Using a User-Level Memory Thread for Correlation Prefetching

Yan Solihin\textsuperscript{1}  Jaejin Lee\textsuperscript{2}  Josep Torrellas\textsuperscript{1}

\textsuperscript{1} University of Illinois at Urbana-Champaign
\textsuperscript{2} Michigan State University

\{solihin,torrellas\}@cs.uiuc.edu  jlee@cse.msu.edu
Motivation

- Processor-memory speed gap growing
- Memory wall: memory becoming the bottleneck
- Challenges in prefetching irregular apps
- Want to have prefetching:
  - Effective for both irregular and regular apps
  - Customizable to applications
  - No major hardware cost
  - No compiler support
Intelligent Memory

Can we use ULMT for Prefetching?
Contribution

**Correlation prefetching in software as a ULMT in memory**

- **Widely applicable**: prefetches irregular apps
- **Flexible**: prefetching can be customized for applications
- **Inexpensive**: few hardware changes, no compiler support needed
- **Effective**:
  - Conventional prefetching only: *1.22* speedup
  - ULMT only: *1.32* speedup
  - Conventional + ULMT: *1.46* speedup
  - Conventional + ULMT + Customization: *1.53* speedup
Outline

- Overview of correlation prefetching
- ULMT
- Prefetching algorithms
- Results
Correlation Prefetching [Joseph&Grunwald,97]

- Records sequences of miss addresses in a correlation table
- When the head of a sequence is seen, prefetch the rest
- Effective: any miss patterns that repeat

(1) $a[4*(i++)]$

(2) $a[\text{foo}(i)]$

(3) $\rightarrow$

\begin{align*}
\text{solihin, lee, torrellas} & \quad \text{Using ULMT for Correlation Prefetching} & \quad \text{isca'02}
\end{align*}
Proposal: ULMT + Correlation Pref

<table>
<thead>
<tr>
<th>Aspects</th>
<th>Past Correlation Pref</th>
<th>ULMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefetcher</td>
<td>Custom hardware[1-4]</td>
<td>General purpose core</td>
</tr>
<tr>
<td>Location</td>
<td>On-chip at L1[1,2,4] or in Memory (buffering [3])</td>
<td>Memory (prefetching to L2)</td>
</tr>
<tr>
<td>Table cost</td>
<td>High (1 – 7.6 MB SRAM) [1-4]</td>
<td>Low (DRAM), dynamically allocated</td>
</tr>
<tr>
<td>Multiple apps</td>
<td>Cross-pollution [1-4]</td>
<td>One instance per app</td>
</tr>
<tr>
<td>Custom &amp; control</td>
<td>No [1-4]</td>
<td>Yes, by application/OS</td>
</tr>
</tbody>
</table>

Timeline of ULMT Prefetching

Miss address observed

Prefetch addresses issued to Memory

ULMT free

Prefetching step

Update step

Response time

Occupancy time

Ideal algorithm:

– lowest response time

– occupancy time < time between misses
## Correlation Table

### Basic Organization [JG97]

Addr of immediate successors

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NumSucc = 2

### Our Advanced Organization

Addr of next immediate successors

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next²</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NumSucc = 2

NumLevels = n

---

[JG97] Joseph & Grunwald, ISCA 97

solihin, lee, torrellas, Using ULMT for Correlation Prefetching, isca'02
Update Step

### Basic Organization [JG97]

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

### Our Advanced Organization

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next²</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current miss

A, B, C, A, D, C, ...

solihin, lee, torrellas  Using ULMT for Correlation Prefetching  isca'02
Update Step

Basic Organization [JG97]  Our Advanced Organization

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current miss

A, B, C, A, D, C, ...

solihin, lee, torrellas  Using ULMT for Correlation Prefetching  isca'02
### Update Step

**Basic Organization [JG97]**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

**Our Advanced Organization**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current miss

A, B, C, A, D, C, ...
### Update Step

#### Basic Organization [JG97]

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>

#### Our Advanced Organization

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next²</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
Update Step

Basic Organization [JG97]

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>

Our Advanced Organization

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next²</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Current miss

A, B, C, A, D, C, ...

solihin, lee, torrellas
Using ULMT for Correlation Prefetching

isca'02
Update Step

Basic Organization [JG97]  Our Advanced Organization

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Next</th>
<th>Next²</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Current miss

