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

Context

- **Real-time embedded systems (RTES)**
  - **Real-time: must process information and produce responses within a specified time, else risk severe consequences, including failure**
    - **Predictability**: compute system temporal timing behaviors at **design time**
    - **Real-time scheduling analysis:** verify the feasibility/schedulability of a system: feasibility tests, scheduling simulation
  - **Challenge: more and more parallelism and complexity at both software and hardware**
    - Even in small systems (drones, cars)
    - Legacy software are no longer executed on specific hardware
    - Less usage of dedicated hardware / more COTS (Federal aviation administration, Commercial Off The Shelf Avionics Software Study, 2011)
      - **Multicore architecture with memory hierarchy**

Context

- **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 in AADL Inspector
  - **Need of support for multi-core**
    - Usage of Cheddar in the Project PLATO (Plasson et al., 2022)

Problem Statement

- **Lack of scheduling simulator 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)**

Simulation without interference

Simulation with interference

- **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
   - **Related work**
   - **Background**
     - CRPD computation models: $\boldsymbol{C^{off}}$ and $\boldsymbol{C^{on}}$
     - Sustainability analysis
   - **CRPD-aware scheduling simulation**
     - $\boldsymbol{C^{on-lim}}$: an improved CRPD computation model
     - Sustainability analysis of $\boldsymbol{C^{on-lim}}$
     - Feasibility interval of $\boldsymbol{C^{on-lim}}$
   - **Evaluation**
4. Conclusion

Scheduling simulation with interference

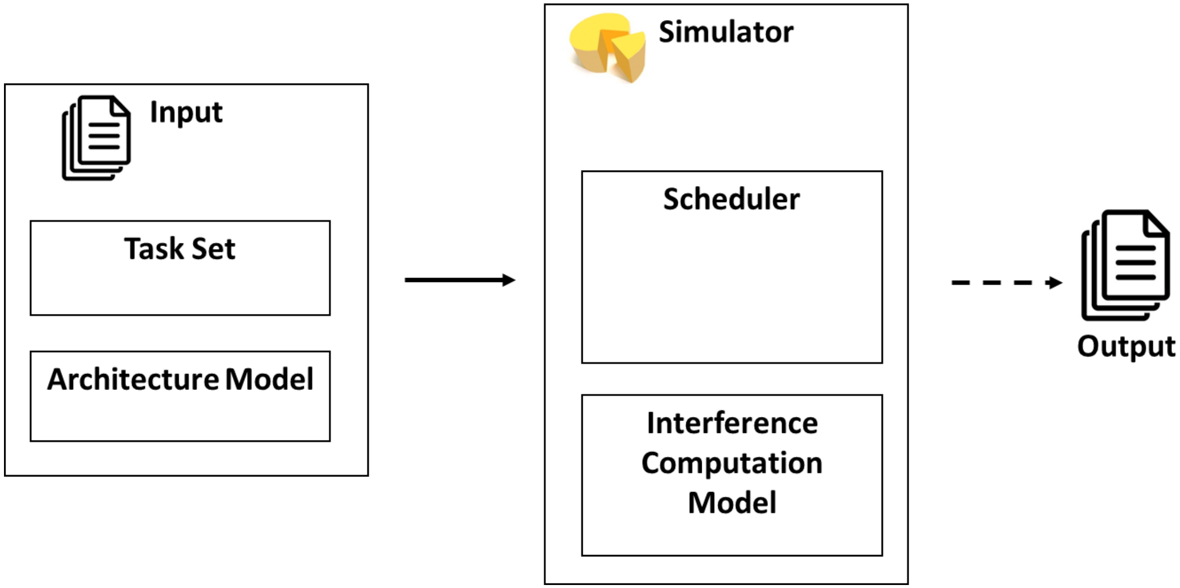
- **Why scheduling simulation ?**
  - **Advantages**
    - **Observed reduced pessimism** compared to static WCRT analysis
      - Used as a **sufficient condition** to compare interference-aware WCRT analysis
    - **Adaptability/Flexibility** - manageable integration of additional scheduling parameters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

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, e.g: level of abstraction (timing, system type) during simulation
    - **Engineering challenges** - how to implement the simulator
    - **2 theoretical problems: sustainability and feasibility interval** (to be detailed later)

