# Tightening the Bounds on Cache-Related Preemption Delay in Fixed Preemption Point Scheduling

**Filip Marković**, Jan Carlson, Radu Dobrin
Mälardalen University, Sweden

MÄLARDALEN UNIVERSITY SWEDEN

EUROWEB+
European Research and Educational Collaboration with Western Balkan

# Content

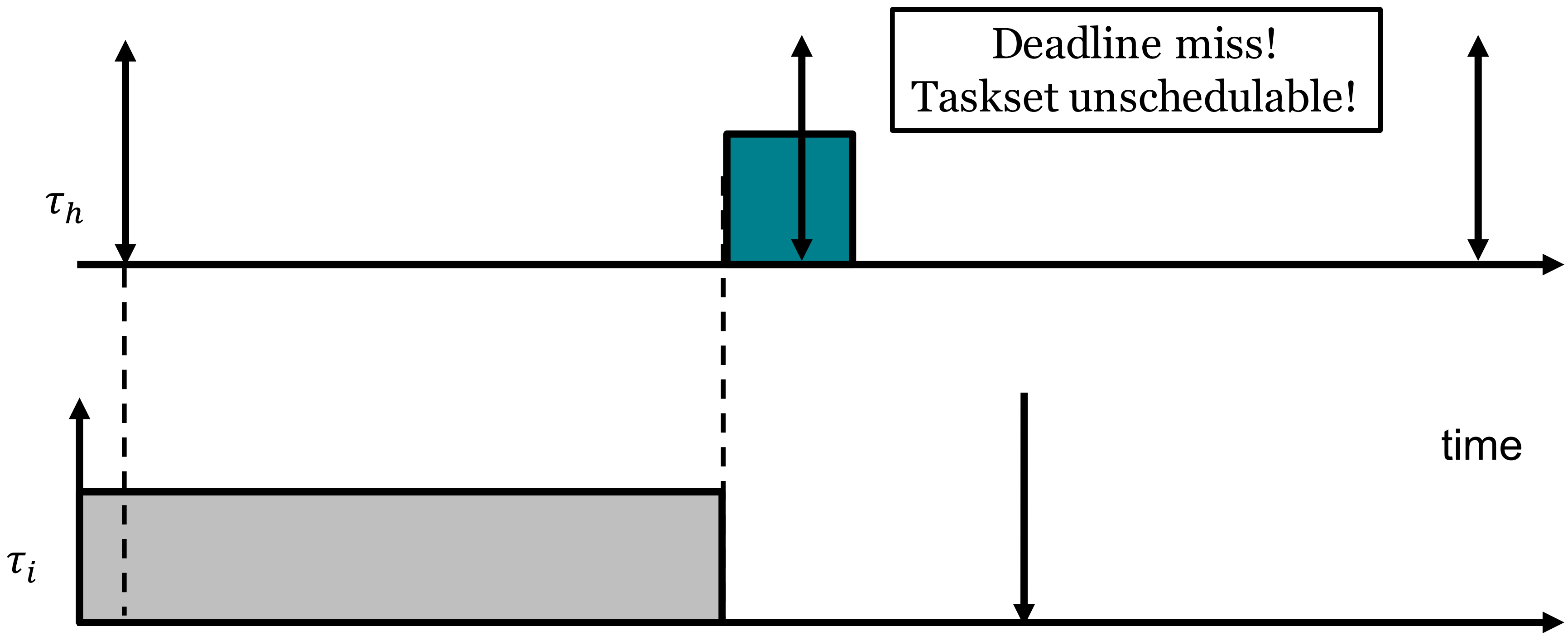
- Background and Motivation
- Problem formulation
- Proposed approach
- Evaluation
- Conclusions and Future Work

