Complexity of scheduling real-time tasks subjected to cache-related preemption delays

Guillaume PHAVORIN, Pascal RICHARD
LIAS/Univ. de Poitiers
guillaume.phavorin@univ-poitiers.fr
pascal.richard@univ-poitiers.fr

Claire MAIZA
Verimag/Univ. Grenoble-Alpes
claire.maiza@imag.fr

September 9th, 2015

ETFA'2015

Context

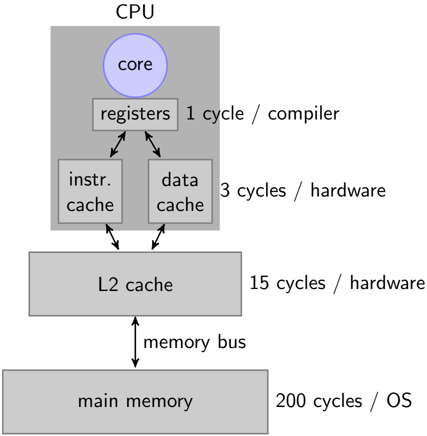
- more and more embedded applications
- $\nearrow$ processing needs
- use of Component off-the-shelf (COTS) → **cache**

Context

- more and more embedded applications
- $\nearrow$ processing needs
- use of Component off-the-shelf (COTS) → **cache**
- **hard real-time scheduling** → usually: preemption costs $= 0$

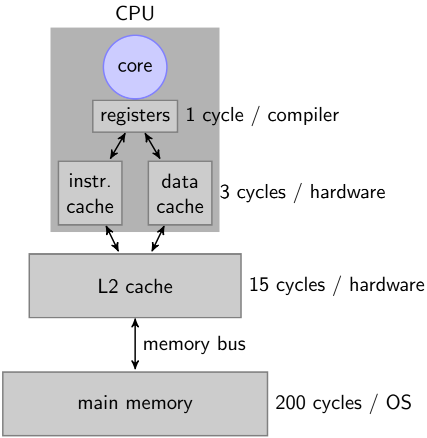
  $\hookrightarrow$ still valid with cache?

Cache

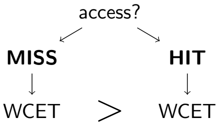


- bridge the gap between the CPU speed and the main memory access time
- cost(**cache miss**) $\gg$ cost(**cache hit**)

Cache



- bridge the gap between the CPU speed and the main memory access time
- cost(**cache miss**) $\gg$ cost(**cache hit**)
- *General assumptions:*
  - → only one instruction cache
  - → no timing anomaly:



Cache-Related Preemption Delays (CRPD)

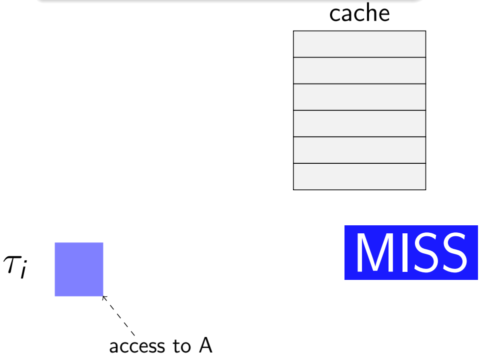
CRPD

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

Cache-Related Preemption Delays (CRPD)

CRPD

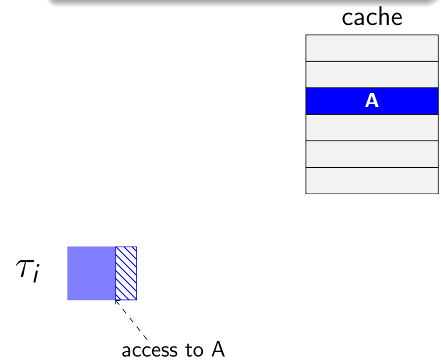
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

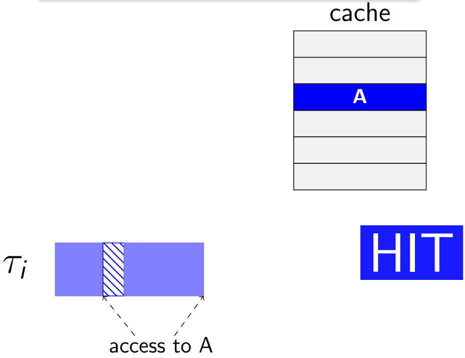
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

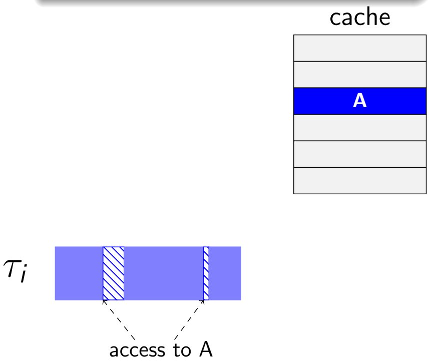
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

