Graceful Degradation of Low-Criticality Tasks in Multiprocessor Dual-Criticality Systems
Lin Huang, I-Hong Hou, Sachin S.Sapatnekar and Jiang Hu
Outline

- Motivation
- Previous Work
- Variable Precision Scheduling Methods
- Experiment Results
- Conclusion
Outline

- Motivation
- Previous Work
- Variable Precision Scheduling Methods
- Experiment Results
- Conclusion
Hard Real Time Scheduling

• Real time system: job execution has hard deadline

• WCET (worst case execution time)
Mixed Criticality System (MCS)

- Integrate multiple functionalities (tasks with different criticality levels)

- Flight Management System
  - Avionic Flight Control Service (high criticality)
  - Entertainment, Multimedia (low criticality)
Conventional System Model for MCS

- $T = \{\tau_1, \tau_2, \cdots, \tau_n\}$: a set of independent sporadic tasks.
- Task: $(p_i, C_i, L_i)$. $p_i$: period $C_i$: WCET $L_i$: criticality level. high(hi), low(lo)
- For high criticality task, $C_i(HI) > C_i(LO)$

When any high criticality job’s execution time exceeds $C_i(LO)$

High criticality mode, drop all low criticality tasks

Mode switch
Imprecise Mixed Criticality System (IMCS)

- For high criticality task, $C_i(HI) > C_i(LO)$
  
- For low criticality task, $C_i(HI) < C_i(LO)$

When any high criticality job’s execution time exceeds $C_i(LO)$

High criticality mode, low criticality tasks in imprecise mode

Mode switch
Our Work

- Variable Precision Mixed Criticality System (VPMCS)
  Do precision optimization for low criticality tasks

An motivation example of doing precision optimization

<table>
<thead>
<tr>
<th>task</th>
<th>$L_i$</th>
<th>$C_i$(LO)</th>
<th>$C_i$(HI)</th>
<th>$p_i$</th>
<th>$e_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>hi</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>t2</td>
<td>lo</td>
<td>6</td>
<td>3</td>
<td>15</td>
<td>0.1</td>
</tr>
<tr>
<td>t3</td>
<td>lo</td>
<td>4</td>
<td>2</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

No optimization: Average_error=2.55
With our precision optimization: Average_error=0.55
Outline

- Motivation
- Previous Work
- Variable Precision Scheduling Methods
- Experiment Results
- Conclusion
Earliest Deadline First-Virtual Deadline (EDF-VD) Scheduling

➢ Classic EDF-VD scheduling on single processor\cite{1}
  • Each high criticality task has a virtual deadline \((vd \leq d, 0 < x \leq 1)\)
  • Speedup factor=4/3, optimal

\[
vd = r_{i,j} + x \cdot p_i \\
\text{Mode switch} \\
\text{reserve for } C_i(HI) \\
d = r_{i,j} + p_i
\]

EDF-VD vs EDF

EDF-VD has less conservative schedulability condition

<table>
<thead>
<tr>
<th>task</th>
<th>$L_i$</th>
<th>$C_i(Lo)$</th>
<th>$C_i(Hi)$</th>
<th>$p_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>lo</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>t2</td>
<td>hi</td>
<td>1/2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

- Schedulability condition for EDF

$$U_{sum} = U_{lo}^{LO} + U_{hi}^{HI} = \frac{1}{2} + \frac{3}{4} > 1 \text{ not schedulable}$$

Where: $$U_a^b = \sum_{\tau_i \in \tau \land L_i = a} \frac{c_i(b)}{p_i}, \ a \in \{hi, lo\}, \ b \in \{HI, LO\}$$

- Schedulability condition for EDF-VD

$$U_{lo}^{LO} + \min \left( U_{hi}^{HI}, \frac{U_{lo}^{LO}}{1 - U_{hi}^{HI}} \right) = \frac{1}{2} + \min \left( \frac{3}{4}, \frac{1}{2} \right) = 1 \leq 1 \text{ schedulable}$$
Fluid Based Method

- R.M. Pathan “Improving the quality of service for scheduling mixed-criticality systems on multiprocessors”. ECRTS, 2017
- Not directly implementable in practice

Our work: partitioned and global scheduling on multiprocessors
Outline

- Motivation
- Previous Work
- **Variable Precision Scheduling Methods**
- Experiment Results
- Conclusion
Multiprocessor Scheduling

