Non-Work-Conserving Scheduling of Non-Preemptive Hard Real-Time Tasks Based on Fixed Priorities

Mitra Nasri  Gerhard Fohler

Chair of Real-Time Systems
Technische Universität Kaiserslautern
Germany

Nov. 2015
Why Non-preemptive Scheduling?

It is inevitable in many systems
- Because of design or architecture
- CAN networks
- GPU

More timing predictability
- Better estimation of the worst-case execution time (WCET)
- More predictability in cache behavior

Preemption is expensive
- Context switch overheads
- Destructing cache affinity
- Shared resources (need mutual exclusion)

Application’s Desire
- Control applications are affected by I/O delay (preemption length)
Why Non-preemptive Scheduling is Hard?

Schedulable by npEDF

Not schedulable by npEDF

Schedulable by a non-work conserving scheduling algorithm
Without considering idle times in the schedule, we cannot find a solution.

- No known optimal scheduling policy
- No known strategy for idle time insertion

Needs an exhaustive search over all jobs and all possible values/locations of idle times

Preemptive, D=T  Preemptive, D<T  Non-Preemptive
State of the Art

Non-Preemptive Scheduling

Schedulability Test (npEDF, npRM, FP)

Non-Work Conserving Algorithms (for Harmonic Tasks)

No deadline miss if
- Constant integer period ratio $K \geq 3$

No deadline miss if tasks have
- Integer period ratio $K_i \geq 3$
- Constant period ratio $K = 2$

Precautious-RM
- Integer period ratio $K_i \geq 3$
- Constant period ratio $K = 2$
- Integer period ratio $k_i \geq 1$ and enough vacant intervals

Heuristic Algorithms (gEDF, cEDF)
Ekelin 2006, Li 2007

Period Ratio
$k_i = \frac{T_i}{T_i - 1}$

Exponential complexity
A Closer Look at the Idea of Precautious-RM
Precautious-RM Idea: An Efficient Decision

\[ c_i \leq 2(T_1 - c_1) \]

Feasible by a non-work conserving scheduling algorithm

Necessary Condition

Missed

\[ t=4 \]
Precautious-RM Idea: An Efficient Decision

Feasible by a non-work conserving scheduling algorithm

\[ c_i \leq 2(T_1 - c_1) \]

Necessary Condition
How Precautious-RM Works

- **Rule 1**: Use RM priorities (shorter periods have higher priority)
- **Rule 2**: Schedule a task only if it will not cause a deadline miss for the next instance of $\tau_1$, otherwise, insert an idle interval until the next release of $\tau_1$

- Online algorithm (online decisions)
- $O(1)$ computational complexity

- **Limitations of the existing schedulability test**
  - It is only for harmonic tasks
  - It is pessimistic
    - it assumes each task has $c_i = 2(T_1 - c_1)$
$K = \max\{k_i\}$, where $k_i$ is the individual period ratio in the task set
Simple Idea, Interesting Results

• How good is this idea?

It is a big progress!
Contributions of This Work

• Extending the schedulability of Precautious-RM to Loose Harmonic tasks
  • Loose harmonic tasks:

$$\frac{T_i}{T_1} \in \mathbb{N}$$

- Tasks are assigned to priority groups and they are only allowed to be scheduled if the head of the group is scheduled

• Improving the schedulability by priority grouping

• Presenting a priority grouping algorithm which theoretically dominates schedulability test for Precautious-RM
  • The wise fit!
Calculating the vacant intervals

\[ v_i = \begin{cases} 
[k_i]v_{i-1} - 0.5, & c_i \leq T_1 - c_1 \text{ and } 1 < i \leq n \\
[k_i]v_{i-1} - 1, & c_i > T_1 - c_1 \text{ and } 1 < i \leq n 
\end{cases} \]

\[ v_1 = 0.5 \]

The schedulability test

\[ v_i \geq 0.5 \quad \forall i, \; 1 < i < n \]

\[ v_n \geq 0 \]
Next Improvement: Priority Grouping

- **Priority grouping**
  - It helps to improve the schedulability by wasting less vacant intervals

- **The restriction**

\[
2(T_1 - c_1) = 18
\]

\[
\begin{align*}
\{\tau_1\} & \quad c_2 + c_4 = 18 \\
\{\tau_2 + \tau_4\} & \quad c_3 + c_5 = 18
\end{align*}
\]

**The solution**

Permit the tail tasks to be executed only if the head task is scheduled in the same vacant interval.
Schedulability of the Priority Groups

- Each group has $C_i \leq 2(T_1-c_1)$, thus we can use Precautious-RM schedulability.
  - We need $V_i \geq 0.5$

**Easy proof for head tasks**

How can we guarantee schedulability of the **tail tasks**?
• How to guarantee the schedulability of the tail tasks?

