Integrating Cache-Related Pre-emption Delays into Analysis of Fixed-Priority Scheduling with Pre-emption Thresholds

Reinder J. Bril, Sebastian Altmeyer, Martijn M.H.P. v.d. Heuvel, Robert I. Davis, and Moris Behnam

RTSS-2014 (Rome, Italy)
Background and motivation

• **Why caches?**
  - bridges speed-gap between processors and memory;
  - reduces conflicts on the system bus;
  - give rise to additional delays upon pre-emption...

• **Why Fixed-Priority Pre-emptive Scheduling (FPPS)?**
  - described in standards, e.g. OSEK/AUTOSAR [1, 2];
  - supported by most COTS RTOS;
  - de facto standard in industry;
  - arbitrary pre-emption.

• **SotA:** integrated CRPD into analysis for FPPS [3].

Background and motivation

- **Basic notions for CRPD [26, 4]:**
  - **evicting cache blocks (ECBs):** any block that can be accessed by a task;
  - **useful cache blocks (UCBs):** blocks that may be re-used without self-eviction;
  - **block reload time (BRT):** time needed to load a block in cache;
  - **number of cache sets (N):** size of the cache;
  - **cache utilization of task τ_i (U_i^C):** |ECB_i|/N;
  - **total cache utilization (U^C):** Σ_i |ECB_i|/N.

Background and motivation

- **SotA**: integrated CRPD into analysis for FPPS [3].

  **Experiment**:
  - \( n = 10 \) tasks;
  - \( T_i (= D_i) \in [10,1000] \) ms;
  - \( U_i \) task util. by UUnifast [11];
  - 1000 systems per \( U \) set util.

  **Cache related**:
  - \( N = 512 \) cache sets, \( BRT = 8 \mu s \);
  - \( U^C = 4 \) total cache utilization (total # of ECBs is 512 x 4);
  - \( U^C_i \) task cache util. by UUnifast;
  - 40% of ECBs are UCBs.

CRPD has a **significant** impact on schedulability!

Background and motivation

• **SotA: integrated CRPD into analysis for FPPS [3].**

![Graph showing weighted schedulability ratio vs. block reload time]

**Experiment:**
- $n = 10$ tasks;
- $T_i (= D_i) \in [10,1000] \text{ ms}$;
- $U_i$ task util. by UUnifast [11];
- 1000 systems per $U$ set util.

**Cache related:**
- $N = 512$ cache sets;
- $U^C = 4$ total cache utilization (total # of ECBs is $512 \times 4$);
- $U^C_i$ task cache util. by UUnifast;
- 40% of ECBs are UCBs.

**CRPD has a significant impact on schedulability!**

Background and motivation

• Problem: *Arbitrary* pre-emption of FPPS

• Challenge:
  • reduce CRPD for FPS by *limiting arbitrary* pre-emption

• Approach:
  • use a “practical” limited-pre-emptive FPS
  • forth-coming de facto standard in industry?
Background and motivation

• Orthogonal limited pre-emptive FPS approaches
  1. non-pre-emptive regions (NPRs):
     a) static, each job is a sequence of NPRs [16];
     b) dynamic or floating, upon another activation [34];
  2. pre-emption thresholds [33].

• Evaluation
  • theoretical ⇒ static NPRs (FPDS) [18];
  • practical ⇒ pre-emption thresholds (FPTS).

[34] G. Yao, G. Buttazzo, and M. Bertogna, RTCSA, 2009.
Background and motivation

• Why FPTS?
  • pre-emption thresholds assignment at integration time
    – (NPR configuration problematic for FPDS);
  • supported by
    – code-wrappers;
    – OSEK/AUTOSAR (partially via internal resources);
    – (dedicated RTOS-support required by floating NPRs);
  • limits arbitrary pre-emption
    – (unlike floating NPRs).
  • Evolutionary successor of FPPS as defacto standard!

RTTS-2014 (Rome, Italy)
Goals / contributions

1. Integrate CRPD in Analysis of FPTS for *sporadic* tasks with *arbitrary* deadlines;

2. Conceive an OTA (Optimal Threshold Assignment) algorithm for FPTS with CRPD;

3. Evaluate schedulability ratio of FPPS with CRPD and FPTS with CRPD for *constrained* deadlines.
Recap: CRPD in Analysis for FPPS [3]

- Worst-case response time $R_i$ of task $\tau_i$:

$$R_i = C_i + \sum_{\forall j \in \text{hp}(\pi_i)} \left( E_j(R_i) \cdot C_j + \gamma_{i,j}(R_i) \right)$$

- where
  - $E_j(R_i) = \left\lfloor \frac{R_i}{T_j} \right\rfloor$
  - $\gamma_{i,j}(R_i)$: CRPD of task $\tau_i$ due to task $\tau_j$.

- Definition of $\gamma_{i,j}(R_i)$ is CRPD-approach specific.