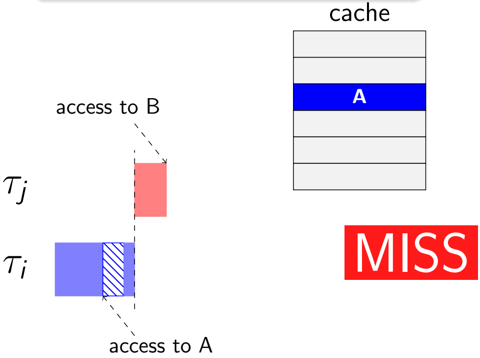
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

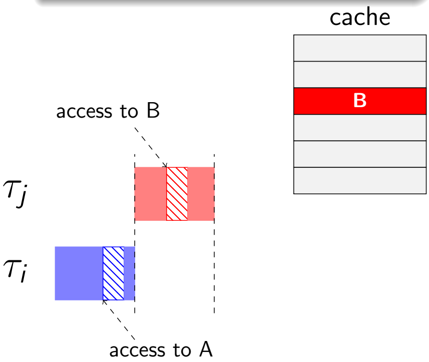
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

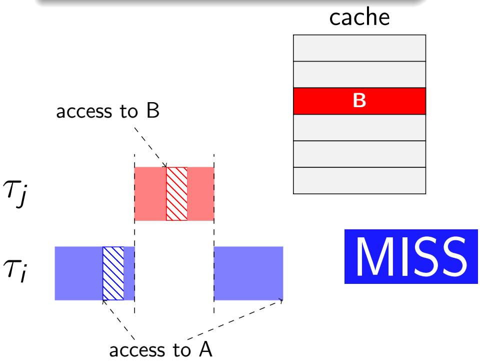
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

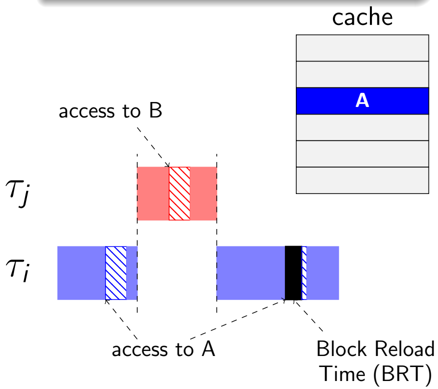
Additional reloads because of cache evictions due to **preempting jobs**



Cache-Related Preemption Delays (CRPD)

CRPD

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

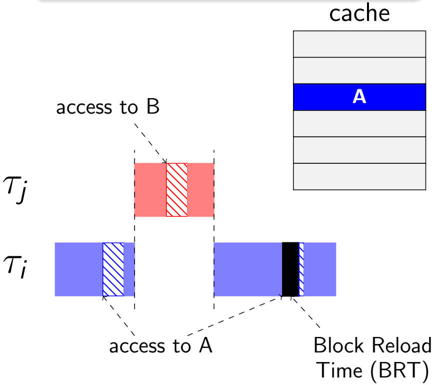


Cache-Related Preemption Delays (CRPD)

CRPD

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

- ↗ WCET

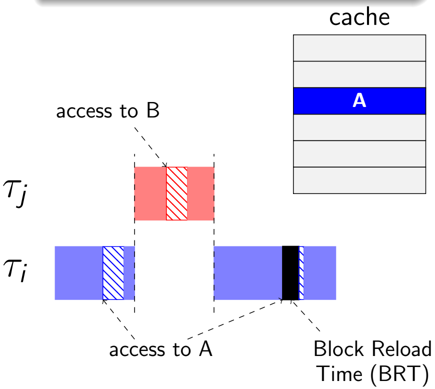


Cache-Related Preemption Delays (CRPD)

CRPD

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

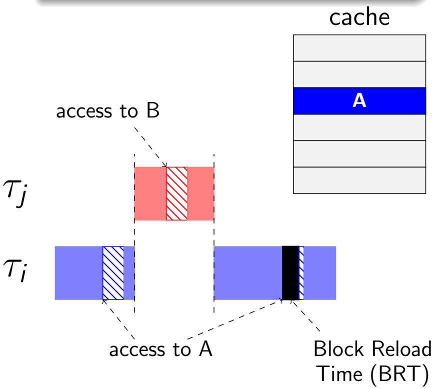
- ↗ WCET
- predictability?



Cache-Related Preemption Delays (CRPD)

CRPD

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



- ➢ ↗ WCET
- ➢ predictability?
- ➢ schedulability?