Scheduling simulation with interference

- **Scheduling simulation**
  - **Simulation of a task set T on an architecture M under a scheduler S over an interval of time F**
- **Interference-aware scheduling simulation**
  - **Scheduling simulation with an interference computation model I**
    - Describe the method of computing the interference added to the execution time of a task during its execution



Scheduling simulation with interference

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

… and we found the following problems

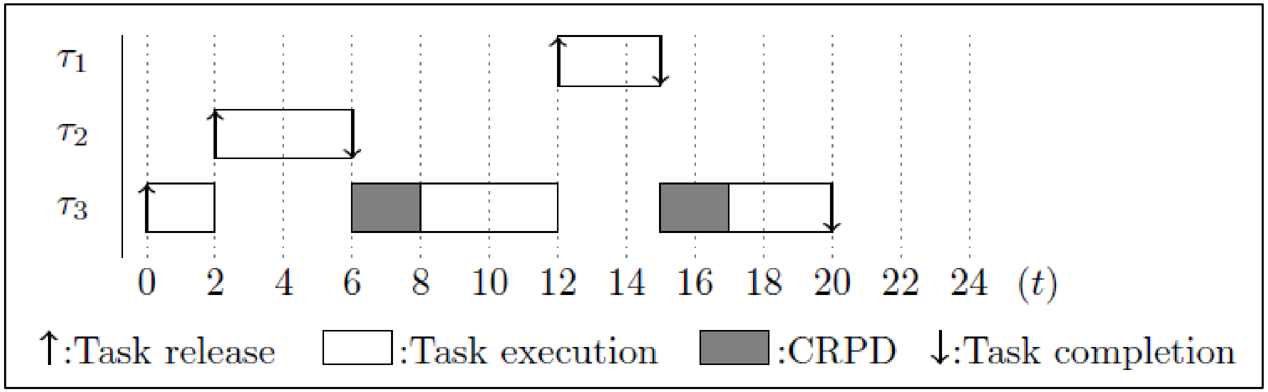
  - **Problem 1: How to model and compute the interference**
    - What should we consider to simulate the worst-cases
    - Pessimistic of the computation model
  - **Problem 2: Sustainability problem**
    - If a system is considered to be schedulable by simulation with the worst-case parameter, is it schedulable in better cases ?
  - **Problem 3: Feasibility interval problem**
    - How long should we simulate ?
  - **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**
   - **Related work**
   - **Background**
     - CRPD computation models: $\boldsymbol{C^{off}}$ and $\boldsymbol{C^{on}}$
     - Sustainability analysis
   - **CRPD-aware scheduling simulation**
     - $\boldsymbol{C^{on-lim}}$: an improved CRPD computation model
     - Sustainability analysis of $\boldsymbol{C^{on-lim}}$
     - Feasibility interval of $\boldsymbol{C^{on-lim}}$
   - **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: a system requires significantly more computing resources to be schedulable**
  2. **Non sustainable: a system is schedulable under its worst-case specification but not schedulable in better cases when interference is present**
  3. **Unidentified feasibility interval: how long should we run the simulation ?**

