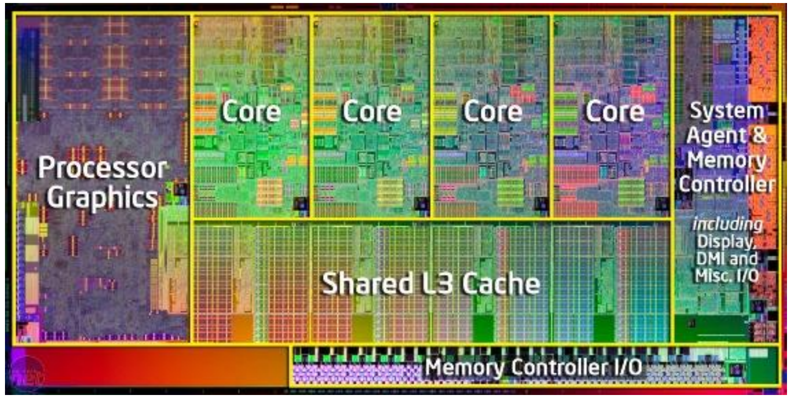
Deterministic Memory Abstraction and Supporting Multicore System Architecture

Farzad Farshchi[$], Prathap Kumar Valsan[^], Renato Mancuso[*], Heechul Yun[$]

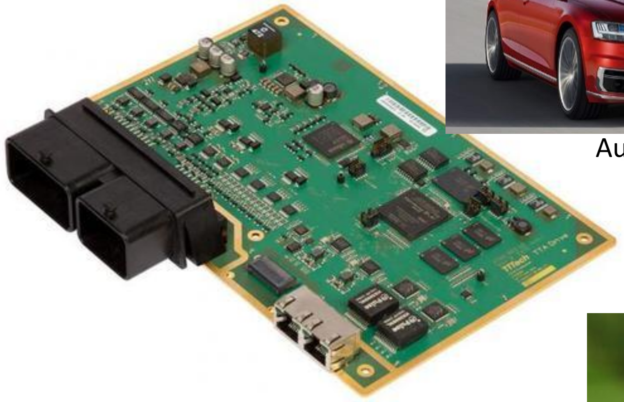
[$] University of Kansas, [^] Intel, [*] Boston University

Multicore Processors in CPS

- Provide high computing **performance** needed for intelligent CPS
- Allow **consolidation** reducing cost, size, weight, and power.



