQ affinity</td>
<td>51us</td>
<td>42us</td>
</tr>
<tr>
<td>Above + 1000 tasks load</td>
<td>403us</td>
<td>53us</td>
</tr>
<tr>
<td>Above + CPU affinity</td>
<td>49us</td>
<td>40us</td>
</tr>
</tbody>
</table>
Conclusion

- Kernel configurations and use of appropriate APIs are necessary to enable Linux real-time features
- Current mainline Linux
  - has lots of real-time features
  - may be available for near real-time processing under limited situation (limited load, multi-core, not quick response, etc.)
- PREEMPT_RT patchset
  - provides more deterministic behavior
  - Is necessary for hard real-time processing
Trademarks

• Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
• Other company, product, or service names may be trademarks or service marks of others.