CARE:
A Concurrency-Aware Enhanced Lightweight Cache Management Framework

Xiaoyang Lu, Rujia Wang, Xian-He Sun

ILLINOIS TECH

Concurrent Memory Accesses

- Modern processors support data concurrency
- Data access **concurrency** is widely available
  - Some misses are **isolated**
  - Some misses overlap with other hits (**hit-miss overlapping**)
  - Some misses overlap with other miss (**miss-miss overlapping**)
- Cache miss penalty can be **hidden** by assess overlapping
- **The cost of cache misses varies**

Motivation

1

The cache bottleneck

- Performance gap between CPU and memory
- Multi-core poses challenges on shared cache management

2

The cache management solution

- **Reduce the number of cache misses (data locality)**
- **Reduce costly misses (data concurrency)**

Existing Solutions

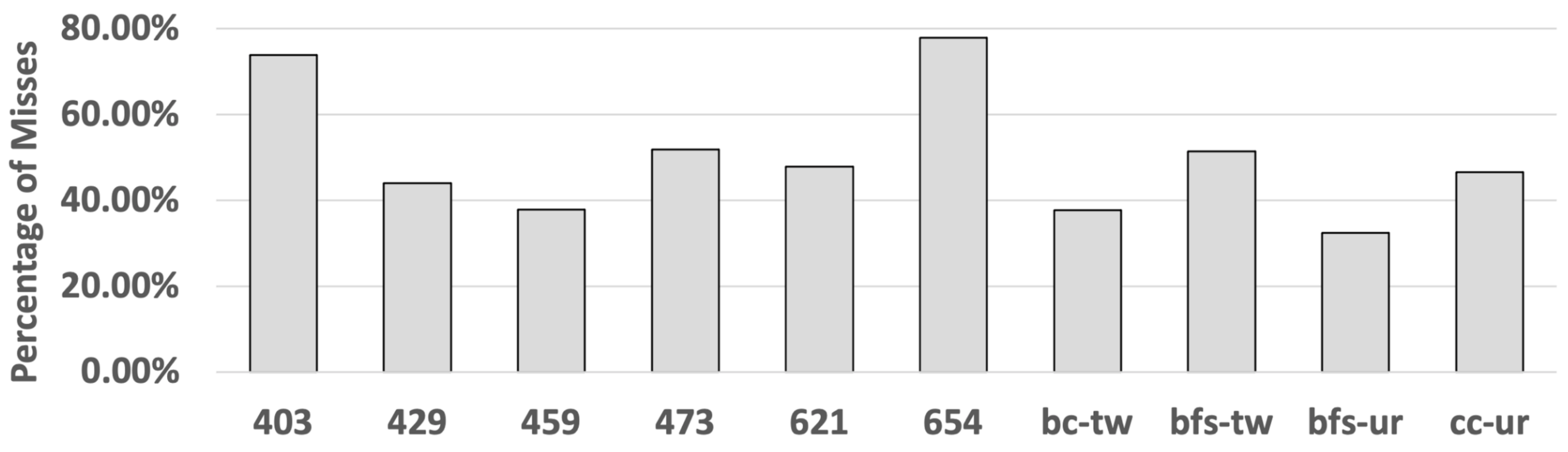
1 **Tradition: Locality-based cache management**

- Tries to reduce miss count
- Assumption: Reducing miss count reduces memory-related stalls
- **Enhanced by consider data concurrency**

2 **Advanced: MLP-based cache management**

- Considering miss-miss overlapping
- Assumption: Reducing an isolated miss helps performance more than reducing a parallel miss
- **Possible enhancement: hit-miss overlapping?**

Observation - Hit-Miss Overlapping



Percentage of misses with hit-miss overlapping.

- **30% - 80%** misses in LLC have **hit-miss overlapping**
- Hit-miss overlapping **cannot be ignored**
- A miss without hit-miss overlapping **(pure miss) can increase the latency of providing data to the upper-level cache**

Our Solution

A **comprehensive** cache management framework that considers both **data locality** and **full data concurrency**

Key Contributions

CARE, a dynamic adjustable, concurrency-aware, low-overhead cache management framework

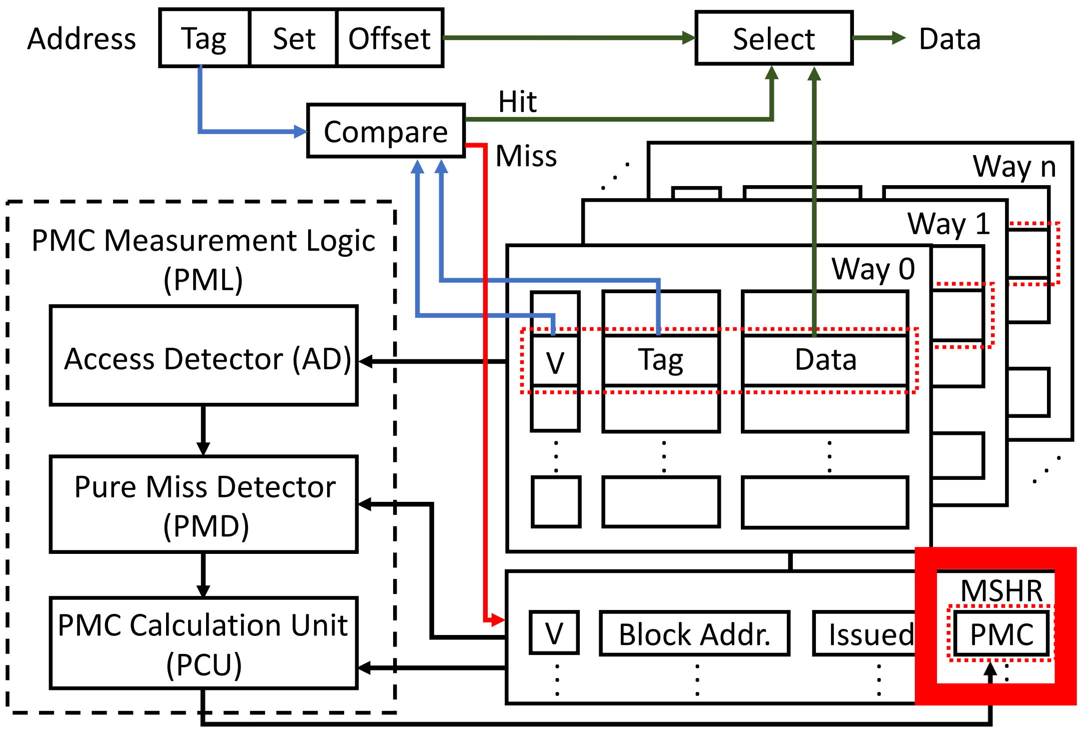
Pure Miss Contribution (PMC), to quantify the cost and performance impact of outstanding cache misses

Pure Miss Contribution (PMC) - Definition

- PMC is the number of cycles of the miss that causes the memory stall
- The contribution of each miss to the **active pure miss cycles**
    - **No hit accesses appear in pure miss cycles**, the latency of miss accesses cannot be **hidden**, causing memory stalls

How to measure it?