A, B, C, A, D, C, ...

solihin, lee, torrellas  Using ULMT for Correlation Prefetching  isca'02
Prefetching Step

**Basic Organization [JG97]**

On miss A

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chaining

**Basic**: 1 miss ⇒ next misses
- no far ahead prefetching
- low coverage and late prefetches

**Basic + Chaining**: high response time

**Our Advanced Organization**

On miss A

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Advanced**: 1 miss ⇒ next, next², …
- far ahead prefetching
- high coverage
- more timely prefetches
- low response time
- more accurate

Basic + Chaining: high response time

solihin, lee, torrellas Using ULMT for Correlation Prefetching isca'02
Advanced Improves Accuracy

Basic Organization [JG97]    Our Advanced Organization

\[
\begin{array}{c|c|c}
A & B & \\
B & C & E \\
C & D & \\
D & & \\
E & F & \\
\end{array}
\quad
\begin{array}{c|c|c|c}
A & B & & C & D & \\
B & C & E & D & F & \\
C & D & & \\
D & & & \\
E & F & & \\
\end{array}
\]

On miss

A, B, C, D, …, A, B, C, D, …, A, B, C, D, …, B, E, F, …,

⇒ ideally prefetch B, C, D
Advanced Improves Accuracy

Basic Organization [JG97]

On miss A

B prefetched (Basic)
B, C, E, F prefetched (Basic+Chaining)

Our Advanced Organization

On miss A

B, C, D prefetched

⇒ ideally prefetch B, C, D

A, B, C, D, …, A, B, C, D, …, A, B, C, D, …, B, E, F, …,
Simulation Environment

- **Main processor:**
  - 1.6 GHz, 6-issue out-of-order
  - L1: 2-way 16 KB; L2: 4-way 512 KB
  - Mem: 152 ns round-trip (RT)

- **Memory processor:**
  - 800 MHz, 2-issue out-of-order, integer only
  - L1: 2-way 32 KB
  - Mem: 63 ns RT (in NorthBridge), 35 ns cycle RT (in DRAM)

- **Bus bandwidth:** 3.2 GB/sec

- **Correlation table:** app-specific (64K entries, 3 levels, 2 successors)

- **Applications:** memory intensive, mostly irregular
  - Specint2000, Specfp2000, NAS, Olden
• Miss sequences are predictable
Execution Time (Mem Proc in DRAM)

- Advanced memory prefetching works

solihin, lee, torrellas  Using ULMT for Correlation Prefetching  isca'02
Customization Delivers Further Gains

Types of Customization:

- CG: ULMT includes n-stream stride prefetcher
- MST: ULMT prefetches up to 4 levels of successors (high predictability)

Simple customization successful in some apps
Other Results

- More on predictability
- Timing analysis
- Prefetching effectiveness
- Bus utilization
- Processor in North Bridge vs. DRAM
  - Comparable speedups
- OS issues
Conclusions

- **Simple Intelligent Memory**: performance significantly
- Correlation prefetching in ULMT is effective
  - Conventional only: 1.22 speedup
  - ULMT only: 1.32 speedup
  - ULMT + Conventional: 1.46 speedup
  - ULMT + Conventional + Customization: 1.53 speedup
- Flexible systems: customizable
- Few hardware changes
- No compiler support needed
Using a User-Level Memory Thread for Correlation Prefetching

Yan Solihin\textsuperscript{1} \quad Jaejin Lee\textsuperscript{2} \quad Josep Torrellas\textsuperscript{1}

\textsuperscript{1} University of Illinois at Urbana-Champaign
\textsuperscript{2} Michigan State University
Slide Map

- (example) (vs. past work)
- (architecture and mechanism)
- (timeline)
- (update and prefetching steps)
- **Accuracy of Chaining vs. Advanced**
- Results (params) (predictability) (execute) (custom)

- **Size of correlation table**
- **Additional Results** (L0pred)
  (Chain vs. Adv) (time between misses) (response time) (MC)
  (Effectiveness) (Bus)
- **OS Issues** (multitasking) (page remap) (sharing)
- **Multichip DRAM**
- (Related work)
## Correlation Table Size

