Scheduling with preemption delays: anomalies and issues

PHAVORIN Guillaume[1] RICHARD Pascal[1] GOOSSENS Joël[2]
CHAPEAUX Thomas[2] MAIZA Claire[3]

[1]LIAS/Université de Poitiers
{*guillaume.phavorin,pascal.richard*}*@univ-poitiers.fr*

[2]PARTS/Université Libre de Bruxelles
{*joel.goossens,thomas.chapeaux*}*@ulb.ac.be*

[3]Verimag/Université Grenobles-Alpes
claire.maiza@imag.fr

November 5th, 2015

Context

hard real-time scheduling

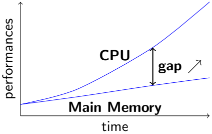
$\hookrightarrow$ common assumption for a long time: preemption costs $= 0$

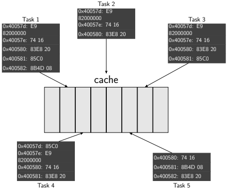
Context

hard real-time scheduling

$\hookrightarrow$ common assumption for a long time: preemption costs $= 0$

- components off-the-shelf (COTS) in embedded systems:
  - ➢ CPU $\rightarrow$ fast,
  - ➢ main memory $\rightarrow$ slow,
  - ➢ **cache** $\rightarrow$ small
    - $\hookrightarrow$ potential bottleneck



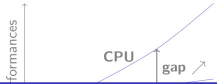


Context

~~**hard real-time scheduling**~~

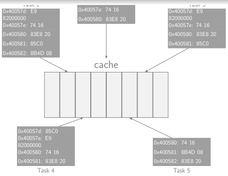
~~↪ common assumption for a long time: preemption costs = 0~~

- components off-the-shelf (COTS) in embedded systems:
  - ➢ CPU → fast,

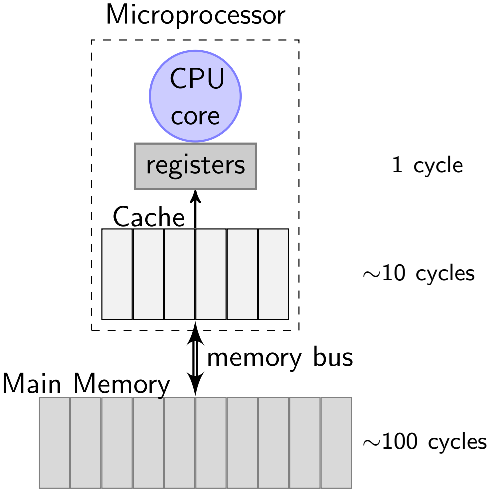


➢ preemption cost as high as 44% of the task WCET *(Pellizzoni et al. 2007)*

  - ➢ **cache** → small
    - ↪ potential bottleneck



Cache-Related Preemption Delays (CRPDs)

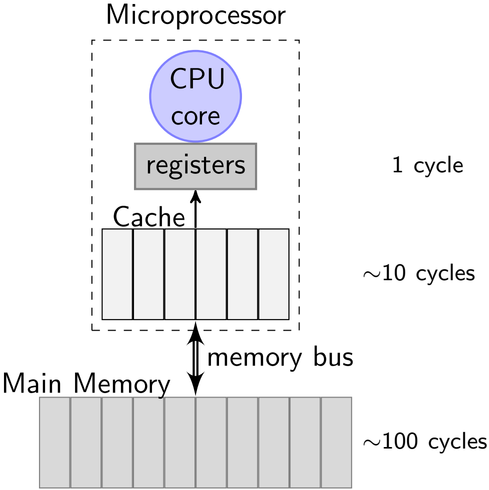


- cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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

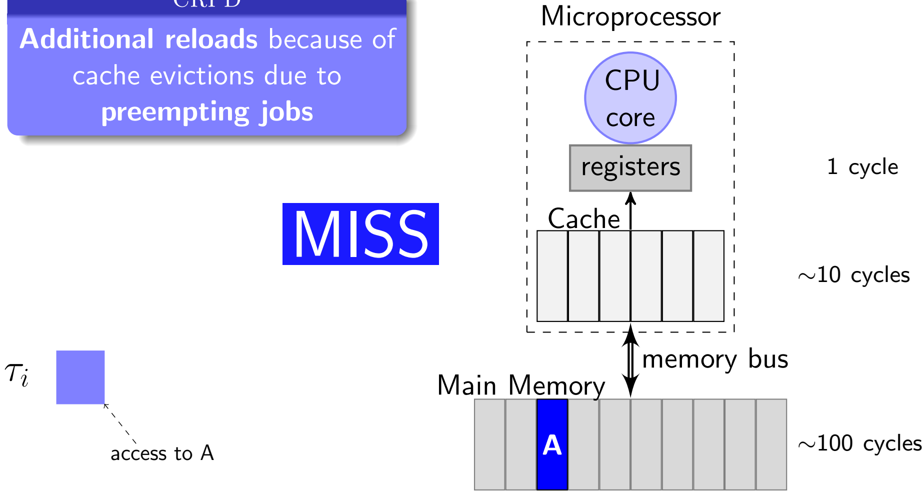


➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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