# Real-time systems

- **Non-preempive** vs **Fully-preemptive Scheduling**

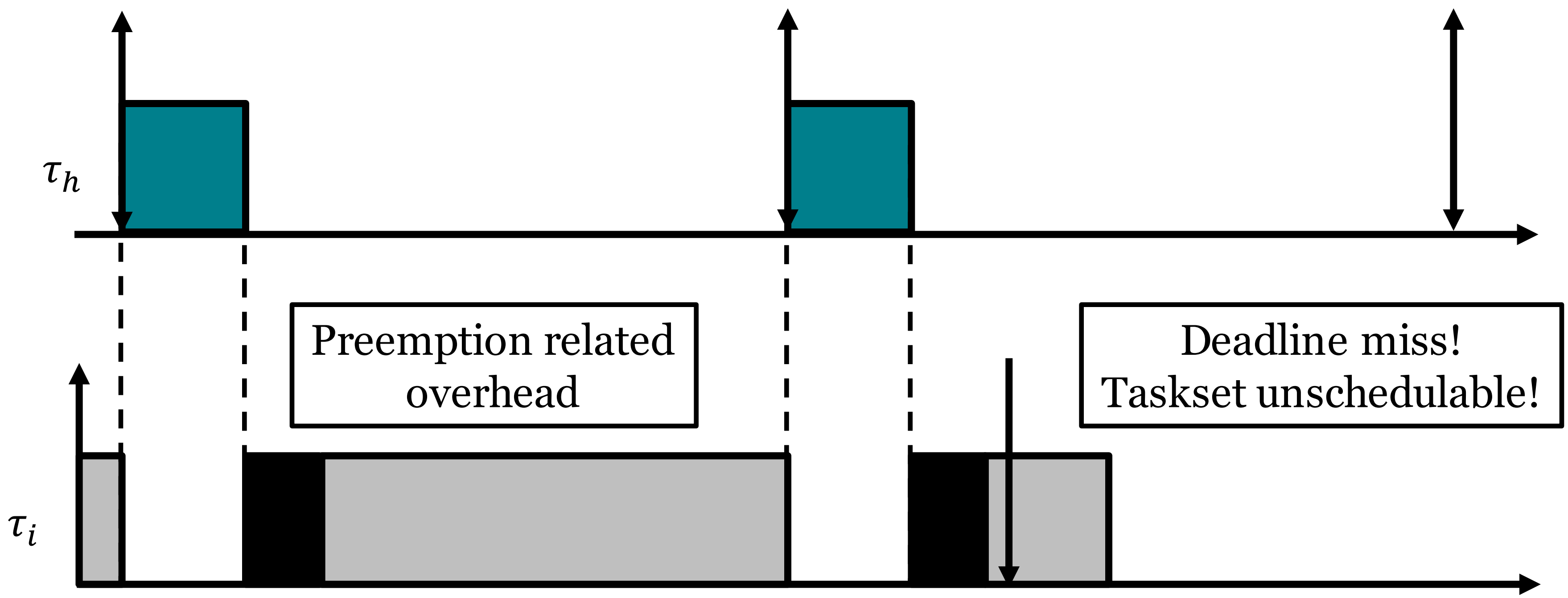
# Real time scheduling

- **Non-preemptive scheduling**



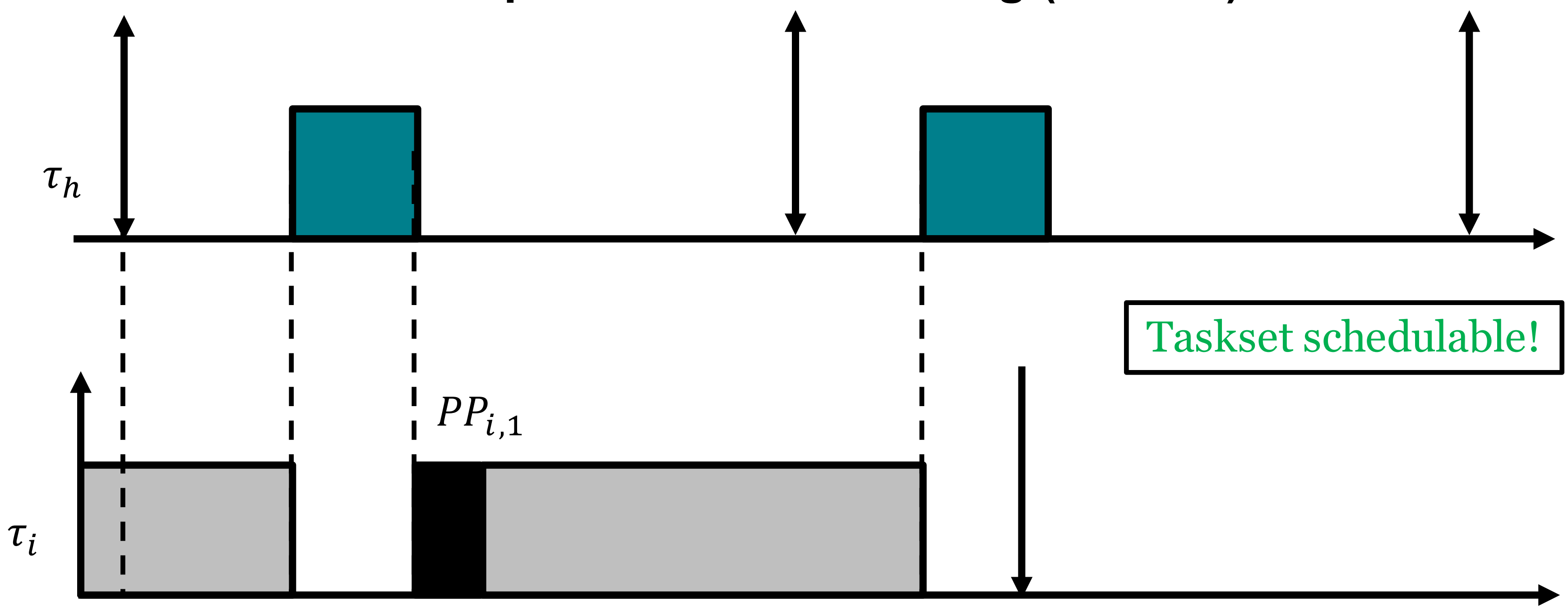
# Real time scheduling

- **Fully-preemptive scheduling**

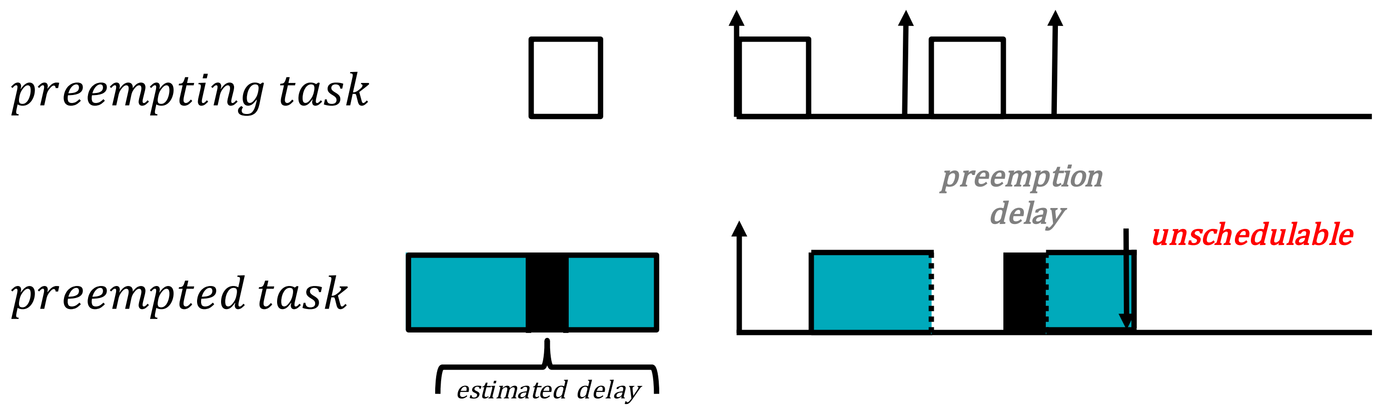


# Limited Preemptive Scheduling

- **Fixed Preemption Point Scheduling (LP-FPP)**



# Preemption-Related Delay




**Preemption**-related **delay** consists of different delay types: bus-related, scheduling-related, pipeline-related, etc.

