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

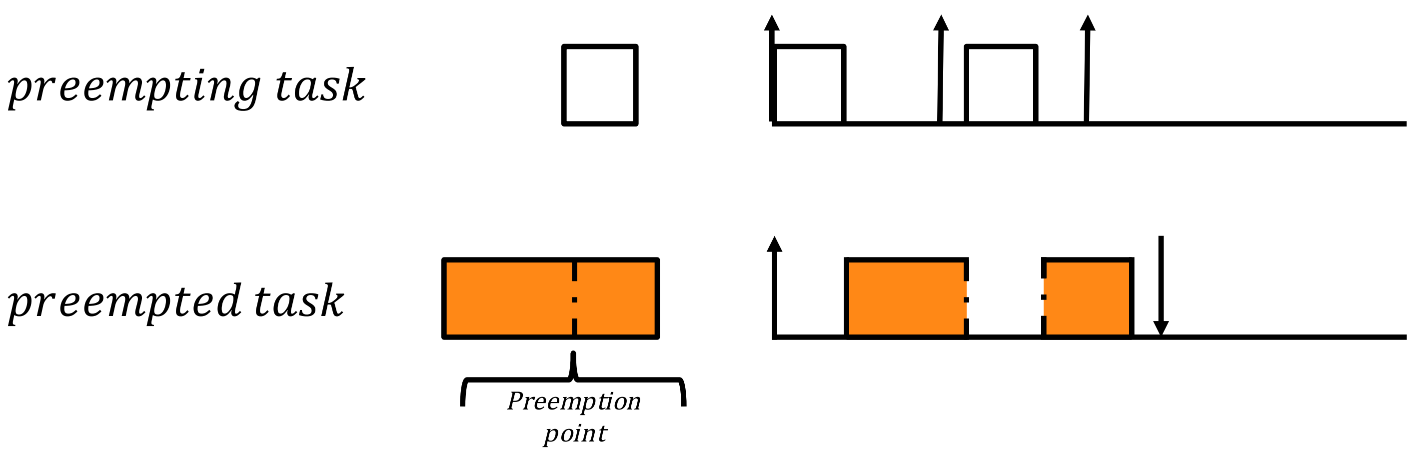
EUROWEB+
European Research and Educational Collaboration with Western Balkan

Content

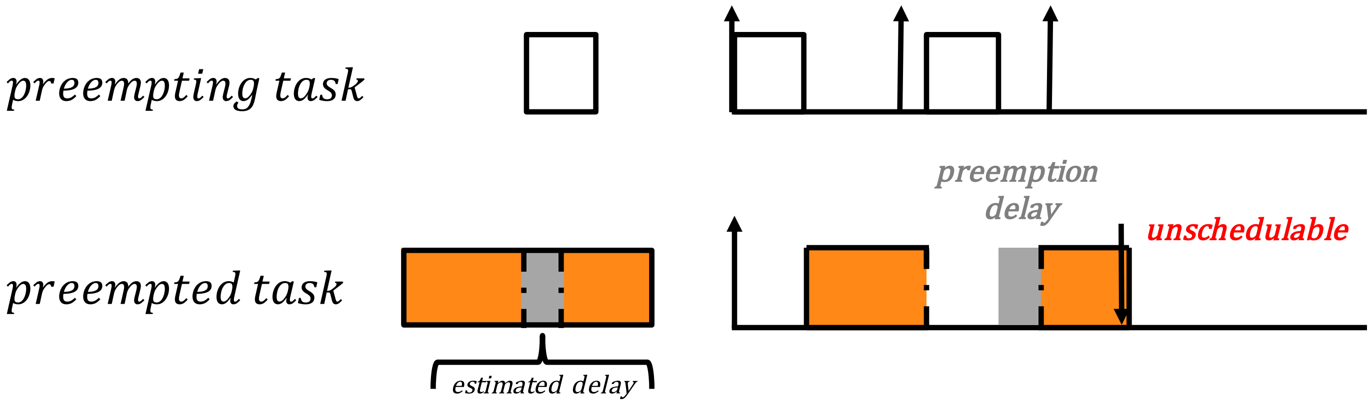
- Background and Motivation
- Problem formulation
- Proposed approach
- Evaluation
- Conclusions and Future Work