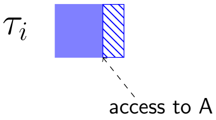


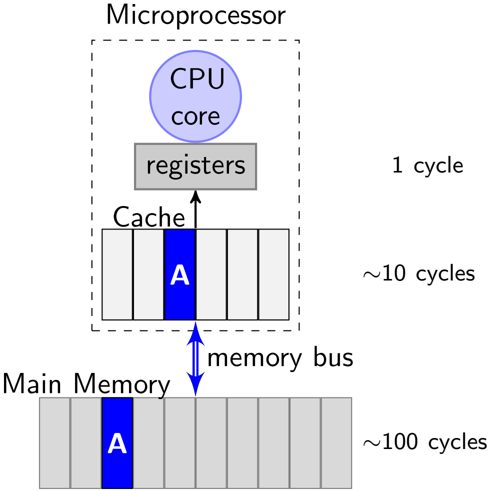
➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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





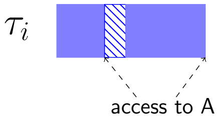
➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

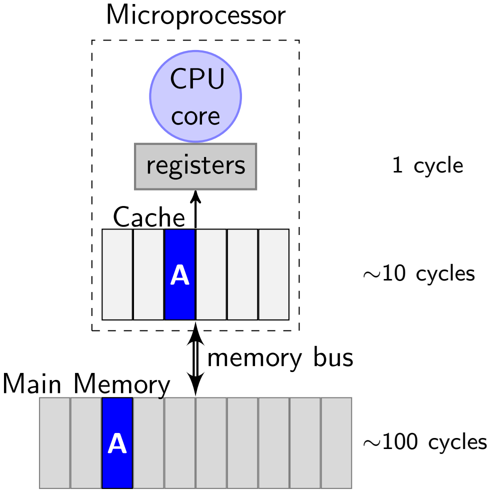
Cache-Related Preemption Delays (CRPDs)

CRPD

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

HIT



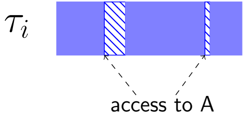


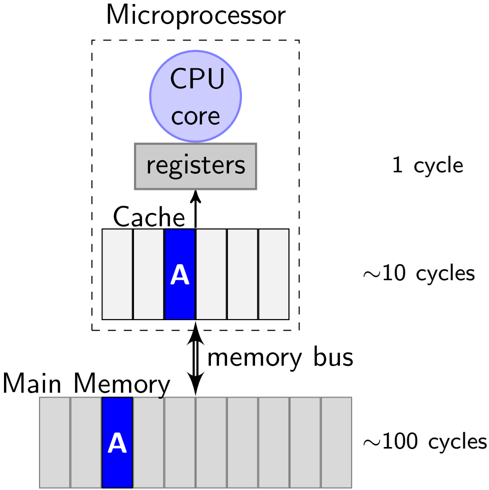
- cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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



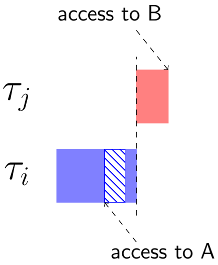


➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

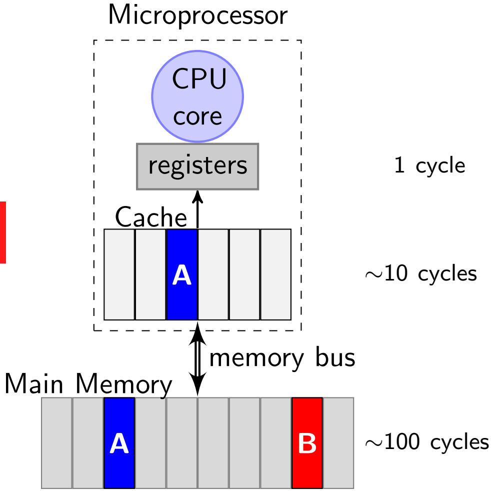
Cache-Related Preemption Delays (CRPDs)

CRPD

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



MISS

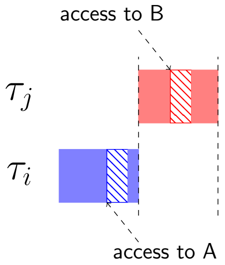


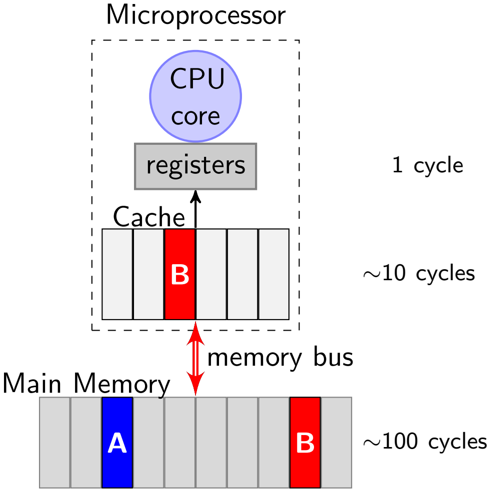
➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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



