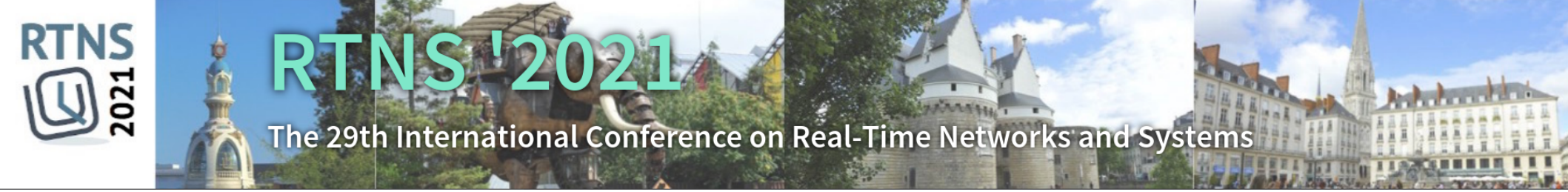


Migrating Constant Bandwidth Servers on Multi-Cores

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

AMPERE
A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

HORIZON 2020

Tommaso Cucinotta & Luca Abeni

Real-Time Systems Laboratory
Scuola Superiore Sant'Anna
Pisa, Italy

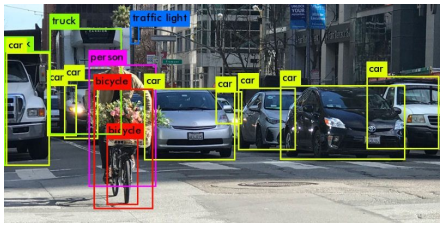
Background and motivations #1/2

Relentless growth of computing demand

- satisfied for years by the "**frequency-race**"
- up to the impossibility of handling **heat** dissipation
  - Intel's cancelled plans for 5GHz-10GHz CPUs in 2004
- => we've been living a "**multi-core race**" in the last decade

Multi-core/multi-processing pervasive

- not only in High-Performance Computing (HPC)
- but also in embedded and real-time systems
- noteworthy modern use-cases:
  high-performance embedded scenarios
  - intelligent real-time driving assistants
  - real-time video processing and object recognition
  - real-time time-series forecasting
- Need for a multitude of "acceleration" technologies:
  - **Multi-core**, GP-GPU, FPGA



.AMPERE

A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

Background and motivations #2/2

- **Open real-time systems**
  - **dynamic real-time task sets on multi-core platforms**
    - new RT tasks may ask to be admitted at any time
    - running RT tasks may terminate and leave the system
  - requirement of **minimizing energy consumption** (mobile devices)
- **Frequent migration of RT tasks among cores**
  - when using utilization-based heuristics for task/core assignment
  - **First-Fit**
    - **good**: simple utilization-based overall test (based on biggest U_i)
    - **bad:** for energy-management
    - on DVFS frequencies scale up/down => **need to migrate** tasks to restore FF invariant
  - **Worst-Fit**
    - **good**: load spread across cores
      - allows for keeping **lower DVFS frequencies**
    - **bad**: if a new big U arrives => **need to migrate** tasks to make room

Problem Presentation #1/2

Focus of the paper

- **Adaptive Partitioning** of RT tasks on **multi-cores**
  - focus on independent, periodic tasks
- RT tasks scheduled using **CBS/EDF reservations** (i.e., **SCHED_DEADLINE**)
  - each task with parameters (WCET C_i, T_i) has associated a **reservation** with (Q_i, P_i)
    - normally $Q_i = C_i$ and $P_i = T_i$
  - at any time tasks (and their reservations) are partitioned among cores
    - reservations $\{(Q_i, P_i)\}_i$ deployed onto each core respect

$$\sum_i \frac{Q_i}{P_i} \leq U^{lub} \qquad (1)$$

    - CBS handles a current budget $q_i(t)$ and abs deadline $d_i(t)$

Problem Presentation #2/2

- RT **tasks arrive dynamically** at run-time
- each RT task is **accepted** on a specific core
  - so that Eq. (1) is satisfied – incremental test
    - total utilization of core h at time t: $V_h(t)$ $V_h(t) = \sum_{\tau_i \in \Gamma_h(t)} U_i.$ (2)
- RT reservations may **need to be migrated** across cores
- **Focus of the paper**:
  - on-line tracking of $V_h(t)$
  - and its use for admitting new tasks/reservations

