
Xiaoyang Lu, Rujia Wang, Xian-He Sun
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

- **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

Percentage of misses with hit-miss overlapping.
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: $PMC += 1/N$

- $N =$ 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: \( PMC += \frac{1}{N} \)

- \( N \) = 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

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

PMC values of the misses caused by the same PC are relatively stable
CARE: CONCURRENCY-AWARE CACHE MANAGEMENT

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
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
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
• 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

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.

SPEC workloads:

- CARE can accurately predict the cache behavior of different workloads.

GAP workloads:

- CARE consistently provides the highest performance in all core configurations.
- CARE’s gain increases with core count.
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

<table>
<thead>
<tr>
<th>Core</th>
<th>M-CARE</th>
<th>CARE</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-core</td>
<td>1.05</td>
<td>1.07</td>
<td>2.6%</td>
</tr>
<tr>
<td>8-core</td>
<td>1.10</td>
<td>1.14</td>
<td>4.2%</td>
</tr>
<tr>
<td>16-core</td>
<td>1.15</td>
<td>1.23</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

### GAP workloads

<table>
<thead>
<tr>
<th>Core</th>
<th>M-CARE</th>
<th>CARE</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-core</td>
<td>1.05</td>
<td>1.17</td>
<td>1.8%</td>
</tr>
<tr>
<td>8-core</td>
<td>1.10</td>
<td>1.13</td>
<td>3.0%</td>
</tr>
<tr>
<td>16-core</td>
<td>1.15</td>
<td>1.22</td>
<td>7.3%</td>
</tr>
</tbody>
</table>
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**

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

Xiaoyang Lu, Rujia Wang, Xian-He Sun
xlu40@hawk.iit.edu
FAQs
Pure Miss Contribution (PMC) - Predictability

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

<table>
<thead>
<tr>
<th>PMC$_\delta$</th>
<th>403</th>
<th>429</th>
<th>433</th>
<th>436</th>
<th>437</th>
<th>450</th>
<th>459</th>
<th>462</th>
<th>470</th>
<th>473</th>
<th>482</th>
<th>603</th>
<th>621</th>
<th>623</th>
<th>649</th>
<th>654</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,50)</td>
<td>89.40%</td>
<td>62.63%</td>
<td>64.02%</td>
<td>79.57%</td>
<td>68.19%</td>
<td>60.29%</td>
<td>57.74%</td>
<td>62.17%</td>
<td>59.99%</td>
<td>79.23%</td>
<td>57.72%</td>
<td>60.69%</td>
<td>64.77%</td>
<td>63.31%</td>
<td>50.66%</td>
<td>64.17%</td>
</tr>
<tr>
<td>[50,100)</td>
<td>5.89%</td>
<td>16.49%</td>
<td>14.52%</td>
<td>10.00%</td>
<td>18.70%</td>
<td>16.78%</td>
<td>14.95%</td>
<td>13.11%</td>
<td>16.25%</td>
<td>10.00%</td>
<td>18.18%</td>
<td>15.82%</td>
<td>15.24%</td>
<td>14.80%</td>
<td>19.85%</td>
<td>14.65%</td>
</tr>
<tr>
<td>[100,150)</td>
<td>5.56%</td>
<td>12.23%</td>
<td>12.27%</td>
<td>5.42%</td>
<td>9.33%</td>
<td>12.58%</td>
<td>11.00%</td>
<td>15.86%</td>
<td>7.22%</td>
<td>6.04%</td>
<td>12.93%</td>
<td>7.59%</td>
<td>9.09%</td>
<td>14.21%</td>
<td>16.87%</td>
<td>13.95%</td>
</tr>
<tr>
<td>≥ 150</td>
<td>1.15%</td>
<td>8.65%</td>
<td>9.18%</td>
<td>4.96%</td>
<td>3.79%</td>
<td>10.36%</td>
<td>16.30%</td>
<td>8.87%</td>
<td>16.56%</td>
<td>4.67%</td>
<td>11.18%</td>
<td>15.90%</td>
<td>10.90%</td>
<td>7.68%</td>
<td>12.62%</td>
<td>7.23%</td>
</tr>
</tbody>
</table>

The majority of PMC$_\delta$ values are less than 50 cycles

The median PMC$_\delta$ of each workload is low

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

<table>
<thead>
<tr>
<th></th>
<th>LRU</th>
<th>SHiP++</th>
<th>Hawkeye</th>
<th>Glider</th>
<th>M-CARE</th>
<th>CARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pMR</td>
<td>0.56</td>
<td>0.52</td>
<td>0.51</td>
<td>0.50</td>
<td>0.52</td>
<td>0.50</td>
</tr>
<tr>
<td>PMC</td>
<td>114.46</td>
<td>97.98</td>
<td>99.44</td>
<td>101.43</td>
<td>97.80</td>
<td>95.11</td>
</tr>
</tbody>
</table>

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