- Partitioned scheduling: no inter-processor migration is allowed
Global Scheduling

- Inter-processor migration is allowed, overhead
Lemma 1: If a task set satisfies the condition $\max(U_{lo}^{LO} + U_{hi}^{LO}, U_{lo}^{HI} + U_{hi}^{HI}) \leq \frac{3}{4}$, it is schedulable by EDF–VD on a single processor.

$$U_a^b = \sum_{\tau_i \in \tau \wedge L_i = a} \frac{c_i(b)}{p_i}, a \in \{hi, lo\}, b \in \{HI, LO\}$$

VPMCS Partitioning with EDF-VD Scheduling

- Speedup factor: \( (8m - 4)/3m \), same as conventional MCS
Enhanced VPMCS Partitioning

Lemma 2: If a task set satisfies the condition \( \frac{U_{hi}^{LO}}{1-U_{lo}^{lo}} \leq \frac{1-(U_{hi}^{HI} + U_{lo}^{HI})}{U_{lo}^{LO} - U_{lo}^{HI}} \), it is schedulable by EDF-VD on a single processor.

Lemma 1 \( \Rightarrow \) Lemma 2
Global Scheduling

- Classic fpEDF method on m multiprocessors\[^3\]
- Optimal w.r.t schedulable utilization

Global Scheduling

- fpEDF-VD (fpEDF and EDF-VD)
  - Speedup factor: $\sqrt{5} + 1$, same as conventional MCS
  - Low criticality task may lose its job once

- Dual virtual-deadline for fpEDF
  - Guarantee no job is abandoned
Fluid Scheduling

- Classic fluid scheduling
  - Optimal w.r.t speedup factor (4/3)

- Deadline Partition-Fair
  - Correctness of the implementation
Offline Precision Optimization

- Formulate the problem as a 0-1 knapsack problem
  - Objective: minimize average error for low criticality tasks in high criticality mode
  - Constraint: total utilization slack

![Diagram with low critical tasks and total utilization slack]
Outline

- Motivation
- Previous Work
- Variable Precision Scheduling Methods
- Experiment results
- Conclusion
Experiment Setup

Simulation

- Random test cases
- The probability of a task being high criticality: 0.5
- Utilization range: [0.05, 0.9]
- Period range: [50, 500]
- Imprecise computing error range: [1, 10]

Linux prototyping

- 1.9GHZ Intel i3 4-processor machine
- Linux 4.10
- Test cases: newton-raphson method, steepest descent method
Experiment Comparison

Partitioned scheduling methods:
- Partition-MC: partitioned scheduling with conventional model (drop low critical tasks)
- Partition-VPMC: our VPMCS Partitioning with EDF-VD Scheduling
- Partition-VPMC-E: our enhanced VPMCS Partitioning

Global scheduling methods:
- fpEDF-VD-MC: fpEDF-VD scheduling with conventional model (drop low critical tasks)
- fpEDF-VD-VPMC: our fpEDF-VD scheduling method
- fpEDF-DVD-VPMC: our dual virtual-deadline for fpEDF

Fluid scheduling methods:
- Fluid-VPMC: fluid method which is theoretically optimal
- VPMC-DP-Fair: real hardware implementation of Fluid-VPMC
Acceptance Ratio versus Utilization

- Our Partition-VPMC-E performs best when considering scheduling overhead.

Acceptance ratio versus utilization on 4 processors

Acceptance ratio versus utilization on 4 processors considering overhead
Mean Error versus Utilization

- IMC: all low critical tasks in imprecise mode

Mean error with standard derivation versus Utilization on 4 processors

Mean error with standard derivation versus Utilization on 8 processors
Linux Prototyping

- Mean error and overhead
  Partitioned method has lowest overhead ratio and smallest mean error.

Overhead ratio versus Utilization

Mean error versus Utilization
Number of context switching

VPMC-DP-Fair has much larger number of context switching than global and partitioned scheduling method.
Outline

- Motivation
- Previous Work
- Variable Precision Scheduling Methods
- Experiment Results
- Conclusion
Conclusion

- Our proposed methods can significantly reduce the error compared to IMCS scheduling.

- The proposed methods could achieve smaller overhead compared to fluid based method.
Thank you