UNIVERSITY OF PADOVA
Department of Mathematics

Evaluating a multicore Mixed-Criticality System implementation against a temporal isolation kernel

M. Bottaro
mattiabottaro9@gmail.com

T. Vardanega
tullio.vardanega@unipd.it

DIPARTIMENTO MATEMATICA

Are MCS practical alternatives to TSP? – 1

- **Time and Space Partitioning (TSP)** is the standard industrial practice in the critical-systems domain
- The application is split into multiple partitions according to per-task criticality ranking (LO, …, HI)
- Each partition is statically assigned a segregated area of processor memory
- And it is attached to a static cyclc scheduling plan
- The implementation technology needs to enforce memory isolation and strict adherence to scheduling plan
- HI-crit certification (very costly) is applied *solely* to HI-crit partitions
- Resources are apportioned *conservatively*, which incurs waste

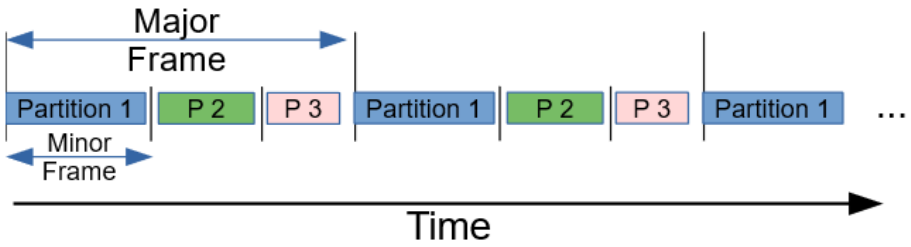


Image source: https://www.rapitasystems.com/blog/assured-partitioning-face-systems

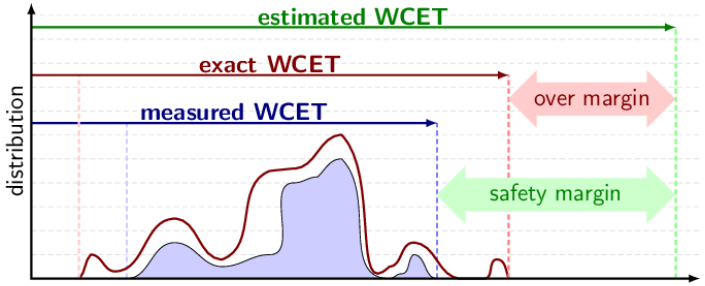


Image source: https://hal.archives-ouvertes.fr/hal-02271379

Are MCS practical alternatives to TSP? – 2

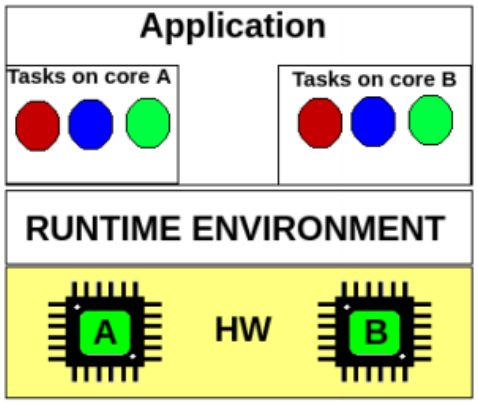
- **Mixed Criticality Systems (MCS)** models have been explored in the RTS literature to **higher utilization** without jeopardising HI-crit guarantees
  - Where TSP budgets CPU time according to extreme cases
  - MCS budgets according to average cases with contingency strategies to mitigate transient overload situations
  - Only temporal isolation to date
- An interesting MCS model for multicore targets uses controlled migration to further improve CPU utilization (Xu & Burns, JSS, 2019)
- The question becomes: what's the gain over TSP?