**Cache-Related Preemption Delay** (CRPD) has the largest impact on preemption-related delay.

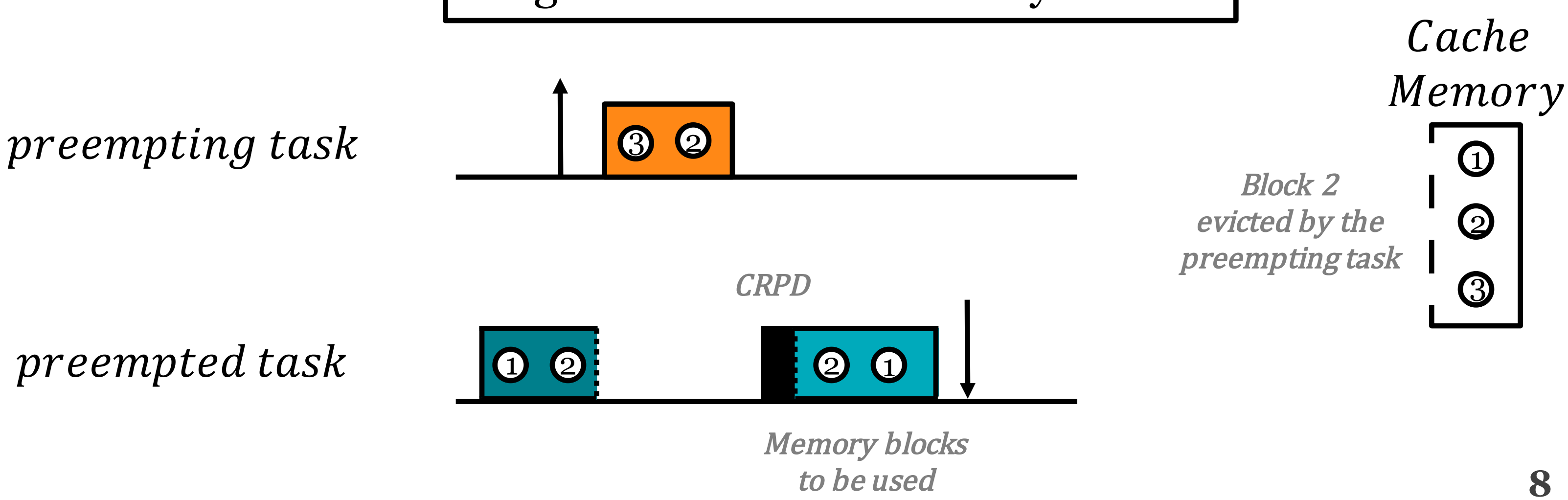
Therefore, it is important to accurately and as tightly as possible compute its upper bound.

# CRPD calculation

- **CRPD depends upon two important factors:**
  1. Where the preemption occurs?
  2. Which preempting tasks affect the CRPD at this point?

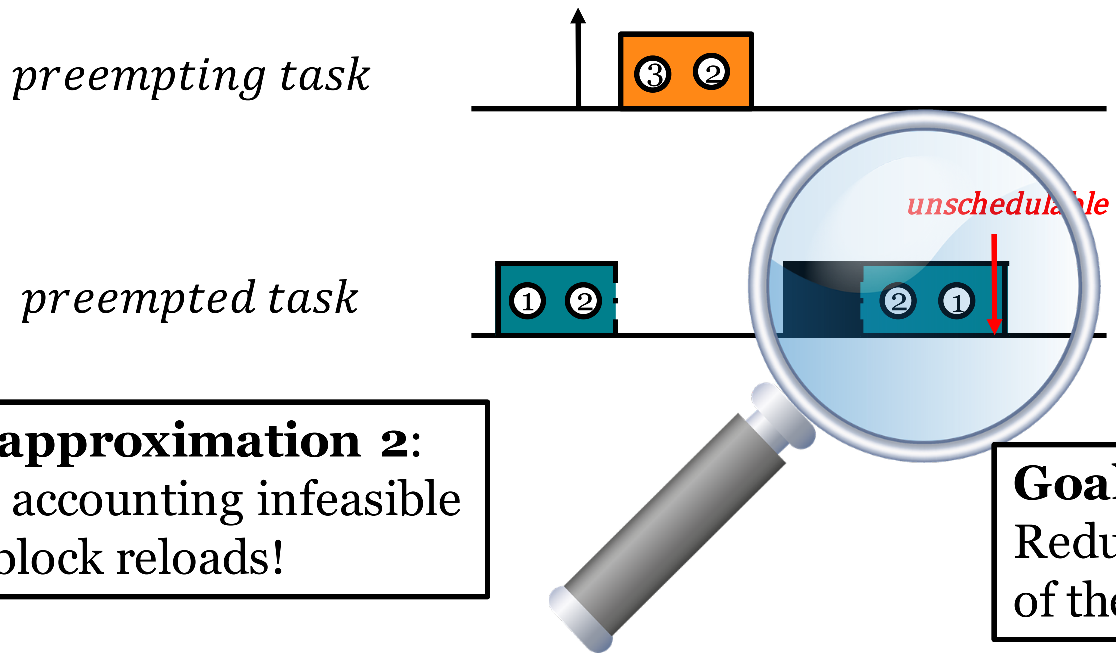
We compute CRPD by calculating
the maximum number of cache block reloads!

