Arbitration-Induced Preemption Delays

Farouk Hebbache[1]
with Florian Brandner,[2] Mathieu Jan,[1] Laurent Pautet[2]

[1]CEA List, L3S
[2]LTCI, Télécom Paris

Memory Arbitration

Memory arbitration scheme in multi-core architectures

Time-Division Multiplexing (TDM)

- Fixed time windows
- Exclusive memory access

- Isolation between cores
- More precise[1] analysis techniques compared to Round-Robin

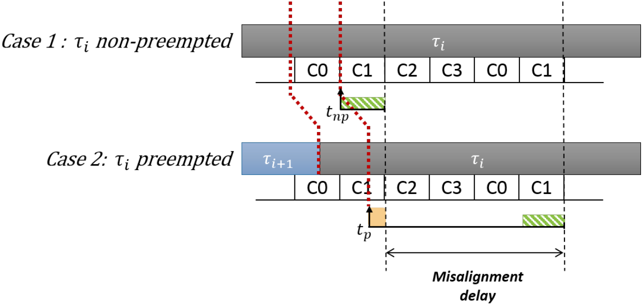
- Low memory utilization since TDM is non-work-conserving

Goal: improve memory utilization while keeping TDM guarantees considering a preemptive system model.

- Improve the execution times of non-critical tasks
- Support more non-critical tasks in the system

[1] H. Rihani et al, WCET Analysis in Shared Resources Real-Time Systems with TDMA Buses, RTNS 2015

Resuming After a Preemption

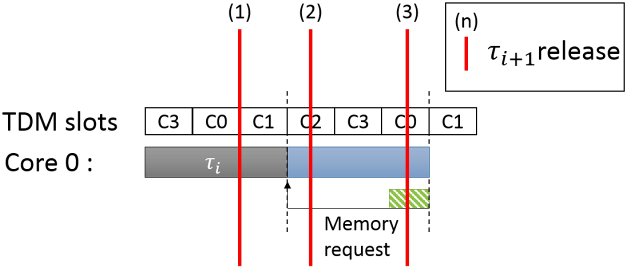


TDM MisAlignment Delay (*MA*)

Number of additional clock cycles of a memory request, w.r.t. the worst-case analysis,[2] when resuming after a preemption.

[2] H. Rihani et al, WCET Analysis in Shared Resources Real-Time Systems with TDMA Buses, RTNS 2015

Preemption



Memory Blocking Delay (*MB*)

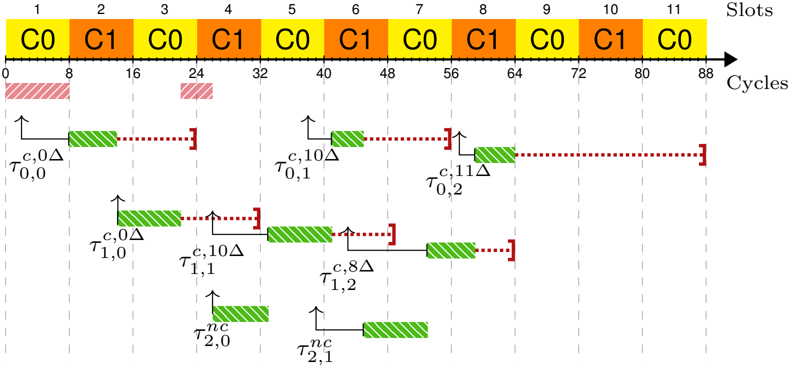
Number of clock cycles a higher-priority task τ_{i+1} is blocked by a pending memory request of a lower-priority task τ_i.

Dynamic TDM-Based Arbitration

Dynamic TDM-Based Arbitration [1,2]

- **Goal:** Eliminate TDM 's non-work-conserving sources . . .
  - **[1]** F. Hebbache et al, "Dynamic Arbitration of Memory Requests with TDM-Like Guarantees," (Workshop CRTS'17)
  - **[2]** F. Hebbache et al, "Shedding The Shackles of Time-Division Multiplexing," (RTSS'18)
- **How?**
  - Criticality-aware arbitration
  - Each critical memory request is associated with a deadline
  - Deadline/slack driven arbitration
  - Track slack when critical requests complete before deadline
  - Schedule request at any moment
  - Critical request will **always respects their deadline**
  - **Converges to traditional `TDM` in the worst-case**

Dynamic TDM-Based Arbitration [1,2]



Restricted system model: only one task per core

Critical tasks τ_0^c and τ_1^c with dedicated TDM slots as well as non-critical task τ_2^{nc}.

More slack $\longrightarrow$ further is the deadline.

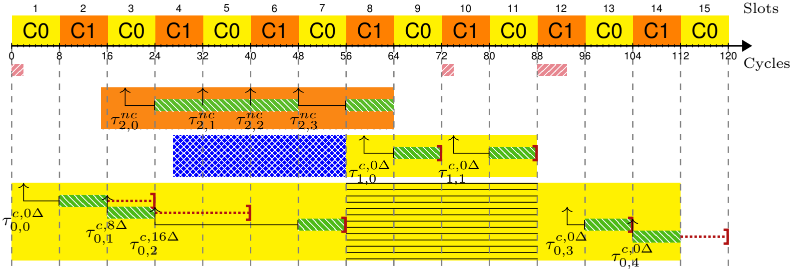
Preemption Effects under the Dynamic Approach

Preemption Effects for Dynamic Approaches

Dynamic approaches inherit the memory blocking and misalignment delays from TDM.

- Memory blocking delay:
  - Non-critical request might be unbounded (if no TDM slot is allocated to the core)
  - The **slack** accumulated may **increase** the MB

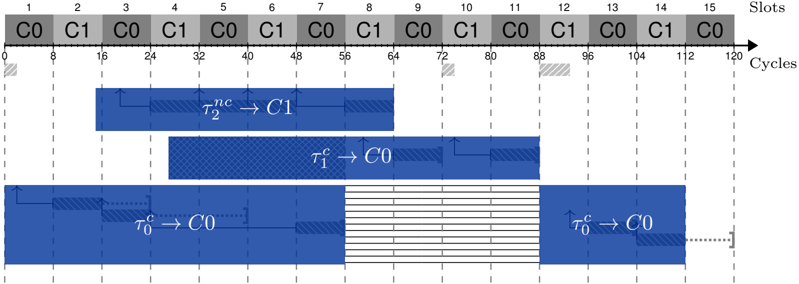
Example: Scheduling Wait (`SHDw`)



Critical tasks τ_0^c and τ_1^c on core C0
as well as non-critical task τ_2^{nc} on core C1.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

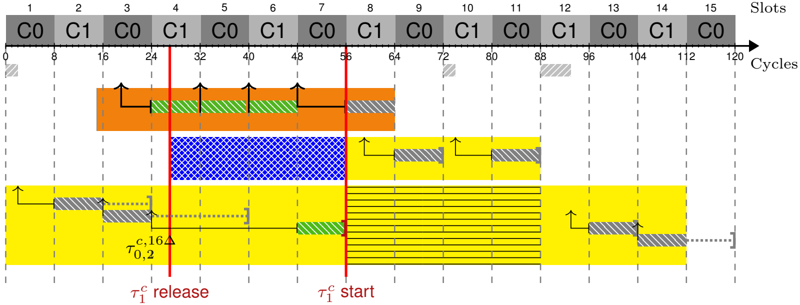
Example: Scheduling Wait (SHDw)



Critical tasks τ_0^c and τ_1^c on core C0
as well as non-critical task τ_2^{nc} on core C1.
Tasks execution times

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Example: Scheduling Wait (SHDw)



Memory blocking delay induced by request $\tau_{0,2}^{c,16\Delta}$ on critical task τ_1.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Contributions in the Paper

- Three approaches to handle the memory blocking delay:
  - `SHDw`: Wait until request completion
  - `SHDp`: Preempt pending request
  - `SHDi`: Criticality inheritance (update current deadline)
- Preemption mechanism
  - Support for delayed preemptions
  - No perturbation of preempted task
- Response-Time Analysis (RTA) for each approach

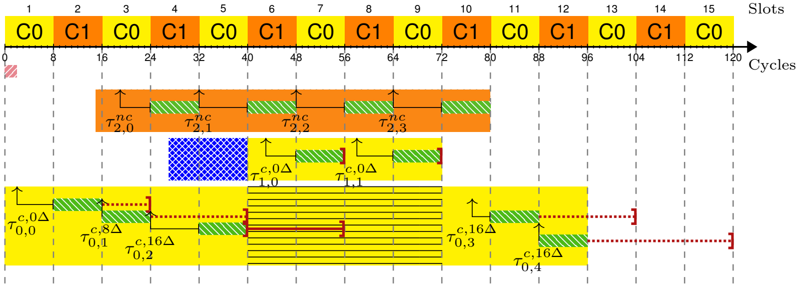
Scheduling Inheritance (SHDi)

- **Goal:** Bound the memory blocking delay of preemptions.
- **How?**
  - Control the impact of slack accumulation
  - Impose a new deadline on a pending request
  - Regardless of the criticality of the preempted task
    → Non-critical tasks may briefly become critical.

The deadline will certainly fall within the current or next `TDM` period.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

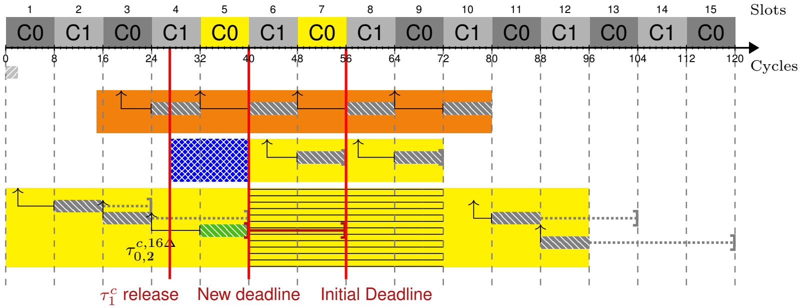
Example: Scheduling Inheritance (SHDi)



Critical tasks τ_0^c and τ_1^c on core C0
as well as non-critical task τ_2^{nc} on core C1.
Update pending request deadline $\tau_{0,2}^{c,16\Delta}$.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Example: Scheduling Inheritance (`SHDi`)



Update deadline of pending request $\tau_{0,2}^{c,16\Delta}$ at τ_1^c release.

The new deadline always falls within the current or next `TDM` period.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Response-Time Analysis

Response-time analysis equations:

$$R_i^{n+1} = C_i + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Response-Time Analysis

Response-time analysis equations:

$$R_i^{n+1} = (C_i + \mathbf{MB_i}) + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

Memory blocking delay induced on task τ_i
by a lower priority task.
Only once, at τ_i's release.

Response-Time Analysis

Response-time analysis equations:

$$R_i^{n+1} = (C_i + MB_i) + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil (C_j + \mathbf{MB_j})$$

Memory blocking delay induced
on higher priority tasks τ_j.
On every higher-priority task preemption.

F. Hebbache, F. Brandner, M. Jan, L. Pautet

Response-Time Analysis

Response-time analysis equations:

$$R_i^{n+1} = (C_i + \mathrm{MB}_i) + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil ((C_j + \mathrm{MB}_j) + \textbf{MA})$$

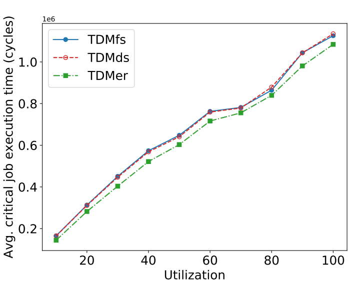
Misalignment delay induced
by higher priority task τ_j.
Each time some tasks (τ_i or τ_j) resumes from a preemption.

Experiments

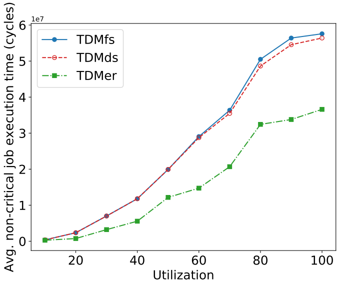
Benchmark Setup

- Traditional `TDM` with non-critical tasks as baseline (`TDMfs`)
- `TDMds`: Deadline/slack driven arbitration
- `TDMer`: `TDMds` + independent from TDM slots
- Overall 12 600 simulation runs
  - Based on randomized memory traces
    (calibrated from actual traces of MiBench on Patmos)
- **Evaluation metrics:**
  - Average job execution times
  - Average schedulability success rates
  - Maximum memory blocking delays

Results: Average job execution times



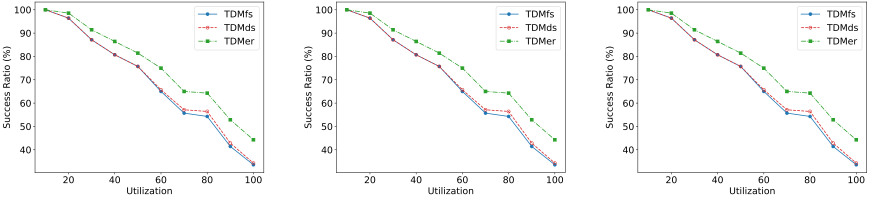
(a) Critical job execution time.



(b) Non-critical job execution time.

Improved average execution times of non-critical jobs.

Results: Schedulability

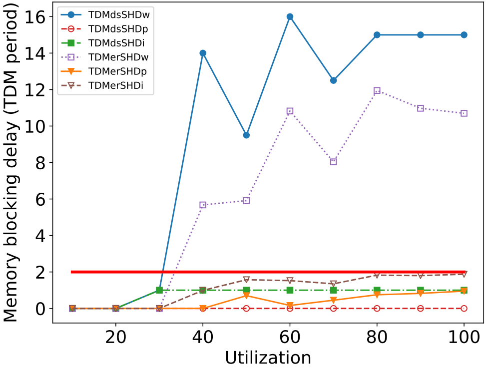


(a) SHDw (b) SHDp (c) SHDi

Average *schedulability* success ratio under varying normalized system utilization.

Overall, no significant difference in the success ratio.

Results: Memory Blocking Delay



Maximum memory blocking delay for critical tasks under varying normalized system utilization.

$$MB^{\texttt{SHDi}} \leq 2 \cdot P$$

Conclusion

- Dynamic `TDM`-based arbitration
  - Improve memory utilization (work-conserving)
  - Guaranteed progress for critical tasks (converging to `TDM`)
  - **Inherit and amplifies the memory blocking delay**
- Support for a preemptive execution model
  - Bound the *memory blocking delay* (`SHDi`)
  - Response-time analysis for each approach
- Future work
  - Extend the task model to mixed-criticality systems