Improvement of Scheduling Granularity for Deadline Scheduler

Yoshitake Kobayashi
Advanced Software Technology Group
Corporate Software Engineering Center
TOSHIBA CORPORATION
Outline

- Motivation
- Deadline scheduler
- SCHED_DEADLINE and its evaluation
- Budget management
- Conclusion
Motivation

- We would like to use Linux on control systems
- Real-time is one of the most critical topics

Problem statement

- Need to evaluate deeply to meet the deadline
- CPU resource used too much by higher priority tasks

EDF scheduler
Definition of deadline

- **Wakeup time**: The time of an event occurred (Ex. Timer interrupt) and target task’s state changed to runnable.
- **Event response time**: Interrupt latency
- **Deadline**: The time for a target task must finish

![Diagram showing the relationship between Wakeup time, Event response time, Process time, Deadline, and Period.]

TOSHIBA Confidential
Earliest Deadline First scheduling (EDF)

- The earliest deadline task has the highest priority
- Task’s priority is dynamically changed and managed
  - SCHED_FIFO is static priority management
- Theoretically the total CPU usage by all tasks is up to 100%
  - Includes the kernel overhead
  - If usage of CPU by all tasks is less than 100%, all tasks meet the deadline

Reference
An example of EDF Scheduling

- Task1: budget 1ms  period 8ms
- Task2: budget 2ms  period 5ms
- Task3: budget 4ms  period 10ms

CPU usage = 0.925% < 100%
Rate-Monotonic Scheduling (RMS)

- One of the popular scheduling algorithm for RTOS
- Assumptions for task behavior
  - NO resource sharing such as hardware, a queue, or any kind of semaphore
  - Deterministic deadlines are exactly equal to periods
  - Static priorities (the task with the highest static priority that is runnable immediately preempts all other tasks)
  - Static priorities assigned according to the rate monotonic conventions (tasks with shorter periods/deadlines are given higher priorities)
  - Context switch times and other thread operations are free and have no impact on the model
- CPU utilization
  - \( n \): number of periodic tasks, \( T_i \): Release period, \( C_i \): Computation time
  \[
  U = \sum_{i=0}^{n} \frac{C_i}{T_i} \leq n(\sqrt{n} - 1) \xrightarrow[n=\infty]} \ln 2 \approx 0.69
  \]
  - CPU utilization depends on the combination of periodic tasks and it is possible to meet the deadline even the CPU utilization is around 80%
- Reference
Compared with the RMS scheduling

- Task1: budget 1ms, period 4ms
- Task2: budget 2ms, period 6ms
- Task3: budget 3ms, period 8ms

CPU usage = 0.958%
Comparison of deadline algorithms

<table>
<thead>
<tr>
<th></th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS</td>
<td>Easier to implement</td>
<td>Evaluation for scheduling possibility is required to meet the deadline</td>
</tr>
<tr>
<td>EDF</td>
<td>No evaluation for scheduling possibility is required to meet the deadline</td>
<td>Difficult to implement</td>
</tr>
</tbody>
</table>
SCHED_DEADLINE

  - Implements the EDF scheduling algorithm
  - Posted to LKML by Dario Faggioli and Juri Lelli
  - Latest version is V6 (2012/10/24)
    - But V2 and V4 are used in our evaluation.

Key features of SCHED_DEADLINE

- Temporal isolation
  - The temporal behavior of each task (i.e., its ability to meet its deadlines) is not affected by the behavior of any other task in the system
  - Each task is characterized by the following aspects:
    - Budget: sched_runtime
    - Period: sched_period, equal to its deadline
Build SCHED_DEADLINE (linux kernel)

- Get rt-deadline from the following place
  - git clone git://gitorious.org/rt-deadline (for V2)

- Kernel configuration
  - CONFIG_EXPERIMENTAL = y
  - CONFIG_CGROUPS = y
  - CONFIG_CGROUP_SCHED = n
  - CONFIG_HIGH_RES_TIMERS = y
  - CONFIG_PREEMPT = y
  - CONFIG_PREEMPT_RT = y
  - CONFIG_HZ_1000 = y

- Note: For V6
  - git clone git://github.com/jlelli/sched-deadline.git
Overview of SCHED_DEADLINE
Overview of SCHED_DEADLINE

EDF task

- Set deadline and budget
- System call
- sched_prama_ex

User

- Set parameters for SCHED_DEADLINE
- procfs
- sysctl_sched_dl_runtime
- sysctl_sched_dl_period

Kernel

- task_struct
  - sched_dl_entity
    - dl_deadline
    - rb_node
    - dl_runtime
    - dl_timer
    - sched_class
  - enqueue_task_dl
  - dequeue_task_dl
  - task_tick_dl
  - set_curr_task_dl

- dl_sched_class
Setting CPU utilization for EDF tasks

- **Parameters can be setted via procfs**
  - CPU utilization for rt(SCHED_FIFO and SCHED_RR) and dl(SCHED_DEADLINE) should be under 100%
  - Parameters for EDF scheduler
    - `/proc/sys/kernel/sched_dl_period_us`
    - `/proc/sys/kernel/sched_dl_runtime_us`

- **When a task requires more than above limit, the task cannot submit to run**

- **An example setting (rt: 50%, dl:50%)**
  - `# echo 500000 > /proc/sys/kernel/sched_rt_runtime_us` (500ms)
  - `# echo 100000 > /proc/sys/kernel/sched_dl_period_us` (100ms)
  - `# echo 50000 > /proc/sys/kernel/sched_dl_runtime_us` (50ms)
Overview of SCHED_DEADLINE