Pure Miss Contribution (PMC) - Measurement



MSHR tracks all in flight misses

Add a field PMC to each MSHR entry

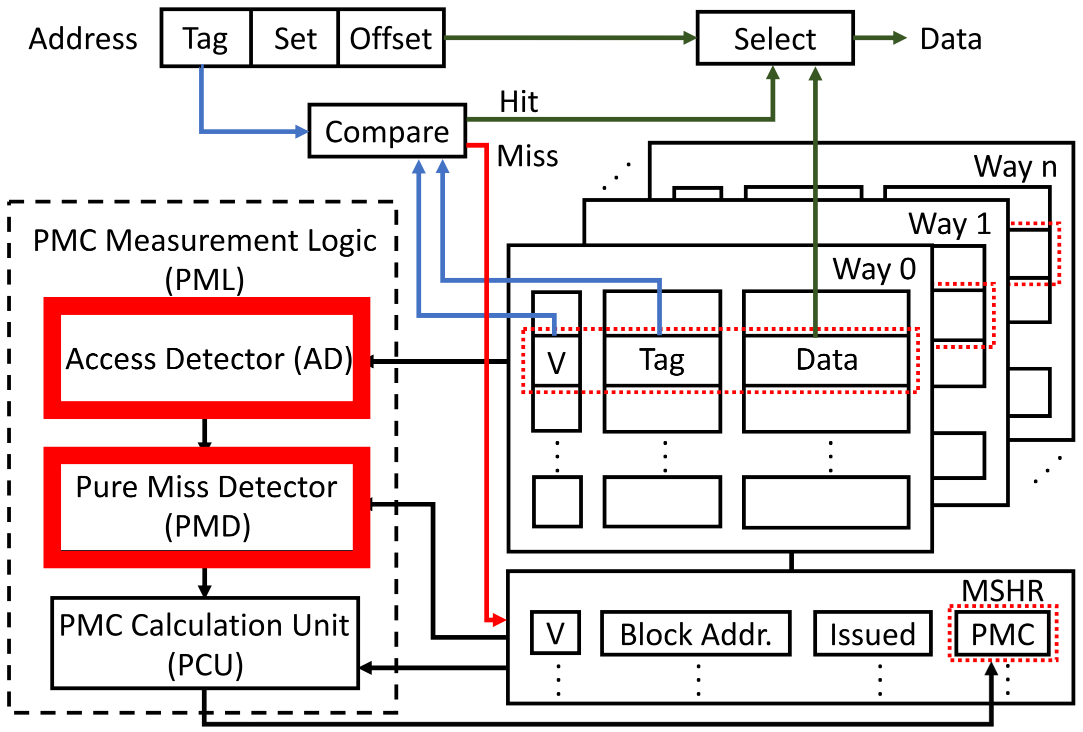
AD tracks the hit activities

PMD determines whether the current cycle is an active pure miss cycle

Every pure miss cycle for each entry in MSHR: $PMC \mathrel{+}= 1/N$

- N = Number of outstanding misses in MSHR

Pure Miss Contribution (PMC) - Measurement



MSHR tracks all in flight misses

Add a field PMC to each MSHR entry

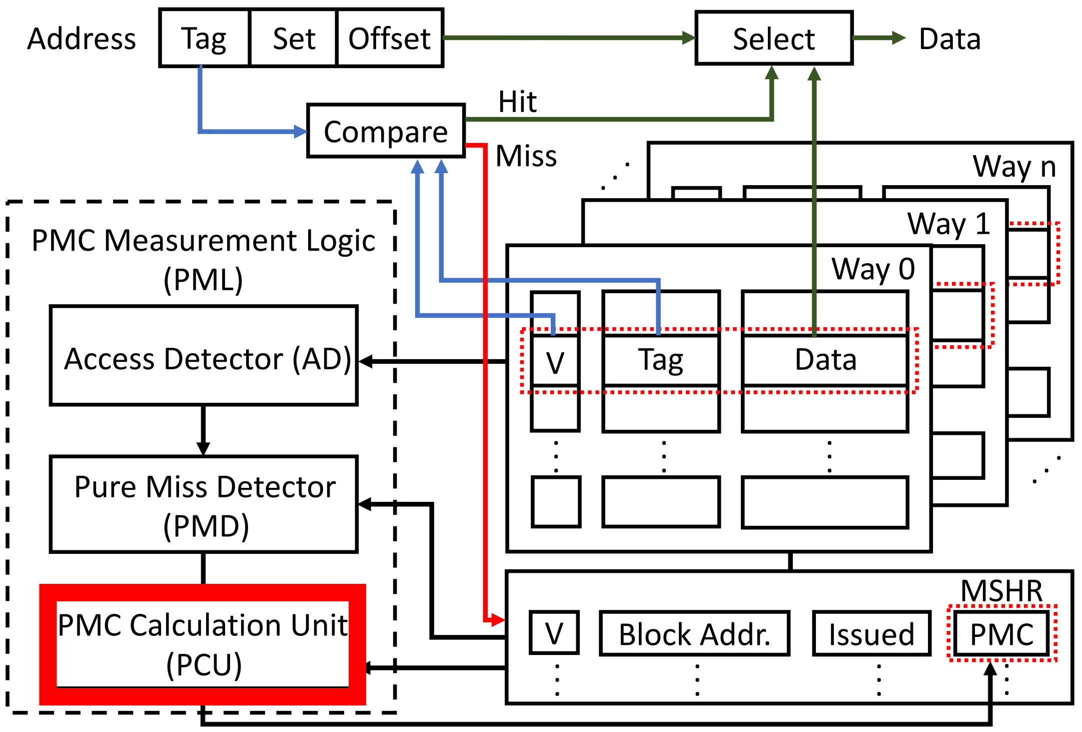
AD tracks the hit activities

PMD determines whether the current cycle is an active pure miss cycle

Every pure miss cycle for each entry in MSHR: $PMC += 1/N$

- N = Number of outstanding misses in MSHR

Pure Miss Contribution (PMC) - Measurement



MSHR tracks all in flight misses

Add a field PMC to each MSHR entry

AD tracks the hit activities

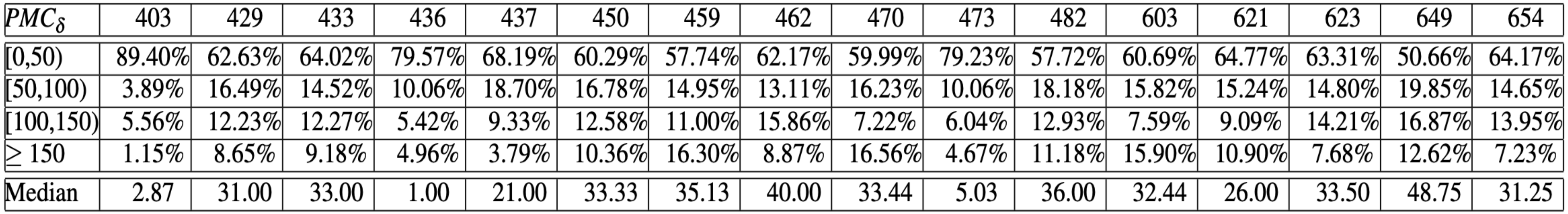
PMD determines whether the current cycle is an active pure miss cycle

