Cache-aware Schedulability Analysis of PREM Compliant Tasks

Syed Aftab Rashid, Muhammad Ali Awan, Pedro F. Souto, Konstantinos Bletsas, Eduardo Tovar

CISTER - Research Center in
Real-Time & Embedded Computing Systems

Presentation Contents

01
Introduction and Motivation

PREM Model, and
Problem definition

02
Cache-aware PREM Model

DRCB-only Approach
FDCB-DRCB Approach

03
Experimental Setup and Results

Experiments performed under different settings and parameter values

04
Conclusions and Future works

Conclusions and potential future research directions

Introduction: Multi-core Processors have become mainstream

Introduction: Multi-core Processors have become mainstream

Introduction: Multi-core Processors have become mainstream

- Several advantages over single-core/custom build hardware.
  - ✓Improved performance
  - ✓Instant availability
  - ✓Low cost
  - ✓Low integration complexity, etc.
- Design mantra of "average case faster" makes MCPs a popular choice in low-criticality/soft real-time systems.

Introduction: Multi-core Processors have become mainstream

- Several advantages over single-core/custom build hardware.
  - ✓Improved performance
  - ✓Instant availability
  - ✓Low cost
  - ✓Low integration complexity, etc.
- Design mantra of "average case faster" makes MCPs a popular choice in low-criticality/soft real-time systems.
- The performance-oriented design of MCPs makes them non-deterministic, e.g., due to sharing of hardware resources such as caches, bus and the main memory.

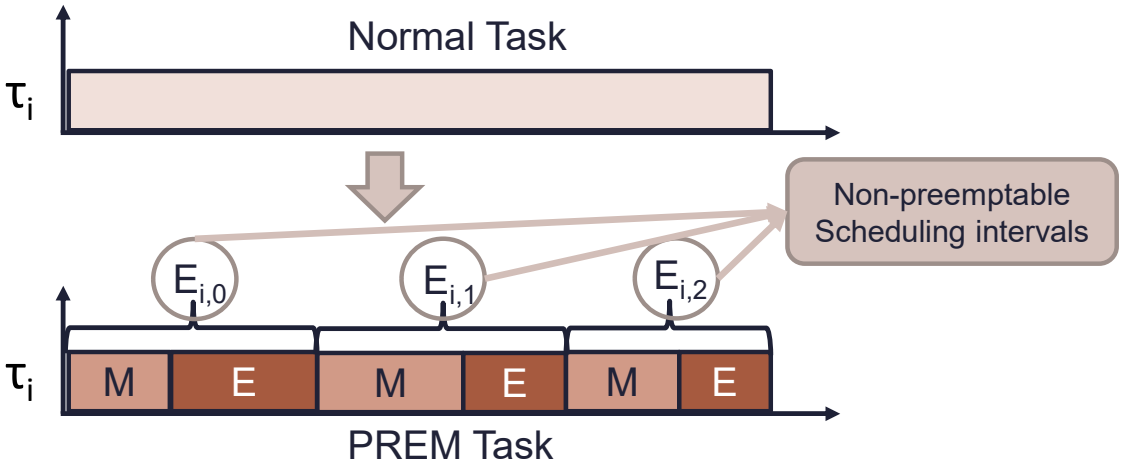
Introduction: Multi-core Processors have become mainstream

- Several advantages over single-core/custom build hardware.
  - ✓Improved performance
  - ✓Instant availability
  - ✓Low cost
  - ✓Low integration complexity, etc.
- Design mantra of "average case faster" makes MCPs a popular choice in low-criticality/soft real-time systems.
- The performance-oriented design of MCPs makes them non-deterministic, e.g., due to sharing of hardware resources such as caches, bus and the main memory.

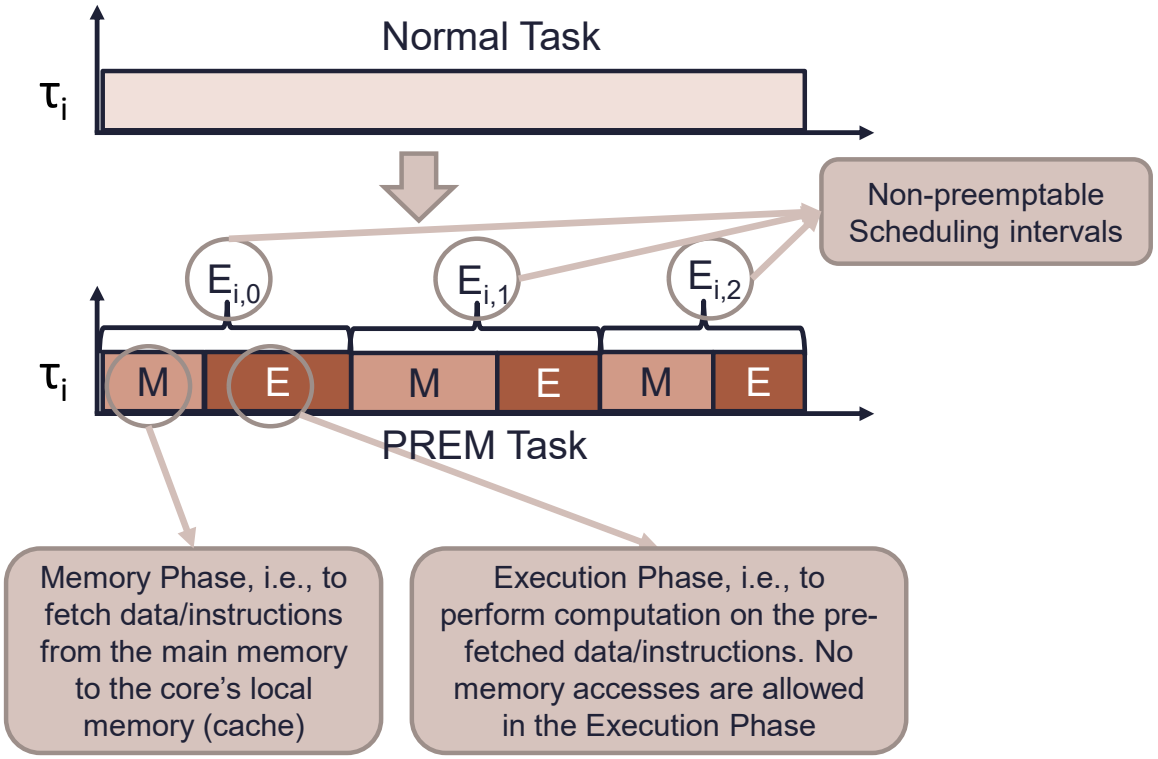
Contention due to shared resources, e.g., main memory, between tasks can have a significant impact on WCET/WCRT of tasks

Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem

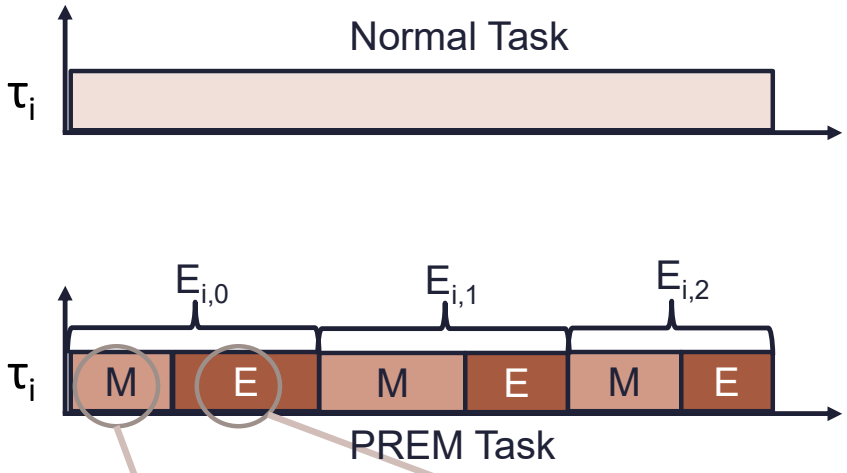
Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem

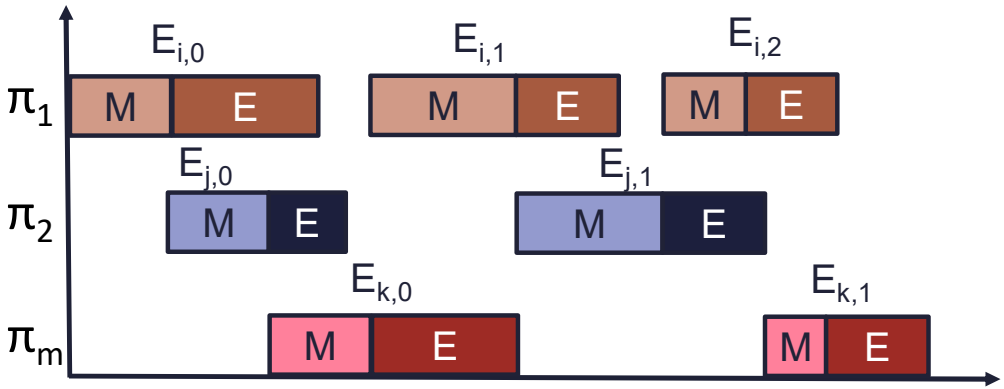


Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem



Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem

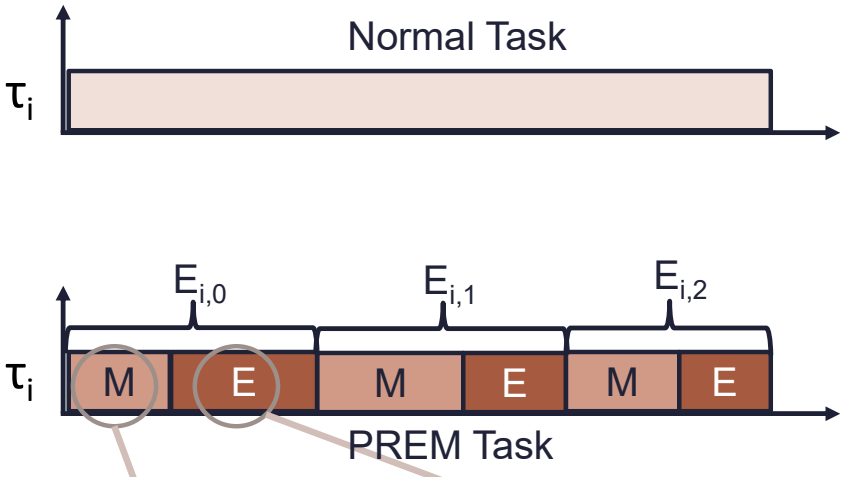




Memory Phase, i.e., to fetch data/instructions from the main memory to the core's local memory (cache)

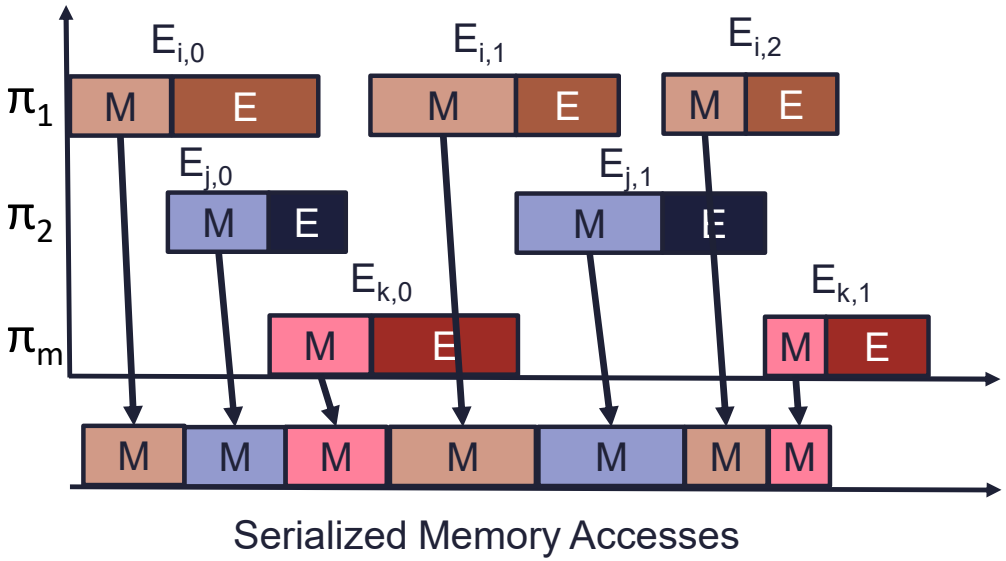
Execution Phase, i.e., to perform computation on the pre-fetched data/instructions. No memory accesses are allowed in the Execution Phase

Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem



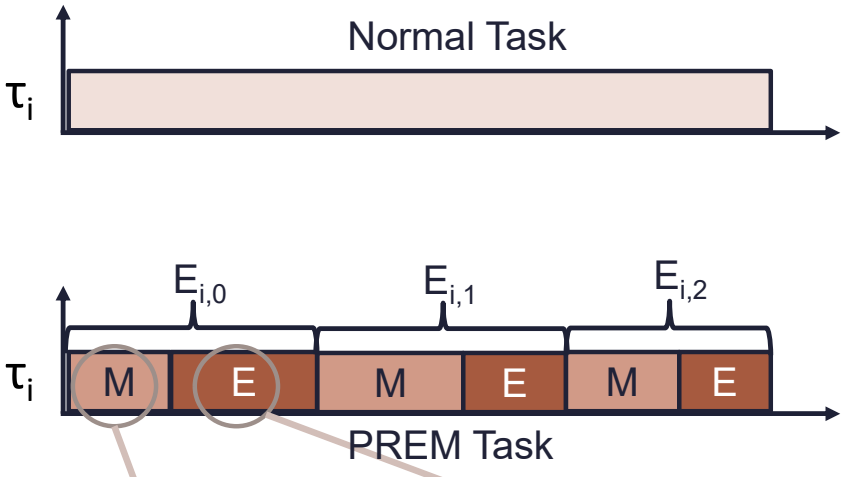
Memory Phase, i.e., to fetch data/instructions from the main memory to the core's local memory (cache)

Execution Phase, i.e., to perform computation on the pre-fetched data/instructions. No memory accesses are allowed in the Execution Phase



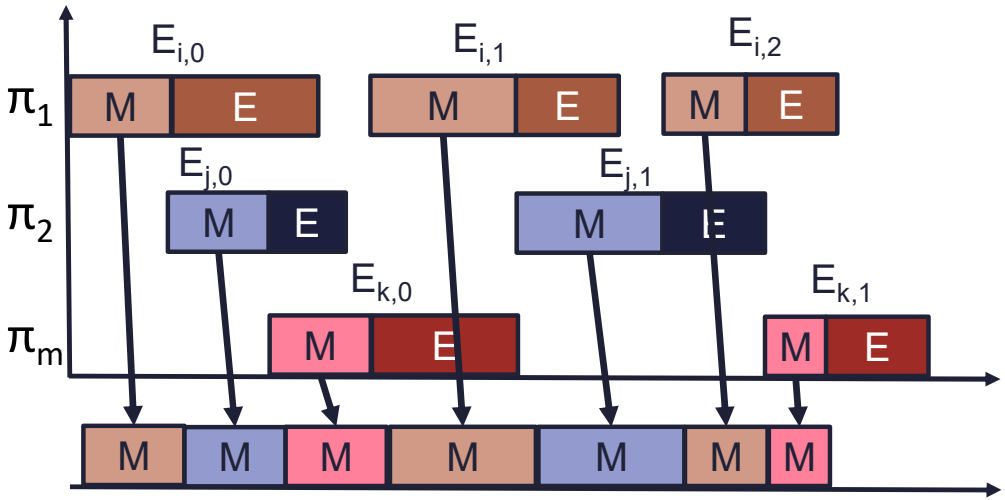
Serialized Memory Accesses

Introduction: The PRedictable Execution Model (PREM), a solution to memory contention problem



Memory Phase, i.e., to fetch data/instructions from the main memory to the core's local memory (cache)

Execution Phase, i.e., to perform computation on the pre-fetched data/instructions. No memory accesses are allowed in the Execution Phase

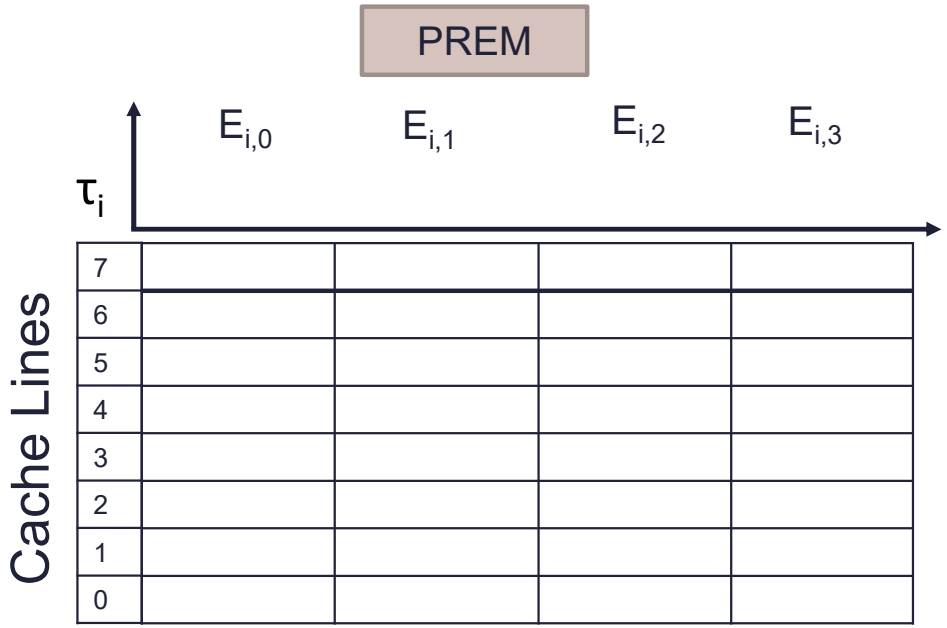


Serialized Memory Accesses

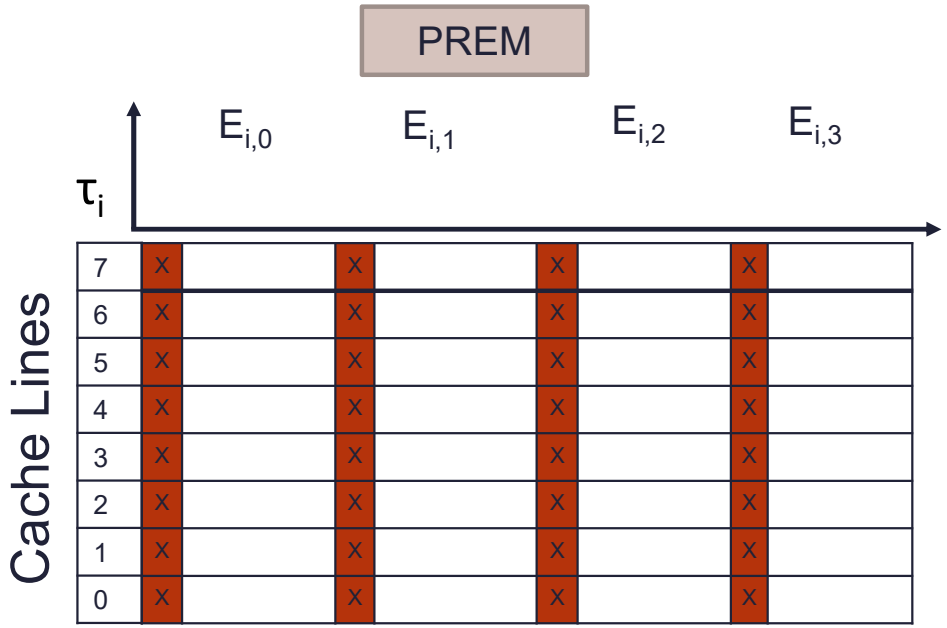
PREM can significantly reduce/eliminate contention due to shared main memory

Motivation: Existing PREM based schedulability analyses are cache agnostic

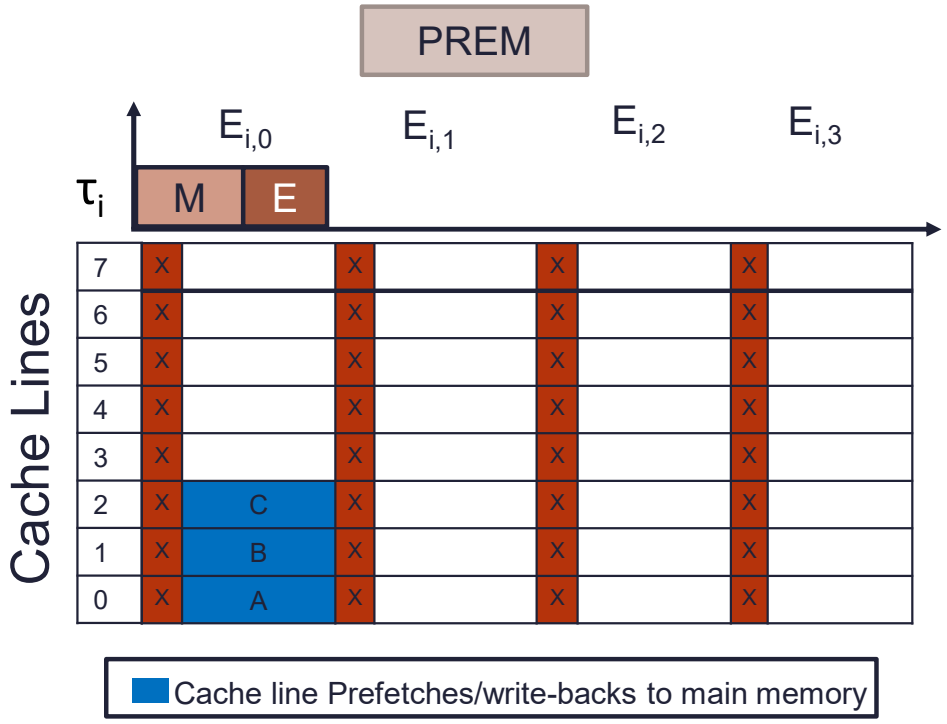
Motivation: Existing PREM based schedulability analyses are cache agnostic



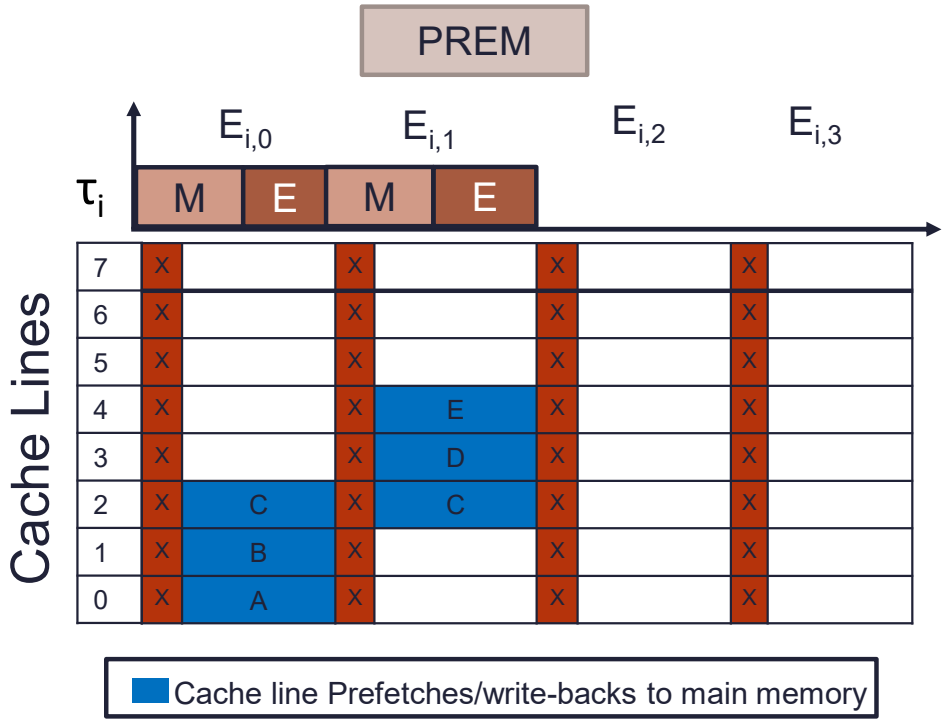
Motivation: Existing PREM based schedulability analyses are cache agnostic



