Simultaneous Multithreading in Mixed-Criticality Real-Time Systems

Joshua Bakita, Shareef Ahmed, Sims Hill Osborne, Stephen Tang, Jingyuan Chen, F. Donelson Smith, James H. Anderson

Department of Computer Science
University of North Carolina, Chapel Hill

How do we get more capacity out of multicore?

How do we get more capacity out of multicore?

What if we also combine this?

Mixed-Criticality Provisioning

- ➔ Reduces capacity loss by reclaiming slack for low-criticality work

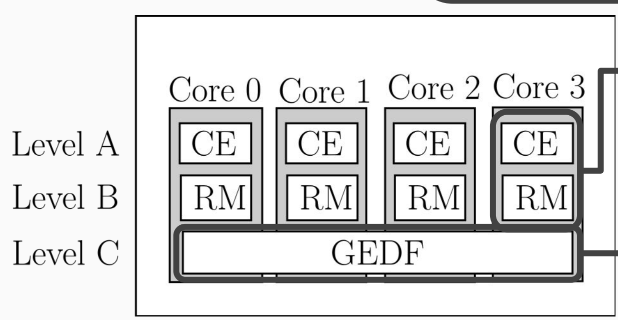
Hardware Partitioning

- ➔ Reduces capacity loss by removing interference that inflates execution times

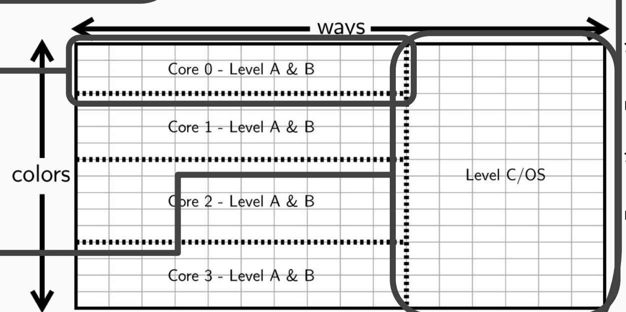
Combined in MC², Mixed-Criticality on Multicore

Simultaneous Multithreading (SMT)

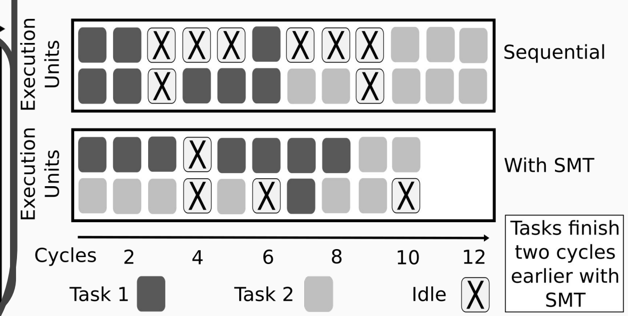
- ➔ Reduces capacity loss due to intra-core execution unit stalls
- ➔ Allows processor cores to dispatch from two i-streams ("threads") simultaneously
- ➔ Available in many CPUs now



Mixed-Criticality Scheduler Hierarchy



Shared Last-Level Cache Partitioning



Two Tasks on a Core Without/With SMT

Key Questions

SMT + Cache Partitioning	SMT + Mixed-Criticality (MC) Provisioning	Evaluation of SMT + Cache Partitioning + MC Provisioning
Can we handle many shared cache levels?	How to map SMT into a mixed-criticality context?	What are the quantitative benefits?
Does it help SMT?		Can we validate the benefits via a case study?