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

Related Work

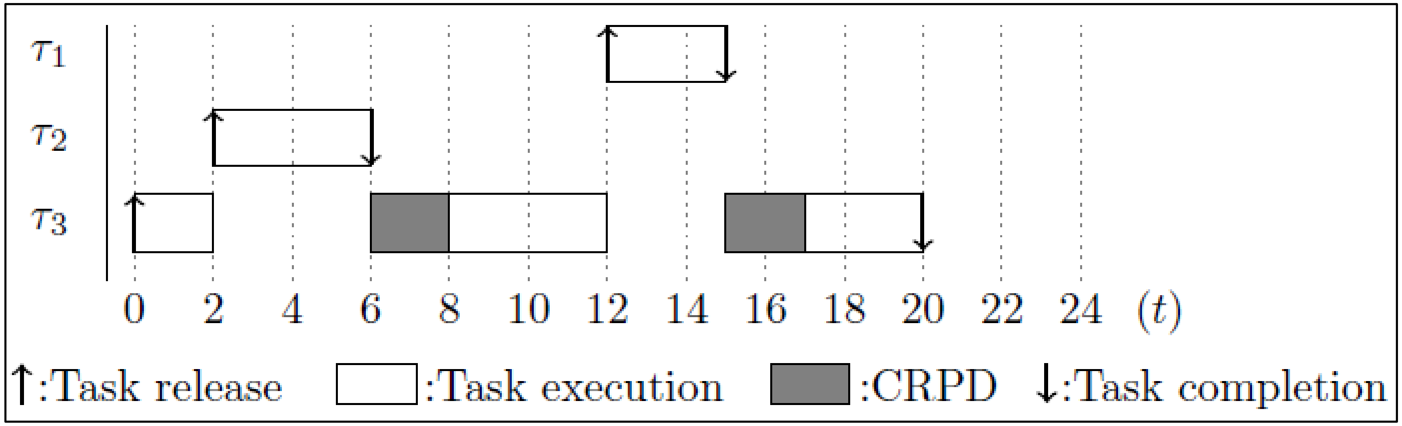
- **CRPD-aware scheduling analysis**
  - **Analytical-based approaches**
    - **CRPD-aware worst-case response time analysis**: Lee et al., 1998; Busquets-Mataix et al., 1996, Tomiyama et al., 2000; Staschulat et al., 2005; Altmeyer et al., 2012; Lunniss et al., 2014
    - **Eliminate or limit the effect 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 without cache support**
      - **MAST** (Harbour et al., 2001),
      - **STORM** (Urunuela et al., 2010),
      - **YARTISS** (Chandarli et al., 2012)
    - **Simulators with cache support**
      - **SimSo** (Cheramy et al., 2015): Stack Distance Profile
      - **Cheddar** (Tran et al., 2014): Useful Cache Block/Evicting Cache Block

Background

- **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 $\boldsymbol{\gamma_i}$ 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

- $\boldsymbol{C^{on}}$**: online CRPD computation model**
  - **For task $\boldsymbol{\tau_i}$ a set of useful cache blocks ($\boldsymbol{UCB_i}$) and evicting cache blocks ($\boldsymbol{ECB_i}$) 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**
  - $\boldsymbol{UCB_i^t}$**: the set of UCB of $\boldsymbol{\tau_i}$ in the cache at time $\boldsymbol{t}$**
  - $\boldsymbol{\tau_i}$ **is preempted by $\boldsymbol{\tau_j}$ at time t**

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

  - $\boldsymbol{\tau_i}$ **resumes execution at time t+Δ**

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

Background

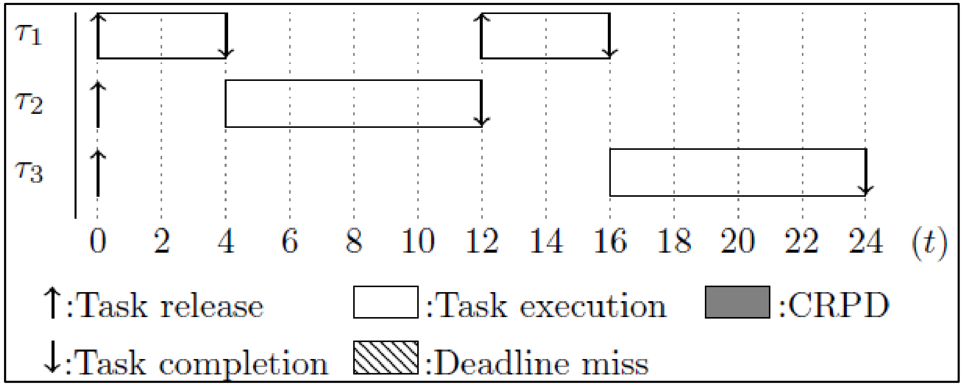
- **Sustainability analysis**
  - **Definition (Goossens et al., 1997): a given scheduling policy and/or a schedulability test is sustainable if any system that is schedulable under its worst-case specification remains so when its behavior is better than the worst-case**
  - **The term "better" means 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

