Efficient Locality Approximation from Time

Xipeng Shen
The College of William and Mary

Joint work with Jonathan Shaw
at Shaw Technologies Inc., Tualatin, OR
Locality is Important

• Traditional reasons: memory wall, deeper memory hierarchy.
• New trends: more common and complex cache sharing.
A Locality Model

- Reuse distance (LRU stack distance)
  - Def: number of distinct elements between reuse
    - [Mattson et. al. 1970]
    - $Rd = 2$

- Connection with cache
  - $Rd > \text{cache size}$ → a likely cache miss
## Appeal of Reuse Distance

**Rd = 2**

### More rigorous & machine independent

<table>
<thead>
<tr>
<th></th>
<th>Reuse distance</th>
<th>Cache miss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularity</strong></td>
<td>Point to point</td>
<td>✔️ Interval</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Exact</td>
<td>✔️ Average</td>
</tr>
<tr>
<td><strong>Adaptive to cache sizes</strong></td>
<td>Yes</td>
<td>✔️ No</td>
</tr>
</tbody>
</table>

Xipeng Shen  @  College of William & Mary
Many Uses in Research

- Study cache reuses [Ding+:SC04, Huang+:ASPLOS05]
- Guide and evaluate program transformation [Almasi+:MSP02, Ding+:PLDI03]
- Predict locality phases [Shen+:ASPLOS04]
- Discover locality-improving refactoring [Beyls+:HPCC06]
- Model cache sharing [Chandra+:HPCA05, Jiang+:EuroPar08]
- Insert cache hints [Beyls+:JSA05]
- Manage superpages [Cascaval+:PACT05]
- Guide memory disambiguation [Fang+:PACT05]
- Predict program performance [Marin+:SIGMETRICS04, Zhong+:TOC07]
- Model reference affinity [Zhong+:PLDI04]
- ...

Xipeng Shen  @  College of William & Mary
# Properties of Reuse Distance

<table>
<thead>
<tr>
<th></th>
<th>Reuse distance</th>
<th>Cache miss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularity</strong></td>
<td>Point to point</td>
<td>✔ Interval</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Exact</td>
<td>✔ Average</td>
</tr>
<tr>
<td><strong>Adaptive to cache sizes</strong></td>
<td>Yes</td>
<td>✔ No</td>
</tr>
<tr>
<td><strong>Practical uses</strong></td>
<td>Few</td>
<td>✔ Many</td>
</tr>
</tbody>
</table>

- **Our objective:** Making reuse distance faster to obtain.
Outline

• Reuse distance measurement
• Efficient approximation of reuse distance (17X speedup)
  • Algorithmic extensions (1 order of magnitude less)
  • Implementation optimizations (3.3X speedup)
• Evaluation tool: trace generator
• Evaluation
• Conclusions
Previous Research

T: execution time
N: data size

1970 (Mattson+)
1975 (Bennett+)
1981 (Olken)
1991 (Kim+)
1993 (Sugumar+)
2002 (Almasi+)
2003 (Ding+)

O \( (T \cdot N) \)
O \( (T \cdot \log \log N) \)

But measuring 1-min execution still takes several hours!
A Different Path

• Key obstacle: Counting out repetitive references in an arbitrarily long interval.

• Previous methods
  implement the definition of reuse distance:
  “Counting” distinct data.

• Our approach
  uses some “cheap” program behavior to statistically approximate reuse distance.
The “Cheap” Behavior

- **Time distance (TD)**
  - *Def*: number of elements between reuse.

```
Td = 5
bcaacdb
```

- **Reuse distance (RD)**
  - *Def*: number of distinct elements between reuse.

```
Rd = 2
bcaacdb
```
**TD $\rightarrow$ RD : Intuition**

- **Is it possible?**
  
  TD=5 $\rightarrow$ RD=1, 2, 3, or 4 ?
  
  
  No idea.

- **What if we know the following:**
  
  totally 4 reuses; one TD is 5, three TDs are all 1.

  TD histogram

  RD=1
Problem to Solve

TD histogram

<table>
<thead>
<tr>
<th>Time Distance</th>
<th>Reference %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>18</td>
</tr>
<tr>
<td>512</td>
<td>13</td>
</tr>
<tr>
<td>2K</td>
<td>6</td>
</tr>
<tr>
<td>8K</td>
<td>2</td>
</tr>
<tr>
<td>32K</td>
<td>1</td>
</tr>
<tr>
<td>128K</td>
<td>0</td>
</tr>
</tbody>
</table>

RD histogram

<table>
<thead>
<tr>
<th>Reuse Distance</th>
<th>Reference %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>128</td>
<td>5</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
</tr>
<tr>
<td>512</td>
<td>0</td>
</tr>
</tbody>
</table>