---


RTTS-2014 (Rome, Italy)
Recap: CRPD in Analysis for FPPS

- Approaches for $\gamma_{i,j}(R_i)$:
  - *pre-empting* tasks only [17]:
    - based on ECBs only;
  - *pre-empted tasks* only [26]:
    - based on UCBs only;
  - *pre-empting and pre-empted* tasks [32, 3]:
    - based on ECBs and UCBs.

- For details, see [3].

Recap: CRPD in Analysis for FPPS

<table>
<thead>
<tr>
<th></th>
<th>FPPS ($D_i \leq T_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>[0, $R_i$)</td>
</tr>
<tr>
<td>Execute</td>
<td>hep($\pi_i$)</td>
</tr>
<tr>
<td>Affected by $\tau_j$</td>
<td>lp($\pi_j$) $\cap$ hep($\pi_i$)</td>
</tr>
</tbody>
</table>
| Number of jobs   | \[
\begin{cases}
E_h(R_i) & \text{if } h \in \text{hep}(\pi_i) \\
0 & \text{otherwise}
\end{cases}
\] |

Basics to determine $\gamma_{i,j}(R_i)$ for FPPS.

- **When we focus on:**
  - *pre-empting* tasks, we need “Affected by”;
  - *pre-empted* tasks, we also need “Number of jobs”.

RTTS-2014 (Rome, Italy)
Recap: Analysis for FPTS

• Distinguishing characteristic of FPTS:
  • pre-emption threshold $\theta_i$ for task $\tau_i$ with $\pi_i \leq \theta_i \leq \pi_1$;

• Compared to FPPS, a job of task $\tau_i$ may now be
  • *blocked* by a task $\tau_b$ with a *lower* priority and a *higher* pre-emption threshold than $\pi_i$: $b \in \text{lp}(\pi_i) \setminus \text{lt}(\pi_i)$;
  • *delayed* by its previous job, even for *constrained* deadlines;
  • *pre-empted* by a task $\tau_j$ with a priority *higher* than the pre-emption threshold $\theta_i$ of $\tau_i$: $j \in \text{hp}(\theta_i)$. 

Recap: Analysis for FPTS

- Distinguishing characteristic of FPTS:
  - pre-emption threshold $\theta_i$ for task $\tau_i$ with $\pi_i \leq \theta_i \leq \pi_1$;
- To determine $R_i$ of task $\tau_i$ [30, 25]:
  - determine worst-case blocking $B_i$;
  - consider all jobs of $\tau_i$ in a level-i active period $L_i$;
    - determine worst-case start time $S_{i,k}$ of job $k$;
    - optional blocking + delays of tasks $\tau_j$ with $\pi_j > \pi_i$;
    - determine worst-case finalization time $F_{i,k}$ of job $k$;
    - interference in $[S_{i,k}, F_{i,k})$ by tasks $\tau_j$ with $\pi_j > \theta_i$;
  - $R_i = \max_k (F_{i,k} - kT_i)$.

A job can only be pre-empted during its hold-time.

Worst-case hold-time $H_i$ and worst-case response $R_i$ are equal ($H_i = R_i$) for:

- FPPS, independent tasks and constrained deadlines [12].

Hold times versus response times

<table>
<thead>
<tr>
<th>$T$</th>
<th>$D$</th>
<th>$C$</th>
<th>$\pi = \theta$</th>
<th>$R$</th>
<th>$H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>7</td>
<td>9</td>
<td>4.2</td>
<td>1</td>
<td>8.6</td>
</tr>
</tbody>
</table>

A previous job may delay the current job.

Worst-case hold-time $H_i$ can be *smaller* than $R_i$ for:
- FPPS, *independent* tasks and *arbitrary* deadlines [13].

## Hold times versus response times

<table>
<thead>
<tr>
<th></th>
<th>$T = D$</th>
<th>$C$</th>
<th>$\pi$</th>
<th>$\theta$</th>
<th>$R$</th>
<th>$H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure illustrates $H_i$ and $R_i$ for $\tau_3$.

Task $\tau_3$ can be:
- *blocked* by task $\tau_4$;
- *delayed* (without pre-emption) by task $\tau_2$;
- *pre-empted* only once by task $\tau_1$.

Task $\tau_2$ cannot be pre-empted by $\tau_1$. 

---

RTTS-2014 (Rome, Italy)
Hold times versus response times

<table>
<thead>
<tr>
<th></th>
<th>FPPS ((D_i \leq T'_i))</th>
<th>FPTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>([0, R_i))</td>
<td>([0, H_i))</td>
</tr>
<tr>
<td>Execute</td>
<td>hep((\pi_i))</td>
<td>({i} \cup hp(\theta_i))</td>
</tr>
<tr>
<td>Affected by (\tau_j)</td>
<td>(lp(\pi_j) \cap hep(\pi_i))</td>
<td>(lt(\pi_j) \cap ({i} \cup hp(\theta_i)))</td>
</tr>
</tbody>
</table>
| Number of jobs   | \[
\begin{cases}
  E_h(R_i) & \text{if } h \in hep(\pi_i) \\
  0 & \text{otherwise}
\end{cases}
\]
|                  | \[
\begin{cases}
  E_h(H_i) & \text{if } h \in hp(\theta_i) \\
  1 & \text{if } i \\
  0 & \text{otherwise}
\end{cases}
\]