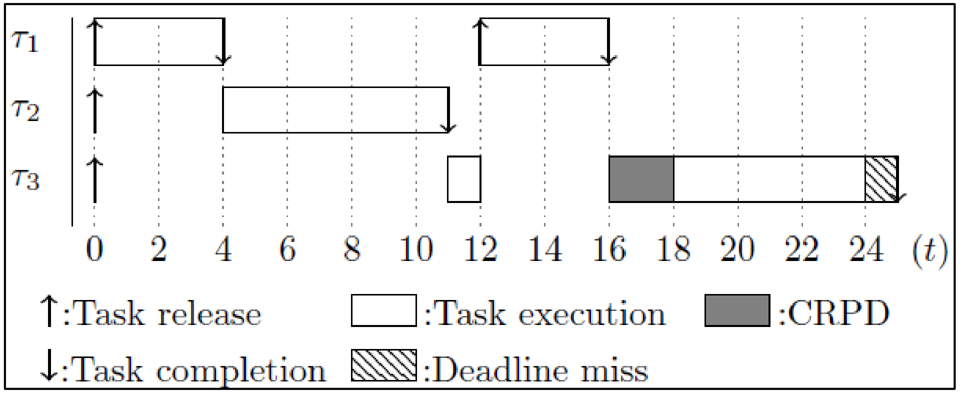
- **Sustainability of C^{on}**
  - **Example 1**

Task	C_i	T_i	D_i	O_i	Π_i	$\mathrm{UCB_i}$	$\mathrm{ECB_i}$
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}

  - **Case 1: original task set**



  - **Case 2: reduced capacity of τ_2**
    - $C'_2 = 7$ ($< C_2 = 8$)
    - τ_3 missed its deadline
    - **Non sustainable scheduling with regard to the capacity parameter**



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 $t + \Delta$**

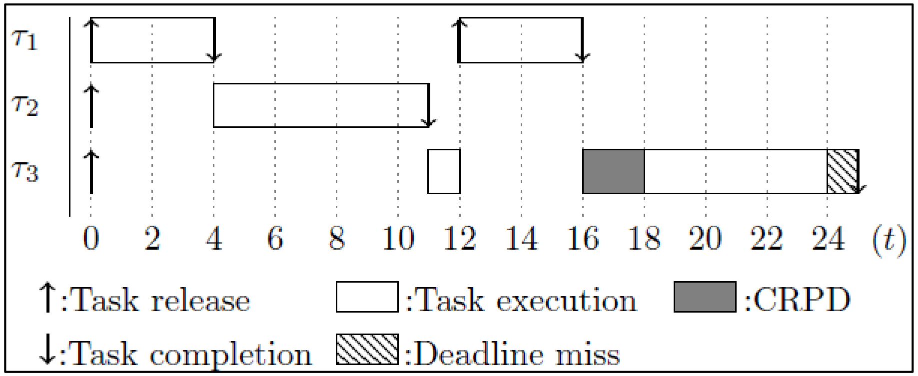
$$\rho_i^{t+\Delta} = min(|UCB_i|, \rho_i^t + \left\lfloor \frac{\Delta}{BRT} \right\rfloor)$$

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

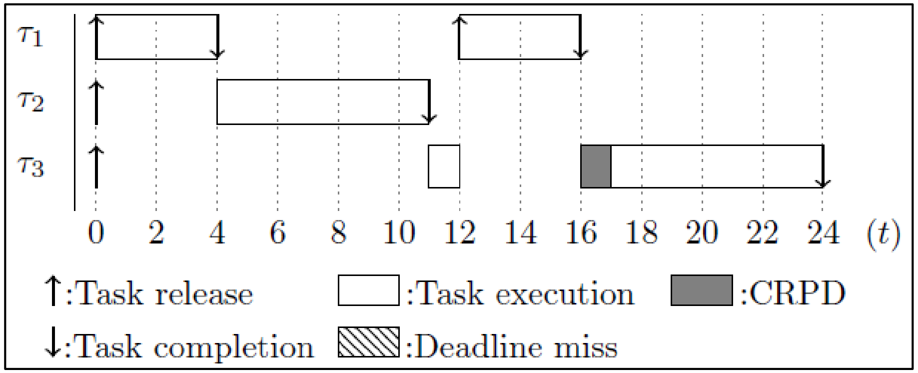
$$\gamma_i^t = min\left(\left|UCB_i - UCB_i^t\right|, \rho_i^t\right) \cdot BRT$$