Die-shot of a multicore processor



Audi zFAS platform

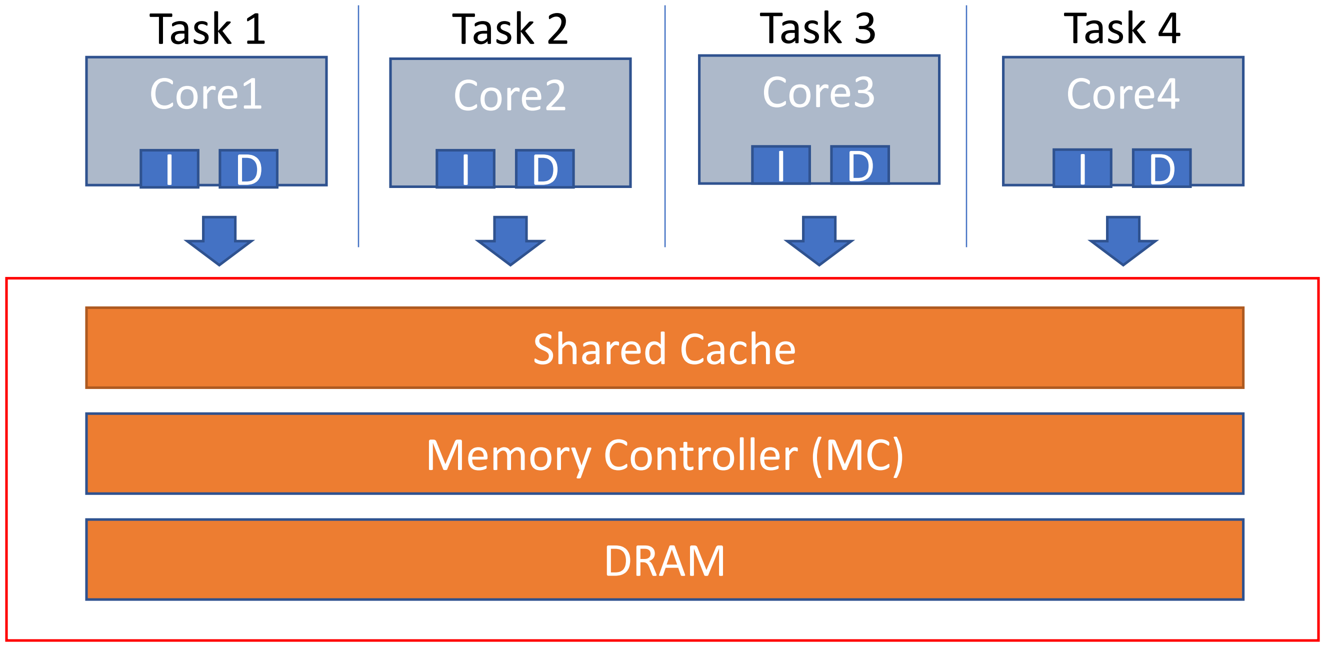
Audi A8

Google/Waymo self-driving car

DJI drone

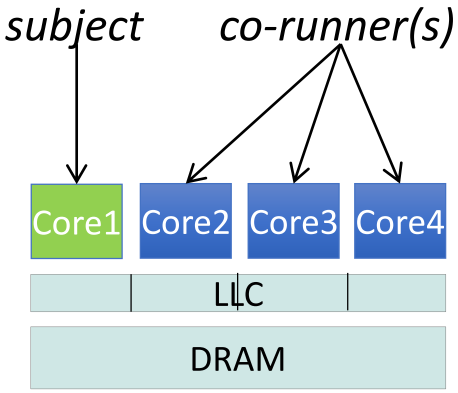
Challenge: Shared Memory Hierarchy

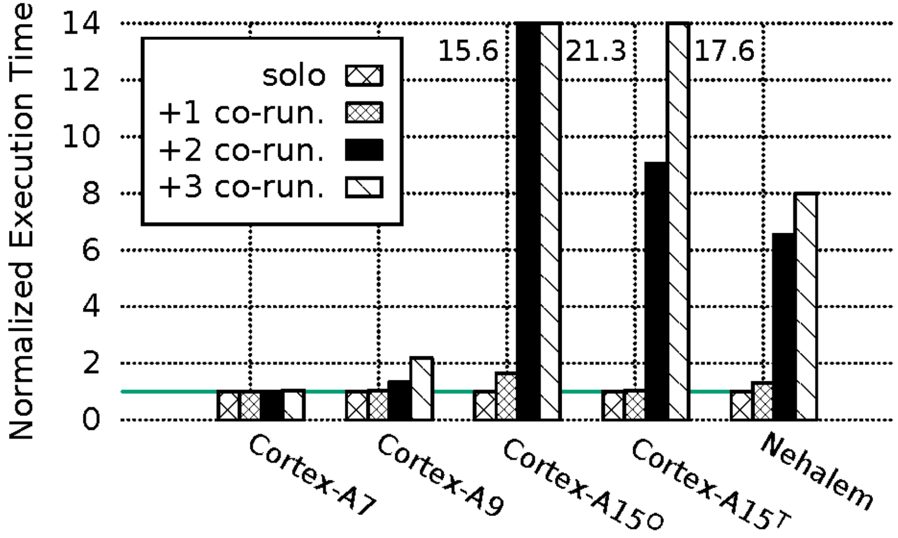
- **Memory performance** varies widely due to **interference**
- Task WCET can be **extremely pessimistic**



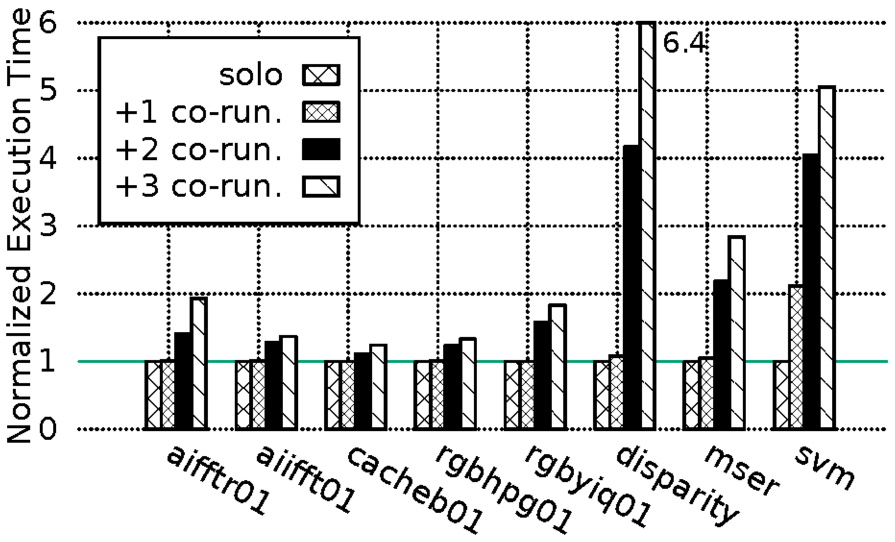
Challenge: Shared Memory Hierarchy

- Many shared resources: cache space, MSHRs, dram banks, MC buffers, …
- Each **optimized for performance** with no high-level insight
- Very **poor worst-case behavior**: >10X, >100X observed in real platforms
  - even after cache partitioning is applied.





IsolBench

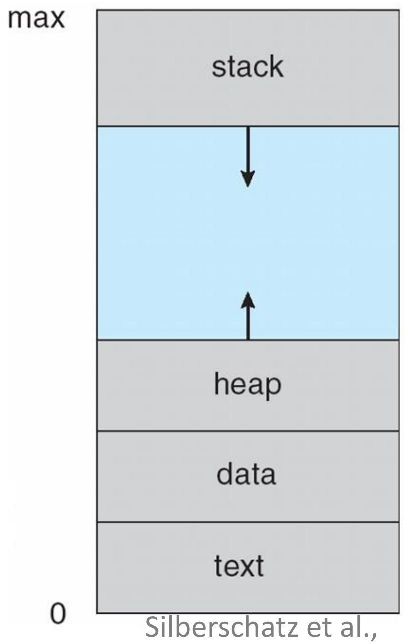


EEMBC, SD-VBS on Tegra TK1

- P. Valsan, et al., “Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems.” RTAS, 2016
- P. Valsan, et al., “Addressing Isolation Challenges of Non-blocking Caches for Multicore Real-Time Systems.” Real-time Systems Journal. 2017

The Virtual Memory Abstraction

- Program's logical view of memory
- No concept of timing
- OS maps *any* available physical memory blocks
- Hardware treats *all* memory requests as same
- But some memory may be more important
  - E.g., code and data memory in a time critical control loop
- Prevents OS and hardware from making **informed allocation and scheduling decisions** that affect memory timing



Silberschatz et al., "Operating System Concepts."