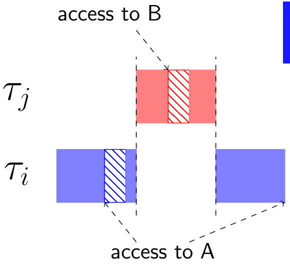


➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

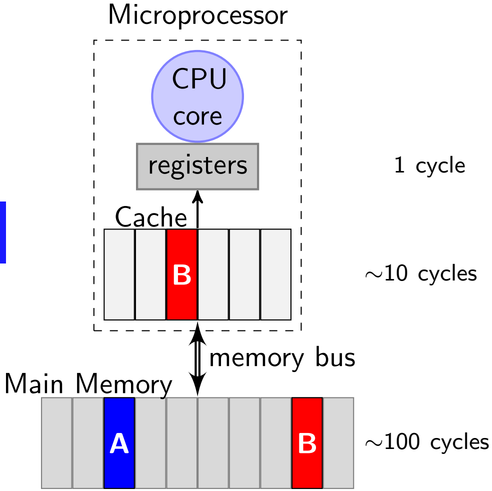
Cache-Related Preemption Delays (CRPDs)

CRPD

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



MISS

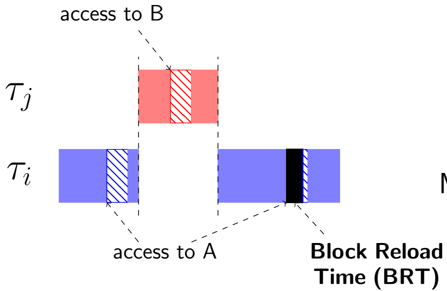


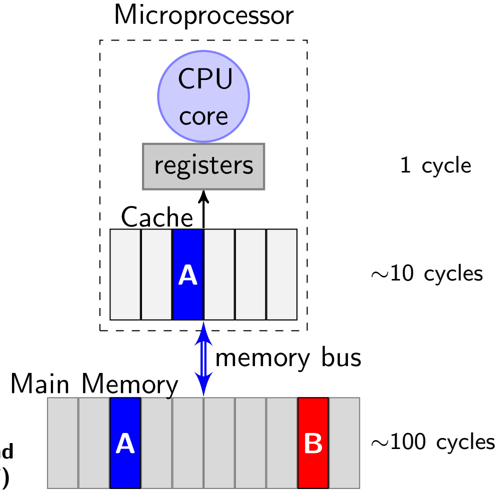
➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

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





➢ cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache-Related Preemption Delays (CRPDs)

CRPD

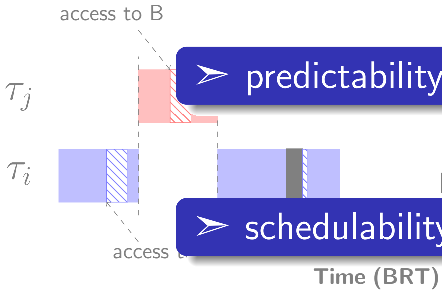
Additional reloads because of

cache ev

preem

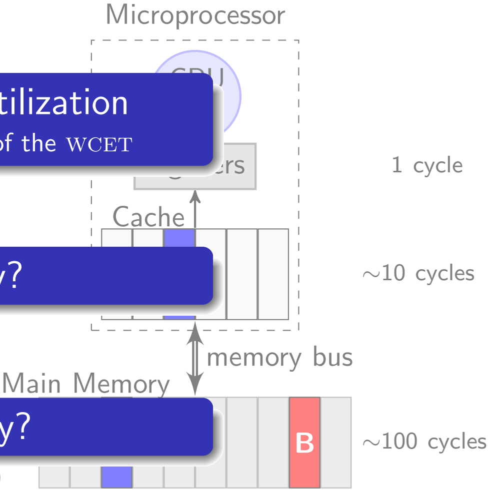
➢ ↗ system utilization

↪ up to 44% of the WCET



➢ predictability?

➢ schedulability?



➢ cost(**cache miss**) ≫ cost(**cache hit**)

Outline

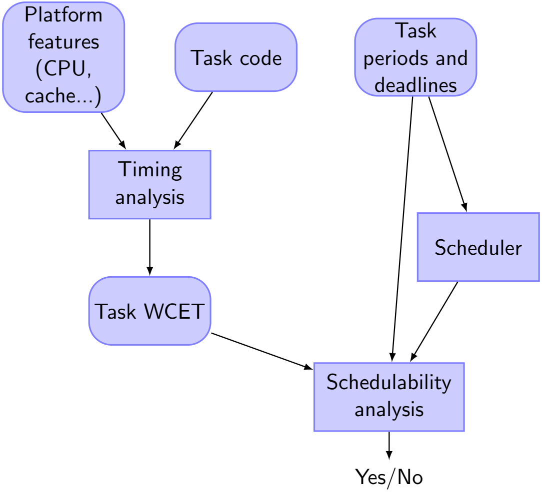
Goal:

Study the impact of CRPDs on hard real-time scheduling.

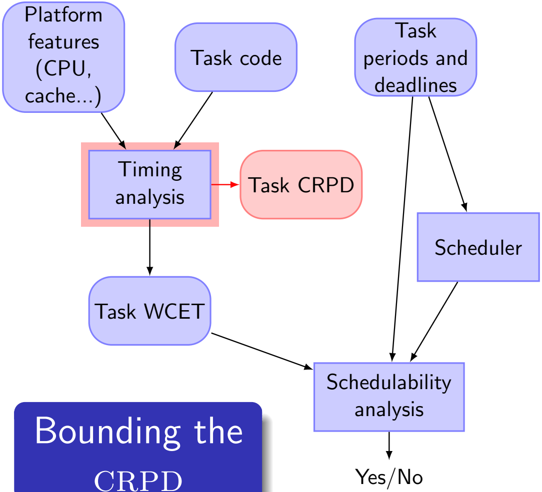
- **sustainability** of classic online scheduling policies subjected to CRPDs?
- **optimal** online CRPD-aware scheduling policy?
- **loss of schedulability** of classic online scheduling policies subjected to CRPDs?

