Graduate Operating Systems

Fall 2019

Paper “DVS”

- Real-Time Systems
- Dynamic Voltage Scaling (DVS, DFS)
- Over-designed systems (peak performance)
- Periodic task model
- Earliest Deadline First (EDF)
- Rate Monotonic Scheduling (RM)
- Schedulability test
Peak vs. Average Performance

DVS Fundamentals

- Processors are based on CMOS technology where dynamic power is the bottleneck

- Dynamic power (due to switching activity)
  - Power depends on $V^2$ and $f$
  - Achievable $f$ depends on $V$

- Energy = $P \times t_{\text{execution}}$
DVS Fundamentals

Periodic Task Model

Task = \{T, C, D\}

jobs (j_1, j_2, j_3, ...)

Deadline = D

Period = T

Computation time

WCET = C

Release Time
**RMS (Rate Monotonic Scheduling)**

Process $P_1$: service time = 20, period = 50, deadline = 50
Process $P_2$: service time = 35, period = 100, deadline = 100

<table>
<thead>
<tr>
<th>Deadlines</th>
<th>$P_1$</th>
<th>$P_1, P_2$</th>
<th>$P_1$</th>
<th>$P_1, P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>130</td>
<td>140</td>
<td>150</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>170</td>
<td>180</td>
<td>190</td>
<td>200</td>
</tr>
</tbody>
</table>

**Missed Deadlines with RMS**

Process $P_1$: service time = 25, period = 50, deadline = 50
Process $P_2$: service time = 35, period = 80, deadline = 80

**RMS is guaranteed to work if**

$N = \text{number of processes, sufficient condition}$

$\mu = \sum_{i=1}^{N} \frac{t_i}{p_i} \leq N\left(\sqrt{2} - 1\right);$

$\lim_{N \to \infty} N\left(\sqrt{2} - 1\right) = \ln 2 \approx 0.693147$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N\left(\sqrt{2} - 1\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.828427</td>
</tr>
<tr>
<td>3</td>
<td>0.779763</td>
</tr>
<tr>
<td>4</td>
<td>0.756828</td>
</tr>
<tr>
<td>5</td>
<td>0.743491</td>
</tr>
<tr>
<td>10</td>
<td>0.717734</td>
</tr>
<tr>
<td>20</td>
<td>0.705298</td>
</tr>
</tbody>
</table>
EDF (Earliest Deadline First)

Process $P_1$: service time = 25, period = 50, deadline = 50
Process $P_2$: service time = 35, period = 80, deadline = 80

Static Voltage Scaling EDF: Motivation

$W_{C_i}$ = worst case computation time @ $F_{\text{max}}$

Holes in the schedule imply:

EDF Test: $\sum (W_{C_i} / p_i) < 1$ at frequency = $F_{\text{max}}$
Static Voltage Scaling EDF

EDF Test:
\[ \sum \left( \frac{w_i}{p_i} \right) < 1 \quad \text{at maximum frequency} = F_{\text{max}} \]

Static-VS EDF Test:
\[ K \times \left( \sum \left( \frac{w_i}{p_i} \right) \right) = 1 \quad \text{at frequency} = F_{\text{max}} / K \]

Static EDF: Example

<table>
<thead>
<tr>
<th>Task</th>
<th>Computing Time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 ms</td>
<td>8 ms</td>
</tr>
<tr>
<td>2</td>
<td>3 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>3</td>
<td>1 ms</td>
<td>14 ms</td>
</tr>
</tbody>
</table>

Available frequencies: 1.00, 0.75

Schedule test for \( \alpha = 1.00 \)
\[ 3/8 + 3/10 + 1/4 \leq 1 \quad \rightarrow \text{Return true} \]

Schedule test for \( \alpha = 0.75 \)
\[ 3/8 + 3/10 + 1/4 \leq 0.75 \quad \rightarrow \text{Return true} \]
What if $C_i < WC_i$?

Next arrival of $T_1$

More holes left unexploited

$K\cdot c_1$ $K\cdot c_2$ $K\cdot c_3$ $K\cdot c_4$

Task $T_1$ completes

Hole of size = $(wc_1 - c_1)$

Slow down all these tasks proportionally

Next arrival of $T_1$
What if $C_i < WC_i$?

Next arrival of T1

CPU Cycles are conserved by slowing down the remaining tasks

Cycle Conserving RT-DVS

- When a task set completes its first release, compare real execution time with worst case specified initially.

- Any idle time in that period can be used to conserve energy.

- Rescale frequency that avoid idle cycles, surplus time is used to run other remaining tasks at lower frequency.
Cycle Conserving EDF: Example

Task set @ (Fmax): T1 = (3,6) and T2 = (6,12)

\[ U = \frac{3}{6} + \frac{6}{12} = 1 @ (Fmax) \]

\[
\begin{align*}
\text{Frequency} && \text{Time} \\
0 && 1 \\
T1 && \\
T2 && 10
\end{align*}
\]

New utilization = \[ \frac{1}{6} + \frac{6}{12} = 0.67 \]

Finding the right "k"

\[
\frac{(1*k)}{6} + \frac{(6*k)}{12} = 1
\]

\[ k = \frac{1}{0.67} \]

New freq = \( (0.67) \cdot F_{\text{max}} \)

Look-Ahead EDF

- Defer as much works as possible and set initially to the minimum possible frequency.
- Hence at later stage if a task uses much less than it worst case, deferred work may never be needed.
- It ensures that there are sufficient cycles available for each task to meet its deadline after reserving cycles for higher priority jobs.
- Best saving of energy.
(a) Plan to defer T3’s execution time until after D1 but by D3

(b) Find u so as to finish T1 by D1 and T2 by D2
Look-ahead EDF Step 3 of 6

(c) $u = 0.75$, T1 finishes earlier, find new $u$ for T2 to finish by D2

Look-ahead EDF Step 4 of 6

(d) $\alpha = 0.5$, T2 finishes earlier, enough time until D1, but EDF is work conserving, launch T3 at $u = 0.5$
Look-ahead EDF Step 5 of 6

(e) Guess for T1 again

Look-ahead EDF Step 6 of 6

(f) $u = 0.5$, every task is dynamically scheduled successfully
Relative Performance

- Energy savings
- Look Ahead EDF
- Cycle Conserving EDF
- Static Voltage Scaling EDF

Implementation

- Scheduler hooking in the kernel

![Diagram](#)

Figure 14: Software architecture for RT-DVS implementation