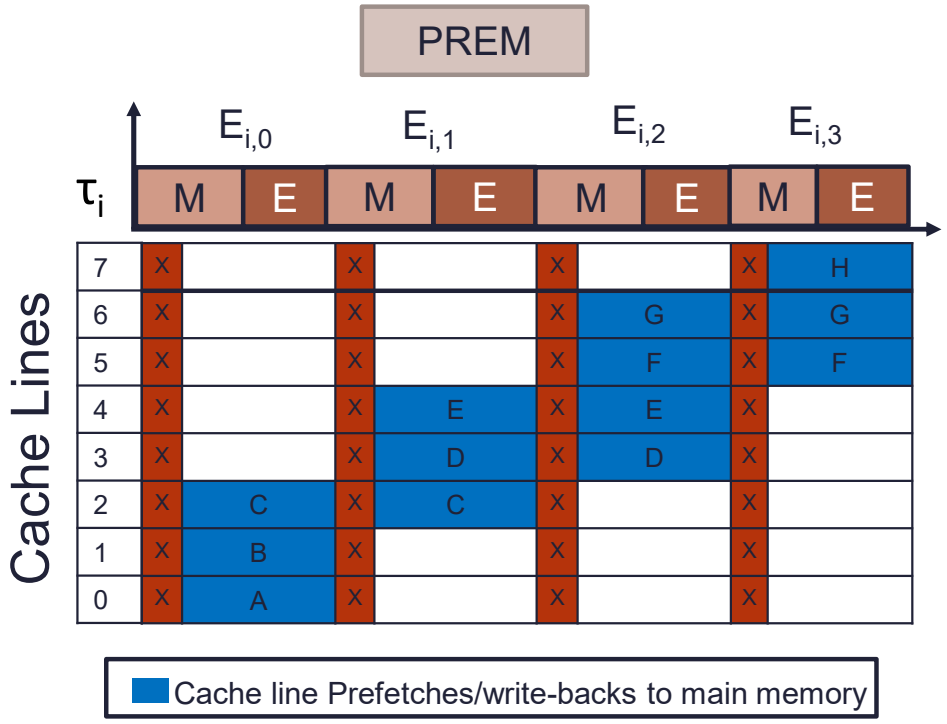
Motivation: Existing PREM based schedulability analyses are cache agnostic



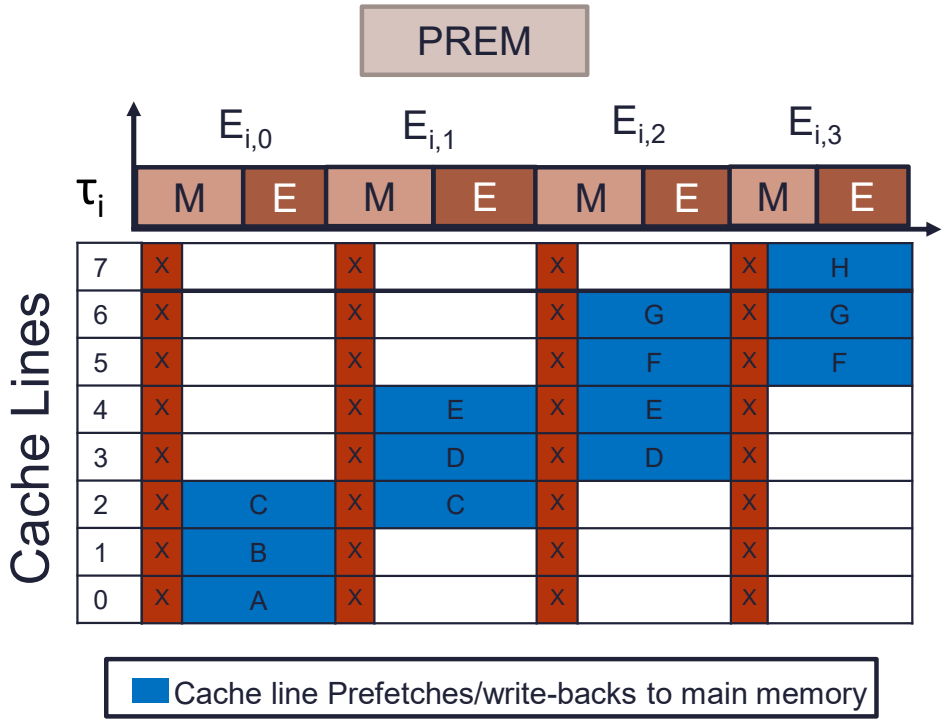
Motivation: Existing PREM based schedulability analyses are cache agnostic



Motivation: Existing PREM based schedulability analyses are cache agnostic

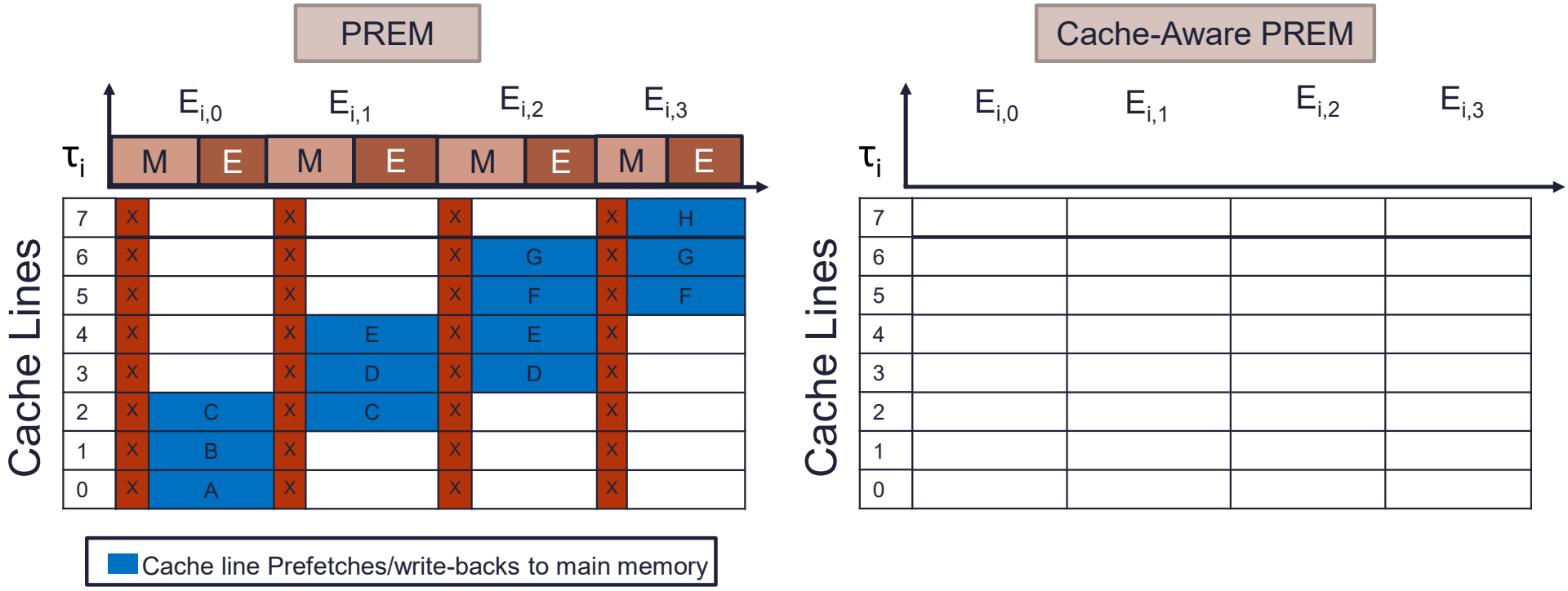


Motivation: Existing PREM based schedulability analyses are cache agnostic



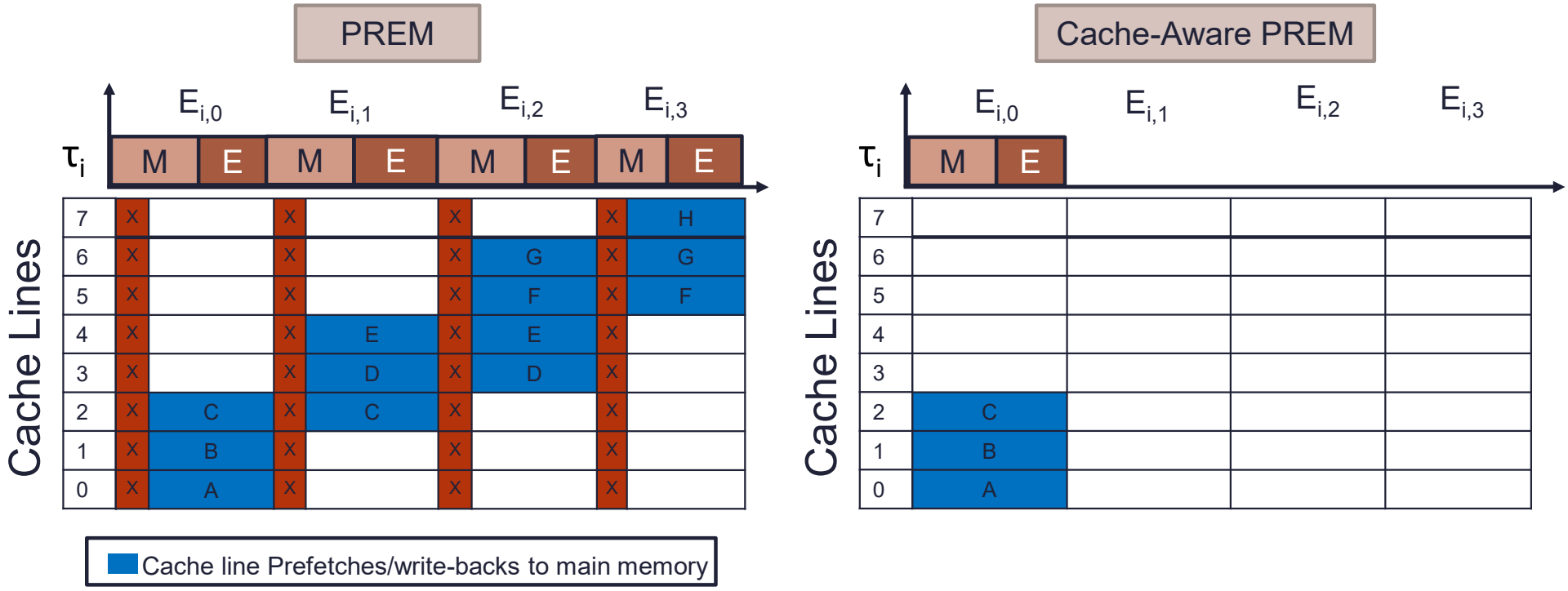
Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26

Motivation: Existing PREM based schedulability analyses are cache agnostic



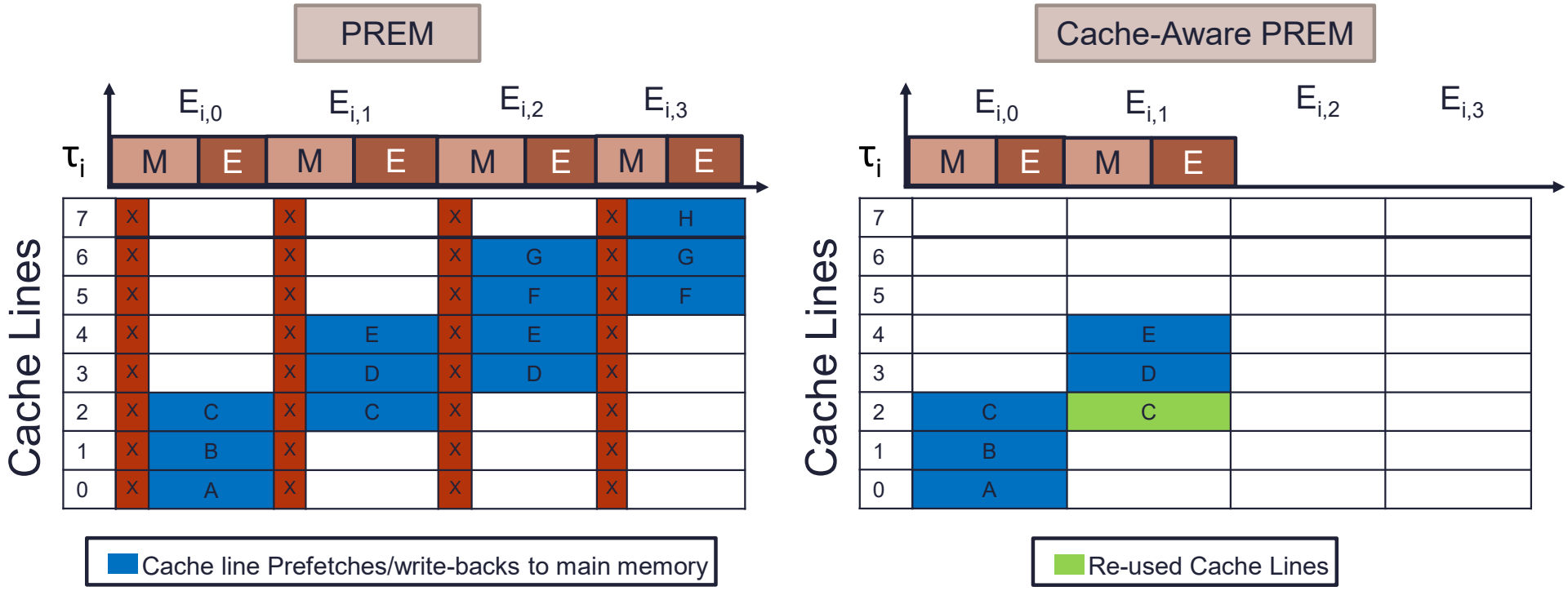
Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26