Related Work

CRPDs in real-time scheduling



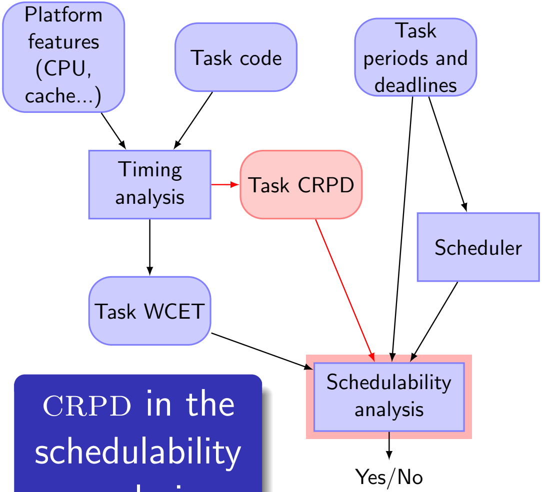
CRPDs in real-time scheduling



Bounding the CRPD

- preempted task *(Lee et al. 1997)*
- preempting task *(Busquets-Mataix et al. 1996)*
- combined approaches *(Altmeyer et al. 2012)*

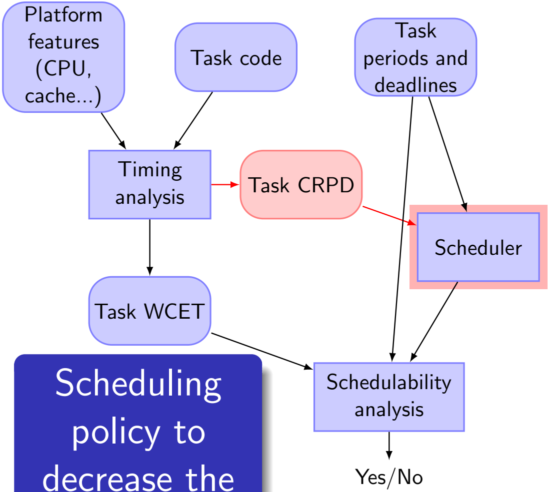
CRPDs in real-time scheduling



CRPD in the schedulability analysis

- FP scheduling → Response Time Analysis *(Busquets-Mataix et al. 1996)*
- EDF → Demand Bound Function *(Lunniss et al. 2013)*

CRPDs in real-time scheduling



Scheduling policy to decrease the CRPD

- FPP placement *(Bertogna et al. 2011)*
- optimal CRPD-aware scheduling:
  - ↪ NP-hard in the strong sense *(Phavorin et al. 2015)*

Online scheduling with CRPDs

CRPD-aware task model

Periodic synchronously released tasks

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

- C_i: WCET without CRPD
  - $\hookrightarrow$ τ_i executed fully non preemptively
- T_i: period
- D_i: relative deadline
  - $\hookrightarrow$ implicit deadline $D_i = T_i$ or constrained deadline $D_i \leq T_i$
- γ_i: CRPD paid by τ_i each time it resumes its execution after a preemption
  - $\hookrightarrow$ max. delay for every possible preemption point in the task code

$\rightarrow$ infinite sequence of jobs: $\tau_{ij}(r_{ij} = (j-1) \cdot T_i, C_i, d_{ij} = r_{ij} + D_i, \gamma_i)$

Known results

Online scheduling with CRPDs:

- EDF and FP scheduling algorithms (RM, DM) $\rightarrow$ not optimal *(Phavorin et al. 2015)*
- for FP scheduling: synchronous releases $\rightarrow$ not necessarily the critical instant worst-case scenario *(Ramaprasad et al. 2006, Meumeu et al. 2007)*

Sustainability

Sustainability: *(Burns et al. 2008)*

A scheduling policy is *sustainable* if any system deemed schedulable remains schedulable if:

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

Online scheduling **without** CRPD:

- EDF $\rightarrow$ w.r.t: WCET, deadline, period
- FP scheduling policies (RM, DM) $\rightarrow$ sustainable w.r.t.: WCET, deadline BUT NOT period

Sustainability

Sustainability:

A scheduling policy is *sustainable* if any system deemed schedulable remains schedulable if:

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

Online scheduling **without** CRPD:

Online scheduling with CRPDs
⇒ EDF and FP scheduling **NOT SUSTAINABLE**

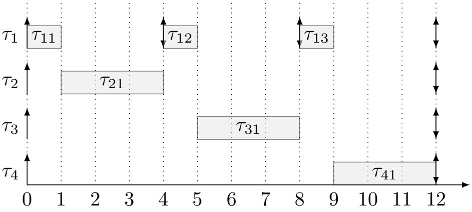
deadline BUT NOT period

Sustainability w.r.t the WCET

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

- $\tau_1(1, 4, 4, 0.6)$, $\tau_2(3, 12, 12, 0.6)$, $\tau_3(3, 12, 12, 0.6)$, $\tau_4(2, 12, 12, 0.6)$

  $\hookrightarrow$ EDF, RM, DM $\rightarrow$ same job priority assignment (task index as tie breaker).