Fixed Preemption Points

- **Limited Preemptive Scheduling**
  - An attractive paradigm that combines the benefits of **fully-preemptive** and **non-preemptive** scheduling.
- **Fixed Preemption Points (FPP)**
  - Preemption allowed only at **predefined selected** locations inside the code, called preemption points.
  - The selection enables control of preemption related delays, possibly reducing their impact on schedulability.



Preemption-Related Delay



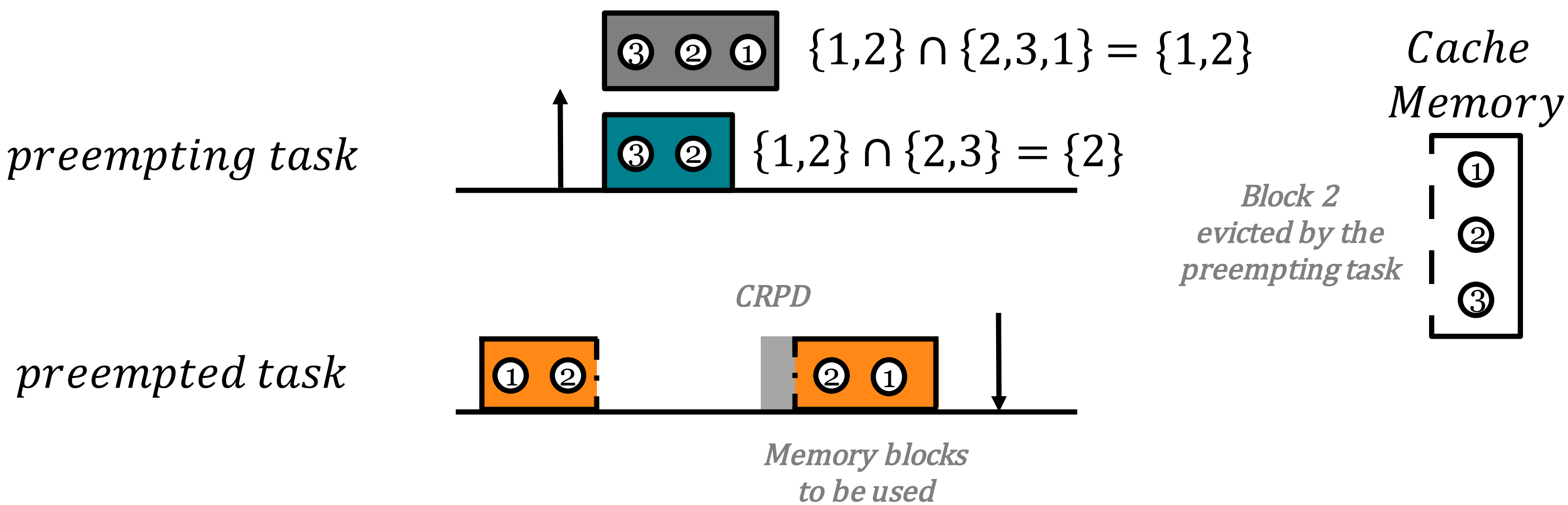
Preemption-related **delay** consists of different delay types: bus-related, scheduling-related, pipeline-related, etc.