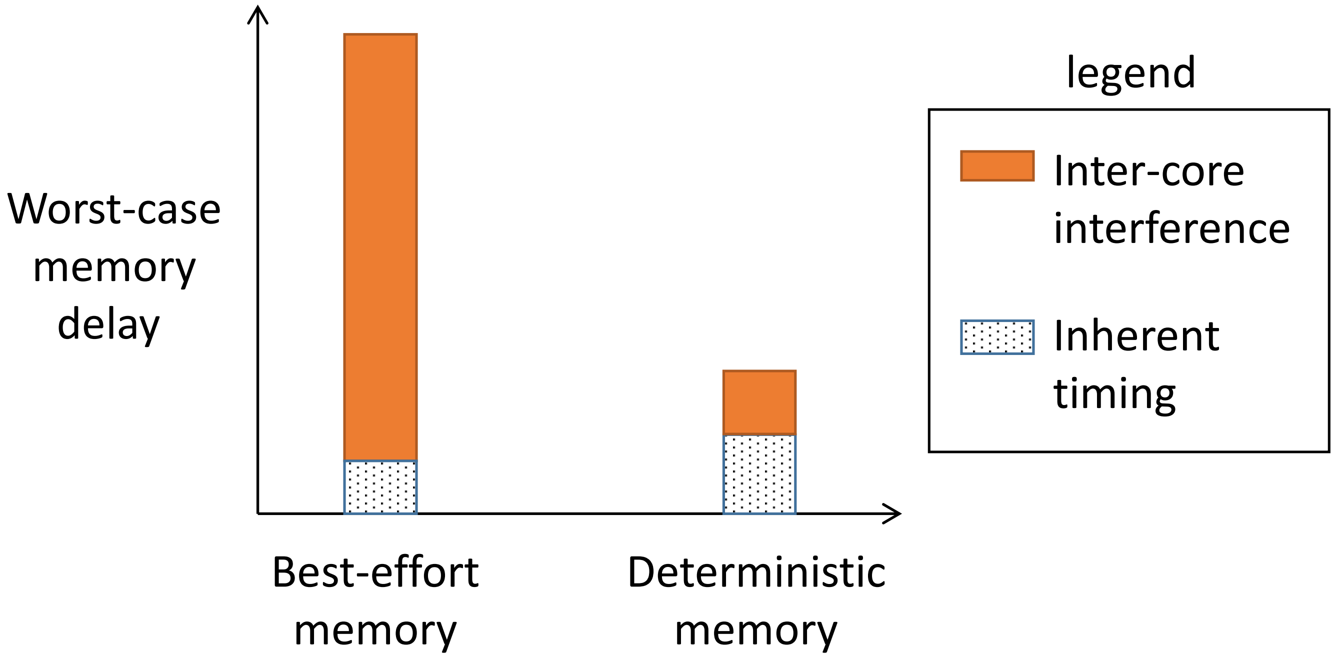
New memory abstraction is needed!

Outline

- Motivation
- **Our Approach**
  - **Deterministic Memory Abstraction**
  - **DM-Aware Multicore System Design**
  - **Implementation**
- Evaluation
- Conclusion

Deterministic Memory (DM) Abstraction

- **Cross-layer** memory abstraction with bounded worst-case timing
- Focusing on tightly bounded **inter-core interference**
- Best-effort memory = conventional memory abstraction



DM-Aware Resource Management Strategies

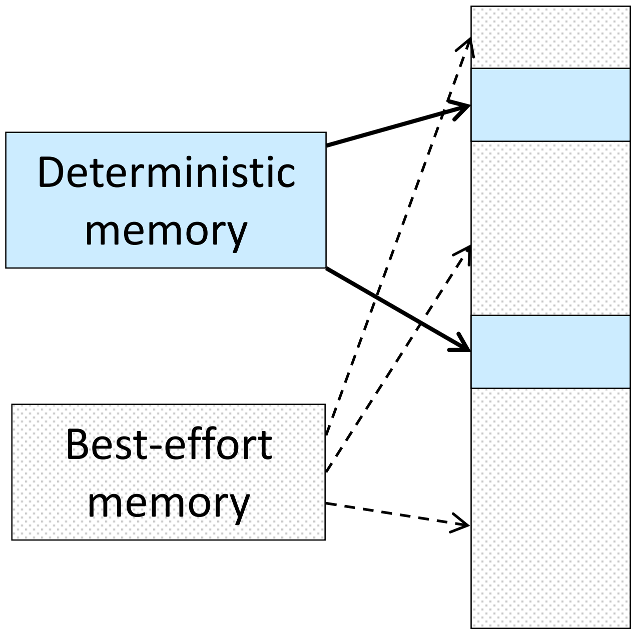
- Deterministic memory: Optimize for time predictability
- Best effort memory: Optimize for average performance

	Space allocation	Request scheduling	WCET bounds
Deterministic memory	Dedicated resources	Predictability focused	Tight
Best-effort memory	Shared resources	Performance focused	Pessimistic

