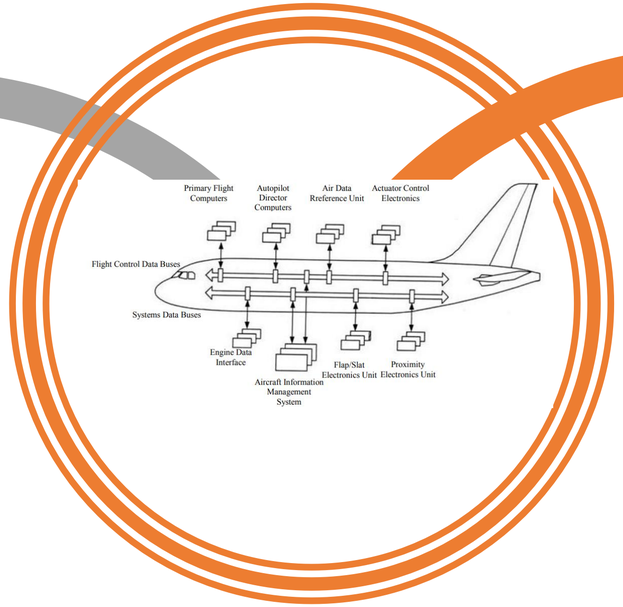
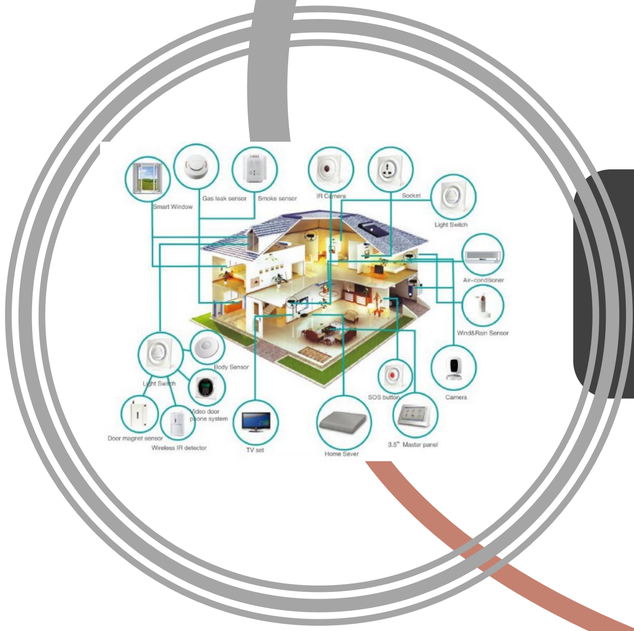
Discriminative Coherence: Balancing Performance and Latency Bounds in Data-sharing Multi-Core Real-Time Systems

Mohamed Hassan

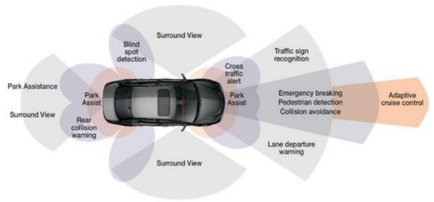

MOTIVATION
01
EXISTING SOLUTIONS
02
OBSERVATIONS
03
DISCO
04
RESULTS
05

Outline





Data is a Key in all modern applications



MOTIVATION

Common Data Communication Approaches

Ignore

- Adopts an independent-task model → No communication amongst tasks

Prevent

- Enforcing complete isolation between tasks.
  - At the shared cache: strict cache partitioning and coloring
  - At the DRAM: bank privatization

Common Data Communication Approaches

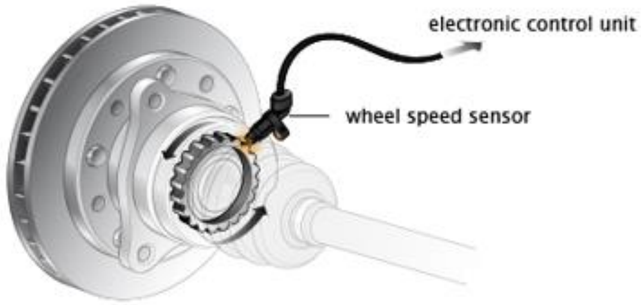
- May result in a poor memory or cache utilization
  - e.g.: a task has conflict misses, while other partitions may remain underutilized
- Does not scale with increasing number of cores
  - e.g.: number of PEs ≤ number of DRAM banks
- Not viable in emerging systems due to increased functionality and massive data

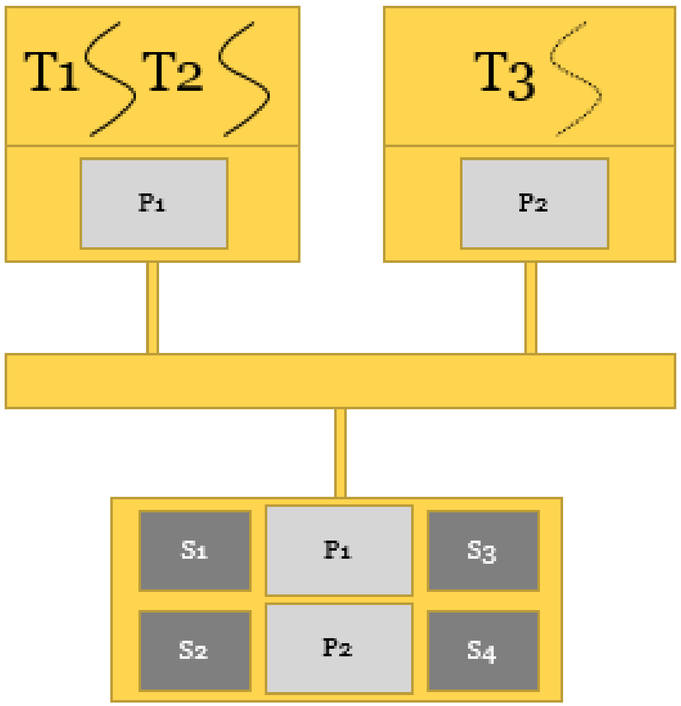
Ignore

- Adopts an independent-task model → No communication amongst tasks

Prevent

- Enforcing complete isolation between tasks.
  - At the shared cache: strict cache partitioning and coloring
  - At the DRAM: bank privatization





✓ Simpler timing analysis
✗ Hardware changes
✗ Long execution time

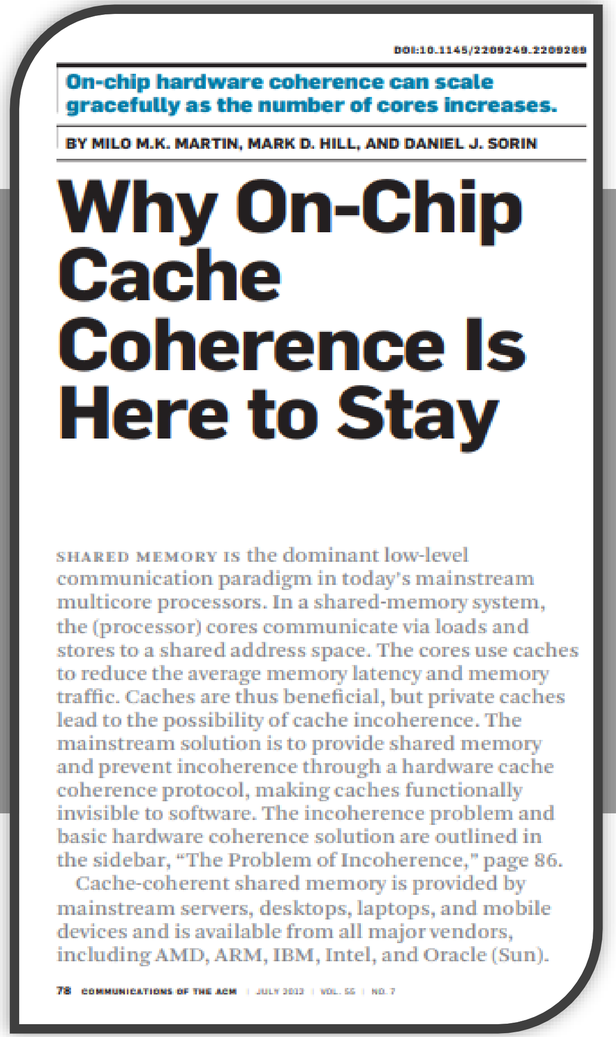
Solution:

No caching of shared data

[Hardy et al., RTSS'09]
[Lesage et al., RTNS'10]
[Bansal et al., arXiv'19]
[Chisholm et al., RTSS'16]

Existing Solutions

The mainstream solution is to provide shared memory and prevent incoherence through a hardware cache coherence protocol, making caches functionally invisible to software.



DOI:10.1145/2209249.2209269

On-chip hardware coherence can scale gracefully as the number of cores increases.