$\rightarrow$ schedule with $C_2 = 3$

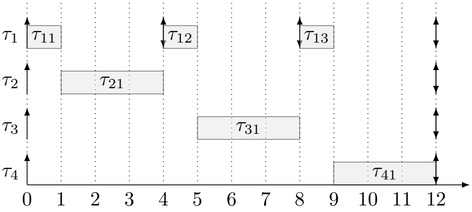
Sustainability w.r.t the WCET

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

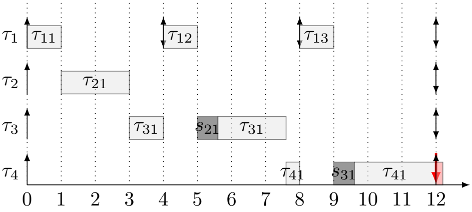
- $\tau_1(1, 4, 4, 0.6)$, $\tau_2(3, 12, 12, 0.6)$, $\tau_3(3, 12, 12, 0.6)$, $\tau_4(2, 12, 12, 0.6)$

  $\hookrightarrow$ EDF, RM, DM $\rightarrow$ same job priority assignment (task index as tie breaker).

➢ $\searrow$ 1 task execution time ($C_2 = 3 \rightarrow C_2 = 2$)



$\rightarrow$ schedule with $C_2 = 3$



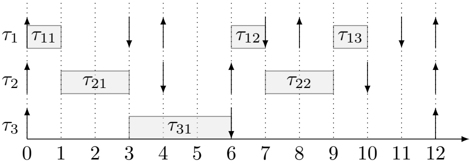
$\rightarrow$ schedule with $C_2 = 2$

Sustainability w.r.t the deadline

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

- $\tau_1(1,3,4,1)$, $\tau_2(2,4,6,1)$, $\tau_3(3,6,12,1)$

  $\hookrightarrow$ EDF $\rightarrow$ same job priority assignment (task index as tie breaker).



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

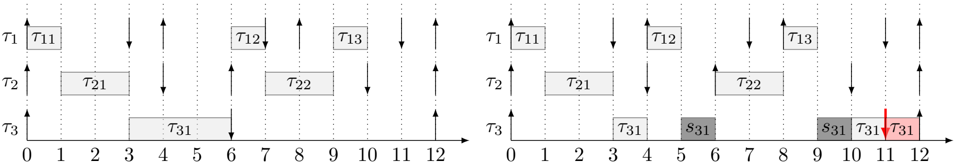
Sustainability w.r.t the deadline

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

- $\tau_1(1,3,4,1)$, $\tau_2(2,4,6,1)$, $\tau_3(3,6,12,1)$

  $\hookrightarrow$ EDF $\rightarrow$ same job priority assignment (task index as tie breaker).

➢ $\nearrow$ 1 task relative deadline ($D_3 = 6 \rightarrow D_3 = 11$)



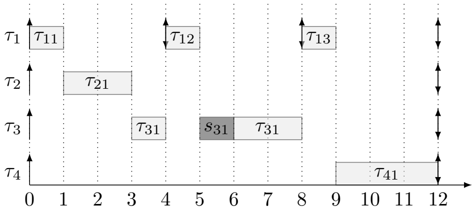
$\rightarrow$ EDF schedule with $D_3 = 6$ $\rightarrow$ EDF schedule with $D_3 = 11$

Sustainability w.r.t the CRPD

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

- $\tau_1(1,4,4,1)$, $\tau_2(3,12,12,1)$, $\tau_3(3,12,12,1)$, $\tau_4(2,12,12,1)$

  $\hookrightarrow$ EDF, RM, EDF $\rightarrow$ same job priority assignment (task index as tie breaker).



$\rightarrow$ schedule with $\gamma_3 = 1$

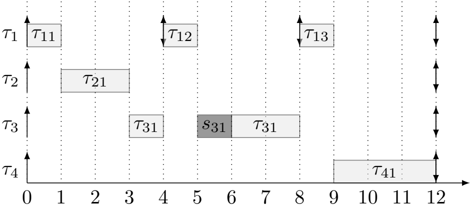
Sustainability w.r.t the CRPD

$\tau_i(C_i, D_i, T_i, \gamma_i)$:

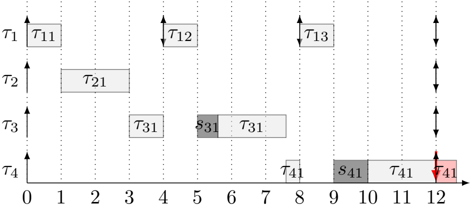
- $\tau_1(1,4,4,1)$, $\tau_2(3,12,12,1)$, $\tau_3(3,12,12,1)$, $\tau_4(2,12,12,1)$

  $\hookrightarrow$ EDF, RM, EDF $\rightarrow$ same job priority assignment (task index as tie breaker).