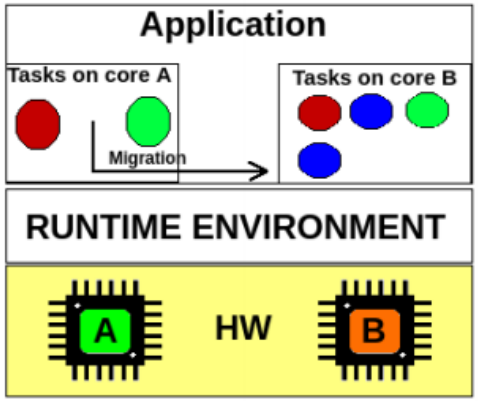
- Our contribution is a reference OSS implementation of the Xu & Burns' model, with featherweight event tracing capabilities
- An infrastructure to create synthetic workloads and stress-test scenarios for equivalent MCS vs TSP mapping
- Automation engines to run controlled experiments, capture trace logs, and analyse the results comparatively

The Xu & Burns model – dual-core implementation

Static view

- A task can be: LO-crit (green), HI-crit (red), migratable (blue)
  - Migratable tasks are LO-crit
- LO-crit tasks have LO-crit budget
- HI-crit tasks have {LO-crit, HI-crit} budgets





Dynamic view

- Scheduling is partitioned
- Tasks begin to execute LO-crit budget
- Both cores are in LO-crit mode

- If a HI-crit task exceeds its LO-crit budget, its CPU starts to execute in HI-crit mode
  - HI-crit tasks on that core may execute up to their HI-crit budget
  - Migratable tasks on that core are moved to the other CPU

- Test challenges occur when
- Both cores enter HI-crit mode
- Any task exceeds its limit budget
- Any task misses its deadline in spite of being deemed feasible

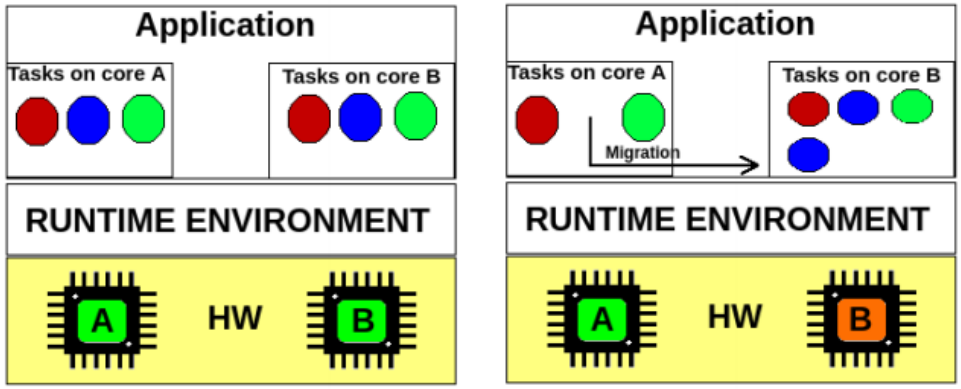
The working tools

Processor target

Digilent Cora Z7, Zynq-7000 SoC, dual-core ARM Cortex-A9, 333 MHz

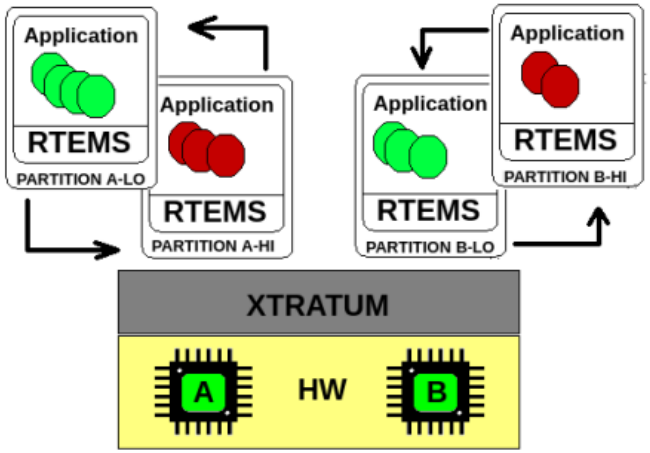
MCS Reference Implementation

- The application is written in Ada 2012
- The Ada runtime is modified and extended to support the required MCS and tracing features
- The basis was provided by
  - https://doi.org/9.1016/j.sysarc.2021.102236
- Derived from GNAT CE 2018 provided by AdaCore
- Distributed under GPL v3.0