Single reload of the memory block 2



# Problem formulation

**Over-approximation 1**:
Due to accounting infeasible preemption combinations!

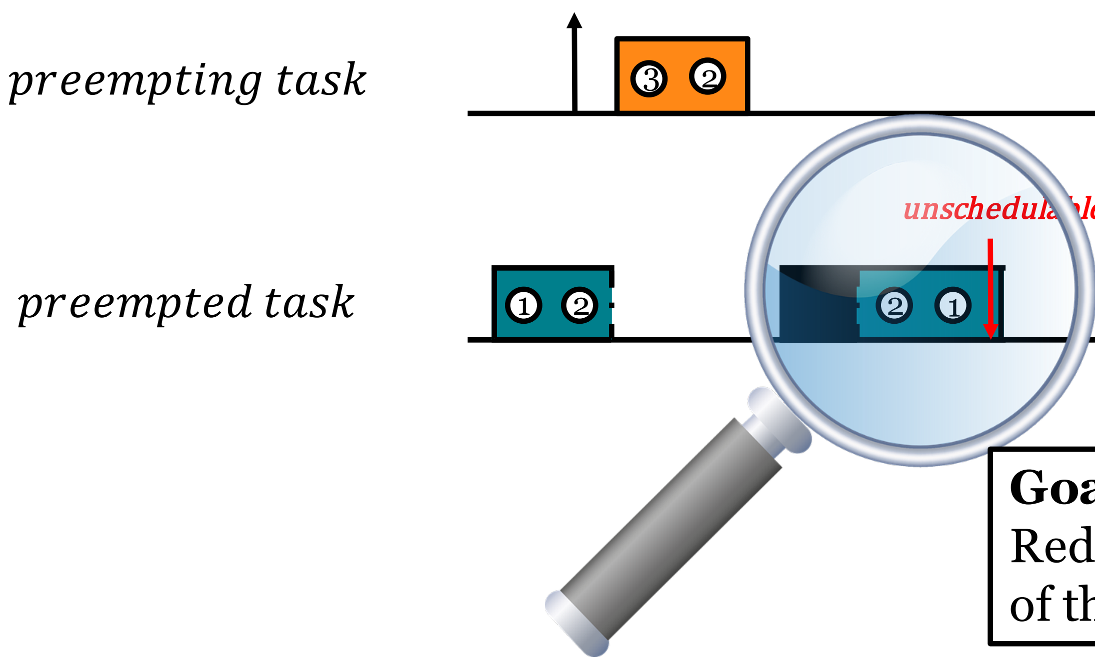



**Over-approximation 2**:
Due to accounting infeasible cache block reloads!

**Goal**:
Reduce the pessimism of the approximation!

# Proposed approach

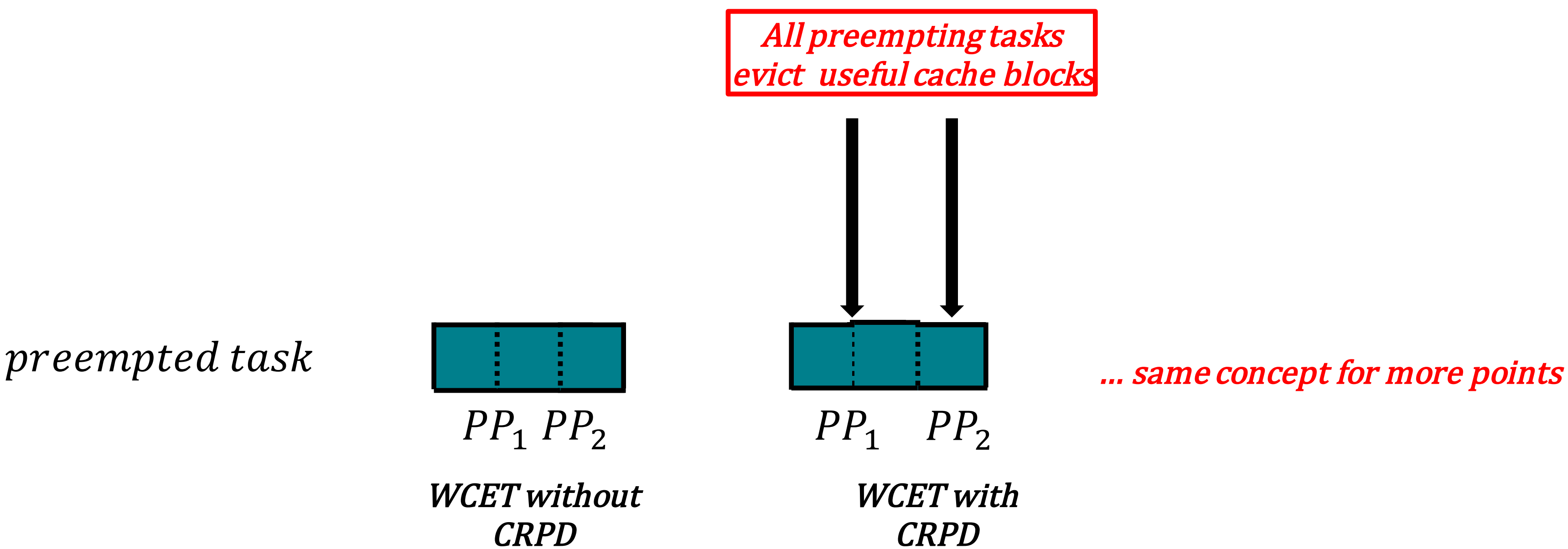
**How?**
By investigating the infeasible preemption combinations!




**Goal**:
Reduce the pessimism of the approximation!

# First source of over-approximation

- CRPD for each point is computed in isolation, which leads to:
  - Pessimism regarding the preemption scenarios.
  - Pessimistic CRPD upper bounds

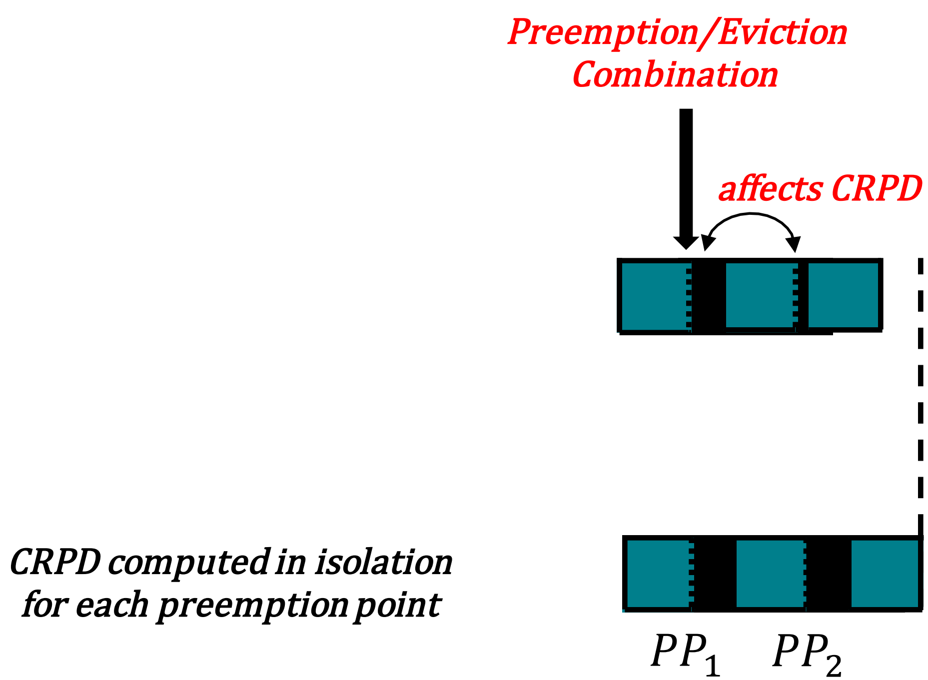


# First source of over-approximation

What if we want to calculate the CRPD defined per task?

- To account for each CRPD computed in isolation is pessimistic.