➢ $\searrow$ 1 task CRPD ($\gamma_3 = 1 \rightarrow \gamma_3 = 0.6$)



$\rightarrow$ schedule with $\gamma_3 = 1$



$\rightarrow$ schedule with $\gamma_3 = 0.6$

Online scheduling of a set of jobs

- Online scheduling model:
  - ➢ set of jobs released over time
  - ➢ at each job release, all its parameters are known

→ *optimal online scheduling policy?*

Online scheduling of a set of jobs

- Online scheduling model:
  - ➢ set of jobs released over time
  - ➢ at each job release, all its parameters are known

$\rightarrow$ *optimal online scheduling policy?*

Result: Optimal online scheduling is impossible

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

Proof

Proof sketch

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

Jobs: τ_1(0,5,12,1), τ_2(4,5,10,1) $\rightarrow$ scheduling decision at $t = 4$

Adversary strategy:

- At time 4:

  Case 1: the online scheduler **continues** to execute τ_1
  $\hookrightarrow$ the adversary generates a new job $\underline{\tau_3(9,1,10,1)}$

  Case 2: the online scheduler **preempts** τ_1 to execute τ_2
  $\hookrightarrow$ the adversary generates a new job $\underline{\tau_3(10,1,11,1)}$

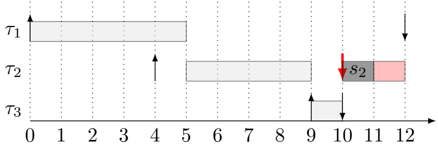
Proof

Proof sketch

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

Jobs: τ_1(0,5,12,1), τ_2(4,5,10,1) $\rightarrow$ scheduling decision at $t = 4$

Case 1. The online scheduler continues to execute job τ_1 at time 4. Adversary generates a job τ_3(9,1,10,1).



$\rightarrow$ Online algorithm

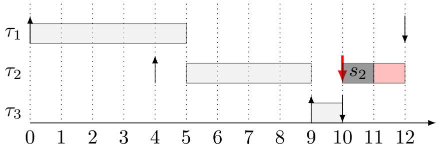
Proof

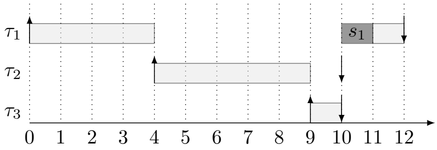
Proof sketch

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

Jobs: τ_1(0,5,12,1), τ_2(4,5,10,1) $\rightarrow$ scheduling decision at $t = 4$

Case 1. The online scheduler continues to execute job τ_1 at time 4. Adversary generates a job τ_3(9,1,10,1).





$\rightarrow$ Online algorithm

$\rightarrow$ Adversary's feasible schedule

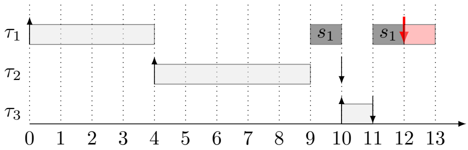
Proof

Proof sketch

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

Jobs: τ_1(0,5,12,1), τ_2(4,5,10,1) $\rightarrow$ scheduling decision at $t = 4$

Case 2. the online scheduler preempts τ_1 to execute at time 4 τ_2. Adversary generates a job τ_3(10,1,11,1).



$\rightarrow$ Online algorithm

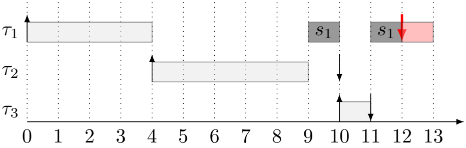
Proof

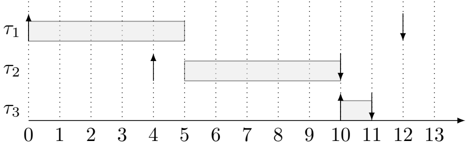
Proof sketch

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

Jobs: τ_1(0,5,12,1), τ_2(4,5,10,1) $\rightarrow$ scheduling decision at $t = 4$

Case 2. the online scheduler preempts τ_1 to execute at time 4 τ_2. Adversary generates a job τ_3(10,1,11,1).





$\rightarrow$ Online algorithm

$\rightarrow$ Adversary's feasible schedule

An offline solution

Offline scheduling

- set of tasks $\tau_i(C_i, D_i, T_i, \gamma_i)$

  $\hookrightarrow$ find a valid schedule whenever it is possible.

Offline scheduling

- set of tasks $\tau_i(C_i, D_i, T_i, \gamma_i)$

  $\hookrightarrow$ find a valid schedule whenever it is possible.

$\rightarrow$ Mixed Integer Linear Program (MILP) formulation

Objective function $\rightarrow$ define an offline schedule to:

- minimize the total workload
- *or equivalently,* minimize the total CRPD (since the WCET contributes as a constant in the objective function)

MILP formulation

Schedule construction:

- schedule:
  - ➢ finite set of slices S_j,
  - ➢ separated by releases/deadlines
  - $\Rightarrow$ no job release inside a slice
- in every slice:
  - $\rightarrow$ job-piece execution times + related CRPDs must fit in the slice interval

Every job resumes at most once in every slice.