Probabilistic Model

“Locality approximation from time”, Shen+: POPL’07.
Connection between TD and RD

[Shen+:POPL’07]

Expectation of the probability for a variable to appear in a $\Delta$-long interval:

$$P_T(\delta)$$

TD histogram

Reference %

Time Distance

Xipeng Shen  @  College of William & Mary
Connection between TD and RD

[Shen+:POPL’07]

Expectation of the probability for a variable to appear in a $\Delta$-long interval:

$$p(\Delta) = \sum_{\tau=1}^{\Delta} \sum_{\delta=\tau+1}^{T} \frac{1}{N-1} P_T(\delta)$$

Probability for the interval to have $k$ distinct variables (Bernoulli process):

$$p(k,\Delta) = \binom{N}{k} p(\Delta)^k (1-p(\Delta))^{N-k}$$
Compute $p(\Delta)$

- $p(\Delta)$: a variable to appear in a $\Delta$-long interval.

  $$p(\Delta) = \sum_{\tau=1}^{\Delta} \sum_{\delta=\tau+1}^{T} \frac{1}{N-1} P_T(\delta)$$

- $p'(\tau)$: a variable’s last access before $t$ is at time $(t-\tau)$.

  $$p(\Delta) = \sum_{T=1}^{\Delta} p'(\tau)$$

- The following is proved in [Shen+:POPL07]

  $$p'(\tau) = \sum_{\delta=\tau+1}^{T} P_T(\delta)/(N-1)$$
Implementation Issues

• Scale: The model applies to every access, but not to histograms.
  • The width of a bar must be 1.

• Overhead: high cost in measuring time distance.
  • Bookkeeping and buffer boundary checking at every memory access.
Outline

• Reuse distance measurement
• Efficient approximation of reuse distance (17X speedup)
  • Algorithmic extensions (1 order of magnitude less)
  • Implementation optimizations (3.3X speedup)
• Evaluation tool: trace generator
• Evaluation
• Conclusions
Algorithm Extension

- Basic extension: assume all references in a bar have the same \( p(\Delta) \), denoted as \( p(b_i) \).

\[
p(\Delta) = \sum_{\tau=1}^{\Delta} \sum_{\delta=\tau+1}^{T} \frac{1}{N-1} P_T(\delta) \quad \Rightarrow \quad p(b_i) = \sum_{\tau=1}^{2} \sum_{\delta=\tau+1}^{T} \frac{1}{N-1} P_T(\delta)
\]

\[
p(k, \Delta) = \binom{N}{k} p(\Delta)^k (1-p(\Delta))^{N-k} \quad \Rightarrow \quad p(k, b_i) = \binom{N}{k} p(b_i)^k (1 - p(b_i))^{N-k}
\]

- Time complexity: \( O(L_T^3) \)

\( L_T \): number of bars in a TD histogram.
Algorithmic Optimizations

- Decompose $p(b_i)$ into 3 sub-equations to remove redundant computations.

\[
p(b_i) = P_2(b_i)/2 + \sum_{j=0}^{i-1} P_2(b_j)
\]

\[
P_2(b_i) = \left[ \sum_{j=i+1}^{L_T} P_1(b_j) \frac{\overrightarrow{b_i} - \overrightarrow{b_i}}{\overrightarrow{b_j} - \overrightarrow{b_j}} \frac{1}{\tau - 1} \right]
\]

\[
+ P_1(b_i) \frac{1}{\overrightarrow{b_i} - \overrightarrow{b_i}} \sum_{\tau=\overrightarrow{b_j}}^{\overrightarrow{b_i}-1} \frac{\tau - \overrightarrow{b_i}}{\tau - 1}
\]

\[
P_1(b_i) = \frac{\overrightarrow{b_i} + \overrightarrow{b_i} - 3}{2(N - 1)} P_T(b_i).
\]
Algorithmic Optimizations

• Further optimizations
  • mathematical approximation
    \[
    \sum_{i=m_1}^{m_2} \frac{1}{i} \approx \ln \frac{m_2}{m_1 - 0.5}
    \]
  • statistical approximation
    • Normal distribution with table-lookup for binomial distribution calculation

• Time complexity
  \[
  O(L_T^3) \quad \Rightarrow \quad O(L_T^2)
  \]

Details in [Shen+:LCPC’08].
Measure TD

• Invocation to record function after every load/store.

Basic record function

Procedure RecordMemAcc (addr)
    buffer [index++] = addr;
    if (index== BUFFERSIZE) then
        ProcessBuff();
    endif
end

After optimization

Procedure RecordMemAcc (addr)
    buffer [index++] = addr;
end

• Fewer operations
• Fewer branch miss predictions
• Amenable to runtime inlining
• 3.3X speedup
MMU Control

• Typical scheme:
  • Control page permission & modify registers
  • Not portable across architectures.