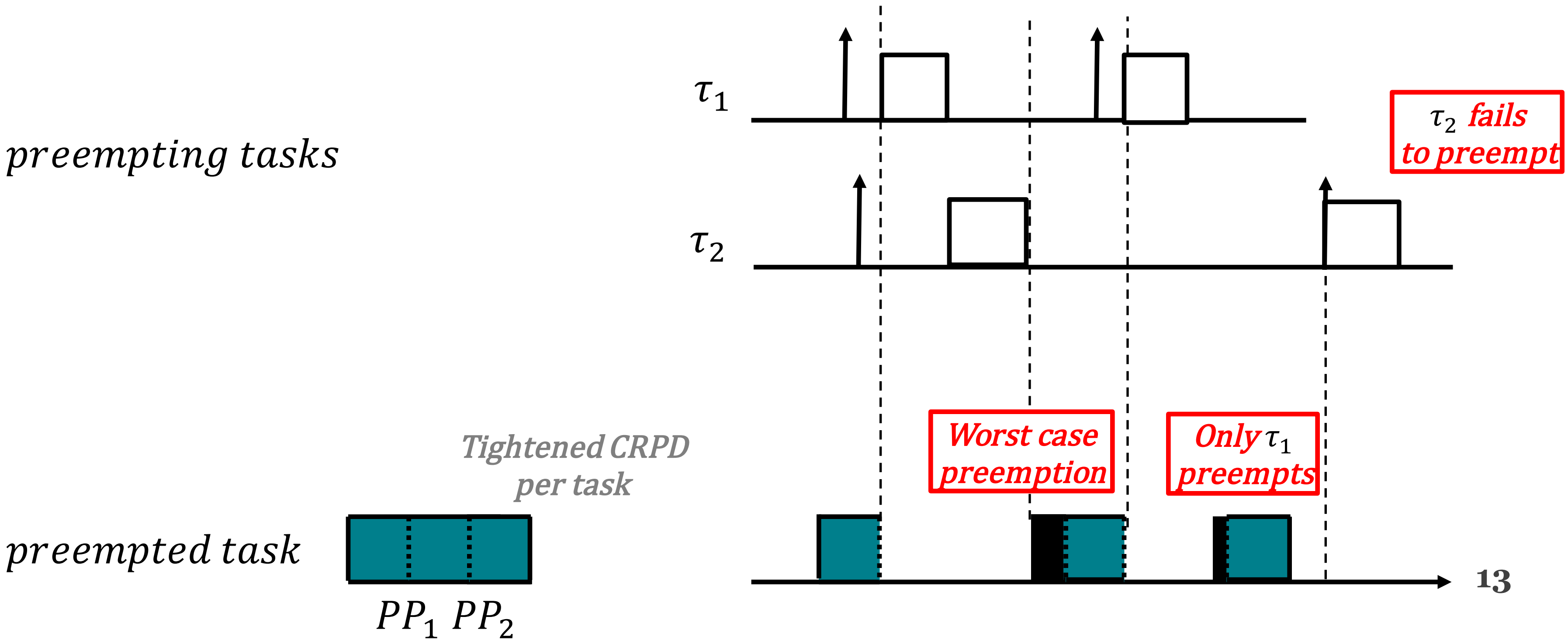
Take into account that preemption scenario at one point affects the possible preemption scenarios of the succeeding ones.



# Tightening CRPD bounds

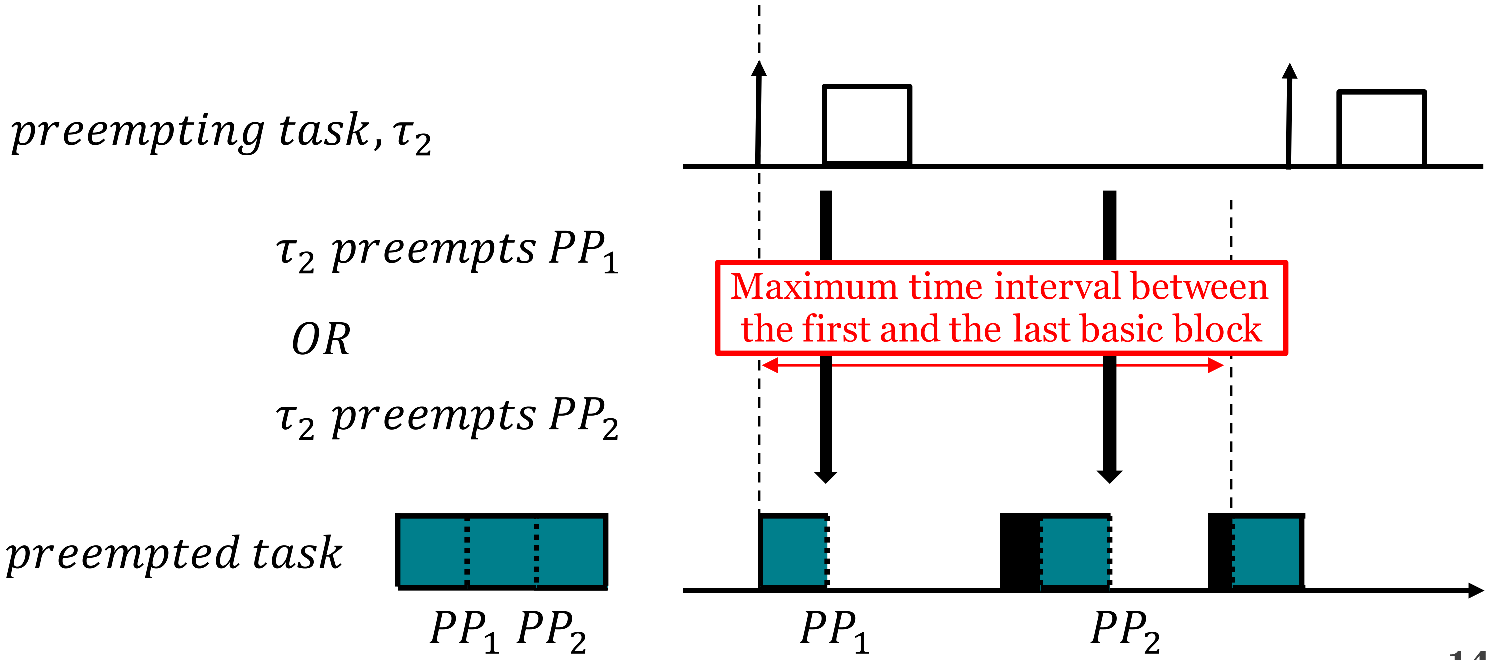
For each task:

1. Identify **infeasible preemption scenarios**.
2. Among the remaining preemption scenarios identify the one causing the worst CRPD.



# Identifying Infeasible Preemption Scenario?

- Scenario when the preempting task cannot affect the CRPD of both succeeding preemption points of the preempted task.
- Case when the preempting task cannot be released twice during the maximum time interval from the start time of one basic block until the start time of the succeeding basic block.



# Why it is not a trivial problem?

- There are many different preemption scenarios. Which one causes the worst CRPD?