MILP formulation

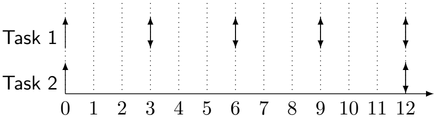
Schedule construction:

- schedule:
  - ➢ finite set of slices S_j,
  - ➢ separated by releases/deadlines
  - ⇒ <u>no job release inside a slice</u>
- in every slice:
  - → job-piece execution times + related CRPDs <u>must fit</u> in the slice interval

Every job resumes <u>at most once</u> in every slice.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



MILP formulation

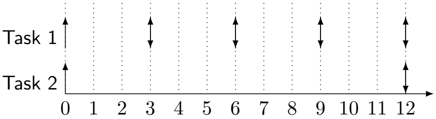
Schedule construction:

- schedule:
  - ➢ finite set of slices S_j,
  - ➢ separated by releases/deadlines
  - ⇒ no job release inside a slice
- in every slice:
  - → job-piece execution times + related CRPDs must fit in the slice interval

Every job resumes at most once in every slice.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



slice
1
1
2
2
3
3
4
4

→ 4 Slices:

- ↪ S_1=[0, 3)
- ↪ S_2=[3,6)
- ↪ S_3=[6,9)
- ↪ S_4=[9,12)

MILP formulation

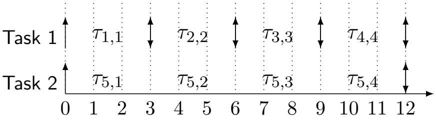
Schedule construction:

- schedule:
  - ➢ finite set of slices S_j,
  - ➢ separated by releases/deadlines
  - ⇒ no job release inside a slice
- in every slice:
  - → job-piece execution times + related CRPDs must fit in the slice interval

Every job resumes at most once in every slice.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



slice	job
1	τ_1
1	τ_5
2	τ_5
2	τ_2
3	τ_3
3	τ_5
4	τ_5
4	τ_4

→ 4 Slices:

- ↪ S_1=[0, 3)
- ↪ S_2=[3,6)
- ↪ S_3=[6,9)
- ↪ S_4=[9,12)

→ 8 job-pieces

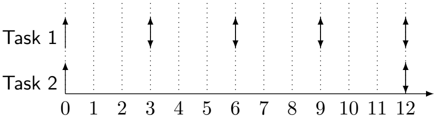
MILP formulation

Schedule construction:

- MILP variables fore each slice S_j:
  - $t_{i,j} \in \mathbb{R} \rightarrow$ starting time of job-piece τ_i in S_j
  - $p_{i,j} \in \mathbb{R} \rightarrow$ execution time of job-piece τ_i in S_j
  - $\Delta_{i,j} \in \{0,1\} \rightarrow$ job-piece has to pay a CRPD in S_j.

Exple: 2 periodic tasks within [0,12)

- Task 1 (1,3,3,0.2)
- Task 2 (7,12,12,0.5)



job	slice	$t_{i,j}$	$p_{i,j}$	$\Delta_{i,j}$
1	τ_1			
1	τ_5			
2	τ_5			
2	τ_2			
3	τ_3			
3	τ_5			
4	τ_5			
4	τ_4			

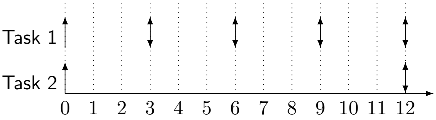
MILP formulation

Schedule construction:

- MILP variables fore each slice S_j:
  - $t_{i,j} \in \mathbb{R} \rightarrow$ starting time of job-piece τ_i in S_j
  - $p_{i,j} \in \mathbb{R} \rightarrow$ execution time of job-piece τ_i in S_j
  - $\Delta_{i,j} \in \{0,1\} \rightarrow$ job-piece has to pay a CRPD in S_j.

Exple: 2 periodic tasks within [0,12)

- Task 1 (1,3,3,0.2)
- Task 2 (7,12,12,0.5)



Constraints $\rightarrow$ construct a valid schedule:

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

job	slice	$t_{i,j}$	$p_{i,j}$	$\Delta_{i,j}$
1	τ_1	0	1	0
1	τ_5	1	2	0
2	τ_5	3	2	0
2	τ_2	5	1	0
3	τ_3	6	1	0
3	τ_5	7.5	1	1
4	τ_5	9	2	0
4	τ_4	11	1	0

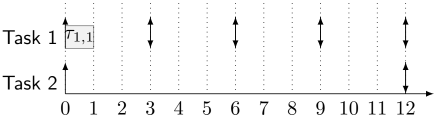
MILP formulation

Schedule construction:

- MILP variables fore each slice S_j:
  - ➢ $t_{i,j} \in \mathbb{R} \rightarrow$ starting time of job-piece τ_i in S_j
  - ➢ $p_{i,j} \in \mathbb{R} \rightarrow$ execution time of job-piece τ_i in S_j
  - ➢ $\Delta_{i,j} \in \{0,1\} \rightarrow$ job-piece has to pay a CRPD in S_j.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



Constraints $\rightarrow$ construct a valid schedule:

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

job	slice	$t_{i,j}$	$p_{i,j}$	$\Delta_{i,j}$
1	τ_1	0	1	0
1	τ_5	1	2	0
2	τ_5	3	2	0
2	τ_2	5	1	0
3	τ_3	6	1	0
3	τ_5	7.5	1	1
4	τ_5	9	2	0
4	τ_4	11	1	0

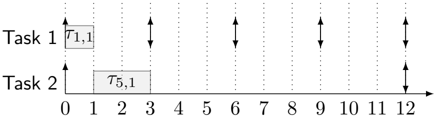
MILP formulation

Schedule construction:

- MILP variables fore each slice S_j:
  - ➢ $t_{i,j} \in \mathbb{R} \rightarrow$ starting time of job-piece τ_i in S_j
  - ➢ $p_{i,j} \in \mathbb{R} \rightarrow$ execution time of job-piece τ_i in S_j
  - ➢ $\Delta_{i,j} \in \{0,1\} \rightarrow$ job-piece has to pay a CRPD in S_j.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