<table>
<thead>
<tr>
<th>App</th>
<th>Basic (MB)</th>
<th>Advanced (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>1.3</td>
<td>1.8</td>
</tr>
<tr>
<td>Equake</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td>FT</td>
<td>5.0</td>
<td>7.0</td>
</tr>
<tr>
<td>Gap</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Mcf</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>MST</td>
<td>5.0</td>
<td>7.0</td>
</tr>
<tr>
<td>Parser</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Sparse</td>
<td>5.0</td>
<td>7.0</td>
</tr>
<tr>
<td>Tree</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>2.7</strong></td>
<td><strong>3.8</strong></td>
</tr>
</tbody>
</table>
Predictability of Miss Addresses

- Miss patterns are application-specific
- Base is able to capture most of the patterns
Advanced vs. Basic+Chaining

advanced is superior in all cases

solihin, lee, torrellas  Using ULMT for Correlation Prefetching  isca'02
Cycles Between Misses

- Occupancy time of prefetch thread must be < 200 cycles
ULMT Response and Occupancy Time

- All algo feasible: Occupancy Time < 200 cycles
- Advanced (≡ Repl) has the best Response Time
DRAM vs. Mem Controller Chip

Normalized Execution Time

N: No Prefetch; +: Mem Proc in DRAM; +2: Mem Proc in MC

solihin, lee, torrellas Using ULMT for Correlation Prefetching isca'02
Prefetching Effectiveness

- Hits
- DelayedHits
- NonPrefMisses
- Replaced
- Redundant

Normalized Effectiveness

Sparse

Tree

Avg7

solihin, lee, torrellas

Using ULMT for Correlation Prefetching

isca'02
Bus Utilization

Advanced: avg increase of 8%
Multitasking Environment

- ULMT is flexible (vs. hardware correlation prefetching)
- Shared correlation table:
  + Smaller table
  - Interference or
  - Cold start at every context switch
- Private correlation table: one instance per application
  + No interference
  - Larger space requirement, but solutions available:
    • Still a small portion of main memory
    • OS imposes quota on total table size through malloc()
    • Dynamic resizing eliminates table fragmentation
Handling Page Mapping Changes

- ULMT is flexible (vs. hardware correlation prefetching):
  - Software solution:
    - OS informs page mapping changes of the main program
    - ULMT remaps or invalidates entries of the old page and (for 8KB-page: overhead ≈ 4 µs - 8 µs)
    - Only tags are changed, successors may be incorrect
  - Hardware solution:
    - A small cache in MC stores invalid pages for each process
    - Prefetches are filtered by this cache
    - If cache overflows, prefetches may be incorrect
  - Lazy solution: ignore page mapping changes
OS Issues

- Scheduling:
  - An instance of ULMT is attached to the main program
  - Scheduled and context switched together
  - States are equivalent

- Sharing and protection
  - ULMT and main program do not share data and instruction
  - Main program is protected: ULMT only reads physical addresses
  - ULMT is protected: it has its own address space
  - OS provides sys calls for main program to control life of ULMT
Multi-chip DRAM

Processor chip
- Main Proc
- L1 $ 
- L2 $

North Bridge Chip
- Memory Controller
- P+$
- DRAM
- P+$
- DRAM
- P+$
- DRAM

Interconnect
Related Work

- Memory-side prefetching
  - Yang&Lebeck (ICS’00), Hughes (MS Thesis’00), Cooksey et.al. (Content-based prefetcher, WIMS’00)

- Correlation-based prefetching
  - Baer (’96), Pomerene (Patent ‘89), Joseph&Grunwald (ISCA’97), Charney&Reeves (TR’95), Alexander&Kedem (HPCA’96), Sherwood et.al. (Micro’00), Lai et.al. (ISCA’01)

- Helper thread
  - Chappel et.al. (SSMT, ISCA’99), Intel’s SP (ISCA’01, HPCA’02), Collins et.al. (MICRO’01), Roth et.al.’s DDMT (HCPA’01),

- Prefetching for LDS
  - Luk&Mowry (ASPLOS’96), Roth et.al. (ASPLOS’98, ISCA’99), Choi et.al. (PACT’01), Hughes (MS Thesis’00), Yang&Lebeck (ICS’00),

- Other
  - Carter et.al. (HPCA’99), Burger et.al. (DataScalar, ISCA’97).