Migrating CBS servers #1/2

- When a **CBS with $U_i = Q_i/P_i$ leaves a CPU h** (migration or termination)
  - we **cannot immediately subtract U_i** from $V_h(t)$
  - otherwise the incremental test based on $V_h(t)$ **might mistakenly** admit new tasks that **would lead to deadline misses**
  - we can **safely do it**
    - on the **current deadline** $d_i(t)$
    - earlier, at the **0-lag time** $\delta_i = d_i(t) - q_i(t)/U_i$
  - a scheduler should track the 0-lag times
    - SCHED_DEADLINE in Linux has the *inactive timer* that can be used for this purpose

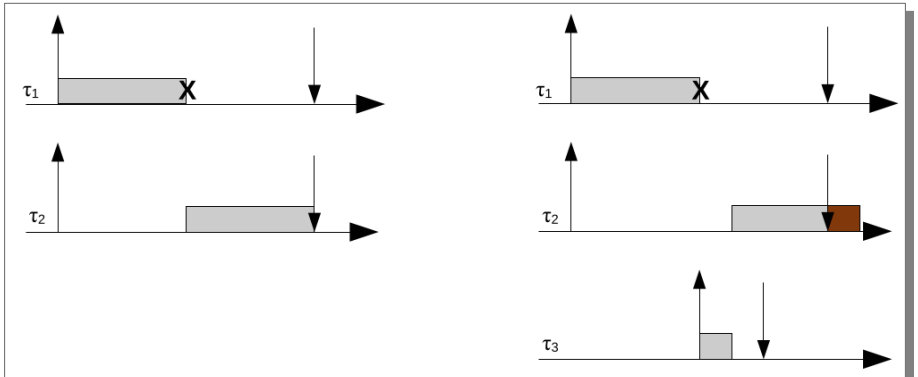


Figure 1: Example of deadline miss when the utilization of a task $t1$ that is removed from the CPU runqueue, is forgotten immediately and a new task $t3$ is mistakenly admitted too early.

Migrating CBS servers #2/2

- When a **CBS with $U_i = Q_i/P_i$ leaves a CPU h**
  - **decreasing immediately** U_i from $V_h(t)$
    => risk of deadline misses (**incorrect**)
  - **post-poning U_i subtraction** to $d_i(t)$ or 0-lag times
    => **safe**, but **pessimistic**
- In this paper, we demonstrate **we can do better**
  - **keep simplicity** of utilization-based admission tests
  - **refine how exactly a task is admitted, considering the (still active) utilization of tasks that recently left the CPU**

Proposed Admission Test

- Consider at each time t the sum of all (still active) utilizations of migrated reservations in the future

$$V_{i,h}^m(t) = \begin{cases} U_i \text{ if } \tau_i \notin \Gamma_h(t) \wedge t < \delta_i \\ 0 \text{ otherwise (if } \tau_i \in \Gamma_h(t) \vee t \geq \delta_i). \end{cases} \quad (4)$$

$$V_h^m(t) = \sum_{\tau_i \in \Gamma_h^m(t)} V_{i,h}^m(t). \quad (5)$$

- $V_h(t)+V_h^m(t)$ is a step function
- New task with $U_i = Q_i/P_i$ admitted when:
  - known utilization test:

$$V_h(t)+V_h^m(t)+U_i \leq U^{lub} \iff U_i \leq U^{lub} - (V_h(t)+V_h^m(t)) \quad (6)$$

$$Q_i \leq P_i \left(U^{lub} - V_h(t) - V_h^m(t)\right). \quad (7)$$

  - proposed budget test:

$$Q_i \leq P_i \left(U^{lub} - V_h(t)\right) - \int_{s=t}^{t+P_i} V_h^m(s)ds, \quad (8)$$

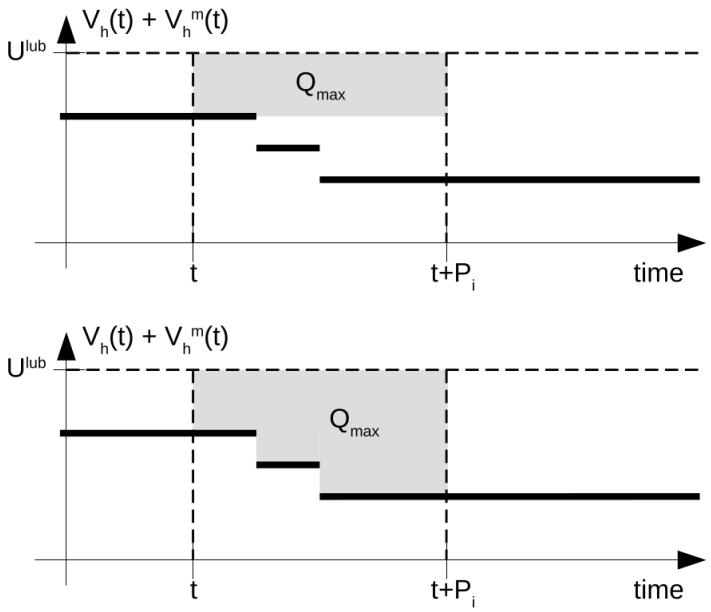


Figure 3: Highlight of the gain in the admitted budget for the proposed technique relying on Equation (8) (highlighted area in the bottom plot), compared to the one admitted using Equation (7) (highlighted area in the top plot).