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 $[\boldsymbol{t_a}, \boldsymbol{t_a} + \boldsymbol{\Delta_a}), \boldsymbol{a} \in (\mathbf{0}, \mathbf{1}, \dots, \boldsymbol{n}-\mathbf{1})$ and experiences the preemptions costs $\boldsymbol{\gamma_i^{t_b}}, \boldsymbol{b} \in (\mathbf{1}, \dots, \boldsymbol{n})$, we have:

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

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

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

  - Then we need to prove

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

"CRPD added to task is limited by its executed capacity"

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 a job of a lower priority task where $\gamma \leq \Delta$

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

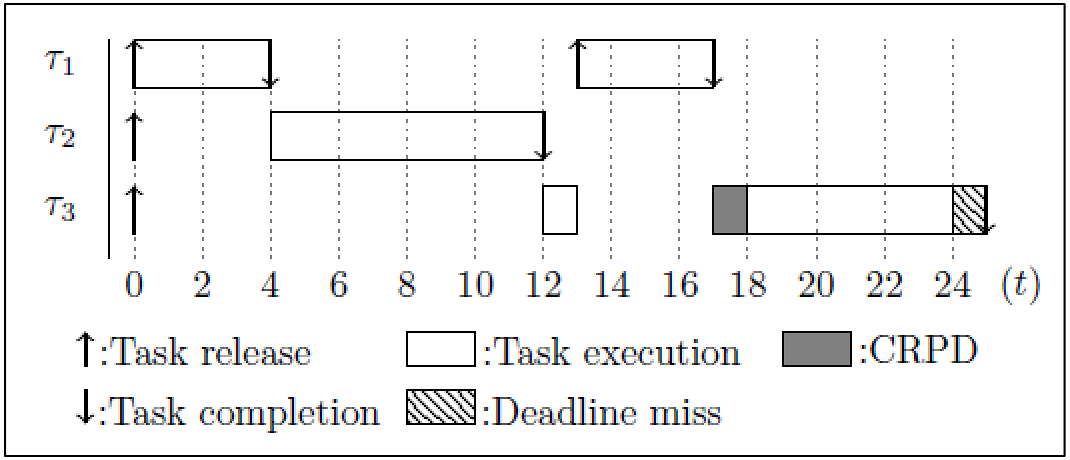
Theorem 3 : Scheduling simulation with C^{on-lim} is sustainable with regard to the capacity parameter

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

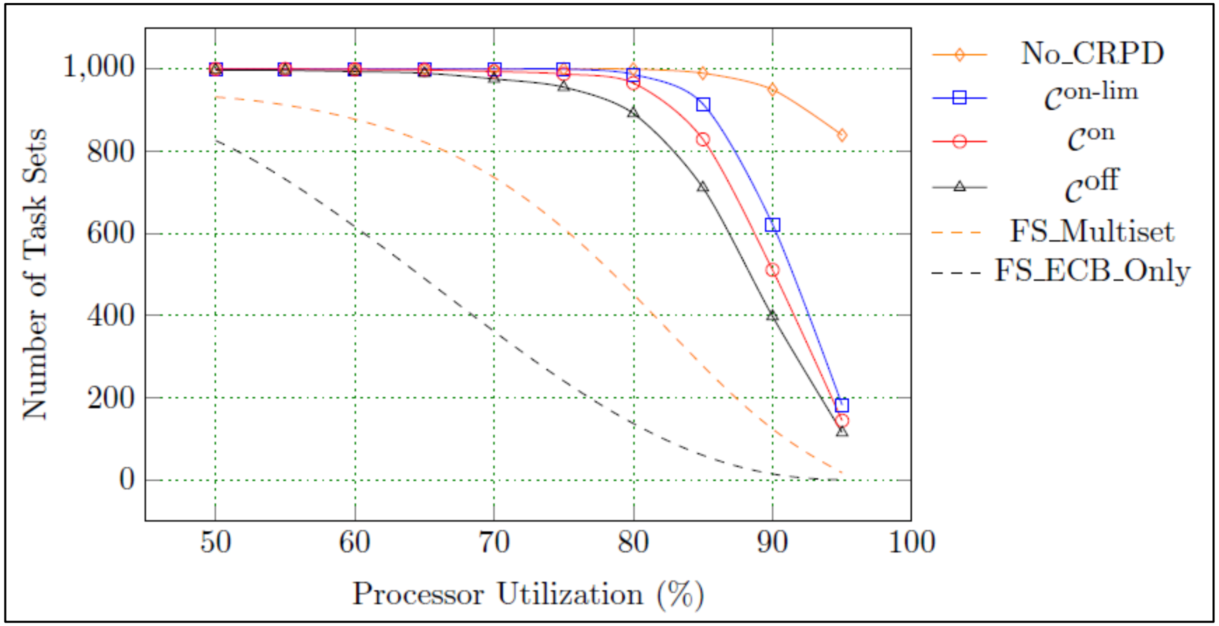
- **Asynchronous task set**
  - $\boldsymbol{F = [0, S_n + H)}$
    - S_n: the stabilization time of the lowest priority task (Audsley, 1991)
      - Tasks are ordered by their priorities
      - $S_1 = 0_1, S_i = \max(O_i, O_i + \left\lceil \frac{S_{i-1} - O_i}{T_i} \right\rceil \cdot T_i)\ (i = 2,3,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

