Temporal Memory Streaming

Babak Falsafi

Team: Mike Ferdman, Brian Gold, Nikos Hardavellas, Jangwoo Kim, Stephen Somogyi, Tom Wenisch
Collaborator: Anastassia Ailamaki & Andreas Moshovos

STEMS
Computer Architecture Lab
Carnegie Mellon
http://www.ece.cmu.edu/CALCM

© 2005 Babak Falsafi
The Memory Wall

Logic/DRAM speed gap continues to increase!
Current Approach

Cache hierarchies:
- Trade off capacity for speed
- Exploit “reuse”

But, in modern servers
- Only 50% utilization of one proc. [Ailamaki, VLDB’99]
- Much bigger problem in MPs

What is wrong?
- Demand fetch/repl. data
Prior Work (SW-Transparent)

Prefetching [Joseph 97] [Roth 96] [Nesbit 04] [Gracia Pérez 04]
  - Simple patterns or low accuracy

Large Exec. Windows / Runahead [Mutlu 03]
  - Fetch dependent addresses serially

Coherence Optimizations [Stenström 93] [Lai 00] [Huh 04]
  - Limited applicability (e.g., migratory)

Need solutions for arbitrary access patterns
Our Solution: Spatio-Temporal Memory Streaming

Observation:
- Data spatially/temporally correlated
- Arbitrary, yet repetitive, patterns

Approach → Memory Streaming
- Extract spat./temp. patterns
- Stream data to/from CPU
  - Manage resources for multiple blocks
  - Break dependence chains
- In HW, SW or both
Contribution #1: Temporal Shared-Memory Streaming

• Recent coherence miss sequences recur
  - ≥50% misses closely follow previous sequence
  - Large opportunity to exploit MLP

• Temporal streaming engine
  - Ordered streams allow practical HW
  - Performance improvement:
    - 7%-230% in scientific apps.
    - 6%-21% in commercial Web & OLTP apps.
Contribution #2: Last-touch Correlated Data Streaming

- Last-touch prefetchers
  - Cache block deadtime >> livetime
  - Fetch on a predicted “last touch”
  - But, designs impractical (> 200MB on-chip)

- Last-touch correlated data streaming
  - Miss order ~ last-touch order
  - Stream table entries from off-chip
  - Eliminates 75% of all L1 misses with ~200KB
Outline

- STEMS Overview
- Example Temporal Streaming
  1. Temporal Shared-Memory Streaming
  2. Last-Touch Correlated Data Streaming
- Summary
Temporal Shared-Memory Streaming [ISCA’05]

- Record sequences of memory accesses
- Transfer data sequences ahead of requests

- Accelerates arbitrary access patterns
  - Parallelizes critical path of pointer-chasing
Relationship Between Misses

- Intuition: Miss sequences repeat
  - Because code sequences repeat

- Observed for uniprocessors in [Chilimbi’02]

- Temporal Address Correlation
  - Same miss addresses repeat in the same order

Correlated miss sequence = stream
Relationship Between Streams

• Intuition: Streams exhibit temporal locality
  □ Because working set exhibits temporal locality
  □ For shared data, repetition often across nodes

Node 1: \[ Q \ W \ A \ B \ C \ D \ E \ R \]

Node 2: \[ T \ A \ B \ C \ D \ E \ Y \]

• Temporal Stream Locality
  □ Recent streams likely to recur

Addr. correlation + stream locality = temporal correlation
Memory Level Parallelism

- Streams create MLP for dependent misses

Baseline

CPU → A → B → C

Must wait to follow pointers

Temporal Streaming

CPU → A → B → C

Fetch in parallel

- Not possible with larger windows / runahead

Temporal streaming breaks dependence chains
Temporal Streaming

Node 1

Miss A
Miss B
Miss C
Miss D

Directory

Node 2

Record
Temporal Streaming

Node 1

Miss A
Miss B
Miss C
Miss D

Record

Directory

Node 2

Req. A
Fill A

Miss A
Temporal Streaming

Node 1

Miss A
Miss B
Miss C
Miss D

Locate A

Node 2

Miss A

Locate

Directory

Req. A
Fill A

Stream B, C, D
Temporal Streaming

Node 1

Miss A
Miss B
Miss C
Miss D

Node 2

Dir

Locate A

Stream B, C, D

Record

Locate

Stream

Locate

Stream

Fetch B, C, D

Hit B

Hit C

Req. A

Fill A

Miss A
Temporal Streaming Engine

Record

- Coherence Miss Order Buffer (CMOB)
  - ~1.5MB circular buffer per node
  - In local memory
  - Addresses only
  - Coalesced accesses