Every pure miss cycle for each entry in MSHR: $PMC += 1/N$

- N = Number of outstanding misses in MSHR

Pure Miss Contribution (PMC) - Predictability

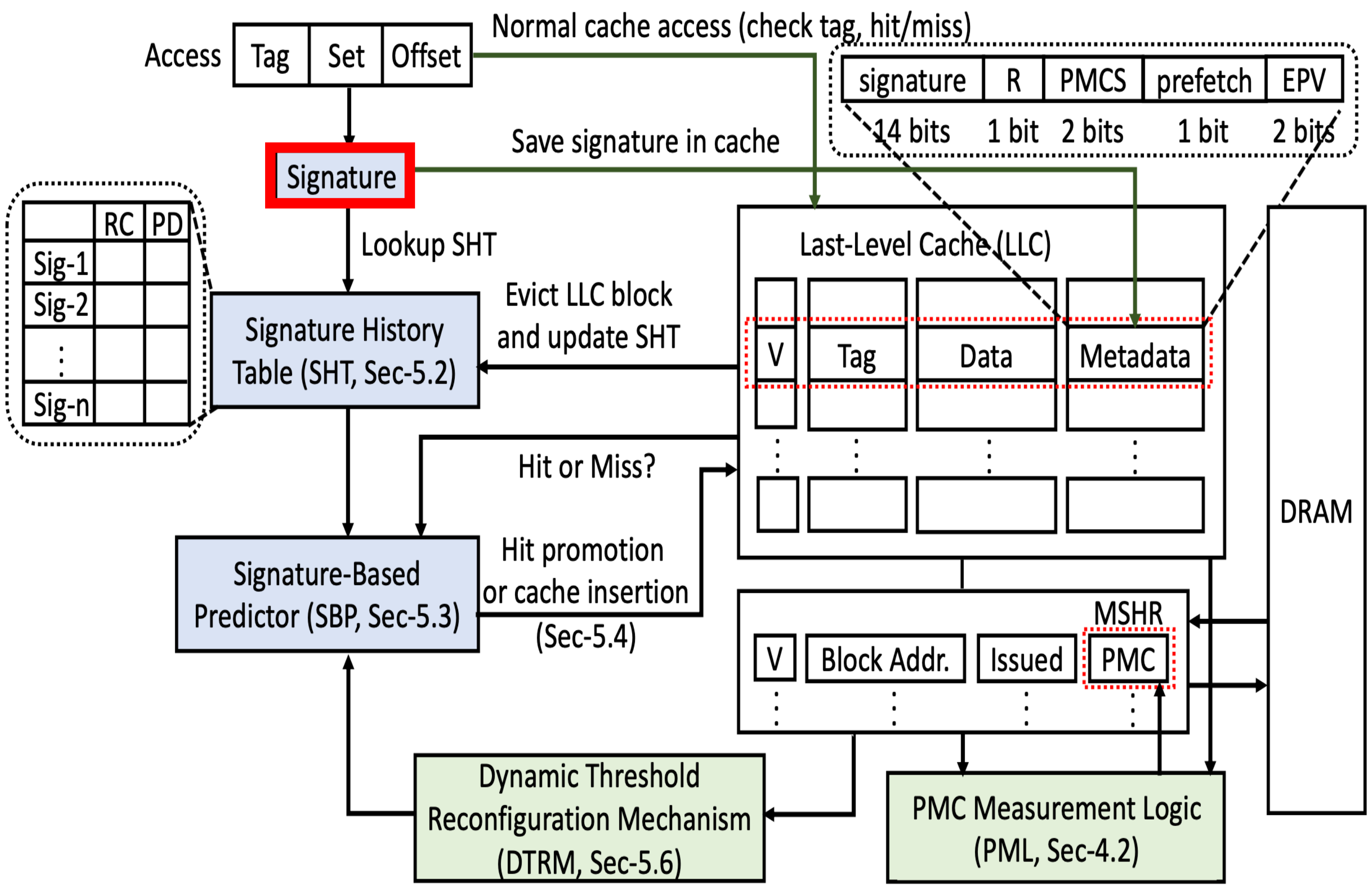
- Can current PMC be used to predict future PMC?
- Using Program Counter (PC) to predict PMC



PMC_δ	403	429	433	436	437	450	459	462	470	473	482	603	621	623	649	654
[0,50)	89.40%	62.63%	64.02%	79.57%	68.19%	60.29%	57.74%	62.17%	59.99%	79.23%	57.72%	60.69%	64.77%	63.31%	50.66%	64.17%
[50,100)	3.89%	16.49%	14.52%	10.06%	18.70%	16.78%	14.95%	13.11%	16.23%	10.06%	18.18%	15.82%	15.24%	14.80%	19.85%	14.65%
[100,150)	5.56%	12.23%	12.27%	5.42%	9.33%	12.58%	11.00%	15.86%	7.22%	6.04%	12.93%	7.59%	9.09%	14.21%	16.87%	13.95%
$\geq$ 150	1.15%	8.65%	9.18%	4.96%	3.79%	10.36%	16.30%	8.87%	16.56%	4.67%	11.18%	15.90%	10.90%	7.68%	12.62%	7.23%
Median	2.87	31.00	33.00	1.00	21.00	33.33	35.13	40.00	33.44	5.03	36.00	32.44	26.00	33.50	48.75	31.25

PMC values of the misses caused by the same PC are relatively stable

CARE: CONCURRENCY-AWARE CACHE MANAGEMENT



Access
Tag
Set
Offset
Normal cache access (check tag, hit/miss)
signature
R
PMCS
prefetch
EPV
14 bits
1 bit
2 bits
1 bit
2 bits
Save signature in cache
Signature
RC
PD
Sig-1
Sig-2
Sig-n
Lookup SHT
Signature History Table (SHT, Sec-5.2)
Evict LLC block and update SHT
Last-Level Cache (LLC)
V
Tag
Data
Metadata
Hit or Miss?
DRAM
Hit promotion or cache insertion (Sec-5.4)
Signature-Based Predictor (SBP, Sec-5.3)
MSHR
V
Block Addr.
Issued
PMC
Dynamic Threshold Reconfiguration Mechanism (DTRM, Sec-5.6)
PMC Measurement Logic (PML, Sec-4.2)

