Optimal Selection of Preemption Points to Minimize Preemption Overhead

Marko Bertogna, Orges Xhani, Mauro Marinoni, Francesco Esposito, Giorgio Buttazzo

Scuola Superiore Sant’Anna, Pisa, Italy

ECRTS’11 – July 6-8, 2011, Porto, Portugal
Introduction

Do we really need a fully preemptive scheduler?

• Non-Preemptive Scheduling (NPS) has a large **schedulability overhead**
  – Large blocking times

• Fully Preemptive Scheduling (FPS) has a large **preemption overhead**
  – Cache-related delay, scheduler cost, CPU pipeline flushing, bus contention, …
  – Difficult to predict when a preemption takes place
  – Difficult to find tight upper bounds on the preemption cost
Deferred Preemption Model

- Tasks cooperate to offer suitable preemption points
  - a.k.a. Cooperative scheduling or Fixed Preemption Point (FPP) model
- Each task is divided into a sequence of statically defined non-preemptive chunks
- Preemptions can take place only at chunk’s boundaries (Preemption Points)
How to select preemption points?

• Given a set of tasks \(\{\tau_1, \ldots, \tau_n\}\), each with a set of Potential Preemption Points (PPP)

What is the best way to select which PPP to activate to maximize feasibility?
Task model

- Set $\tau$ of $n$ periodic or sporadic tasks $\tau_i$ with relative deadline $D_i$ and period $T_i$
- Single processor Fixed Priority or EDF
- Task $\tau_i$ composed of a sequence of $N_i$ Basic Blocks (BB) separated by $N_i - 1$ Potential Preemption Points (PPP), each with a corresponding preemption cost $\xi^k_i$

\[ C_i^{NP} = \sum_{k=1}^{N_i} b_i^k \]
Selecting Preemption Points

- An activated PPP becomes an Effective Preemption Point (EPP)
- Assumptions:
  A. Each EPP leads to a preemption
  B. The cache is cold after each context switch
  C. Conditional branches and critical sections are entirely contained inside a basic block (see RTSOPS’11)
Existing results

- Upper bound on the maximum non-preemptive region $Q_i$ allowed for each task $\tau_i$
  - Bertogna, Baruah TII’10: EDF case (tight)
  - Yao et al. RTSJ’11: FP case
  - Tasks ordered for increasing relative deadlines (EDF) or decreasing priorities (FP)
- $Q_i$ can be computed in pseudo-polynomial time both for EDF and FP
  - Same complexity as EDF/FP schedulability tests
  - $Q_i$ is a function of $(C_k, D_k, T_k)$ of lower index tasks ($k < i$)
EPP selection order

• Start with task $\tau_1$ and execute it non-preemptively
  $\Rightarrow C_1 = C_1^{NP}$

• Compute $Q_2$

• Activate the “best EPPs” for $\tau_2$ given the constraint on $Q_2$
  $\Rightarrow C_2 = C_2^{NP} + \sum_{PPP_k \equiv EPP} \xi_k$

• Compute $Q_3$

• …

• Activate the “best EPPs” for $\tau_n$ given the constraint on $Q_n$
  $\Rightarrow C_n = C_n^{NP} + \sum_{PPP_k \equiv EPP} \xi_k$
The selection problem

- For a task $\tau_i$ having a non-preemptive region bound of $Q_i$
- Activate the subset of PPPs of $\tau_i$ such that
  1. Preemption overhead $\rightarrow$ WCET of $\tau_i$ is minimized
  2. The distance between any two consecutive EPPs is $\leq Q_i$

- Example:
Example

• Which PPP to activate?

\[ Q = 8 \]

\[ 2 + 2 + 2 + 1 + 3 + 2 + 3 = 15 \]

• A possibility is to activate only one point (PPP\(_4\)) with a resulting WCET of **15**
Example

• Which PPP to activate?