Notes:
- exactly 1 job of \(\tau_i\) in \(R_i\) for FPPS \((D_i \leq T'_i)\);
- exactly 1 job of \(\tau_i\) in \(H_i\) for FPTS.
# Level-\(i\) active period, \(S_{i,k}\) and \(F_{i,k}\)

<table>
<thead>
<tr>
<th>Interval</th>
<th>FPTS</th>
<th>FPTS</th>
<th>FPTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>([0, L_i))</td>
<td>([0, S_{i,k}))</td>
<td>([0, F_{i,k}))</td>
</tr>
<tr>
<td>Execute</td>
<td>({b} \cup \text{hep}(\pi_i))</td>
<td>See ([0, L_i))</td>
<td>See ([0, L_i))</td>
</tr>
<tr>
<td>Affected by (\tau_j)</td>
<td>(\text{lt}(\pi_j) \cap ({b} \cup \text{hep}(\pi_i)))</td>
<td>See ([0, L_i))</td>
<td>See ([0, L_i))</td>
</tr>
</tbody>
</table>
| Number of jobs    | \[
E_h(L_i) \quad \text{if} \quad h \in \text{hep}(\pi_i) \\
1 \quad \text{if} \quad b \\
0 \quad \text{otherwise}
\] | \[
E_h(S_{i,k}) \quad \text{if} \quad h \in \text{hp}(\pi_i) \\
1 \quad \text{if} \quad b \\
0 \quad \text{otherwise}
\] | \[
E_h(F_{i,k}) \quad \text{if} \quad h \in \text{hp}(\theta_i) \\
E_h(S_{i,k}) \quad \text{if} \quad h \in \text{hp}(\pi_i) \setminus \text{hp}(\theta_i) \\
1 \quad \text{if} \quad b \\
0 \quad \text{otherwise}
\] |

Notes:
- exactly 1 job of \(\tau_b\) in \(L_i, S_{i,k}\) and \(F_{i,k}\)
- exactly \(k\) jobs of \(\tau_i\) in \(S_{i,k}\)
- exactly \(k+1\) jobs of \(\tau_i\) in \(F_{i,k}\).
CRPD in FPTS

• Given this table,
  • determining CRPD for FPTS is mainly math...
    (for each of the approaches described in [3])
  • see (and enjoy) paper!

An OTA algorithm for FPTS with CRPD

• Recap: OTA alg. for FPTS without CRPD [33, 31]:
  • assumptions:
    1. priorities are given;
    2. schedulability test for $\tau_i$ independent of pre-emption thresholds of tasks with a higher priority than $\pi_i$;
  • alg. traverses priorities in **ascending** order.
• Problem: 2 doesn’t hold for FPTS and CRPD
  • pre-emption of $\tau_h$ by $\tau_j$ with $\pi_i < \pi_h < \pi_j$ depends on $\theta_h$.
• Solution (see paper):
  • traverse priorities in **descending** order.

Comparative evaluation – I

• Ratio of schedulable task sets (example revisited)

Experiment:
• $n = 10$ tasks;
• $T_i = D_i \in [10,1000]$ ms;
• $U_i$ task util. by UUnifast [11];
• $1000$ systems per $U$ set util.

Cache related:
• $N = 512$ cache sets, $BRT = 8\mu$s;
• $U^C = 4$ total cache utilization (total # of ECBs is $512 \times 4$);
• $U^C_i$ task cache util. by UUnifast;
• $40\%$ of ECBs are UCBs.

FPTS significantly improves schedulability ratio!

Comparative evaluation – II

- **Weighted schedulability ratio [7] versus BRT**

  **Experiment:**
  - $n = 10$ tasks;
  - $T_i (= D_i) \in [10,1000]$ ms;
  - $U_i$ task util. by UUnifast [11];
  - 1000 systems per $U$ set util.

  **Cache related:**
  - $N = 512$ cache sets;
  - $U^C = 4$ total cache utilization (total # of ECBs is 512 x 4);
  - $U^C_i$ task cache util. by UUnifast;
  - 40% of ECBs are UCBs.

  **FPTS remains better than FPNS!**

Contributions

1. Integrated CRPD into analysis for FPTS
   - for sporadic tasks and arbitrary deadlines, covering the most effective approaches known to date;
   - specializes to existing analysis for FPPS [3] for sporadic tasks with constrained deadlines.

2. Optimal Threshold Assignment (OTA) algorithm
   - minimizes effects of CRPD.

3. Comparative evaluations for FPTS and FPPS
   - significant extension of performance advantage;
   - performance advantage of FPTS over FPNS (!)

Conclusion and Future work

• **Conclusion:**
  - FPTS as forth-coming defacto standard in industry
    - (partially) supported by OSEK and AUTOSAR

• **Future work:**
  1. CRPD reduction via layout of tasks in memory
  2. Optimal Priority and Pre-emption Threshold Assignment (OPTA) algorithm
    - computational tractable method...