Hard Real-Time Scheduling with Cache-Related Preemption Delays

Guillaume PHAVORIN   Pascal RICHARD

October 15th, 2015

Seminar: http://labex-digicosme.fr/GT+OVSTR

Outline

Part I - Survey

- Background
  - Caching problems
  - Cache-related scheduling problem
- Cache-Related Scheduling Problems: State-of-The-Art
  - Main approaches
  - Classification and open issues

Part II - Several contributions

- CRPD-aware scheduling
  - Adding CRPD in the task model and analysis (optimality, sustainability, computational complexity)
  - Offline method and numerical results
- Cache-aware Scheduling
  - Adding cache state information in the task model
  - Computational complexity analysis

Context

Two major resources, that must be fully utilized

- computation: processors are fast
- memory: main memory is very slow and caches are very small

➢ Cache-Related Preemption Delays (CRPD) can represent up to 44% of the overall Worst-Case Execution Time (WCET) [1]–[4].

How to define efficient scheduling techniques for these resources...

- Where is the bottleneck (processors, caches, bus contention,...combination of them)?
- What is the impact of caches on scheduling problems?

Which Cache metrics for Real-Time Systems?

Hard Real-Time Systems

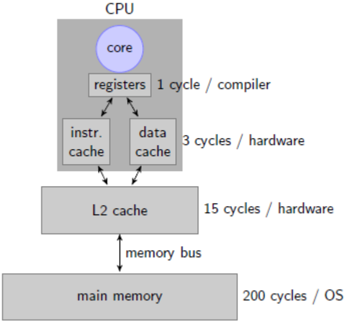
- CRPD bounds and Enhanced Schedulability Analysis (Jan Reineke's talk).
- Using CRPD and cache information to take scheduling decisions (this talk)

Soft-real Time Systems (not covered hereafter)

- Working Set Size: amount of cache lines accessed by a task (memory footprint)
- Stack Distance: number of different cache-lines accessed between two consecutive accesses to the same reference
- ...(see Hennessy and Patterson [5] for additional metrics)
- ...(see Calendrino and Anderson [6] for soft real-time syst.)

Caching Problems

- Current processors have cache memories



- Caches are resources with their own management unit (Replacement algorithm)
- cost(**cache miss**) $\gg$ cost(**cache hit**)
- Optimized for a single stream of sequential block requests

Cache replacement policy

Clairvoyant optimal algorithm [7]

(Belady's 1966 algorithm) Evict the block that will be used the furthest in the future

Online algorithms (competitive analysis [8])

➢ Adversary: optimal sequence for a size-k cache causes m misses.
➢ No (deterministic) on-line algorithm for a size-k cache can have a competitive ratio better than k (i.e., $k \times m$ cache evictions)

Basic observations

- Caches are mainly designed to reduce *intra*-task cache misses
- Periodic real-time tasks generate numerous *inter*-task misses (due to preemptions)

Cache-Related Preemption Delays (CRPD)

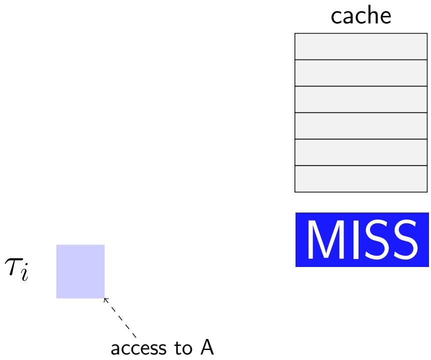
CRPD

Additional reloads because of cache evictions due to **preempting jobs**

Cache-Related Preemption Delays (CRPD)

CRPD

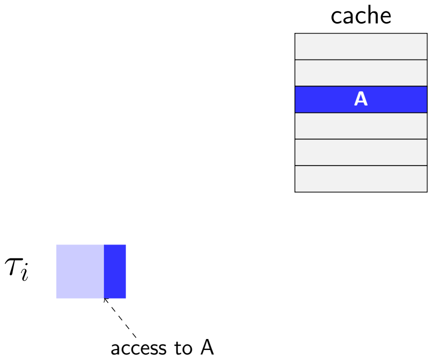
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

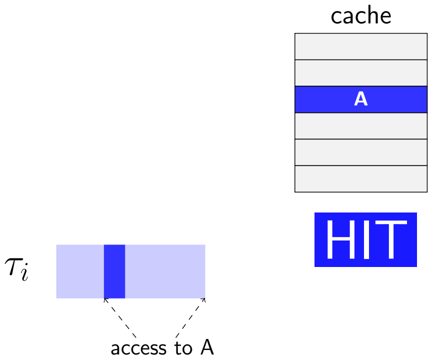
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

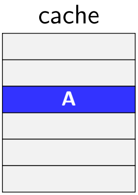
Additional reloads because of cache evictions due to **preempting jobs**

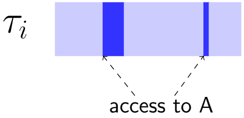


Cache-Related Preemption Delays (CRPD)

CRPD

Additional reloads because of cache evictions due to **preempting jobs**

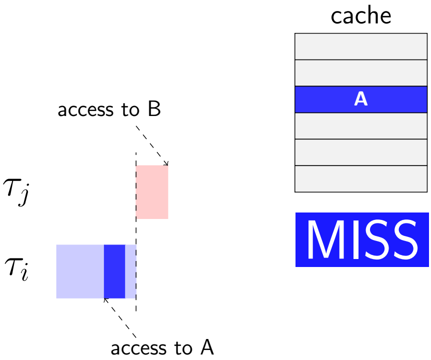




Cache-Related Preemption Delays (CRPD)

CRPD

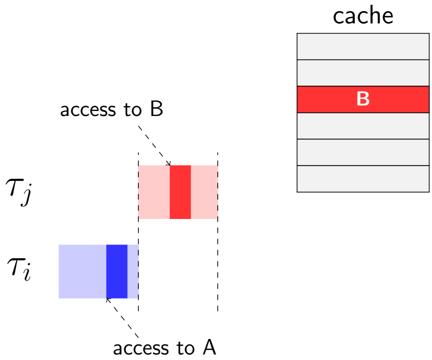
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

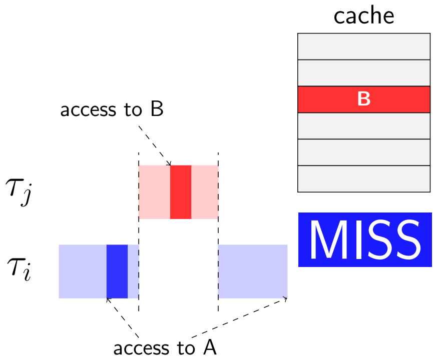
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

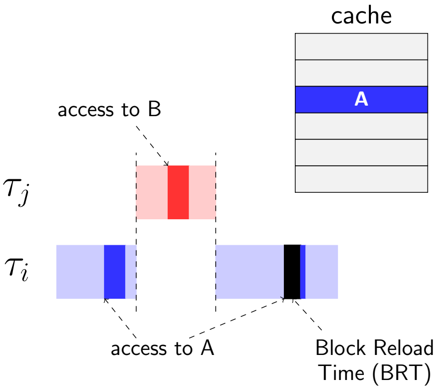
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

Additional reloads because of cache evictions due to **preempting jobs**

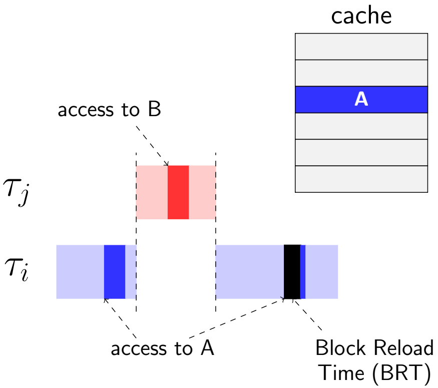


Cache-Related Preemption Delays (CRPD)

CRPD

Additional reloads because of cache evictions due to **preempting jobs**

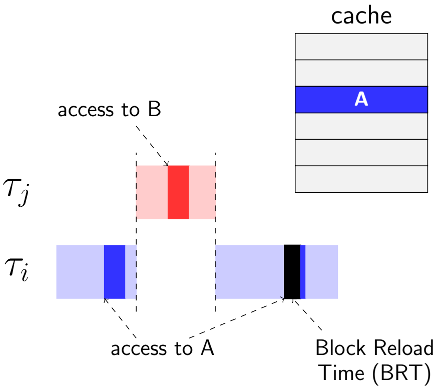
➢ ↗ System utilization (Up to 44% [1]–[4])



Cache-Related Preemption Delays (CRPD)

CRPD

Additional reloads because of cache evictions due to **preempting jobs**

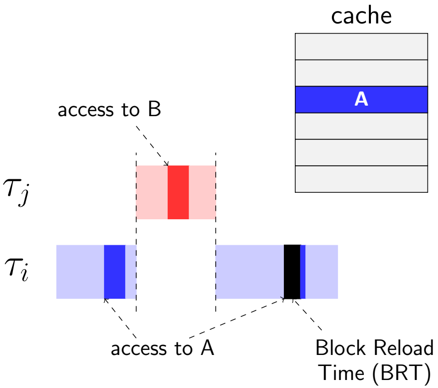


- ↗ System utilization (Up to 44% [1]–[4])
- predictability?

