Workshop CAPITAL: sCalable And PrecIse Timing AnaLysis for multicore platforms

Memory hierarchy in scheduling simulation: problems, implementation & return of experience

Hai Nam Tran[+], Stéphane Rubini[+], Jalil Boukhobza[*], Frank Singhoff[+]

[+]: Lab-STICC, CNRS UMR 6285, Univ. Brest, France
[*]: Lab-STICC, CNRS UMR 6285, ENSTA Bretagne, France

UBO Université de Bretagne Occidentale

Lab-STICC

cnrs

Introduction

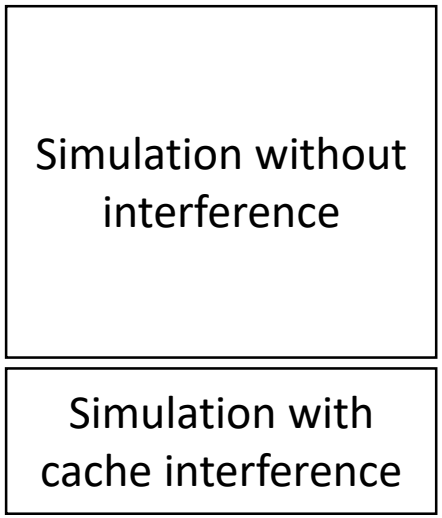
- **Real-time embedded systems (RTES)**
  - **Real-time: must process information and produce responses within a specified time, else risk severe consequences, including failure**
    - **Real-time scheduling analysis** verify the feasibility/schedulability of a system at design time
  - **Challenge: more and more parallelism and complexity at both software and hardware**
    - Even in small systems (drones, cars)
    - Less usage of dedicated hardware / more COTS (Federal aviation administration, Commercial Off The Shelf Avionics Software Study, 2011)
      - **Multicore architectures**
      - **Memory hierarchy**

Introduction

- **Memory hierarchy problem in RTES**
  - **Improve the overall system performance, but leads to execution time variability due to interference (Lugo et al., 2022)**
    - Cache memory ← Addressed in this talk
    - Memory bus
    - Main memory
- **Verification by scheduling simulation**
  - **A common practice of actors in the real-time community**
    - Integration of Cheddar scheduling simulator in the commercial tool AADL Inspector by Ellidiss Technologies
  - **Need of support for multi-core and memory hierarchy**
    - Usage of Cheddar in the Project PLATO (Plasson et al., 2022 - https://sci.esa.int/web/plato)

Introduction

- **Lack of scheduling simulators with support for interference-aware scheduling simulation**
  - **RTSim (Manacero et al., 2001)**
  - **MAST (Harbour et al., 2001)**
  - **ARTISST (Decotigny et al., 2002)**
  - **STORM (Urunuela et al., 2010)**
  - **YARTISS (Chandarli et al., 2012)**
  - **SimSo (Cheramy et al., 2015)**
  - **Cheddar (Singhoff et al., 2004)**



- **Lack of theoretical research to guarantee the applicability of scheduling simulation as a schedulability test**

Outline

1. Introduction
2. Scheduling simulation with interference
3. CRPD-aware scheduling simulation
   - **Problem statement & contribution**
   - **Related work**
   - **Background**
     - CRPD computation models: C^{off} and C^{on}
     - Sustainability analysis
   - **Approach**
     - C^{on-lim}: an improved CRPD computation model
     - Sustainability analysis of C^{on-lim}
     - Feasibility interval of C^{on-lim}
   - **Evaluation**
4. Conclusion

Scheduling simulation with interference

- **Why scheduling simulation ?**
  - **Advantages**
    - **Observed reduced pessimism** compared to static WCRT analysis
      - Used as a **sufficient condition** to compare interference-aware WCRT analysis
    - **Adaptability/Flexibility** - integration of additional scheduling parameters
    - **Observability** - record and analyze properties such as numbers of preemptions, total preemption costs, and various scheduling events that are not observable by static analysis
    - **Analysis** - understand why a system is not schedulable

Scheduling simulation with interference

- **Why scheduling simulation ?**
  - **Limitations**
    - **Scalability** - especially when mixing timing specifications of different orders of magnitude; e.g: WCET and cache block reload time.
    - **Analysis** - tons of trace
    - **Engineering challenge** - how to implement the simulator
    - **2 theoretical problems: sustainability and feasibility interval** (to be detailed later)

Scheduling simulation with interference

- **We investigated cache memory interference - cache related preemption delay - for uniprocessor with one level of direct-mapped instruction cache**
  - **A tiny portion of the interference-aware scheduling simulation topic !**