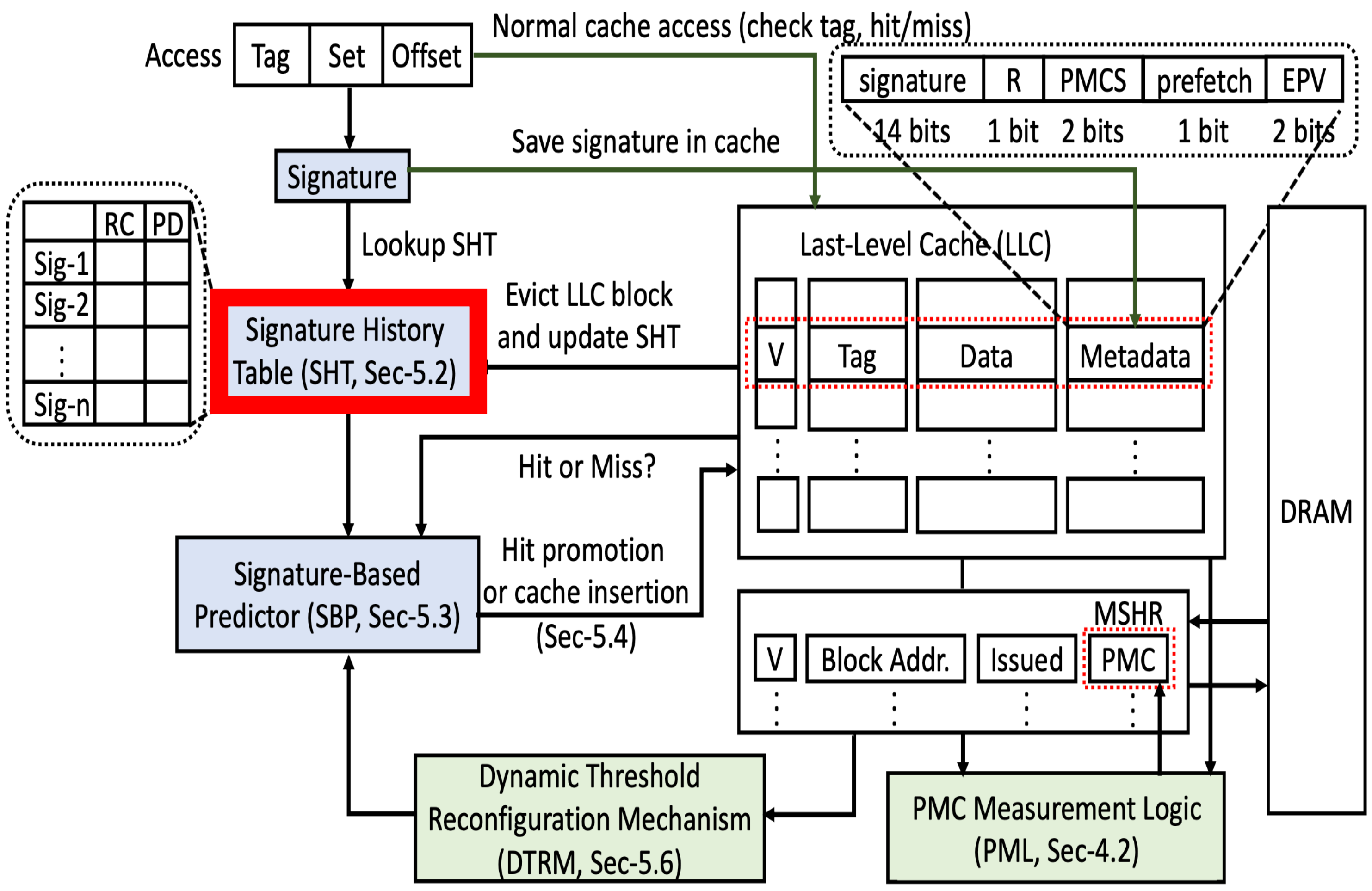
Signature History Table (SHT)

- Keeping track of the re-reference and PMC behaviors of LLC blocks
- Associating accesses with PC signature

Signature-Based Predictor (SBP)

- Making re-reference and PMC predictions
- Performing cache insertions and hit promotions

CARE: CONCURRENCY-AWARE CACHE MANAGEMENT



Access
Tag
Set
Offset
Normal cache access (check tag, hit/miss)
signature
R
PMCS
prefetch
EPV
14 bits
1 bit
2 bits
1 bit
2 bits
Save signature in cache
Signature
RC
PD
Sig-1
Sig-2
⋮
Sig-n
Lookup SHT
Signature History Table (SHT, Sec-5.2)
Evict LLC block and update SHT
Last-Level Cache (LLC)
V
Tag
Data
Metadata
Hit or Miss?
Signature-Based Predictor (SBP, Sec-5.3)
Hit promotion or cache insertion (Sec-5.4)
MSHR
V
Block Addr.
Issued
PMC
DRAM
Dynamic Threshold Reconfiguration Mechanism (DTRM, Sec-5.6)
PMC Measurement Logic (PML, Sec-4.2)

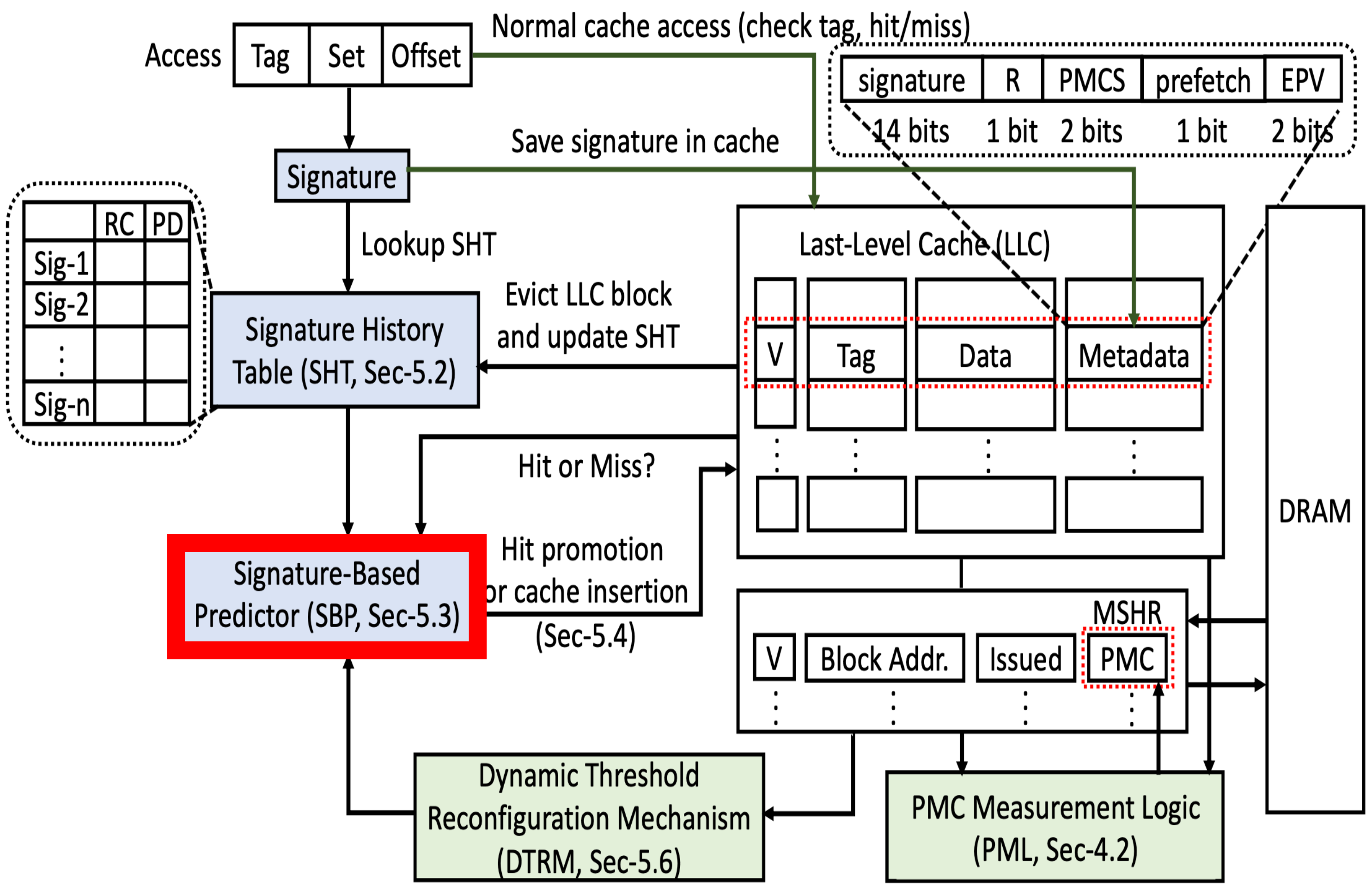
Signature History Table (SHT)

- Keeping track of the re-reference and PMC behaviors of LLC blocks
- Associating accesses with PC signature

Signature-Based Predictor (SBP)

- Making re-reference and PMC predictions
- Performing cache insertions and hit promotions