Cache-Related Preemption Delays (CRPD)

CRPD

Additional reloads because of cache evictions due to **preempting jobs**



- ↗ System utilization (Up to 44% [1]–[4])
- predictability?
- schedulability?

Scheduling Theory: Basic Task models

Periodic (Sporadic) task

- Every task τ_i releases an infinite set of **jobs** $(\tau_{ij}, j = 1..\infty)$
- Every job as the same WCET C_i
- jobs are periodically released every T_i units of time (minimum interarrival time in the sporadic case)
- jobs are subjected to a deadline D_i units of time after their releases.

Deadlines

- Implicit: $D_i = T_i$
- Constrained: $D_i \leq T_i$
- Arbitrary: otherwise.

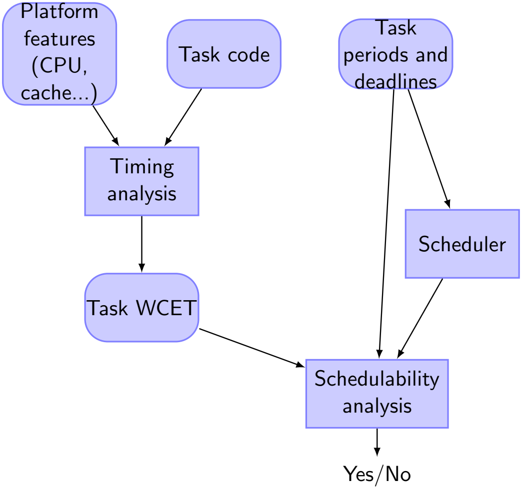
Outline

State-of-the-art (Tentative)

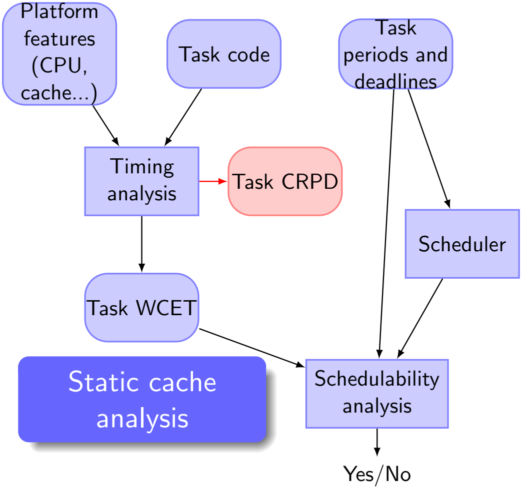
Main Approaches [9]

- Bounding CRPD and introducing it in schedulability tests
  ➢ WCET community
- Cache management: partitioning/locking cache lines
  ➢ Platform community
- Controlling preemptions points: static and dynamic approaches
  ➢ Scheduling Community

Bounding the CRPD



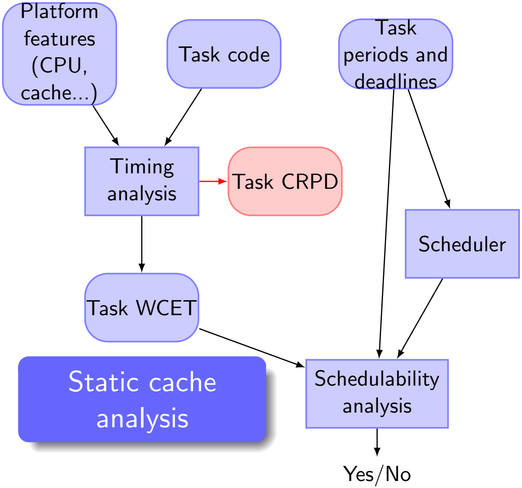
Bounding the CRPD



- preempted task
  ↪ Useful Cache Blocks (UCBs)

- *Lee et al. (1997, [10])*

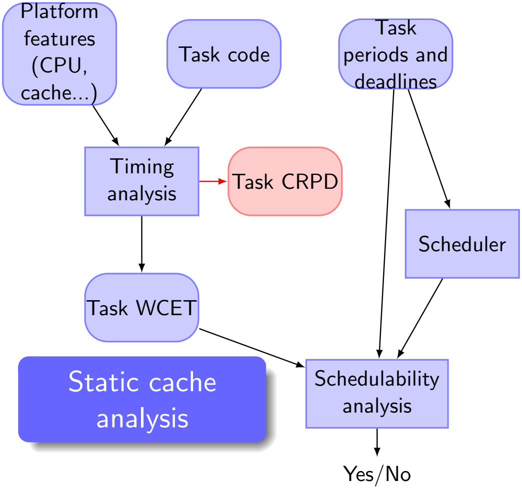
Bounding the CRPD



- preempted task
  - ↪ Useful Cache Blocks (UCBs)
- preempting task
  - ↪ Evicting Cache Blocks (ECBs)

- *Busquets-Mataix et al.(1996, [11])*

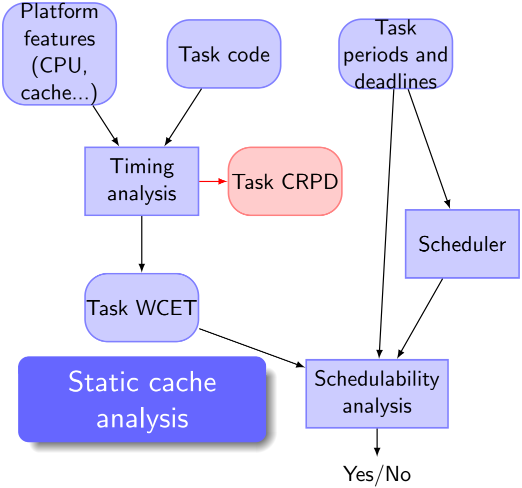
Bounding the CRPD



- preempted task
  - ↪ Useful Cache Blocks (UCBs)
- preempting task
  - ↪ Evicting Cache Blocks (ECBs)
- combined approaches
  - ↪ both tasks

- *Altmeyer et al. (2012, [12])*

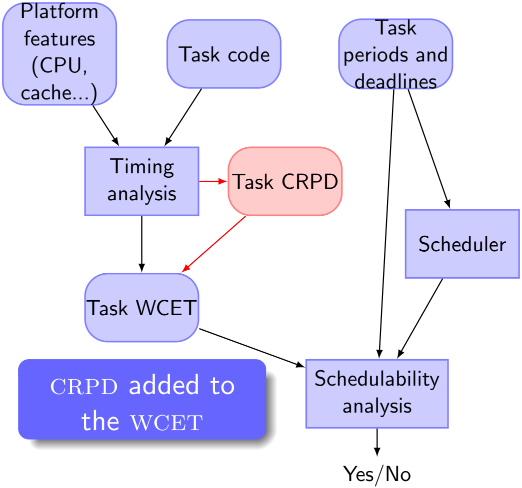
Bounding the CRPD



- preempted task
  - ↪ Useful Cache Blocks (UCBs)
- preempting task
  - ↪ Evicting Cache Blocks (ECBs)
- combined approaches
  - ↪ both tasks
- improvements:
  - ↪ Definitely-Cached UCBs

- *Altmeyer et al.(2009, [13])*

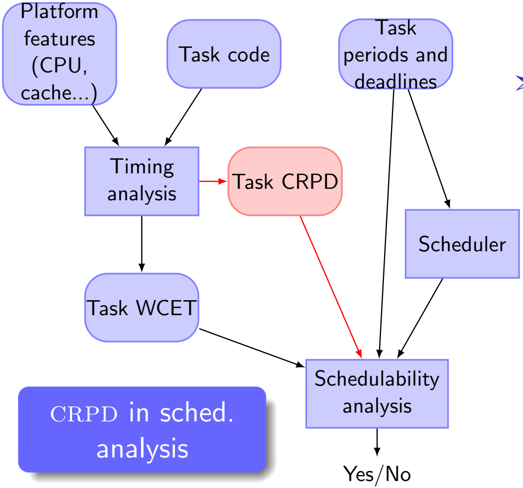
Bounding the CRPD



- $\text{WCET}_{\text{w/o preemption}} + n \times \text{CRPD}$
  - $\hookrightarrow n$?