… and we found the following problems

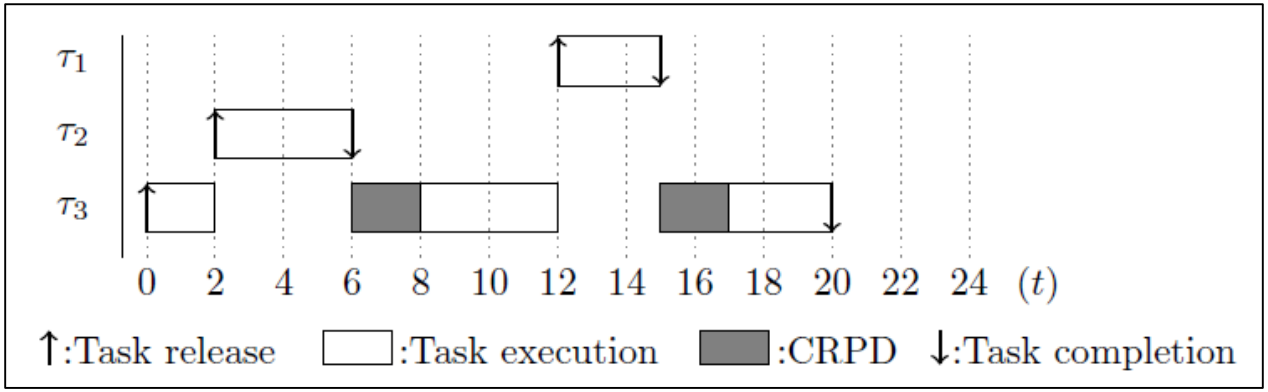
- **Problem 1: How to model and compute the interference**
  - What should we consider to simulate the worst-cases
  - Pessimistic of the computation models
- **Problem 2: Sustainability**
  - If a system is considered to be schedulable by scheduling simulation with the worst-case parameters, is it schedulable in better cases ?
- **Problem 3: Feasibility interval**
  - How long should we run the simulation ?
- **Problem 4: Simulator performance**
  - Mixing timing specifications of different orders of magnitude = (very) long simulation period

Outline

1. Introduction
2. Scheduling simulation with interference
3. **CRPD-aware scheduling simulation**
   - **Problem statement & Contribution**
   - **Related work**
   - **Background**
     - CRPD computation models: C^{off} and C^{on}
     - Sustainability analysis
   - **Approach**
     - C^{on-lim}: an improved CRPD computation model
     - Sustainability analysis of C^{on-lim}
     - Feasibility interval of C^{on-lim}
   - **Evaluation**
4. **Conclusion**

Problem Statement

- **Cache memory in RTES**
  - **Cache related preemption delay (CRPD): the additional time to refill the cache with memory blocks evicted by preemptions**



  - CRPD is a non-negligible preemption cost, can present up to 44% of the WCET (Pellizzoni et al., 2007)
  - Create scheduling anomalies and complex optimization problems (Phavorin et al., 2015) which require extensions of classical scheduling analysis

Problem Statement

- **CRPD-aware scheduling simulation**
  1. **Pessimistic**
  2. **Non sustainable**
  3. **Unidentified feasibility interval**

The three problems limit the applicability and the usage of scheduling simulation as a verification methods for RTES with cache memory

Contribution

- A formalization of CRPD-aware scheduling simulation
- A CRPD computation model (called C^{on-lim})
  - Less pessimistic
  - Sustainable with regard to the capacity parameter
  - Proof of the feasibility interval

Enable the usage of scheduling simulation as a verification method for RTES with cache

Related work

- **Cache-aware scheduling analysis**
  - **Cache accesses are considered precisely and the cache state at each instant has to be known during the system execution (Phavorin et al., 2015)**
    - Impossible to achieve in practice except for very simple systems
- **CRPD-aware scheduling analysis**
  - **Cache effect is assessed through induced preemption delays**
  - **Compute the upper-bounds on the additional delays due to the cache every time a task resumes after a preemption**
    - The cache state at each instant are not required
    - More applicable in scheduling simulation

Related work

- **CRPD-aware scheduling analysis**
  - **Analytical-based approaches**
    - **CRPD-aware WCRT analysis**: Lee et al., 1998; Busquets-Mataix et al., 1996, Tomiyama et al., 2000; Staschulat et al., 2005; Altmeyer et al., 2012; Lunniss et al., 2014
    - **Eliminate or limit the effect of CRPD**: Bertogna et al., 2011; Luniss et al., 2012; Altmeyer et al., 2015;
    - **Optimal scheduling**: Phavorin et al., 2017
  - **Scheduling simulation based approaches**
    - **Simulators with cache support**
      - **SimSo** (Cheramy et al., 2015): Stack Distance Profile
      - **Cheddar** (Tran et al., 2014): Useful Cache Block/Evicting Cache Block