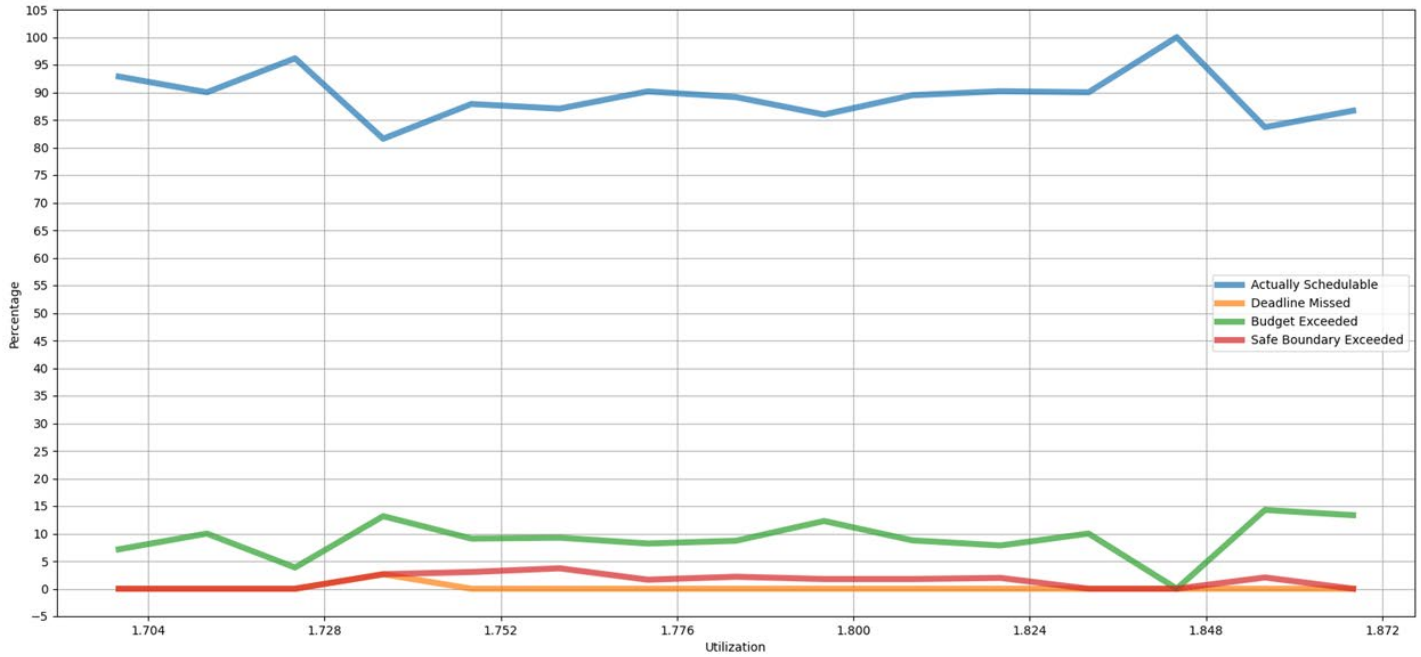
TSP Trial Implementation

- XtratuM 2.0.5 hypervisor by fentISS
- Each CPU hosts two partitions (LO-crit, HI-crit)
- Each partition runs RTEMS and the same Ada application as the MCS counterpart



Fine-tuning the experiments – 1

Actually Schedulable	Deadline Missed	Budget Exceeded	Safe Boundary Exceeded
88.98%	0.16%	***9.27%***	1.60%