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
      - 500 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**



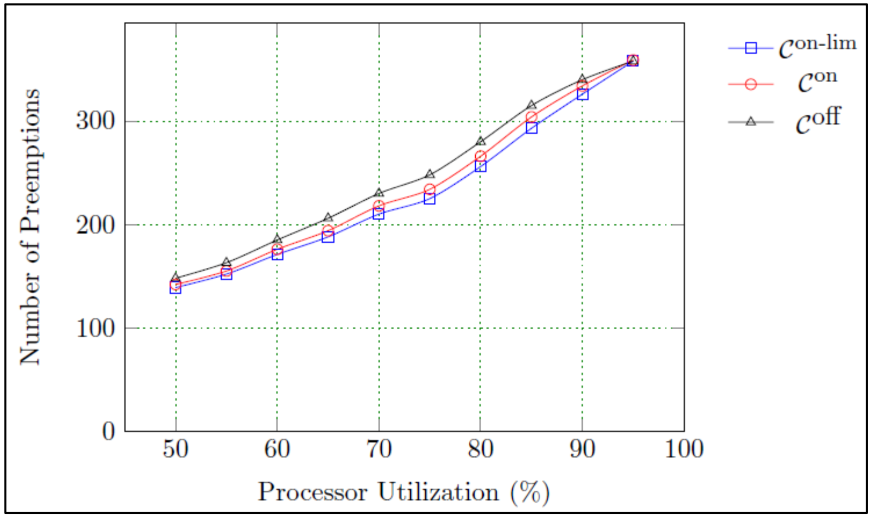
- **Evaluate CRPD computation models and feasibility tests in term of schedulability task set coverage**

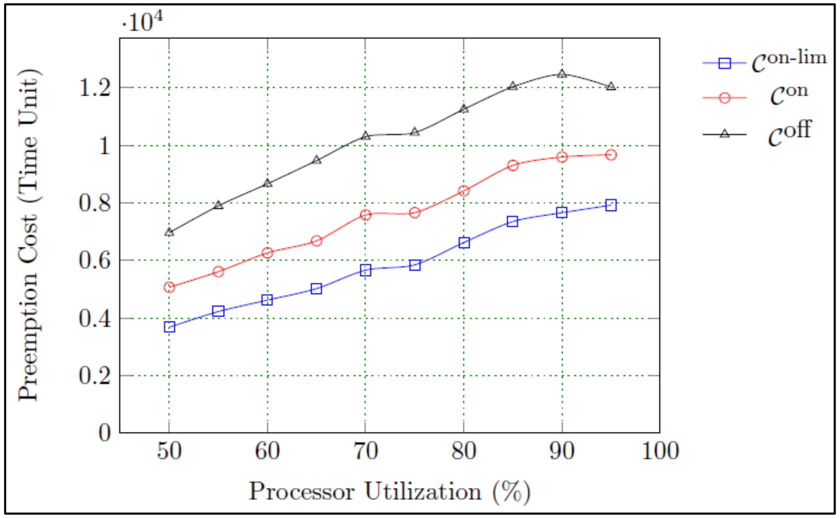
$$sched\_coverage = \frac{\#task\_sets\_schedulable}{\#generated\_task\_sets}\%$$