Outline

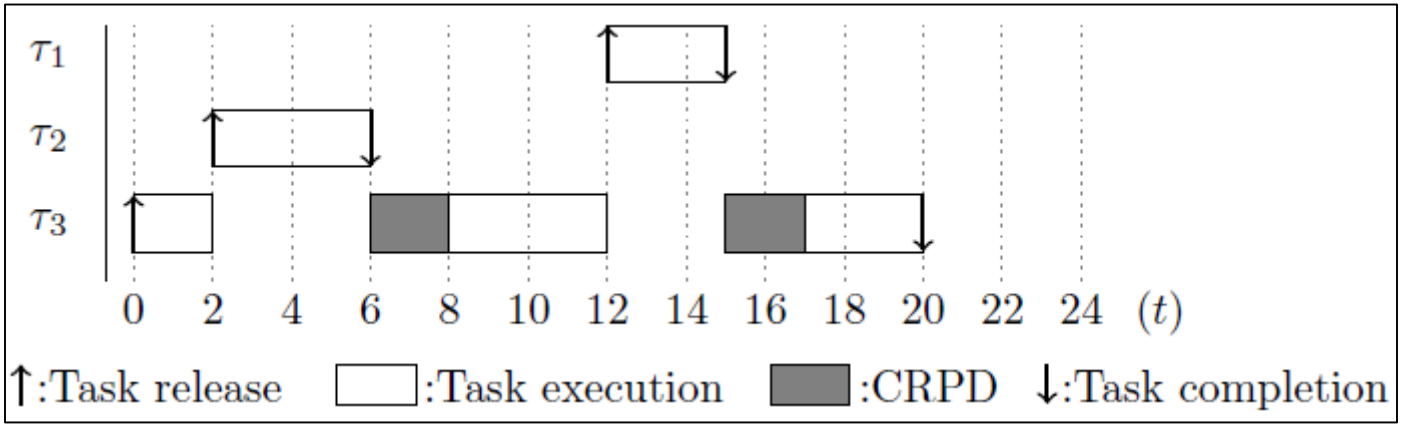
1. Introduction
2. Scheduling simulation with interference
3. **CRPD-aware scheduling simulation**
   - Problem statement & Contribution
   - Related work
   - **Background**
     - CRPD computation models: C^{off} and C^{on}
     - Sustainability analysis
   - **Approach**
     - C^{on-lim}: an improved CRPD computation model
     - Sustainability analysis of C^{on-lim}
     - Feasibility interval of C^{on-lim}
   - **Evaluation**
4. **Conclusion**

Background

- **Scheduling simulation**
  - **Simulation of a task set $\boldsymbol{T}$ on an architecture $\boldsymbol{M}$ under a scheduler $\boldsymbol{S}$ over an interval of time $\boldsymbol{F}$**
- **CRPD-aware scheduling simulation**
  - **Scheduling simulation with a CRPD computation model $\boldsymbol{C}$**
    - Describe the method of computing the CRPD added to the execution time of a task when it resumes **after a preemption**
- **System model and assumptions**
  - **$\boldsymbol{T}$: a set of periodic tasks $\tau_i(C_i, T_i, D_i, \Pi_i, O_i)$ - capacity, period, deadline, priority and offset**
  - **$\boldsymbol{M}$: uniprocessor with one level of direct mapped instruction cache**
  - **$\boldsymbol{S}$: fixed priority preemptive scheduling**
  - **$\boldsymbol{C}$: $\boldsymbol{C^{off}}$, $\boldsymbol{C^{on}}$, $\boldsymbol{C^{on-lim}}$**
  - **$\boldsymbol{F}$: to be defined**

Background

- C^{off}: offline CRPD computation model
  - The CRPD when a task τ_i is preempted is fixed and computed offline
    - CRPD γ_i is added to the remaining capacity of τ_i whenever the task is preempted



  - Pessimistic because the preempting tasks may not evict the data in the cache of the preempted task
    - The pessimism also depends on the method of computing the CRPD offline

Background

- **C^{on}: online CRPD computation model**
  - **For task τ_i a set of useful cache blocks (UCB_i) and evicting cache blocks (ECB_i) are computed before simulation**
    - UCB_i (Lee et al., 1998): cache blocks used by a task that are reused later on and will have to be reloaded if evicted from the cache due to preemption
    - ECB_i (Busquets-Mataix et al., 1996): cache blocks used by a task that may override some cache locations used by the preempted task
- **CRPD computation**
  - **UCB_i^t: the set of UCB of τ_i in the cache at time t**
  - **τ_i is preempted by τ_j at time t**

$$UCB_i^t = UCB_i^{t-1} - (UCB_i^{t-1} \cap ECB_j)$$

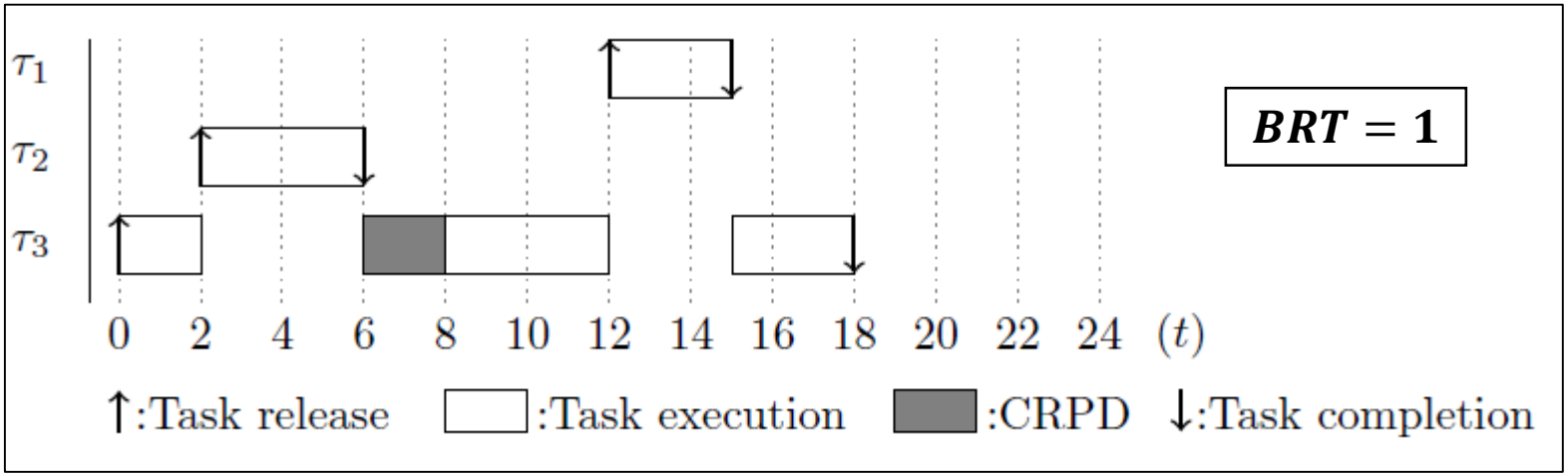
  - **τ_i resumes execution at time t+Δ**