CARE: CONCURRENCY-AWARE CACHE MANAGEMENT



Access
Tag
Set
Offset
Normal cache access (check tag, hit/miss)
signature
R
PMCS
prefetch
EPV
14 bits
1 bit
2 bits
1 bit
2 bits
Save signature in cache
Signature
Lookup SHT
RC
PD
Sig-1
Sig-2
Sig-n
Signature History Table (SHT, Sec-5.2)
Evict LLC block and update SHT
Last-Level Cache (LLC)
V
Tag
Data
Metadata
Hit or Miss?
Signature-Based Predictor (SBP, Sec-5.3)
Hit promotion or cache insertion (Sec-5.4)
MSHR
V
Block Addr.
Issued
PMC
DRAM
Dynamic Threshold Reconfiguration Mechanism (DTRM, Sec-5.6)
PMC Measurement Logic (PML, Sec-4.2)

Signature History Table (SHT)

- Keeping track of the re-reference and PMC behaviors of LLC blocks
- Associating accesses with PC signature

Signature-Based Predictor (SBP)

- Making re-reference and PMC predictions
- Performing cache insertions and hit promotions

CARE

- Making re-reference and PMC predictions
- CARE identifies the re-reference behavior of a cache block as
  - **High-Reuse** if the related RC counter is 7
  - **Low-Reuse** if the related RC counter is 0
  - All other cache accesses are classified as **Moderate-Reuse**
- CARE predicts the impact of a cache block on performance as
  - **High-Cost** if the related PD counter is 7
  - **Low-Cost** if the related PD counter is 0

CARE - Management Policies

- Improve data **locality**
  - Keep **High-Reuse** cache blocks
  - Give higher eviction priority to **Low-Reuse** blocks
- Take data **concurrency** into account
  - For **Moderate-Reuse** blocks:
    - Keep **High-Cost** cache blocks to reduce expensive misses
    - Give higher eviction priority to the **Low-Cost** blocks
- 2-bit Eviction Priority Value (EPV)
  - Reflect the eviction priorities of the cache block

More in the Paper

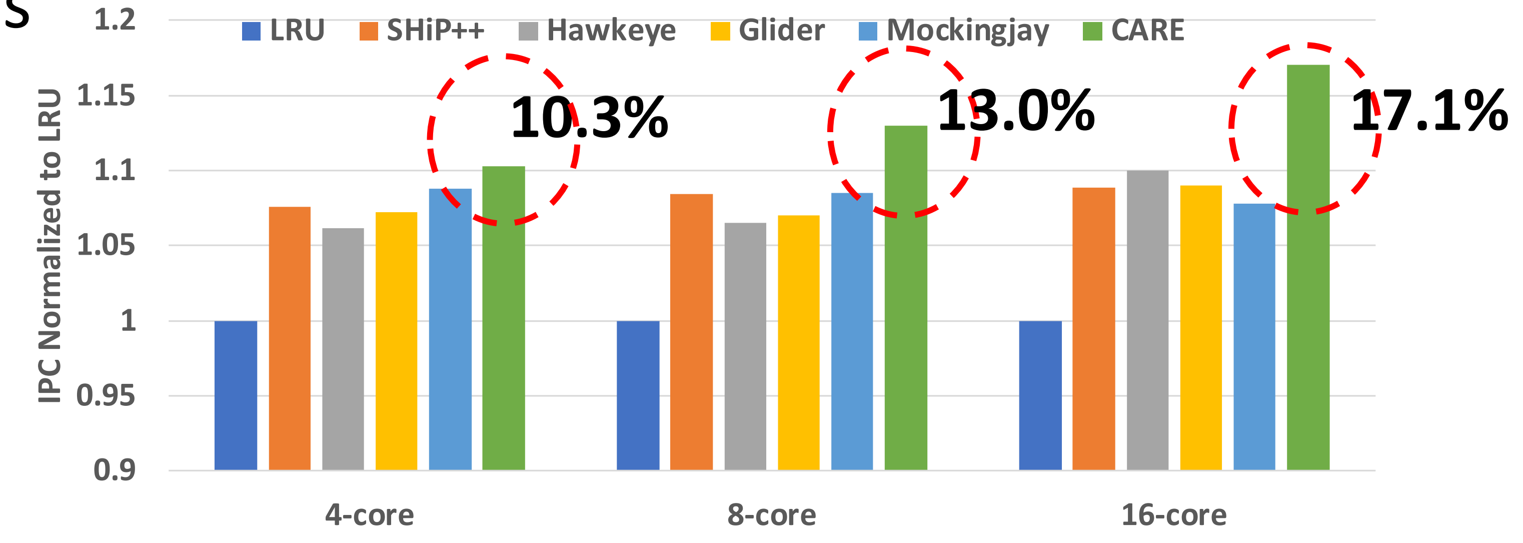
- CARE's Cache Management Policies
  - Insertion policy
  - Hit-promotion policy
  - Victim selection
- Collaboration with Prefetching
- **Dynamic Threshold Reconfiguration Mechanism**
  - Quantize PMC values to suit different workloads and different phases

Simulation Methodology

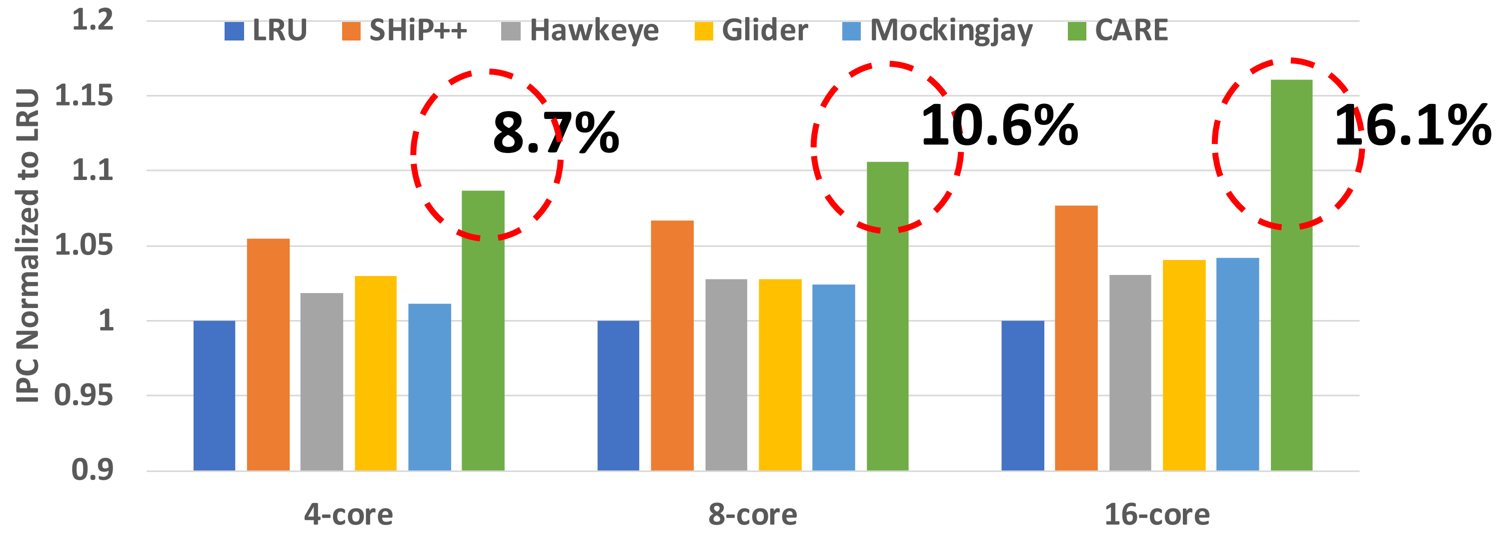
- **Champsim** trace-driven simulator
- **45** single-core memory-intensive workload traces
  - SPEC CPU2006 and CPU2017
  - GAP
- Homogeneous and heterogeneous **multi-core mixes**
- **Prefetcher**
  - L1D: Next-line prefetcher
  - L2: IP-stride prefetcher
- **Five** state-of-the-art LLC management schemes
  - LRU
  - SHiP++ **[Young+, 2nd Cache Replacement Championship, 2017]**
  - Hawkeye **[Jain+, ISCA'16]**
  - Glider **[Shi+, MICRO'19]**
  - Mockingjay **[Shah+, HPCA'22]**