Motivation: Existing PREM based schedulability analyses are cache agnostic



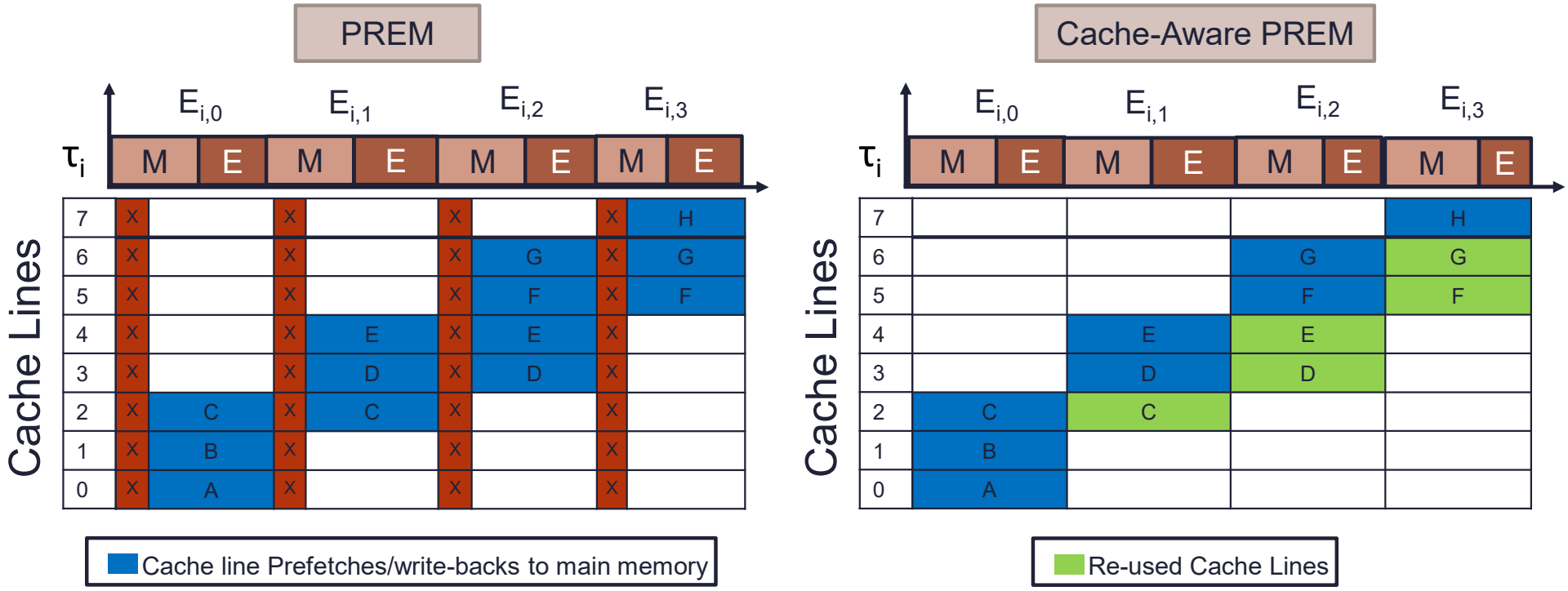
Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26

Motivation: Existing PREM based schedulability analyses are cache agnostic



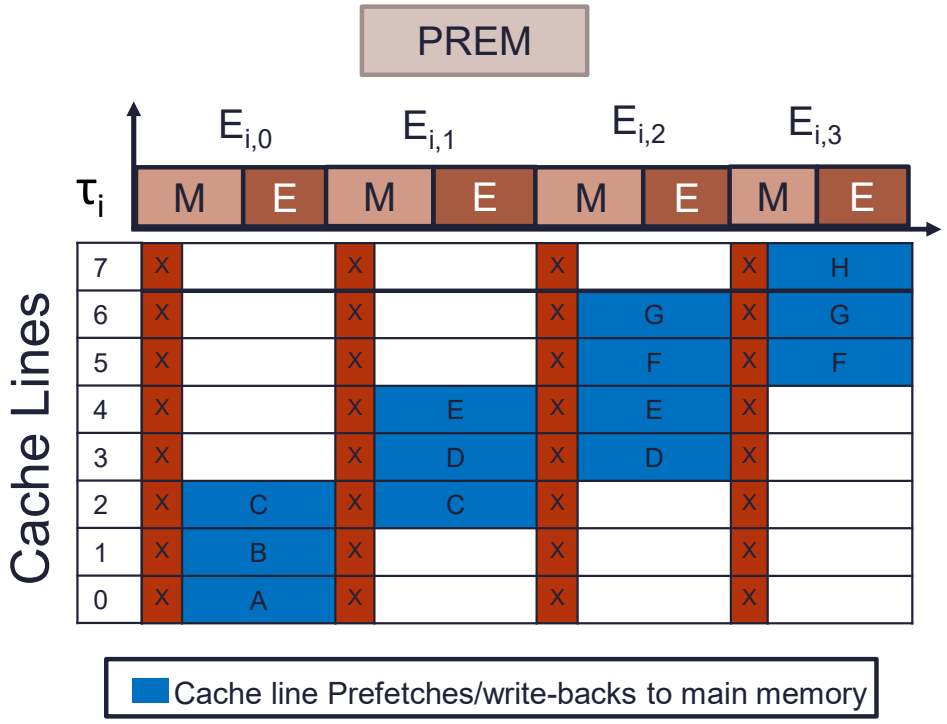
Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26

Motivation: Existing PREM based schedulability analyses are cache agnostic

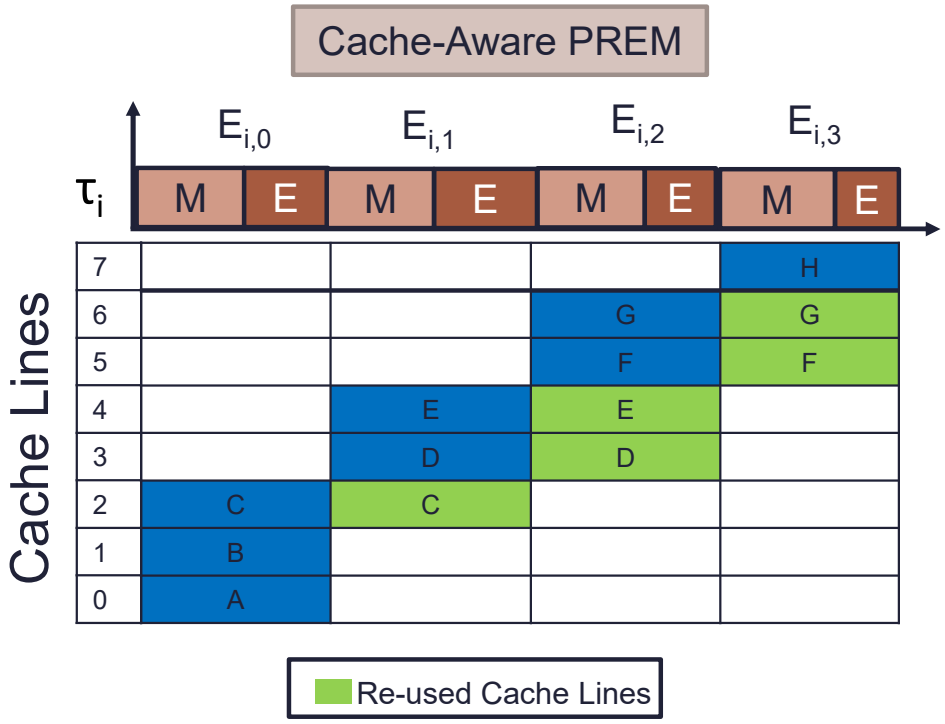


Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26

Motivation: Existing PREM based schedulability analyses are cache agnostic



Total Memory Accesses = Total Prefetches + Total Write-backs
= 13 + 13 = 26



Total Memory Accesses = Total Prefetches + Total Write-backs
= 8 + 8 = 16

Contribution: Cache-aware schedulability analysis of PREM tasks

- Preciser estimate of cache line prefetches -> DRCB-only Approach
- Tighter bound on cache line write-backs -> FDCB-DRCB Approach

Cache-Aware PREM: DRCB-only Approach

- What are Definitely Reused Cache Blocks (DRCBs)?

Cache-Aware PREM: DRCB-only Approach

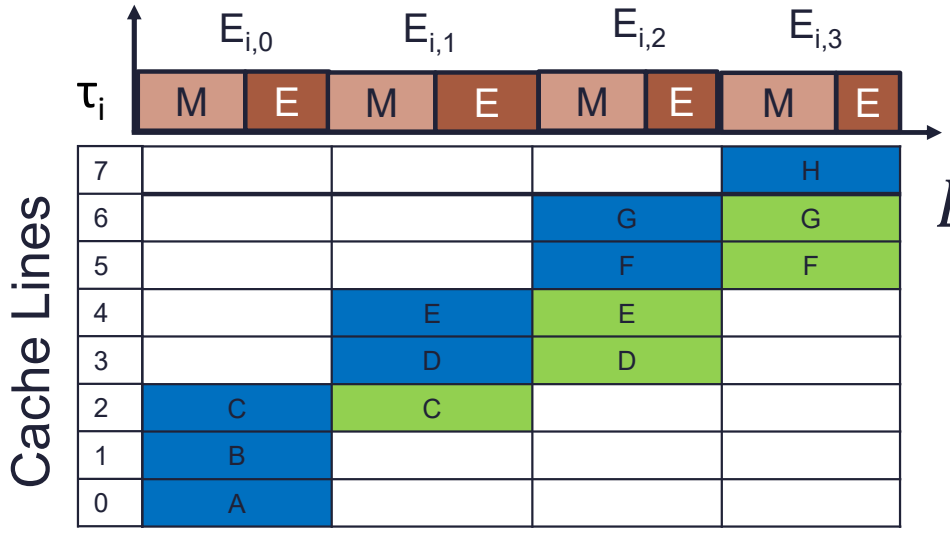
- What are Definitely Reused Cache Blocks (DRCBs)?

  Memory blocks that are cached at the end point E of a scheduling interval E_{i,j-1} and are reused at some program point F in scheduling interval E_{i,j}, without being evicted on any possible execution path between E to F.

Cache-Aware PREM: DRCB-only Approach

- What are Definitely Reused Cache Blocks (DRCBs)?

  Memory blocks that are cached at the end point E of a scheduling interval E_{i,j-1} and are reused at some program point F in scheduling interval E_{i,j}, without being evicted on any possible execution path between E to F.



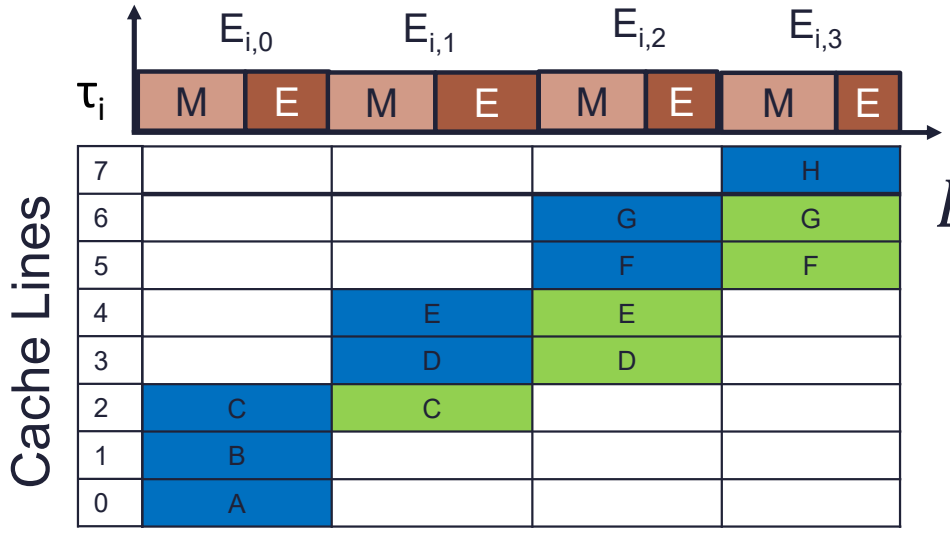
$$DRCB_{i,0} = \{\}$$

$$DRCB_{i,1} = \{C\}, DRCB_{i,2} = \{D, E\}$$

Cache-Aware PREM: DRCB-only Approach

- What are Definitely Reused Cache Blocks (DRCBs)?