Outline

Goals:

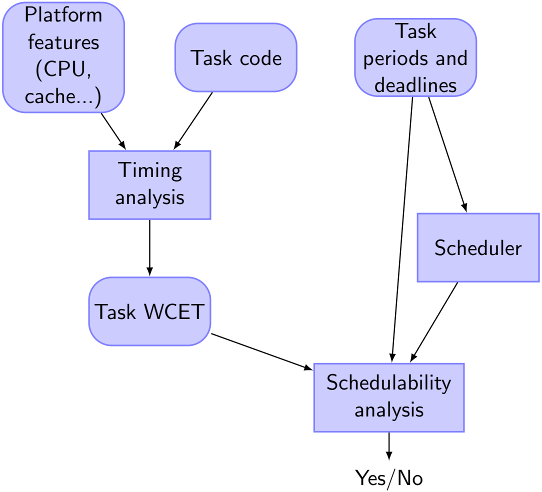
- studying the problem of scheduling hard real-time tasks subjected to cache-related preemption delays
- studying the complexity of taking scheduling decisions based on cache-related constraints

1. Related work
2. Scheduling problems
   - CRPD-aware scheduling problem
   - Cache-aware scheduling problem
3. Conclusion
   - Future works

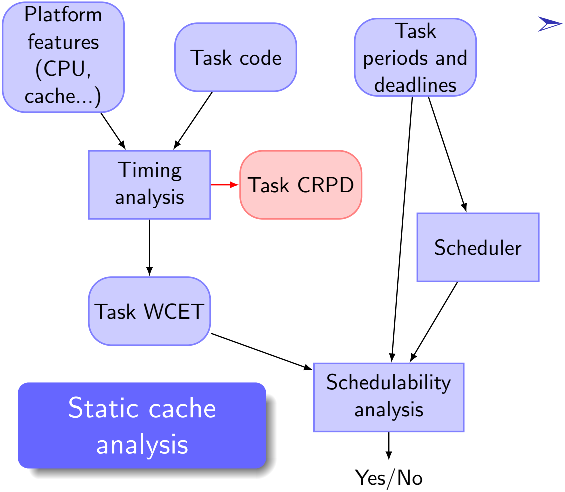
Related work

1. Related work
2. Scheduling problems
   - CRPD-aware scheduling problem
   - Cache-aware scheduling problem
3. Conclusion
   - Future works

Bounding the CRPD



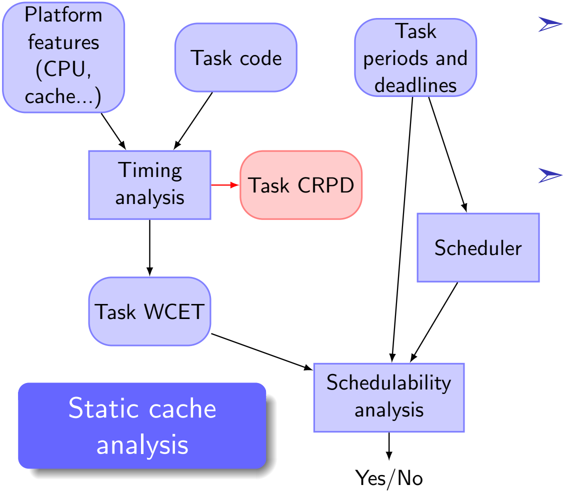
Bounding the CRPD



- preempted task
  ↪ Useful Cache Blocks (UCBs)

- *Lee et al., "Enhanced analysis of cache-related preemption delay in fixed-priority preemptive scheduling"*

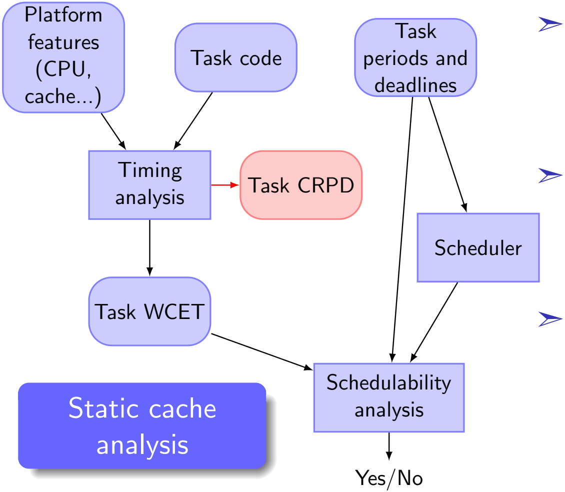
Bounding the CRPD



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

- *Busquets-Mataix et al., "Adding instruction cache effect to schedulability analysis of preemptive real-time systems"*