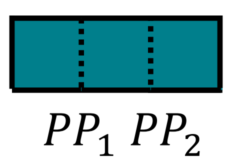
$preempting\ task, \tau_2$

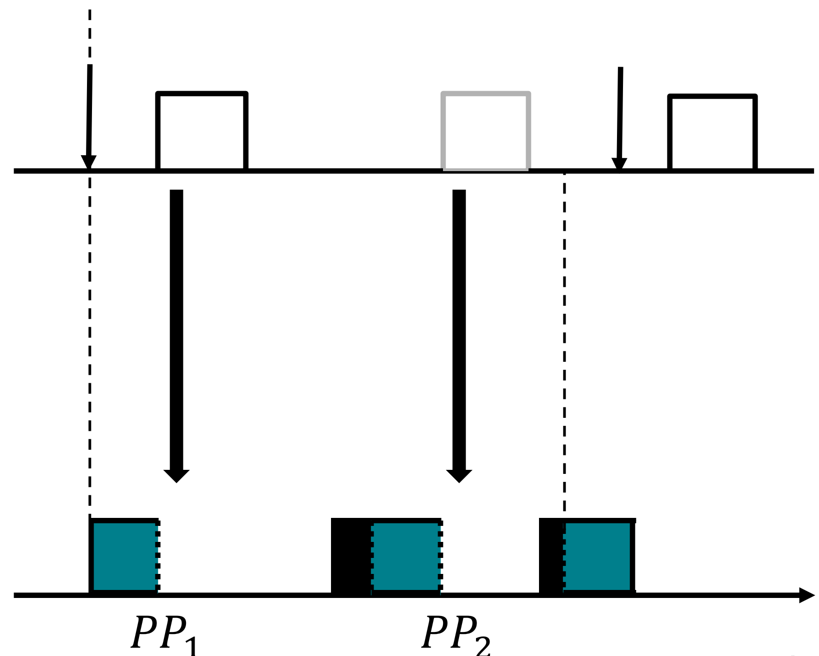
$scenario\ 1\text{:}\ \tau_2\ preempts\ PP_1 \Rightarrow CRPD_1$

$scenario\ 2\text{:}\ \tau_2\ preempts\ PP_2 \Rightarrow CRPD_2$

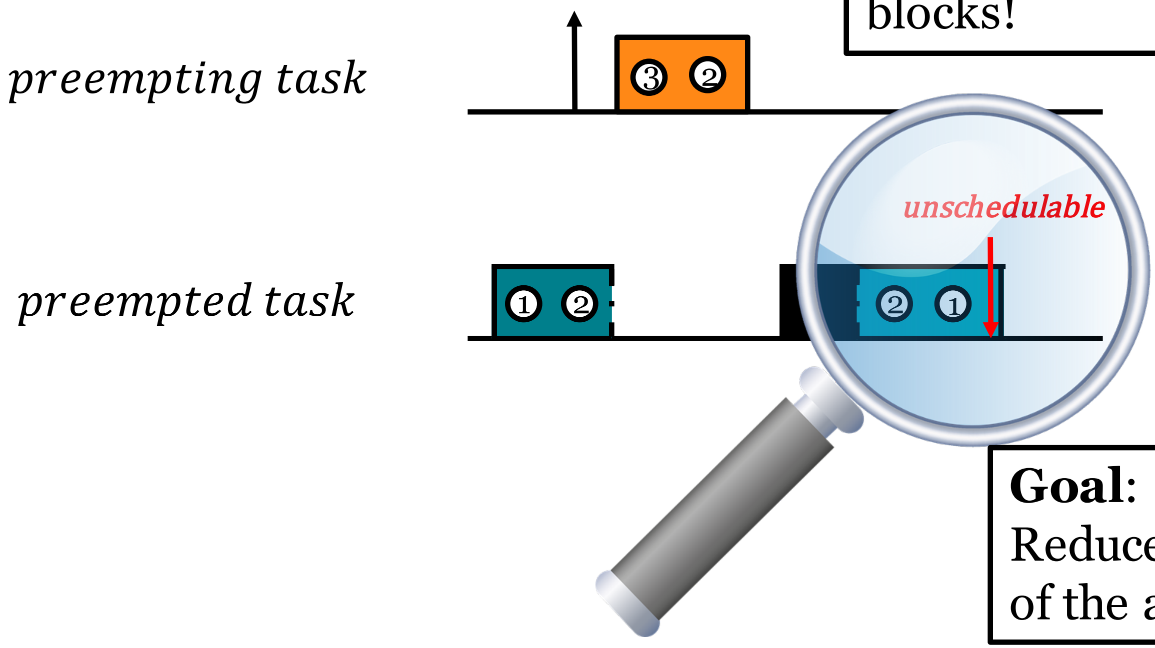
$Which\ one\ is\ the\ \boldsymbol{maximum}?$

$preempted\ task$



# Second source of over-approximation

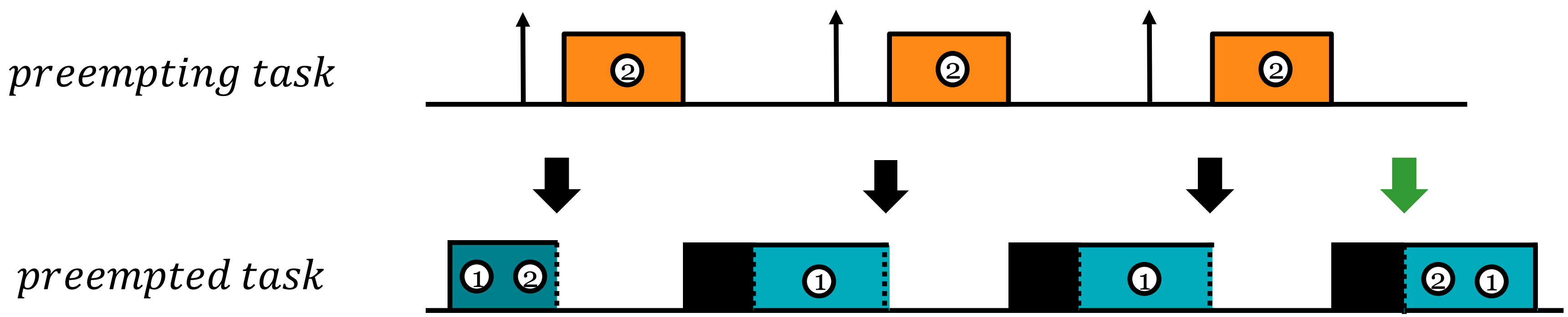



**How?**
By investigating the infeasible reloads of the useful cache blocks!

**Goal**:
Reduce the pessimism of the approximation!

# Second source of over-approximation

Cache block 2 can be evicted at any of the preemption points, but only once, i.e. it can be reloaded only once!