- *Altmeyer et al.(2011, [14])*

Bounding the CRPD

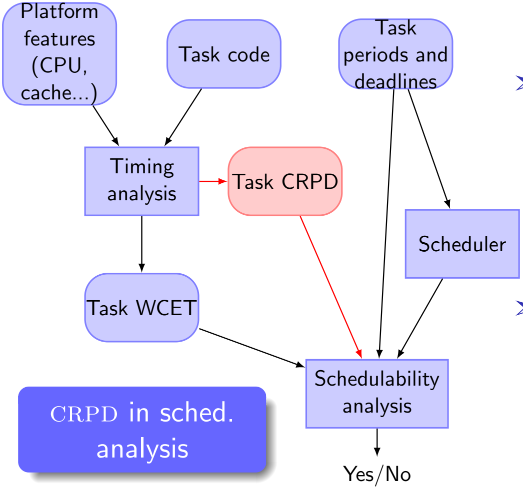


CRPD in sched. analysis

- Fixed-Priority (FP):
  $\hookrightarrow$ $R_i = C_i + \sum \left\lceil \frac{R_i}{T_j} \right\rceil \cdot (C_j + \gamma_{i,j})$

- *Busquets-Mataix et al.(1996, [11])*

Bounding the CRPD

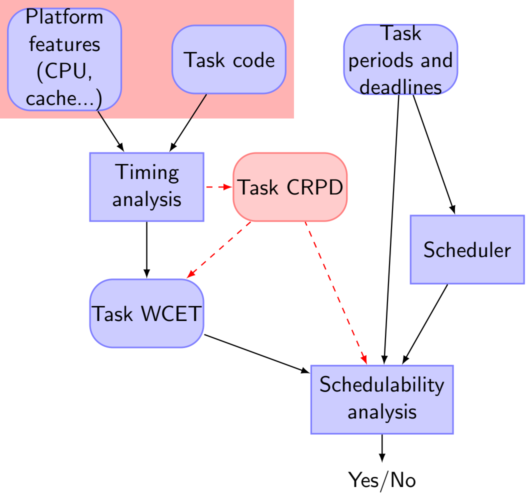


- Fixed-Priority (FP):
  - ↪ $R_i = C_i + \sum \left\lceil \frac{R_i}{T_j} \right\rceil \cdot (C_j + \gamma_{i,j})$
- EDF scheduling:
  - ↪ Similar to Fixed-Priority

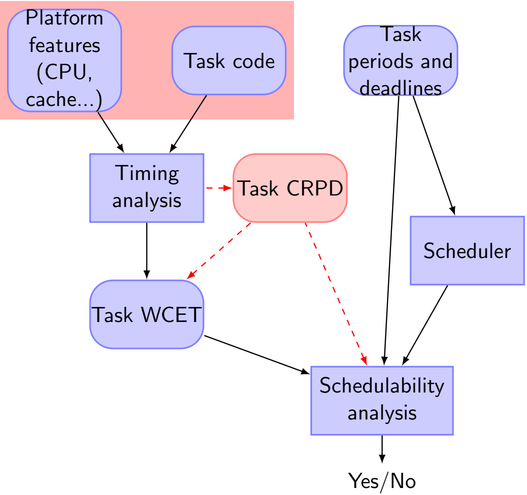
CRPD in sched. analysis

- *Lunniss et al., (2013,[15])*

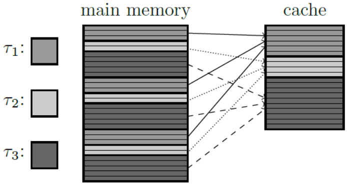
Cache management



Cache management

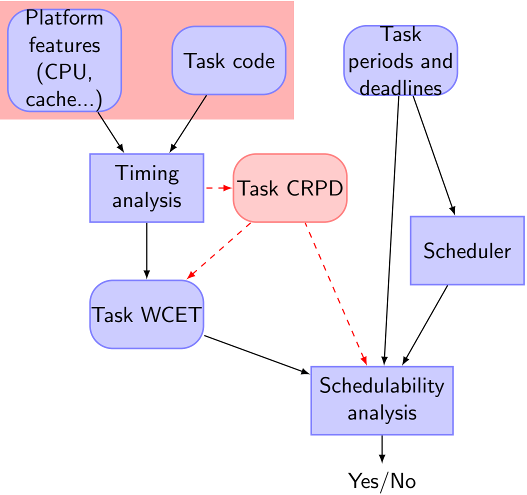


- cache partitioning



- *Bui et al. (2008,[16])*

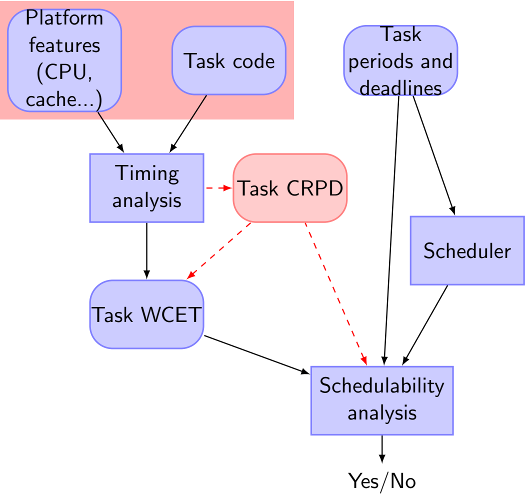
Cache management



- cache partitioning
- cache locking
  - → cache content fixed
    - ⇒ predictability

- *Ding et al. (2013,[17])*

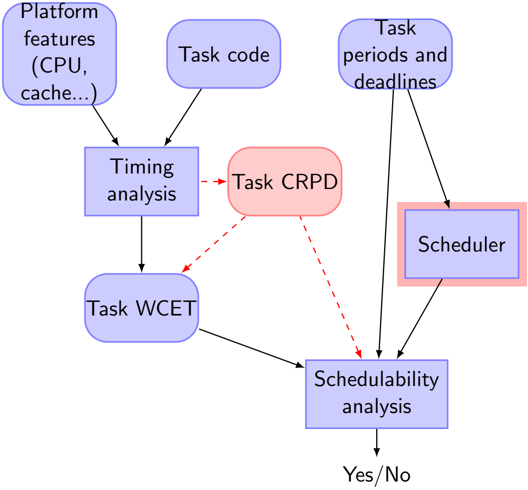
Cache management



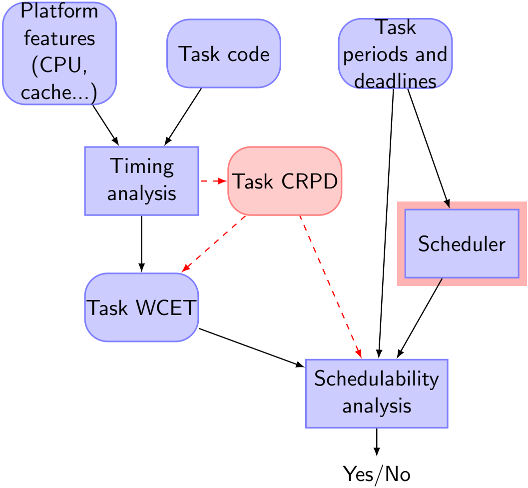
- cache partitioning
- cache locking
  - → cache content fixed
    - ⇒ predictability
- memory layout
  - code positioning
    - ⇒ ↘ WCET
  - task positioning
    - ⇒ ↘ CRPD

- *Lunniss et al (2012,[18])*

Scheduling (Controlling preemptions)



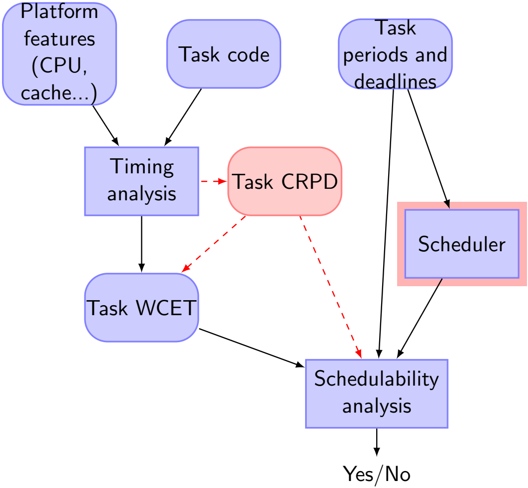
Scheduling (Controlling preemptions)



- Preemption Thresholds
  - ↪ preemption possible only if: priority(preempting task) > threshold(preempted task)

- *Bril et al. (2014,[19])*

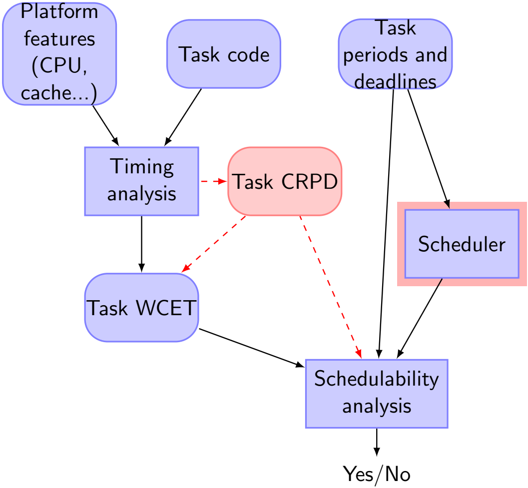
Scheduling (Controlling preemptions)



- Preemption Thresholds
  - ↪ preemption possible only if: priority(preempting task) > threshold(preempted task)
- Deferred Preemptions
  - ↪ preemption postponed as much as possible

- *Bertogna et al. (2010,[20])*

Scheduling (Controlling preemptions)



- Preemption Thresholds
  - ↪ preemption possible only if: priority(preempting task) > threshold(preempted task)
- Deferred Preemptions
  - ↪ preemption postponed as much as possible
- Fixed Preemptive Points
  - ↪ Offline Stage: preemptions allowed only at precomputed program points, to minimize CRPD;
  - ↪ Online stage: standard online priority scheduling.