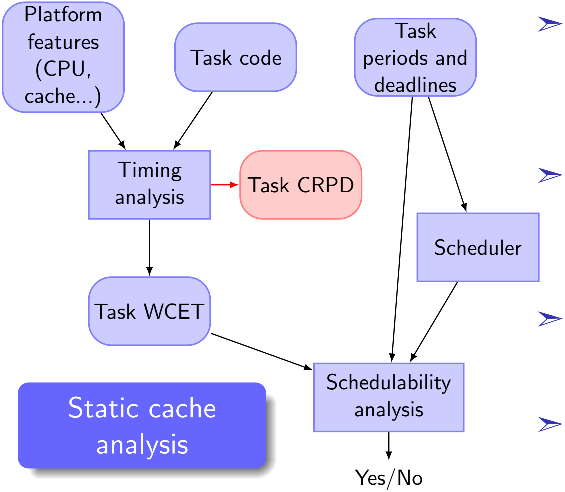
Bounding the CRPD



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

- *Altmeyer, Davis, and Maiza, "Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems"*

Bounding the CRPD

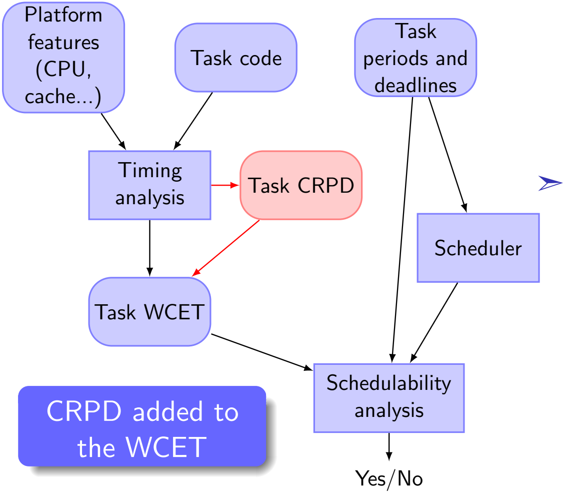


Static cache analysis

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

- *Altmeyer and Maiza-Burguière, "A New Notion of Useful Cache Block to Improve the Bounds of Cache-Related Preemption Delay"*

Bounding the CRPD

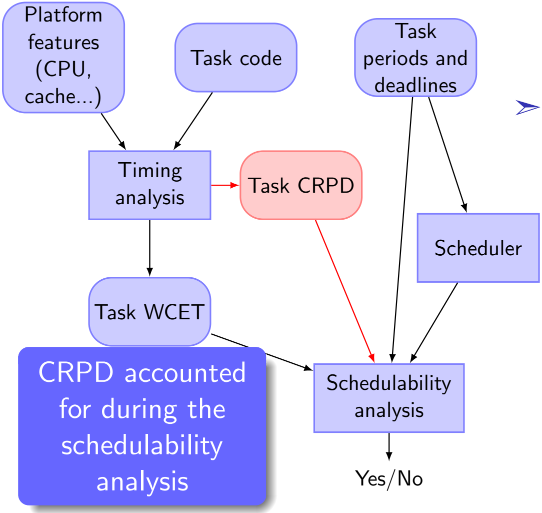


CRPD added to the WCET

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

- *Altmeyer and Burguière, "Cache-related preemption delay via useful cache blocks: Survey and redefinition"*

Bounding the CRPD



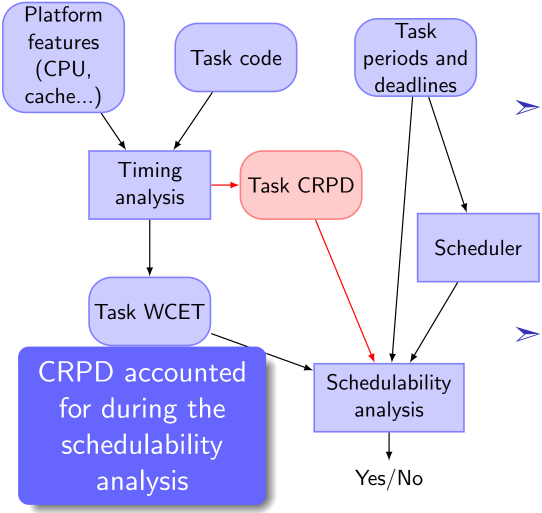
CRPD accounted for during the schedulability analysis

➢ Response Time Analysis:

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

➤ *Busquets-Mataix et al., "Adding instruction cache effect to schedulability analysis of preemptive real-time systems"*

Bounding the CRPD



CRPD accounted for during the schedulability analysis

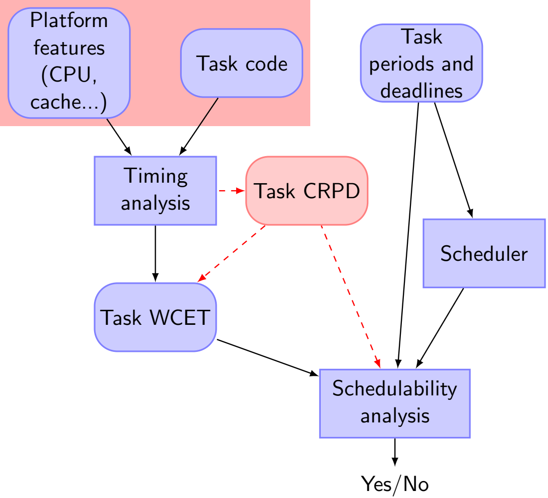
- Response Time Analysis:

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

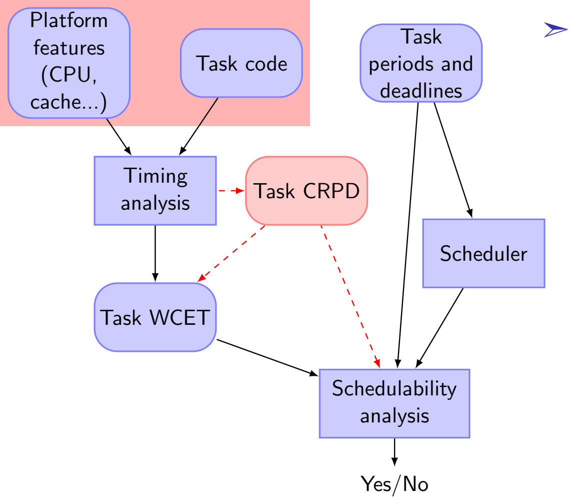
- EDF → time demand analysis.

- *Lunniss et al., "Integrating cache related pre-emption delay analysis into EDF scheduling"*

Cache management



Cache management



Platform features (CPU, cache...)

Task code

Task periods and deadlines

Timing analysis

Task CRPD

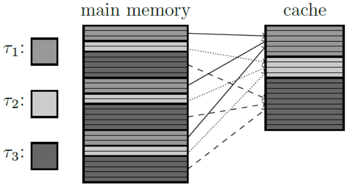
Scheduler

Task WCET

Schedulability analysis

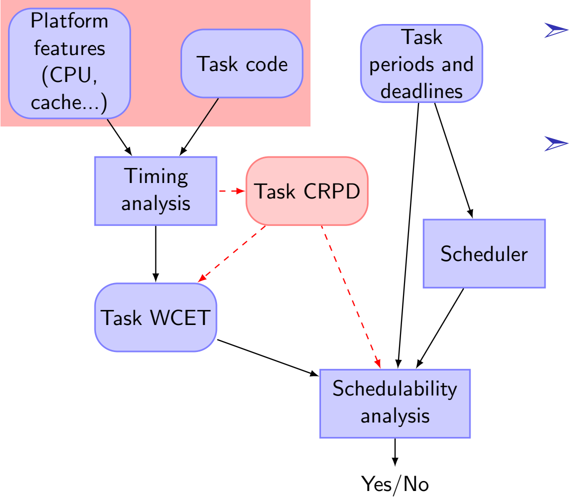
Yes/No

- cache partitioning



- *Bui et al., "Impact of cache partitioning on multi-tasking real time embedded systems"*

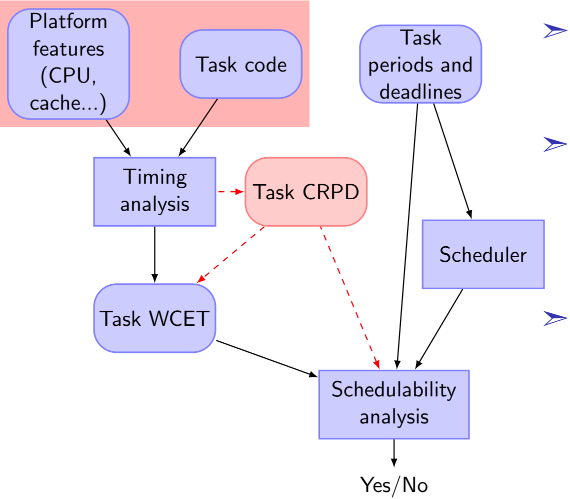
Cache management



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

- *Ding, Liang, and Mitra, "Integrated Instruction Cache Analysis and Locking in Multitasking Real-time Systems"*