  Memory blocks that are cached at the end point E of a scheduling interval E_{i,j-1} and are reused at some program point F in scheduling interval E_{i,j}, without being evicted on any possible execution path between E to F.



$$DRCB_{i,0} = \{\}$$

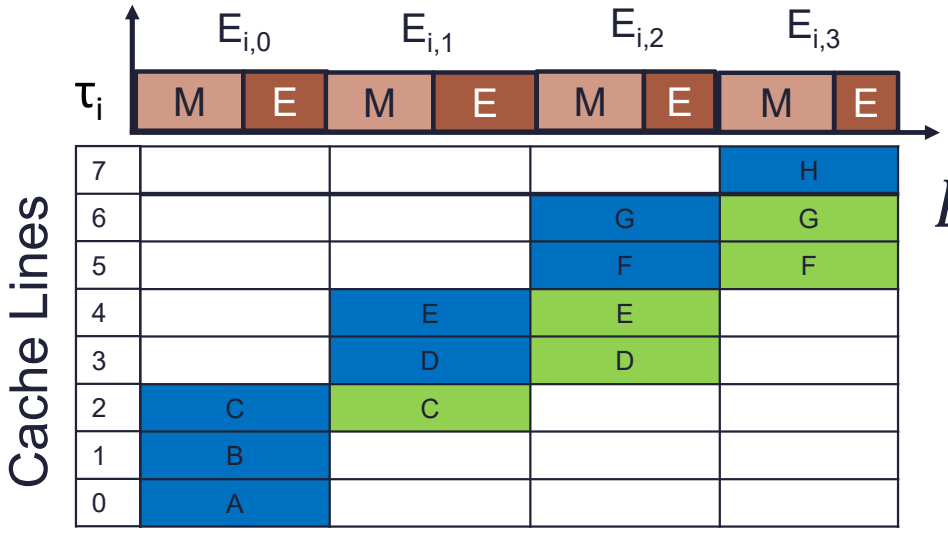
$$DRCB_{i,1} = \{C\}, DRCB_{i,2} = \{D, E\}$$

$$DRCB_{i,3} = \{F, G\}$$

Cache-Aware PREM: DRCB-only Approach

- What are Definitely Reused Cache Blocks (DRCBs)?

  Memory blocks that are cached at the end point E of a scheduling interval E_{i,j-1} and are reused at some program point F in scheduling interval E_{i,j}, without being evicted on any possible execution path between E to F.



$$DRCB_{i,0} = \{\}$$

$$DRCB_{i,1} = \{C\}, DRCB_{i,2} = \{D, E\}$$

$$DRCB_{i,3} = \{F, G\}$$

$$DRCB_i = \{C, D, E, F, G\}$$

Cache-Aware PREM: DRCB-only Approach

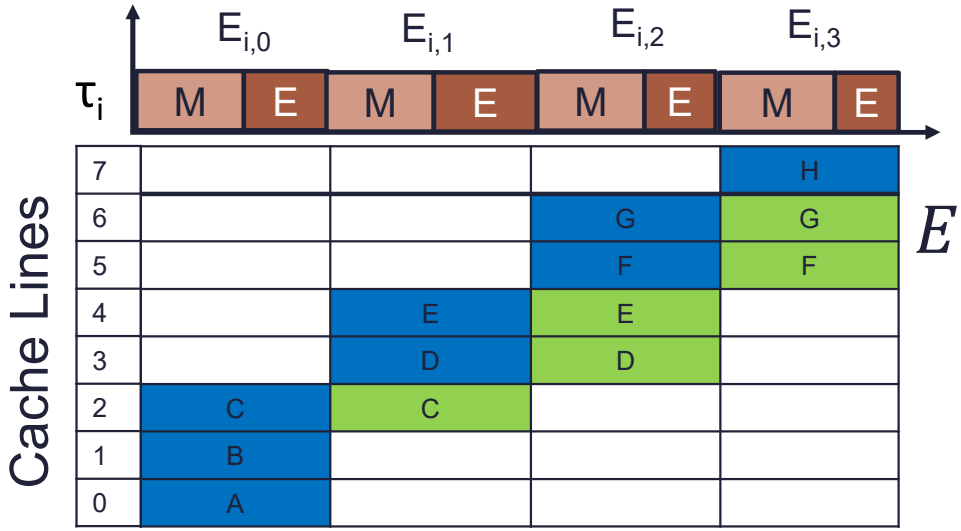
- What are Evicting Cache Blocks (ECBs)?

  All Memory blocks used by a task (or scheduling interval) during its execution

Cache-Aware PREM: DRCB-only Approach

- What are Evicting Cache Blocks (ECBs)?

  All Memory blocks used by a task (or scheduling interval) during its execution

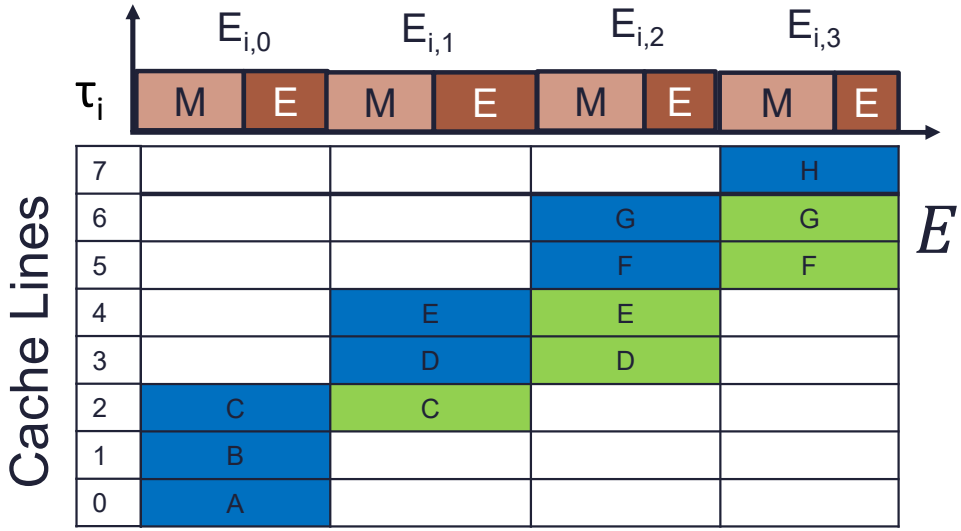


$$ECB_{i,0} = \{A, B, C\}, ECB_{i,1} = \{C, D, E\}$$

Cache-Aware PREM: DRCB-only Approach

- What are Evicting Cache Blocks (ECBs)?

  All Memory blocks used by a task (or scheduling interval) during its execution



$$ECB_{i,0} = \{A, B, C\}, ECB_{i,1} = \{C, D, E\}$$

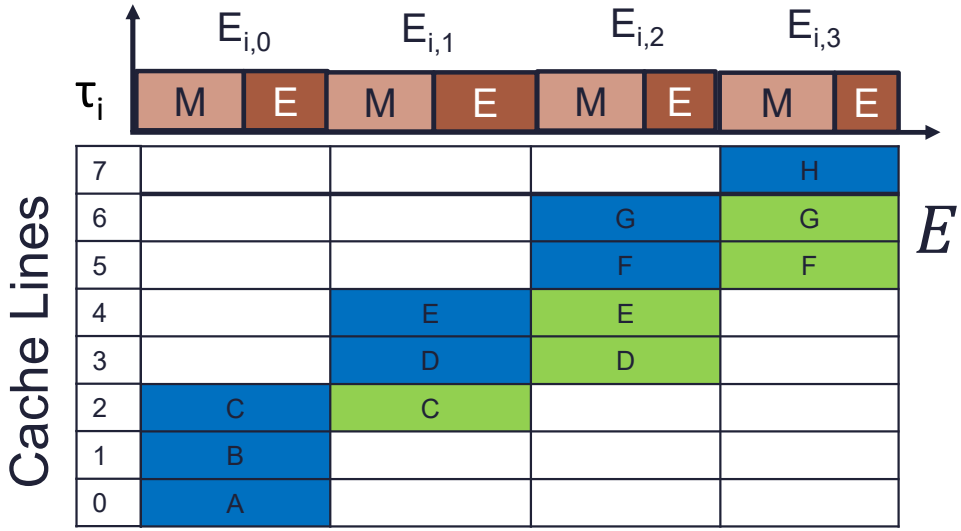
$$ECB_{i,2} = \{D, E, F, G\}$$

$$ECB_{i,3} = \{F, G, H\}$$

Cache-Aware PREM: DRCB-only Approach

- What are Evicting Cache Blocks (ECBs)?

  All Memory blocks used by a task (or scheduling interval) during its execution



$$ECB_{i,0} = \{A, B, C\}, ECB_{i,1} = \{C, D, E\}$$

$$ECB_{i,2} = \{D, E, F, G\}$$

$$ECB_{i,3} = \{F, G, H\}$$

$$ECB_i = \{A, B, C, D, E, F, G, H\}$$

Cache-Aware PREM: DRCB-only Approach

- How the DRCB approach work?

  A scheduling interval E_{i,j} only needs to prefetch ECBs that are not its DRCBs

Cache-Aware PREM: DRCB-only Approach

- How the DRCB approach work?

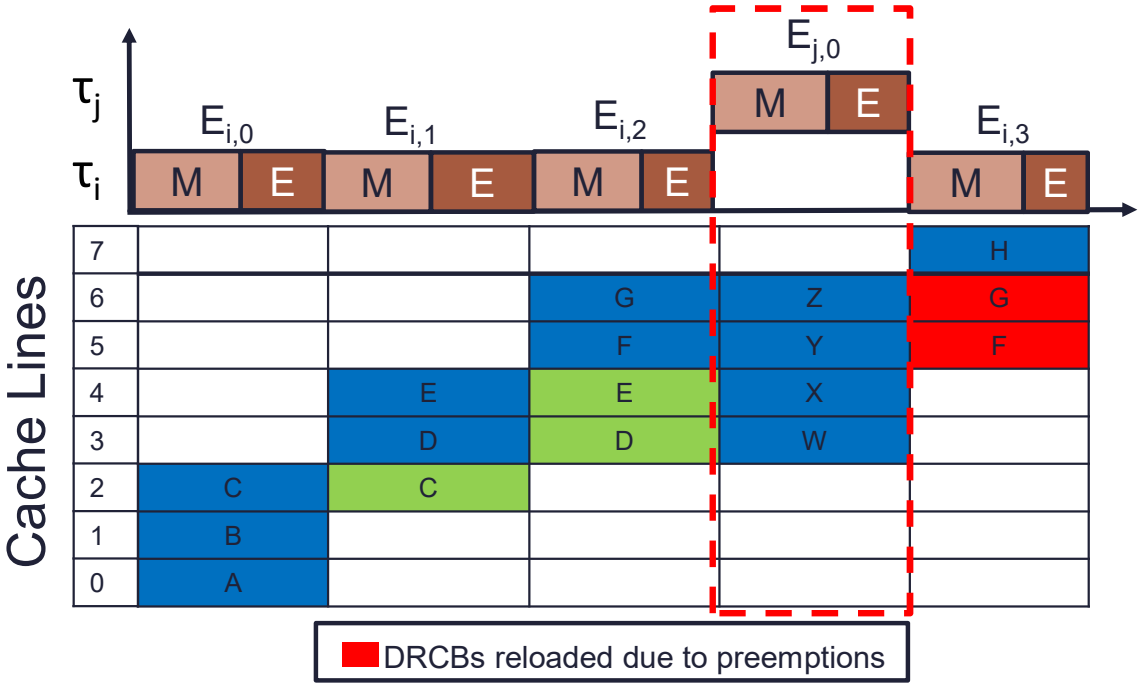
  A scheduling interval E_{i,j} only needs to prefetch ECBs that are not its DRCBs

$$Memory\ blocks\ prefeteched\ by\ E_{i,j} = ECB_{i,j}/DRCB_{i,j}$$

Cache-Aware PREM: DRCB-only Approach

- How the DRCB approach work?