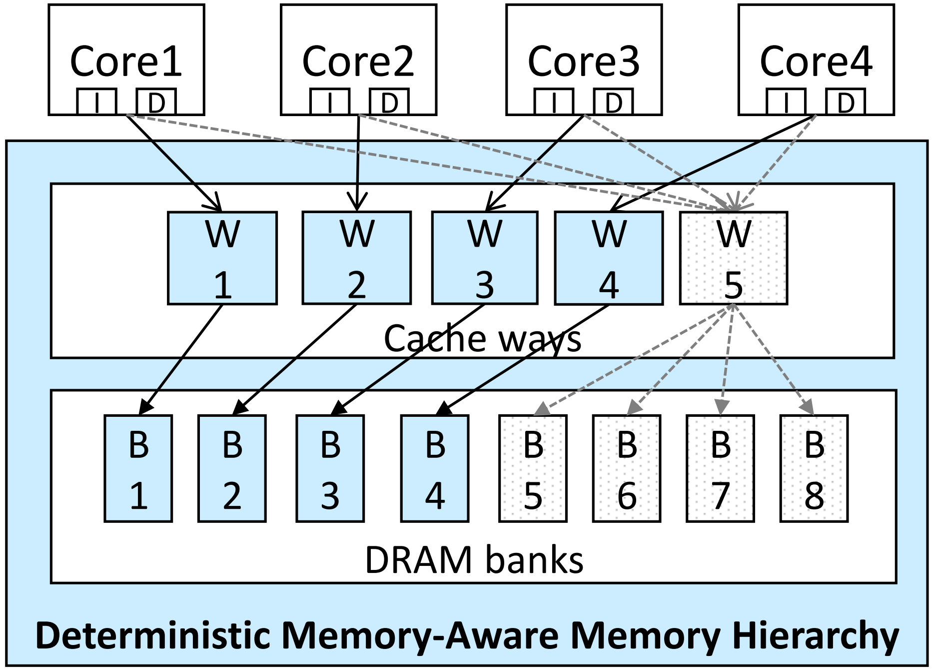
Space allocation: e.g., cache space, DRAM banks
Request scheduling: e.g., DRAM controller, shared buses

System Design: Overview

- Declare all or part of RT task's address space as deterministic memory
- End-to-end DM-aware resource management: from OS to hardware
  - Assumption: partitioned fixed-priority real-time CPU scheduling



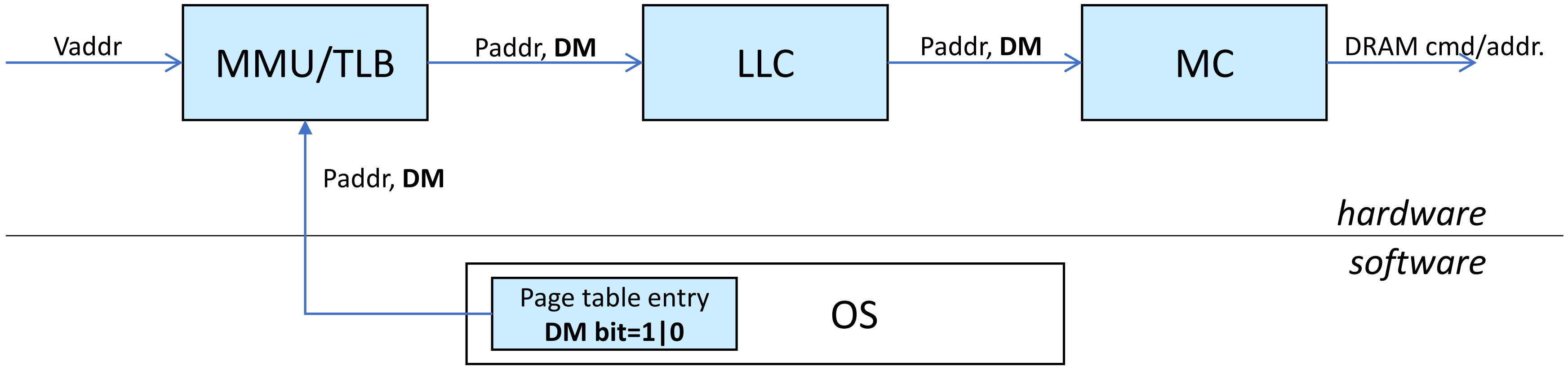
Application view (logical)



System-level view (physical)

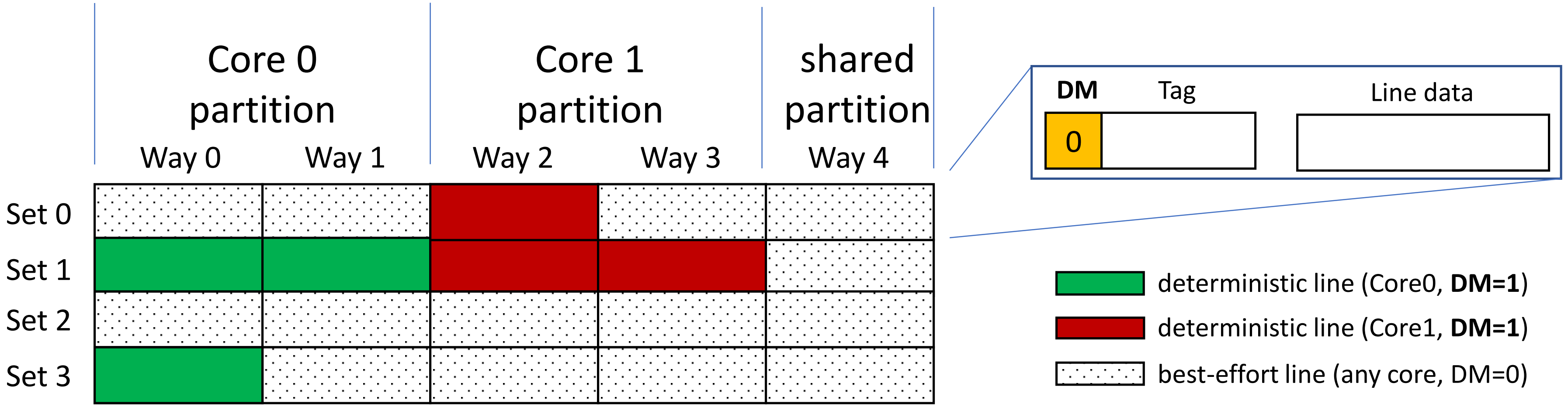
OS and Architecture Extensions for DM

- OS *updates* the **DM bit** in each page table entry (PTE)
- MMU/TLB and bus *carries* the DM bit.
- Shared memory hierarchy (cache, memory ctrl.) *uses* the DM bit.