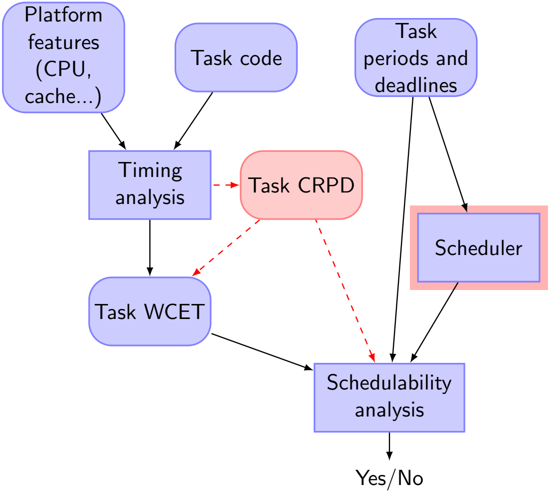
Cache management



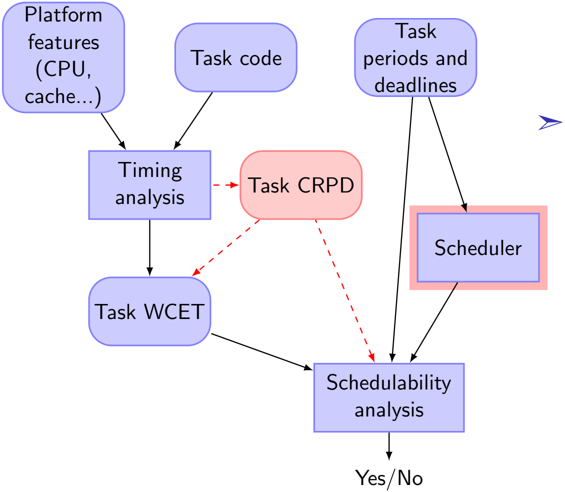
- cache partitioning
- cache locking
  → cache content fixed
  ⇒ predictability
- memory layout
  - code positioning
    ⇒ ↘ WCET
  - task positioning
    ⇒ ↘ CRPD

- *Lunniss, Altmeyer, and Davis, "Optimising Task Layout to Increase Schedulability via Reduced Cache Related Pre-emption Delays"*

Schedulability



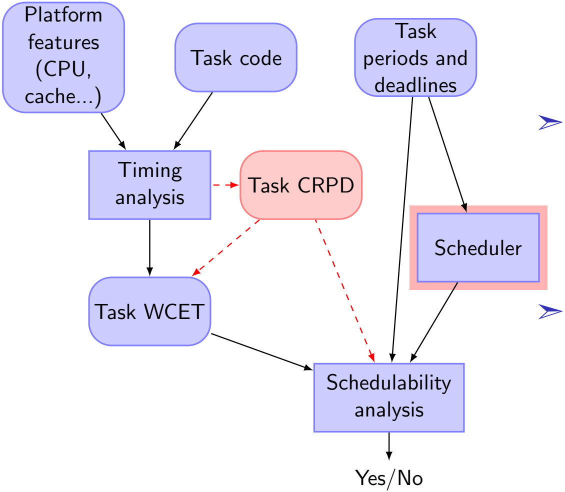
Schedulability



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

- *Bril et al., "Integrating Cache-Related Pre-Emption Delays into Analysis of Fixed Priority Scheduling with Pre-Emption Thresholds"*

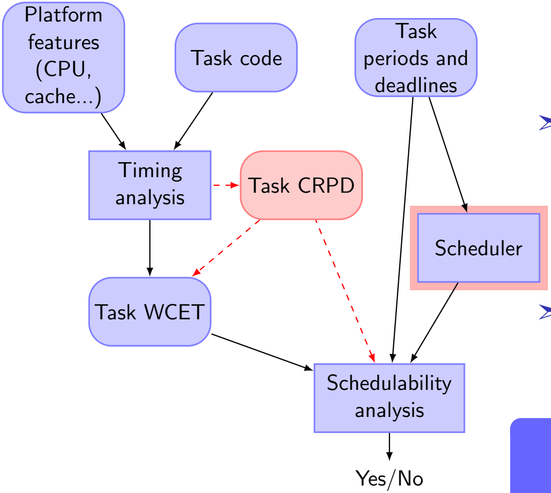
Schedulability



- preemption thresholds
  - ↪ preemption possible only if: priority(preempting task) > threshold(preempted task)
- deferred preemptions
  - ↪ preemption postponed as much as possible

- *Bertogna and Baruah, "Limited Preemption EDF Scheduling of Sporadic Task Systems"*

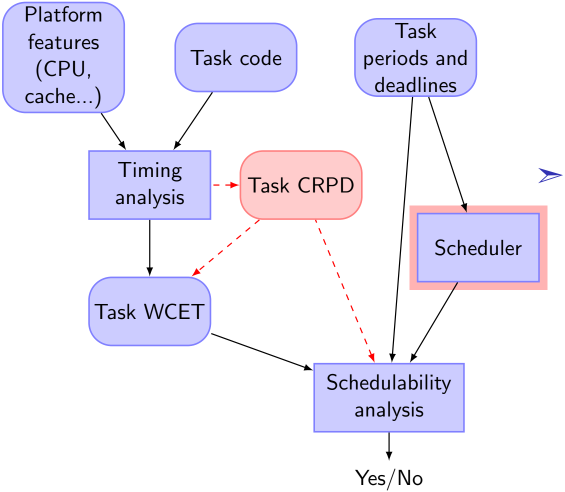
Schedulability



- ➢ preemption thresholds
  - ↪ preemption possible only if: priority(preempting task) > threshold(preempted task)
- ➢ deferred preemptions
  - ↪ preemption postponed as much as possible

Scheduling decisions are not directly based on a CRPD parameter.