Proof

- $\sigma(t')$: schedule when τ_i released at t
- $\sigma'(t')$: schedule when τ_i released at $t' < t$
- scheduler: EDF
- $d_i(t)$ unchanged
  => same prio in σ and σ'
  => τ_i cannot miss
- can other tasks miss due to the early release of τ_i ?
- t_0: time after which all jobs active in $[t',t]$ and with deadline earlier than $d_i(t)$ completed
- if $t_0 >= t$, then $\sigma = \sigma'$
- If $t_0 < t$, a task τ_j with $d_j(t) > d_i(t)$ might be delayed in $[t_0,t]$
  => a deadline miss can only happen for a task j with deadline $d_j(t) > d_i(t)$
- τ_i has Q_i time units in $[t, d_i(t)]$ in σ
- if τ_i preempts τ_j for an interval q_0 in $[t_0, t]$
  => q_0 of his time units in $[t, d_i(t)]$ are usable by τ_j to recover the delay
  => no deadline miss possible in σ'

LEMMA 6.1. *If a taskset $\Gamma = \{\tau_1, ...\tau_n\}$ is schedulable on a core h when task $\tau_i \in \Gamma$ is activated at time t with deadline $d_i(t)$ and budget $q(t)$, the activation time can be moved to $t' < t$ (leaving $d_i(t)$ unchanged) without breaking the taskset's schedulability.*

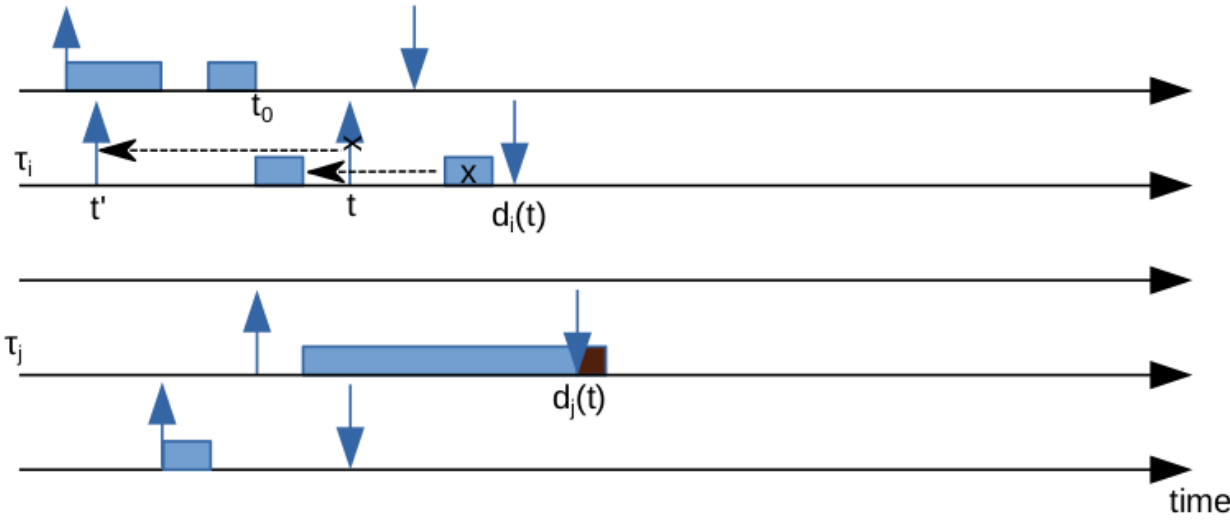


Figure 4: Visualization useful for a better understanding of the proof of Lemma 6.1.