- *Bertogna et al. (2011,[21])*

State-of-the-art (Tentative)

category	techniques		references
Memory Management	cache partitioning	fully-partitioning	[22]–[25]
		hybrid-partitioning	[16], [26]
	cache locking	full locking	[27]–[33]
		partial locking	[17], [34]
	partitioning + locking		[35]
	memory layout		[18], [36]–[42]
WCET	WCET integrating cache analysis		[43]–[48]
	CRPD in WCET	preempting task	[49], [50]
		preempted task	[13], [14], [51]
		both tasks	[52]–[54]

State-of-the-art (Tentative (Cont'))

category	techniques			references
Sched.	CRPD in Sched. Analysis	Preempting task		[11], [55]
		Preempted task		[56]
		both tasks		[12], [15], [57], [58]
	Preemption Control	Floating Non Preemptive Region	CRPD as a function	[59], [60]
		Preemption Thresholds	Opt+CRPD in Sched. Analysis	[19], [61]

State-of-the-art (Tentative (Cont'))

category	techniques			references
Sched. (Cont')	Preemption Control (Cont')	Deferred Preemption	opt.+ CRPD in Sched. Analysis	[4], [21], [62]–[65]

State-of-the-art scheduling techniques

- Integrating CRPD into existing scheduling techniques (i.e., FP, EDF, and their variants are considered).
- Results mainly for direct-mapped caches

Open issues (from scheduling point of view)

Approved!!

Timing and Schedulability Analysis must take into account CRPD.

Which granularity of the task model for improving scheduling decisions?

- fine-grained: exploiting cache state information?
- coarse-grained: worst-case CRPD as the maximum preemption delay for a task set, for a task, for a segment within a task,...?

Scheduling algorithms

- Extend and mix known techniques?
- Need of new approaches?

Current Work in Poitiers (Guillaume's PhD Thesis)

Cache impact on *optimally* taking **scheduling decisions**

- Objective: define the most basic task models to cover:
  - ↪ the largest set of scheduling problems
  - ↪ all cache architectures (direct-mapped, Set-Associative and Full-Associative)
  - scheduling with *cache-related preemption delays*
    - → **crpd-aware scheduling problem**
  - scheduling with *cache state information*
    - → **Cache-aware scheduling problem**

Current collaborations

- Claire Maiza (Verimag, Grenoble)
- Joël Goossens (ULB, Brussels)

Outline

CRPD scheduling problem statement

Scheduling decisions exploit cache-related preemption delays
→ **minimize the total overhead** (i.e., sum of CRPDs).

Task model: $\tau_i(C_i, D_i, T_i, \gamma)$

- C_i: WCET without CRPD
  ↪ τ_i executed fully non preemptively
- γ: worst-case CRPD for one preemption
  ↪ the same for all program points and all tasks

Presented results [66], [67]

- Optimality and Complexity Analyses
- Sustainability Analysis (scheduling anomalies)
- Competitive Analysis (i.e., online v.s. offline schedulers)
- Optimal offline scheduling method and numerical results

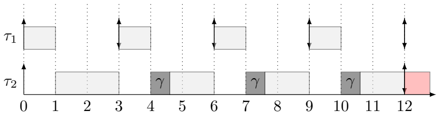
Optimality Analysis

Example ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

Optimality Analysis

Example ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

- Fixed-Task/Fixed-Job Priority Scheduling:

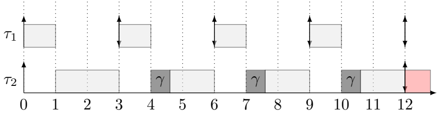


$\rightarrow$ not schedulable

Optimality Analysis

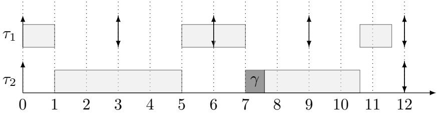
Example ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

- Fixed-Task/Fixed-Job Priority Scheduling:



$\rightarrow$ not schedulable

- CRPD-aware scheduling:

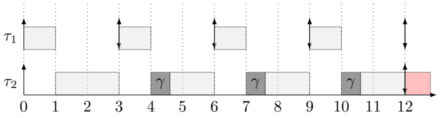


$\rightarrow$ schedulable

Optimality Analysis

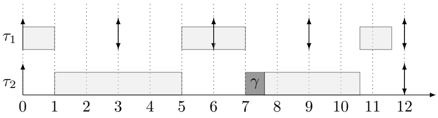
Example ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

- Fixed-Task/Fixed-Job Priority Scheduling:



$\rightarrow$ not schedulable

- CRPD-aware scheduling:



$\rightarrow$ schedulable

$\Rightarrow$ Fixed-Task and Fixed-Job Priority schedulers $\rightarrow$ **not optimal**.

Optimality Analysis (Cont')

Finite set of synchronous tasks $\tau_i(C_i, D_i, T_i, \gamma)$,

$\hookrightarrow$ Is there a uniprocessor preemptive schedule meeting the deadlines?

Optimality Analysis (Cont')

Finite set of synchronous tasks $\tau_i(C_i, D_i, T_i, \gamma)$,

$\hookrightarrow$ Is there a uniprocessor preemptive schedule meeting the deadlines?

$\Rightarrow$ **NP-hard** in the *strong sense*.

Proof [67]: transformation from the 3-Partition decision problem

Optimality Analysis (Cont')

Proof (Sketch)

3-Partition [68]

- Instance: a set A of $3m$ elements, a bound $B \in N$, and a size $s_j \in N$ for each $j = 1..3m$ such that $B/4 < s_j < B/2$ and $\sum_{j=1..3m} s_j = mB$.
- Question: Can A be partitioned into m disjoint sets $A_1, A_2, ..., A_m$ such that, for $1 \leq i \leq m$, $\sum_{j \in A_i} s_j = B$ (each A_i must therefore contain exactly three elements from A)?

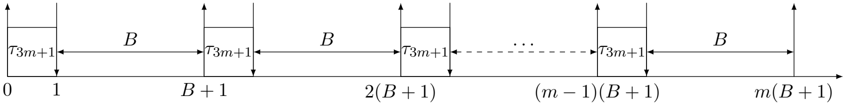
Transformation: To every 3-Partition instance

- $3m$ tasks $\tau_1, \ldots, \tau_{3m}$ with the parameters: $C_i = s_i, D_i = T_i = m(B+1)$, $1 \leq i \leq 3m$.
- Task τ_{3m+1} with: $C_{3m+1} = D_{3m+1} = 1$ and $T_i = (B+1)$

Optimality Analysis (Cont')

Proof sketch (Cont') In the task set defined from a 3-Partition instance:

- the workload is 100% without preemption
- In every feasible schedule, the pattern follows:



Property

A schedule is feasible if, and only if, there is no preemption (one preemption leads to a workload $> 100\%$).

Optimality Analysis (Cont')

- Set of jobs with 2 distinct releases and deadlines and $\gamma = 1$

$\Rightarrow$ NP-hard in the *weak sense*.

Proof [69]: transformation from the 2-Partition decision problem

Optimality Analysis (Cont')

- Set of jobs with 2 distinct releases and deadlines and $\gamma = 1$

$\Rightarrow$ NP-hard in the *weak sense*.

Proof [69]: transformation from the 2-Partition decision problem

- EDF still optimal for finite set of jobs with either:
  - $r_i = 0$
  - $d_i = d$
  - $C_i = 1$
  - similarly ordered release times and deadlines: $r_i \leq r_j \Rightarrow d_i \leq d_j$

$\rightarrow$ since no job is preempted using EDF

Sustainability analysis

Sustainability definition

A scheduling policy is sustainable if any systems deemed schedulable remain schedulable if:

- a WCET is decreased
- a period is increased
- a relative deadline is increased

NEW preemption delay is decreased

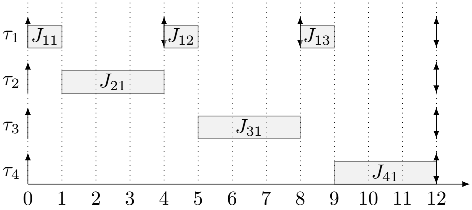
Known results for uniprocessor without CRPD [70]

- EDF is sustainable with respect to: WCET, period, deadlines
- Fixed-Task Priority (RM,DM) is sustainable w.r.t.: WCET, deadlines

EDF, RM and DM non Sustainability w.r.t. WCET

➢ 4 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 4, 4, 0.6)$, $\tau_2(3, 12, 12, 0.6)$, $\tau_3(3, 12, 12, 0.6)$ and $\tau_4(3, 12, 12, 0.6)$

➢ EDF, RM and DM use the same job priority assignment (task index as tie breaker).