Q=8

• The optimal strategy is to activate two points (PPP₁ and PPP₅) obtaining a WCET of 14

\[ 2+1+2+2+1+2+1+3 = 14 \]
EPP selection algorithm

- Evaluate each PPP\(_k\) in order, starting from the first one
- For each PPP\(_k\) find the preceding PPPs that are within the \(Q_i\) window (including the overhead)
  - If none, then declare failure
- Among these PPPs find the one that minimizes the WCET of the task until the considered PPP\(_k\)
  - It represents the preceding EPP, if PPP\(_k\) will be activated
- Proceed to the next PPP\(_{k+1}\) ...
- When the end is reached, activate recursively all best preceding EPPs until the start of the task
Example

- For the first 4 PPPs, the best preceding EPP is the beginning of the task.

Q=8

0 \[PPP_1\] \[PPP_2\] \[PPP_3\] \[PPP_4\] \[PPP_5\]

Min WCET 2
Example

- For the first 4 PPPs, the best preceding EPP is the beginning of the task

Q = 8

Min WCET
Example

- For the first 4 PPPs, the best preceding EPP is the beginning of the task.

Q = 8

PPP_1  PPP_2  PPP_3  PPP_4  PPP_5
Min WCET 2  4  6
Example

- For the first 4 PPPs, the best preceding EPP is the beginning of the task

Q=8

PPP

Min WCET
Example

- For the 5\textsuperscript{th} PPP, the beginning of the task cannot be the preceding EPP (2+2+2+1+2 > Q)

\[ Q = 8 \]

What are the possible candidates?
Example

- What are the possible candidates for the preceding EPP of the 5\textsuperscript{th} PPP?

Possible candidates:

- \( \text{PPP}_0 \): \(2 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{NO} \)
- \( \text{PPP}_1 \): \(1 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{OK} \)
Example

• What are the possible candidates for the preceding EPP of the 5th PPP?

Possible candidates:

PPP_0 : 2+2+2+1+2 = 8 ≤ Q → NO
PPP_1 : 1+2+2+1+2 = 8 ≤ Q → OK
PPP_2 : 4+2+1+2 = 9 > Q → NO
Example

- What are the possible candidates for the preceding EPP of the 5th PPP?

Possible candidates:

- $\text{PPP}_0 : 2 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{NO}$
- $\text{PPP}_1 : 1 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{OK}$
- $\text{PPP}_2 : 4 + 2 + 1 + 2 = 9 > Q \rightarrow \text{NO}$
- $\text{PPP}_3 : 3 + 1 + 2 = 6 \leq Q \rightarrow \text{OK}$
Example

• What are the possible candidates for the preceding EPP of the 5th PPP?

Possible candidates:

- \( \text{PPP}_0 : 2 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{NO} \)
- \( \text{PPP}_1 : 1 + 2 + 2 + 1 + 2 = 8 \leq Q \rightarrow \text{OK} \)
- \( \text{PPP}_2 : 4 + 2 + 1 + 2 = 9 > Q \rightarrow \text{NO} \)
- \( \text{PPP}_3 : 3 + 1 + 2 = 6 \leq Q \rightarrow \text{OK} \)
- \( \text{PPP}_4 : 3 + 2 = 5 \leq Q \rightarrow \text{OK} \)
Example

• Which one minimizes the WCET up to PPP\(_5\)?

Potential WCET of [0, PPP\(_5\)]:
- PPP\(_1\) : 2+1+2+2+1+2 = 10
- PPP\(_3\) : 6+3+1+2 = 12
- PPP\(_4\) : 7+3+2 = 12
Example

- Which one minimizes the WCET up to \( \text{PPP}_5 \)?

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>3</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

\[ Q = 8 \]

Potential WCET of \([0, \text{PPP}_5]\):

- \( \text{PPP}_1 : 2 + 1 + 2 + 2 + 1 + 2 = 10 \)
- \( \text{PPP}_3 : 6 + 3 + 1 + 2 = 12 \)
- \( \text{PPP}_4 : 7 + 3 + 2 = 12 \)
Example

- $\text{PPP}_1$ is the best preceding EPP for $\text{PPP}_5$
Example

- Reaching the end of the task

Only two potential candidates:

PPP₄: 7+3+2+3=15
PPP₅: 10+1+3=14
Example

- PPP$_5$ is the best preceding EPP of the End point, i.e., it is the last one to be activated.
Example