BY MILO M.K. MARTIN, MARK D. HILL, AND DANIEL J. SORIN

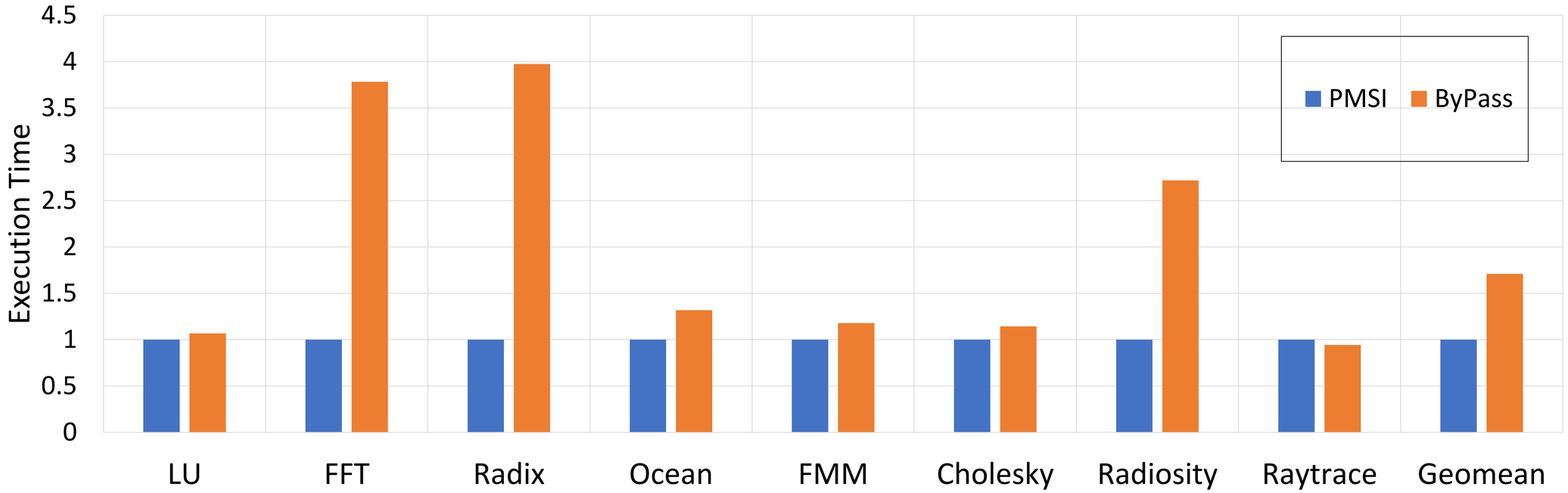
Why On-Chip Cache Coherence Is Here to Stay

SHARED MEMORY IS the dominant low-level communication paradigm in today's mainstream multicore processors. In a shared-memory system, the (processor) cores communicate via loads and stores to a shared address space. The cores use caches to reduce the average memory latency and memory traffic. Caches are thus beneficial, but private caches lead to the possibility of cache incoherence. The mainstream solution is to provide shared memory and prevent incoherence through a hardware cache coherence protocol, making caches functionally invisible to software. The incoherence problem and basic hardware coherence solution are outlined in the sidebar, "The Problem of Incoherence," page 86.

Cache-coherent shared memory is provided by mainstream servers, desktops, laptops, and mobile devices and is available from all major vendors, including AMD, ARM, IBM, Intel, and Oracle (Sun).

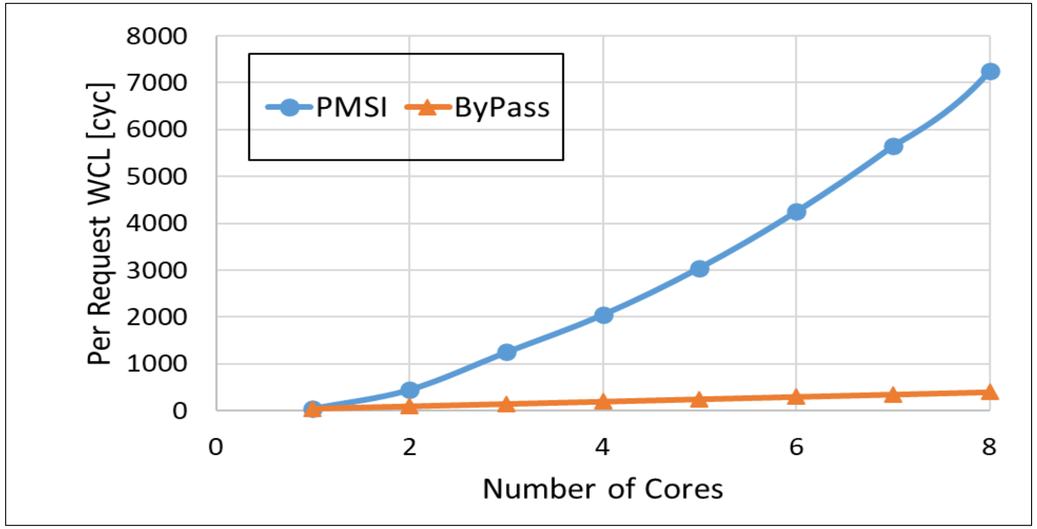
Coherence is the norm in COTS platforms

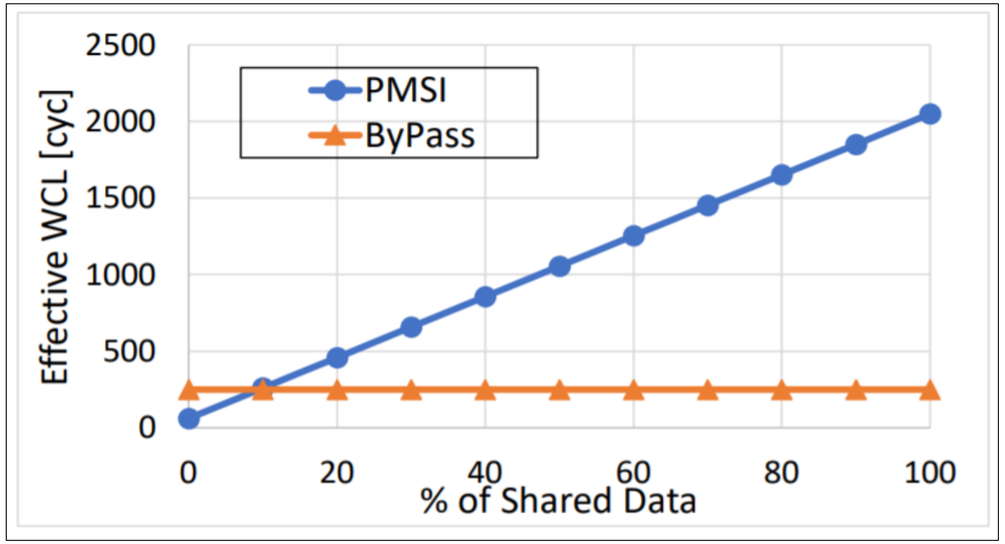
Existing Solutions



Benefit of Coherence: Up to 3x performance

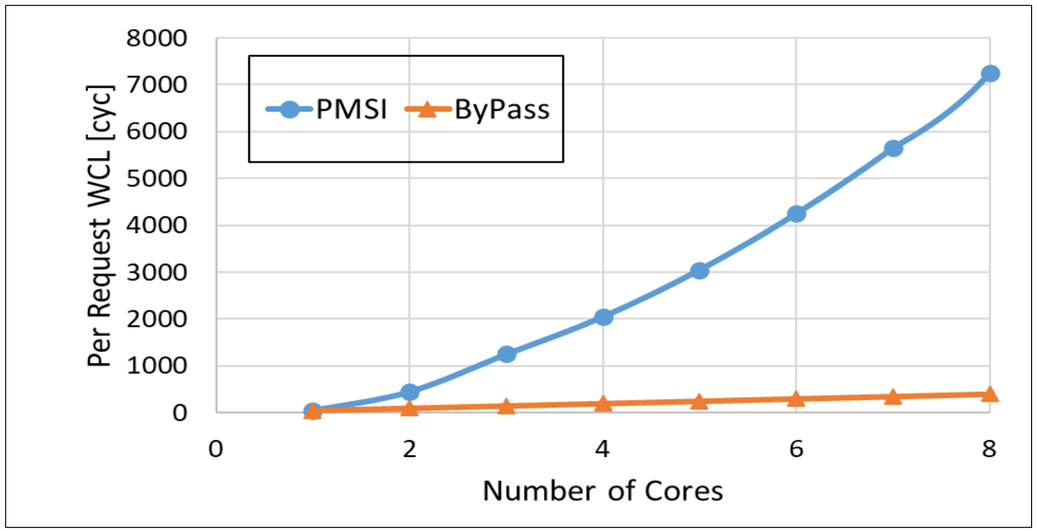
OBSERVATIONS

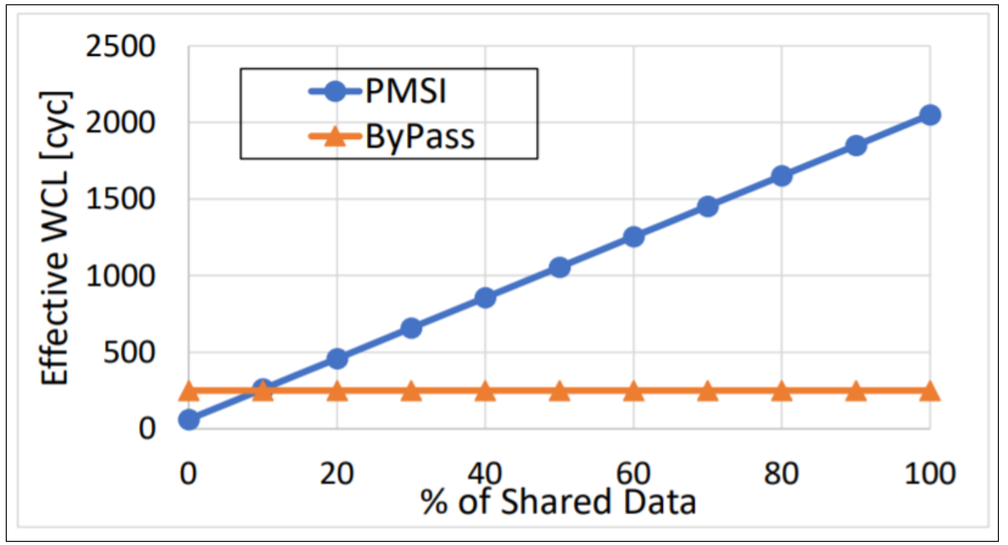




Problem: Coherence effect on WC

OBSERVATIONS

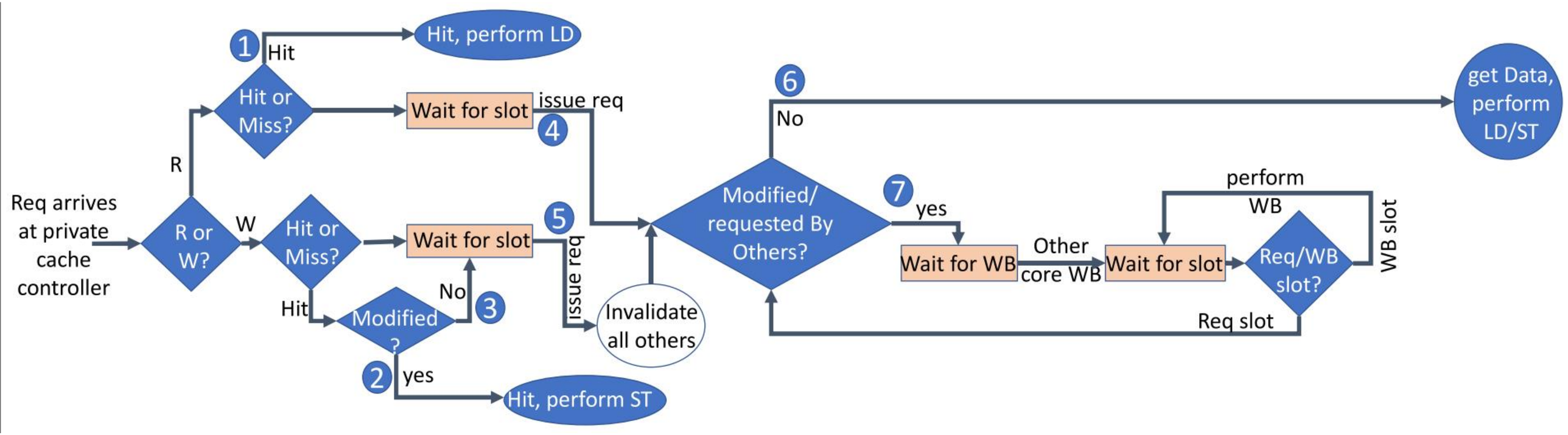




How do we improve over that?
Where does this large WCL come from "exactly"?

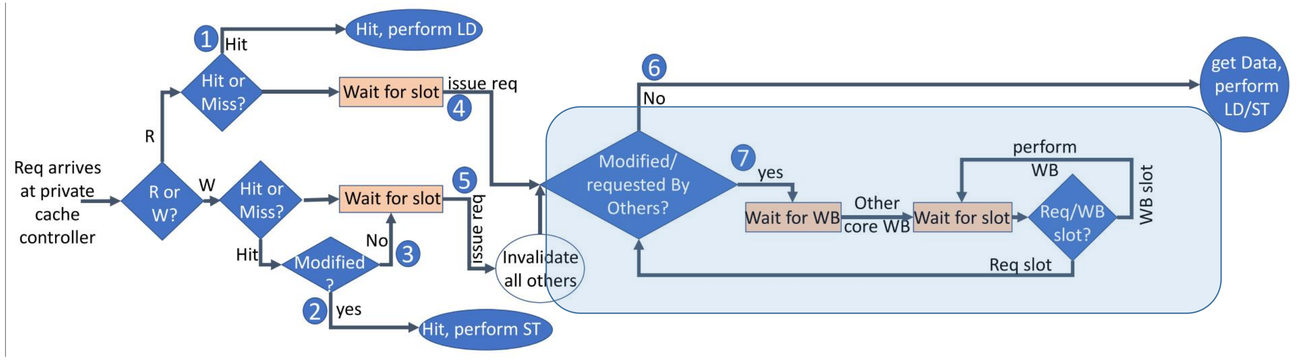
Problem of PMSI: Coherence effect on WC

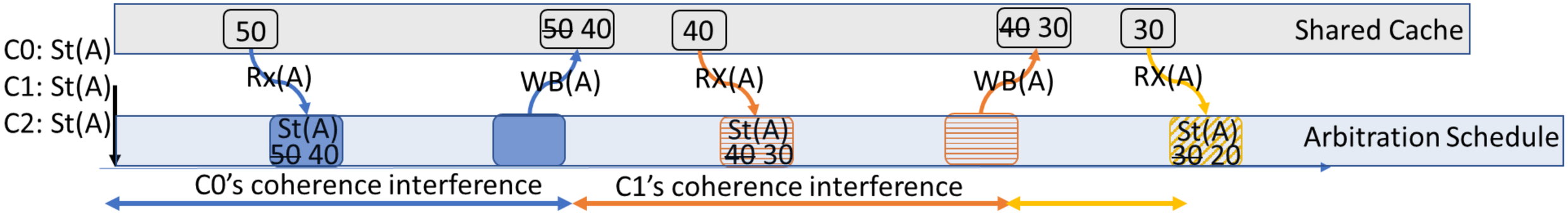
OBSERVATIONS



PMSI Flow Diagram

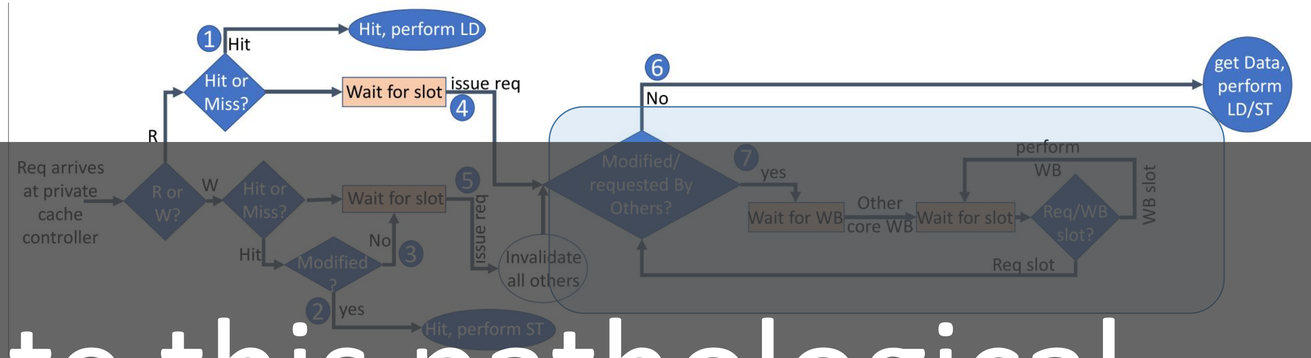
OBSERVATIONS



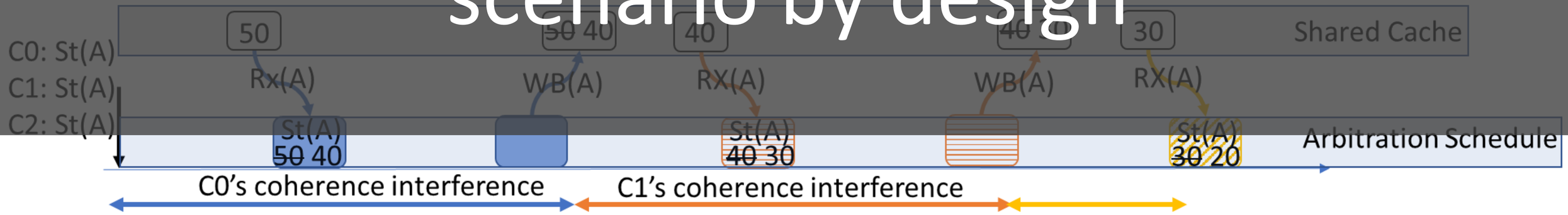


PMSI WC Scenario

OBSERVATIONS

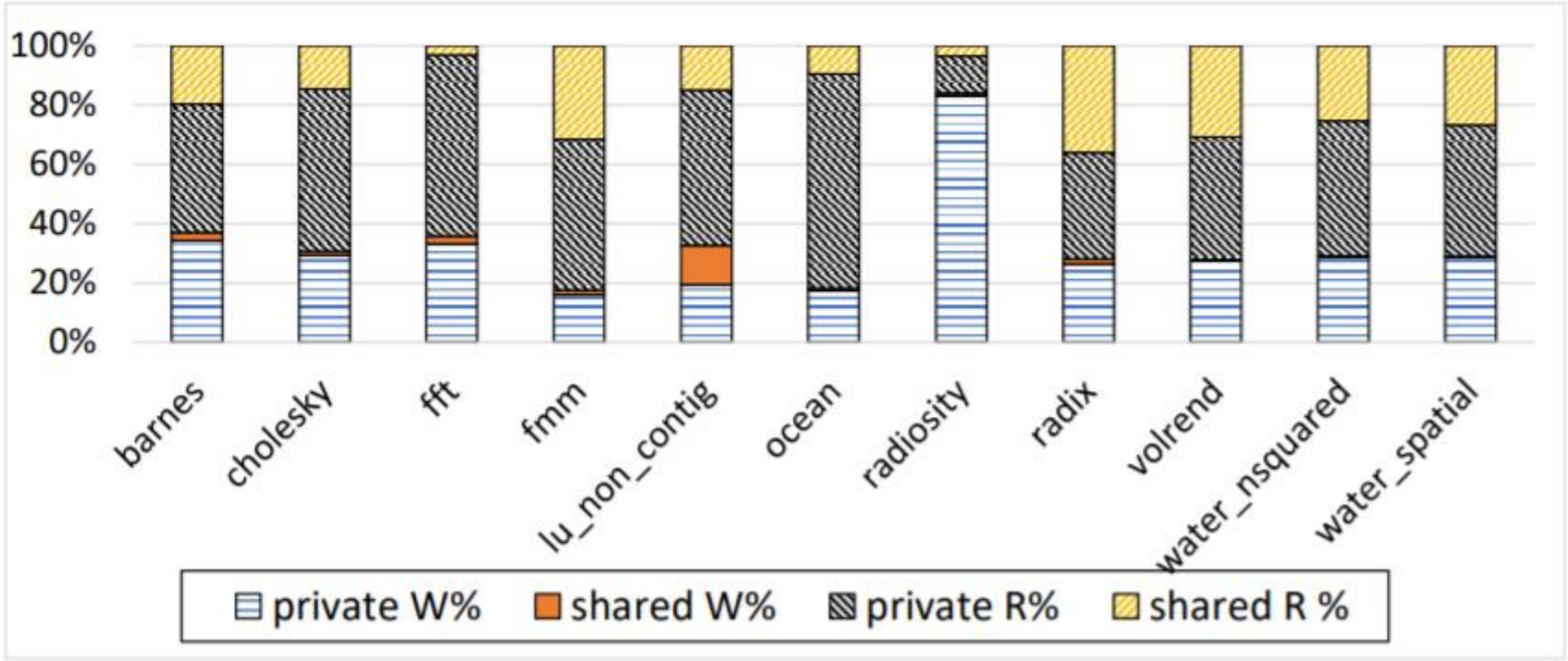


Solution: Eliminate this pathological scenario by design



PMSI WC Scenario

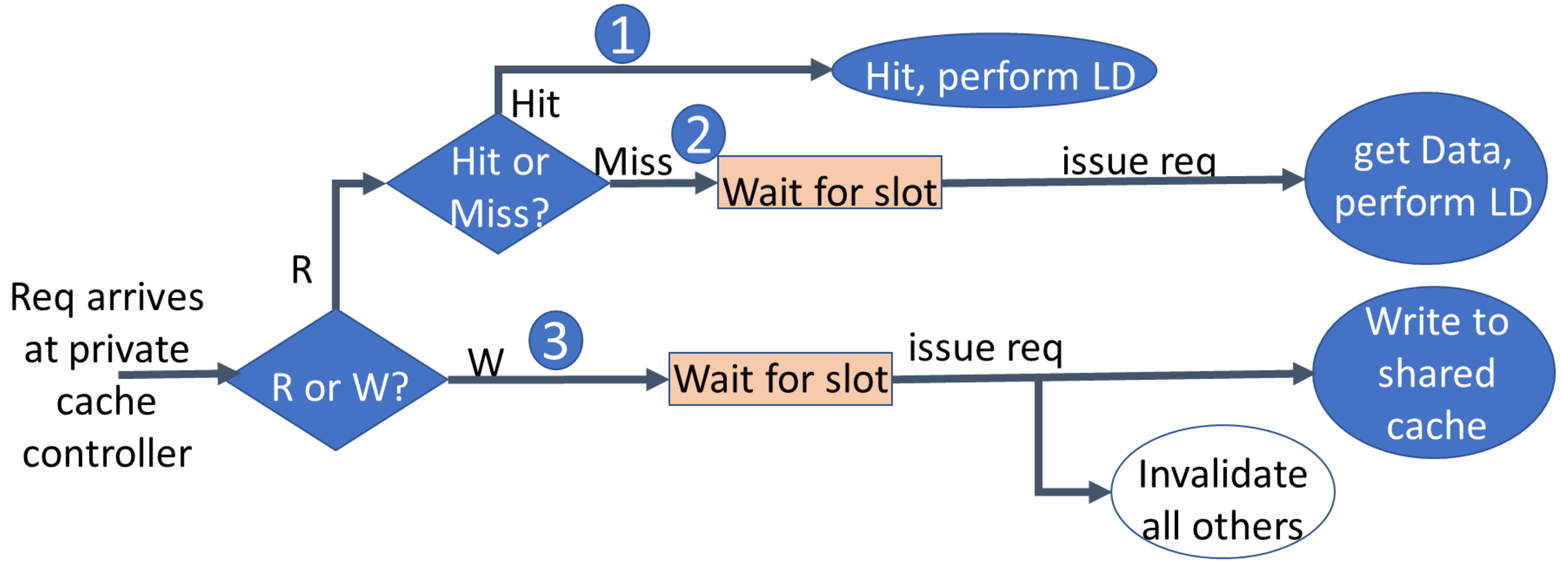
OBSERVATIONS



Another observation:
Writes represent small % of most applications

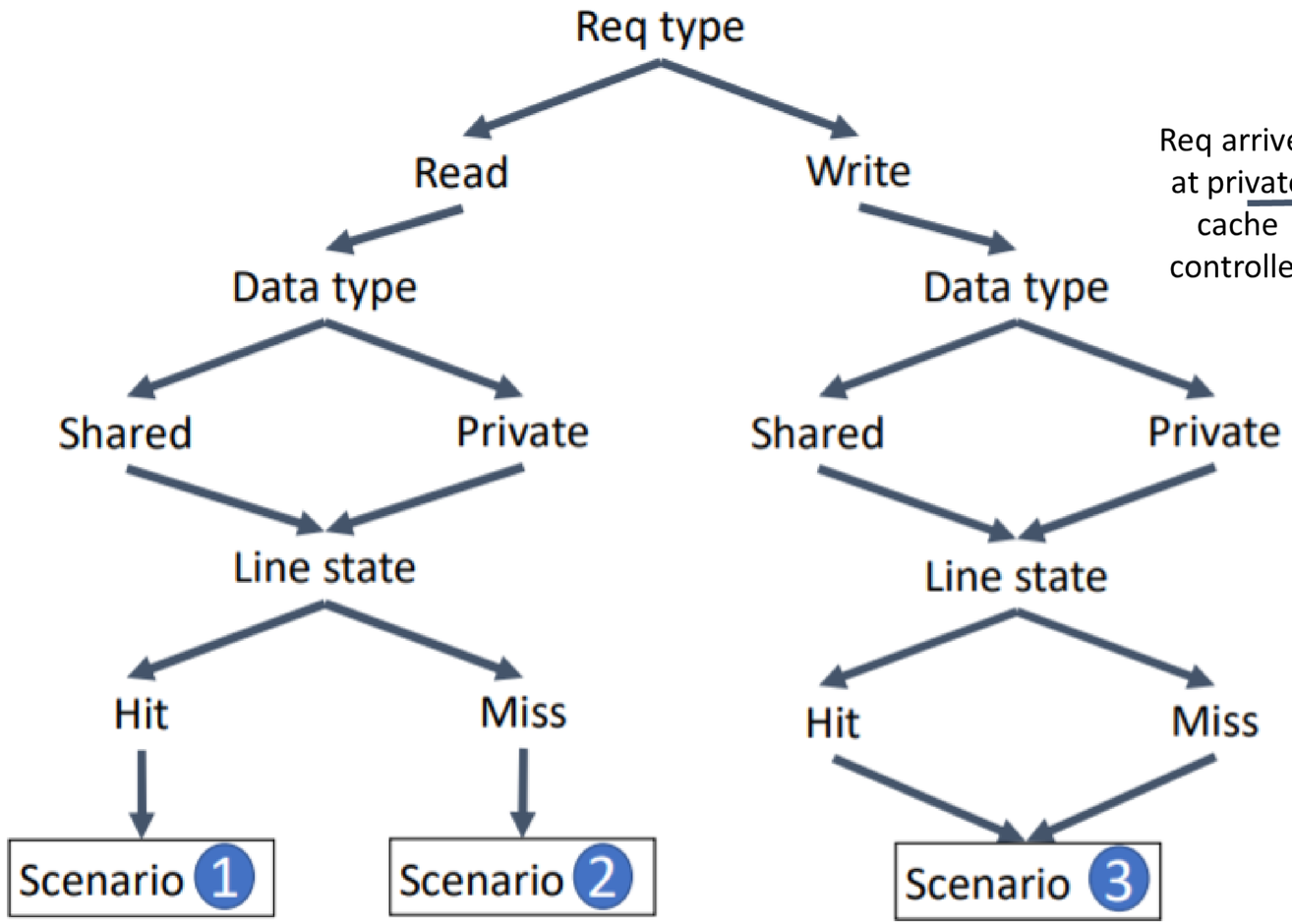
Solution: Eliminate this pathological scenario by design

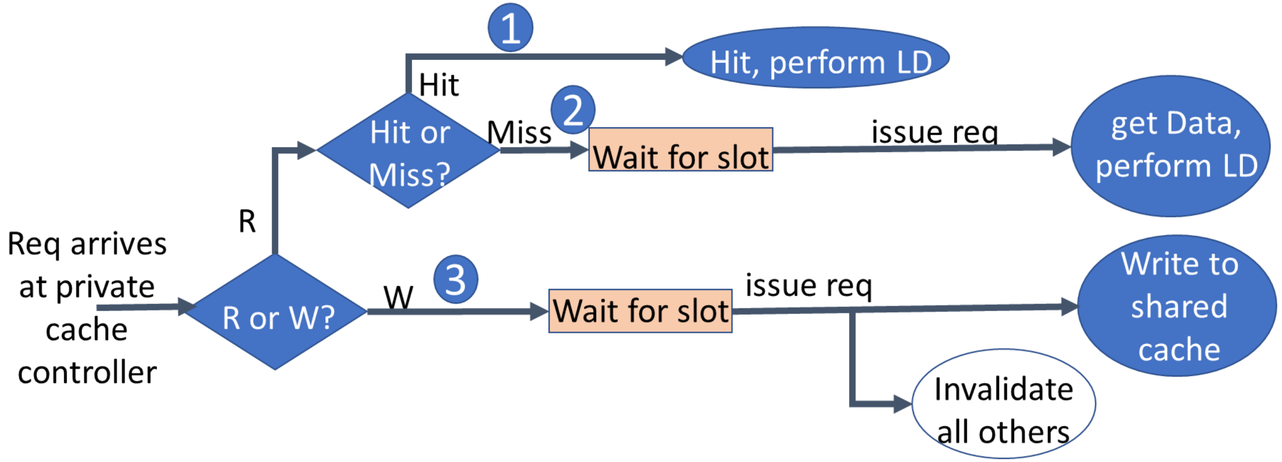
How? → No modified data in private caches



DISCO: Discriminative Coherence

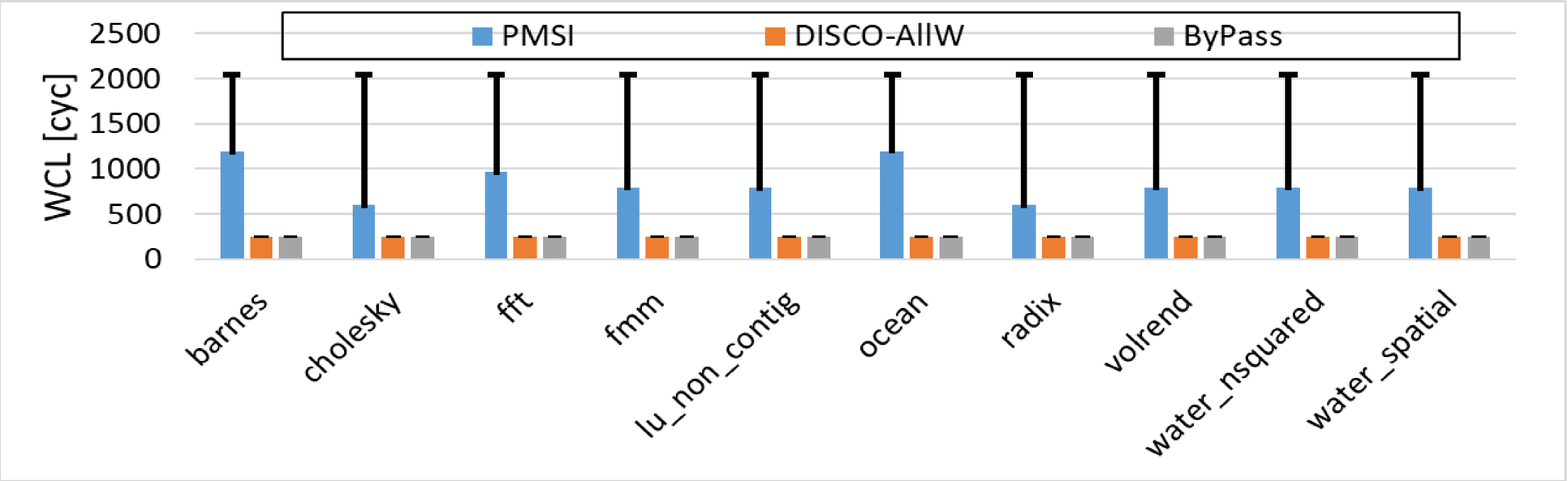
DISCO





DISCO: Discriminative Coherence

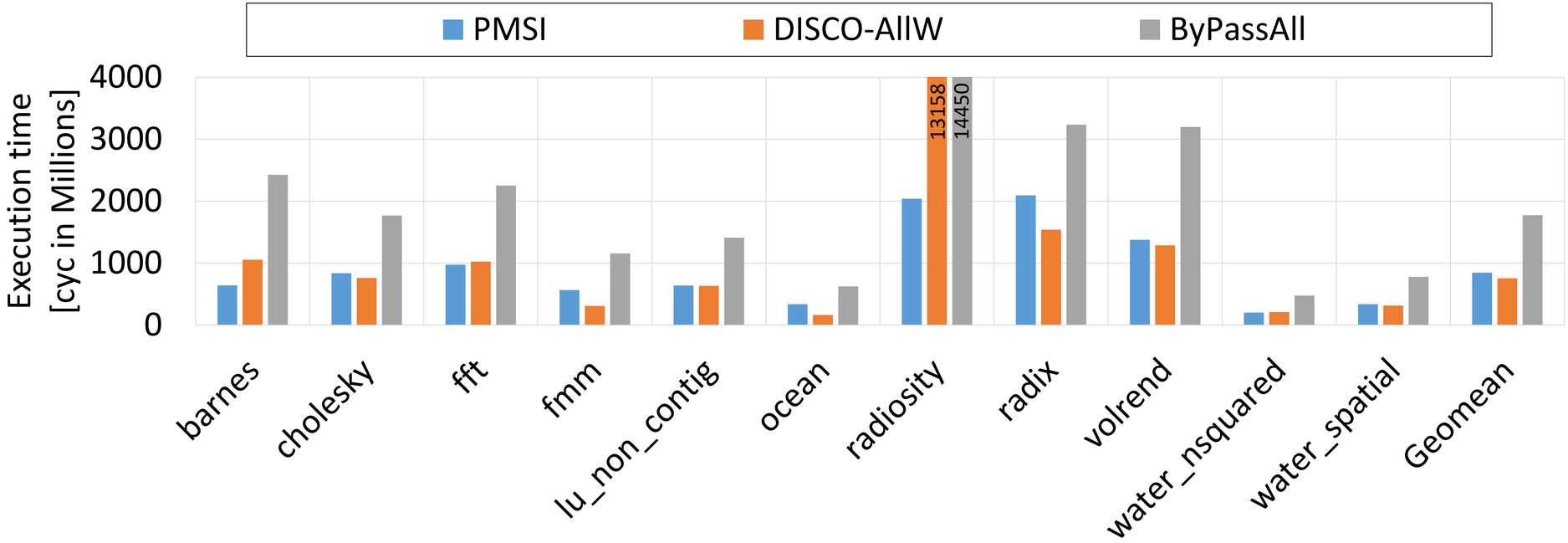
DISCO



Same WCL as ByPassing!

DISCO: Discriminative Coherence

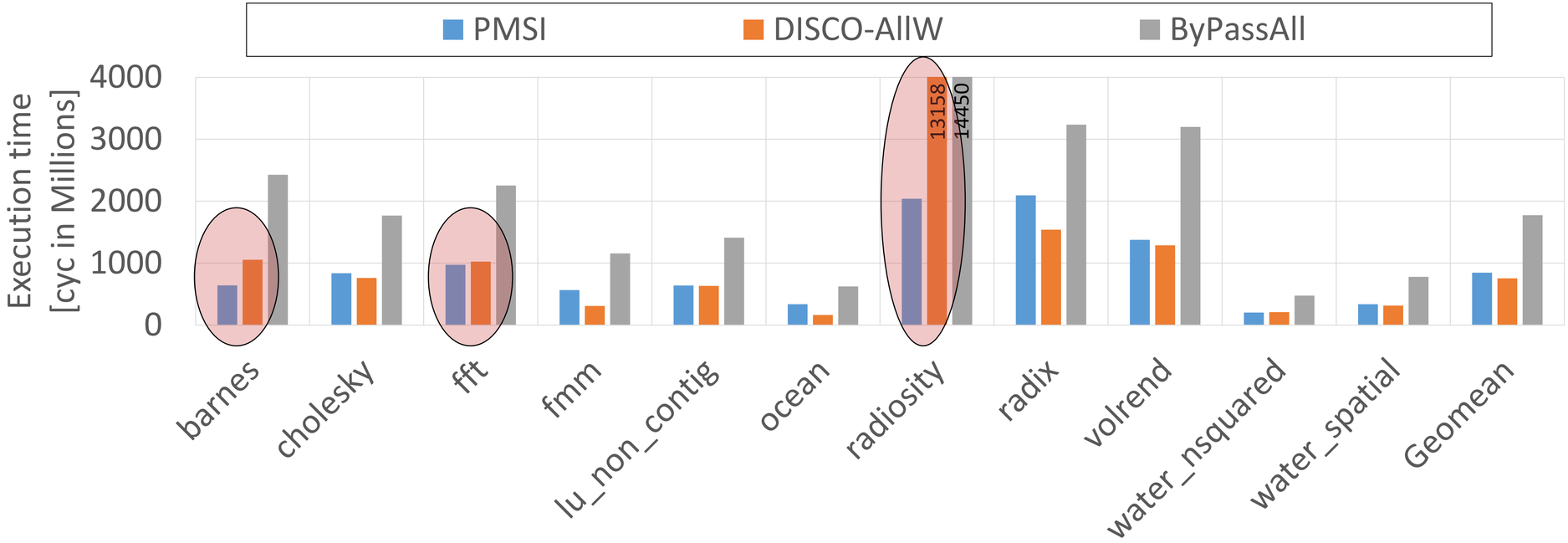
DISCO



Performance?: Better overall compared to PMSI!!

DISCO: Discriminative Coherence

DISCO



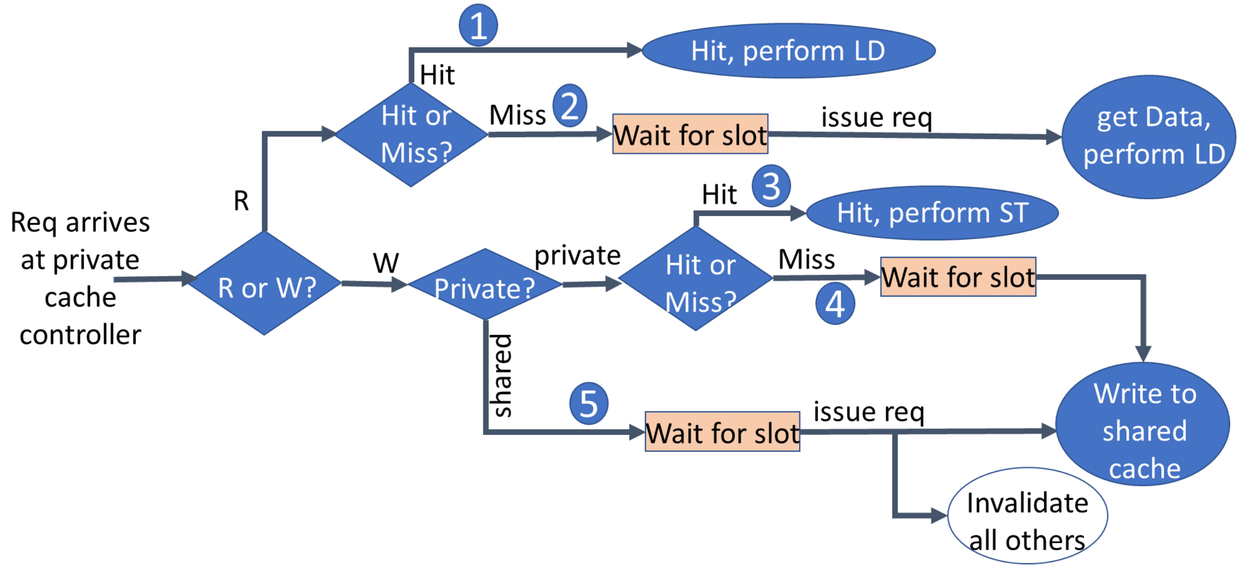
But worse for BMs that leverage write hits ☹

DISCO: Discriminative Coherence

DISCO

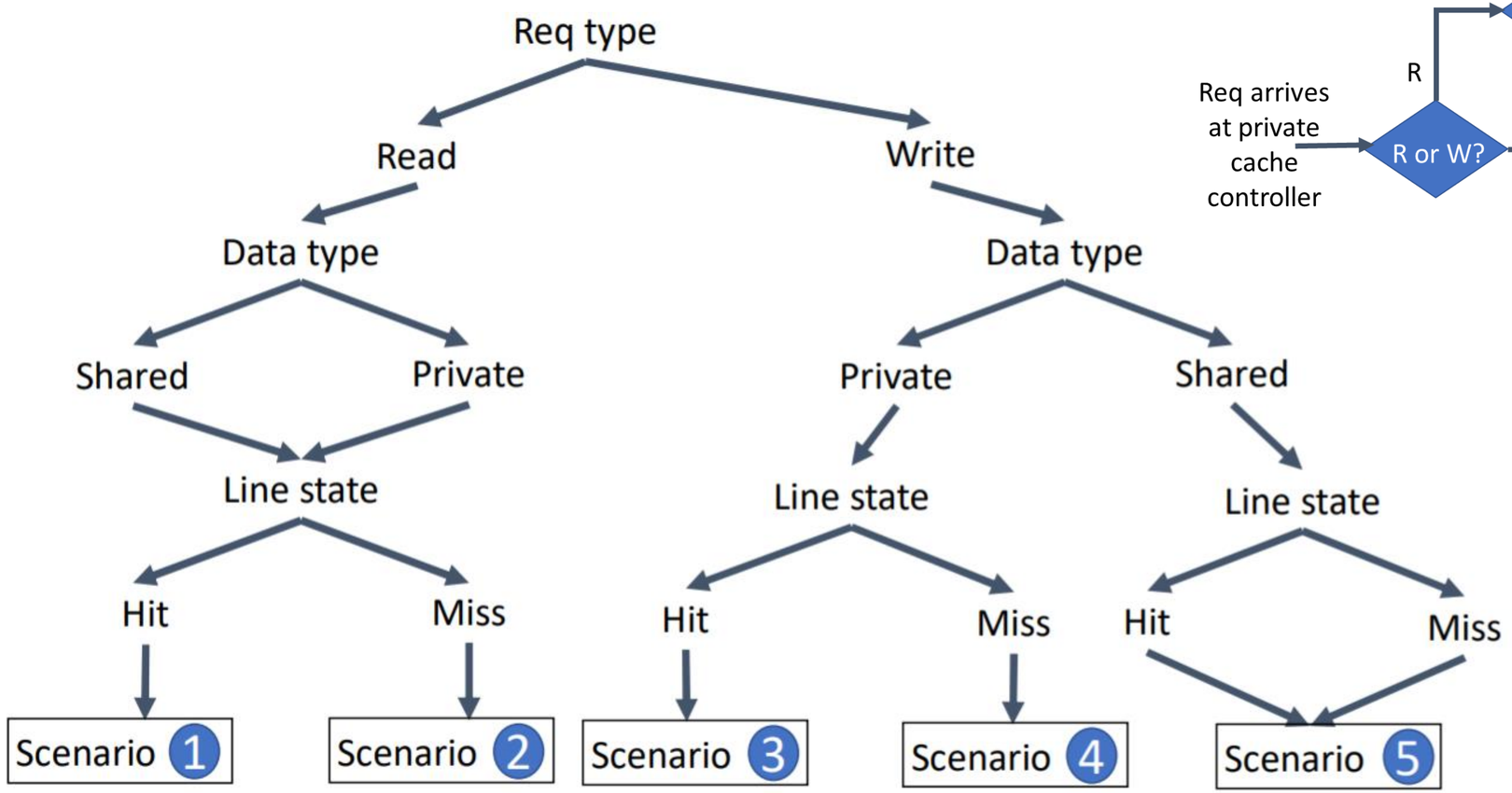
Solution: Eliminate this pathological scenario by design

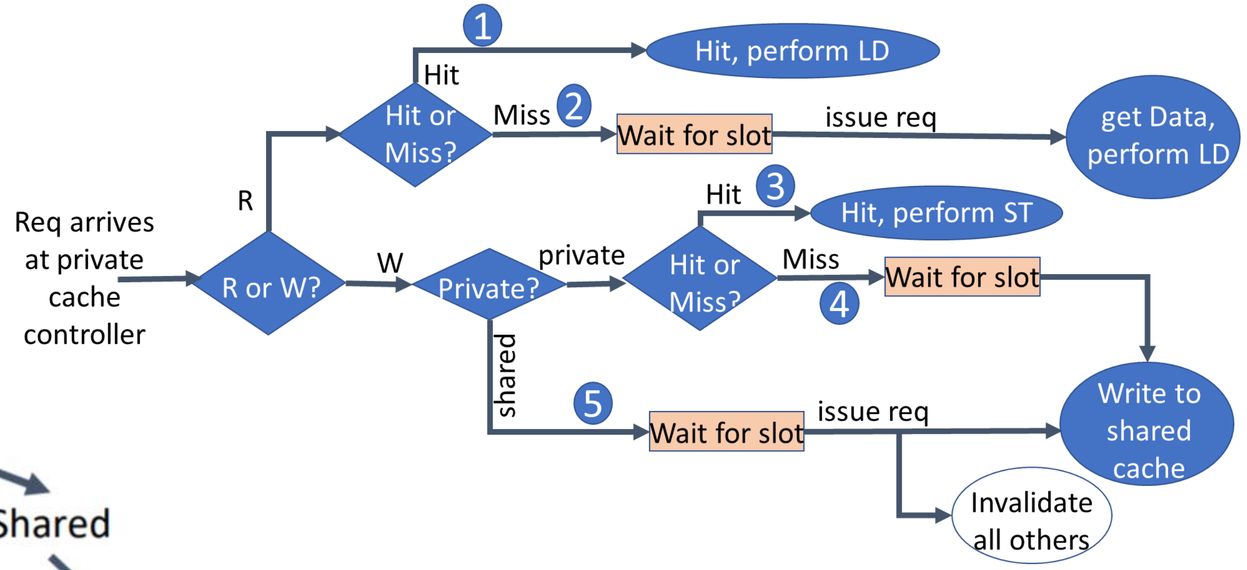
How? → No modified "*Shared*" data in private caches



DISCO-SharedW: Discriminative Coherence for only Shared Writes!

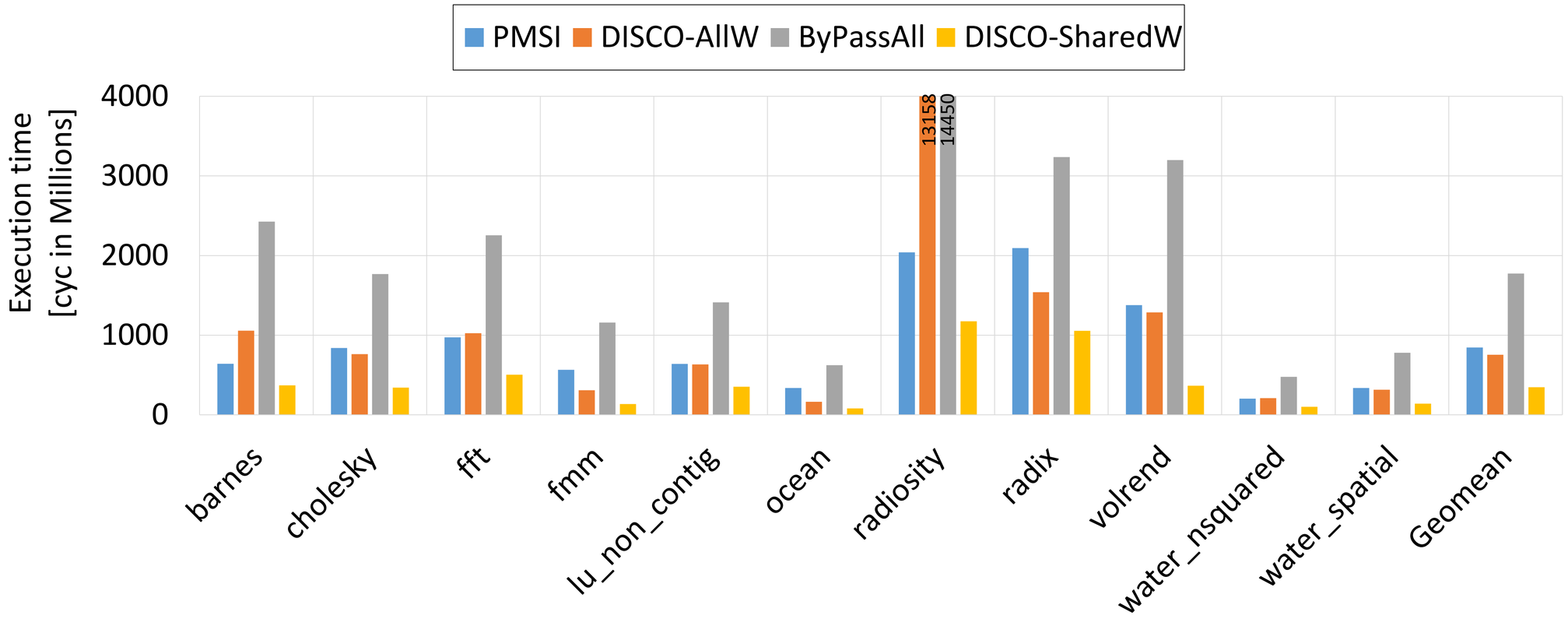
DISCO





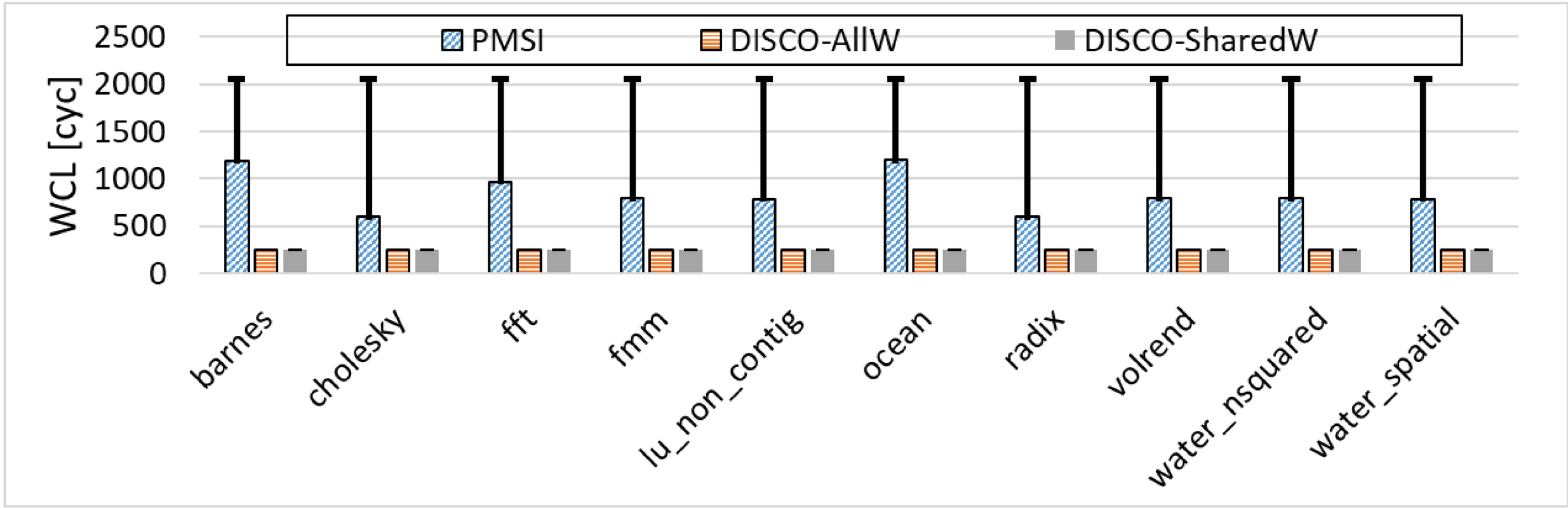
DISCO-SharedW: Discriminative Coherence for only Shared Writes!

DISCO



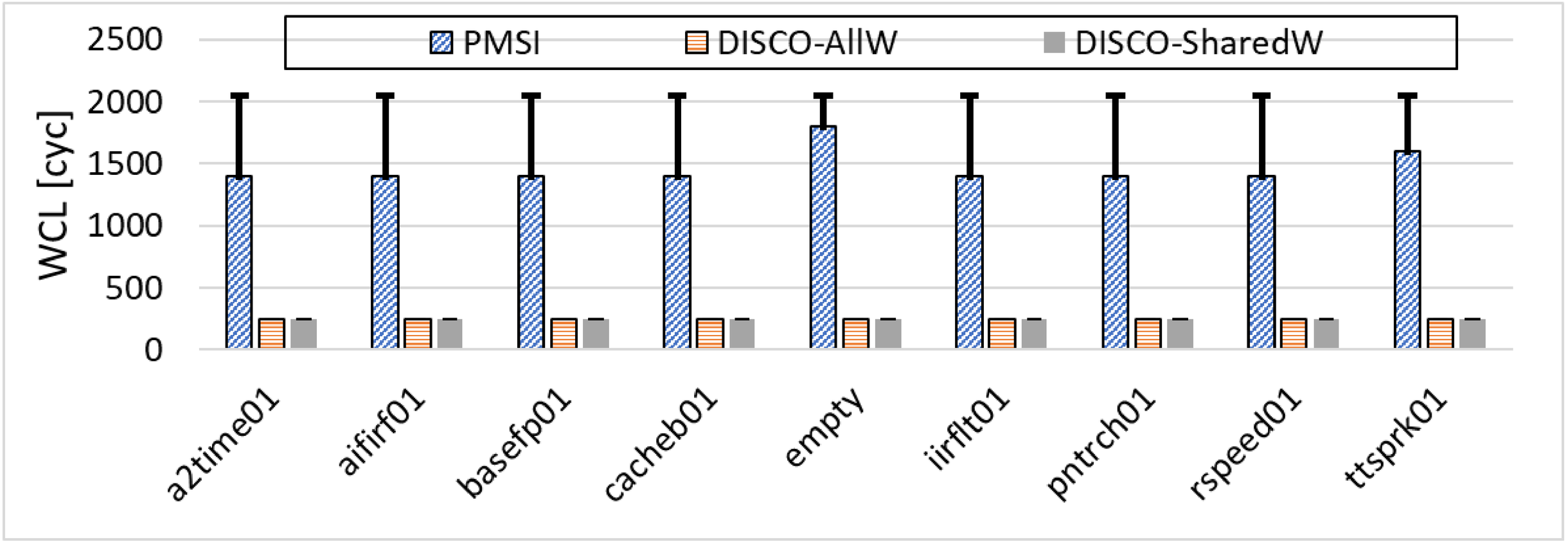
DISCO-SharedW: Discriminative Coherence for only Shared Writes!

RESULTS



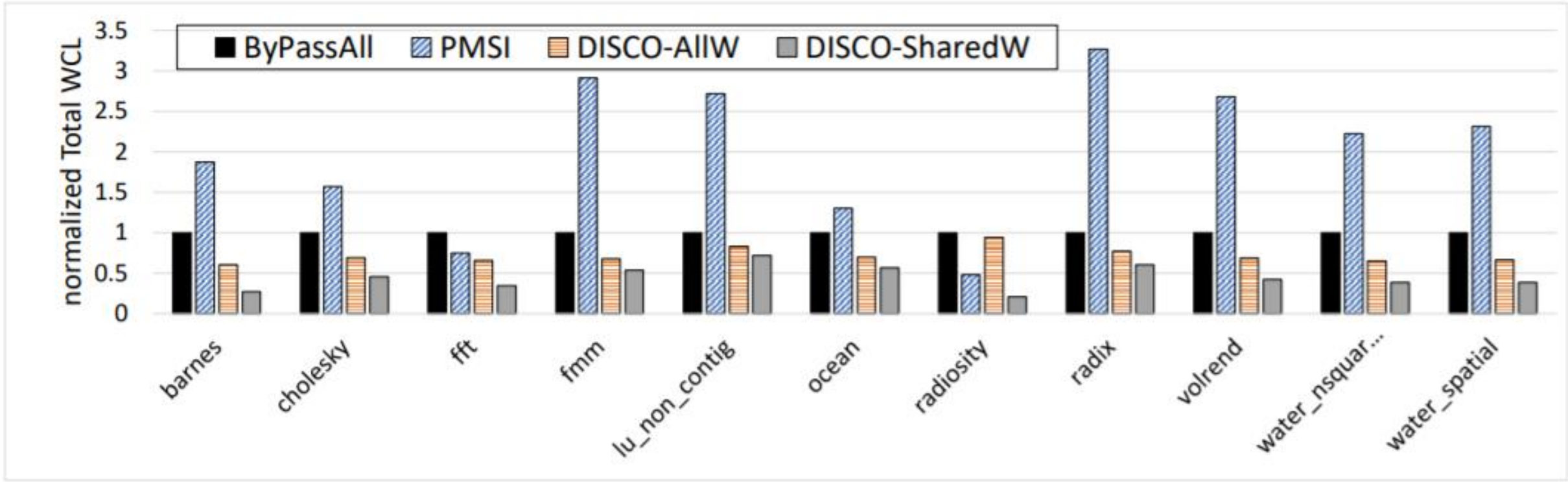
Per-Request WCL

RESULTS



Per-Request WCL

RESULTS



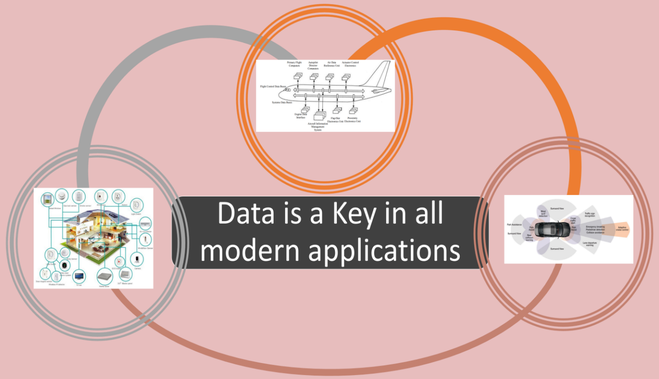
Total WCL

RESULTS

	Per-req WCL		Total WCL				Avg. Performance			
	PMSI	ByPass	PMSI		ByPass		PMSI		ByPass	
	analytical		Up to	Avg.	Up to	Avg.	Up to	Avg.	Up to	Avg.
DISCO-AllW	7.2x	Same	3.3x	2x	65%	42%	100%	12%	2.8x	1.5x
DISCO-SharedW	7.2x	Same	6x	3.5x	3.8x	1.5x	3.2x	1.6x	11.4x	5.3x

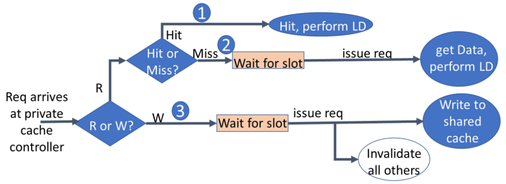
DISCO-SharedW: Discriminative Coherence for only Shared Writes!

RESULTS



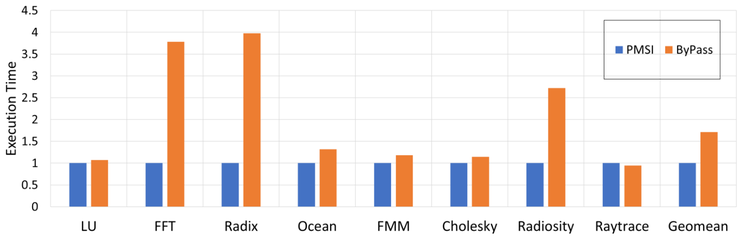
Solution: Eliminate this pathological scenario by design

How? → No modified data in private caches



DISCO: Discriminative Coherence

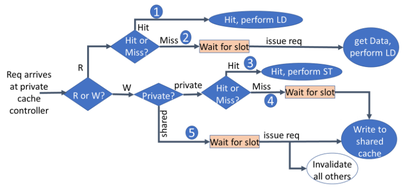
DISCO



Coherence provides up to 3X performance

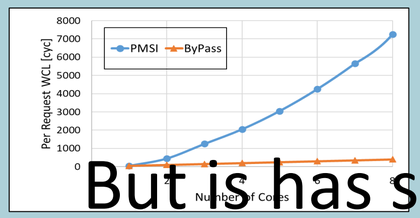
Solution: Eliminate this pathological scenario by design

How? → No modified “***Shared***” data in private caches

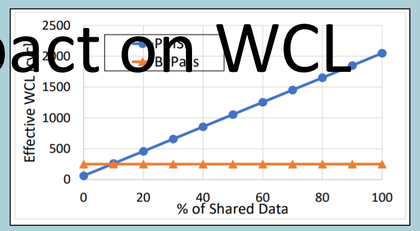


DISCO-SharedW: Discriminative Coherence for only Shared Writes!

DISCO



But is has significant impact on WCL



	Per-req WCL		Total WCL				Avg. Performance			
	PMSI	ByPass	PMSI		ByPass		PMSI		ByPass	
	analytical		Up to	Avg.	Up to	Avg.	Up to	Avg.	Up to	Avg.
DISCO-AllW	7.2x	Same	3.3x	2x	65%	42%	100%	12%	2.8x	1.5x
DISCO-SharedW	7.2x	Same	6x	3.5x	3.8x	1.5x	3.2x	1.6x	11.4x	5.3x

mohamed.Hassan@mcmaster.ca

C0	C1		C0	C1	C2	C0	C1	C2

	C2	C1	C0	C1	C0

C0 1-50		

C0		C2	C0	C1	C2	C0	C1	C2

			C1	C2	C0

C0 1-50		

- PISCOT:
  If (current core “TDM slot” has a pending request)
  - SCHEDULE IT
- MSI:
  If (current core “FCFS order” has a pending request)
  - SCHEDULE IT

- UNIFIED:
  - If (current core “FCFS” has something)
    - If (No request is being serviced)
      - SCHEDULE IT
- OOO solution: ➔ superset of PISCOT
  - If (current core “FCFS” has something)
    - If (No request is being serviced **From same core**)
      - SCHEDULE IT

Results

- [**8 Pending Requests**] as from OOO
- WCL (both SPLASH and EEMBC):
  - Analytical and experimental: PISCOT-C2C and PISCOT-nC2C (both IO and OOO)
  - MSI (SPLIT or Unified) has a bad bound (OOO)
  - PMSI WCL (Use current numbers)
- Average Case (ONLY splash):
  - PISCOT C2C and PISCOT-nC2C and MSI split and MSI unified (both IO and OOO)
    - Execution time
    - Average Latency and BW