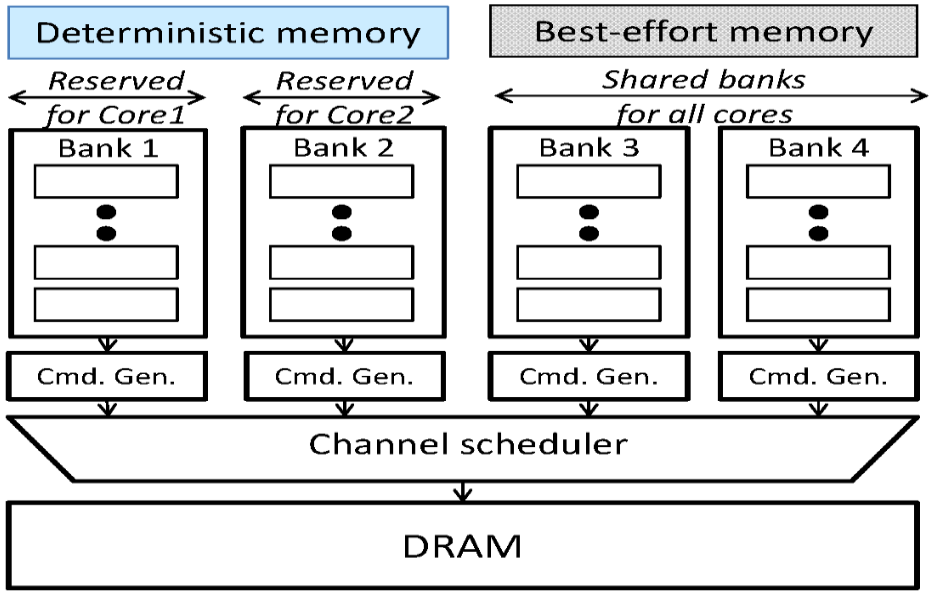
DM-Aware Shared Cache

- Based on cache *way partitioning*
- **Improve space utilization** via DM-aware cache replacement algorithm
  - *Deterministic memory*: allocated on its own dedicated partition
  - *Best-effort memory*: allocated on any non-DM lines in any partition.
  - *Cleanup*: DM lines are recycled as best-effort lines at each OS context switch

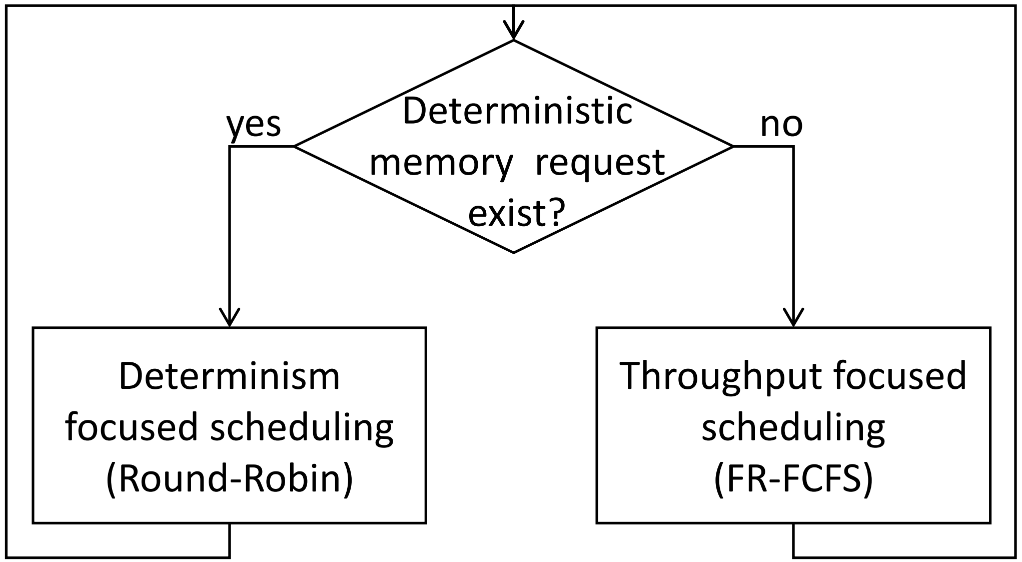


DM-Aware Memory (DRAM) Controller

- OS allocates DM pages on **reserved banks**, others on shared banks
- Memory-controller implements **two-level scheduling** (*)



(a) Memory controller (MC) architecture

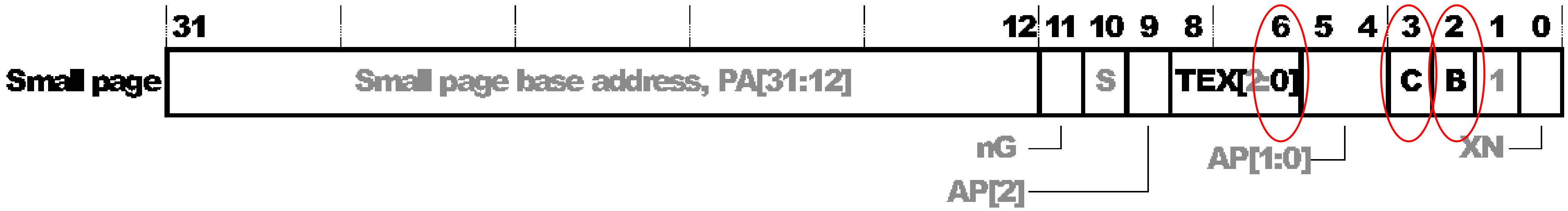


(b) Scheduling algorithm

(*) P. Valsan, et al., "MEDUSA: A Predictable and High-Performance DRAM Controller for Multicore based Embedded Systems." CPSNA, 2015

Implementation

- Fully functional **end-to-end implementation** in Linux and Gem5.
- Linux kernel extensions
  - Use an unused bit combination in ARMv7 PTE to indicate a DM page.
  - Modified the ELF loader and system calls to declare DM memory regions
  - DM-aware page allocator, replacing the buddy allocator, based on PALLOC (*)
    - Dedicated DRAM banks for DM are configured through Linux's CGROUP interface.