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

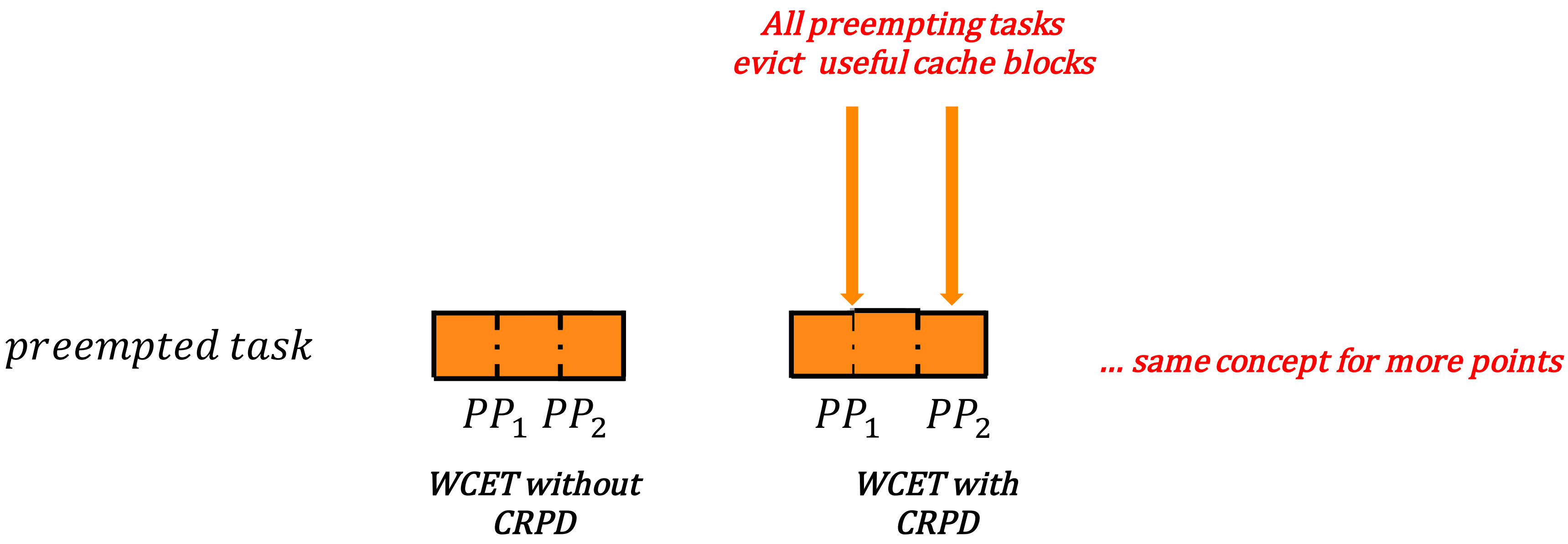
- **CRPD depends upon two important factors:**
  1. Where the preemption occurs?
  2. Which preempting tasks affect the CRPD at this point?



- Different preempting tasks cause different CRPDs due to eviction!

Related FPP approaches

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



Motivation

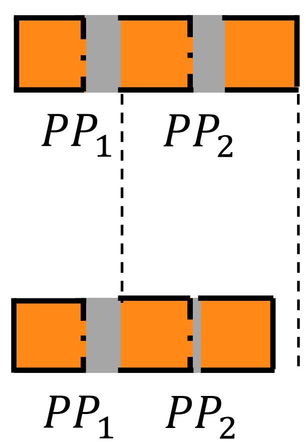
What if we want to calculate the CRPD defined per task?

- 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.

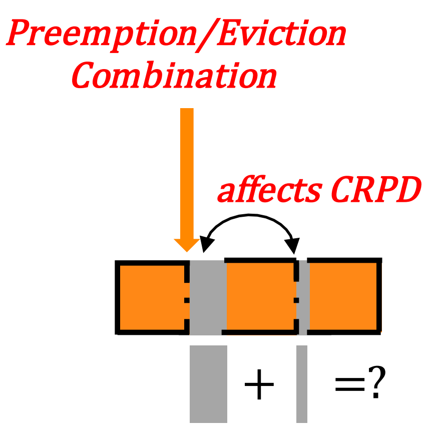
CRPD computed in isolation for each preemption point

Accounted CRPD at one point affects possible CRPD of the succeeding point



Problem formulation

- How to tighten the CRPD bounds by taking into account the information that preemption scenario at a certain point may affect CRPD of the succeeding points?

