Threaded IRQs on Linux PREEMPT-RT

Luís Henriques
Intel, Shannon

OSPERT 2009
Agenda

• Threaded IRQs overview
  – Why threaded IRQs
  – PREEMPT-RT overview
  – Threaded IRQs in PREEMPT-RT

• Experimental results
  – Experiments description
  – Experiments results
  – Conclusions
Why threaded IRQs

• Threaded IRQs is a common design-pattern in other operating systems

• Benefits:
  – Increased observability
  – Interaction between interrupt handlers and softirqs/tasklets can be simplified
    – Reduced locking complexity
  – Improve system predictability

• But:
  – Overall system throughput can decrease
PREEMPT-RT overview

- Linux is a GPOS kernel
  - Give all tasks a **fair** share of resources
- Latencies depend on everything running on the system
- Main cause: preemption may be switched off for an unknown amount of time
- Thus, Linux **does not** guarantee timing
  - Although it is considered ‘good enough’ for many applications
PREEMPT-RT overview
Main characteristics

• Complete kernel preemption
  – Reduces scheduling latency by replacing most of the spinlocks with blocking mutexes
• High-resolution timers
• Priority inheritance protocol
• Threaded IRQs
Threaded IRQs in PREEMPT-RT
ISRs on Linux

• With mainline Linux, when an interrupt occurs, CPU is preempted and ISR is executed
  – ISR is executed at highest priority
    – Typically with interrupts disabled or current interrupt line masked off
    – ISRs can be preempted only by other interrupts

• A well written device driver:
  – Do very little work on ISR
  – Push time-consuming activities to kernel threads, tasklets or softirqs
Threaded IRQs in PREEMPT-RT
ISR}s on PREEMPT-RT Linux

- Device drivers register interrupt handler with usual interface (request_irq())
  - No modifications required in device drivers
- A thread is created for the IRQ
  - Only one thread per IRQ
- Kernel keeps a list of ISRs for each IRQ
  - ISRs are sequentially invoked for shared IRQs
- Some drivers may not want their interrupt handlers threaded (e.g., clock and serial I/O on FreeBSD)
  -IRQ_NODELAY flag for non-threaded IRQs
Threaded IRQs in PREEMPT-RT
ISRs on PREEMPT-RT Linux

- `do_IRQ()`
- `handle_*_irq()`
- `IRQ flag to IRQ_INPROGRESS`
- `redirect_hardirq()`

- **Yes**: `threaded_IRQ()`, `wake_up_process()`
- **No**: `handle_IRQ_event()`
## Agenda

- **Threaded IRQs overview**
  - Why threaded IRQs
  - PREEMPT-RT overview
  - Threaded IRQs in PREEMPT-RT

- **Experimental results**
  - Experiments description
  - Experiments results
  - Conclusions
Experiments description
Test environment

- Traffic Generator
- SUT
- Workstation

Ethemet connection
Experiments description
Test environment

- **Hardware:**
  - Intel® EP80579 processor
  - UP SoC at 800MHz
  - Intel® 82572EI Gigabit Ethernet
  - PCI Express

- **Kernel:**
  - 2.6.29.3 and 2.6.29.3-rt13 patchset:
    - Vanilla
    - PREEMPT-RT with Threaded IRQs only
    - PREEMPT-RT with Threaded IRQs + complete preemption

- **Cyclicetest**
  - Measures accuracy of wakeup from sleep (500 usecs)
Experiments description

Test campaigns

• Five test scenarios:
  - Vanilla kernel
  - PREEMPT-RT kernel with Threaded IRQs config options:
    - Cyclictest with low priority
    - Cyclictest with high priority
  - PREEMPT-RT kernel with Threaded IRQs + “Complete preemption” config options:
    - Cyclictest with low priority
    - Cyclictest with high priority

• Traffic injected at several (fixed) rates
  - 64 bytes packets
### Experiments results

#### Vanilla kernel

<table>
<thead>
<tr>
<th>Traffic Rate (usecs)</th>
<th>cyclic test</th>
<th>IRQs</th>
<th>Lost Frames (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>48</td>
<td>585</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>48</td>
<td>1,180</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>52</td>
<td>1,051</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>76</td>
<td>1,435</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>94</td>
<td>1,893</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>193</td>
<td>6,553</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>5</td>
<td>89</td>
</tr>
</tbody>
</table>
## Experiments results
Threaded IRQs kernel, cyclic test priority LOW

<table>
<thead>
<tr>
<th>Traffic Rate (usecs)</th>
<th>cyclic test</th>
<th>IRQs</th>
<th>Lost Frames (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>128,474,238</td>
<td>268,133,338</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>129,139,176</td>
<td>269,522,352</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>136,136,357</td>
<td>274,904,968</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>139,804,614</td>
<td>280,713,676</td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>154,922,510</td>
<td>310,819,204</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>607,364,076</td>
</tr>
</tbody>
</table>
## Experiments results
Threaded IRQs kernel, cyclic test priority HIGH

<table>
<thead>
<tr>
<th>Traffic Rate (usecs)</th>
<th>cyclictest</th>
<th>IRQs</th>
<th>Lost Frames (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>8</td>
<td>479</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>74</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>135,580,508</td>
<td>272,452,238</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>140,860,392</td>
<td>282,329,933</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>156,278,465</td>
<td>313,641,236</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
<td>93</td>
</tr>
</tbody>
</table>
## Experiments results
PREEMPT-RT kernel, cyclictest priority LOW

<table>
<thead>
<tr>
<th>Traffic Rate (usecs)</th>
<th>cyclictest</th>
<th>IRQs</th>
<th>Lost Frames (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>60</td>
<td>1,417</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>74</td>
<td>1,294</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>129</td>
<td>1,466</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>152</td>
<td>1,546</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1,610</td>
<td>75,579</td>
</tr>
<tr>
<td>5</td>
<td>370,627</td>
<td>252,284,884</td>
<td>496,880,098</td>
</tr>
</tbody>
</table>
Experiments results
PREEMPT-RT kernel, cyclic test priority HIGH

<table>
<thead>
<tr>
<th>Traffic Rate (usecs)</th>
<th>cyclic test</th>
<th></th>
<th>IRQs</th>
<th></th>
<th>Lost Frames (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
<td>eth0</td>
<td>eth1</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>9</td>
<td>27</td>
<td>6,158,486</td>
<td>5,835,824</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>9</td>
<td>29</td>
<td>5,792,293</td>
<td>5,617,035</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>32</td>
<td>5,558,052</td>
<td>5,573,145</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>9</td>
<td>30</td>
<td>4,774,998</td>
<td>4,898,368</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>42</td>
<td>25,880</td>
<td>3,398,359</td>
<td>3,431,860</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1,249</td>
<td>50,153</td>
<td>454,401</td>
<td>407,200</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
<td>25</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusions

- Overview of threaded IRQs on PREEMPT-RT
- Experimental results:
  - Real-Fast: vanilla kernel
  - Real-Time: PREEMPT-RT kernel
    - Packets are lost at higher rates
- Probably, there’s still space for optimisations:
  - IRQ threads with same priority sharing same thread
  - No thread if there are no higher priority threads
    - i.e., postpone context switch
  - Others?