Spatial Memory Streaming
(with rotated patterns)

Michael Ferdman,
Stephen Somogyi, and Babak Falsafi

Computer Architecture Lab at
Carnegie Mellon

© 2006 Stephen Somogyi
The Memory Wall

- Memory latency
  - 100’s clock cycles; improving slowly

- Reduce time stalled on memory
  - Raise memory-level parallelism

- Capture all access patterns
  - Strides
  - Pointers (linked lists, trees)
  - Complex layouts (sparse structs)
Our Observation: Spatial Correlation

Large-scale spatial access patterns
- Irregular layout → non-strided
- Sparse → can’t capture with cache blocks
- But, repetitive → predict to improve MLP
DPC Submission

• Code-correlated spatial patterns
  – Pattern storage independent of dataset size
  – Compulsory misses predictable

• Spatial Memory Streaming
  – Observes and records spatial patterns
  – Upon first access, stream remaining blocks
    • Fetch in parallel → increase MLP
    • Sparse patterns → fetch directly into L1
Outline

• Introduction

• Spatial Correlation

• Spatial Memory Streaming

• Pattern Rotation
Spatial Regions

Logically divide memory into regions

• Identify region by base address
• Fixed-size
  – Simplifies hardware
  – Can represent spatial patterns as bit vectors
Why Exploit Spatial Correlation?

Perfect predictor = one miss per spatial pattern

- Large blocks → prohibitive miss rate at L1 → bandwidth inefficient
- Spatial correlation → opportunity to eliminate misses

© 2006-2009 Stephen Somogyi, Michael Ferdman
How to Exploit Spatial Correlation?

• Patterns are code-correlated

• Use PC to predict patterns
  – Storage independent of dataset size
  – Can predict compulsory misses

But, data layout may not be aligned to region
  – PC is not enough [Kumar 98] [Chen 04]
  – Offset within region identifies alignment

Practical hardware can predict spatial correlation
Outline

• Introduction

• Spatial Correlation

• Spatial Memory Streaming

• Rotated Patterns
Spatial Memory Streaming (SMS)

1. **Observe** pattern during generation
2. **Store** pattern at end of generation
3. **Predict** pattern at subsequent generation

\[
\begin{align*}
\text{PC}_1 &\text{ ld } A+4 \\
\text{PC}_2 &\text{ ld } A \\
\text{PC}_3 &\text{ ld } A+3 \\
\text{evict } A+3 \\
\text{PC}_1 &\text{ ld } B+4 \\
\text{PC}_2 &\text{ ld } B \\
\text{PC}_3 &\text{ ld } B+3 \\
\end{align*}
\]

- **1** observe
- **2** store
- **3** predict

**cache hits**
SMS Hardware Overview

- Core
- L1d
- Active Generation Table
- Pattern History Table

1. observe
2. store
3. predict

Tracks current patterns
Stores observed patterns

Direct access to the Pattern History Table

Stream into hierarchy
Learning Patterns

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Active Generation Table
  - Accumulates patterns
  - 32 ~ 64 entries sufficient
Learning Patterns

PC₁ ld A+4
PC₂ ld A
PC₃ ld A+3
evict A+3

PC₁ ld B+4
PC₂ ld B
PC₃ ld B+3

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>00001000</td>
</tr>
</tbody>
</table>

- First access creates new entry
Learning Patterns

PC₁ ld A+4
PC₂ ld A
PC₃ ld A+3
evict A+3

PC₁ ld B+4
PC₂ ld B
PC₃ ld B+3

• Further accesses accumulate bits in pattern

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>10001000</td>
</tr>
</tbody>
</table>
### Learning Patterns

#### PC1 ld A+4
#### PC2 ld A
#### PC3 ld A+3
#### evict A+3

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>10011000</td>
</tr>
</tbody>
</table>

- Further accesses accumulate bits in pattern
Learning Patterns

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
</table>

- Eviction ends pattern

$10011000$ @ $PC_1/4$
Learning Patterns

Pattern History Table

<table>
<thead>
<tr>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC₁ / 4</td>
<td>10011000</td>
</tr>
</tbody>
</table>

PC₁ ld A+4
PC₂ ld A
PC₃ ld A+3
evict A+3
PC₁ ld B+4
PC₂ ld B
PC₃ ld B+3

• Pattern History Table
  – Stores previously-observed patterns
  – Set-associative: 8-way 2k-entries
Predicting Patterns

- First access looks in Pattern History Table
- Stream predicted blocks into L1 cache
Predicting Patterns

<table>
<thead>
<tr>
<th>Pattern History Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC / off</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>PC₁ / 4</td>
</tr>
</tbody>
</table>

- PC₁ ld A+4
- PC₂ ld A
- PC₃ ld A+3
evict A+3

- PC₁ ld B+4
- PC₂ ld B    cache hit
- PC₃ ld B+3  cache hit

• Subsequent accesses hit in L1 cache
SMS Results (SPEC CPU 2006)