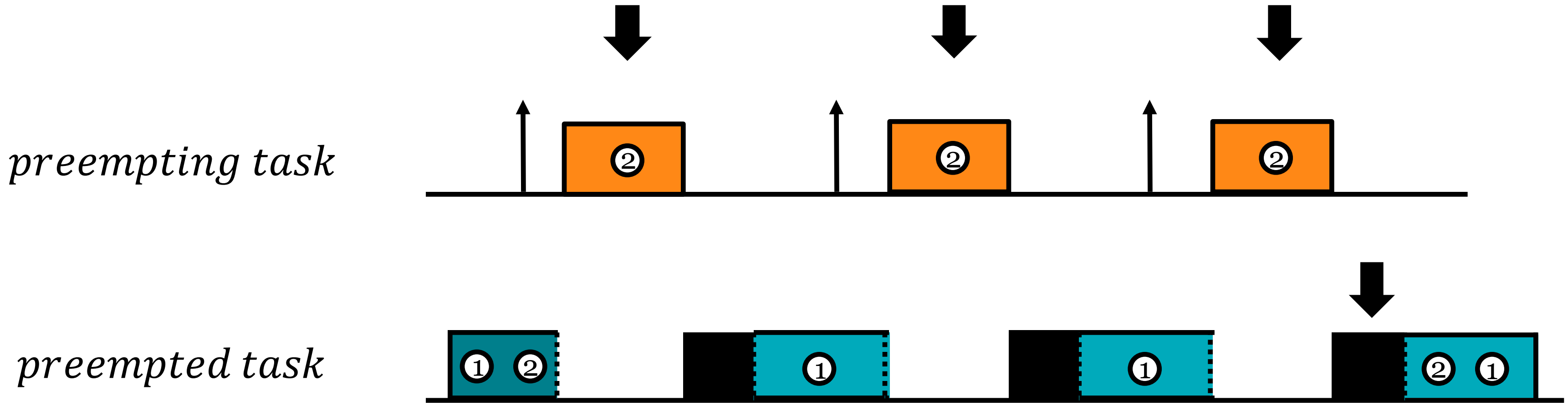



Cache block 2 accessed at the beginning and at the end of the preempting task.

Existing approaches:
3 cache block reloads

In reality:
1 cache block reload

# Solution



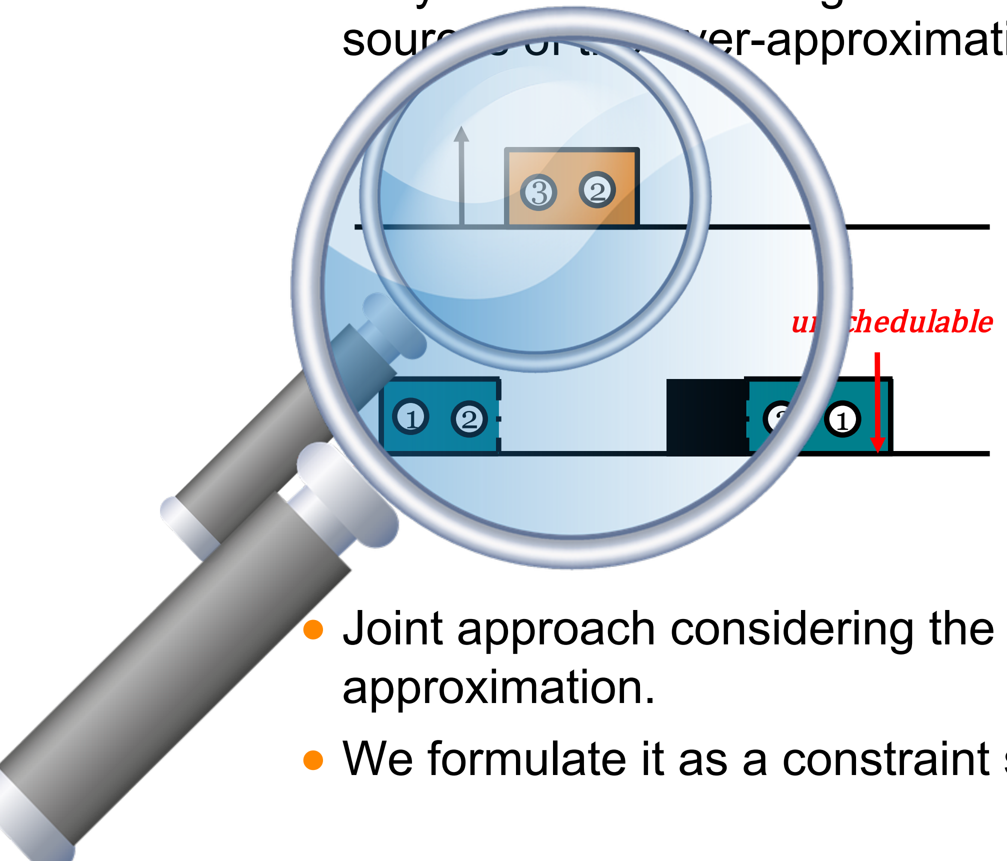

1. Identify if there is a possible eviction of the cache block by the preempting task between the two consecutive accesses.

2.
If there is, account it only once, just before the next access.

If not, do not account it at all.

# Tight approximation of CRPD

- Why it is not trivial to tighten the CRPD although we identified the sources of the over-approximation?



**Over-approximation 1:**
Due to accounting infeasible preemption combinations!

**Over-approximation 2:**
Due to accounting infeasible cache block reloads!

- Joint approach considering the solutions for both sources of over-approximation.
- We formulate it as a constraint satisfaction problem.

# Proposed approach

- Optimization formulation:
  - Constraints
    - Represent feasible preemption combinations.
  - Goal function:
    - Identify the preemption scenario causing the worst CRPD bound, accounting also for the infeasible reloads.

$$\max(\text{reloads}(\boxed{\text{Preemption scenario}} - \boxed{\text{Infeasible reloads}}\,)\,)$$

  - Output
    - Tight CRPD bounds.