Performance with Varying Core Count

SPEC workloads



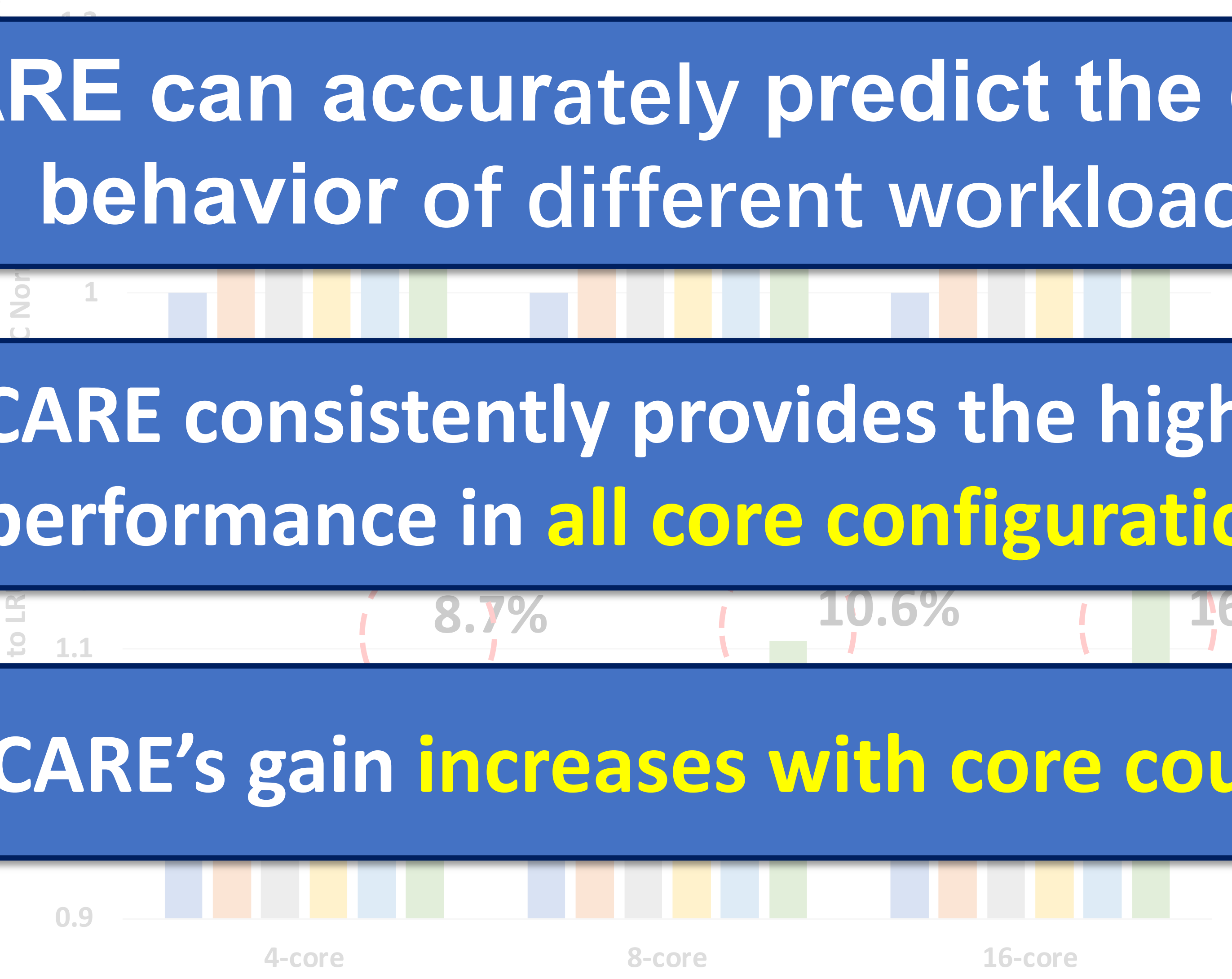
GAP workloads



Performance with Varying Core Count

SPEC workloads

GAP w



CARE can accurately predict the cache behavior of different workloads

CARE consistently provides the highest performance in all core configurations

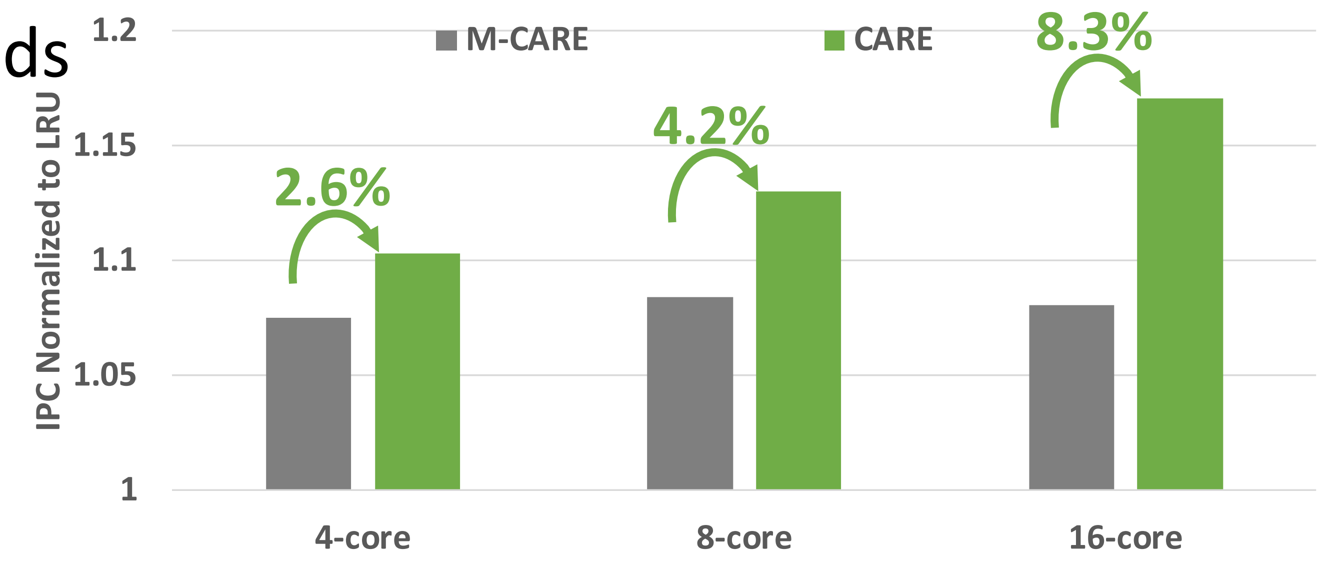
CARE's gain increases with core count

Performance Improvement via PMC

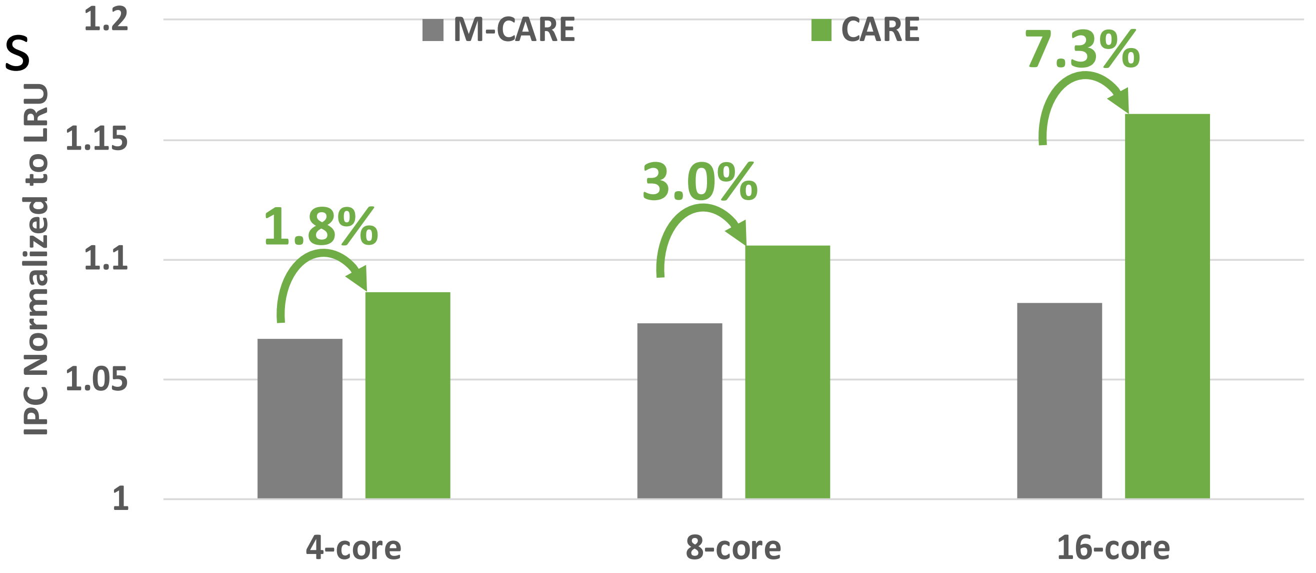