$$\gamma_i^{t+\Delta} = \left|UCB_i - UCB_i^{t+\Delta}\right| \cdot BRT$$

Background

- C^{on} example

	UCB	ECB
τ_1	$\emptyset$	$\{1,2\}$
τ_2	$\emptyset$	$\{3,4\}$
τ_3	$\{3,4\}$	$\{1,2,3,4\}$



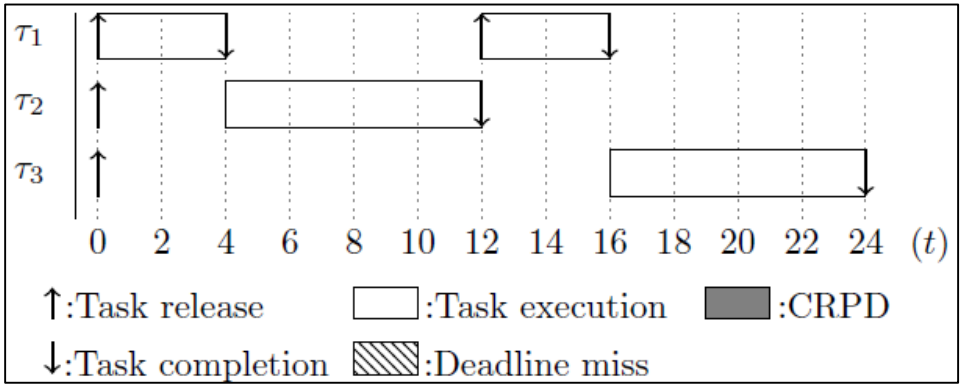
Background

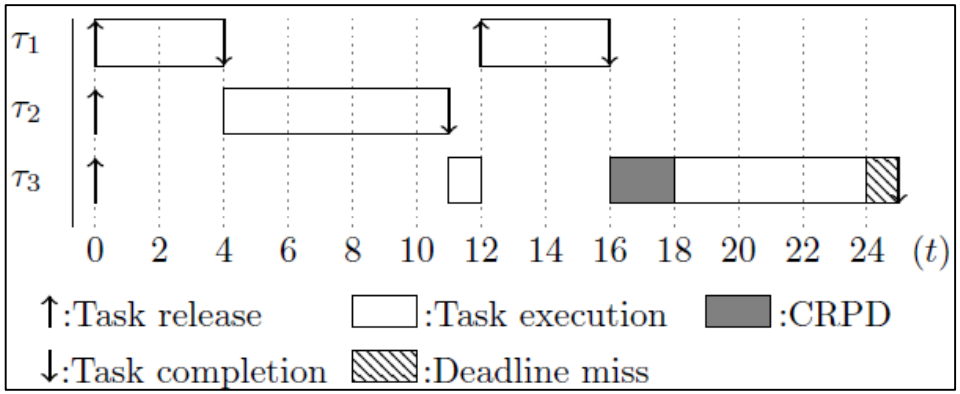
- **Sustainability analysis**
  - **Definition (Goossens et al., 1997): a given scheduling policy and/or a schedulability test is sustainable if any system that is schedulable under its worst-case specification remains so when its behavior is better than the worst-case**
  - **The term "better" means that the parameters of one or more individual task(s) are changed in any, some, or all of the following ways**
    - (1) reduced capacity
    - (2) larger period
    - (3) larger relative deadline

Background

- **Sustainability of C^{on}**
  - **Example 1**
    - Task set $T = \{\tau_1, \tau_2, \tau_3\}$
    - $BRT = 1$
  - **Case 1: original task set**
    - Schedulable, all deadlines are met
  - **Case 2: reduced capacity of τ_2**
    - $C'_2 = 7\ (< C_2 = 8)$
    - $\gamma_3^{16} = \left|UCB_3 - UCB_3^{16}\right| \cdot BRT$
      $= |\{1,2\} - \emptyset| = 2$
    - τ_3 missed its deadline
    - **Non sustainable scheduling with regard to the capacity parameter**

Task	C_i	T_i	D_i	O_i	Π_i	UCB_i	ECB_i
τ_1	4	12	12	0	3	$\emptyset$	{1,2}
τ_2	8	24	24	0	2	{3}	{3,4}
τ_3	8	24	24	0	1	{1,2}	{1,2}