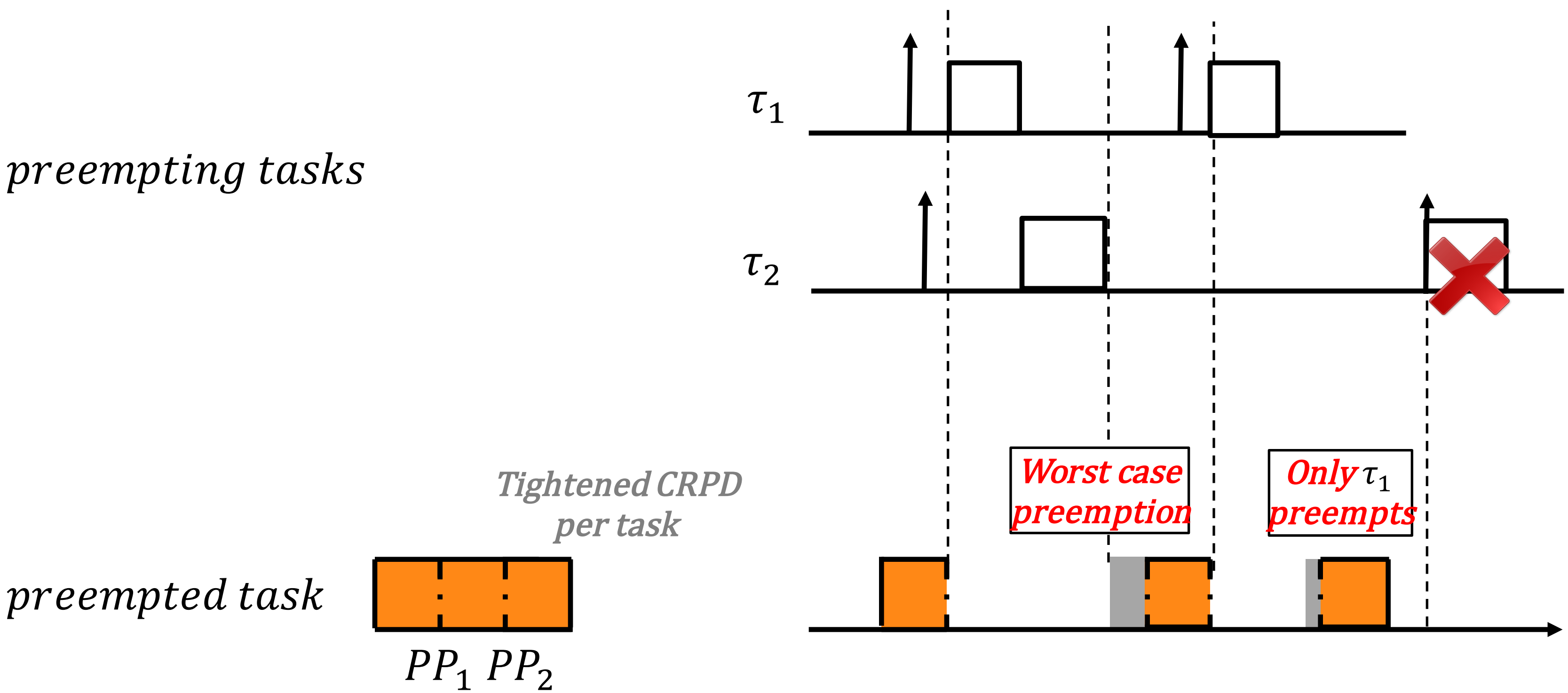


- How do we identify the "*mutual affection*" of CRPDs of the two succeeding preemption points?