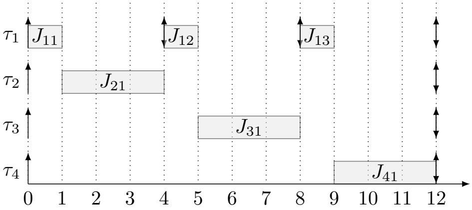


→ schedule with $C_2 = 3$

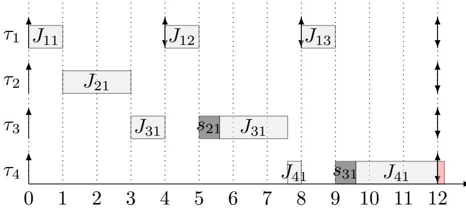
EDF, RM and DM non Sustainability w.r.t. WCET

➢ 4 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1,4,4,0.6)$, $\tau_2(3,12,12,0.6)$, $\tau_3(3,12,12,0.6)$ and $\tau_4(3,12,12,0.6)$

➢ EDF, RM and DM use the same job priority assignment (task index as tie breaker).



→ schedule with $C_2 = 3$

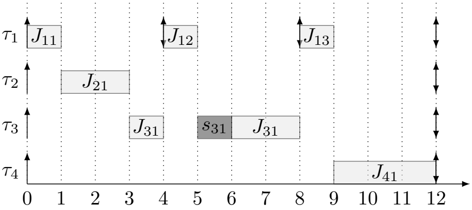


→ schedule with $C_2 = 2$

EDF, RM and DM non Sustainability w.r.t. Preemption delays

➢ 4 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1,4,4,1)$, $\tau_2(3,12,12,1)$, $\tau_3(3,12,12,1)$ and $\tau_4(3,12,12,1)$

➢ EDF, RM and DM use the same job priority assignment (task index as tie breaker).

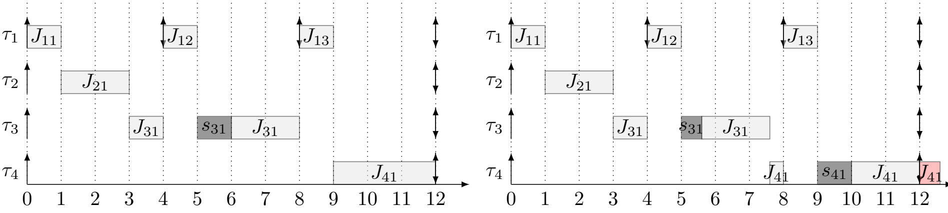


$\rightarrow$ schedule with $\gamma = 1$

EDF, RM and DM non Sustainability w.r.t. Preemption delays

➢ 4 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 4, 4, 1)$, $\tau_2(3, 12, 12, 1)$, $\tau_3(3, 12, 12, 1)$ and $\tau_4(3, 12, 12, 1)$

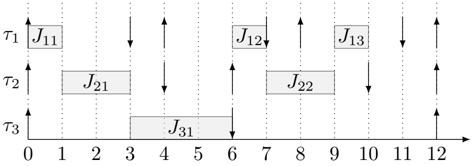
➢ EDF, RM and DM use the same job priority assignment (task index as tie breaker).



→ schedule with $\gamma = 1$ → schedule with $\gamma_{31} \leq 1$

EDF non Sustainability w.r.t. relative deadlines

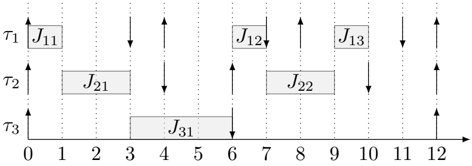
- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1,3,4,1)$, $\tau_2(2,4,6,1)$, $\tau_3(3,6,12,1)$
- EDF (task index as tie breaker).



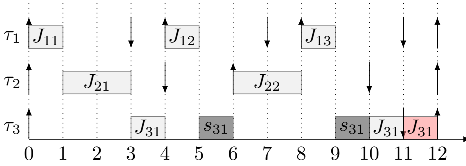
$\rightarrow$ EDF schedule with $D_3 = 6$

EDF non Sustainability w.r.t. relative deadlines

- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 4, 1)$, $\tau_2(2, 4, 6, 1)$, $\tau_3(3, 6, 12, 1)$
- EDF (task index as tie breaker).



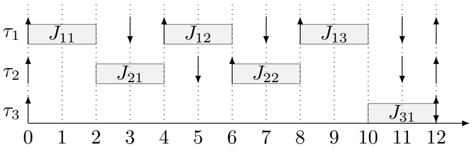
$\rightarrow$ EDF schedule with $D_3 = 6$



$\rightarrow$ EDF schedule with $D_3 = 11$

EDF non Sustainability w.r.t. periods

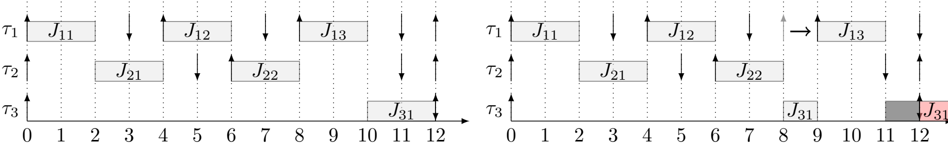
- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(2,3,4,1)$, $\tau_2(2,5,6,1)$, $\tau_3(2,12,12,1)$
- Shifting one release time (sporadic tasks)



$\rightarrow$ EDF schedule with $T_1 = 4$

EDF non Sustainability w.r.t. periods

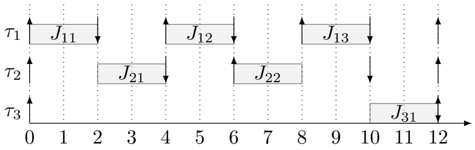
- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(2,3,4,1)$, $\tau_2(2,5,6,1)$, $\tau_3(2,12,12,1)$
- Shifting one release time (sporadic tasks)



$\rightarrow$ EDF schedule with $T_1 = 4$ $\rightarrow$ EDF schedule with $T_1 = 5$

EDF non Sustainability w.r.t. periods

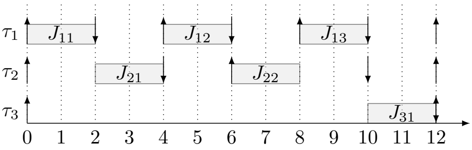
- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(2,3,4,1)$, $\tau_2(2,5,6,1)$, $\tau_3(2,12,12,1)$
- periodic tasks



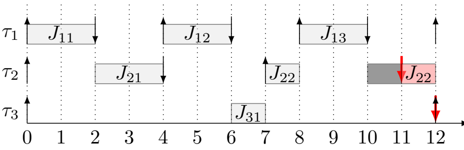
$\rightarrow$ EDF schedule with $T_2 = 6$

EDF non Sustainability w.r.t. periods

- 3 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(2, 3, 4, 1)$, $\tau_2(2, 5, 6, 1)$, $\tau_3(2, 12, 12, 1)$
- periodic tasks



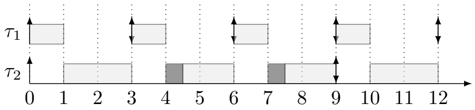
$\rightarrow$ EDF schedule with $T_2 = 6$



$\rightarrow$ EDF schedule with $T_2 = 7$

FP and critical instant worst-case scenario

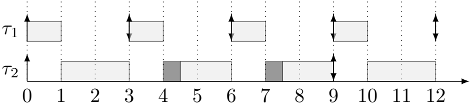
- 2 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.5)$, $\tau_2(5, 9, 9, 0.5)$
- sporadic tasks scheduled by RM
- Similar examples in [47], [71]

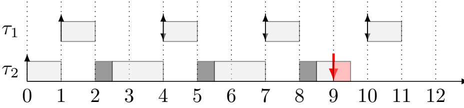


$\rightarrow$ wcrt(τ_2)=9

FP and critical instant worst-case scenario

- 2 tasks ($\tau_i(C_i, D_i, T_i, \gamma)$): $\tau_1(1, 3, 3, 0.5)$, $\tau_2(5, 9, 9, 0.5)$
- sporadic tasks scheduled by RM
- Similar examples in [47], [71]





$\rightarrow$ wcrt(τ_2)=9

$\rightarrow$ wcrt(τ_2)=9.5

Competitive analysis

Online scheduling model

- Set of jobs released over time
- at each job release, all its parameters are known

Result: Optimal online scheduling is impossible

Job release times need to be a priori known to define an optimal online scheduler (i.e., clairvoyant).

Competitive analysis (Cont')

Proof sketch

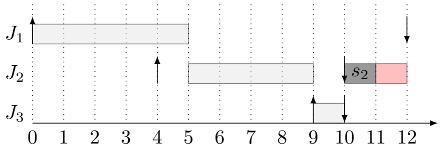
Optimal offline scheduler (the adversary) generates jobs so that any online scheduler cannot define a feasible schedule whereas the adversary can.

Adversary strategy:

1. Generate two jobs J_1(0,5,12,1) and J_2(4,5,6,1)
2. At time 4
   - Case 1 the online scheduler continues to execute τ_1
   - Case 2 the online scheduler preempts τ_1 to execute τ_2
3. According to the case, the adversary defines a new job τ_3.

Proof sketch (Cont')

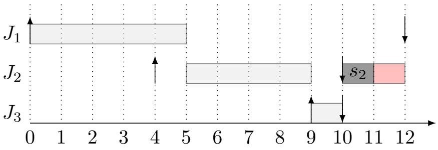
Case 1. The online scheduler continues to execute Job τ_1 at time 4. Adversary generates a job J_3(9,1,1,1).