Constraints $\rightarrow$ construct a valid schedule:

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

job	slice	$t_{i,j}$	$p_{i,j}$	$\Delta_{i,j}$
1	τ_1	0	1	0
1	τ_5	1	2	0
2	τ_5	3	2	0
2	τ_2	5	1	0
3	τ_3	6	1	0
3	τ_5	7.5	1	1
4	τ_5	9	2	0
4	τ_4	11	1	0

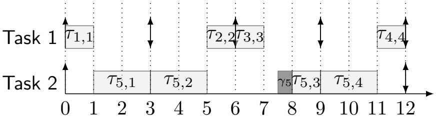
MILP formulation

Schedule construction:

- MILP variables fore each slice S_j:
  - ➢ $t_{i,j} \in \mathbb{R} \rightarrow$ starting time of job-piece τ_i in S_j
  - ➢ $p_{i,j} \in \mathbb{R} \rightarrow$ execution time of job-piece τ_i in S_j
  - ➢ $\Delta_{i,j} \in \{0,1\} \rightarrow$ job-piece has to pay a CRPD in S_j.

Exple: 2 periodic tasks within [0,12)

- ➢ Task 1 (1,3,3,0.2)
- ➢ Task 2 (7,12,12,0.5)



Constraints $\rightarrow$ construct a valid schedule:

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

job	slice	$t_{i,j}$	$p_{i,j}$	$\Delta_{i,j}$
1	τ_1	0	1	0
1	τ_5	1	2	0
2	τ_5	3	2	0
2	τ_2	5	1	0
3	τ_3	6	1	0
3	τ_5	7.5	1	1
4	τ_5	9	2	0
4	τ_4	11	1	0

Evaluation

Experiments

Goal:

Evaluate the loss of schedulability of classic online scheduling policies.

Experiments

Goal:

Evaluate the loss of schedulability of classic online scheduling policies.

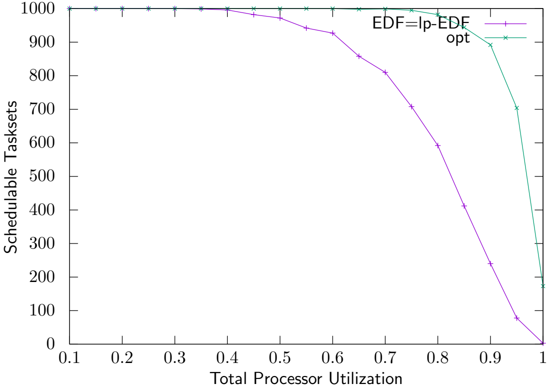
- Synthetic tasksets:
  - $C_i, T_i \rightarrow$ UUnifast *(Bini et al. 2005)*
    - $\hookrightarrow$ to generate processor utilization factors
  - $\gamma_i \rightarrow$ maximum CRPD Factor (PDF): % of C_i
    - $\hookrightarrow \gamma_i = \mathrm{PDF} \times C_i$
  - limited to 200 jobs over the hyperperiod
    - $\hookrightarrow$ to limit the MILP solving time
- Monitored algorithms:
  - EDF: arbitrary tie breaker
  - LP-EDF: tie breaker avoiding unnecessary preemptions

  } EDF schedulability analysis $\rightarrow$ *Lunniss et al. 2013*
  - OPT: MILP solved using CPLEX 12.6.1

Schedulability

Experiment parameters:

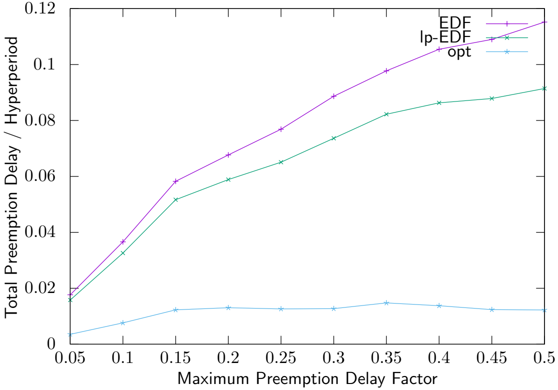
- maximum CRPD Factor $(\mathrm{PDF}) = 20\%$,
- $\#$ of schedulable tasksets as a function of the total processor utilization.



Total CRPD

Experiment parameters:

- Total Processor Utilization = 0.8,
- Total CRPD over the hyperperiod as a function of maximum PDF.



Conclusions and Future Work

- **Conclusions:**
  - scheduling with CRPD $\rightarrow$ several issues:
    - ➢ classic policies (EDF, RM, DM) **not sustainable**
    - ➢ **no optimal** online scheduling policy
  - optimal offline scheduling using a MILP formulation
    - ➢ evaluation of schedulability loss for EDF

Conclusions and Future Work

- **Conclusions:**
  - scheduling with CRPD $\rightarrow$ several issues:
    - ➢ classic policies (EDF, RM, DM) **not sustainable**
    - ➢ **no optimal** online scheduling policy
  - optimal offline scheduling using a MILP formulation
    - ➢ evaluation of schedulability loss for EDF

- **Future work:**
  - ➢ evaluation of schedulability loss for other policies/techniques
  - ➢ MILP with a more accurate CRPD parameter $\rightarrow$ **DIFFICULT**
  - ➢ online scheduling using heuristics

Thank you!

Questions?