SMT + Cache Partitioning

Question 1 of 3

SMT + Cache Partitioning

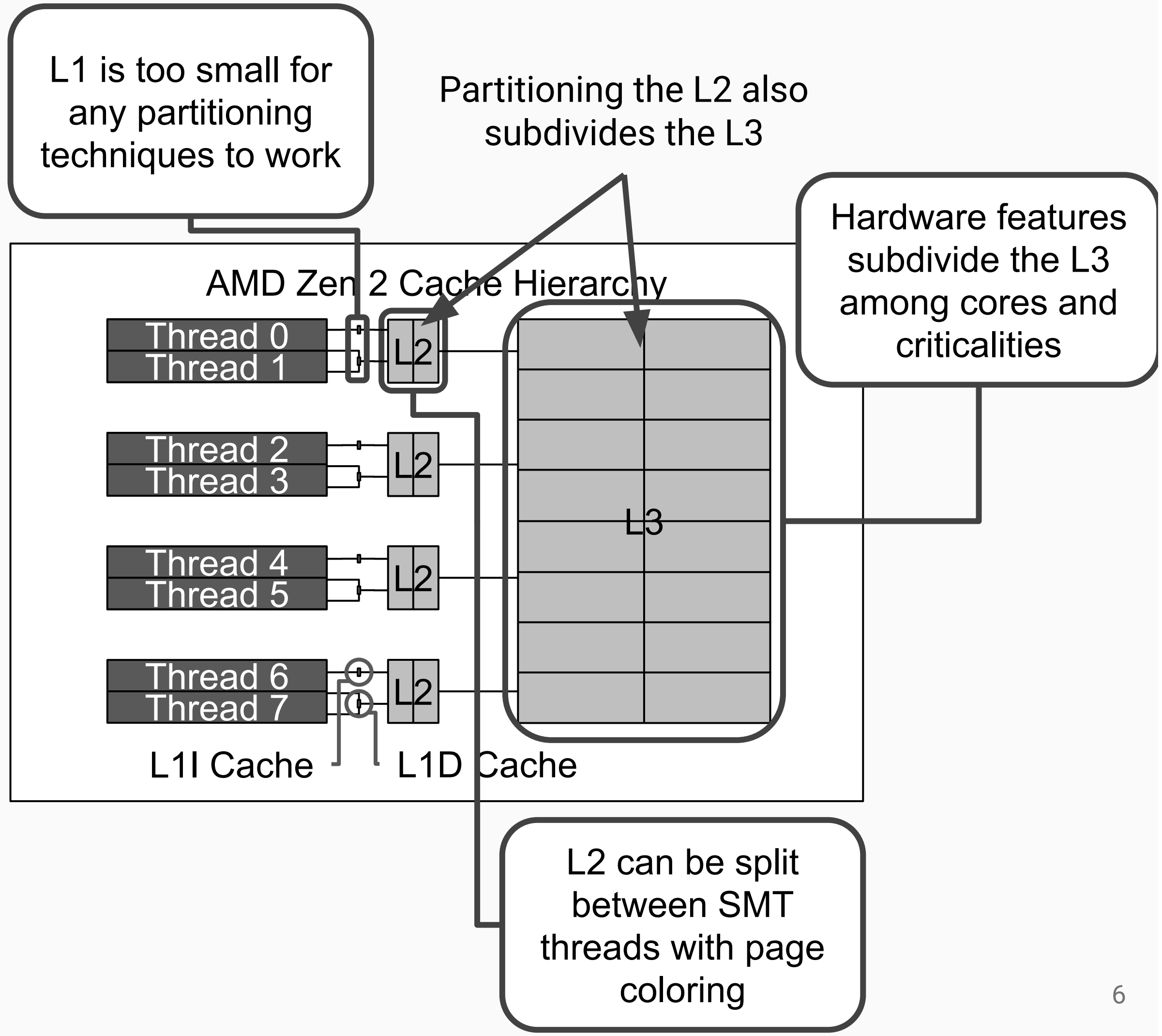
Can we handle many shared cache levels?

Consider our platform: the AMD Ryzen 9 3950X (chosen for its similarity to upcoming embedded ARM designs)

SMT threads share the L1I, L1D, L2, and L3 caches!

Can we simultaneously partition that many caches?

- ➔ Implementation is 23 lines, versus hundreds before
- ➔ More efficient and comprehensive than prior page coloring work



SMT + Cache Partitioning

Does this help SMT?

We measure the maximum execution time of all possible task pairings under all cache partitioning approaches and compare to sequential execution times.

Observations:

- ➔ SMT is broadly beneficial
- ➔ Cache isolation minimally impacts SMT effectiveness

Bench Suite	**Configuration**	**% of Pairings where SMT is Beneficial**
TACLe	No Cache Iso.	85%
	L3 Isolation	83%
	L2+L3 Iso	85%
DIS	No Cache Iso.	100%
	L3 Isolation	100%
	L2+L3 Isolation	100%
SD-VBS	No Cache Iso.	95%
	L3 Isolation	95%
	L2+L3 Isolation	95%