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

Evaluation

- **Preemption cost and number of preemptions**

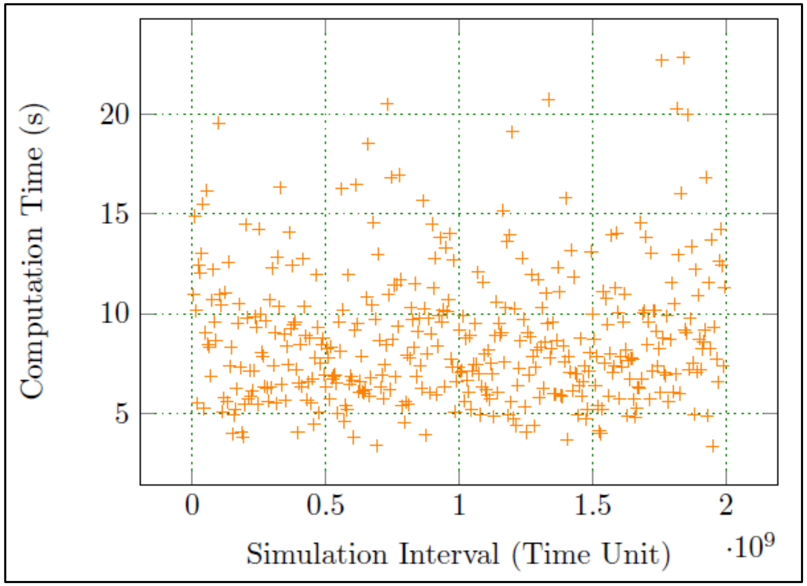


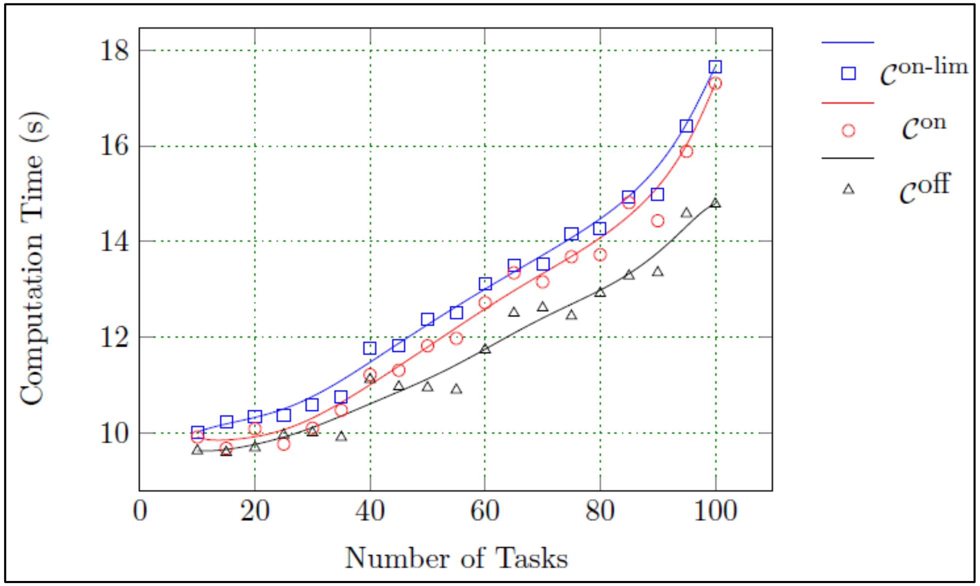


- $\mathcal{C}^{on-lim}$ **number of preemptions is 7% less than** $\mathcal{C}^{off}$ **and 3% less than** $\mathcal{C}^{on}$
- $\mathcal{C}^{on-lim}$ **preemption cost is 50% less than** $\mathcal{C}^{off}$ **and 30% less than** $\mathcal{C}^{on}$

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**

Outline

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

Conclusion & Future work

- **We have investigated the case of scheduling simulation with CRPD**
  - **For a uniprocessor systems with many hypothesis**
  - **A tiny portion of the interference-aware scheduling simulation problem !**
  - **Implementation in Cheddar scheduling simulator**
- **Problems identified**
  - **What to put in our models**
  - **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 (Kim et al, 2016)**
  - **Kalray memory model (Tran et al., 2018)**
  - **Wormhole NoC Model (Dridi et al., 2021)**