Outline

1. Introduction
2. Scheduling simulation with interference
3. CRPD-aware scheduling simulation
   - Problem statement & Contribution
   - Related work
   - Background
     - CRPD computation models: C^{off} and C^{on}
     - Sustainability analysis
   - **Approach**
     - C^{on-lim}: an improved CRPD computation model
     - Sustainability analysis of C^{on-lim}
     - Feasibility interval of C^{on-lim}
   - **Evaluation**
4. Conclusion

Approach

- C^{on-lim}: an improved online CRPD computation model
  - **The CRPD is at most be proportional to the executed capacity (Luniss, 2014)**
    - The CRPD is related to the amount of useful information that has to be reloaded into the cache
    - If a task is preempted shortly after it starts, it has not yet loaded all of the UCBs and will therefore not experience the maximum CRPD
- CRPD computation
  - **Notation: ρ_i^t - number of UCBs loaded into the cache at time t**
  - **The number of UCBs in the cache at time $t + \Delta$**

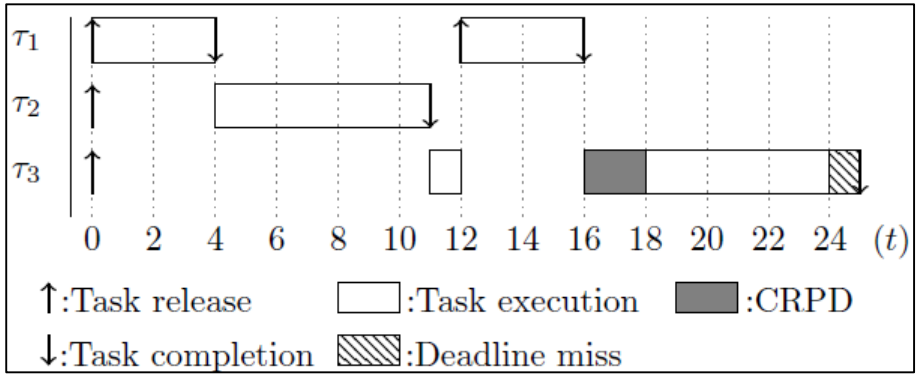
$$\rho_i^{t+\Delta} = min(|UCB_i|, \rho_i^t + \left\lfloor \frac{\Delta}{BRT} \right\rfloor)$$

  - **CRPD computation when τ_i resumes at time t**

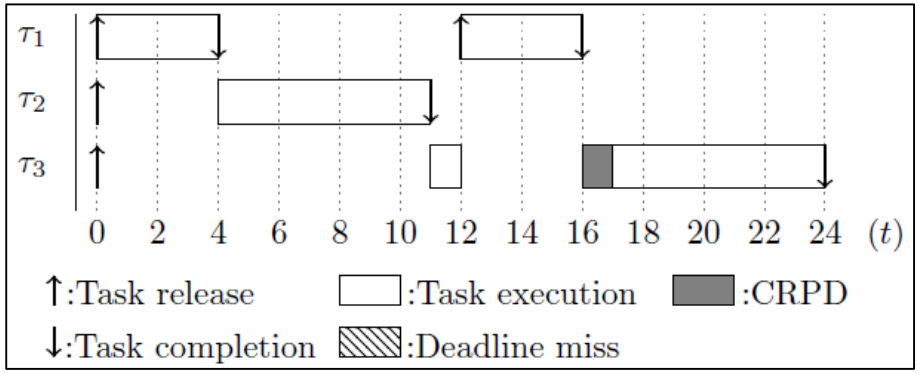
$$\gamma_i^t = min\left(\left|UCB_i - UCB_i^t\right|, \rho_i^t\right) \cdot BRT$$

Approach

- **C^{on-lim}: an improved online CRPD computation model**
  - **Example 1 case 2 with C^{on}**
    - $C'_2 = 7 \ (< C_2 = 8)$
    - τ_3 missed its deadline



  - **Example 1 case 2 with C^{on-lim}**
    - Schedulable task set



Sustainability analysis of C^{on-lim}

- **Reduced capacity**

Theorem 1 : The added CRPD cannot be larger than the executed capacity of task $\boldsymbol{\tau_i}$. In other words, if $\boldsymbol{\tau_i}$ executes in $\boldsymbol{n-1}$ discrete intervals $[\boldsymbol{t_a}, \boldsymbol{t_a} + \boldsymbol{\Delta_a}), \boldsymbol{a} \in (\boldsymbol{0}, \boldsymbol{1}, \dots, \boldsymbol{n-1})$ and experiences the preemptions costs $\boldsymbol{\gamma_i^{t_b}}, \boldsymbol{b} \in (\boldsymbol{1}, \dots, \boldsymbol{n})$, we have:

$$\sum_{b=1}^{n} \gamma_i^{t_b} \leq \sum_{a=0}^{n-1} \Delta_a$$

- **Proof sketch: Prove by induction**
  - Base case: $\gamma_i^{t_1} \leq \Delta_0$, $\gamma_i^{t_1} + \gamma_i^{t_2} \leq \Delta_0 + \Delta_1$
  - Inductive step: assume that

$$\sum_{b=1}^{n} \gamma_i^{t_b} \leq \sum_{a=0}^{n-1} \Delta_a$$

  - Then we need to prove

$$\left(\sum_{b=1}^{n} \gamma_i^{t_b}\right) + \gamma_i^{t_{n+1}} \leq \left(\sum_{a=0}^{n-1} \Delta_a\right) + \Delta_n$$