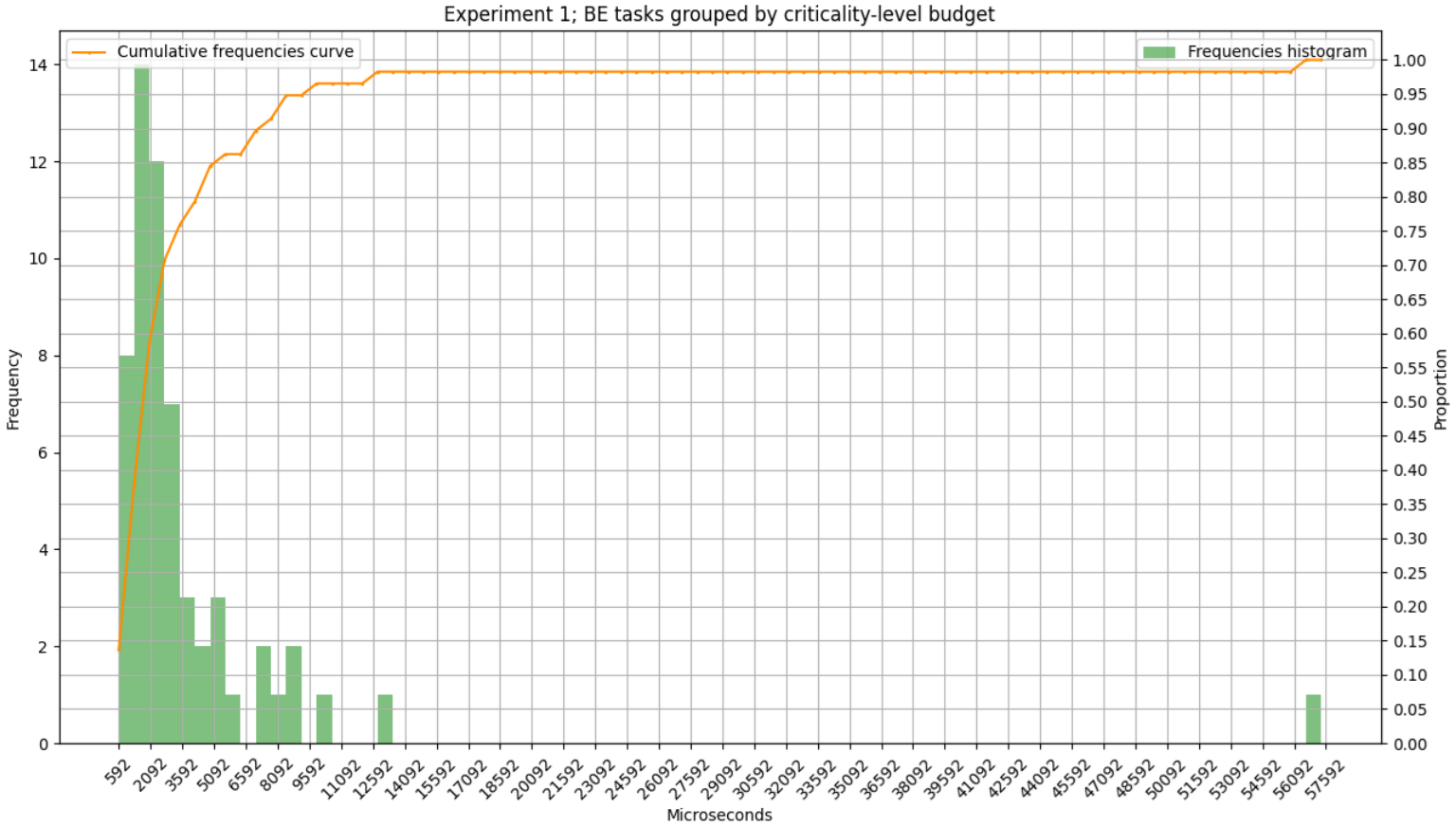


Ratio of LO-crit to HI-crit: 2. Taskset size: 12. Max harmonic: 2.
Small periods: [10, 200] ms. Large periods: [400, 1000] ms.

Fine-tuning the experiments – 2

Tasks that are found to exceed their limit budget at run time invalidate the experiment

Why does that happen?