M-CARE

- Extend the **MLP-based cost** [Qureshi+, ISCA'06] so it can work under CARE
- Use **MLP-based cost** to analyze data access concurrency and guide cache management

SPEC workloads



GAP workloads

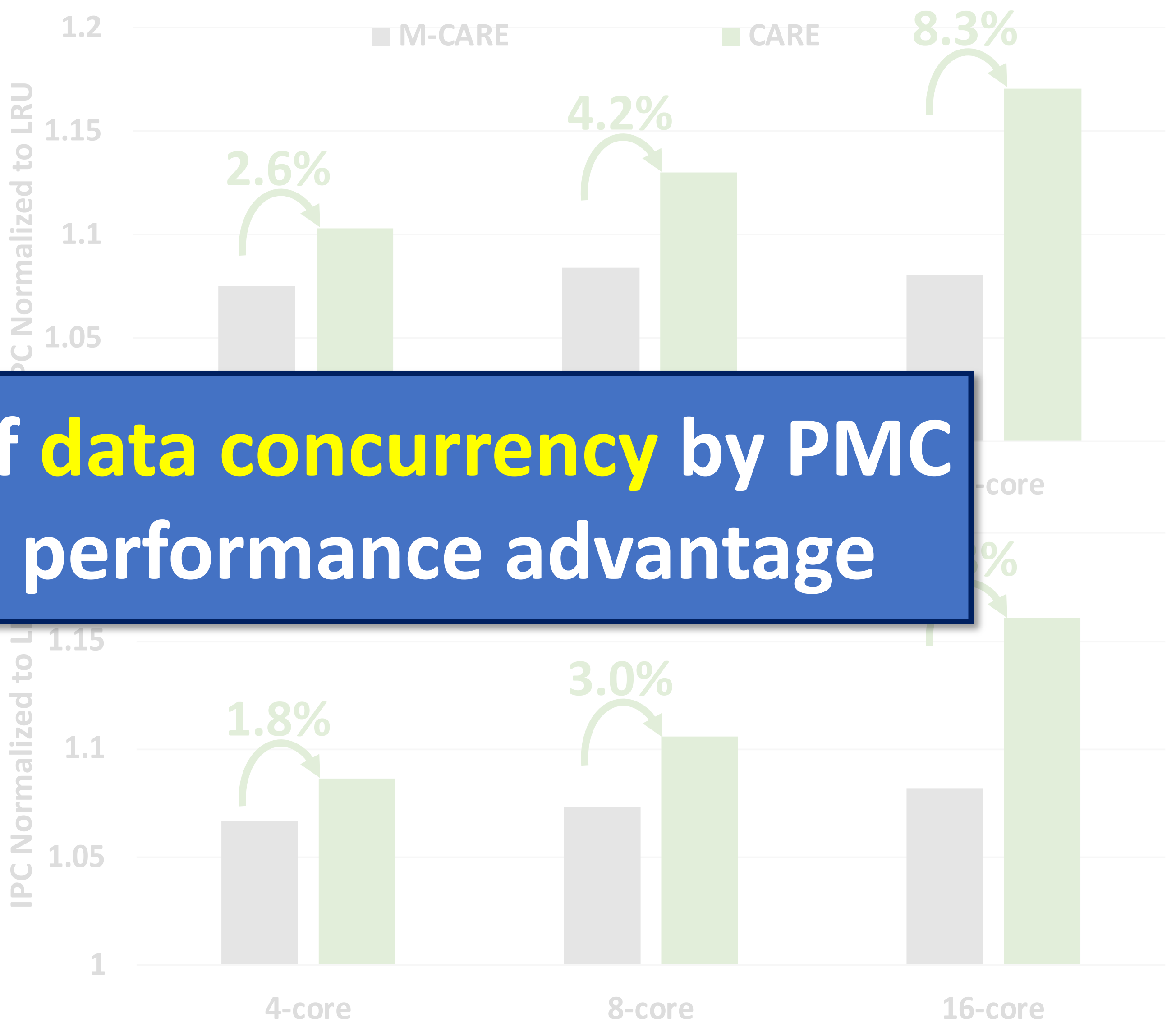


Performance Improvement via PMC

M-CARE

- Extend the **MLP-based cost** [Qureshi+, ISCA'06]
- The workflow of M-CARE is **similar** to CARE
- Use M
to anal
concurrency and guide cache management