ARMv7 page table entry (PTE)

(*) H. Yun et. al, "PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms." RTAS'14

Implementation

- Gem5 (a cycle-accurate full-system simulator) extensions
  - MMU/TLB: Add DM bit support
  - Bus: Add the DM bit in each bus transaction
  - Cache: implement way-partitioning and DM-aware replacement algorithm
  - DRAM controller: implement DM-aware two-level scheduling algorithm.

- Hardware implementability
  - MMU/TLB, Bus: adding 1 bit is not difficult. (e.g., Use AXI bus QoS bits)
  - Cache and DRAM controllers: logic change and additional storage are minimal.

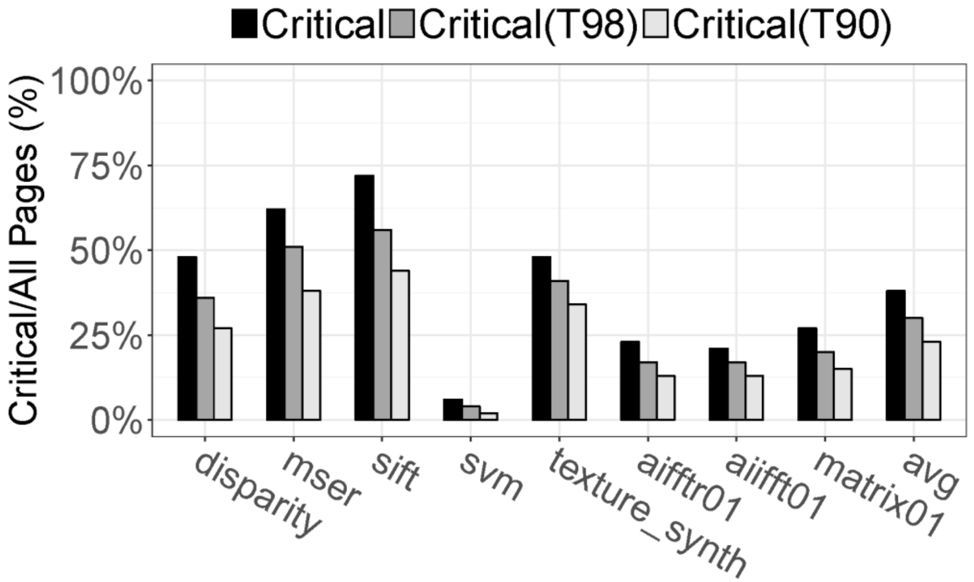
Outline

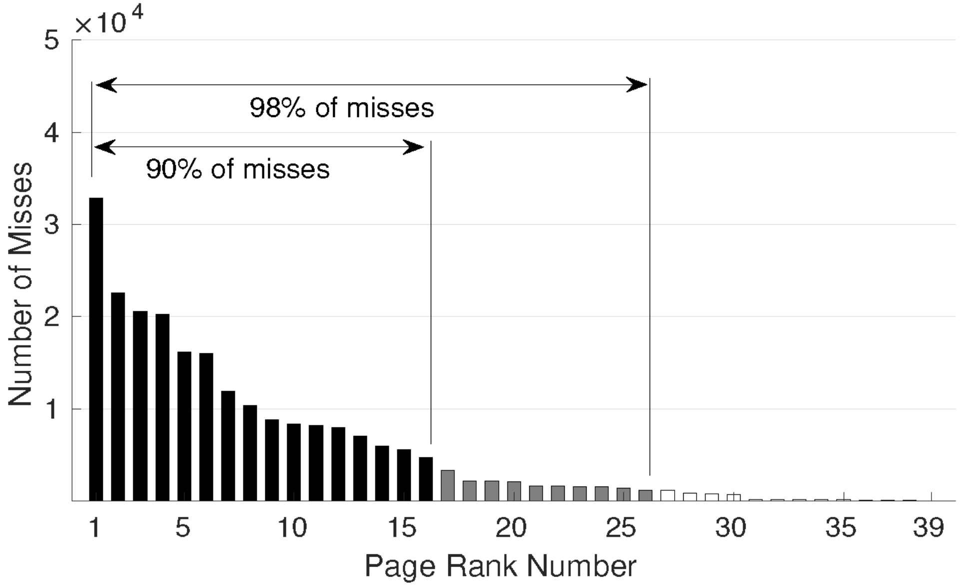
- Motivation
- Our Approach
  - Deterministic Memory Abstraction
  - DM-Aware Multicore System Design
  - Implementation
- **Evaluation**
- Conclusion

Simulation Setup

- Baseline Gem5 full-system simulator
  - 4 out-of-order cores, 2 GHz
  - 2 MB Shared L2 (16 ways)
  - LPDDR2@533MHz, 1 rank, 8 banks
  - Linux 3.14 (+DM support)
- Workload
  - EEMBC, SD-VBS, SPEC2006, IsolBench