Proposed approach

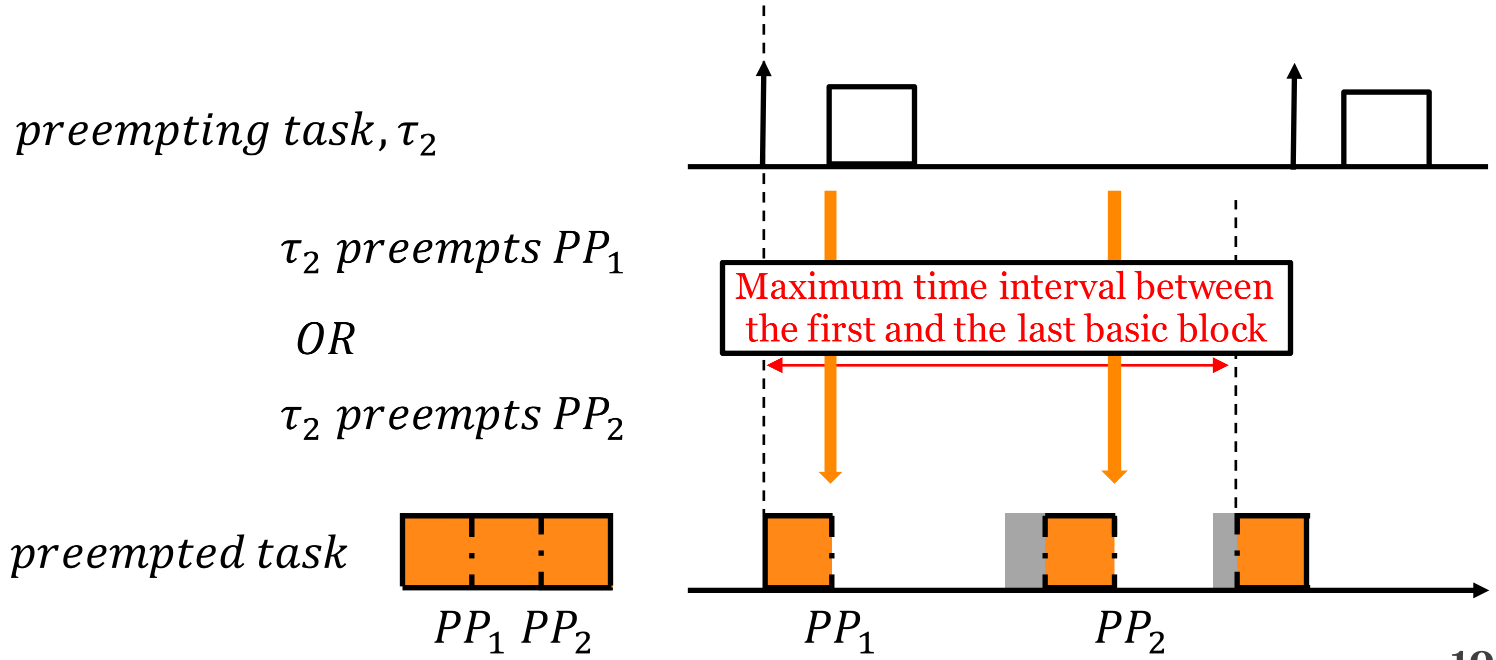
For each task:

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?

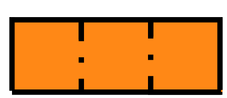
$preempting\ task, \tau_2$

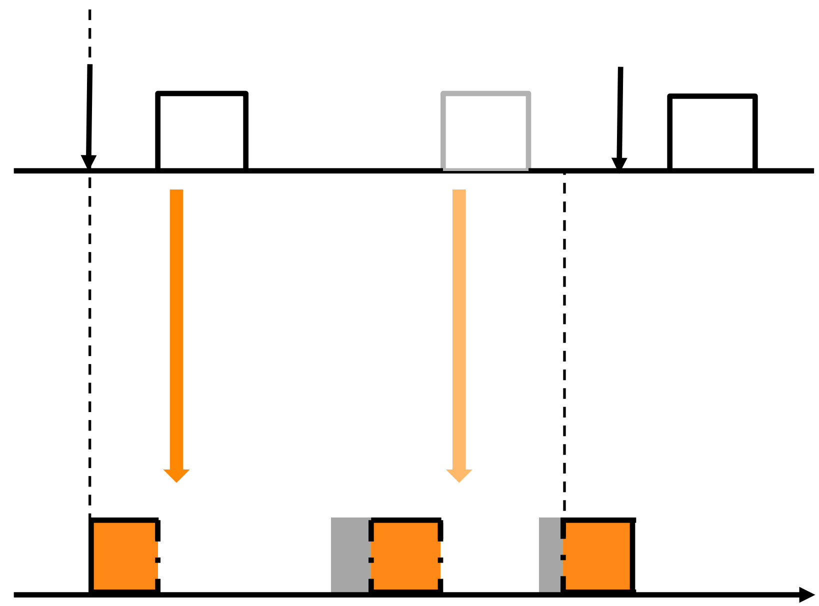
$scenario\ 1\!: \tau_2\ preempts\ PP_1$ ➡ $CRPD_1$