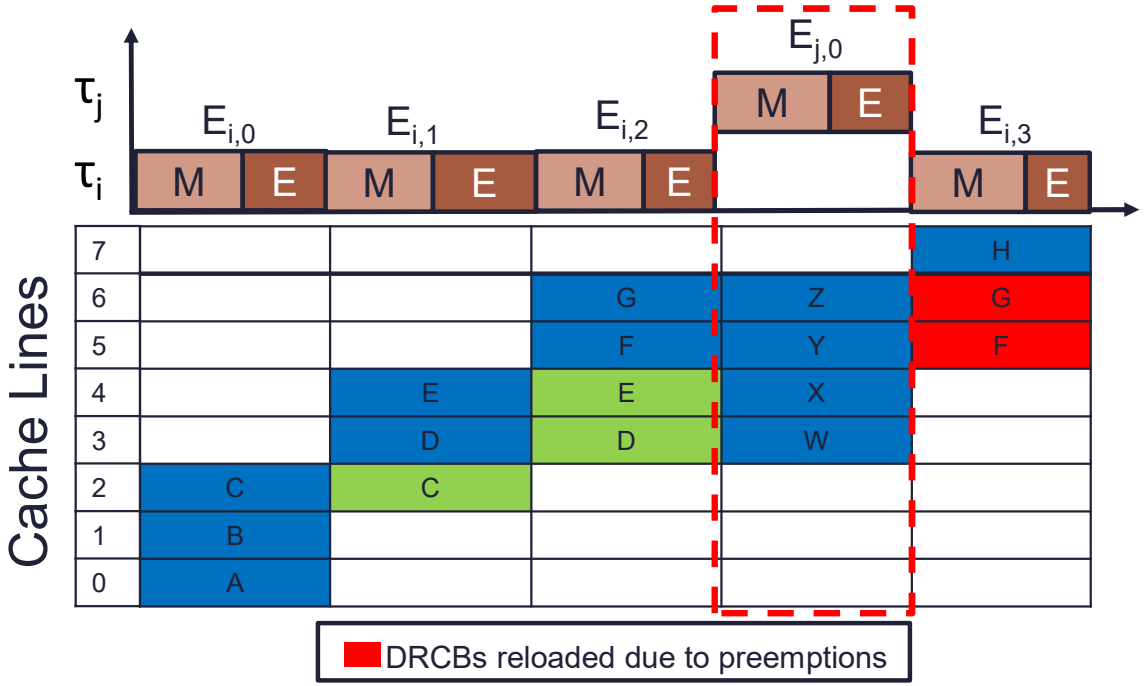
  A scheduling interval E_{i,j} only needs to prefetch ECBs that are not its DRCBs



Cache-Aware PREM: DRCB-only Approach

- How the DRCB approach work?

A scheduling interval E_{i,j} only needs to prefetch ECBs that are not its DRCBs

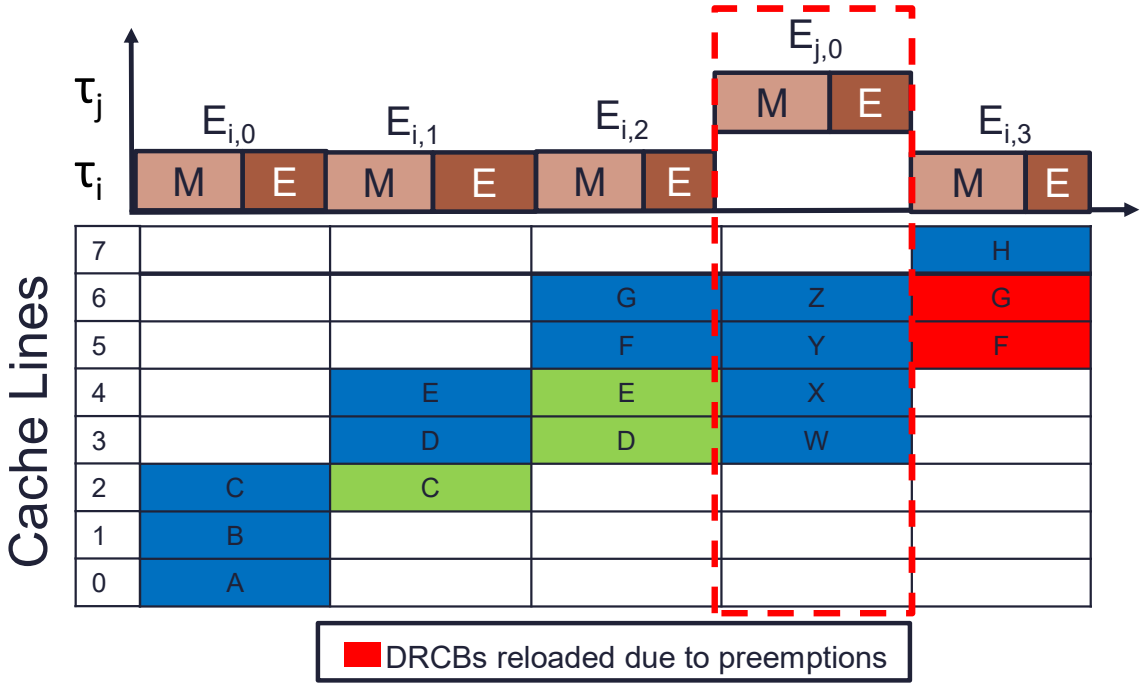


$$\underbrace{DRCB^{E}_{i,j}}_{\text{DRCB of E\_\{i,j\} evicted due to preemptions}} = DRCB_{i,j} \bigcap \{ \bigcup_{\forall k \in hp(i)} ECB_k \}$$

Cache-Aware PREM: DRCB-only Approach

- How the DRCB approach work?

A scheduling interval E_{i,j} only needs to prefetch ECBs that are not its DRCBs



$$\underbrace{DRCB^{E}_{i,j}}_{\text{DRCB of } E_{i,j} \text{ evicted due to preemptions}} = DRCB_{i,j} \bigcap \{ \bigcup_{\forall k \in hp(i)} ECB_k \}$$

$$\underbrace{ECB^{P}_{i,j}}_{\text{Prefetches during } E_{i,j} \text{ with cache reuse}} = \{ ECB_{i,j} \setminus DRCB_{i,j} \} \bigcup DRCB^{E}_{i,j}$$

$$\left| ECB^{P}_{i,j} \right| \leq |ECB_{i,j}|$$

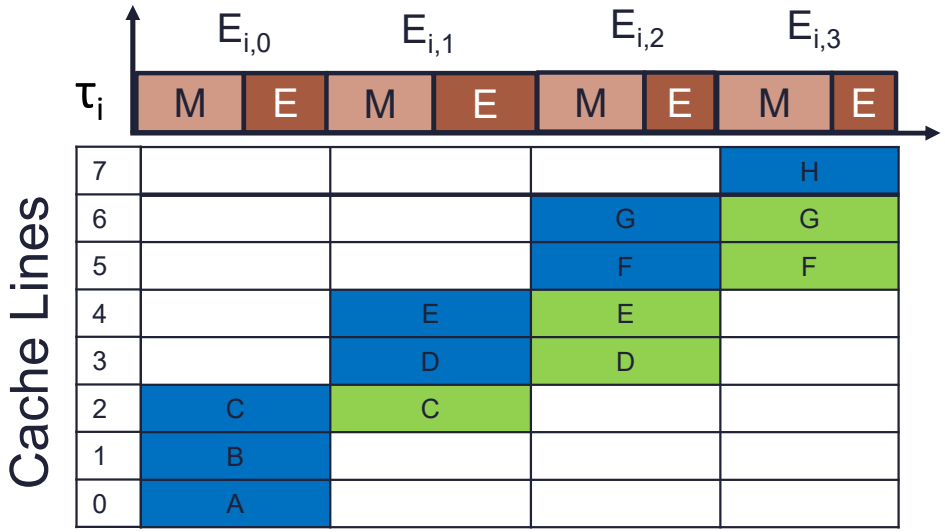
Cache-Aware PREM: FDCB-DRCB Approach

Cache-Aware PREM: FDCB-DRCB Approach

- DRCB-only approach overestimates the number of cache write-backs

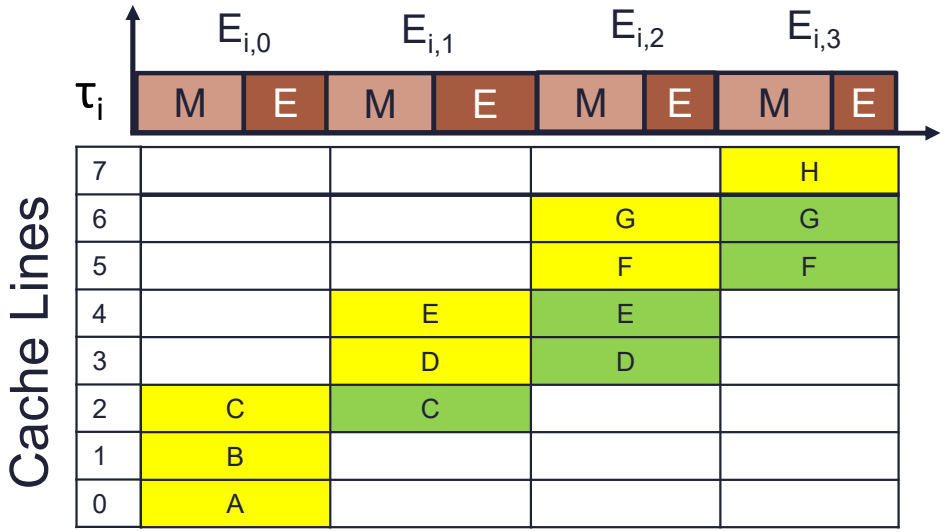
Cache-Aware PREM: FDCB-DRCB Approach

- DRCB-only approach overestimates the number of cache write-backs



Cache-Aware PREM: FDCB-DRCB Approach

- DRCB-only approach overestimates the number of cache write-backs



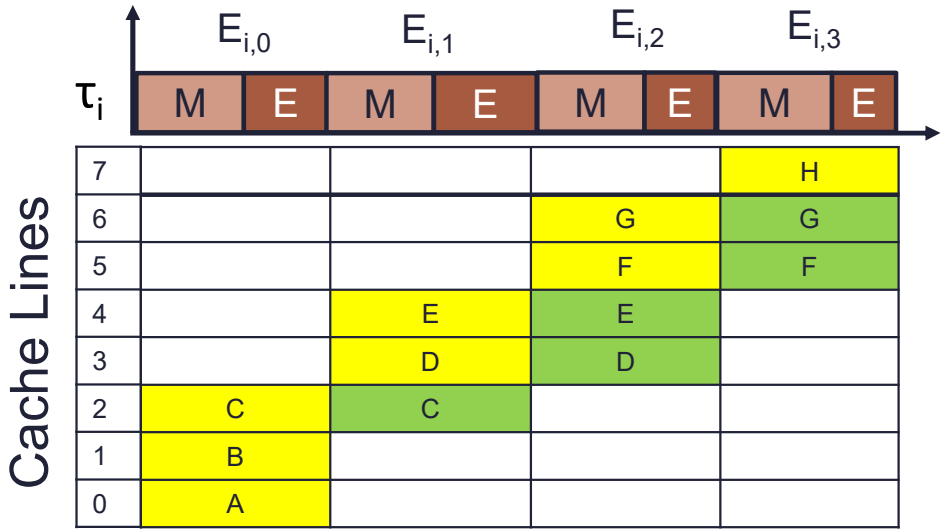
$$WB_{i,0} = \{A, B, C\}, WB_{i,1} = \{D, E\}$$

$$WB_{i,2} = \{F, G\}$$

$$WB_{i,3} = \{H\}$$

Cache-Aware PREM: FDCB-DRCB Approach

- DRCB-only approach overestimates the number of cache write-backs



$$WB_{i,0} = \{A, B, C\}, WB_{i,1} = \{D, E\}$$

$$WB_{i,2} = \{F, G\}$$

$$WB_{i,3} = \{H\}$$

Actual number of write-backs depend on the number of dirty cache lines

Cache-Aware PREM: FDCB-DRCB Approach

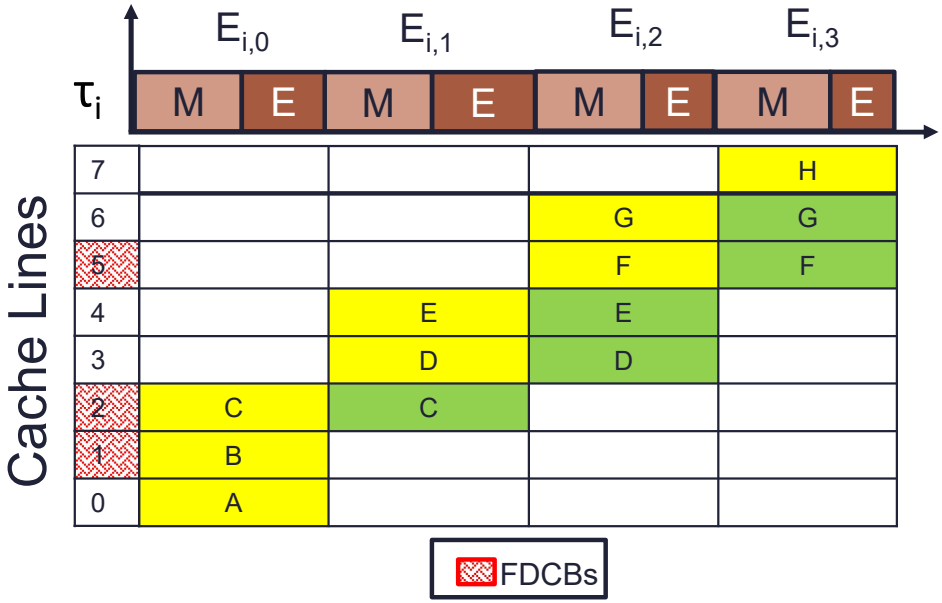
- What are Final Dirty Cache Blocks (FDCBs)?