Sustainability analysis of C^{on-lim}

- **Reduced capacity**

Theorem 2: Assuming C^{on-lim}, a decrease of Δ in the execution times of higher priority tasks can only lead to a maximum increase of γ in the execution time of lower priority tasks where $\gamma \leq \Delta$

- ***"A decrease in timing requirement achieved by a reduced capacity cannot lead to an increase of timing requirement by preemption cost"***

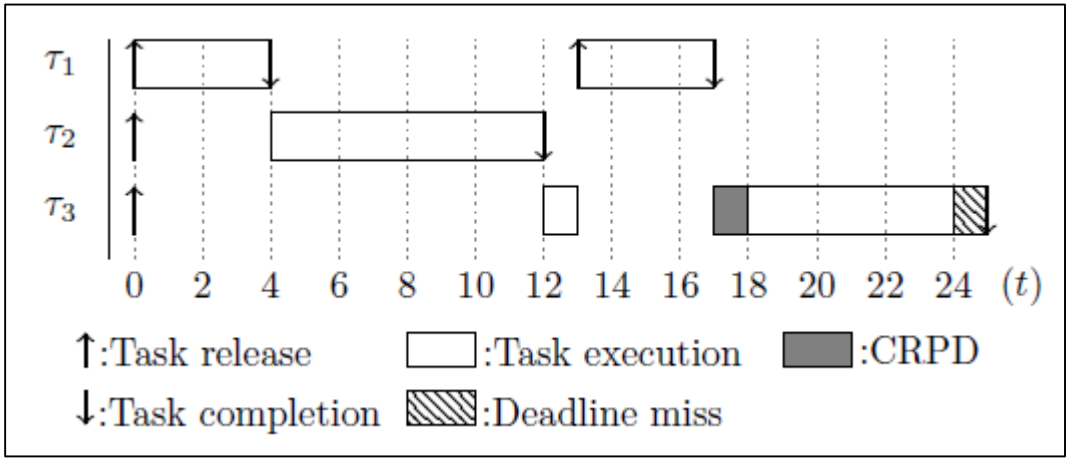
Theorem 3: Scheduling simulation with C^{on-lim} is sustainable with regard to the capacity parameter

Sustainability analysis of C^{on-lim}

- **Larger period**

Theorem 4: Scheduling simulation with C^{on-lim} is **not** sustainable with regard to the period parameter

- **Example 1 case 3**
  - Larger period
  - $T_1' = 13 > T_1 = 12$
  - τ_3 missed its deadline



Sustainability analysis of C^{on-lim}

- **Larger relative deadlines**

Theorem 5: Scheduling simulation with C^{on-lim} is sustainable with regard to the deadline parameter

- **Fixed priority preemptive schedule is generated independently from the deadline parameter**
  - Deadlines do not influence scheduling decisions
  - We do not investigate the cases where task priorities are reassigned according to new deadlines

Feasibility interval of C^{on-lim}

- **Synchronous task set**
  - $\boldsymbol{F = [0, H)}$, $\boldsymbol{H = lcm(T_i | \forall \tau_i \in T)}$
    - The known feasibility interval $[\mathbf{0}, \max(D_i))$ for synchronous task set is not applicable to systems with cache (Phavorin et al., 2017)

- **Asynchronous task set**
  - $\boldsymbol{F = [0, S_n + H)}$
    - S_n: the stabilization time of the lowest priority task (Audsley, 1991)
      - Tasks are ordered by their priorities
      - $S_1 = 0_1, S_i = \max(O_i, O_i + \left\lceil \frac{S_{i-1} - O_i}{T_i} \right\rceil \cdot T_i)$ $(i = 2,3, \dots, n)$
    - This is the known feasibility interval for asynchronous task set (Audsley, 1991). Our proof was heavily inspired by the work of Audsley in 1991