SMT + Mixed-Criticality (MC) Provisioning

Question 2 of 3

SMT + MC Provisioning

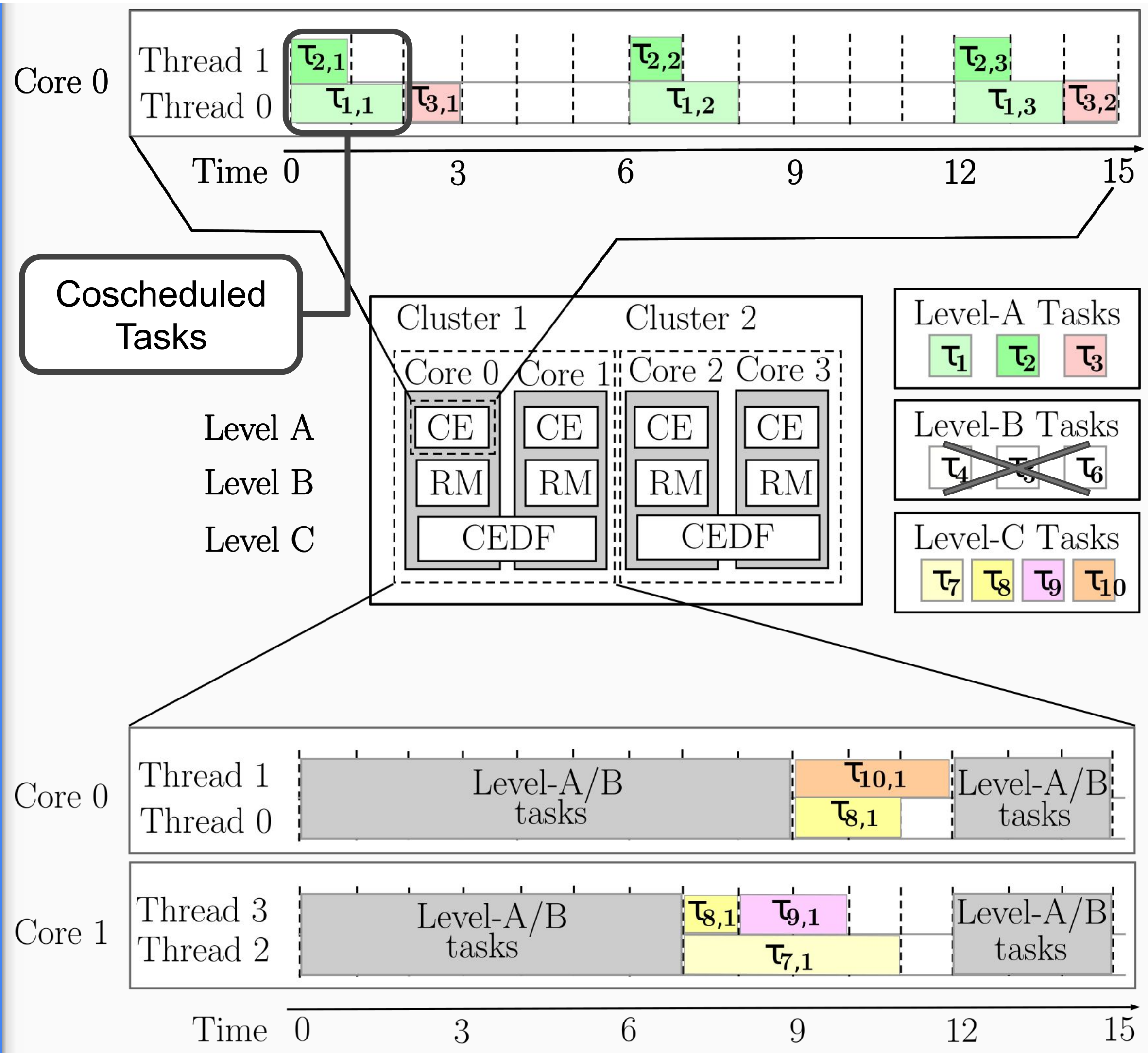
How to map SMT into a MC context?

Level A uses coscheduling [1,3]

Level B uses coscheduling [1,3]

Level C uses clustered EDF

- ➔ Each cluster is either *threaded* or *unthreaded*
- ➔ Threaded clusters treat threads as additional cores (as in [2])
- ➔ Unthreaded clusters behave similarly to standard CEDF



$\tau_{a,x}$ indicates the *x*th job of task *a*.