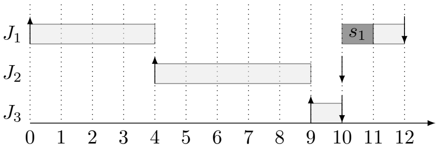
$\rightarrow$ Online algorithm

Proof sketch (Cont')

Case 1. The online scheduler continues to execute Job τ_1 at time 4. Adversary generates a job J_3(9,1,1,1).



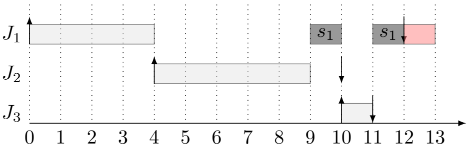
→ Online algorithm



→ Adversary's feasible schedule

Proof sketch (Cont')

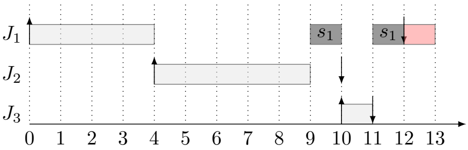
Case 2. the online scheduler preempts τ_1 to execute τ_3 at time 4 τ_2. Adversary generates a job J_3(10,1,1,1).

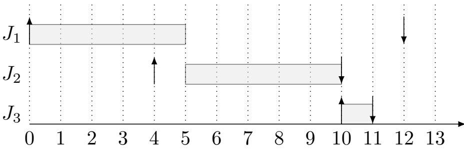


→ Online algorithm

Proof sketch (Cont')

Case 2. the online scheduler preempts τ_1 to execute τ_3 at time 4 τ_2. Adversary generates a job J_3(10,1,1,1).





$\rightarrow$ Online algorithm

$\rightarrow$ Adversary's feasible schedule

Offline scheduling: MILP formulation

Mixed Integer Linear Program (MILP)

Define an offline schedule to:

- Minimize the total workload
- or equivalently, minimize the total preemption delay (since WCET contributes as a constant in the objective function)

Contraints:

- each job is executed for its WCET
- each job is executed between its release time and its deadline
- at most one job is executed at any time instant
- explicit preemption delay

MILP: schedule construction

Schedule Construction

- The schedule is defined as a finite set of slices S_j, separated by release dates/deadlines (No job release inside a slice).
- In every slice, job-pieces and their related preemption delays must fit in the slice interval

Main property to define a feasible schedule

Every job is executed at most once in every slice with no pmtn.

MILP Variables to define the schedule in each slice S_j

- $t_{i,j} \in \mathbb{R}$: starting time of job-piece τ_i in S_j
- $p_{ij} \in \mathbb{R}$: execution time of job-piece τ_i in S_j
- $\Delta_{ij} \in \{0, 1\}$: job-piece τ_i has to pay a pmtn. delay in S_j.

MILP: schedule construction (Cont')

Rewriting simplification

- Execution time modification: $C_i' = C_i - s_i$ (i.e., the first job-piece pays a fictive pmtn delay at release)
- Simpler formulation of pmtn. penalty constraints in the MILP

Simple example with two periodic tasks

Five jobs $J_i(r_i, C_i, s_i, D_i)$ generated within [0,12)

- Task 1: $J_1(0, 1, 0.2, 3)$, $J_2(3, 2, 1, 0.2, 3)$, $J_3(6, 1, 0.2, 3)$, $J_4(9, 1, 0.2, 3)$
- Task 2: $J_5(0, 7, 0.5, 12)$

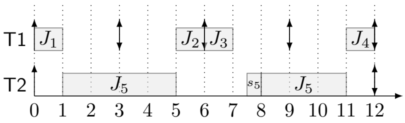
It can be easily checked that EDF generates 3 pmtn and misses a deadline.

MILP: schedule construction (Cont')

- 4 Slices: S_1=[0, 3),S_2=[3,6),S_3=[6,9),S_4=[9,12)
- 8 job-pieces (rows in the following table)

MILP results:

job-pieces		Variables			pmtn
job	slice	t_{ij}	C'_{ij}	Δ_{ij}	s_i
J_1	1	0	0.8	1	0.2
J_5	1	1	1.5	1	0.5
J_5	2	3	2	0	0.2
J_2	2	5	0.8	1	0.2
J_3	3	6	0.8	1	0.2
J_5	3	7.5	1	1	0.5
J_5	4	9	2	0	0.5
J_4	4	11	0.8	1	0.2



→ Optimal schedule

Experiments

Synthetic task sets

- (C_i, T_i) UUnifast to generate utilization factors
- s_i randomly generated between 0 and the Maximum Pmtn Delay Factor (PDF, percentage of the WCET C_i)
- Limited to 200 jobs over the hyperperiod (to limit CPLEX running time)

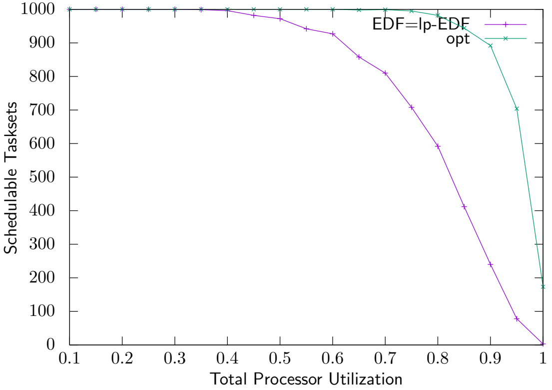
Monitored Algorithms

- EDF based algorithms
  - EDF: arbitrary tie breaker
  - LP-EDF: tie breaker avoid unnecessary preemptions
  - Cache-related schedulability analysis (Lunniss *et al.* RTAS'2013 [15])
- OPT: MILP solved by CPLEX 12.6.1

Results: Total Utilization

Experiment parameters

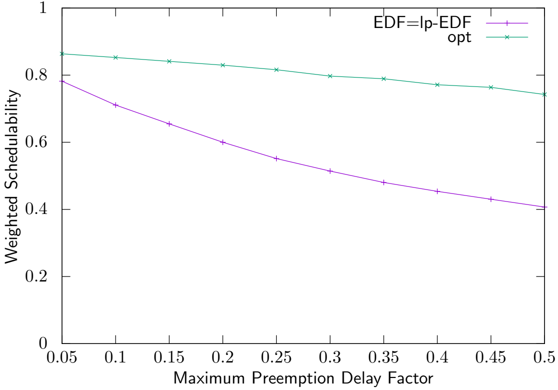
Maximum Preemption Delay Factor (PDF) 20%.



Results: Weighted Schedulability

Experiment parameters

Weighted Schedulability as a function of the Maximum PDF.



$Q = \{u | u = k \cdot 0.1, k \in [\![1, 10]\!]\}$

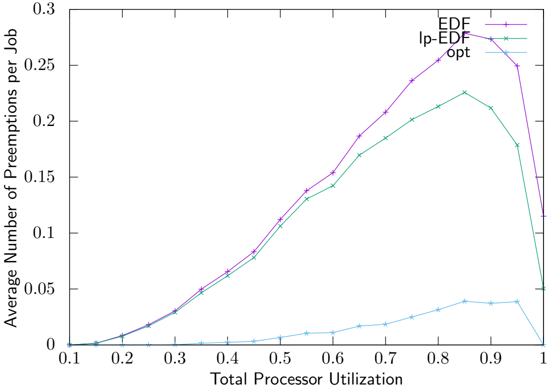
$$W_\ell(PDF) = \frac{\sum_{\forall U \in Q} U \cdot S_\ell(U, PDF)}{\sum_{\forall U \in Q} U}$$

with $S_\ell(U, PDF)$ binary sched. result

Results: Number of preemptions

Experiment parameters

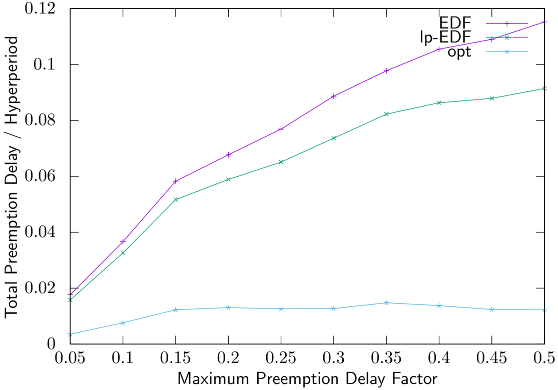
Average number of preemptions per job as function of U (PDF=20%).



Results: Total preemption delays

Experiment parameters

Total preemption delays as a function of maximum PDF.