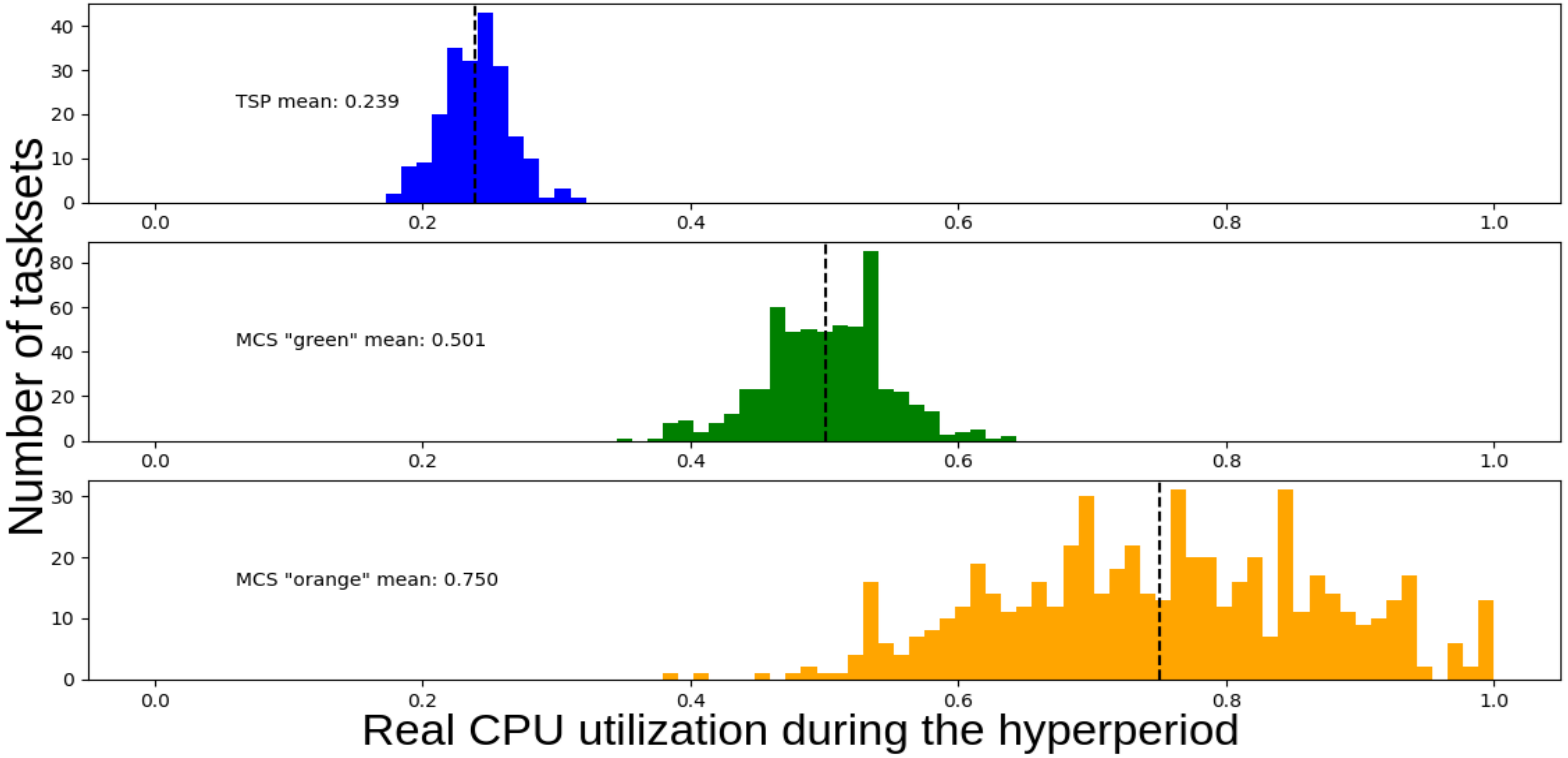


More than 85% of such tasks have a *tiny* limit budget (smaller than 3.6 ms)

Highlight findings

- Blue: CPU utilization measured on TSP Variant
- Orange: CPU utilization measured on MCS Reference Implementation while hosting the **whole** set of migratable tasks
- Green: CPU utilization measured on MCS Reference Implementation with *no* migration

The data shown are measurements performed on a single CPU



Conclusions

- The Xu & Burns' MCS model is **reality-proofed**
  - For each utilization level in the upper half, more than 80% of the tasksets that we run were found to be *actually schedulable*
  - Deadline miss events were very rare in spite of the system load being pushed near the limit steadily
  - The tasks that proved to be most difficult to control (without the risk of failure events) all had very small execution-time budgets
    - More plausible for interrupt handlers than for real application tasks
- The **MCS Reference Implementation** was found to sustain much **higher utilization** than the **TSP** equivalent
- For MCS to become industrial-proof alternatives to TSP they should be augmented with memory isolation features

- **Useful links:**
  - The runtime environment
  - The working repository
  - All the graphs and plots