EDF task

Set deadline and runtime

System call

User

Set parameters for SCHED_DEADLINE

sysctl_sched_dl_runtime

sysctl_sched_dl_period

Kernel

task_struct

sched_dl_entity

dl_deadline

dl_runtime

dl_timer

sched_class

sched_prama_ex

Kernel

dl_sched_class

enqueue_task_dl

dequeue_task_dl

task_tick_dl

set_curr_task_dl
Run a EDF task

- **Schedtool**
  - `# schedtool -E -t 10000:100000 -a 0 -e ./yes`
  - Options
    - `-E`: a task runs on `SCHED_DEADLINE`
    - `-t`: `<execution time>` and `<period>` in micro seconds
    - `-a`: Affinity mask
    - `-e`: command

- **System call**
  - `sched_setscheduler_ex()`
Budget management for EDF tasks

EDF task

Set deadline and runtime

System call

User

Set parameters for SCHED_DEADLINE

tasks

task_struct

sched_dl_entity

dl_deadline

dl_runtime

dl_timer

... 

sched_class

Kernel

dl_sced_class

enqueue_task_dl

dequeue_task_dl

task_tick_dl

set_curr_task_dl

...
Budget management for EDF tasks

- Each task on SCHED_DEADLINE has budget which allows it to use CPU

- Budget management
  - Refill budget: `dl_timer` (high resolution timer)
  - Use budget: `task_tick_dl` (tick based)

![Diagram of budget management for EDF tasks]
Evaluation (Period:100ms, Budget: 50ms)

<table>
<thead>
<tr>
<th>#</th>
<th>CPU</th>
<th>Time Stamp</th>
<th>Task</th>
<th>PID</th>
<th>Latency</th>
<th>Event</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>156.213747 &lt;(ide)</td>
<td>d,0</td>
<td>sched_switch</td>
<td>swapper/0:0</td>
<td>11201</td>
<td>R --- deadline-vm-test:1040</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_start</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_start</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_start</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_start</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_start</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_cancel</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>156.213861</td>
<td>d,0</td>
<td>hrtimer_expired_entry</td>
<td>hrtimer</td>
<td>405803702752</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation (Period: 100ms, Budget: 10ms)

![Kernel traces screenshot]

<table>
<thead>
<tr>
<th>#</th>
<th>CPU</th>
<th>Time Stamp</th>
<th>Task</th>
<th>PID</th>
<th>Latency</th>
<th>Event</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>586.129958</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>586.139924</td>
<td>yes</td>
<td>2237</td>
<td>d:3</td>
<td>sched_switch</td>
<td>yes:2237 [2247483649] R ==&gt; swapper:0 [120]</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>586.229944</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>yes:2237 [2247483649] R ==&gt; swapper:0 [120]</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>586.239968</td>
<td>yes</td>
<td>2237</td>
<td>d:3</td>
<td>sched_switch</td>
<td>yes:2237 [2247483649] R ==&gt; swapper:0 [120]</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>586.330326</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>586.480292</td>
<td>yes</td>
<td>2237</td>
<td>d:3</td>
<td>sched_switch</td>
<td>yes:2237 [2247483649] R ==&gt; swapper:0 [120]</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>586.430811</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>586.430811</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>586.520969</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>586.530663</td>
<td>yes</td>
<td>2237</td>
<td>d:3</td>
<td>sched_switch</td>
<td>yes:2237 [2247483649] R ==&gt; swapper:0 [120]</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>586.610379</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>586.640345</td>
<td>yes</td>
<td>2237</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>586.710655</td>
<td>&lt;idle&gt;</td>
<td>0</td>
<td>d:3</td>
<td>sched_switch</td>
<td>swapper:0 [120] R ==&gt; yes:2237 [2247483649]</td>
</tr>
</tbody>
</table>

overriding event (19) trace/blktrace with new print handler
Evaluation

- Task T1: budget 1ms period 4ms
- Task T2: budget 2ms period 6ms
- Task T3: budget 3ms period 8ms
Evaluation (Period: 1ms, Budget: 0.5ms)
Budget management for EDF tasks

- Each task on SCHED_DEADLINE has budget which allows it to use CPU
- **Budget management**
  - Refill budget: dl_timer (high resolution timer)
  - Use budget: task_tick_dl (tick based)
- **An Issue**
  - Difficult to keep task’s budget if the budget has micro seconds granularity

**Step 1: Refill budget**
- 1.5 ms
  - dl_timer

**Step 2: Use budget**
- Task execution
  - 1 ms
  - 2 ms
- Period
- Wakeup

**Time**
Support for micro seconds granularity

- **Overview**
  - When a task’s budget is less than 1ms, set HRTICK for the rest of budget

---

**Step1: Refill budget**

**Step2: Use budget**

Task execution

Wakeup

Period

Set HRTICK here

Time
Evaluation (Cycle: 1ms, Budget: 0.5ms)
Advantage and Disadvantage

- **Advantage**
  - Easy to support high resolution budget

- **Disadvantage**
  - Increase overhead
Conclusion

- **SCHED_DEADLINE** is useful for real time systems
- **An Enhancement for budget management**
  - Support fine grained budget such as 100 micro seconds
  - HRTICK is needed to support fine grained budget

- **What we’ve done:**
  - Backport the **SCHED_DEADLINE** v4 to kernel v3.0-rt
    - Because v3.0 is the base version of LTSI
  - Implement this improvement into **SCHED_DEADLINE** v4 on kernel v3.0-rt
  - Forwardport it to original **SCHED_DEADLINE** v4
  - Send a patch to the author of **SCHED_DEADLINE**
Thank you