Normalized Execution Time

asiar, bwaves, bzip2, cactusADM, deall, gcc, GemsFDTD, gromacs, h264ref, hmmer, ibm, leslie3d, libquantum, mcf, milc, omnetpp, soplex, xalancbmk, zeusmp

© 2006-2009 Stephen Somogyi, Michael Ferdman
Outline

• Introduction

• Spatial Correlation

• Spatial Memory Streaming

• Rotated Patterns
Our Observation: Rotated Patterns

• PC is insufficient to predict pattern
  – Offset of first access highly variable
  – *But*: Access pattern almost always the same

• Can store “rotated” patterns in PHT
  – Rotate as needed before prediction
Learning Patterns

- Active Generation Table
  - Accumulates patterns
  - 32 ~ 64 entries sufficient
Learning Patterns

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7
evict A+7
PC₁ ld B+2
PC₂ ld B+6
PC₃ ld B+5

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>100000000</td>
</tr>
</tbody>
</table>

- First access creates new entry
- Bits are recorded rotated left by initial offset
Learning Patterns

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7
evict A+7
PC₁ ld B+2
PC₂ ld B+6
PC₃ ld B+5

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>100010000</td>
</tr>
</tbody>
</table>

- Further accesses accumulate bits in pattern
- Bits are recorded *rotated left* by initial offset
Learning Patterns

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7
evict A+7

PC₁ ld B+2
PC₂ ld B+6
PC₃ ld B+5

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PC₁ / 4</td>
<td>100110000</td>
</tr>
</tbody>
</table>

• Further accesses accumulate bits in pattern
• Bits are recorded *rotated left* by initial offset
Learning Patterns

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7

Evict A+7

PC₁ ld B+2
PC₂ ld B+6
PC₃ ld B+5

• Eviction ends pattern

Active Generation Table

<table>
<thead>
<tr>
<th>Region</th>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
</table>

1001100000 @ PC₁

PC only – no offset

© 2006-2009 Stephen Somogyi, Michael Ferdman
Learning Patterns

- Pattern History Table
  - Stores previously-observed patterns
  - Set-associative: 8-way 2k-entries

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7
evict A+7

PC₁ ld B+2
PC₂ ld B+6
PC₃ ld B+5

PC only – no offset

Pattern History Table

<table>
<thead>
<tr>
<th>PC</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC₁</td>
<td>10011000</td>
</tr>
</tbody>
</table>

© 2006-2009 Stephen Somogyi, Michael Ferdman
Predicting Patterns

- First access looks in Pattern History Table
- Stream predicted \textit{rotated} blocks into L1 cache
Predicting Patterns

PC₁ ld A+4
PC₂ ld A+8
PC₃ ld A+7
evict A+7

PC₁ ld B+2
PC₂ ld B+6 cache hit
PC₃ ld B+5 cache hit

• Subsequent accesses hit in L1 cache
### Rotation: Theoretical Benefit

#### Before

**Pattern History Table**

<table>
<thead>
<tr>
<th>PC / off</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC₁ / 4</td>
<td>000010011</td>
</tr>
<tr>
<td>PC₁ / 2</td>
<td>001001100</td>
</tr>
<tr>
<td>PC₁ / 5</td>
<td>100001001</td>
</tr>
<tr>
<td>PC₁ / 1</td>
<td>100110000</td>
</tr>
</tbody>
</table>

#### After

**Pattern History Table**

<table>
<thead>
<tr>
<th>PC</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC₁</td>
<td>100110000</td>
</tr>
</tbody>
</table>

*Rotated patterns ⇒ saves PHT storage*
Rotation: Practical Benefit

Rotated patterns ⇒ saves 2x PHT storage
Rotation: Applicability

• Commercial workloads (e.g., OLTP, web, DSS)
  – Large instruction footprints (>1MB [cidr 07])
  – Benefits from rotation

• Desktop/engineering (e.g., SPEC CPU 2000)
  – Small instruction footprints (fit in L1-I)
  – Unlikely to benefit from rotation [hpca 04]
  – SPEC CPU 2006 very similar to CPU 2000

Need broad range of workloads to observe benefit of rotated patterns
Conclusion

• **Spatial Memory Streaming**
  – Learns large-scale spatial access patterns
  – Streams remaining blocks upon first access in pattern
  – Accurate predictor with small hardware cost

• **Rotated Patterns**
  – Stores one rotated version of spatial pattern per PC
  – Significant reduction in number of patterns
  – Needed in PHT-capacity constrained environment
Questions?

STeMS Project
Spatio-Temporal Memory Streaming
www.ece.cmu.edu/~stems

Computer Architecture Laboratory
Carnegie Mellon University
www.ece.cmu.edu/~calcm