Implementation of Overrun and Skipping in VxWorks

Mikael Åsberg, Moris Behnam and Thomas Nolte
MRTC/Mälardalen University
Sweden

Reinder J. Bril
Technische Universiteit Eindhoven
The Netherlands

PROGRESS
A national Swedish Strategic Research Centre
Overview

- Background (Resource sharing in the hierarchical scheduling environment)
  - Overrun
  - Skipping
- Preliminaries:
  - Stack resource policy (SRP)
  - Hierarchical scheduling framework (HSF)
- VxWorks implementation:
  - SRP
  - Overrun/Skipping
    - Common (Overrun/Skipping)
    - Overrun
    - Skipping
- Comparison (Overrun/Skipping)
- Evaluation
- Conclusion
Background

- **Problem:** Resource sharing across subsystems
- **Solution:** Protocols such as Overrun or Skipping
  - Overrun: extend budget to complete critical sec.
  - Skipping: Skip task exec. if budget is too small
- **Both are based on SRP (at both subsystem and task level)**
Preliminaries: SRP

- Synchronization protocol (FPS and EDF)
- SRP notations:
  - Task priority ceiling
  - Resource ceiling
  - System ceiling
- Example: FPS (RM)
  \[ \text{prio}_1 < \text{prio}_2 < \text{prio}_3 \]
Preliminaries: HSF

- Two-level HSF (*OSPERT’08*)
  - Supports FPPS and EDF in both levels
  - Periodic tasks
  - Subsystems are impl. as periodic servers
  - Impl. made in VxWorks

\[
\text{Interruption handler} \quad \text{Global scheduler} \quad \text{Local scheduler} \quad \text{VxWorks scheduler}
\]

\[ \mathcal{O} = \min \left( \begin{array}{c}
\text{TEQ} \\
\text{TEQ} \\
\text{TEQ}
\end{array} \right) - \chi \]

- Budget expiration
- Server release
- Task release

(absolute time)
Implementation: SRP

- **Data-structures**
  - Resource ceiling stack (system ceiling)
  - Blocked task queue (FIFO)
- **Added/modified functionality**
  - Lock and Unlock
  - Modification to local scheduler
Implementation: Common

- **Resources**
  - Globally shared (mapped to local)
  - Locally shared
- **Global/local system ceiling**
- **Data-structures**
  - Resource ceiling stack (system ceiling)
  - Blocked server queue (FIFO)
- **Check sys. ceil. at server release**
Implementation: Overrun

- Data-structures
  - Overrun flag
  - Resource counter
- + Low lock overhead
- + Low amount of data-structures
- - Amount of calls to global scheduler

```c
srp_lock();
locked++;
if (locked > 0)
stamp;  
budget=\infty;
overrun=TRUE;
```

```c
srp_unlock();
if (server_switch || overrun) {
  if (locked > 0)
    timestamp;
    budget=\infty;
    overrun=TRUE;
```
Implementation: Skipping

- Data-structures
  - Resource locking time
  - FIFO queue (for self-blocking)
- + Less calls to global scheduler
- - Overhead in lock and local scheduler
- - Need to save data for critical section length

```c
if (bud_left < cri_sec)
    srp_lock();
    task_suspend();
while(TASK_FIFO_QUEUE)
    move_to_ready_queue();
    timestamp;
```

```c
unlock

srp_unlock();
if (server_switch) {
    global_sched();
}
```
### Comparison

- **Memory complexity:**
  - Skipping higher than Overrun
    - Self blocking queue, resource holding time
  - Skipping needs modification of local scheduler
  - Both modifies the global scheduler
    - Both use server-level SRP
    - Skipping checks self-blocking at release
    - Overrun checks overrun at budget depletion
- **Less time deviation for Skipping**
  - Overrun calls global scheduler more often
- **Overhead**
  - Skipping: More overhead in Lock function
  - Overrun: More overhead in Unlock function
  - It points to that Skipping has lower than Overrun
    - (Details in next slide)
Evaluation

- Experimental setup:
  - Hardware:
    - Robotics controller (ABB)
    - Pentium Pro (200 MHz) processor
    - VxWorks 5.2
  - 8 generated systems (S1 – S8)
  - Systems were recorded 600 time units (tu)
  - Task period: 40-100 tu, server period: 5-20 tu
  - Task utilization per system: ~15%

<table>
<thead>
<tr>
<th>Protocol</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td># calls to lock/unlock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skipping</td>
<td>306</td>
<td>335</td>
<td>248</td>
<td>275</td>
<td>181</td>
<td>224</td>
<td>202</td>
<td>236</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Overrun</td>
<td>304</td>
<td>335</td>
<td>247</td>
<td>275</td>
<td>181</td>
<td>225</td>
<td>203</td>
<td>236</td>
<td>47</td>
<td>13</td>
<td>40</td>
<td>16</td>
<td>36</td>
<td>17</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>
We have implemented 2 synchronization protocols in VxWorks
  • Overrun and Skipping
Both protocols are based on our previous work: HSF (OSPERT’08)
Evaluation results indicate less overhead for Skipping
  • Although memory allocation grows with the nr. of un-nested global resources
Overrun causes more time deviation
Skipping needs modification of local scheduler
Future work:
  • More evaluation (measure entire overhead)
  • Optimizations
Thank you!
Questions?