On the Theory and Potential of LRU-MRU Collaborative Cache Management

Xiaoming Gu and Chen Ding

06.04.2011
Outline

• Introduction
• Theoretical properties of LRU-MRU cache
• Potential of LRU-MRU collaborative caching
• Summary
Motivation

• Cache management is important
• the disparity between CPU and main memory
• an off-chip memory access (aka. cache miss) is very slow
• use on-chip cache to overcome

How to use cache more effectively?
Cache Replacement Algorithm

- A cache replacement algorithm decides which data are evicted
- LRU
  - victim is the one at LRU position
  - deployed in real cache but not optimal
- OPT
  - victim is the one that will be reused in the farthest future
  - optimal but not practical
The Gap between LRU and OPT

- Gradual change for the OPT miss ratio curve
- abrupt for LRU
- Non-uniform gap
- can be very large

BRIDGE THE GAP

SOR from SciMark 2.0
Collaborative Caching

• Collaborative caching
  • the term was coined by Wang et al. in 2002
  • hardware provides multiple caching methods
  • software decides the right caching method for every access
LRU-MRU Collaborative Caching

- Two caching methods: LRU & MRU

**Hit Case**

LRU: 
- $S_1$
- $S_2$
- $S_3(w)$
- $\vdots$
- $S_{m-1}$
- $S_m$

MRU: 
- $S_1$
- $S_2$
- $S_3(w)$
- $\vdots$
- $S_{m-1}$
- $S_m$

**Miss Case**

LRU: 
- $S_1$
- $S_2$
- $S_3$
- $\vdots$
- $S_{m-1}$
- $S_m$

MRU: 
- $S_1$
- $S_2$
- $S_3$
- $\vdots$
- $S_{m-1}$
- $S_m$

$w$
Outline

• Introduction
• Theoretical properties of LRU-MRU cache
• Potential of LRU-MRU collaborative caching
• Summary
Inclusion Property

• Inclusion property: the content of a smaller cache is always contained in a larger cache [Mattson et al., 1970]

• cache miss ratio keeps non-increasing with larger cache sizes

• LRU & OPT both have inclusion property
LRU-MRU Cache Has Inclusion Property

- Inductively proved that inclusion property is satisfied
  
  \[ |C_1| < |C_2| \]
  
  \[ \text{content}(C_1) \subseteq \text{content}(C_2) \]
  
  after every access \( a_i \)

- the base step
- the inductive step
Stack Distance

- Stack distance is the **minimal** cache size to make an access become a hit
- inclusion property is the precondition
- One-pass stack distance analyzer
  - simulates all cache sizes at the same time
  - the core is to maintain a priority list
    - LRU: the priority is the current access time
    - OPT: the priority is the next access time
An Example of LRU Stack Distance Computation

<table>
<thead>
<tr>
<th>access No.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td>e</td>
<td>b</td>
<td>e</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

7 misses when cache size = 3
The Algorithm for LRU-MRU Stack Distance

• Based on the general one [Mattson et al., 1970]
• The most significant change---the priority is a variant of access time
  • the current access time for LRU
  • the negation of the current access time for MRU
An Example of LRU-MRU Stack Distance Computation

<table>
<thead>
<tr>
<th>access No.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td>e</td>
<td>b</td>
<td>e</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>LRU or MRU?</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
</tr>
</tbody>
</table>

6 misses when cache size = 3
Outline

• Introduction

• Theoretic properties of LRU-MRU cache

• Potential of LRU-MRU collaborative caching

• Summary
LRU-MRU Cache Can Be Optimal

- Do MRU selection based on an OPT simulation
  - at the beginning, all accesses use LRU
  - at an eviction, the most recent access to the victim is selected to use MRU

This LRU-MRU cache is optimal [Gu et al., 2008]
• Program-assisted Cache Management
  • do LRU-MRU collaborative caching at program level
    • restriction: run-time accesses from the same static memory reference must use the same access type
  • a simple model to select static memory references to use MRU
    • based on the optimal LRU-MRU selection
    • a reference has an **MRU ratio** of $x$ if $x$ fraction of accesses by this reference are selected to use MRU in the optimal LRU-MRU selection
  • select a static memory reference to use MRU if **MRU ratio** $\geq 50\%$
Testing Configurations

• Collect memory traces
  • do instrumentation in LLVM
• Cache simulator
  • single-level fully-associative cache
  • cache line size: 8 bytes
  • cache sizes: from 1KB to the double size of data set
173.applu

- Avg. OPT imprv.: 17%
- Avg. PACMAN imprv.: 8.2%

OPT imprv.(C) = \frac{miss_{LRU}(C) - miss_{OPT}(C)}{miss_{LRU}(C)}

PACMAN imprv.(C) = \frac{miss_{LRU}(C) - miss_{PACMAN}(C)}{miss_{LRU}(C)}
## Overall Results

<table>
<thead>
<tr>
<th></th>
<th>the OPT improv. over LRU</th>
<th>the PACMAN improv. over LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>average</td>
<td>largest</td>
</tr>
<tr>
<td>SOR</td>
<td>25%</td>
<td>91%</td>
</tr>
<tr>
<td>171.swim</td>
<td>19%</td>
<td>64%</td>
</tr>
<tr>
<td>172.mgrid</td>
<td>31%</td>
<td>60%</td>
</tr>
<tr>
<td>173.applu</td>
<td>17%</td>
<td>50%</td>
</tr>
<tr>
<td>183.equake</td>
<td>22%</td>
<td>54%</td>
</tr>
<tr>
<td>189.lucas</td>
<td>34%</td>
<td>67%</td>
</tr>
<tr>
<td>410.bwaves</td>
<td>25%</td>
<td>80%</td>
</tr>
<tr>
<td>433.milc</td>
<td>31%</td>
<td>62%</td>
</tr>
<tr>
<td>434.zeusmp</td>
<td>12%</td>
<td>79%</td>
</tr>
<tr>
<td>437.leslie3d</td>
<td>27%</td>
<td>50%</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td><strong>24%</strong></td>
<td><strong>66%</strong></td>
</tr>
</tbody>
</table>

- **Half** possible improvement is achieved in average
The Impact of MRU Ratio Threshold

- The threshold matters
  - PACMAN improv.=4.6% if MRU ratio threshold=50%
  - PACMAN improv.=20% if MRU ratio threshold=30% or 35%

PACMAN with different MRU ratio thresholds

173.applu, cache size=512KB
The Effect of Different Inputs

- Similar improvement showed with different inputs
- Possible to enable a feedback-based optimization from a training run with a smaller input
Summary

• LRU-MRU collaborative caching
  • holds inclusion property
    • an algorithm to compute the LRU-MRU stack distance
  • has promising potential
  • achieves half possible improvement with 10 benchmarks

Q/A