# Evaluation

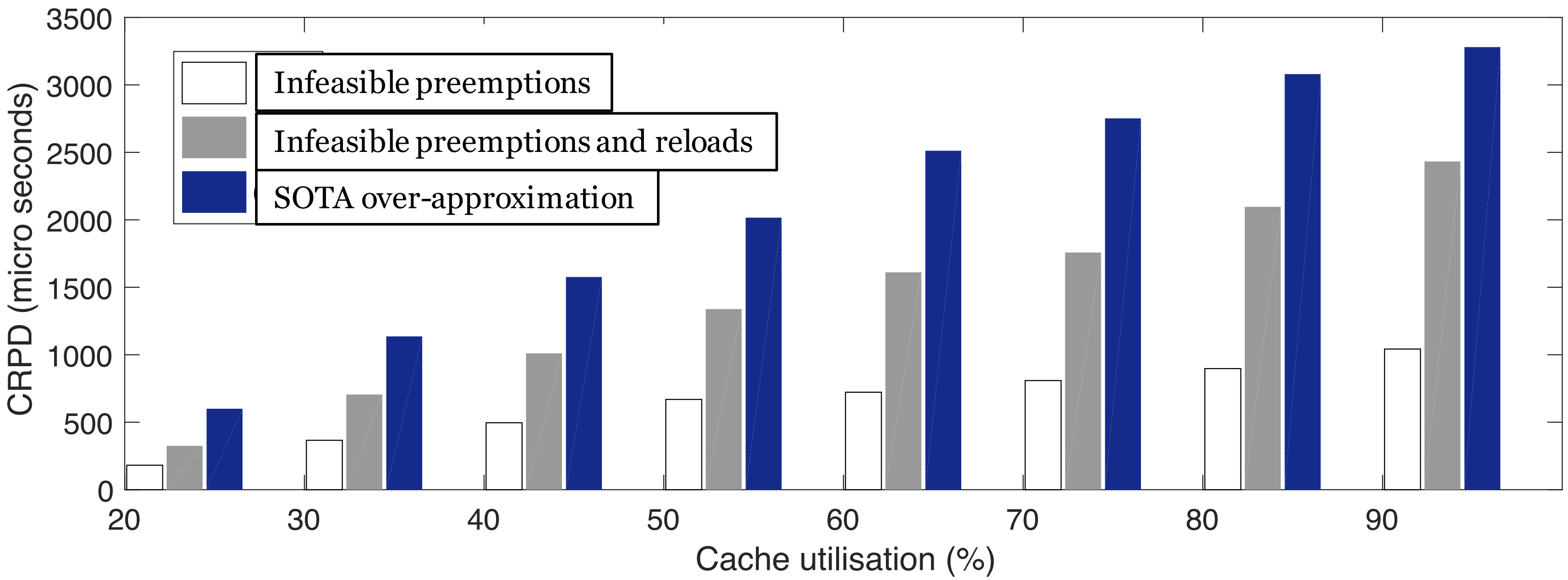
- **Goal of the experiment:**
  - To investigate to what extent the CRPD bounds are tightened, compared to
    - the simplified CRPD approximation and
    - optimisation which does not account for the infeasible UCB reloads.

**General Experiment setup:**

2000 generated tasksets per the parameter under investigation (**cache utilisation** or the **number of tasks in a taskset**).

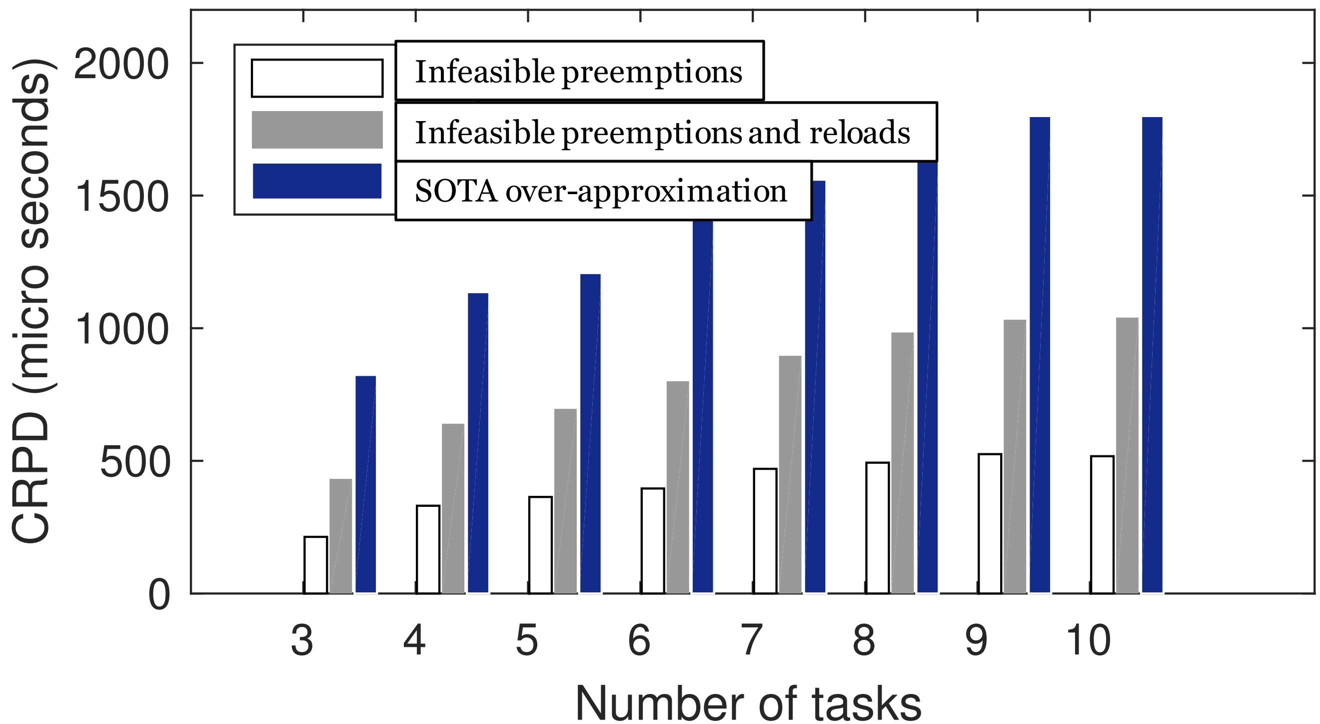
# Evaluation

- **Experiment setup:**
  - Taskset size fixed to 10
  - Taskset utilisation fixed to 80%
  - Total cache utilisation (20%, 90%)
- **Results:**
  - Tightening improved the CRPD bounds.
  - CRPD bounds tightened by 50% to 70%.



# Evaluation

- **Experiment Setup**
  - Taskset size (3 - 10)
  - Total cache utilization fixed to 40%
- **Results**
  - Bounds tightened by 50% to 70%
  - Tightening scales well with the taskset increase.



# Conclusions

- We propose **a novel method for computing the CRPD** in sporadic task model scheduled under the Fixed Preemption Point approach.
- The novelty of the method comes from the **more detailed analysis of the infeasible eviction scenarios and infeasible useful cache block reloads**, compared to the SOTA.
- The proposed **method achieves to significantly tighten the bounds** compared to the previous methods.

# Future work

- A preemption point selection algorithm that exploits the proposed method.

- Method for tightening the bounds in Fully-preemptive systems.