Proof

- At **time *t***, we need to **admit a new reservation** (Q_{n+1}, P_{n+1})
- For each migrated utilization U_i scheduled to be subtracted by $V_h^m(t)$ at some time δ_i
- A new task of utilization U_i can safely enter the CPU at time δ_i
- All these new tasks may be given the **same absolute deadline**: $t + P_{n+1}$
- Their **release time** can be **safely moved back to time *t*** (for Lemma 6.1)
- They are **equivalent to a single task** with the maximum budget of Theorem 1

Theorem 1. *A new task τ_{n+1} served by a CBS (Q_{n+1}, P_{n+1}) can be admitted on core h at time t if the following condition is respected:*

$$Q_{n+1} \leq \left(U^{lub} - V_h(t) - \sum_{\tau_j \in \Gamma_h^m(t)} U_j \right) P_{n+1} + \sum_{\substack{\tau_j \in \Gamma_h^m(t): \\ \delta_j \leq t+P_i}} U_j \left(t + P_{n+1} - \delta_i\right).$$

Experimental Validation by Simulation

- EDF scheduler in RTSim
- Random task sets (Emberson's taskgen.py)
  - overall utilization U_{tot} of 0.90, 0.95 and 0.99
  - number of tasks n from 4 to 10
  - periods with log-uniform distribution in [1000, 2000], and period granularity of 100 time units
- Scenario (1000 repetitions for each value of U, n, k)
  - task sets started synchronously
  - **simulation paused at random times**, until we find a time in which there are *at least k tasks with 0-lag time in the future* ($k = 1, 2, 3$)
  - k of said tasks are **killed**
  - **new task added** to the CPU with
    - random period within the minimum and twice the maximum among the times to the deadlines of the k killed tasks
    - runtime Q^{new} equal to the maximum allowed by Theorem 1
  - simulation continued till 10 times the biggest period in the tasks set
  - verified that **no deadline misses** occurred
  - log maximum runtime Q^{old} according to old utilization-based test of Eq.(7) and the **bandwidth gain** defined as $(Q^{new} - Q^{old}) / Q^{old}$

Experimental Validation by Simulation

Table 1: Obtained maximum relative response times and average bandwidth gain for the simulated scenarios.

U_{tot}	k	maximum resptime/T	average bw gain
0.90	1	0.9000	2.03741
0.90	2	0.8973	2.99117
0.90	3	0.8973	4.21386
0.95	1	0.9500	3.23395
0.95	2	0.9467	5.18756
0.95	3	0.9467	7.77282
0.99	1	0.9900	12.8519
0.99	2	0.9867	22.8740
0.99	3	0.9867	35.3014

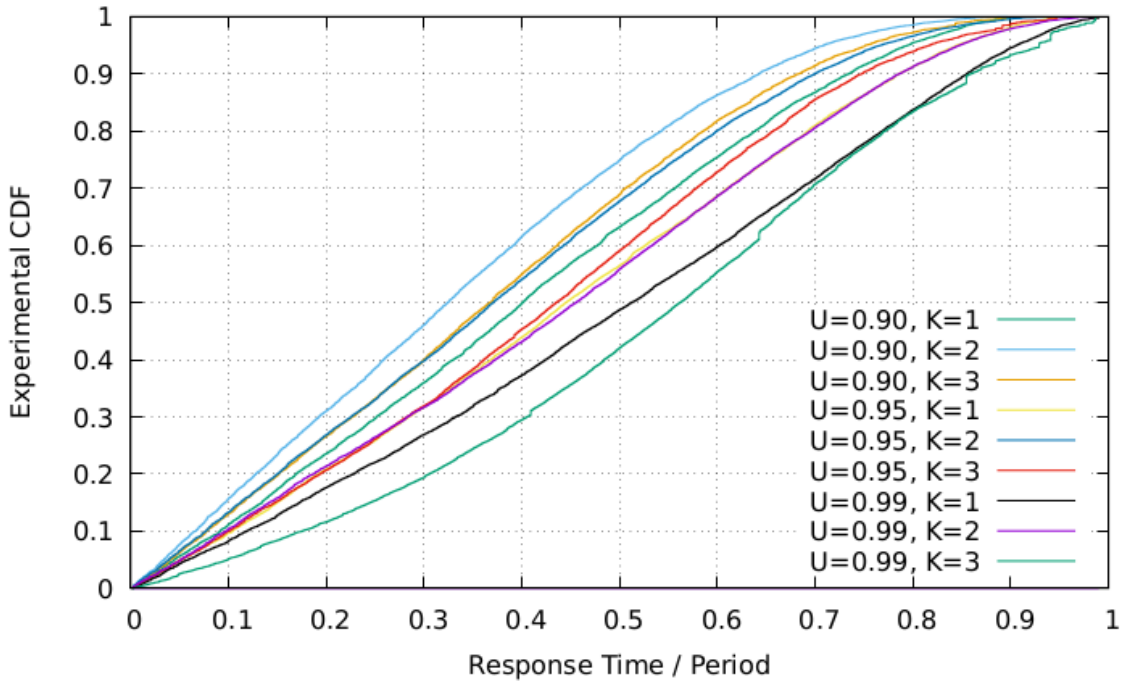


Figure 6: Experimental CDF of the response times relative to the periods in various configurations.