Evaluation of SMT + Cache Partitioning + Mixed Criticality Provisioning

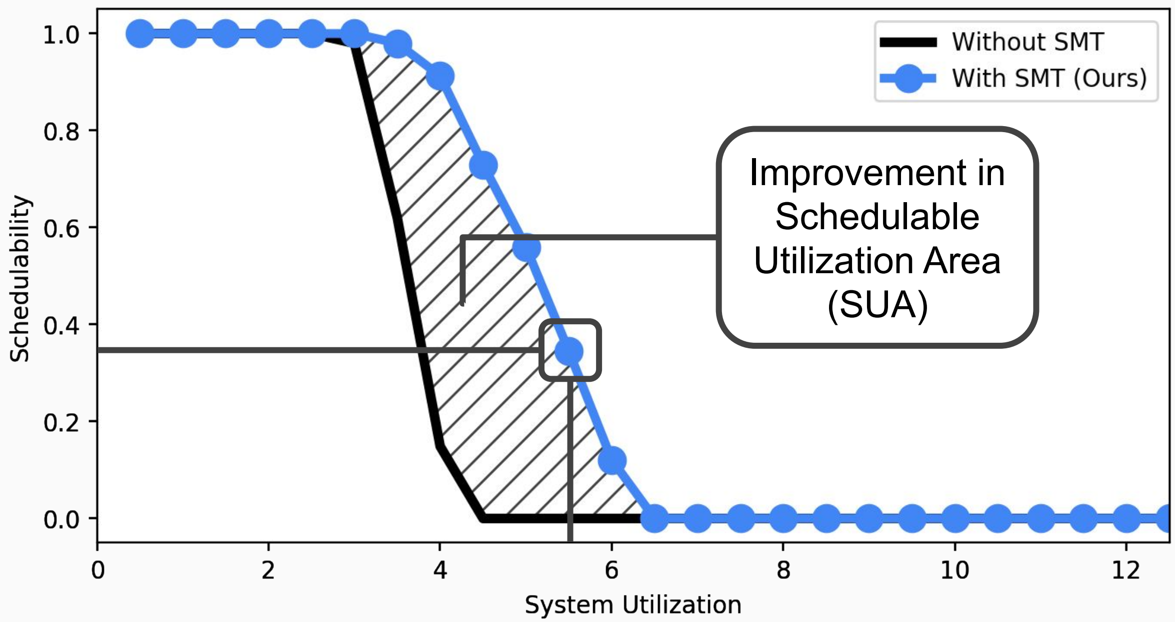
Question 3 of 3

Evaluation

What are the benefits?

We measure improvement with an overhead-aware *schedulability study*

- ➔ Results show what percentage of synthetic task systems of a specific total utilization can be scheduled such that they meet all deadlines.
- ➔ We consider 240 different synthetic system configurations (with parameters informed by benchmarks).



Sample Schedulability Graph

32%

Average Improvement in Schedulable Utilization Area (SUA)

Evaluation of SMT + Cache Partitioning + MC Provisioning

Can we validate the benefits via a case study?

Case Study:

- ➔ Do tasksets claimed schedulable by our schedulability study run without deadline misses on our platform?

We implemented our system combining SMT + Multi-Level Cache Partitioning + Mixed Criticality Provisioning in LITMUS^{RT} 5.4.

Results:

- ➔ Tested 10 tasksets for 60 minutes (tens of thousands of jobs)
- ➔ No deadline misses at any criticality level!
  - ◆ Surprising due to the presence of soft-real time tasks
  - ◆ May indicate that our provisioning is conservative

Conclusions

SMT + Cache Partitioning	SMT + Mixed-Criticality (MC) Provisioning	Evaluation of SMT + Cache Partitioning + MC Provisioning
Can we handle many shared cache levels?	How to map SMT into a mixed-criticality context?	What are the quantitative benefits?
Yes!	Coscheduling for high-criticality Clustered scheduling for low-criticality	32%
Does it help SMT?		Can we validate the benefits via a case study?
No		Yes!

Thanks!
Questions?

Read our paper!

Future work:

➔ Effects of other isolation techniques on SMT behavior?

➔ GPU sharing in a mixed-criticality system?

➔ SSD sharing in a mixed-criticality system?

Contact:
Email: jbakita@cs.unc.edu
Web: https://cs.unc.edu/~jbakita

Flowers of the University of North Carolina at Chapel Hill, Own Work, Spring 2021