$scenario\ 2\!: \tau_2\ preempts\ PP_2$ ➡ $CRPD_2$

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

$preempted\ task$





Proposed approach

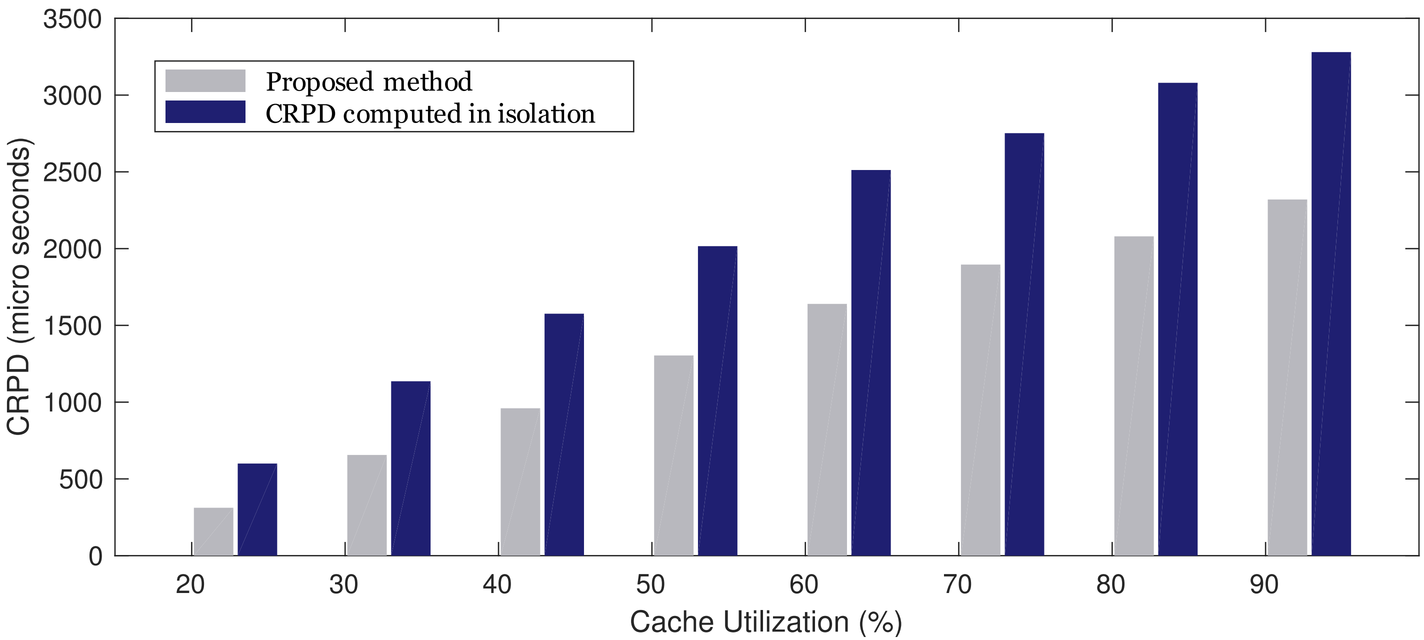
- Optimization formulation:
  - Input
    - Preemption scenarios (not deemed infeasible)
  - Goal
    - Considering the CRPD computations of those scenarios, identify the one causing the worst CRPD bound.
  - Output
    - CRPD bound, defined for task.

Evaluation

- **Goal of the experiment:**
  - To investigate to what extent the CRPD bounds are tightened, compared to the simplified CRPD approximation.
- **Experiment setup**
  - Taskset size fixed to 10
  - Taskset utilization fixed to 80%
  - Total Cache utilization (20% - 90%)

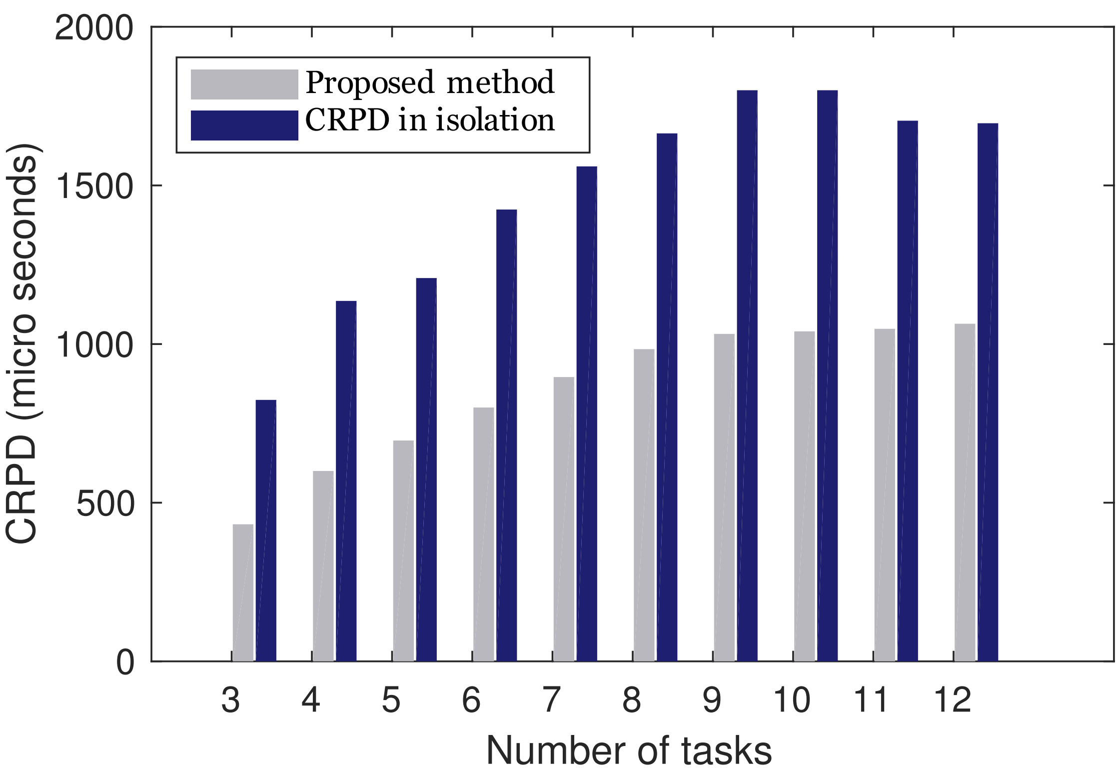
Evaluation

- **Results:**
  - Tightening improved the CRPD bounds.
  - CRPD bounds tightened by 30% to 50%.
  - Taskset size increase does not significantly deteriorates the tightening.



Evaluation

- **Experiment Setup**
  - Taskset size (3 - 12)
  - Total cache utilization fixed to 40%
- **Results**
  - Bounds tightened by 40% to 50%
  - 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**, compared to the SOTA.
- The proposed **method achieves to tighten the bounds by 30% to 50%** compared to the over-approximating worst-case eviction methods.

Future work

- A preemption point selection algorithm that exploits the proposed method.
- Scalability improvement.
- Heuristics and approximation methods with multiset-based CRPD estimation approaches combined with the analysis of infeasible preemption combinations.
- Method for tightening the bounds in Fully-preemptive systems.