Conclusions

Proposed novel utilization test for admitting CBS reservations to a CPU

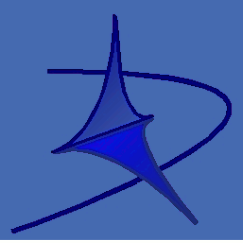
- less pessimistic than existing utilization-based test
- ease of use similar to a utilization-based test

Useful in scenarios with frequent migrations of RT tasks (and their reservations) among CPUs

- for example, with adaptive partitioning
- on mobile platforms, when applying DVFS changes to CPU frequency

Correctness of the new test

- Formally demonstrated
- Verified by simulations



Future Work

- Integrate within **Adaptive Partitioning** variant of **SCHED_DEADLINE** we have at RETIS
- Integrate with **power-aware** variant of **SCHED_DEADLINE** being investigated for **big.LITTLE** and **DynamIQ** Arm architectures
- Evaluate applicability in the context of the use-cases of the **AMPERE** EU project (2020-2022) https://ampere-euproject.eu/
  - Automotive
  - Tramway

.AMPERE

A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

Our related publications

Journals

- A. Mascitti, T. Cucinotta, M. Marinoni, L. Abeni. **"Dynamic Partitioned Scheduling of Real-Time Tasks on ARM big.LITTLE Architectures,"** Elsevier Journal of Systems and Software (JSS), Vol. 173, March 2021
- D. B. De Oliveira, R. S. De Oliveira, T. Cucinotta. **"A thread synchronization model for the PREEMPT_RT Linux kernel,"** Elsevier Journal of Systems Architecture (JSA), Vol. 107, August 2020
- L. Abeni, T. Cucinotta. **"EDF scheduling of real-time tasks on multiple cores: adaptive partitioning vs. global scheduling,"** ACM SIGAPP Applied Computing Review, Vol. 20, Issue 2 (June 2020), pp. 5-18, July 2020
- A. Balsini, T. Cucinotta, L. Abeni, J. Fernandes, P. Burk, P. Bellasi, M. Rasmussen. **"Energy-Efficient Low-latency Audio on Android,"** Elsevier Journal of Systems and Software (JSS), Vol. 152, pp. 182-195, June 2019
- **Improving Responsiveness of Time-Sensitive Applications by Exploiting Dynamic Task Dependencies**, Software: Practice and Experience, April'18

Conferences

- A. Mascitti, T. Cucinotta. **"Dynamic Partitioned Scheduling of Real-Time DAG Tasks on ARM big.LITTLE Architectures,"** 29th International Conference on Real-Time Networks and Systems (RTNS 2021), April 7-9, 2021, Nantes, France.
- D. B. de Oliveira, D. Casini, R. S. de Oliveira, T. Cucinotta. **"Demystifying the Real-Time Linux Scheduling Latency,"** 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), July 7-10, 2020, Modena, Italy.
- E. Quiñones, S. Royuela, C. Scordino, L. M. Pinho, T. Cucinotta, B. Forsberg, A. Hamann, D. Ziegenbein, P. Gai, A. Biondi, L. Benini, J. Rollo, H. Saoud, R. Soulat, G. Mando, L. Rucher, L. Nogueira. **"The AMPERE Project: A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimization,"** 23rd IEEE International Symposium on Real-Time Distributed Computing (IEEE ISORC 2020), May 19-21, 2020, Nashville, Tennessee, USA.
- A. Mascitti, T. Cucinotta, L. Abeni. **"Heuristic partitioning of real-time tasks on multi-processors,"** 23rd IEEE International Symposium on Real-Time Distributed Computing (IEEE ISORC 2020), May 19-21, 2020, Nashville, Tennessee, USA.
- G. Serra, G. Ara, P. Fara, T. Cucinotta. **"An Architecture for Declarative Real-Time Scheduling on Linux,"** 23rd IEEE International Symposium on Real-Time Distributed Computing (IEEE ISORC 2020), May 19-21, 2020, Nashville, Tennessee, USA.

Thanks!

Questions ?

tommaso.cucinotta@santannapisa.it
http://retis.santannapisa.it/~tommaso