**WCRT analysis?**

Maximum release offset

\[ R_{2,j}^i, WCRT_1^i \]

Period of each tail ≥ 2×\( T_{head} \)

\[ R_{1,j}^i + WCRT_1^i + \sum_{x=2}^{X_i} c_x^i \leq T_j^i \]
The Wise-Fit Approach

- **Wise-Fit**
  - Picks the first ungrouped task
  - Finds the first group with enough capacity (based on the execution times)
  - Verifies the schedulability of the existing groups if this task is added to the group
  - If there is no such group, it creates a new group

A full group is the one with \( C_i > (T_1 - c_1) \)

\[
v_i = \begin{cases} 
[k_i]v_{i-1} - 0.5, & c_i \leq T_1 - c_1 \\
[k_i]v_{i-1} - 1, & c_i > T_1 - c_1 
\end{cases} \text{ and } 1 < i \leq n
\]

\[
v_1 = 0.5.
\]

Sometimes it is better to leave a group half-empty instead of making it totally full.

**Examples:**

- \( T = 5 \)
  - \( T_1 - c_1 = 10 \)
  - Used capacity = 7
  - Remained capacity = 13

- \( T = 30 \)
  - \( \tau_6 \) and \( \tau_3 \)
  - Used capacity = 5
  - Remained capacity = 15

- \( T = 45 \)
  - \( \tau_1 \)
  - Used capacity = 7
  - Remained capacity = 13
Evaluations
The Effect of Period Ratio

- $K = \max\{k_i\}$, where $k_i$ is the individual period ratio in the task set
- Tasks with random execution time smaller than $2(T_1-c_1)$
- Not necessarily feasible task sets
- 7 tasks
The Effect of Other Parameters

- \( k_i \) is selected randomly from \{1, 2, 3, 4\}
- \( c_i \leq \sigma \times 2(T_1 - c_1) \)
- Not necessarily feasible task sets
- 10 tasks

- \( k_i \) is selected randomly from \{1, 2, 3, 4\}
- \( c_i \leq 2(T_1 - c_1) \)
- Not necessarily feasible task sets
Conclusions

Non-Preemptive Scheduling
- The only applicable solution in many systems
- Reduced overheads by avoiding preemptions
- More timing predictability for the tasks
- Necessary for many applications

A Non-Work Conserving Solution (based on Precautious-RM)
- $O(1)$ online complexity
- $O(n)$ schedulability test
- High schedulability ratio

New Schedulability Test for Non-Harmonic Tasks

Improving the Schedulability by Priority Grouping

Future Work

Extending it to multiprocessor systems
- Both partitioning and global approaches

Applying Precautious-RM in Different Systems
- In CAN networks
- In real-time GPU applications

Schedulability Analysis in General Case
- $D < T$
- Periodic tasks with no condition on periods
We broke an old wall

Thank you
An Example

• First-Fit

\[ \tau_1 = (3, 10) \]
\[ \tau_2 = (2, 20) \]
\[ \tau_3 = (1, 40) \]
\[ \tau_4 = (5, 70) \]
\[ \tau_5 = (12, 130) \]
\[ \tau_6 = (7, 330) \]

\( T = 10 \)

\( T_1 - c_1 = 7 \)

Used = 3

\( T = 20 \)

Used = 5

\( T = 70 \)

Used = 12

\( T = 130 \)

Used = 7

\( T = 330 \)

\( T_1 - c_1 = 7 \)

Rejected by the schedulability test

• Wise-Fit

\( T = 10 \)

\( T_1 - c_1 = 7 \)

Used = 7

\( T = 20 \)

Used = 1

\( T = 40 \)

Used = 12

\( T = 130 \)

Used = 7

\( T = 330 \)

Accepted by the test: Schedulable