Real-Time Benchmark Characteristics

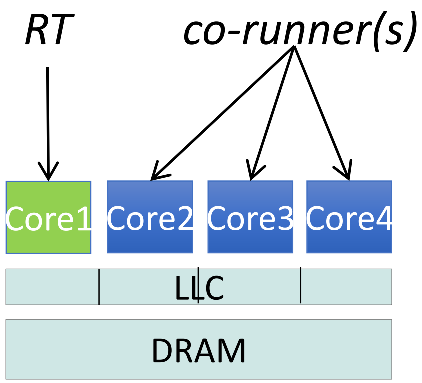




- **Critical pages**: pages accessed in the main loop
  - Critical (T98/T90): pages accounting 98/90% L1 misses of all L1 misses of the critical pages.
- Only 38% of pages are critical pages
- Some pages contribute more to L1 misses (hence shared L2 accesses)
- **Not all memory is equally important**

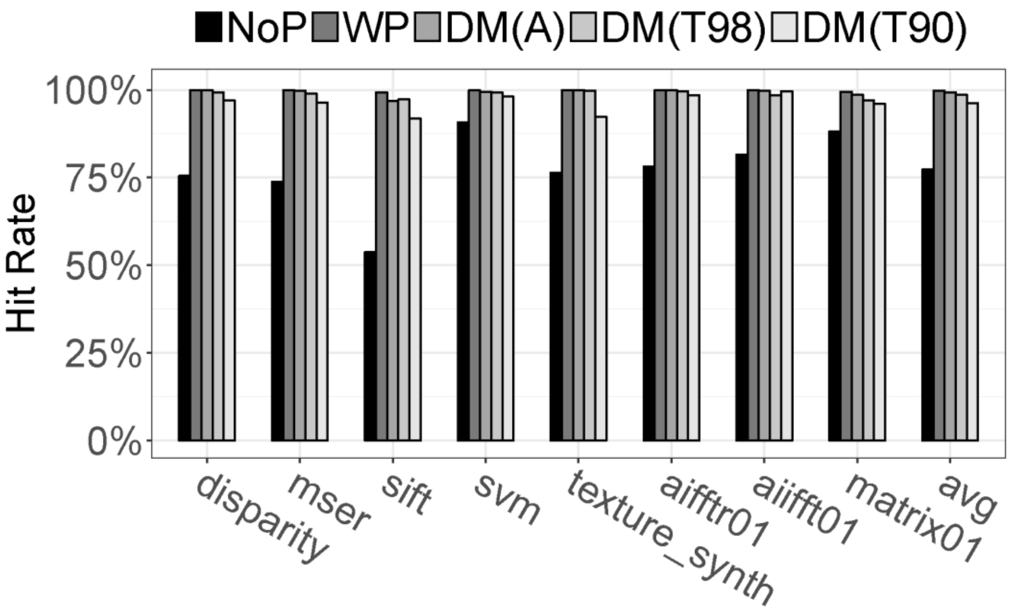
Effects of DM-Aware Cache

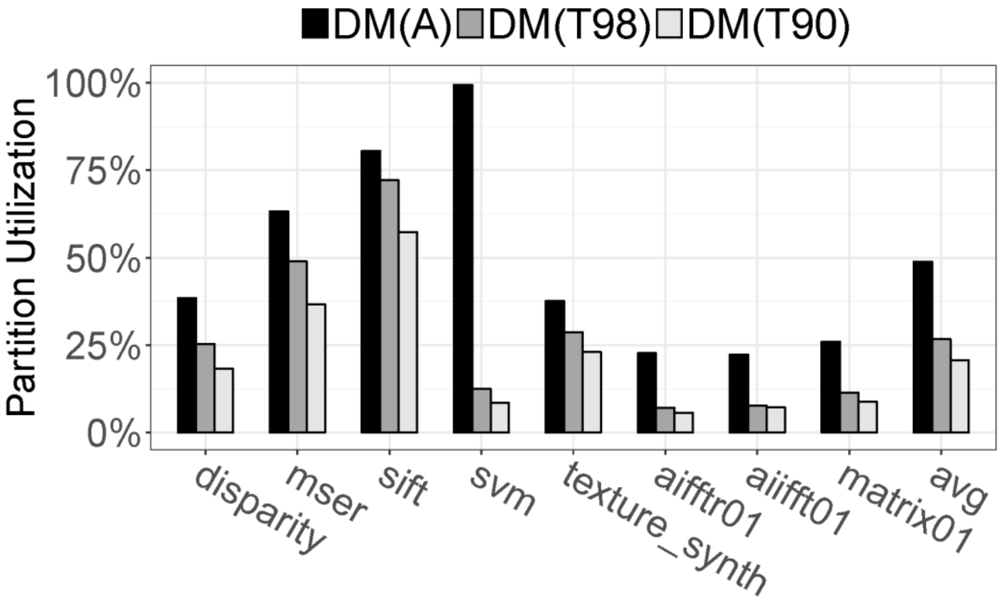
- Questions
  - Does it provide strong cache isolation for real-time tasks using DM?
  - Does it improve cache space utilization for best-effort tasks?
- Setup
  - RT: EEMBC, SD-VBS, co-runners: 3x Bandwidth
- Comparisons
  - NoP: free-for-all sharing. No partitioning
  - WP: cache way partitioning (4 ways/core)
  - DM(A): all critical pages of a benchmark are marked as DM
  - DM(T98): critical pages accounting 98% L1 misses are marked as DM
  - DM(T90): critical pages accounting 98% L1 misses are marked as DM



Effects of DM-Aware Cache

- Does it provide strong cache isolation for real-time tasks using DM?
  - Yes. DM(A) and WP (way-partitioning) offer equivalent isolation.
- Does it improve cache space utilization for best-effort tasks?
  - Yes. DM(A) uses only 50% cache partition of WP.



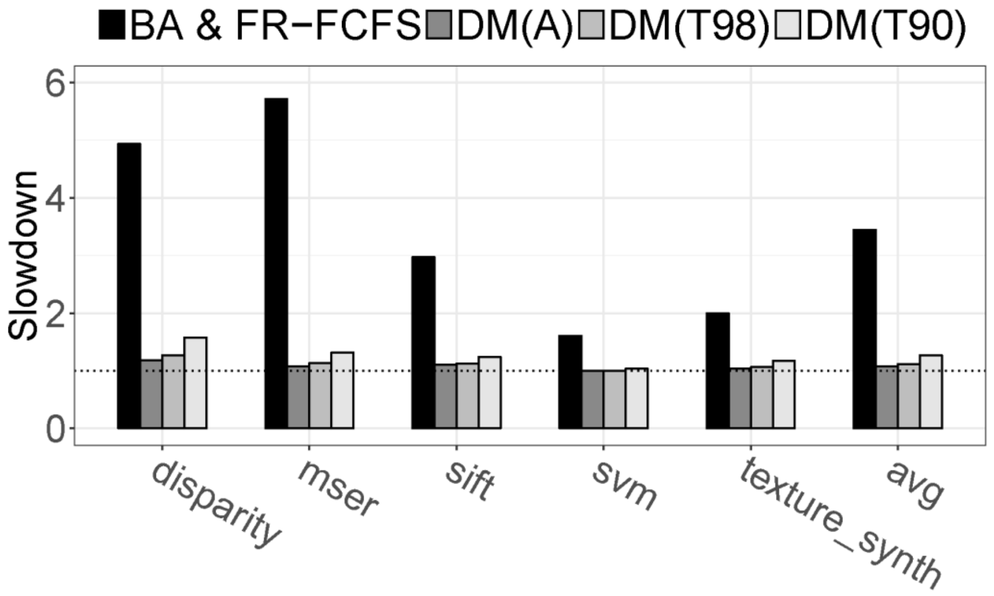


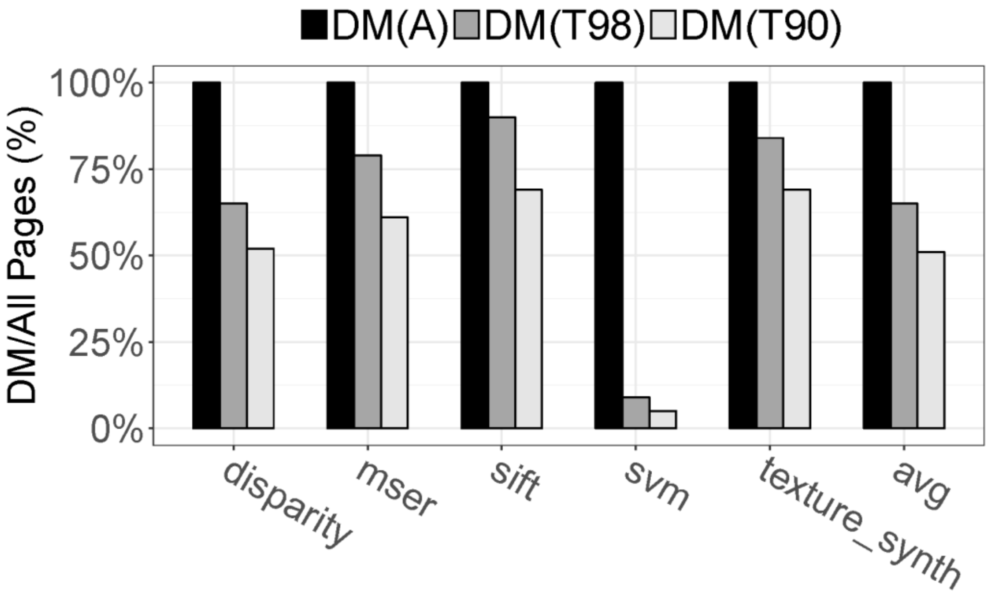
Effects of DM-Aware DRAM Controller

- Questions
  - Does it provide strong isolation for real-time tasks using DM?
  - Does it reduce reserved DRAM bank space?
- Setup
  - RT: SD-VBS (input: CIF), co-runners: 3x Bandwidth
- Comparisons
  - BA & FR-FCFS: Linux's default buddy allocator + FR-FCFS scheduling in MC
  - DM(A): DM on private DRAM banks + two-level scheduling in MC
  - DM(T98): same as DM(A), except pages accounting 98% L1 misses are DM
  - DM(T90): same as DM(A), except pages accounting 90% L1 misses are DM

Effects of DM-Aware DRAM Controller

- Does it provide strong isolation for real-time tasks using DM?
  - Yes. BA&FR-FCFS suffers 5.7X slowdown.
- Does it reduce reserved DRAM bank space?
  - Yes. Only 51% of pages are marked deterministic in DM(T90)





Conclusion

- Challenge
  - Balancing performance and predictability in multicore real-time systems
  - Memory timing is important to WCET
  - The current memory abstraction is limiting: no concept of timing.
- Deterministic Memory Abstraction
  - Memory with tightly bounded worst-case timing.
  - Enable predictable and high-performance multicore systems
- DM-aware multicore system designs
  - OS, MMU/TLB, bus support
  - DM-aware cache and DRAM controller designs
  - Implemented and evaluated in Linux kernel and gem5
- Availability
  - https://github.com/CSL-KU/detmem

Ongoing/Future Work

- SoC implementation in FPGA
  - Based on open-source *RISC-V* quad-core SoC
  - Basic DM support in bus protocol and Linux
  - Implementing DM-aware cache and DRAM controllers

- Tool support and other applications
  - Finding "optimal" deterministic memory blocks
  - Better timing analysis integration (initial work in the paper)
  - Closing micro-architectural side-channels.

Thank You

Disclaimer:

Our research has been supported by the National Science Foundation (NSF) under the grant number CNS 1718880