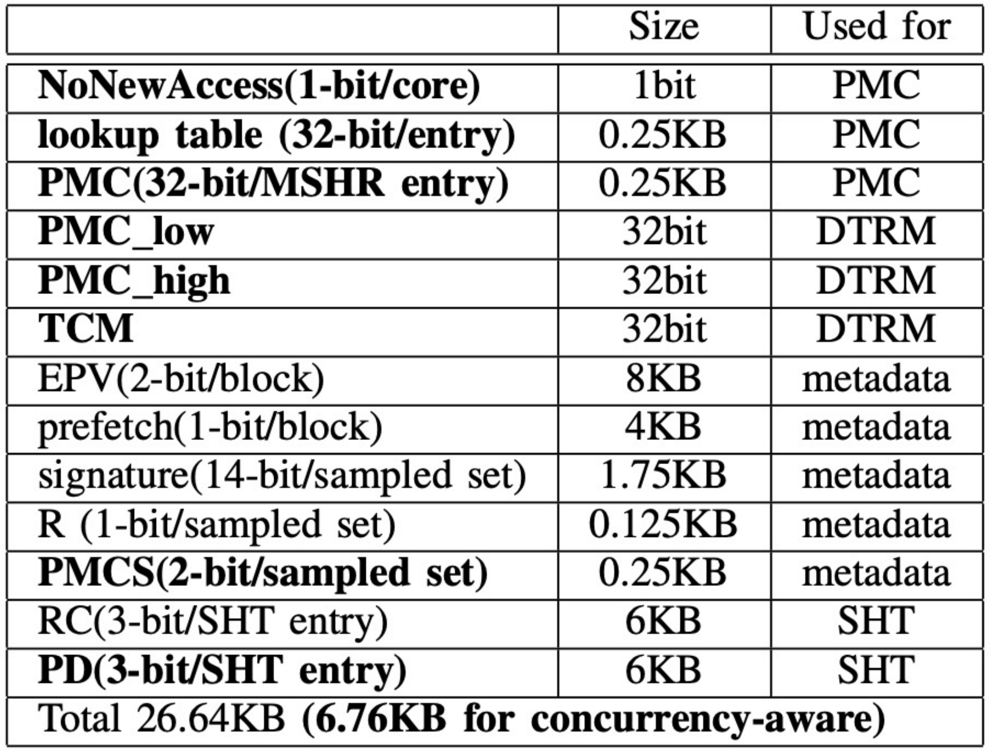
SPEC workloads



The accurate analysis of data concurrency by PMC provides CARE with a performance advantage

Overhead of CARE

- **26.6 KB** of total metadata storage **per core**
  - Only simple tables
  - **6.8 KB** for **concurrency awareness**



	Size	Used for
NoNewAccess(1-bit/core)	1bit	PMC
lookup table (32-bit/entry)	0.25KB	PMC
PMC(32-bit/MSHR entry)	0.25KB	PMC
PMC_low	32bit	DTRM
PMC_high	32bit	DTRM
TCM	32bit	DTRM
EPV(2-bit/block)	8KB	metadata
prefetch(1-bit/block)	4KB	metadata
signature(14-bit/sampled set)	1.75KB	metadata
R (1-bit/sampled set)	0.125KB	metadata
PMCS(2-bit/sampled set)	0.25KB	metadata
RC(3-bit/SHT entry)	6KB	SHT
PD(3-bit/SHT entry)	6KB	SHT
Total 26.64KB **(6.76KB for concurrency-aware)**		

Summary

Pure Miss Contribution (PMC),
a **comprehensive** metric used to weigh the **performance cost** of each cache miss

CARE considers **locality, concurrency,** and **overlapping** to guide cache replacement decision

CARE outperforms state-of-the-art cache management schemes

CARE:
A Concurrency-Aware Enhanced Lightweight Cache Management Framework

Xiaoyang Lu, Rujia Wang, Xian-He Sun

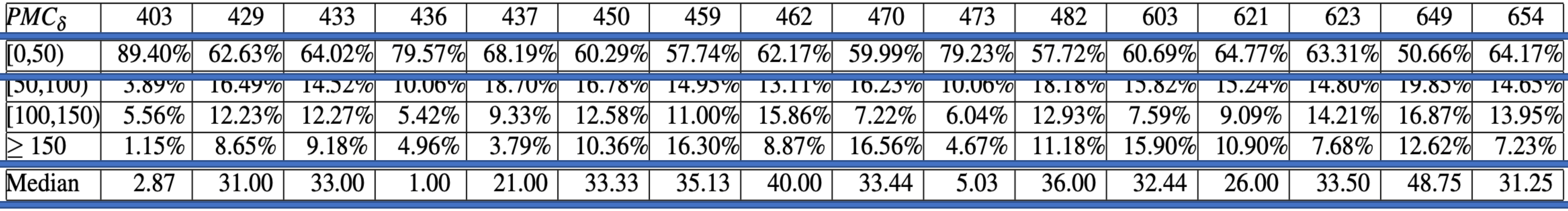
xlu40@hawk.iit.edu

ILLINOIS TECH

FAQs

Pure Miss Contribution (PMC) - Predictability

- Can current PMC be used to predict future PMC?
- Using Program Counter (PC) to predict PMC
- PMC_δ is the absolute difference in PMC values between two consecutive cache misses for the same PC



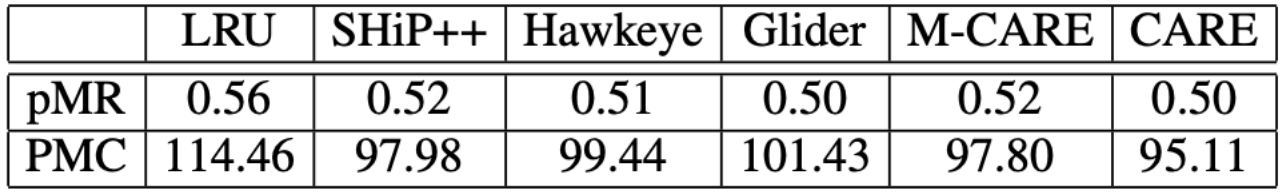
PMC_δ	403	429	433	436	437	450	459	462	470	473	482	603	621	623	649	654
[0,50)	89.40%	62.63%	64.02%	79.57%	68.19%	60.29%	57.74%	62.17%	59.99%	79.23%	57.72%	60.69%	64.77%	63.31%	50.66%	64.17%
[50,100)	3.89%	16.49%	14.52%	10.06%	18.70%	16.78%	14.95%	13.11%	16.23%	10.06%	18.18%	15.82%	15.24%	14.80%	19.85%	14.65%
[100,150)	5.56%	12.23%	12.27%	5.42%	9.33%	12.58%	11.00%	15.86%	7.22%	6.04%	12.93%	7.59%	9.09%	14.21%	16.87%	13.95%
≥ 150	1.15%	8.65%	9.18%	4.96%	3.79%	10.36%	16.30%	8.87%	16.56%	4.67%	11.18%	15.90%	10.90%	7.68%	12.62%	7.23%
Median	2.87	31.00	33.00	1.00	21.00	33.33	35.13	40.00	33.44	5.03	36.00	32.44	26.00	33.50	48.75	31.25

The majority of PMC_δ values are less than 50 cycles

The median PMC_δ of each workload is low

PMC values of the misses caused by the same PC are relatively stable

CARE Performance Evaluation



	LRU	SHiP++	Hawkeye	Glider	M-CARE	CARE
pMR	0.56	0.52	0.51	0.50	0.52	0.50
PMC	114.46	97.98	99.44	101.43	97.80	95.11

Table. Average pMR and PMC for all 4-core SPEC workloads (collaboration with L1 and L2 prefetcher).

Signature History Table (SHT)

- SHT contains two counters for each signature
- **Re-reference Confidence (RC) – 3bits**
  - Indicates the re-reference behavior for a signature
  - Updates on hit accesses and cache evictions
- **PMC Degree (PD) – 3bits**
  - Indicates the cost degree of the cache miss associated with the signature
  - Updates on cache insertions