- $\text{PPP}_1$ is the best preceding EPP of $\text{PPP}_5$
Example

- The start of the task is the best preceding EPP of PPP$_1$
Example

- The EPP selection that minimizes the WCET of the task is given by \( \text{PPP}_1 - \text{PPP}_5 \)

  ![Diagram showing the EPP selection with WCET of 14]

- The overall WCET is 14
Considerations

• Each $\text{PPP}_k$ is evaluated at most once
• The set of preceding candidates to check is at most $Q$
  – Complexity $O(N \times Q)$
• A smarter implementation maintains the set of preceding candidates in an ordered queue and at each step ($\text{PPP}_k$):
  – Removes from head PPPs outside the $Q_i$ window
  – Removes from tail PPPs causing a WCET worse than $\text{PPP}_k$
  – Inserts $\text{PPP}_k$ in the tail
• Complexity linear in the number of PPPs: $O(N)$
• Memory requirement: $O(N + Q)$
Experiments

• Random task set generator resembling real workloads (overhead and WCET distributions)
  – Task utilizations uniformly distributed
  – Number of BBs in [20,200]
• Fixed Priority (Rate Monotonic)
• Five scheduling policies have been compared:
  – Limited-Preemption (LiP-opt) using the proposed algorithm
  – Limited-Preemption (LiP-naïve) activating always the last PPP within the $Q_i$ window, starting from the beginning
  – Non Preemptive (NoP)
  – Fully preemptive with preemption cost (FuP)
  – Fully preemptive without preemption cost (FuP-nocost)
Simulations (varying $U_{\text{tot}}$)

- FuP
- LiP-opt
- LiP-naïve
- FuP-nocost
- NoP

$n = 7$
Simulations (varying $U_{\text{tot}}$)

At large utilizations, LiP-opt schedules 70% more task sets than FuP and 15% more than LiP-naïve.
Simulations (varying $n$)

$U_{tot} = 0.9$

LiP-opt always very close to FuP-nocost

Increasing $n$
- Tasks have more slack
- Larger blocking tolerances

© 2011 Scuola Superiore Sant’Anna
Simulations (varying $\xi_{\text{max}}$)

$U = 0.95$

$n = 7$

close to the schedulability border

LiP-opt constantly better than LiP-naïve

© 2011 Scuola Superiore Sant’Anna
Conclusions

• Algorithm for the optimal selection of preemption points considering a variable preemption overhead

• Significantly increases number of feasible task sets (especially at large utilizations)

• Increased predictability, simplifying timing analysis and improving WCET estimations

• Applicable to both EDF and Fixed Priority

• Complexity comparable to classic feasibility analysis

• Future work
  – Extend the analysis to conditional branches
  – What about the multiprocessor case?
thank you!

marko.bertogna@sssup.it
Synthetic tasks
Simulation (varying $\xi$'s variance)

Increasing the CRPD variance, all methods suffer a larger overhead.

But LiP-opt has better chances of finding a small $\xi$ value before $Q$ time-units.

$U = 0.95$
$n = 7$
Problem

• Tasks include lots of conditional branches!
• A conditional branch might span a big share of the task code:

• In these cases the deferred preemption model shows its limits (degenerates to non-preemptive)
Open problem

• How to select which EPP to activate for a (DAG-based) task structure including conditional branches?
Naïve idea

- Treating each branch separately does not work
- Example:
Naïve idea

- Applying the linear method to the upper branch activates the red EPPs

\[ Q = 8 \]
Naïve idea

• Applying the linear method to the upper branch activates the **red EPPs**

![Diagram](image)

Q = 8

• Applying the linear method to the lower branch activates the **green EPPs**

© 2011 Scuola Superiore Sant'Anna
Naïve idea

- The optimal solution uses both red and green EPPs
Open problem(s)

• Algorithm that optimally selects which EPP to activate in a task structure including conditional branches, such that
  – Distance between two consecutive EPPs is at most Q
  – Overall WCET is minimized

• How to consider loops?

• Which task structures have to be considered for strictly structured programming languages
  – i.e., those including only (see Böhm-Jacopini theorem)
    • Sequential execution
    • Conditional execution
    • Iteration