• Our approach:
  • 2-page scheme
  • Close & open permissions of the final 2 pages alternatively

Details in [Shen+:LCPC’08].
An option to resume: change target location by modifying register values. **Not portable.**
2-Page Scheme for Using MMU

Buffer

Page 1
Page 2
Page 3
...
Page N-1
Final page

Process buffer
2-Page Scheme for Using MMU

- Using the last 2 pages alternatively to signal the end of buffer
- Remove boundary check
- More portable

Limitations
- 2 page space waste
- 1 data loss per buffer

Buffer
Outline

• Reuse distance measurement
• Efficient approximation of reuse distance
  • Algorithmic extensions
  • Implementation optimizations
• Evaluation tool: trace generator
• Evaluation
• Conclusions
A Reverse Problem: Trace Generation

- Reuse distance measurement or approximation
  
  Trace $\rightarrow$ Reuse distance

- Trace generator
  
  Trace $\leftarrow$ Reuse distance

Use: for evaluating locality techniques on various reuse patterns.
A Reverse Problem: Trace Generation

- Technique: a stochastic process
- Property: The generated trace meets input requirements (proof in [Shen+:LCPC’08])

RD histogram
trace length $T$
data size $N$

trace generator

reference trace $v_1, v_2, v_3, v_{12}, \ldots, v_{89}$
Outline

- Reuse distance measurement
- Efficient approximation of reuse distance
  - Algorithmic extensions
  - Implementation optimizations
- Evaluation tool: trace generator
- Evaluation
- Conclusions
Evaluation (Pulse-like reuse distributions)

Time Distance Histogram

Reuse Distance Histogram
RD Approximation on Synthetic Traces

<table>
<thead>
<tr>
<th></th>
<th>acc%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal (var=20)</td>
<td>92.8</td>
</tr>
<tr>
<td>Normal (var=100)</td>
<td>96.3</td>
</tr>
<tr>
<td>Normal (var=200)</td>
<td>95.8</td>
</tr>
<tr>
<td>Exponential</td>
<td>96.9</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>95.5</strong></td>
</tr>
</tbody>
</table>
## Evaluation on Real Benchmarks

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Intel Xeon 2GHz</td>
</tr>
<tr>
<td><strong>Instrumentor</strong></td>
<td>PIN 3.4</td>
</tr>
<tr>
<td><strong>Compiler</strong></td>
<td>GCC 3.4.4 (&quot;-O3&quot;)</td>
</tr>
<tr>
<td><strong>HW perf. measure</strong></td>
<td>PAPI 3.2</td>
</tr>
<tr>
<td><strong>Benchmarks</strong></td>
<td>SPEC CPU2000 ref</td>
</tr>
</tbody>
</table>

**Baseline:** Ding+:PLDI’03.
## Results on Real Benchmarks

<table>
<thead>
<tr>
<th>Programs</th>
<th>Element</th>
<th>acc%</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>acc%</td>
<td>89.0</td>
<td>21.2X</td>
</tr>
<tr>
<td>gzip</td>
<td>acc%</td>
<td>99.0</td>
<td>19.0X</td>
</tr>
<tr>
<td>mcf</td>
<td>acc%</td>
<td>42.6</td>
<td>8.3X</td>
</tr>
<tr>
<td>twolf</td>
<td>acc%</td>
<td>88.2</td>
<td>5.9X</td>
</tr>
<tr>
<td>ammp</td>
<td>acc%</td>
<td>95.8</td>
<td>14.3X</td>
</tr>
<tr>
<td>applu</td>
<td>acc%</td>
<td>86.1</td>
<td>19.0X</td>
</tr>
<tr>
<td>equake</td>
<td>acc%</td>
<td>57.6</td>
<td>23.7X</td>
</tr>
<tr>
<td>mesa</td>
<td>acc%</td>
<td>97.3</td>
<td>26.3X</td>
</tr>
<tr>
<td>mgrid</td>
<td>acc%</td>
<td>89.7</td>
<td>20.6X</td>
</tr>
<tr>
<td>Average</td>
<td>acc%</td>
<td>82.8</td>
<td>17.6X</td>
</tr>
</tbody>
</table>
Uses in Cache Miss Rate Estimation

For all benchmarks, error < 1.76%, average error = 0.42%.

(Details in Shen+:TR902)
Conclusions

• Strong connection exists between time and locality.
• Reuse distance can be approximated from time efficiently.

* Details in:
  “Locality approximation from time”, POPL’07.
  “Adaptive software speculation for enhancing the efficiency of behavior-oriented parallelization”, LCPC’08.
Acknowledgment

- Chen Ding (U of Rochester) suggested the use of 2 pages for MMU control.
- Brian Meeker (U of Rochester) collected preliminary data in the early stage.
- Supported by NSF.