Outline

Cache-aware scheduling problem

Scheduling with cache state information

Cache assumptions

- consists in a single cache line,
- miss cost $= BRT$ (Block Reload Time), hit=0
- block references: string of letters

Simple Job model

synchronous Job i: $J_i(C_i, D, S_i)$:

- C_i: WCET *without cache miss*,
- D: a common relative deadline for all jobs
- S_i: sequence of memory blocks used during the job execution
  - $\hookrightarrow$ One memory block used per time unit
  - $\hookrightarrow$ no *if-then-else* structure

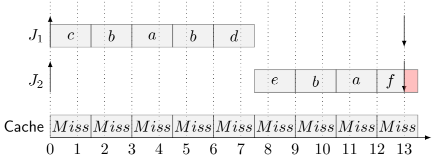
Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, BRT $= 0.5$

$$S_1 = c \to b \to a \to b \to d, \qquad S_2 = e \to b \to a \to f$$

Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, BRT $= 0.5$

$$S_1 = c \rightarrow b \rightarrow a \rightarrow b \rightarrow d, \quad S_2 = e \rightarrow b \rightarrow a \rightarrow f$$

- Fixed-Job Priority Scheduling ($prio(J_1) > prio(J_2)$):

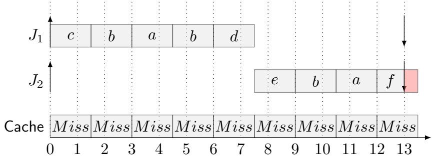


$\rightarrow$ not schedulable

Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, BRT $= 0.5$

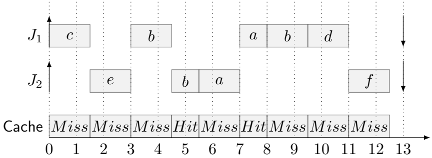
$$S_1 = c \to b \to a \to b \to d, \quad S_2 = e \to b \to a \to f$$

- Fixed-Job Priority Scheduling ($prio(J_1) > prio(J_2)$):



$\to$ not schedulable

- Cache-aware scheduling:

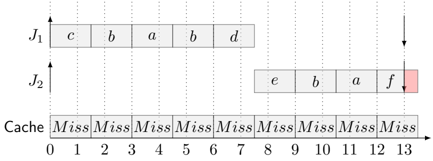


$\to$ schedulable

Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, BRT $= 0.5$

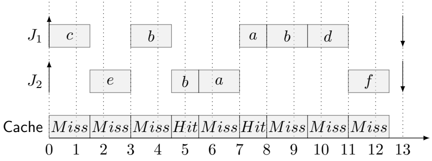
$$S_1 = c \to b \to a \to b \to d, \quad S_2 = e \to b \to a \to f$$

- Fixed-Job Priority Scheduling ($prio(J_1) > prio(J_2)$):



$\to$ not schedulable

- Cache-aware scheduling:



$\to$ schedulable

$\Rightarrow$ Fixed-Task and Fixed-Job Priority schedulers $\to$ **not optimal**.

Complexity result

Finite set of n jobs $J_i(C_i, D, S_i)$ with a common deadline D

$\hookrightarrow$ a uniprocessor preemptive schedule meeting the overall deadline D for every job J_i?

Complexity result

Finite set of n jobs $J_i(C_i, D, S_i)$ with a common deadline D

$\hookrightarrow$ a uniprocessor preemptive schedule meeting the overall deadline D for every job J_i?

$\Rightarrow$ **NP-hard** in the *strong sense*.

Proof (see [67]): Transformation from the Shortest Common Supersequence problem known to be strongly NP-Complete [68]

Outline

Conclusion

- **Real-Time Scheduling** with **Cache-Related Pmtn Delays**
- CRPD-*aware* scheduling problem
  - ↪ Scheduling anomalies for standard scheduling policies (i.e., FP,EDF)
  - ↪ **NP-hard** in the strong sense
    - ⇒ no pseudo-polynomial optimal scheduling algorithm
  - ↪ No optimal online scheduling policies (set of jobs)
- *Cache-aware* scheduling problem
  - ↪ RM, EDF are not optimal
  - ↪ **NP-hard** in the strong sense
    - ⇒ no pseudo-polynomial optimal scheduling algorithm

Perspectives

Continue to study basic task models

- to understand underlying theoretical problems
- to evaluate the performance of a given technique
- to delimit practical solutions for real-world problems

Scheduling under cache constraints

- Uniprocessor: evaluate the loss of schedulability of online schedulers
- Multiprocessors: timing issues are quite tricky

Perspectives (*Cont.*)

Circular dependencies must be taken into account

- Timing issues: CRPD computation under constraints (cache partitioning, Code positioning, enforced scheduling isolation,...)
- Compilers: code/task positioning to avoid bottlenecks in caches
- Cache management: partitioning/locking techniques
- Schedulers: taking into account all previously mentioned constraints

Models and Metrics

- Shared task models to fully exploit them at design stages (timing, scheduling analysis, scheduler)
- Common metrics to measure the relative contributions of mixed techniques

Thank you!
Questions?

References

[1] R. Pellizzoni and M. Caccamo, “Toward the predictable integration of real-time COTS based systems,” in *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS 2007), 3-6 December 2007, Tucson, Arizona, USA*, 2007, pp. 73–82.

[2] R. Pellizzoni, B. D. Bui, M. Caccamo, and L. Sha, “Coscheduling of CPU and I/O transactions in cots-based embedded systems,” in *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS 2008, Barcelona, Spain, 30 November - 3 December 2008*, 2008, pp. 221–231.

[3] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley, “A predictable execution model for cots-based embedded systems,” in *17th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2011, Chicago, Illinois, USA, 11-14 April 2011*, 2011, pp. 269–279.

[4] J. Cavicchio, C. Tessler, and N. Fisher, "Minimizing cache overhead via loaded cache blocks and preemption placement," in *27th Euromicro Conference on Real-Time Systems, ECRTS 2015, Lund, Sweden, July 8-10, 2015*, 2015, pp. 163–173.

[5] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1990.

[6] J. Calandrino and J. Anderson, "Cache-aware real-time scheduling on multicore platforms: Heuristics and a case study," in *Real-Time Systems, 2008. ECRTS '08. Euromicro Conference on*, 2008, pp. 299–308.

[7] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Syst. J.*, vol. 5, no. 2, pp. 78–101, Jun. 1966.

References (cont.)

[8] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, Feb. 1985.

[9] G. Phavorin and P. Richard, "Cahce-related preemption delays and real-time scheduling: A survey for uniprocessor systems, research report 03-2015," LIAS, Université de Poitiers, Tech. Rep., 2015.

[10] C.-G. Lee, J. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim, "Enhanced analysis of cache-related preemption delay in fixed-priority preemptive scheduling," in *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, IEEE, 1997, pp. 187–198.

References (cont.)

[11] J. Busquets-Mataix, J. Serrano, R. Ors, P. Gil, and A. Wellings, “Adding instruction cache effect to schedulability analysis of preemptive real-time systems,” in *Real-Time Technology and Applications Symposium, 1996. Proceedings., 1996 IEEE*, 1996, pp. 204–212.

[12] S. Altmeyer, R. I. Davis, and C. Maiza, “Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems,” *Real-Time Systems*, vol. 48, no. 5, pp. 499–526, 2012.

[13] S. Altmeyer and C. Maiza-Burguière, “A new notion of useful cache block to improve the bounds of cache-related preemption delay,” in *Proceedings of the 21st Euromicro Conference on Real-Time Systems (ECRTS ’09)*, 2009, pp. 109–118.

References (cont.)

[14] S. Altmeyer and C. M. Burguière, "Cache-related preemption delay via useful cache blocks: Survey and redefinition," *Journal of Systems Architecture*, no. 7, pp. 707–719, 2011.

[15] W. Lunniss, S. Altmeyer, C. Maiza, and R. Davis, "Integrating cache related pre-emption delay analysis into edf scheduling," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2013 IEEE 19th*, 2013, pp. 75–84.

[16] B. D. Bui, M. Caccamo, L. Sha, and J. Martinez, "Impact of cache partitioning on multi-tasking real time embedded systems," in *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA'08. 14th IEEE International Conference on*, IEEE, 2008, pp. 101–110.

References (cont.)

[17] H. Ding, Y. Liang, and T. Mitra, "Integrated instruction cache analysis and locking in multitasking real-time systems," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–10.

[18] W. Lunniss, S. Altmeyer, and R. I. Davis, "Optimising task layout to increase schedulability via reduced cache related pre-emption delays," in *Proceedings of the 20th International Conference on Real-Time and Network Systems*, 2012, pp. 161–170.

[19] R. J. Bril, S. Altmeyer, M. M. van den Heuvel, R. I. Davis, and M. Behnam, "Integrating cache-related pre-emption delays into analysis of fixed priority scheduling with pre-emption thresholds," in *Real-Time Systems Symposium (RTSS), 2014 IEEE*, IEEE, 2014, pp. 161–172.

References (cont.)

[20] M. Bertogna and S. Baruah, "Limited preemption edf scheduling of sporadic task systems," *Industrial Informatics, IEEE Transactions on*, pp. 579–591, 2010.

[21] M. Bertogna, O. Xhani, M. Marinoni, F. Esposito, and G. Buttazzo, "Optimal selection of preemption points to minimize preemption overhead," in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*, IEEE, 2011, pp. 217–227.

[22] D. Kirk, "Smart (strategic memory allocation for real-time) cache design," in *Real Time Systems Symposium, 1989., Proceedings.*, 1989, pp. 229–237.