Outline

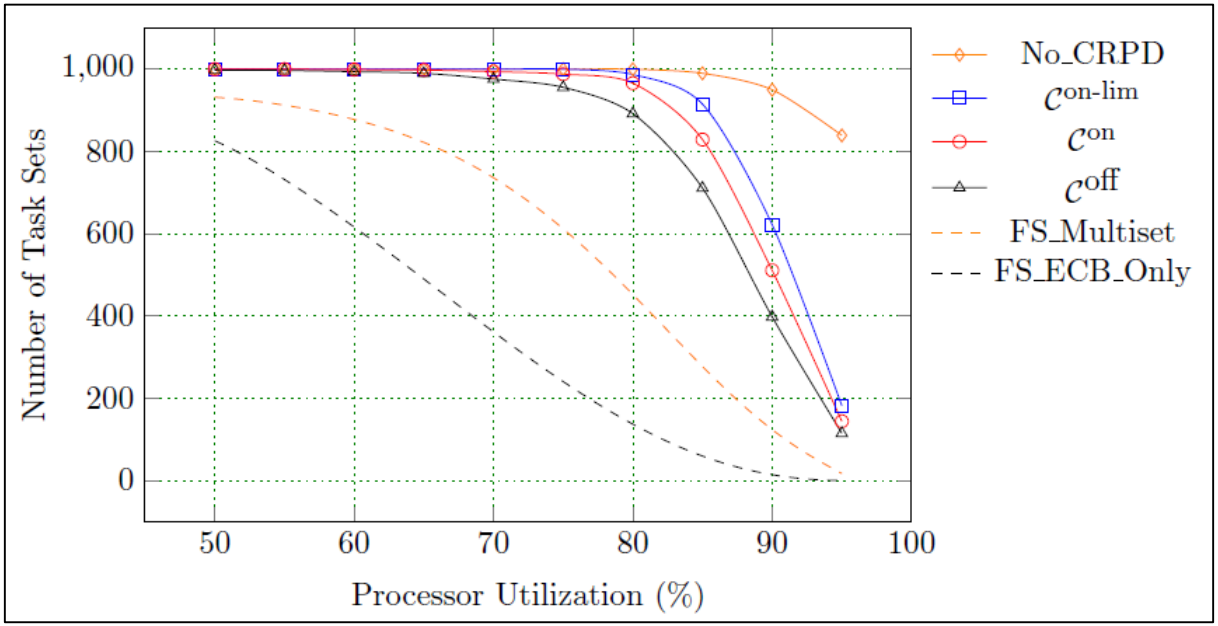
1. Introduction
2. Scheduling simulation with interference
3. CRPD-aware scheduling simulation
   - Problem statement & contribution
   - Related work
   - Background
     - CRPD computation models: C^{off} and C^{on}
     - Sustainability analysis
   - Approach
     - C^{on-lim}: an improved CRPD computation model
     - Sustainability analysis of C^{on-lim}
     - Feasibility interval of C^{on-lim}
   - **Evaluation**
4. Conclusion

Evaluation

- **Base configuration (Altmeyer et al., 2012)**
  - **Task configuration**
    - Harmonic task sets, periods uniformly generated from 5ms to 500ms
      - Number of tasks : 10
    - Processer utilization generated by the UUniFast algorithm
      - From 50% to 90% in step of 5
      - 1000 task sets per utilization
      - Task capacities are generated by taking into account the generated periods and processor utilizations
  - **Cache configuration**
    - Direct-mapped
      - Cache size = 256
      - $BRT = 8\ \mu s$
    - ECB: Cache usage of each task is determined by its number of ECB
      - Generated by UUniFast algorithm for a total cache utilization of 5
    - UCB: Number of ECB multiplies by a cache reuse factor
      - Cache reuse factor: 0.3

Evaluation

- **Schedulability task set coverage**



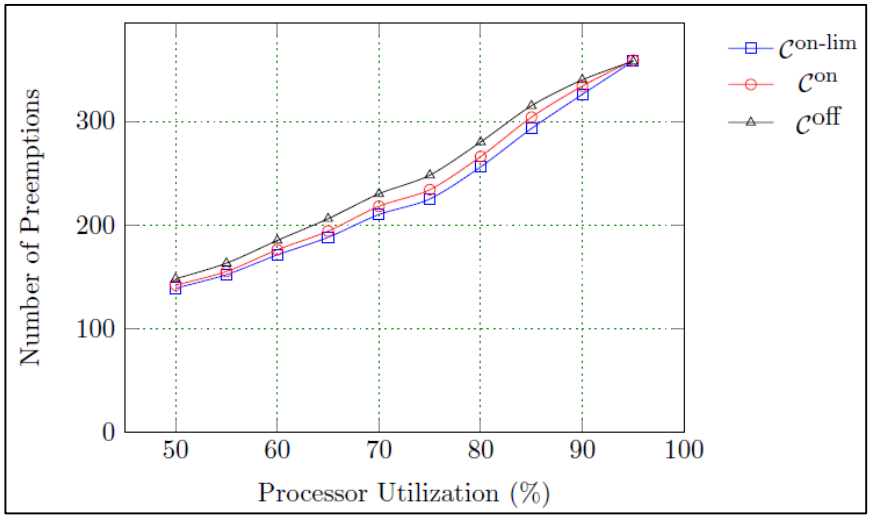
  - **Evaluate CRPD computation models and feasibility tests in term of schedulability task set coverage**

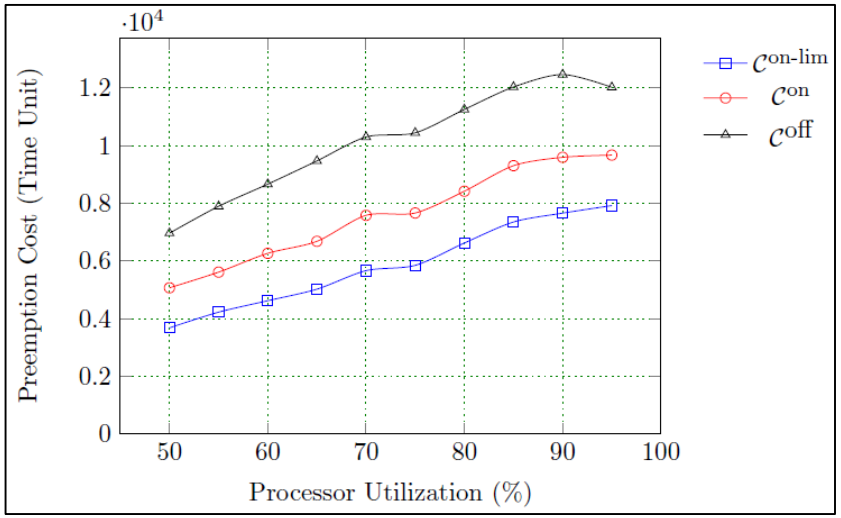
$$sched\_coverage = \frac{\#task\_sets\_schedulable}{\#generated\_task\_sets}\%$$

