Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance

Rachata Ausavarungnirun

Saugata Ghose, Onur Kayiran, Gabriel H. Loh
Chita Das, Mahmut Kandemir, Onur Mutlu

SAFARI

Carnegie Mellon

AMD

PENNSTATE
1855

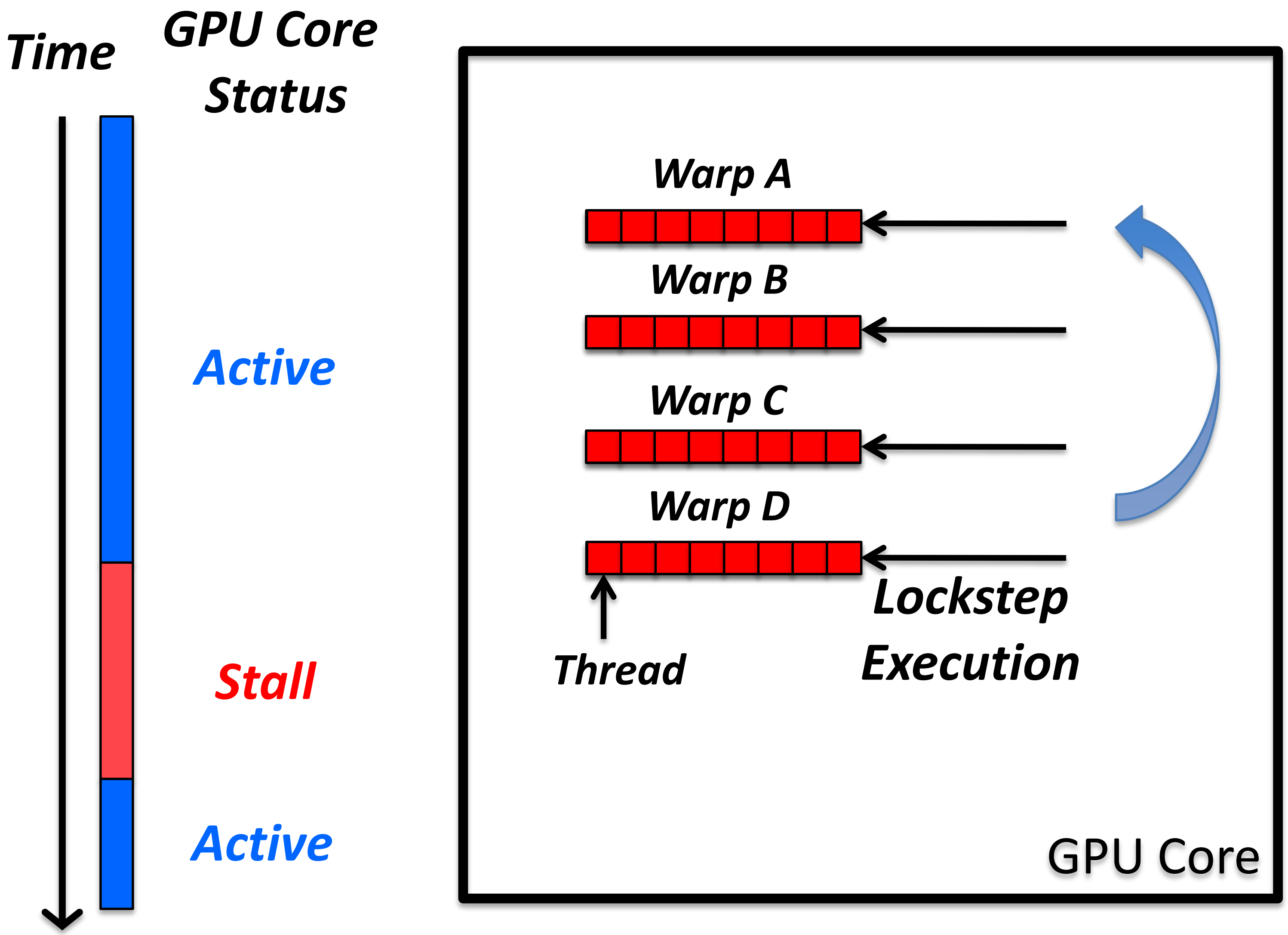
Overview of This Talk

- **Problem: memory divergence**
  - Threads execute in lockstep, but not all threads hit in the cache
  - A single long latency thread can stall an entire warp
- **Key Observations:**
  - Memory divergence characteristic differs across warps
  - Some warps mostly hit in the cache, some mostly miss
  - Divergence characteristic is stable over time
  - L2 queuing exacerbates memory divergence problem
- **Our Solution: Me**mory **Div**ergence **C**orrection
  - Uses **cache bypassing**, **cache insertion** and **memory scheduling** to **prioritize mostly-hit warps** and **deprioritize mostly-miss warps**
- **Key Results:**
  - **21.8% better performance and 20.1% better energy efficiency** compared to state-of-the-art caching policy on GPU

Outline

- Background
- Key Observations
- **Me**mory **Di**vergence **C**orrection (MeDiC)
- Results

Latency Hiding in GPGPU Execution



Time

GPU Core Status

Active

Stall

Active

Warp A

Warp B

Warp C

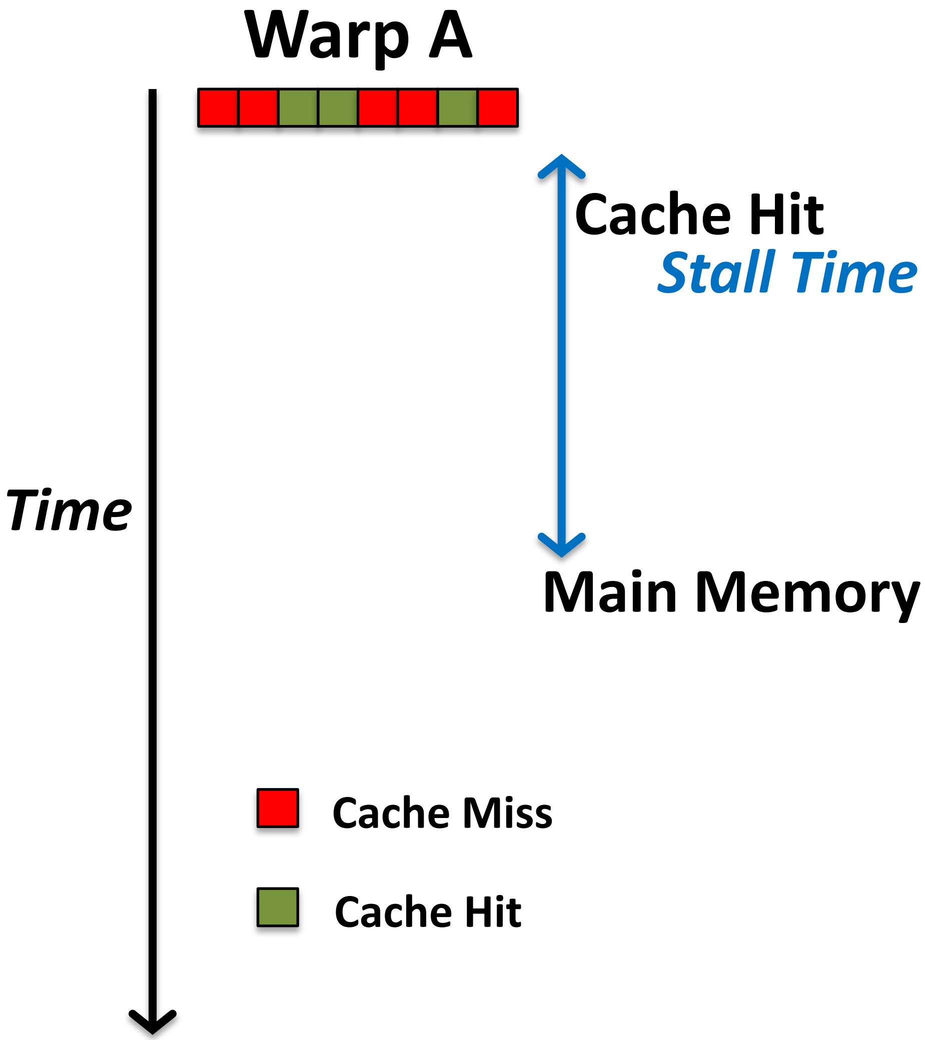
Warp D

Thread

Lockstep Execution

GPU Core

Problem: Memory Divergence



Warp A

Cache Hit

Stall Time

Time

Main Memory

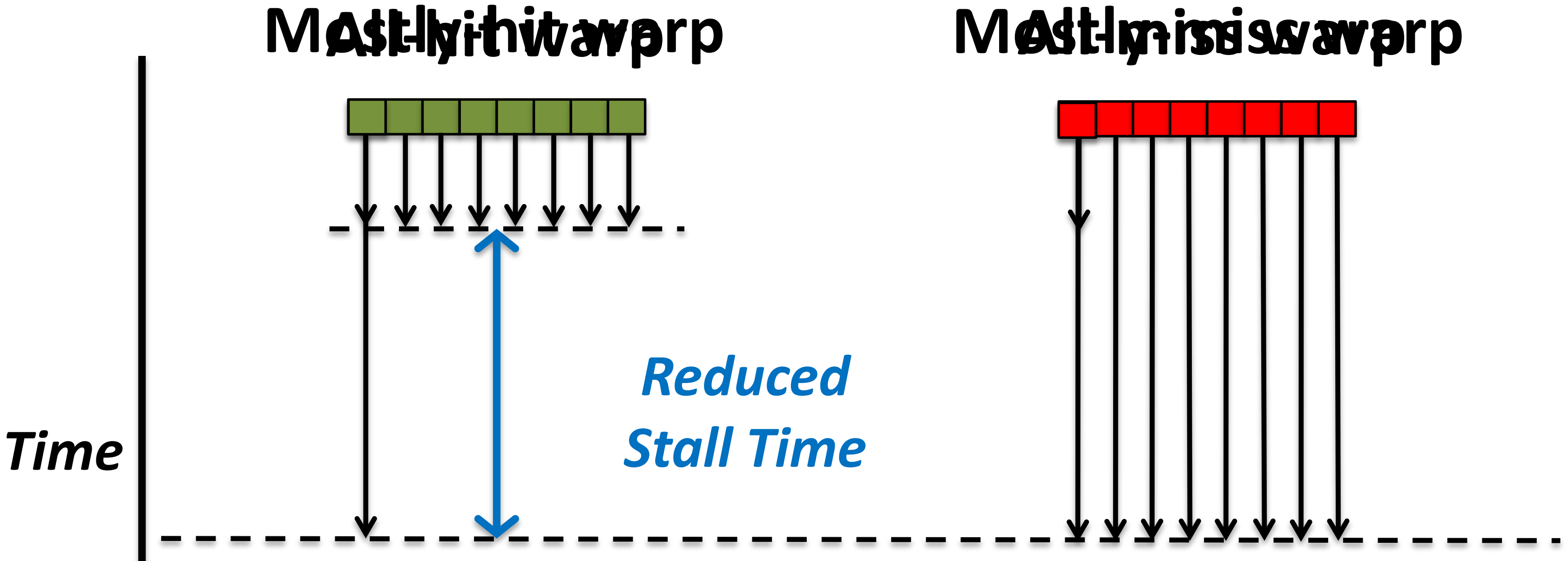
🟥 **Cache Miss**

🟩 **Cache Hit**

Outline

- Background
- **Key Observations**
- **Me**mory **Di**vergence **C**orrection (MeDiC)
- Results

Observation 1: Divergence Heterogeneity



🟥 Cache Miss

🟩 Cache Hit

Key Idea:

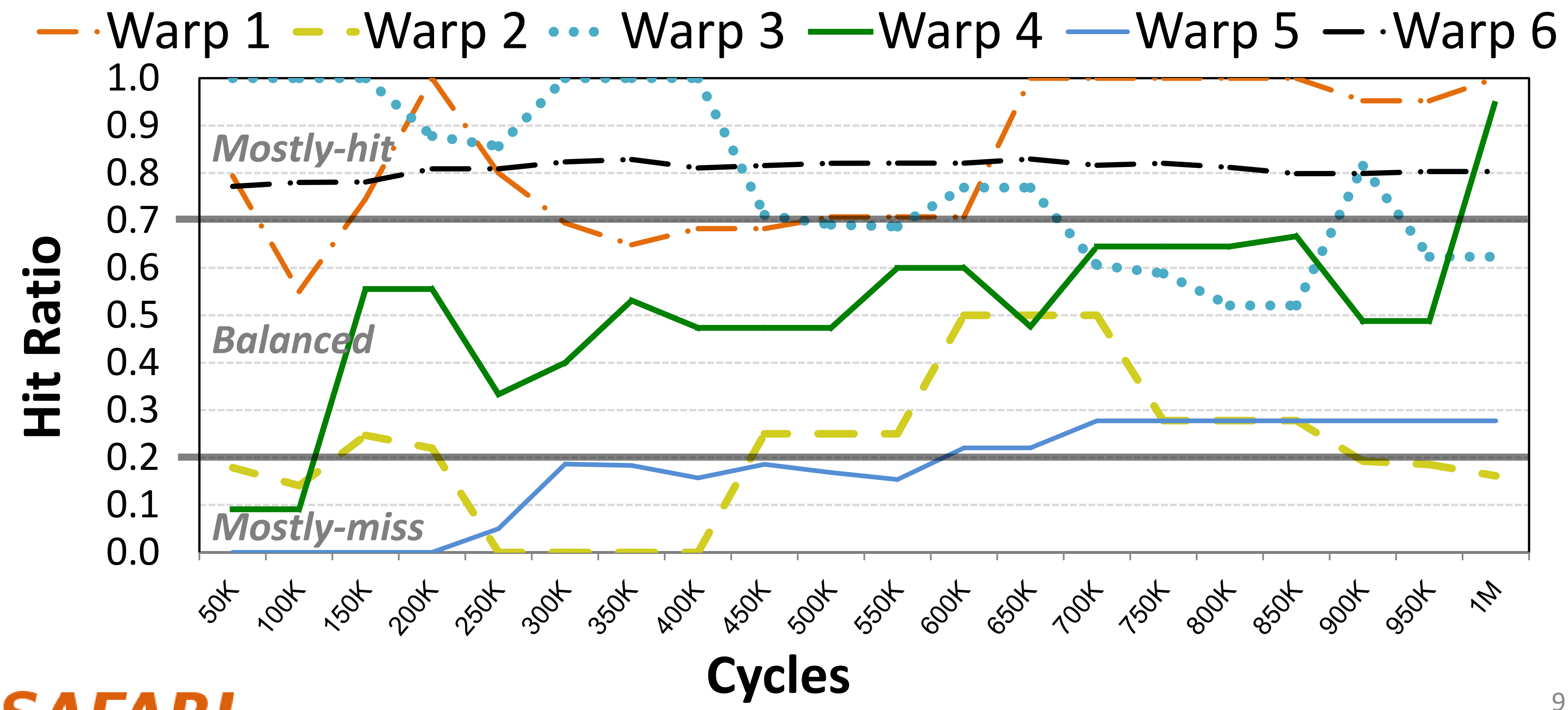
- Convert **mostly-hit** warps to **all-hit** warps
- Convert **mostly-miss** warps to **all-miss** warps

Observation 2: Stable Divergence Char.

- Warp retains its hit ratio during a program phase
  - Hit ratio → number of hits / number of access

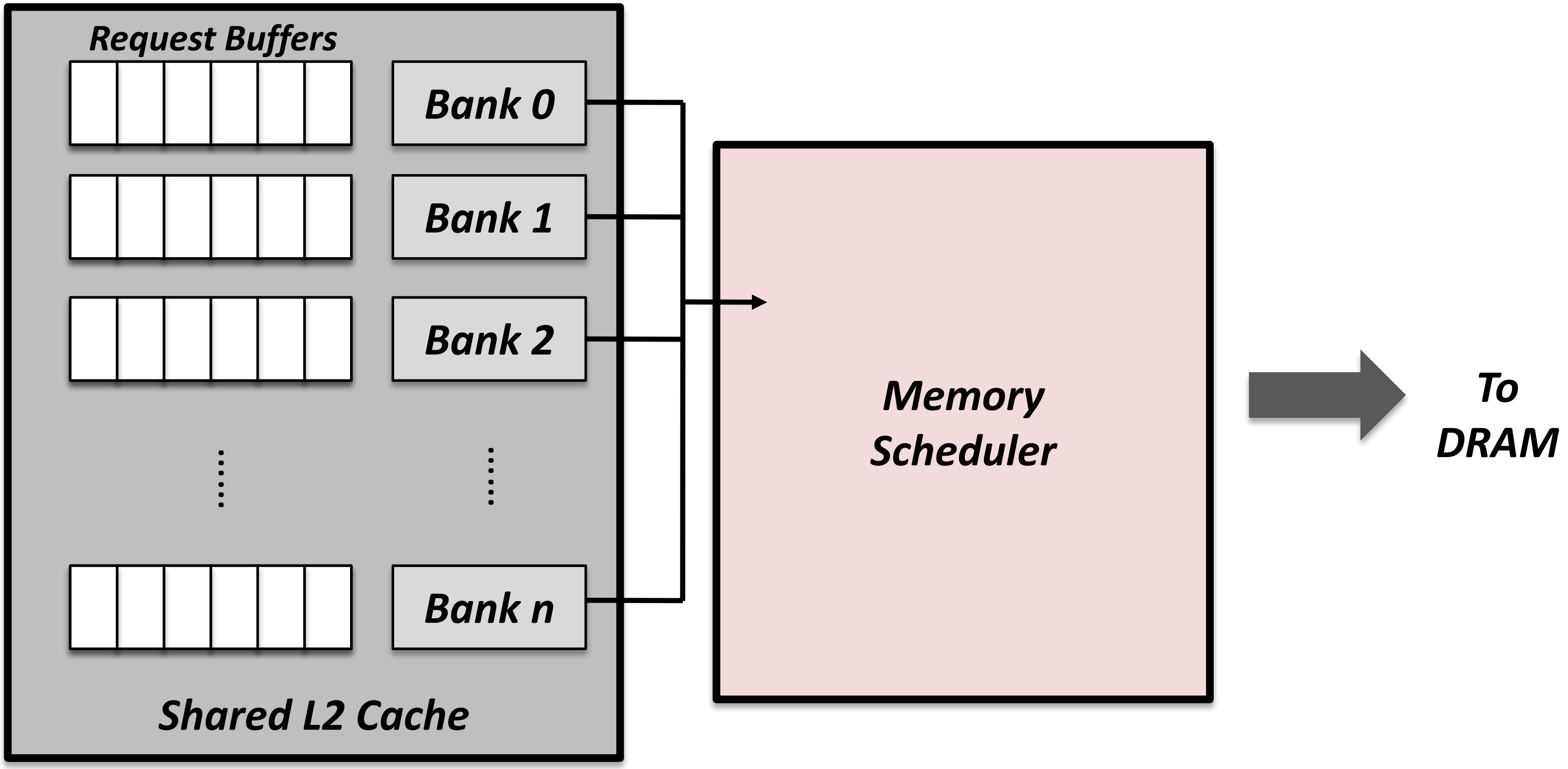
Observation 2: Stable Divergence Char.

- Warp retains its hit ratio during a program phase



SAFARI

Observation 3: Queuing at L2 Banks



45% of requests stall 20+ cycles at the L2 queue

Long queuing delays exacerbate the effect of memory divergence

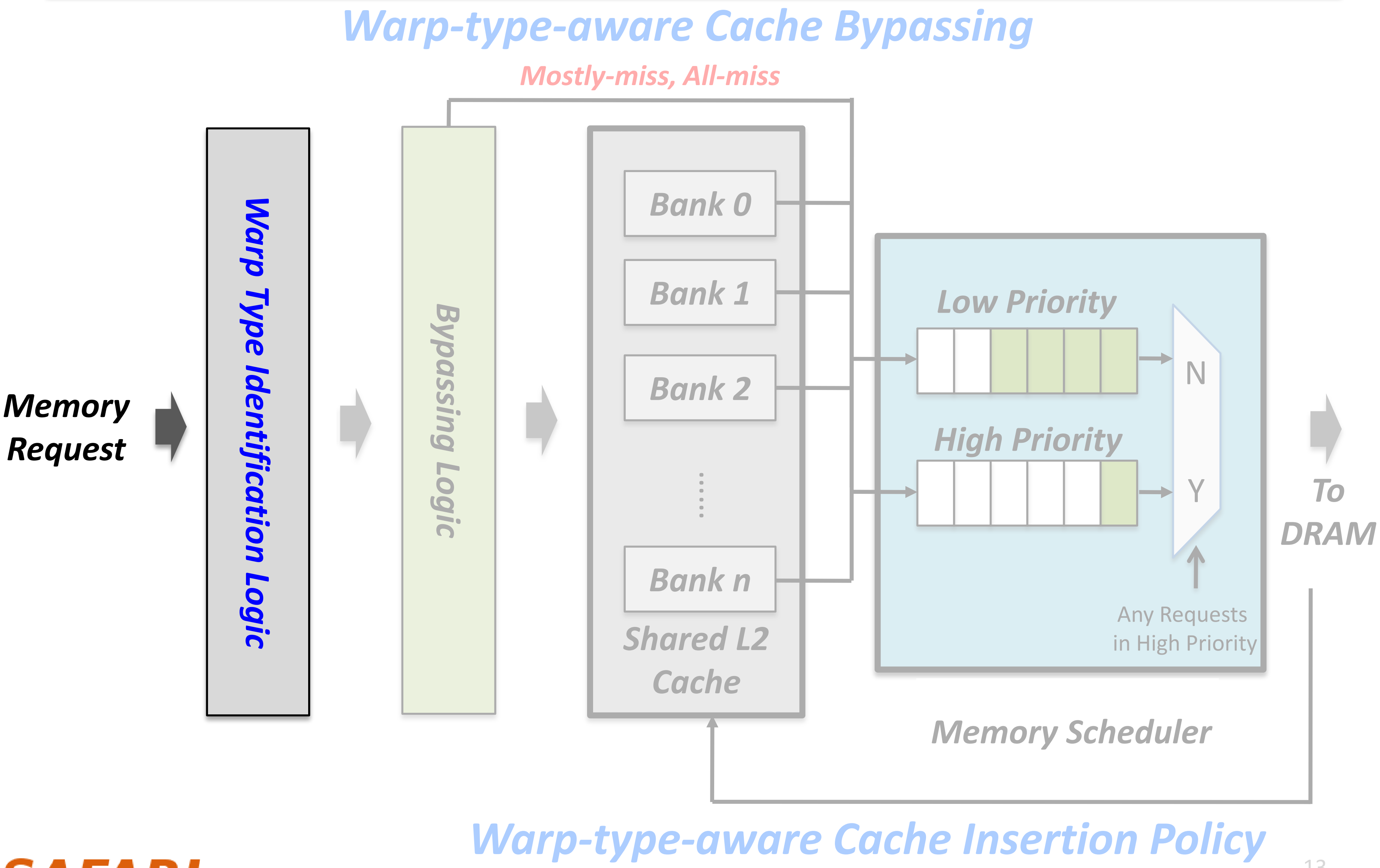
Outline

- Background
- Key Observations
- **Memory Divergence Correction (MeDiC)**
- Results

Our Solution: MeDiC

- Key Ideas:
  - Convert **mostly-hit** warps to **all-hit** warps
  - Convert **mostly-miss** warps to **all-miss** warps
  - Reduce L2 queuing latency
  - Prioritize mostly-hit warps at the memory
  - Maintain memory bandwidth

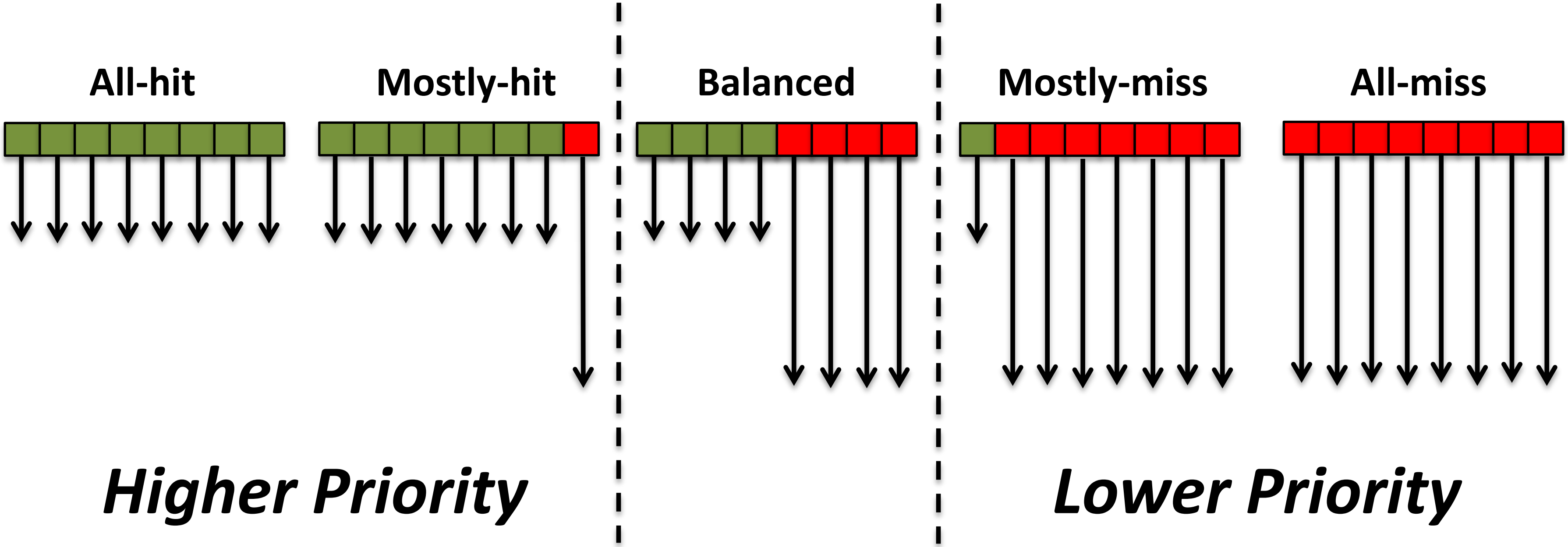
Memory Divergence Correction



SAFARI

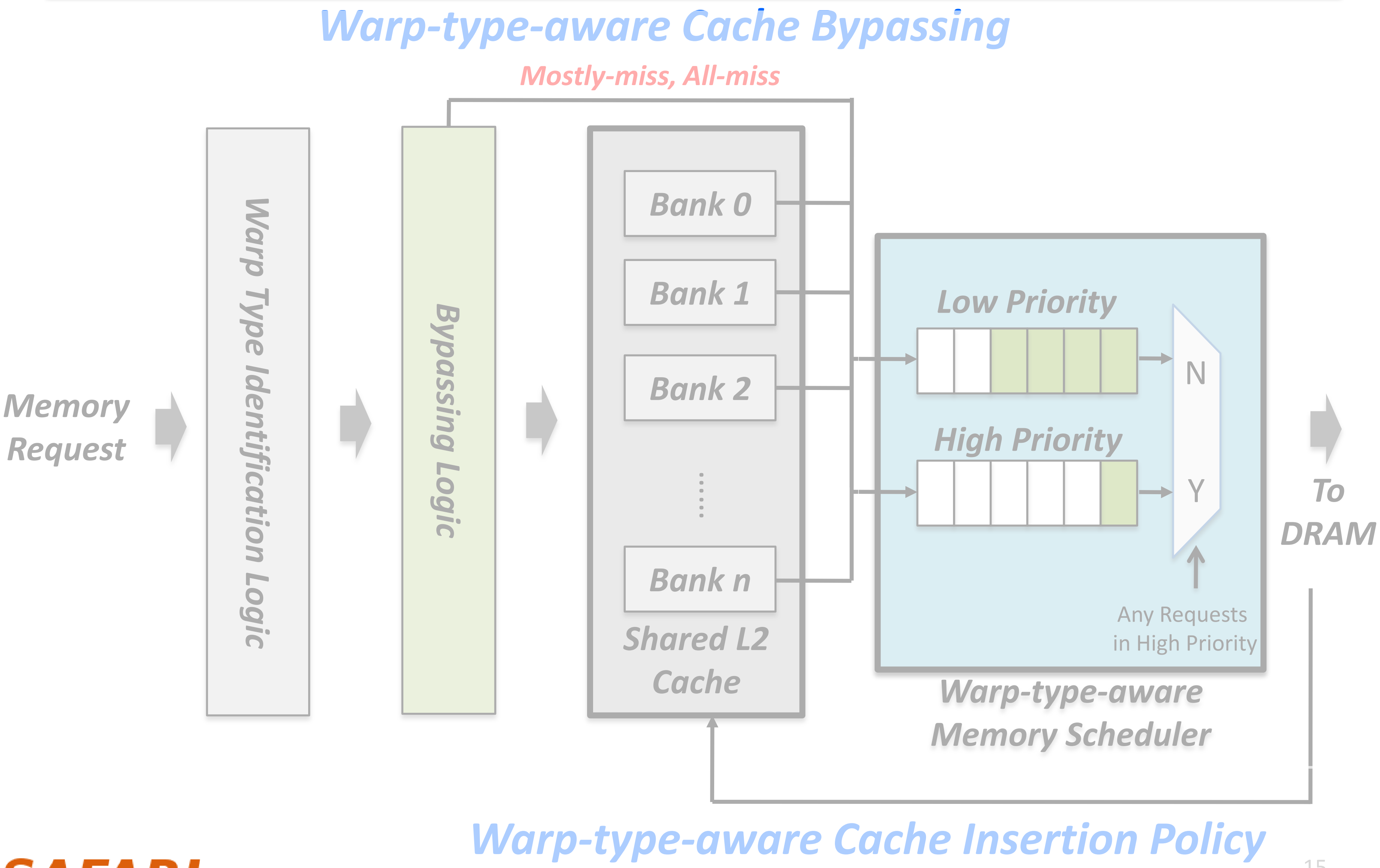
Mechanism to Identify Warp Type

- Profile hit ratio for each warp
- Group warp into one of five categories



- Periodically reset warp-type

Warp-type-aware Cache Bypassing



Warp-type-aware Cache Bypassing

Mostly-miss, All-miss

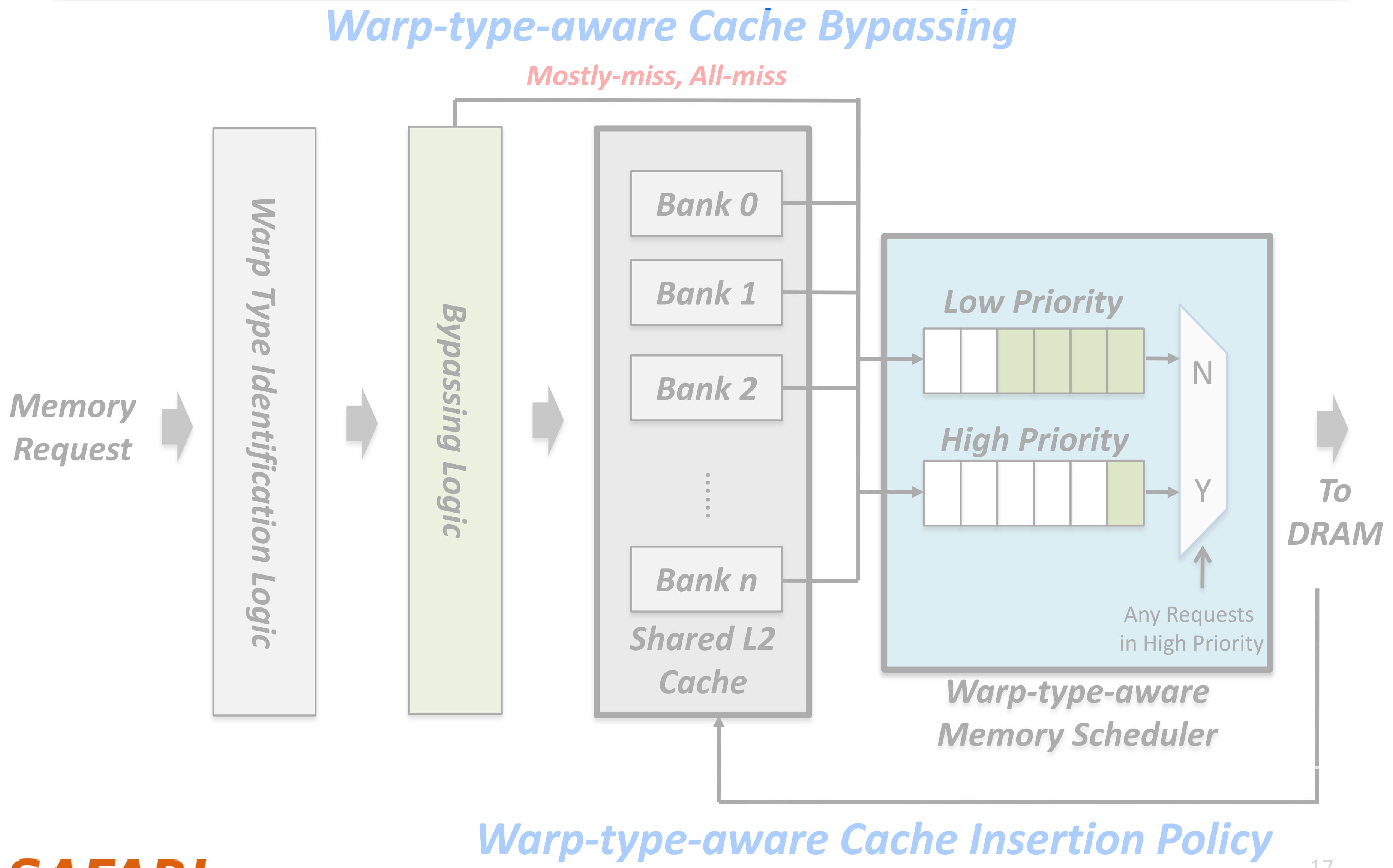
Warp-type-aware Cache Insertion Policy

SAFARI

Warp-type-aware Cache Bypassing

- **Goal:**
  - Convert **mostly-hit** warps to **all-hit** warps
  - Convert **mostly-miss** warps to **all-miss** warps
- **Our Solution:**
  - **All-miss and mostly-miss** warps → **Bypass L2**
  - Adjust how we identify warps to **maintain miss rate**
- **Key Benefits:**
  - **More all-hit** warps
  - Reduce queuing latency for all warps

Warp-type-aware Cache Insertion

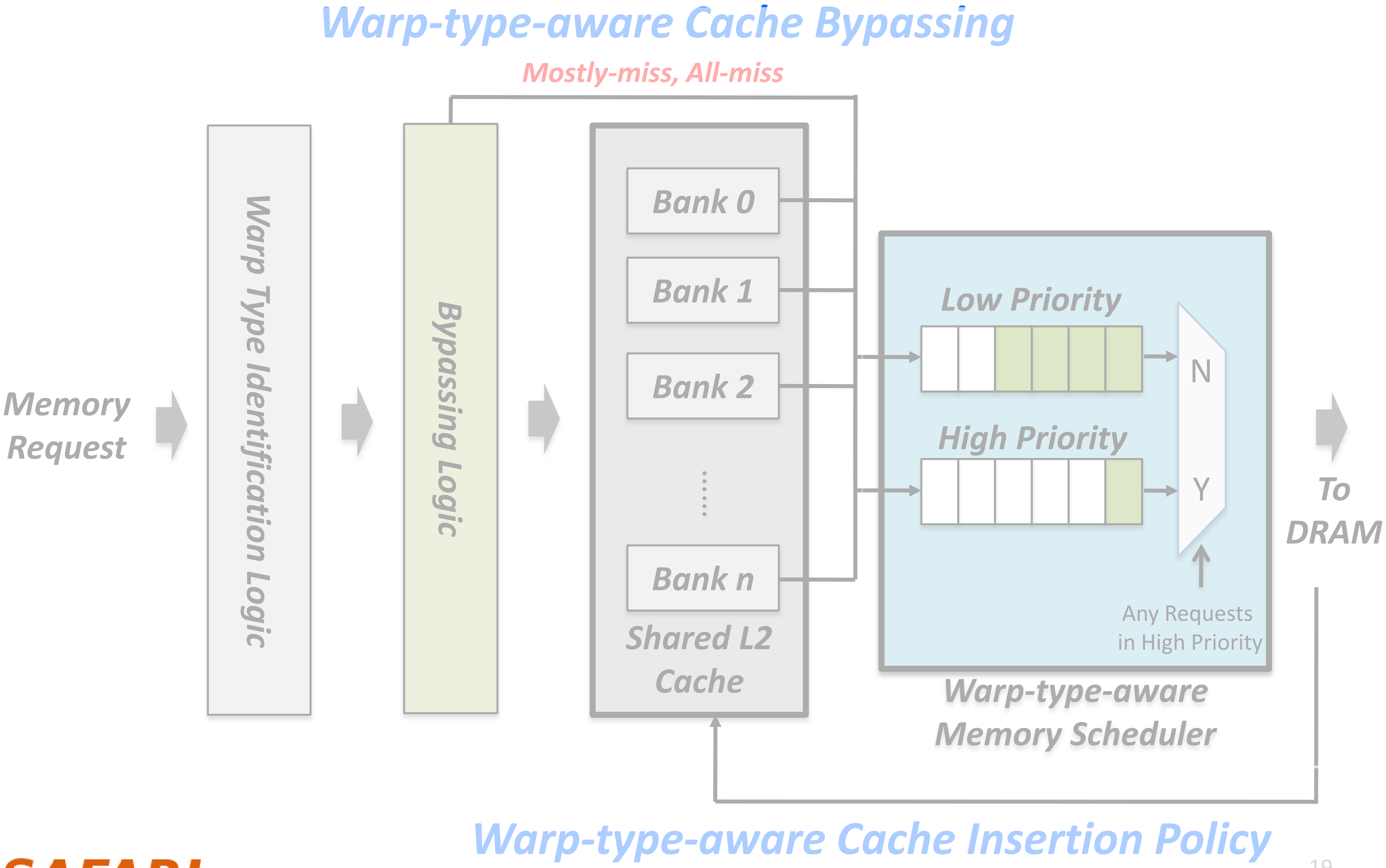

Warp-type-aware Cache Bypassing
Mostly-miss, All-miss
Memory Request
Warp Type Identification Logic
Bypassing Logic
Bank 0
Bank 1
Bank 2
......
Bank n
Shared L2 Cache
Low Priority
High Priority
N
Y
Any Requests in High Priority
Warp-type-aware Memory Scheduler
To DRAM
Warp-type-aware Cache Insertion Policy

SAFARI

Warp-type-aware Cache Insertion

- **Goal:** Utilize the cache well
  - Prioritize mostly-hit warps
  - Maintain blocks with high reuse
- **Our Solution:**
  - **All-miss** and **mostly-miss** → Insert at **LRU**
  - **All-hit, mostly-hit** and balanced → Insert at **MRU**
- **Benefits:**
  - All-hit and mostly-hit are **less likely** to be evicted
  - **Heavily reused cache blocks** from mostly-miss are likely to **remain in the cache**

Warp-type-aware Memory Sched.


Warp-type-aware Cache Bypassing
Mostly-miss, All-miss
Memory Request
Warp Type Identification Logic
Bypassing Logic
Bank 0
Bank 1
Bank 2
......
Bank n
Shared L2 Cache
Low Priority
High Priority
N
Y
Any Requests in High Priority
Warp-type-aware Memory Scheduler
To DRAM
Warp-type-aware Cache Insertion Policy

Not All Blocks Can Be Cached

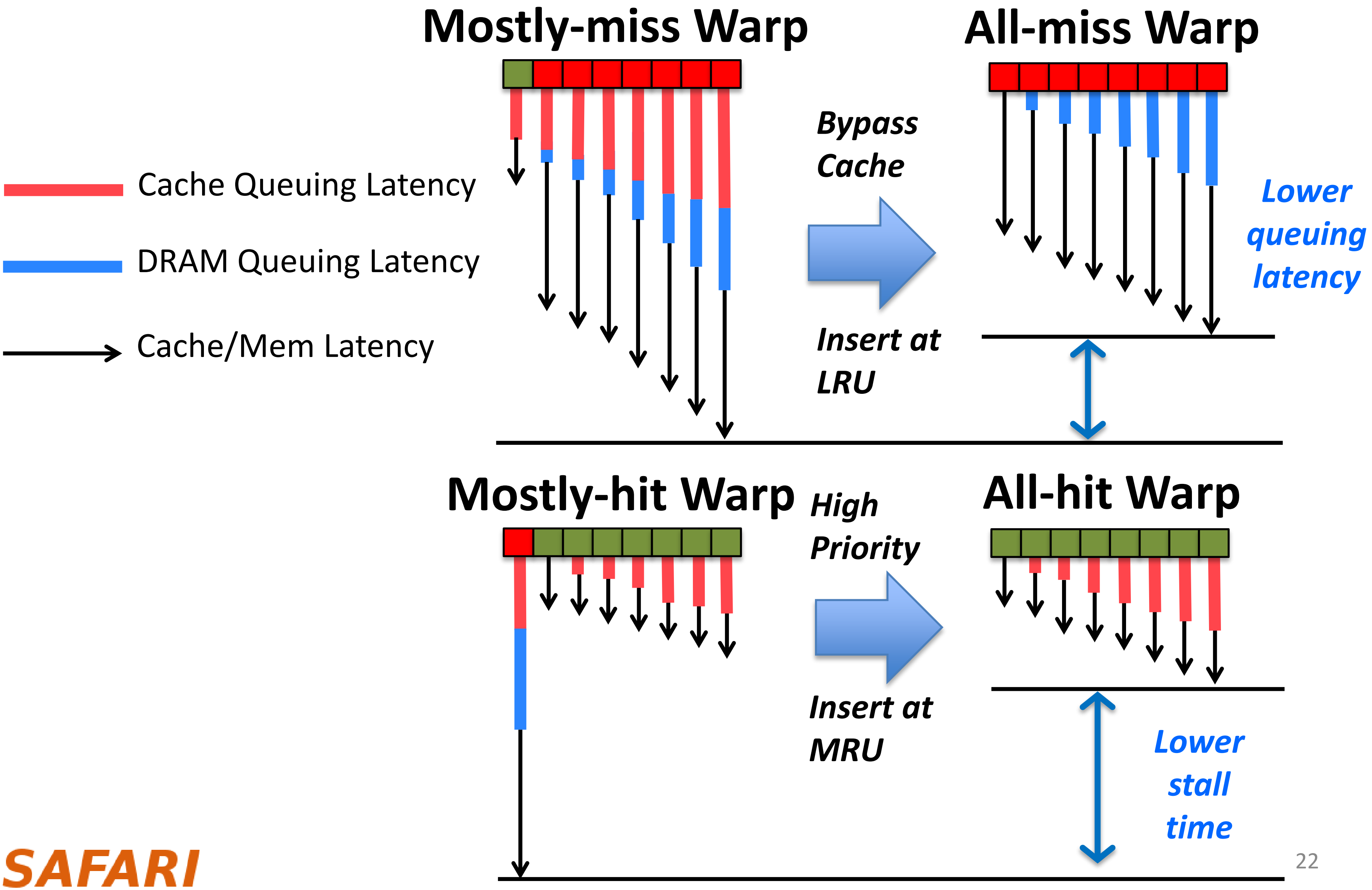
- Despite best efforts, accesses from mostly-hit warps **can still miss** in the cache
  - Compulsory misses
  - Cache thrashing

- **Solution:** Warp-type-aware memory scheduler

Warp-type-aware Memory Sched.

- **Goal:** **Prioritize mostly-hit** over mostly-miss
- **Mechanism:** Two memory request queues
  - **High-priority** → **all-hit** and **mostly-hit**
  - **Low-priority** → balanced, **mostly-miss** and **all-miss**
- **Benefits:**
  - **Mostly-hit** warps **stall less**

MeDiC: Example



Mostly-miss Warp

All-miss Warp

Cache Queuing Latency

DRAM Queuing Latency

Cache/Mem Latency

Bypass Cache

Insert at LRU

Lower queuing latency

Mostly-hit Warp

All-hit Warp

High Priority

Insert at MRU

Lower stall time

Outline

- Background
- Key Observations
- **Me**mory **Di**vergence **C**orrection (MeDiC)
- **Results**

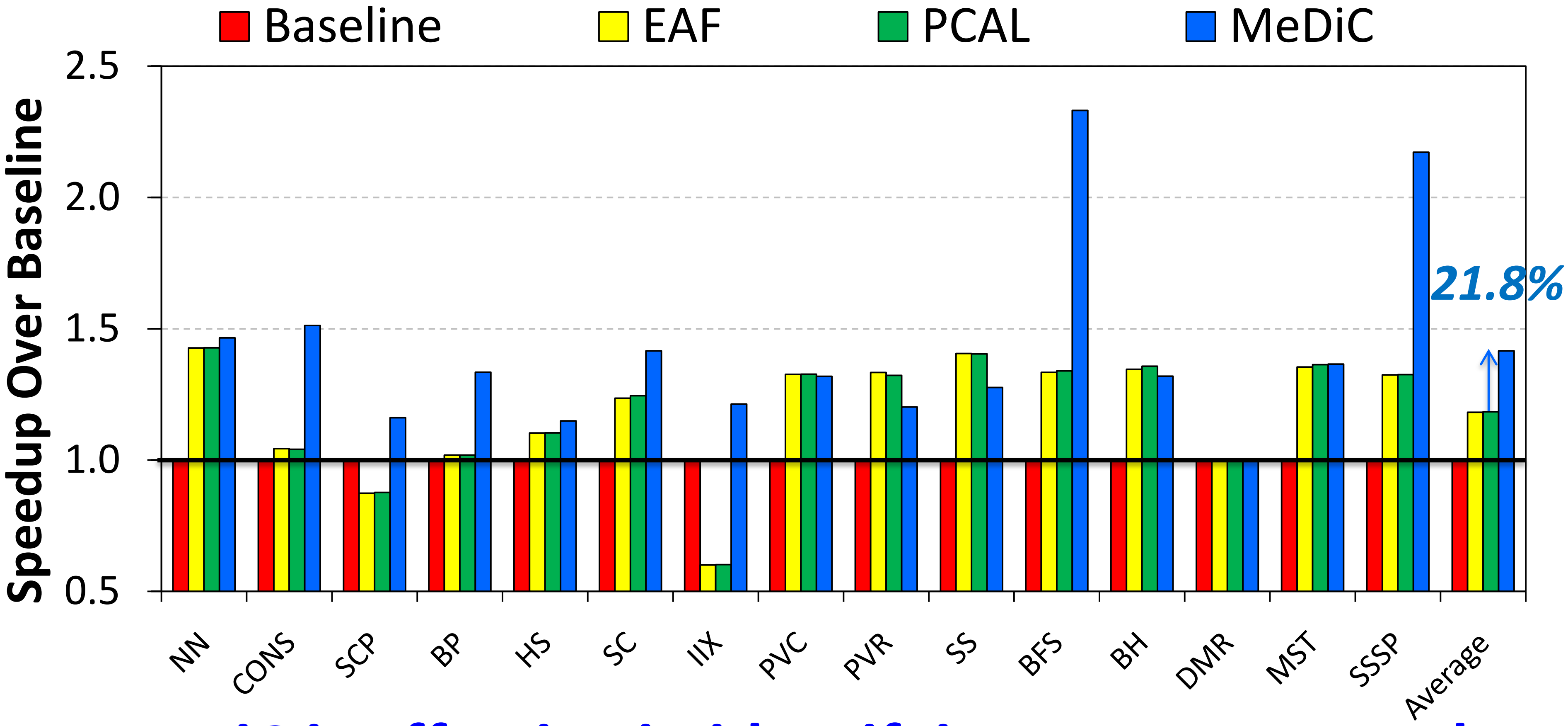
Methodology

- Modified GPGPU-Sim modeling GTX480
  - 15 GPU cores
  - 6 memory partition
  - 16KB 4-way L1, 768KB 16-way L2
  - Model L2 queue and L2 queuing latency
  - 1674 MHz GDDR5
- Workloads from CUDA-SDK, Rodinia, Mars and Lonestar suites

Comparison Points

- **FR-FCFS baseline** [Rixner+, ISCA'00]
- **Cache Insertion:**
  - **EAF** [Seshadri+, PACT'12]
    - Tracks blocks that are recently evicted **to detect high reuse and inserts them at the MRU position**
    - **Does not take divergence heterogeneity into account**
    - **Does not lower queuing latency**
- **Cache Bypassing:**
  - **PCAL** [Li+, HPCA'15]
    - Uses tokens to limit number of warps that gets to access the L2 cache → **Lower cache thrashing**
    - Warps with highly reuse access gets more priority
    - **Does not take divergence heterogeneity into account**
  - PC-based and Random bypassing policy

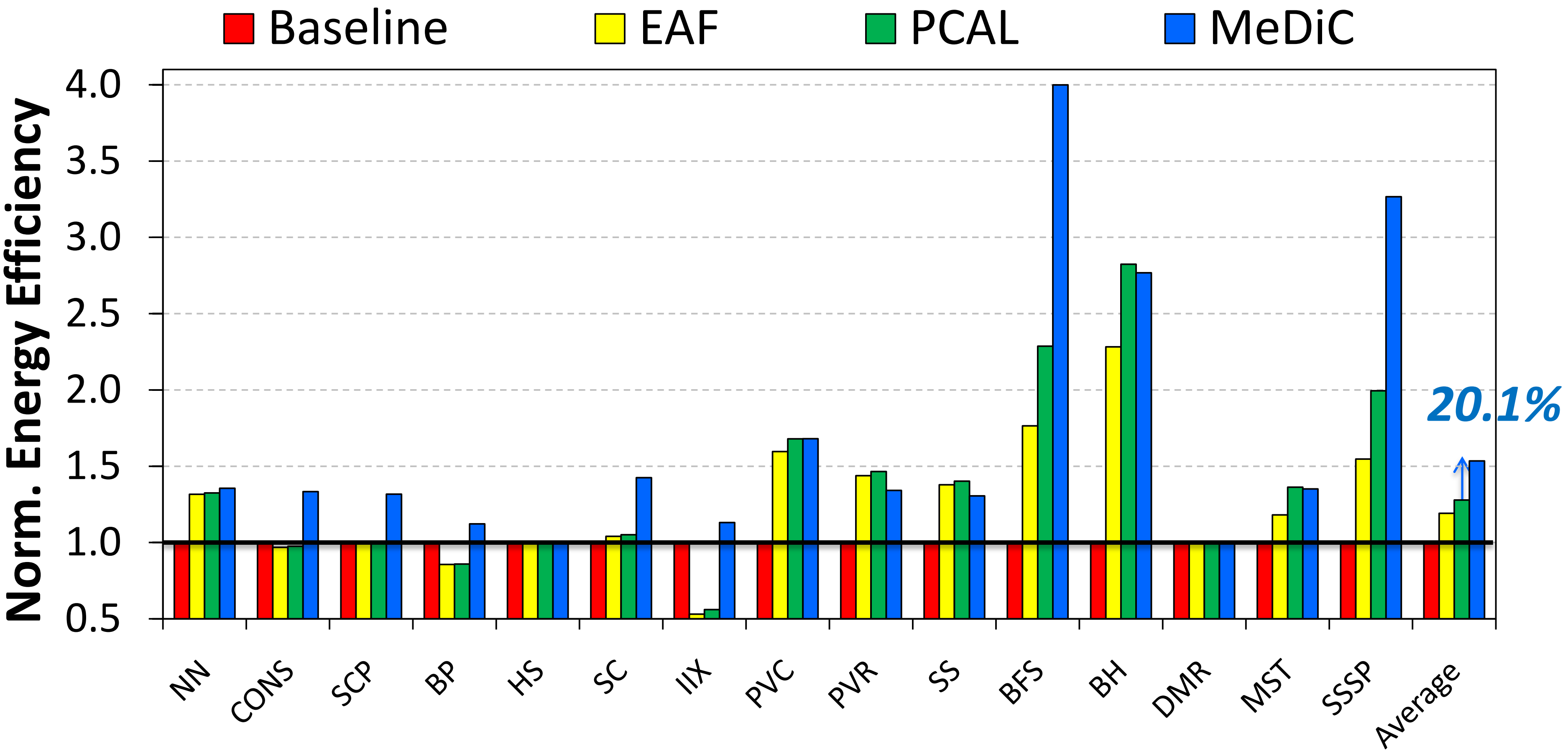
Results: Performance of MeDiC



MeDiC is effective in identifying warp-type and taking advantage of divergence heterogeneity

SAFARI

Results: Energy Efficiency of MeDiC



Performance improvement outweighs the additional energy from extra cache misses

SAFARI

Other Results in the Paper

- Breakdowns of each component of MeDiC
  - Each component is effective
- Comparison against PC-based and random cache bypassing policy
  - MeDiC provides better performance
- Analysis of combining MeDiC+reuse mechanism
  - MeDiC is effective in caching highly-reused blocks
- Sensitivity analysis of each individual components
  - Minimal impact on L2 miss rate
  - Minimal impact on row buffer locality
  - Improved L2 queuing latency

Conclusion

- **Problem: memory divergence**
  - Threads execute in lockstep, but not all threads hit in the cache
  - A single long latency thread can stall an entire warp
- **Key Observations:**
  - Memory divergence characteristic differs across warps
  - Some warps mostly hit in the cache, some mostly miss
  - Divergence characteristic is stable over time
  - L2 queuing exacerbates memory divergence problem
- **Our Solution: Me**mory **Di**vergence **C**orrection
  - Uses **cache bypassing**, **cache insertion** and **memory scheduling** to **prioritize mostly-hit warps** and **deprioritize mostly-miss warps**
- **Key Results:**
  - **21.8% better performance and 20.1% better energy efficiency** compared to state-of-the-art caching policy on GPU

Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance

Rachata Ausavarungnirun

Saugata Ghose, Onur Kayiran, Gabriel H. Loh
Chita Das, Mahmut Kandemir, Onur Mutlu

SAFARI

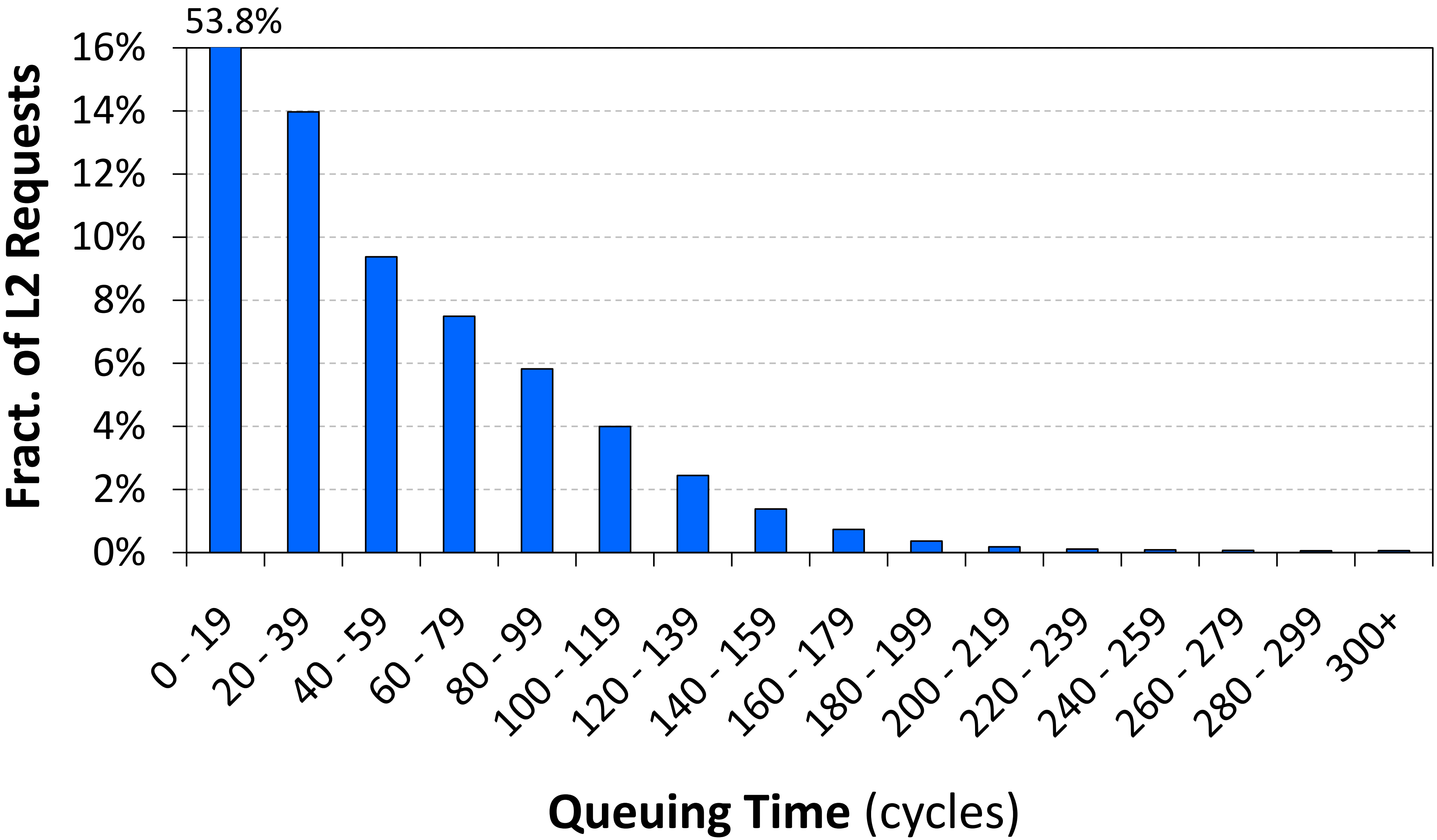
Carnegie Mellon

AMD

PENNSTATE
1 8 5 5

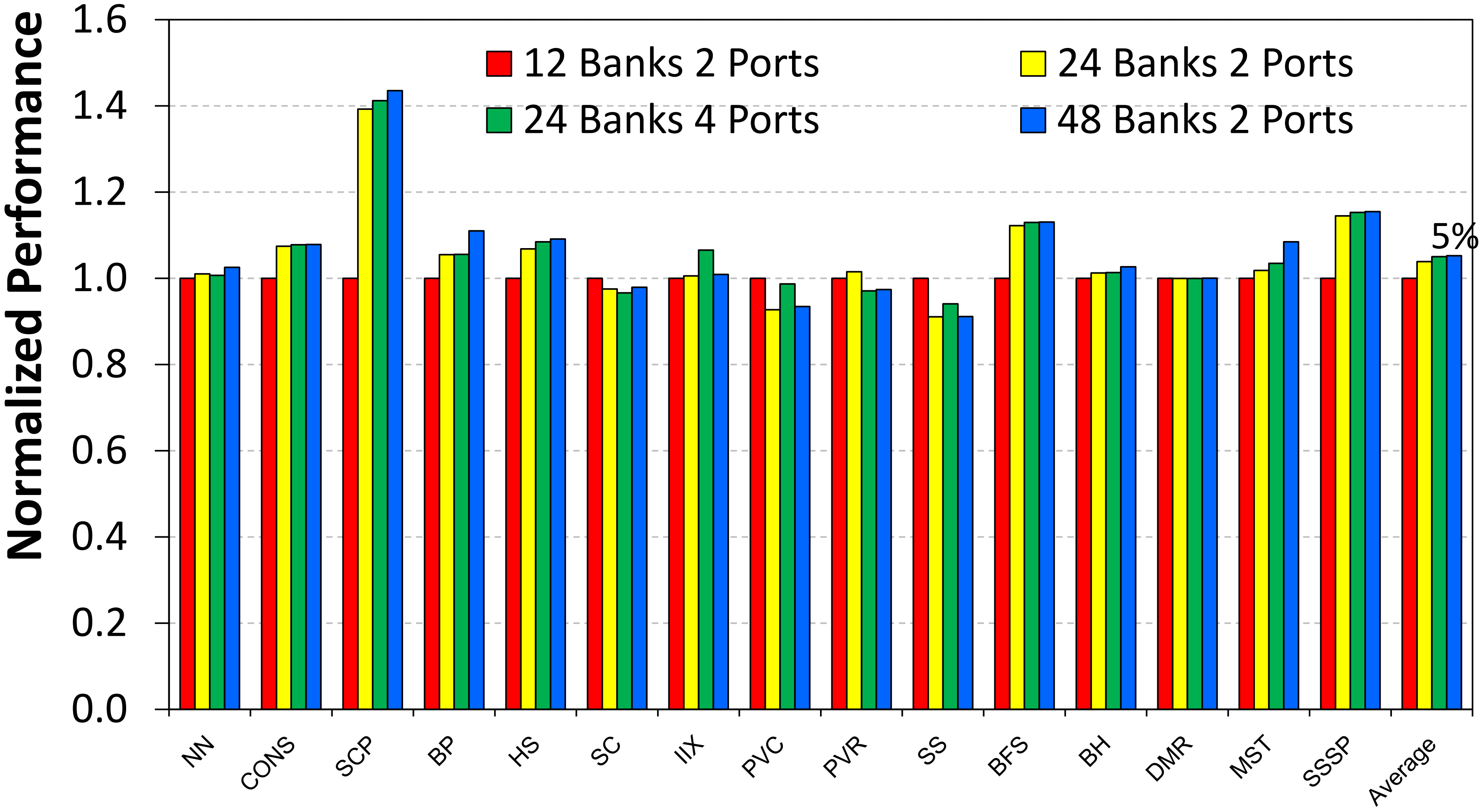
Backup Slides

Queuing at L2 Banks: Real Workloads



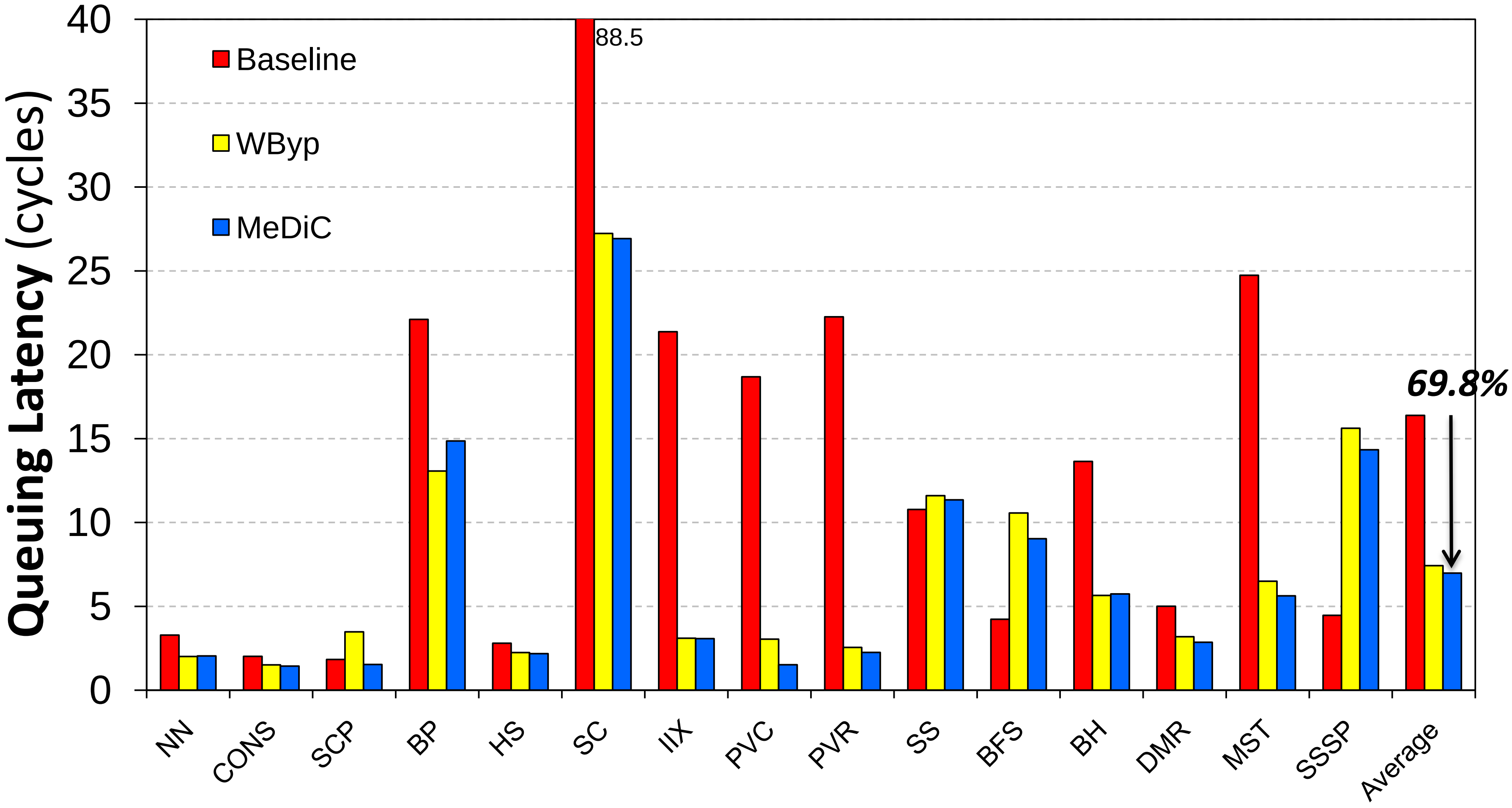
SAFARI

Adding More Banks



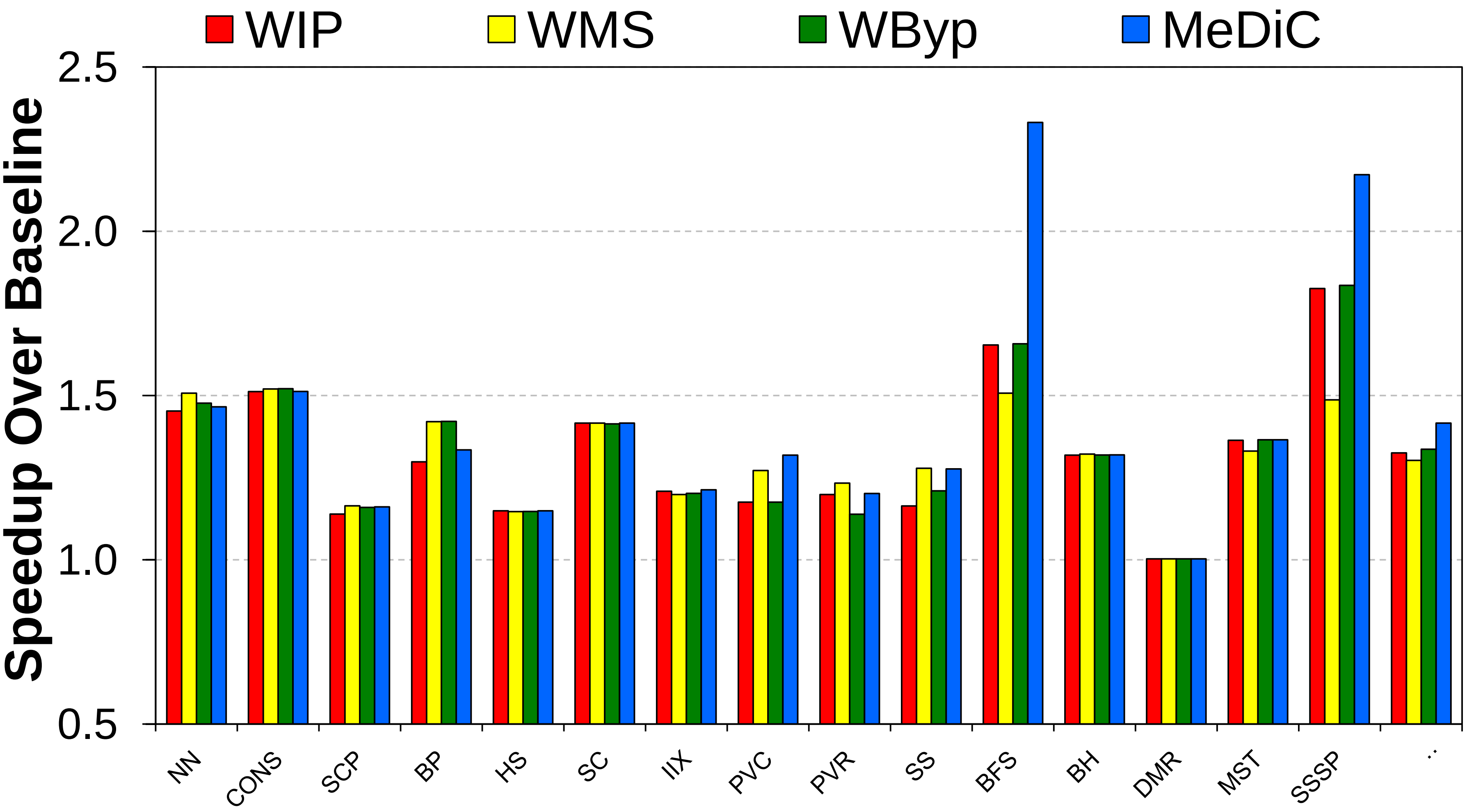
SAFARI

Queuing Latency Reduction

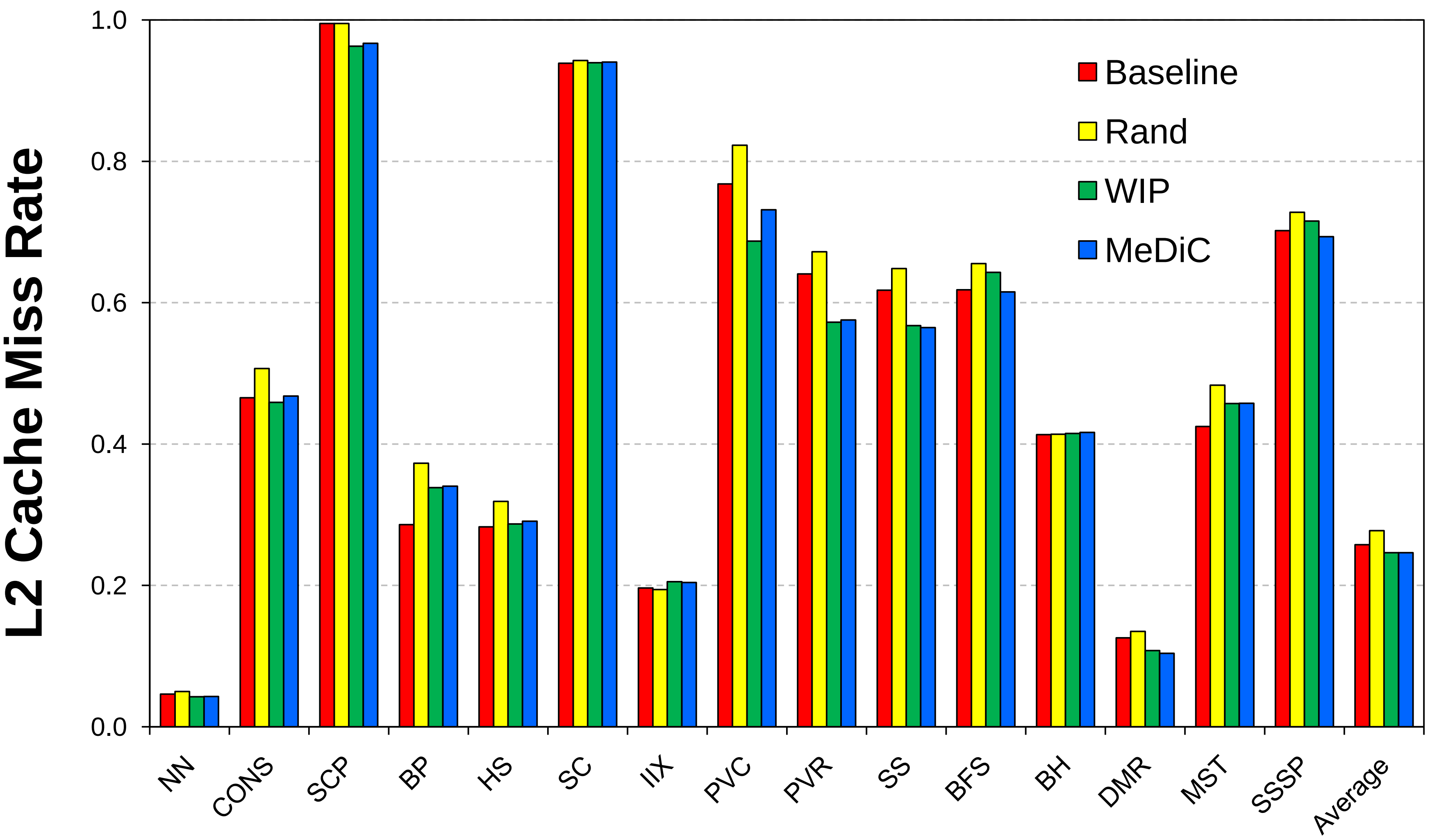


SAFARI

MeDiC: Performance Breakdown

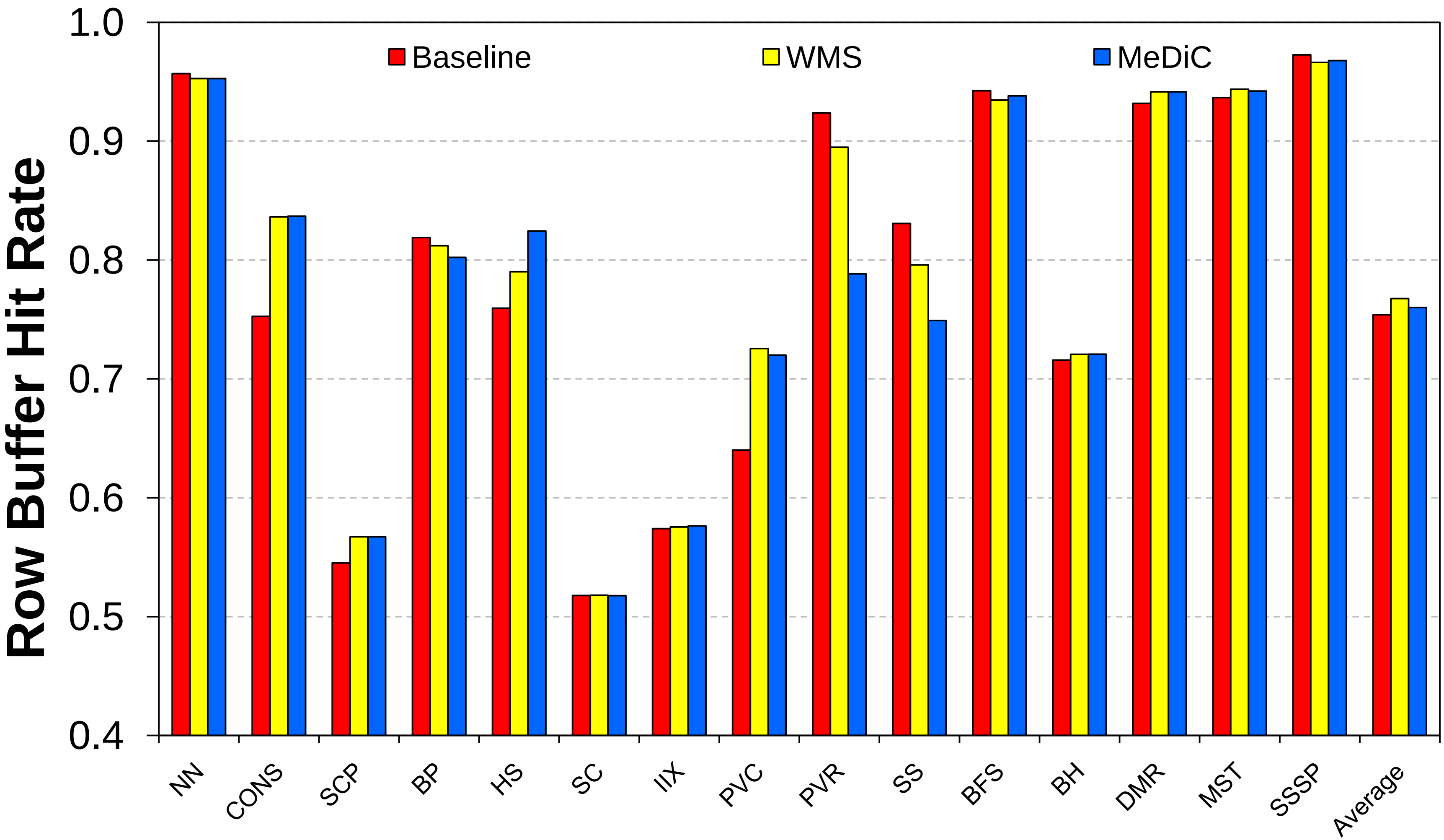


MeDiC: Miss Rate

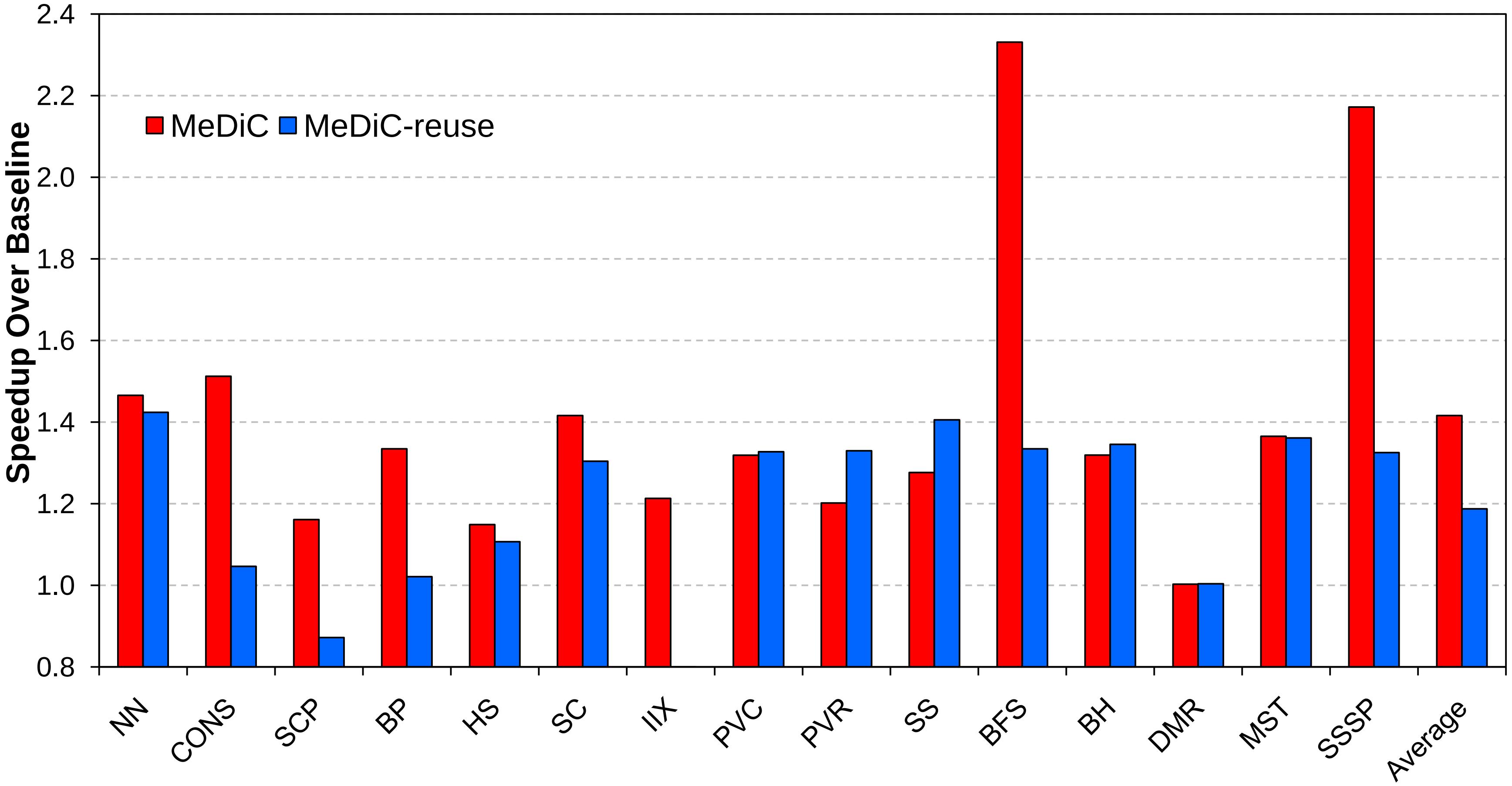


SAFARI

MeDiC: Row Buffer Hit Rate

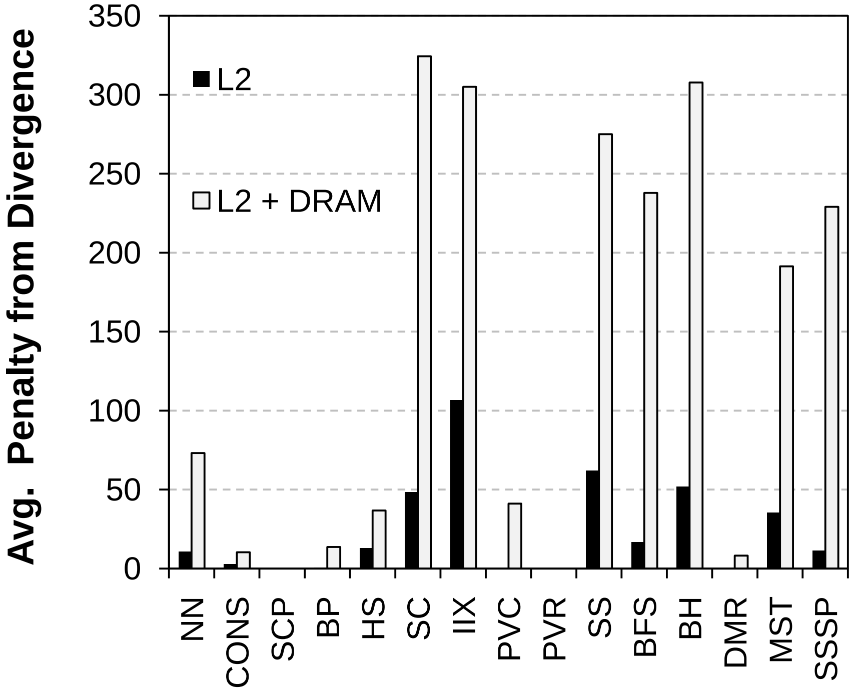


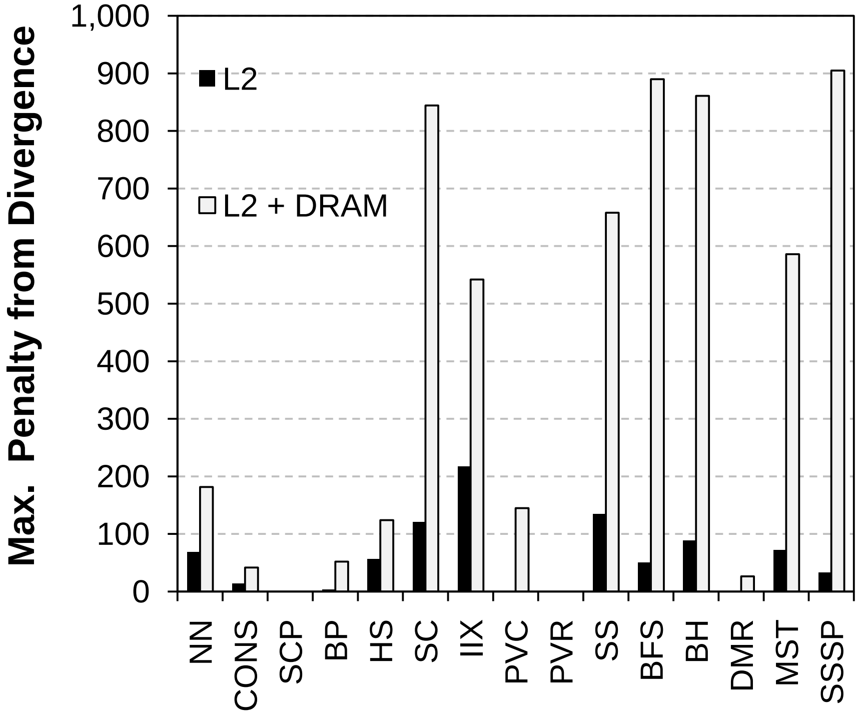
MeDiC-Reuse



SAFARI

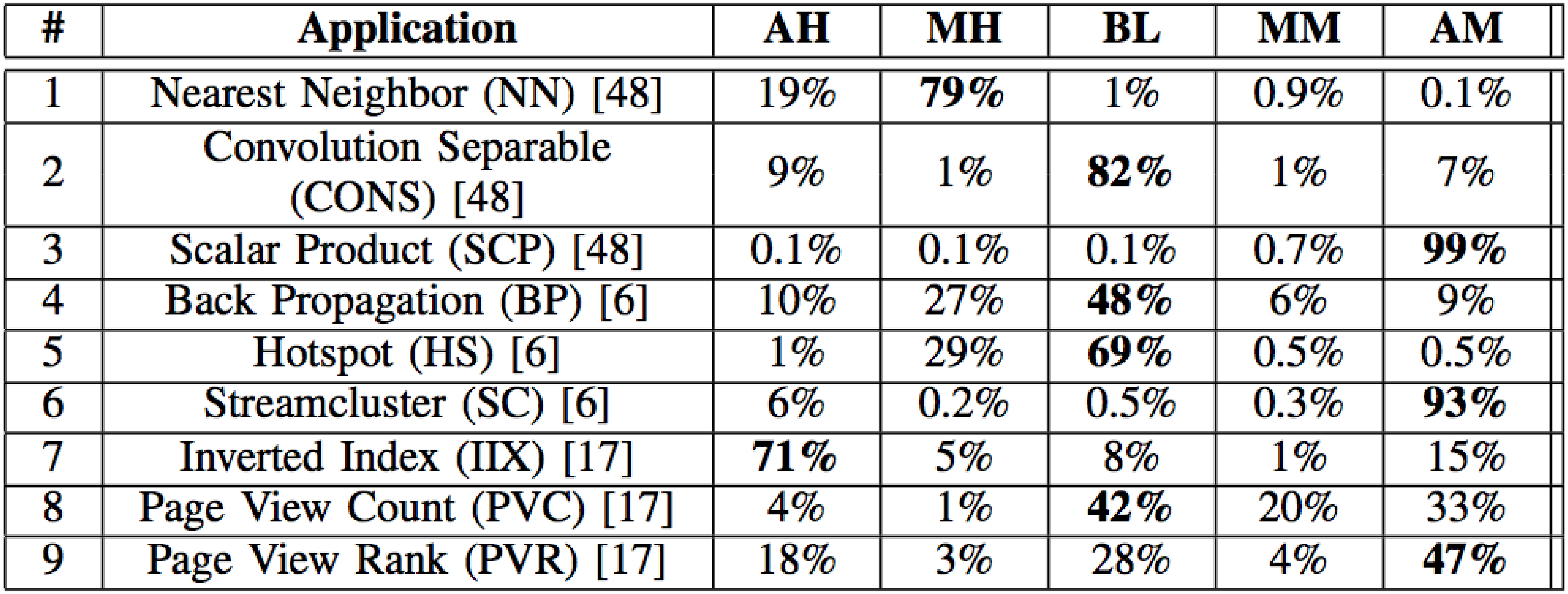
L2 Queuing Penalty





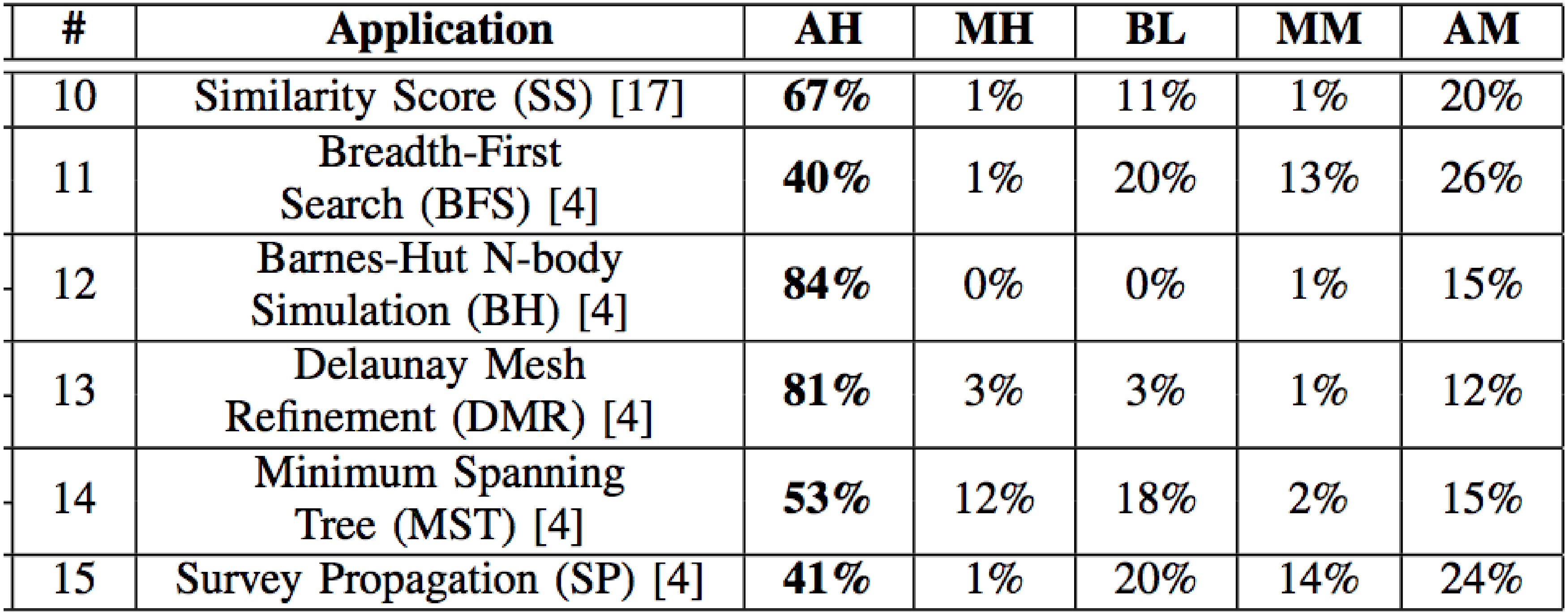
SAFARI

Divergence Distribution



#	Application	AH	MH	BL	MM	AM
1	Nearest Neighbor (NN) [48]	19%	**79%**	1%	0.9%	0.1%
2	Convolution Separable (CONS) [48]	9%	1%	**82%**	1%	7%
3	Scalar Product (SCP) [48]	0.1%	0.1%	0.1%	0.7%	**99%**
4	Back Propagation (BP) [6]	10%	27%	**48%**	6%	9%
5	Hotspot (HS) [6]	1%	29%	**69%**	0.5%	0.5%
6	Streamcluster (SC) [6]	6%	0.2%	0.5%	0.3%	**93%**
7	Inverted Index (IIX) [17]	**71%**	5%	8%	1%	15%
8	Page View Count (PVC) [17]	4%	1%	**42%**	20%	33%
9	Page View Rank (PVR) [17]	18%	3%	28%	4%	**47%**

Divergence Distribution



#	**Application**	**AH**	**MH**	**BL**	**MM**	**AM**
10	Similarity Score (SS) [17]	**67%**	1%	11%	1%	20%
11	Breadth-First Search (BFS) [4]	**40%**	1%	20%	13%	26%
12	Barnes-Hut N-body Simulation (BH) [4]	**84%**	0%	0%	1%	15%
13	Delaunay Mesh Refinement (DMR) [4]	**81%**	3%	3%	1%	12%
14	Minimum Spanning Tree (MST) [4]	**53%**	12%	18%	2%	15%
15	Survey Propagation (SP) [4]	**41%**	1%	20%	14%	24%

Stable Divergence Characteristics

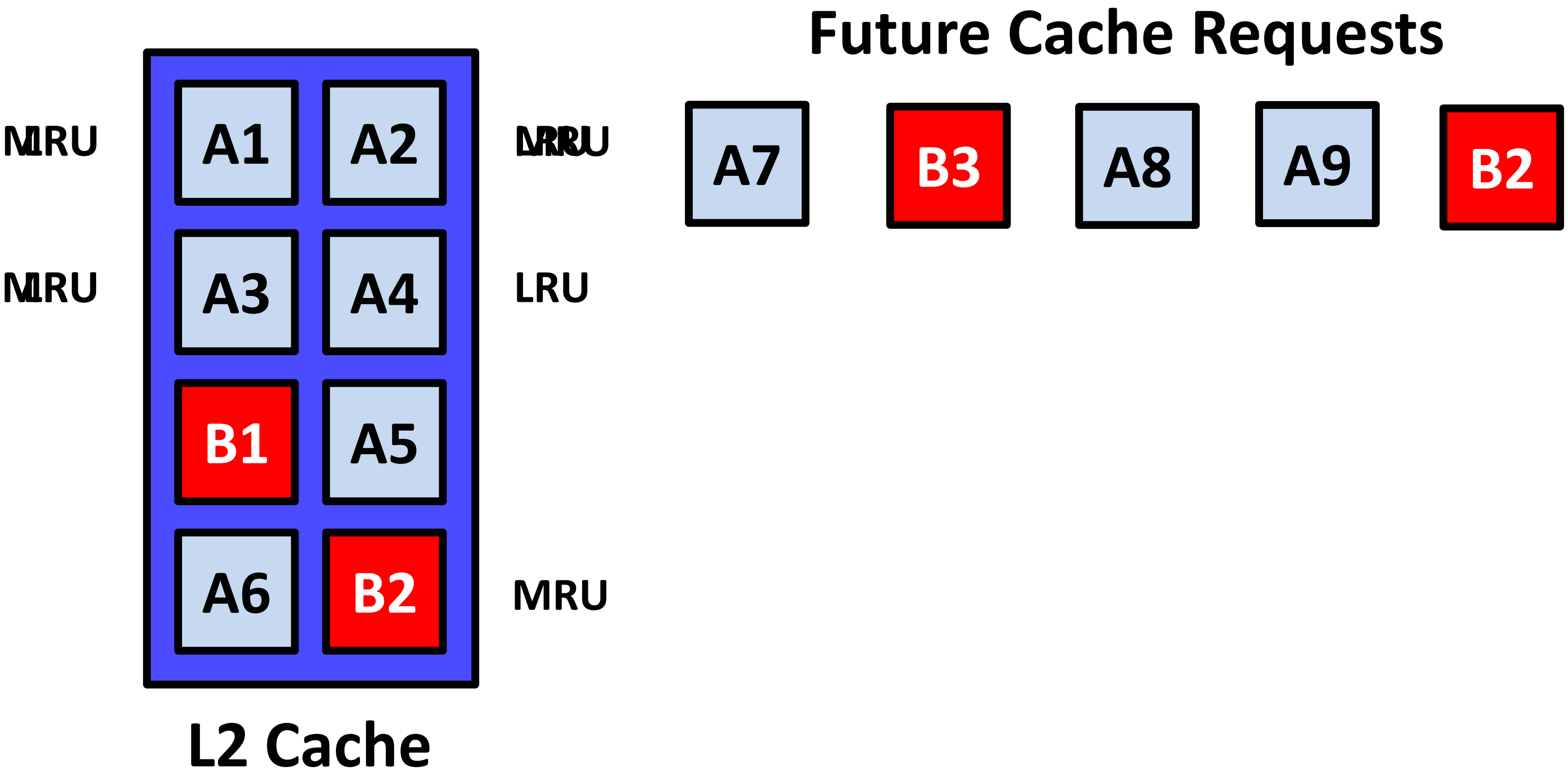
- Warp retains its hit ratio during a program phase
- Heterogeneity
  - Control Divergence
  - Memory Divergence
  - Edge cases on the data the program is operating on
  - Coalescing
  - Affinity to different memory partition
- Stability
  - Temporal + spatial locality

Warps Can Fetch Data for Others

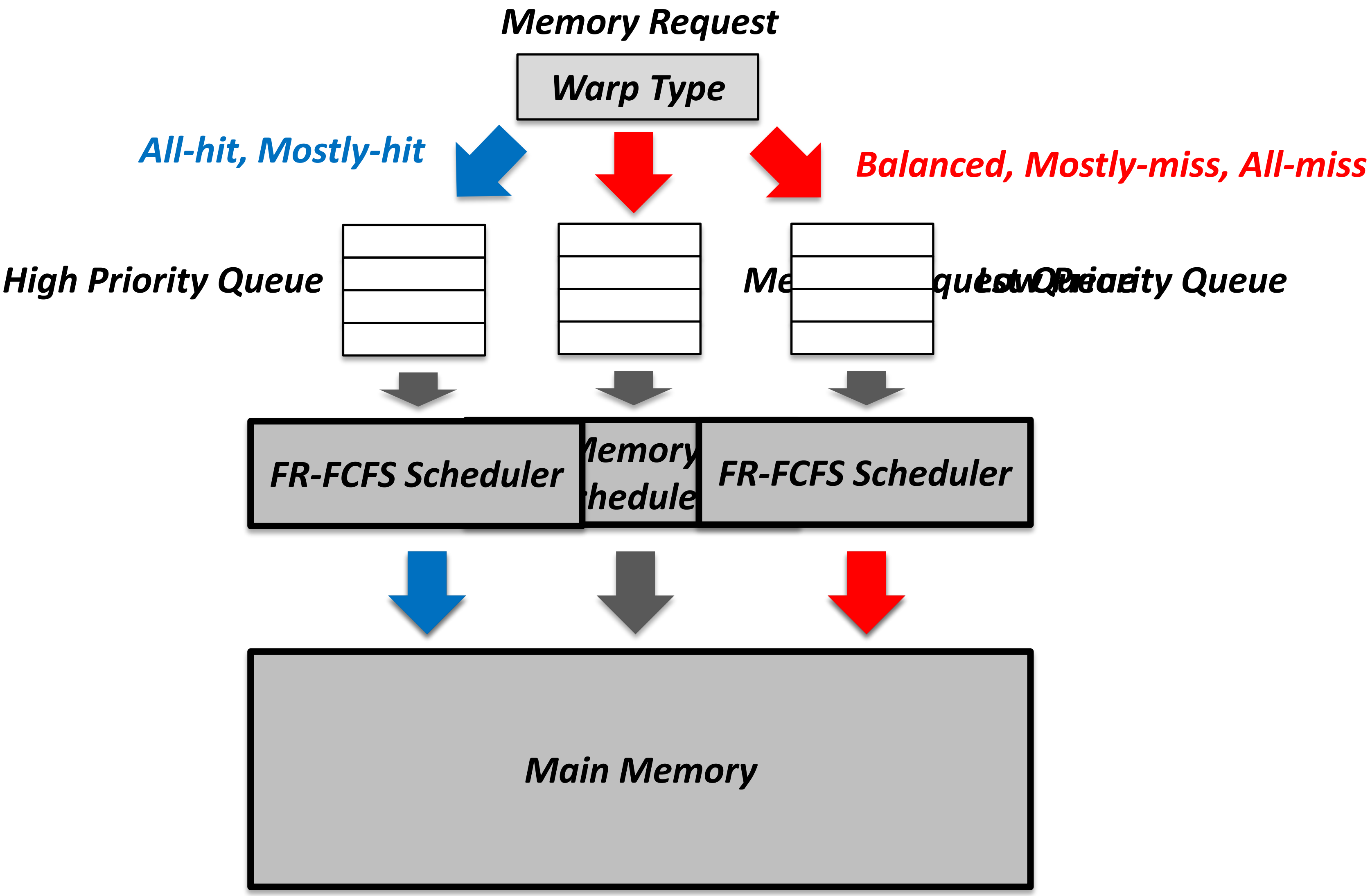
- **All-miss and mostly-miss** warps can fetch cache blocks for other warps
  - Blocks with high reuse
  - Shared address with all-hit and mostly-hit warps

- **Solution:** Warp-type aware cache insertion

Warp-type Aware Cache Insertion



Warp-type Aware Memory Sched.



Memory Request

Warp Type

All-hit, Mostly-hit

Balanced, Mostly-miss, All-miss

High Priority Queue

Me quest Low Priority Queue

FR-FCFS Scheduler

Memory hedule

FR-FCFS Scheduler

Main Memory

SAFARI