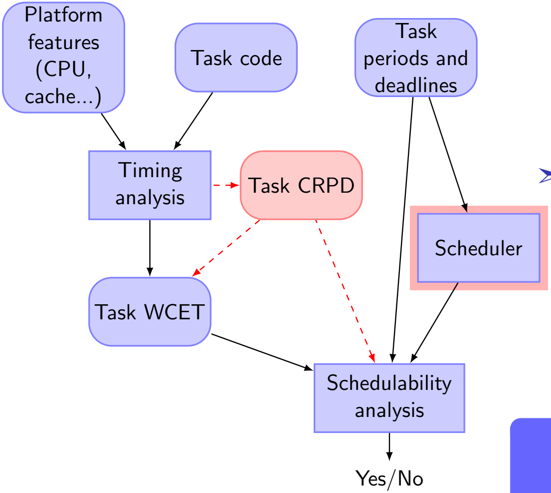
Schedulability



- Fixed Preemptive Points
  - ↪ preemption points chosen to minimize the CRPD

- *Bertogna et al., "Optimal selection of preemption points to minimize preemption overhead"*

Schedulability



- Fixed Preemptive Points
  - ↪ preemption points chosen to minimize the CRPD

Scheduling decisions are not directly based on a CRPD parameter.

Scheduling problems

1. Related work
2. Scheduling problems
   - CRPD-aware scheduling problem
   - Cache-aware scheduling problem
3. Conclusion
   - Future works

General approach

- **cache impact** on the computational complexity of *optimally* taking **scheduling decisions**

General approach

- **cache impact** on the computational complexity of *optimally* taking **scheduling decisions**

  ➢ 2 basic scheduling problems

  ↪ to cover the largest set of scheduling problems

General approach

- **cache impact** on the computational complexity of *optimally* taking **scheduling decisions**

➢ 2 basic scheduling problems

↪ to cover the largest set of scheduling problems

- scheduling with *cache-related preemption delays*

  → **CRPD-aware scheduling problem**

General approach

- **cache impact** on the computational complexity of *optimally* taking **scheduling decisions**

➢ 2 basic scheduling problems

↪ to cover the largest set of scheduling problems

- scheduling with *cache-related preemption delays*
  → **CRPD-aware scheduling problem**
- scheduling with *cache state information*
  → **Cache-aware scheduling problem**

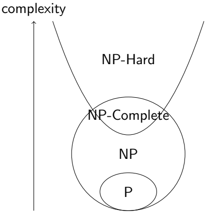
Computational complexity

- problem **classification**
  - ↪ **time** needed **to solve** problem instances of arbitrary size

Computational complexity

- problem **classification**
  - ↪ **time** needed **to solve** problem instances of arbitrary size

- complexity classes:
  - **NP-hard** in the *weak sense*
    - → at least *pseudo-polynomial time* algorithm
  - **NP-hard** in the *strong sense*
    - → at least *exponential time* algorithm

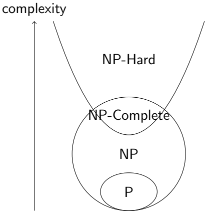


if **P** ≠ **NP**

Computational complexity

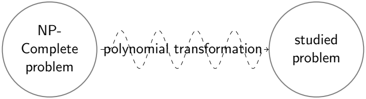
- problem **classification**
  - ↪ **time** needed **to solve** problem instances of arbitrary size

- complexity classes:
  - **NP-hard** in the *weak sense*
    - → at least *pseudo-polynomial time* algorithm
  - **NP-hard** in the *strong sense*
    - → at least *exponential time* algorithm



if **P** ≠ **NP**

- proof technique → polynomial reduction from a NP-complete problem



CRPD-aware scheduling problem

1. Related work
2. Scheduling problems
   - CRPD-aware scheduling problem
   - Cache-aware scheduling problem
3. Conclusion
   - Future works

CRPD-aware scheduling problem

Scheduling decisions taken based on cache-related preemption costs
→ **minimize the general overhead**.

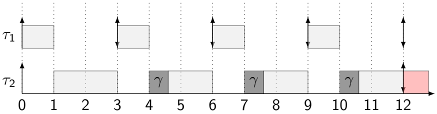
Task model: $\tau_i(C_i, D_i, T_i, \gamma)$

- C_i: WCET without preemption cost
  $\hookrightarrow$ τ_i executed fully non preemptively
- γ: CRPD for one preemption
  $\hookrightarrow$ the same for all program points and all tasks

Example: $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

Example: $\tau_1(1,3,3,0.6)$, $\tau_2(7,12,12,0.6)$

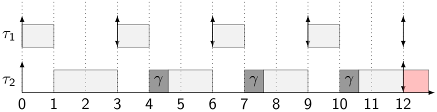
- Fixed-Task/Fixed-Job Priority Scheduling:



→ not schedulable

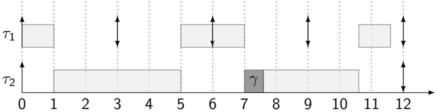
Example: $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

- Fixed-Task/Fixed-Job Priority Scheduling:



→ not schedulable

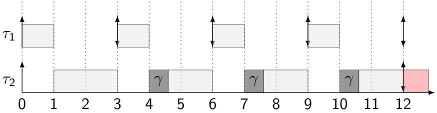
- CRPD-aware scheduling:



→ schedulable

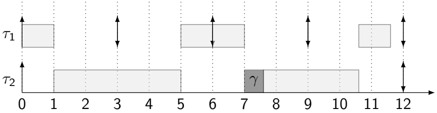
Example: $\tau_1(1, 3, 3, 0.6)$, $\tau_2(7, 12, 12, 0.6)$

- Fixed-Task/Fixed-Job Priority Scheduling:



→ not schedulable

- CRPD-aware scheduling:



→ schedulable

⇒ Fixed-Task and Fixed-Job Priority schedulers → **not optimal**.

Complexity result

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

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

Complexity result

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

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

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

Proof : transformation from the 3-Partition decision problem.

Cache-aware scheduling problem

1. Related work
2. Scheduling problems
   - CRPD-aware scheduling problem
   - Cache-aware scheduling problem
3. Conclusion
   - Future works

Cache-aware scheduling problem

Scheduling with cache state information
→ **maximize block reuse** by the different tasks.

↪ for only 1 task: Bélády's rule → optimal offline caching policy

Assumptions:

- a single cache line,
- synchronous jobs.

Job model: $J_i(C_i, D, S_i)$:

- C_i: WCET considering that all requested memory blocks are hits in the cache,
- D: relative deadline of the job → the same for all jobs,
- S_i: sequence of memory blocks used during the job execution
  ↪ no *if-then-else* structure

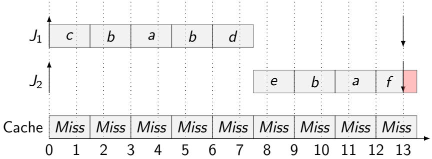
Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, Miss=Hit+0.5

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

Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, Miss=Hit+0.5

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

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

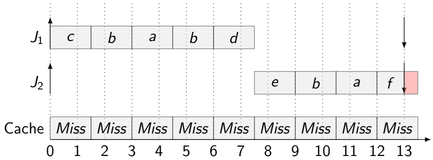


→ not schedulable

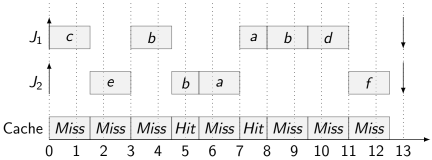
Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, Miss=Hit+0.5

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

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



→ not schedulable

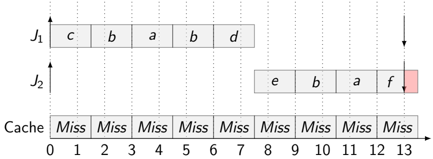


→ schedulable

Example: $J_1(5, 13, cbabd)$, $J_2(4, 13, ebaf)$, Miss=Hit+0.5

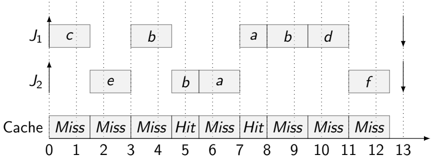
$$S_1 = c \to b \to a \to b \to d, \qquad S_2 = e \to b \to a \to f$$

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



→ not schedulable

- Cache-aware scheduling:



→ schedulable

⇒ Fixed-Task and Fixed-Job Priority schedulers → **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 : transformation from the Shortest Common Supersequence problem.

Conclusion & Future work

1. Related work
2. Scheduling problems
    - CRPD-aware scheduling problem
    - Cache-aware scheduling problem
3. Conclusion
    - Future works

Conclusion

- problem of **real-time scheduling** when dealing with **cache**
- *Cache-aware* scheduling problem
  - ↪ RM, EDF not optimal
  - ↪ **NP-hard** in the strong sense
    - ⇒ no pseudo-polynomial optimal scheduling algorithm
- *CRPD-aware* scheduling problem
  - ↪ RM, EDF not optimal
  - ↪ **NP-hard** in the strong sense
    - ⇒ no pseudo-polynomial optimal scheduling algorithm

Future work

Focus on the CRPD-aware scheduling problem

- ➢ use a simple (γ_i) linear programming to find an optimal solution → *will be presented at RNTS'2015*
- ➢ evaluate the loss of schedulability of different scheduling policies when CRPD are considered
  - Rate-Monotonic, EDF...
  - Preemption Thresholds, Deferred Preemptions

Thank you!
Questions?