  Memory blocks that may be written to during the execution of a task/scheduling interval and may still be available in cache after the completion of that task/scheduling interval.

Cache-Aware PREM: FDCB-DRCB Approach

- What are Final Dirty Cache Blocks (FDCBs)?

  Memory blocks that may be written to during the execution of a task/scheduling interval and may still be available in cache after the completion of that task/scheduling interval.



$$WB_{i,0} = \{A, B, C\}, WB_{i,1} = \{D, E\}$$

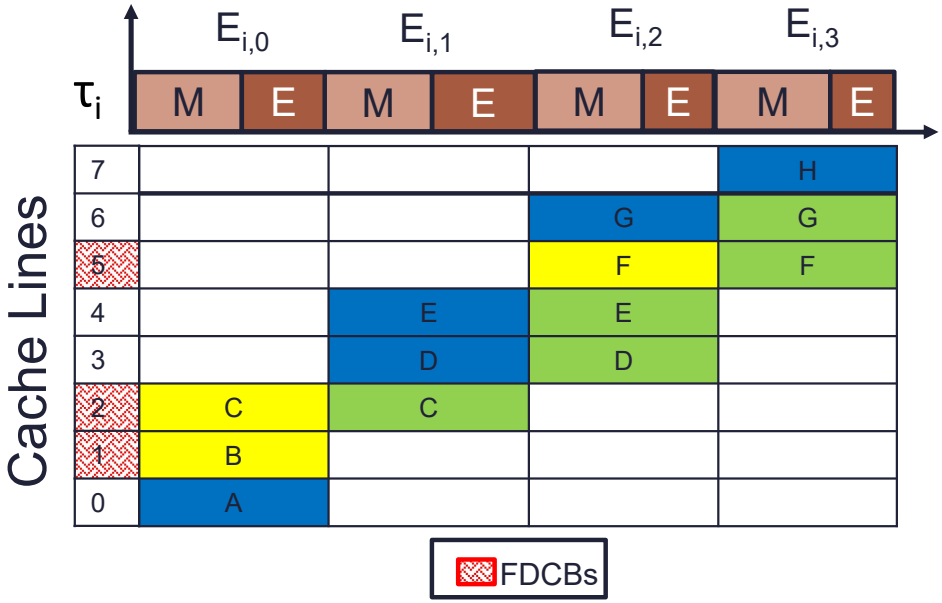
$$WB_{i,2} = \{F, G\}$$

$$WB_{i,3} = \{H\}$$

Cache-Aware PREM: FDCB-DRCB Approach

- What are Final Dirty Cache Blocks (FDCBs)?

  Memory blocks that may be written to during the execution of a task/scheduling interval and may still be available in cache after the completion of that task/scheduling interval.



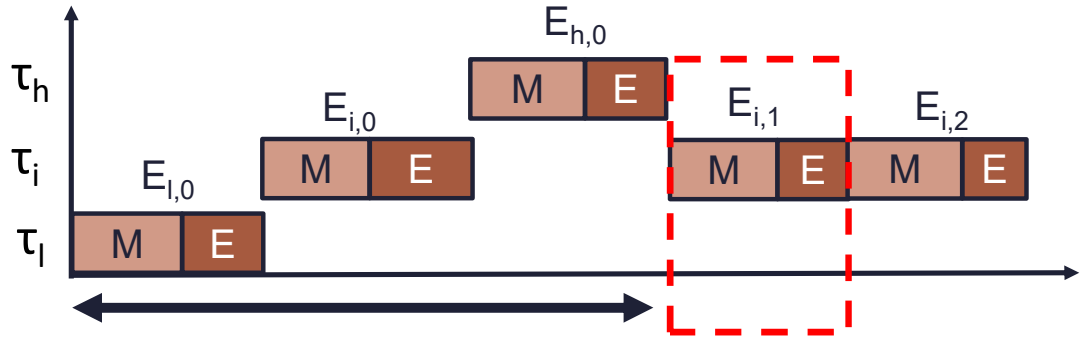
$$WB_{i,0} = \{A, B\}, WB_{i,1} = \{\}$$

$$WB_{i,2} = \{F\}$$

$$WB_{i,3} = \{\}$$

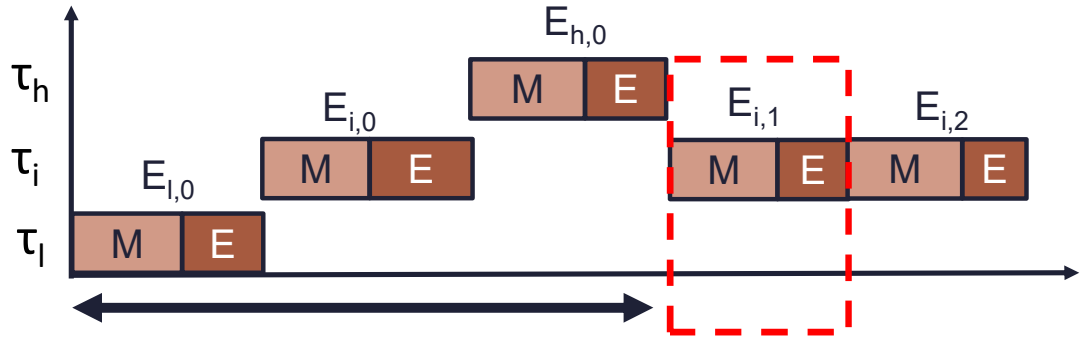
Cache-Aware PREM: FDCB-DRCB Approach

- Cache write-backs during the execution of a scheduling interval E_{i,j} depend on the ECBs of that scheduling interval and FDCBs of all tasks in hep(i) and lp(i)



Cache-Aware PREM: FDCB-DRCB Approach

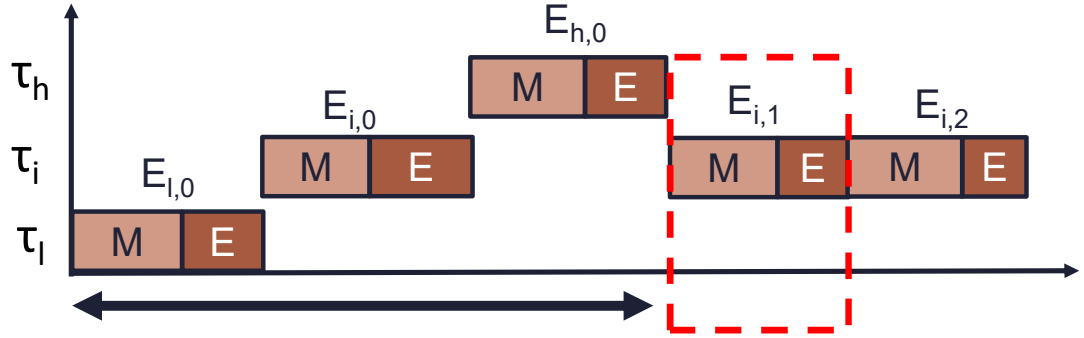
- Cache write-backs during the execution of a scheduling interval E_{i,j} depend on the ECBs of that scheduling interval and FDCBs of all tasks in hep(i) and lp(i)



$$\underbrace{WB_{i,j}^{lp}}_{\text{Write-backs due to tasks in lp(i)}} = \left(\bigcup_{\forall l \in \mathrm{lp}(i)} FDCB_l \setminus \bigcup_{\forall k<j} ECB_{i,k} \right) \bigcap ECB_{i,j}$$

Cache-Aware PREM: FDCB-DRCB Approach

- Cache write-backs during the execution of a scheduling interval E_{i,j} depend on the ECBs of that scheduling interval and FDCBs of all tasks in hep(i) and lp(i)



$$\underbrace{WB^{lp}_{i,j}} = \left(\bigcup_{\forall l \in \mathrm{lp}(i)} FDCB_l \setminus \bigcup_{\forall k<j} ECB_{i,k} \right) \bigcap ECB_{i,j} \qquad \underbrace{WB^{hep}_{i,j}} = \left(\left(\bigcup_{\forall h \in hep(i)} FDCB_h \right) \bigcap ECB^R_{i,j} \right) \bigcup DRCB^E_{i,j}$$

Write-backs due to tasks in lp(i)

Write-backs due to tasks in hep(i)

$$WB^{tot}_{i,j} = WB^{lp}_{i,j} \bigcup WB^{hep}_{i,j}$$

Cache-Aware PREM: WCRT Analysis

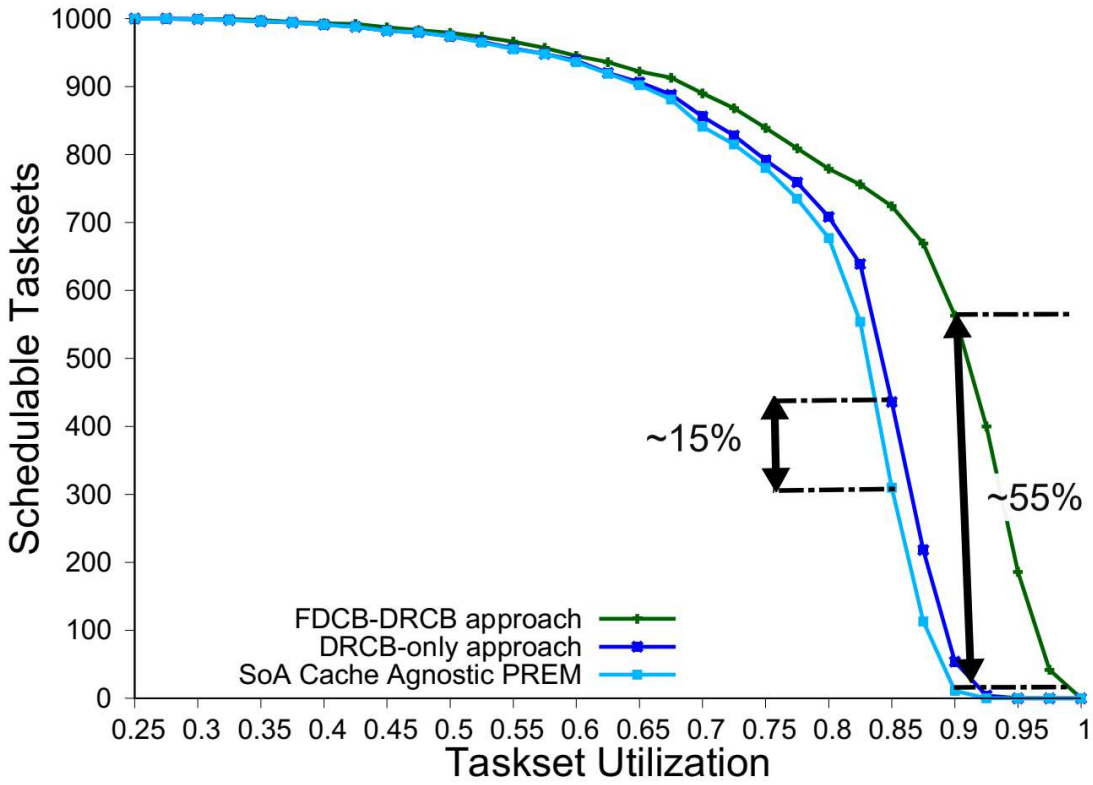
$$\underbrace{C_{i,j}}_{\text{WCET of E\_\{i,j\}}} = (\underbrace{|WB^{tot}_{i,j}|}_{\text{Total write-backs during the execution of E\_\{i,j\}}} + \underbrace{|ECB^{P}_{i,j}|}_{\text{Total prefetches during the execution of E\_\{i,j\}}}) \times \underbrace{d_{mem}}_{\text{Worst-case time to load one block from main memory}} + \underbrace{C^{e}_{i,j}}_{\text{Length of Execution Phase of E\_\{i,j\}}}$$

$$\underbrace{C_i}_{\text{WCET of } \tau_i} = \sum_{j=0}^{N_i-1} C_{i,j} \qquad \underbrace{R_i^{k+1}}_{\text{WCRT of } \tau_i} = B_i + C_i + \sum_{\forall h \in hp(i)} \left\lceil \frac{R_i^k}{T_h} \right\rceil C_h$$