[23] F. Mueller, "Compiler support for software-based cache partitioning," in *ACM Sigplan Notices*, 1995, pp. 125–133.

[24] S. Plazar, P. Lokuciejewski, and P. Marwedel, “Wcet-aware software based cache partitioning for multi-task real-time systems,” in *OASIcs-OpenAccess Series in Informatics*, 2009.

[25] A. Sebastian, R. Douma, W. Lunniss, and R. I. Davis, “Evaluation of cache partitioning for hard real-time systems,” in *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS'14)*, 2014, pp. 15–26.

[26] J. V. Busquets-Mataix, D. Gil, P. Gil, and A. Wellings, “Techniques to increase the schedulable utilization of cache-based preemptive real-time systems,” *Journal of systems architecture*, vol. 46, no. 4, pp. 357–378, 2000.

[27] M. Campoy, A. P. Ivars, and J. Busquets-Mataix, “Static use of locking caches in multitask preemptive real-time systems,” in *Proceedings of IEEE/IEE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, 2001.

[28] A Arnaud and I Puaut, “Dynamic instruction cache locking in hard real-time systems,” in *Proc. of the 14th Int. Conference on Real-Time and Network Systems*, 2006.

[29] I. Puaut, “Wcet-centric software-controlled instruction caches for hard real-time systems,” in *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, 2006, pp. 216–226.

[30] I. Puaut and C. Pais, “Scratchpad memories vs locked caches in hard real-time systems: A quantitative comparison,” in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, IEEE, 2007, pp. 1–6.

References (cont.)

[31] H. Falk, S. Plazar, and H. Theiling, "Compile-time decided instruction cache locking using worst-case execution paths," in *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, ACM, 2007, pp. 143–148.

[32] T. Liu, M. Li, and C. J. Xue, "Minimizing wcet for real-time embedded systems via static instruction cache locking," in *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, IEEE, 2009, pp. 35–44.

[33] ——, "Instruction cache locking for multi-task real-time embedded systems," *Real-Time Systems*, vol. 48, no. 2, pp. 166–197, 2012.

[34] H. Ding, Y. Liang, and T. Mitra, "Wcet-centric dynamic instruction cache locking," in *Proceedings of the conference on Design, Automation & Test in Europe*, European Design and Automation Association, 2014, p. 27.

[35] X. Vera, B. Lisper, and J. Xue, "Data caches in multitasking hard real-time systems," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, 2003, pp. 154–165.

[36] H. Tomiyama and H. Yasuura, "Code placement techniques for cache miss rate reduction," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, pp. 410–429, 1997.

[37] M. Kowarschik and C. Weiß, *An overview of cache optimization techniques and cache-aware numerical algorithms*, 2003.

References (cont.)

[38] P. Lokuciejewski, H. Falk, and P. Marwedel, "Wcet-driven cache-based procedure positioning optimizations," in *Real-Time Systems, 2008. ECRTS'08. Euromicro Conference on*, 2008, pp. 321–330.

[39] H. Falk and H. Kotthaus, "Wcet-driven cache-aware code positioning," in *Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2011, pp. 145–154.

[40] E. Mezzetti and T. Vardanega, "A rapid cache-aware procedure positioning optimization to favor incremental development," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2013 IEEE 19th*, 2013, pp. 107–116.

References (cont.)

[41] G. Gebhard and S. Altmeyer, "Optimal task placement to improve cache performance," in *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, ACM, 2007, pp. 259–268.

[42] S. Altmeyer and G. Gebhard, *Wcet analysis for preemptive scheduling*, 2008.

[43] R. White, F. Mueller, C. Healy, D. Whalley, and M. Harmon, "Timing analysis for data caches and set-associative caches," in *Real-Time Technology and Applications Symposium, 1997. Proceedings., Third IEEE*, 1997, pp. 192–202.

[44] C. A. Healy, R. D. Arnold, F. Mueller, D. B. Whalley, and M. G. Harmon, "Bounding pipeline and instruction cache performance," *Computers, IEEE Transactions on*, vol. 48, 53–70, 1 1999.

References (cont.)

[45] C. Ferdinand and R. Wilhelm, "Efficient and precise cache behavior prediction for real-time systems," *Real-Time Systems*, vol. 17, no. 2-3, pp. 131–181, 1999.

[46] F. Mueller, "Timing analysis for instruction caches," *Real-Time Systems*, pp. 217–247, 2000.

[47] H. Ramaprasad and F. Mueller, "Tightening the bounds on feasible preemption points," in *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International*, IEEE, 2006, pp. 212–224.

[48] S. Chattopadhyay and A. Roychoudhury, "Scalable and precise refinement of cache timing analysis via path-sensitive verification," *Real-Time Systems*, vol. 49, no. 4, pp. 517–562, 2013.

[49] S. Basumallick and K. Nilsen, "Cache issues in real-time systems," in *ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems*, vol. 5, 1994.

[50] H. Tomiyama and N. D. Dutt, “Program path analysis to bound cache-related preemption delay in preemptive real-time systems,” in *Proceedings of the Eighth International Workshop on Hardware/Software Codesign*, 2000, pp. 67–71.

[51] J. Schneider, “Cache and pipeline sensitive fixed priority scheduling for preemptive real-time systems,” in *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, 2000, pp. 195–204.

[52] H. S. Negi, T. Mitra, and A. Roychoudhury, “Accurate estimation of cache-related preemption delay,” in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2003, pp. 201–206.

[53] Y. Tan and V. Mooney, “Integrated intra- and inter-task cache analysis for preemptive multi-tasking real-time systems,” in *Software and Compilers for Embedded Systems*, 2004.

[54] B. C. Ward, A. Thekkilakattil, and J. H. Anderson, "Optimizing preemption-overhead accounting in multiprocessor real-time systems," in *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, 2014, p. 235.

[55] J. Busquets-Mataix, J. Serrano-Martin, R. Ors-Carot, P. Gil, and A. Wellings, "Adding instruction cache effect to an exact schedulability analysis of preemptive real-time systems," in *Real-Time Systems, 1996., Proceedings of the Eighth Euromicro Workshop on*, 1996, pp. 271–276.

[56] C.-G. Lee, J. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim, "Analysis of cache-related preemption delay in fixed-priority preemptive scheduling," *Computers, IEEE Transactions on*, vol. 47, no. 6, pp. 700–713, 1998.

References (cont.)

[57] Y. Tan and V. Mooney, “Timing analysis for preemptive multitasking real-time systems with caches,” *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 1, 2007.

[58] S. Altmeyer, R. I. Davis, and C. Maiza, “Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems,” in *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS)*, 2011.

[59] J. Marinho, V. Nelis, S. Petters, and I Puaut, “Preemption delay analysis for floating non-preemptive region scheduling,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 497–502.

[60] ——, “An improved preemption delay upper bound for floating non-preemptive region,” in *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, 2012, pp. 57–66.

[61] C. Wang, Z. Gu, and H. Zeng, "Integration of cache partitioning and preemption threshold scheduling to improve schedulability of hard real-time systems," in *27th Euromicro Conference on Real-Time Systems, ECRTS 2015, Lund, Sweden, July 8-10, 2015*, 2015, pp. 69–79.

[62] J. Simonson and J. H. Patel, "Use of preferred preemption points in cache-based real-time systems," in *Computer Performance and Dependability Symposium, 1995. Proceedings, International*, 1995, pp. 316–325.

[63] S. Altmeyer, C. Maiza-Burguière, and R. Wilhelm, "Computing the maximum blocking time for scheduling with deferred preemption," in *Workshop on Software Technologies for Future Dependable Distributed Systems*, 2009.

[64] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito, and M. Caccamo, "Preemption points placement for sporadic task sets," in *22nd Euromicro Conference on Real-Time Systems (ECRTS'10)*, IEEE, 2010, pp. 251–260.

[65] B. Peng, N. Fisher, and M. Bertogna, "Explicit preemption placement for real-time conditional code," in *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, IEEE, 2014, pp. 177–188.

[66] G. Phavorin, P. Richard, J. Gooseens, T. Chapeaux, and C. Maiza, "Scheluling with preemption delays: Anomalies and issues," in *Proc. Real-Time and Networked Systems (RTNS'15)*, 2015.

[67] G. Phavorin, P. Richard, and C. Maiza, "Complexity of scheduling real-time tasks subjected to cache-related preemption delays," in *Proc. Emerging Technologies for Factory Automation (ETFA'15)*, 2015.

[68] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979, ISBN: 0716710447.

[69] G. Phavorin, P. Richard, and C. Maiza, "Complexity of scheduling real-time tasks subjected to cache-related preemption delays," in *Research Report, 02-2015, LIAS-LAB, ENSMA and University of Poitiers*, 2015.

[70] A. Burns and S. K. Baruah, "Sustainability in real-time scheduling," *Journal of Computing Science and Engineering*, vol. 2, no. 1, pp. 74–97, 2008.

[71] P. M. Yomsi and Y. Sorel, "Extending rate monotonic analysis with exact cost of preemptions for hard real-time systems," in *Real-Time Systems, 2007. ECRTS'07. 19th Euromicro Conference on*, IEEE, 2007, pp. 280–290.