  - **C^{on-lim} have the highest coverage of 78%**

Evaluation

- **Preemption cost and number of preemptions**

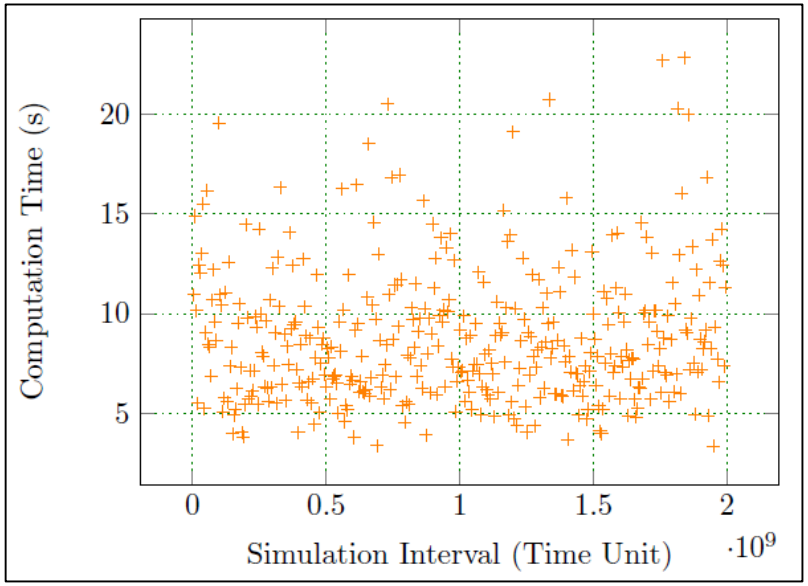


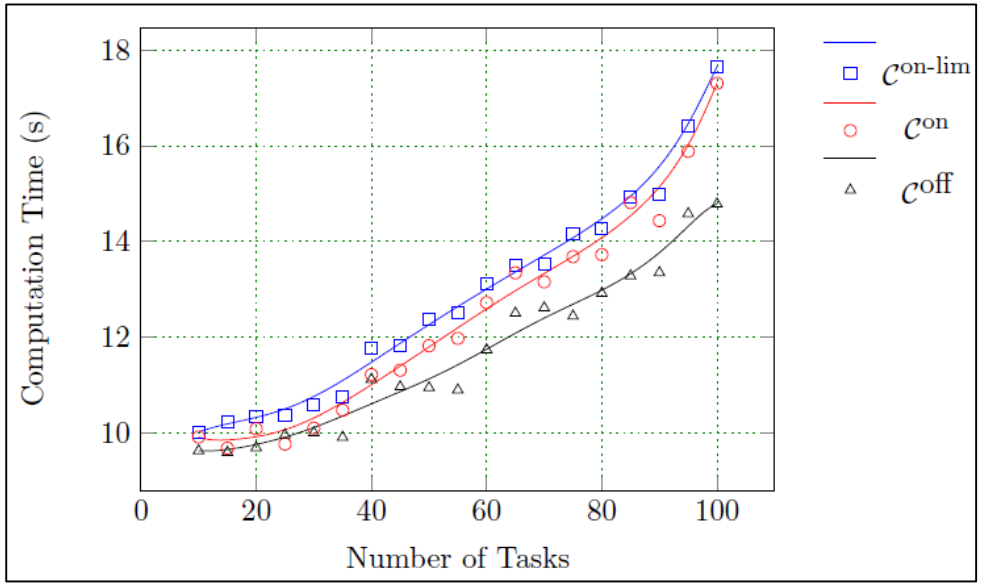


- C^{on-lim} number of preemptions is 7% less than C^{off} and 3% less than C^{on}
- C^{on-lim} preemption cost is 50% less than C^{off} and 30% less than C^{on}

Evaluation

- **Performance of CRPD-aware scheduling simulator**





- **The CRPD computation models are implemented in the Cheddar scheduling simulator**
  - Less than 25 seconds to run a simulation of $\mathbf{10^9}$ time units for a task set of **10**
  - Less than 18 seconds to run a simulation of **100** tasks in **2.000.000** time units
  - Simulation time is largely affected by the number of tasks
- **Computation time to export the complete event table is not taken into account**

Conclusion

- **We have investigated the case of scheduling simulation with CRPD**
  - **For uniprocessor systems with many hypothesis**
    - A tiny portion of the interference-aware scheduling simulation problem !
  - **Implementation in Cheddar scheduling simulator**
- **Identified problems**
  - **Modelling and computing the CRPD**
  - **Correctness: sustainability/feasibility interval**
  - **Technical problems: simulator implementation, I/O performance**
- **Other interference sources in Cheddar**
  - **Multi-core scheduling (Projet PLATO, Plasson et al., 2022), DRAM model based on the work of (Kim et al, 2016), Kalray memory model (Tran et al., 2019), Wormhole NoC Model (Dridi et al., 2021)**