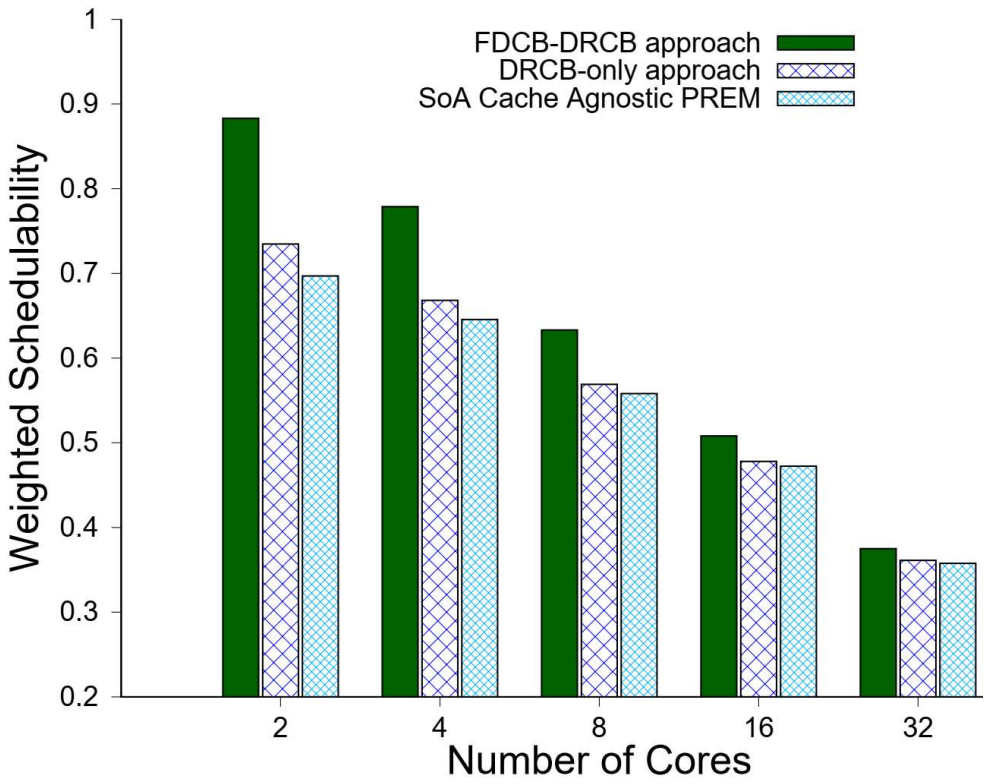
Experimental Evaluation: Experimental Setup and Taskset generation

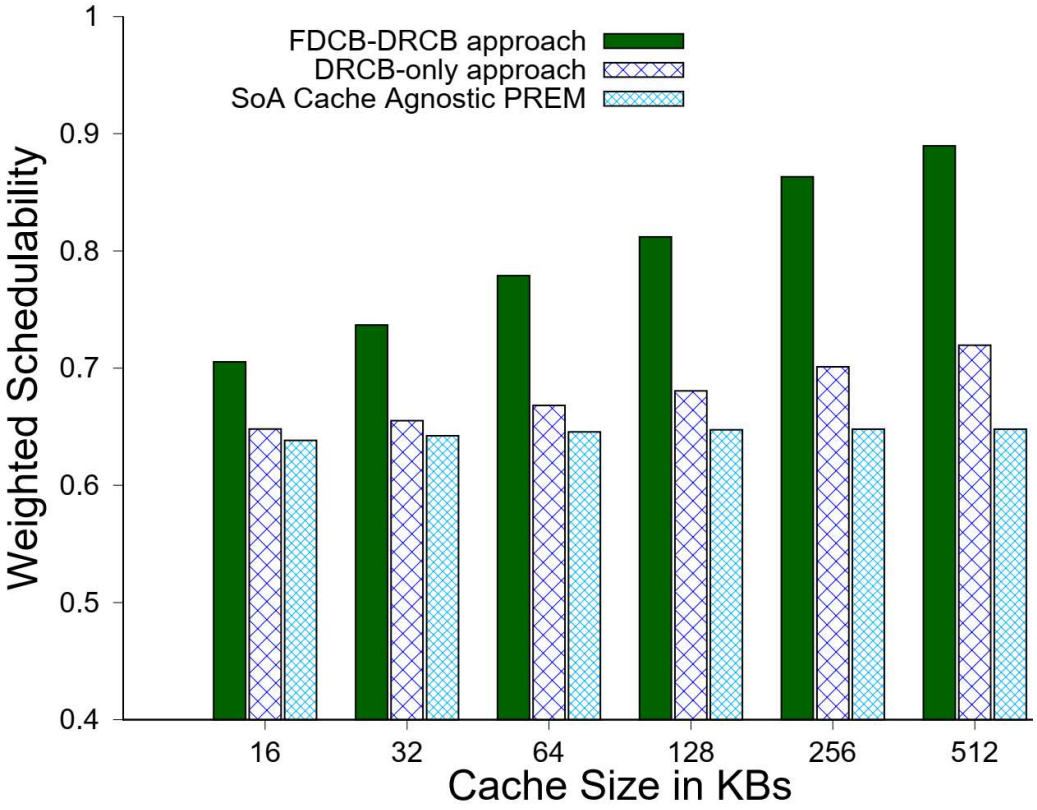
- We modeled a multicore platform with cores m=4, Last-level Cache = 64KB (2048 Cache sets, 32-byte blocks), Cache Prefetch/Write-back Penalty = 100 µSec
- Taskset size = 32, i.e., 8 tasks per core, Taskset Utilizations = UUnifast, Task periods = [5ms, 500ms], WCET = Utilization x Period, …
- 1000 synthetic tasksets per experimental point, compared SoA cache-agnostic PREM, DRCB-only approach and FDCB-DRCB approach.

Experimental Evaluation: Varying per Core Utilization

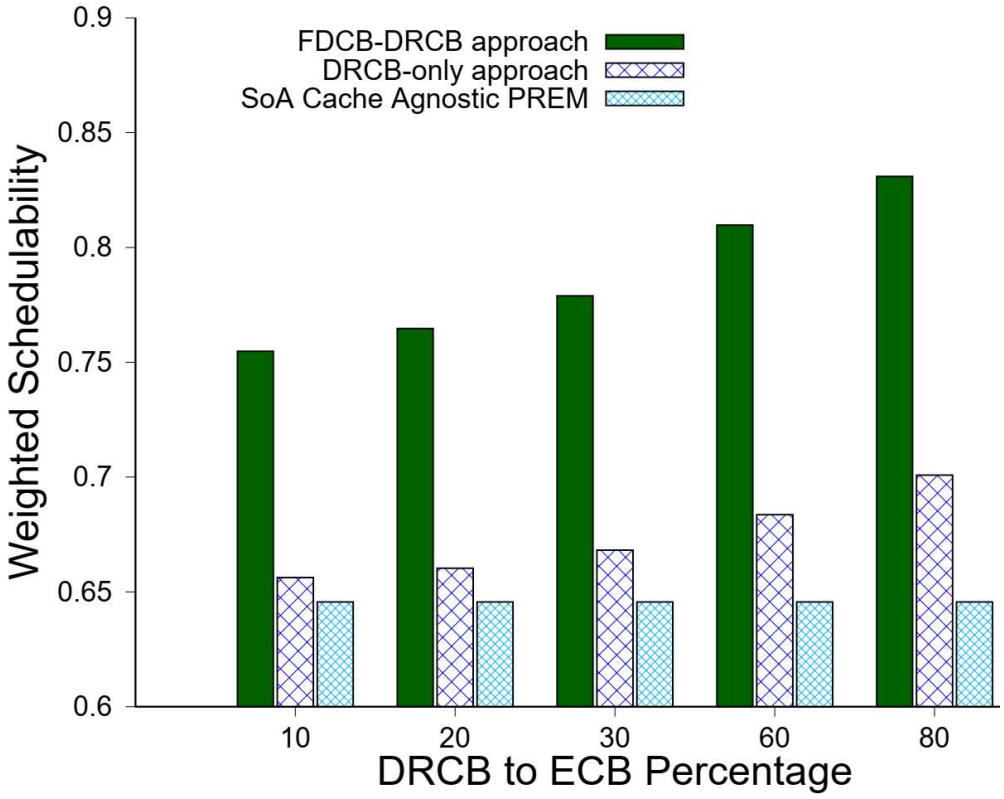


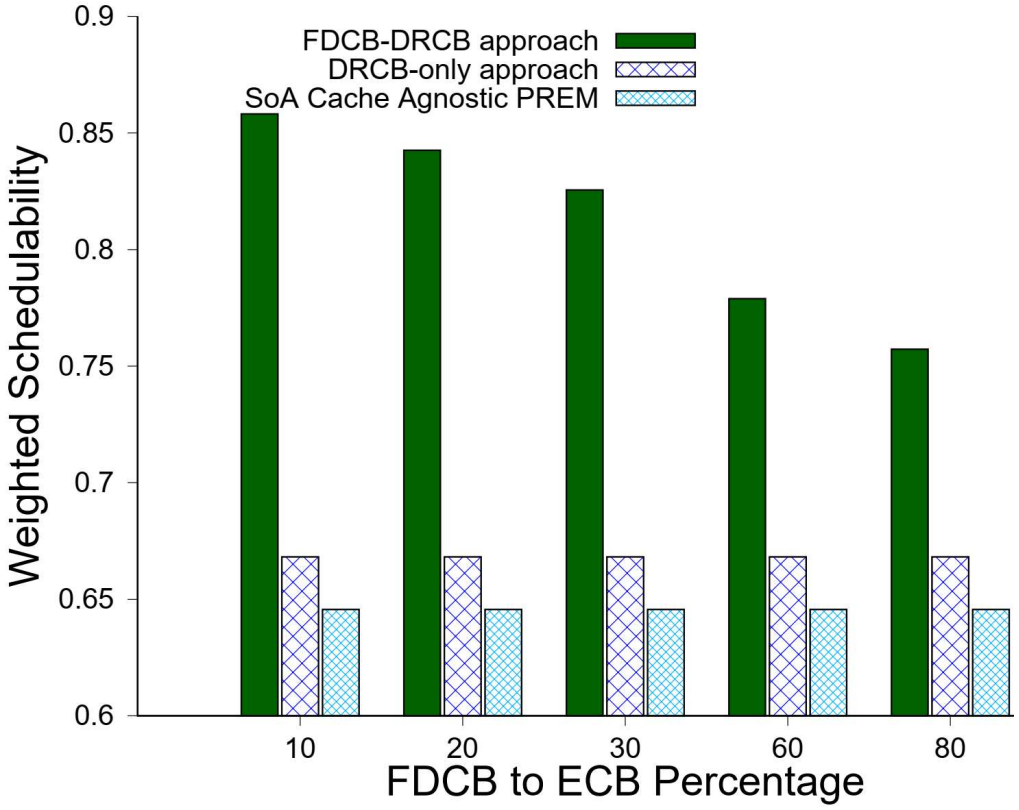
Experimental Evaluation: Varying number of Cores and Cache Size





Experimental Evaluation: Varying DRCB-ECB and FDCB-ECB percentage





Conclusion and Future Work

- Ported established cache-aware schedulability analysis to PREM
- Presented two approaches that tightly estimate cache line loads and write-backs
- DRCB-only approach exploits cache reuse among scheduling intervals to reduce the number of loads. The FDCB-DRCB approach improves on DRCB-only approach by carefully analyzing cache write-backs.
- Experimental evaluation shows that our approaches can achieve up to 55% better schedulability ratio.

- As future work, we aim to extend our analysis to multi-set approaches.
- We also plan to extend our analysis to consider inter-job cache reuse.

Conclusion and Future Work

- Ported established cache-aware schedulability analysis to PREM
- Presented two approaches that tightly estimate cache line loads and write-backs
- DRCB-only approach exploits cache reuse among scheduling intervals to reduce the number of loads. The FDCB-DRCB approach improves on DRCB-only approach by carefully analyzing cache write-backs.
- Experimental evaluation shows that our approaches can achieve up to 55% better schedulability ratio.

- As future work, we aim to extend our analysis to multi-set approaches.
- We also plan to extend our analysis to consider inter-job cache reuse.

Thank You ☺

syed.aftabrashid@vortex-colab.com
syara@isep.ipp.pt

CISTER - Research Center in Real-Time & Embedded Computing Systems
VORTEX CoLab
U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO