Experience with realtime performance of the current Linux technologies

Shinichi Ochiai
Mitsubishi Electric Corporation
Information Technology R&D Center
Agenda

1. Background
2. Realtime Technologies for Linux
3. Evaluation Targets
4. Evaluation
   4.1. 2.4/2.6 Preemptible Kernel Effect
   4.2. Efficiency of New Realtime Patches
   4.3. Porting RT Patches to SH4
   4.4. Case of Interrupt Response
   4.5. Comparison with Hybrid Approach
5. Conclusion
1. Background

Embedded Linux moves into realtime.

- Since 2.4/2.6 kernel, realtime area is going to be one of the target scope of Linux.
- Mainline kernel supports priority based and preemptible scheduling.
- Number of realtime extension patches are also available on Linux.
- To use Linux as an operating system for dedicated system, predictable response time is important factor for system design.
1. Background (*cont.*)

But realtime capability of Linux in practical area is still unclear.

- Number of new technologies are inserted each new version of kernel.
- Target processor and I/O are varied in embedded area. And also application requirements are varied
- Few reference benchmark are available.

Enlightening realtime performance of Embedded Linux from external behavior
## 2. Realtime Technologies for Linux

<table>
<thead>
<tr>
<th></th>
<th>2.4</th>
<th>CE Linux 1.0</th>
<th>2.6.0 Dec, ’03</th>
<th>2.6.8 (Voluntary-preempt patch) Aug, ’04</th>
<th>2.6.14 Oct, ’05</th>
<th>2.6.14 (Preempt-RT patch) Oct, ’05</th>
<th>2.6.16 Jan, ’06</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(1) Scheduler</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Preemptible Kernel</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Lock Breaking</td>
<td>X</td>
<td>O</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Voluntary Kernel Preemption</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Preempt Big Kernel Lock</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Interrupt Threads</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Replace spinlock with PI Mutex</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Read-Copy Update</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>IO Scheduler</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>High Resolution Timer</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>NPTL</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
2. Realtime Technologies for Linux (cont.)

Key Features for Realtime

- **O(1) scheduler**
  - enable priority based scheduling with fixed overhead
- **Preemptible Kernel**
  - enable preemption except spinlock region
- **Voluntary Preemption**
  - insert additional preemption points
- **Threaded IRQ (Interrupt Thread)**
  - handle IRQ routines in thread
- **PI Mutex (preempt-RT)**
  - replace spinlock with priority inheritance mutex
2. Realtime Technologies for Linux (cont.)

2.4 kernel

- Jul. ‘01: 2.4.7
- Nov. ‘01: 2.4.15
- Nov. ‘02: 2.4.20

2.6 kernel

- Dec. ‘03: 2.6.0
- 2.6.11
- 2.6.13
- 2.6.16

O(1) Scheduler

Preemptible kernel

MVL 2.1

CE Linux 1.0

IO Scheduler

NPTL

RCU

Preempt BKL

Voluntary Preempt

HR Timer

Interrupt Threads

Spinlock PI Mutex

Voluntary Preempt

Preempt-RT

Changes for the Better
3. Evaluation Targets

Target Applications
- Video recorder with Network interface
- Smart Phone
- Distributed Controller

Requirements
- Device support: networks, Storage/Flash Storage, VGA, special IO
- Dynamically invoked multi-applications
- Realtime response time:
  10s microseconds – 100s microseconds
### 3. Evaluation Targets (cont.)

#### Target Boards

<table>
<thead>
<tr>
<th>Target</th>
<th>CPU</th>
<th>Memory, Bus</th>
<th>IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target 1</td>
<td>SH4 (Renesas) 240MHz</td>
<td>64MB, 120MHz</td>
<td>serial, CF, LAN</td>
</tr>
<tr>
<td>Target 2</td>
<td>Eden (VIA) 600MHz</td>
<td>256MB, 400MHz</td>
<td></td>
</tr>
<tr>
<td>Target 3</td>
<td>MPC7410 (freescale) 500MHz</td>
<td>512MB, 100MHz</td>
<td></td>
</tr>
</tbody>
</table>
3. Evaluation Targets (cont.)

Measurement Method

Target realtime process (rt-prio. level 95)

Timer interrupt

Interrupt latency: $t_1$ – timer register setting

Process preemption latency: $t_2$ – timer register setting

Measure $t_1$ and $t_2$ using free-run HW timer

Background applications (TSS priority)

Timer interrupt handler

Interrupt disable

HW

Timer interrupt
3. Evaluation Targets (cont.)

Application Scenarios

1. Stable State:
   stable ready state

2. Dynamic Process:
   process/thread creation and termination

3. Dynamic Memory:
   shared/device memory mapping control

4. CF PIO:
   access to CF storage device and other special devices
4. Evaluation

4.1. 2.4/2.6 Preemptible Kernel Effect

- RT performance improvement between 2.4 (CE Linux 1.0) kernel and 2.6.8 kernel
- RT performance improvement between non-preemptible kernel and preemptible kernel
- Target: SH4/240MHz
(1) SH4: [Stable State] 2.4/2.6Kernel

- **CE Linux 1.0 (normal)**
  - worst: 315 us
  - 99.9%: 66 us
  - average: 31 us

- **CE Linux 1.0 (preemptible)**
  - worst: 592 us
  - 99.9%: 154 us
  - average: 40 us

- **2.6.8 kernel (normal)**
  - worst: 73 us
  - 99.9%: 41 us
  - average: 32 us

- **2.6.8 kernel (preemptible)**
  - worst: 283 us
  - 99.9%: 167 us
  - average: 42 us
(2) SH4: [dynamic process] 2.4/2.6 Kernel

- **CE Linux 1.0 (normal)**
  - Worst: 9318 us
  - 99.9%: 501 us
  - Average: 36 us

- **CE Linux 1.0 (preemptible)**
  - Worst: 790 us
  - 99.9%: 182 us
  - Average: 37 us

- **2.6.8 kernel (normal)**
  - Worst: 790 us
  - 99.9%: 182 us
  - Average: 41 us

- **2.6.8 kernel (preemptible)**
  - Worst: 1059 us
  - 99.9%: 279 us
  - Average: 45 us
(3) SH4: [dynamic memory] 2.4/2.6Kernel

- CE Linux 1.0 (normal)
  - worst: 2052 us
  - 99.9%: 1889 us
  - average: 679 us

- CE Linux 1.0 (preemptible)
  - worst: 2022 us
  - 99.9%: 1896 us
  - average: 194 us

- 2.6.8 kernel (normal)
  - worst: 1319 us
  - 99.9%: 1281 us
  - average: 647 us

- 2.6.8 kernel (preemptible)
  - worst: 1300 us
  - 99.9%: 1184 us
  - average: 149 us
(4) SH4: [CF PIO] 2.4/2.6Kernel

worst: 119.7 ms
99.9%: 67.4 ms
average: 9.0 ms

worst: 143.4 ms
99.9%: 78.6 ms
average: 9.3 ms

worst: 66.7 ms
99.9%: 54.8 ms
average: 18.3 ms

worst: 57.8 ms
99.9%: 56.2 ms
average: 24.5 ms
2.4/2.6 Preemptible Kernel Effect (cont.)

Summary

- RT behavior differs on background application type. (memory management, file system and CF access make worse RT performance.)
- Preemptible kernel fairly improved RT performance on 2.4 version.
- 2.6 kernel makes good RT performance even non-preemptible. (RT performance difference by preemptible is less than 2.4.)
- Looks like unusable for RT while CF access has run. (driver implementation issue?)
4. Evaluation (cont.)

4.2. Efficiency of New Realtime Patches

- RT performance improvement between standard 2.6 kernel, voluntary preempt kernel, and preempt-RT kernel

- Reference performance of reference architecture (IA32 compatible).

- Target: VIA Eden/600MHz
(3) Eden: [dynamic memory] 2.6Kernel+RT

- **Eden 2.6.8 kernel (normal)**
  - worst: 1273 us
  - 99.9%: 778 us
  - average: 88 us

- **Eden 2.6.8 kernel (preemptible)**
  - worst: 1263 us
  - 99.9%: 465 us
  - average: 79 us

- **Eden 2.6.8 kernel (voluntary preempt)**
  - worst: 1298 us
  - 99.9%: 209 us
  - average: 73 us

- **Eden 2.6.14 kernel (preempt-rt)**
  - worst: 226 us
  - 99.9%: 88 us
  - average: 68 us
(4) Eden: [CF PIO] 2.6Kernel+RT

- Eden 2.6.8 kernel (normal) [CF PIO load]
  - worst: 4121 us
  - 99.9%: 1872 us
  - average: 246 us

- Eden 2.6.8 kernel (preemptible) [CF PIO load]
  - worst: 3846 us
  - 99.9%: 1041 us
  - average: 219 us

- Eden 2.6.8 kernel (voluntary preempt) [CF PIO load]
  - worst: 1156 us
  - 99.9%: 932 us
  - average: 83 us

- Eden 2.6.14 kernel (preempt-rt) [CF PIO load]
  - worst: 1172 us
  - 99.9%: 669 us
  - average: 133 us
Efficiency of New Realtime Patches \((cont.)\)

Summary

- Each RT technology insertion improves 2.6 RT performance. Especially preempt-RT patch is effective remarkably.

- CF access makes RT performance worse, but it is permissible as comparison with SH4 results.
4. Evaluation (cont.)

4.3. Porting RT patches to SH4

- Preempt BKL and voluntary preemption are merged to standard kernel at 2.6.14. They can be enabled on SH.

- Porting preempt-RT patch (threaded IRQ and PI mutex) to SH is still on going. It is not included in this evaluation.

- Target: SH4 240MHz
(3) Eden/SH4: [dynamic memory] 2.6Kernel+RT

Eden 2.6.8 kernel (preemptible) [dynamic memory load]

worst: 1263 us
99.9%: 465 us
average: 79 us

Eden 2.6.8 kernel (voluntary preempt) [dynamic memory load]

worst: 1298 us
99.9%: 209 us
average: 73 us

Eden 2.6.14 kernel (preemptible) [dynamic memory load]

worst: 1300 us
99.9%: 1184 us
average: 149 us

worst: 642 us
99.9%: 224 us
average: 124 us
(4) Eden/SH4: [CF PIO] 2.6Kernel+RT

- Eden 2.6.8 kernel (preemptible) [CF PIO load]
  - Occurrence
  - worst: 3846 us
  - 99.9%: 1041 us
  - average: 219 us

- Eden 2.6.14 kernel (preemptible) [CF PIO load]
  - Occurrence
  - worst: 57.8 ms
  - 99.9%: 56.2 ms
  - average: 24.5 ms

- Eden 2.6.8 kernel (voluntary preempt) [CF PIO load]
  - Occurrence
  - worst: 1156 us
  - 99.9%: 932 us
  - average: 83 us

- Eden 2.6.14 kernel (voluntary preempt) [CF PIO load]
  - Occurrence
  - worst: 12.0 ms
  - 99.9%: 4.0 ms
  - average: 1.7 ms
Porting RT patches to SH4 (cont.)

Summary

- Relatively same effects were gained both Eden and SH4 on dynamic memory load. (no architecture dependency.)
- 2.6.14 kernel has better RT performance since 2.6.8 kernel on SH4. RT performance while CF access is also improved but not enough.
- Expects preempt-RT patch will improve performance much more either on SH4.
4. Evaluation (cont.)

4.4. Case of Interrupt Response

- Measurement of kernel level latency
- Interrupt response time improvement by each RT patch
- Architecture dependency: difference between Eden and SH4
- Target: Eden 600MHz/SH4 240MHz
(3) Eden: [dynamic memory] 2.6Kernel+RT
(4) Eden: [CF PIO] 2.6Kernel+RT

Elapsed Jitter (micro-seconds)

Eden 2.6.8 kernel (normal)
Eden 2.6.8 kernel (preemptible)
Eden 2.6.8 kernel (voluntary preempt)
Eden 2.6.14 kernel (preempt-rt)

Apr. 12th, 2006 © Mitsubishi Electric Corporation, 2006 28
(3) Eden/SH4: [dynamic memory] 2.6Kernel+RT

![Diagram of Eden/SH4: 2.6Kernel+RT](image1)

![Diagram of Eden 2.6.8 kernel (normal)](image2)

![Diagram of Eden 2.6.14 kernel (preempt-rt)](image3)

![Diagram of SH4 2.6.14 kernel (preemptible)](image4)
(4) Eden/SH4: [CF PIO] 2.6Kernel+RT

Eden 2.6.8 kernel (normal) [CF PIO load]

Eden 2.6.14 kernel (normal) [CF PIO load]

Eden 2.6.14 kernel (preempt-rt) [CF PIO load]

Eden 2.6.14 kernel (preemptable) [CF PIO load]
Case of Interrupt Response (cont.)

Summary

- Preempt-RT patch can realize 10 microseconds response time inner Linux kernel. It is effective to long-term running interrupt handler.

- Bad RT performance is also shown in interrupt response on SH4. (to be solved.)

- Expects threaded IRQ and PI mutex will be a key for performance improvement.
4. Evaluation (cont.)

4.5. Comparison with Hybrid Approach

- Hybrid operating system will enable stable RT performance by its design. Is performance of preempt-RT Linux comparable with the hybrid Linux?

- Target: MPC7410 500MHz/
  Eden 600MHz

<table>
<thead>
<tr>
<th>Adeos</th>
<th>Adeos</th>
<th>RTAI 3.1</th>
<th>RTAI 3.1</th>
<th>Linux 2.6</th>
<th>Linux 2.6</th>
<th>MPC7410</th>
<th>MPC7410</th>
</tr>
</thead>
</table>

Reference of hybrid Linux
PPC7410: Adeos/RTAI vs. Eden preempt-RT
Comparison with Hybrid Approach (cont.)

Summary

- RT performance of hybrid Linux is stable on various application loads on RTAI or Linux.
- Preempt-RT patch can realize comparable RT performance with the hybrid Linux. Interrupt response time is same level as realtime operating system.
Issues

- Priority mapping of current kernel threads: Which service should be prioritize? How to avoid priority inversion?
- Driver design guideline for realtime: Need the reference design for good realtime performance: interrupt level control, nested interrupt handling, interrupt handler, tasklet, kernel thread.
5. Conclusion

- Embedded Linux will be ready for realtime soon. We will accelerate porting architecture dependent code of those RT patches.

- We understand we have solutions to satisfy our requirements. We can choose standard Linux kernel or it’s extensions to adapt required realtime performance.