Local Memory

CMOB

CPU

$
Temporal Streaming Engine

( Locate

- Annotate directory
  - Already has coherence info for every block
  - CMOB append \(\rightarrow\) send pointer to directory
  - Coherence miss \(\rightarrow\) forward stream request

<table>
<thead>
<tr>
<th>A</th>
<th>shared</th>
<th>Node 4 @ CMOB[23]</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>modified</td>
<td>Node 11 @ CMOB[401]</td>
</tr>
</tbody>
</table>
Temporal Streaming Engine

Stream

- Fetch data to match use rate
  - Addresses in FIFO stream queue
  - Fetch into streamed value buffer

Node i: stream \{A, B, C, ...\}

Stream Queue

F E D C B

Fetch A

Streaming Value Buffer

\begin{center}
\textbf{A} \quad \textbf{data}
\end{center}

\begin{center}
\textbf{Streamed Value Buffer}
\end{center}

\begin{center}
\textbf{~32 entries}
\end{center}
Practical HW Mechanisms

• Streams recorded/followed \textit{in order}
  \begin{itemize}
    \item FIFO stream queues
    \item \textasciitilde32-entry streamed value buffer
    \item Coalesced cache-block size CMOB appends
  \end{itemize}

• Predicts \textit{many misses} from one request
  \begin{itemize}
    \item More lookahead
    \item Allows off-chip stream storage
    \item Leverages existing directory lookup
  \end{itemize}
Methodology: Infrastructure

SimFlex [SIGMETRICS’04]

- Statistically sampling $\rightarrow$ uArch sim. in minutes
- Full-system MP simulation (boots Linux & Solaris)
  - Uni, CMP, DSM timing models
- Real server software (e.g., DB2 & Oracle)
- Component-based $\rightarrow$ FPGA board interface for hybrid simulation/prototyping

Publicly available at

http://www.ece.cmu.edu/~simflex
Methodology: Benchmarks & Parameters

Benchmark Applications
- Scientific
  - em3d, moldyn, ocean
- OLTP: TPC-C 3.0 100 WH
  - IBM DB2 7.2
  - Oracle 10g
- SPECweb99 w/ 16K con.
  - Apache 2.0
  - Zeus 4.3

Model Parameters
- 16 4GHz SPARC CPUs
- 8-wide OoO; 8-stage pipe
- 256-entry ROB/LSQ
- 64K L1, 8MB L2
- TSO w/ speculation
TSE Coverage Comparison

TSE outperforms Stride and GHB for coherence misses
Stream Lengths

- Comm: Short streams; low base MLP (1.2-1.3)
- Sci: Long streams; high base MLP (1.6-6.6)
- Temporal Streaming addresses both cases
• TSE eliminates 25%-95% of coherent read stalls
  6% to 230% performance improvement
TSE Conclusions

• Temporal Streaming
  - Intuition: Recent coherence miss sequences recur
  - Impact: Eliminates 50-100% of coherence misses

• Temporal Streaming Engine
  - Intuition: In-order streams enable practical HW
  - Impact: Performance improvement
    ▪ 7%-230% in scientific apps.
    ▪ 6%-21% in commercial Web & OLTP apps.
Outline

• Big Picture
• Example Streaming Techniques
  1. Temporal Shared Memory Streaming
  2. Last-Touch Correlated Data Streaming
• Summary
Enhancing Lookahead

Observation [Mendelson, Wood&Hill]:
- Few live sets
  - Use until last “hit”
  - Data reuse → high hit rate
  - ~80% dead frames!

Exploit for lookahead:
- Predict last “touch” prior to “death”
- Evict, predict and fetch next line
How Much Lookahead?

Predicting last-touches will eliminate all latency!
Dead-Block Prediction [ISCA’00 & ’01]

- Per-block trace of memory accesses to a block
  + Predicts repetitive last-touch events

Accesses to a block frame

PC₀: load/store A0 (hit)
PC₁: load/store A1 (miss)  First touch
PC₃: load/store A1 (hit)
PC₃: load/store A1 (hit)  Last touch
PC₅: load/store A3 (miss)

Trace = A1 ⊙ (PC₁, PC₃, PC₃)
Dead-Block Prefetcher (DBCP)

- History & correlation tables
  - History ~ L1 tag array
  - Correlation ~ memory footprint
- Encoding — truncated addition
- Two bit saturating counter

- History Table (HT)
  - Latest
  - PC₁, PC₃
- Correlation Table
  - A₁, PC₁, PC₃, PC₃
  - A₃
  - Evict A₁
  - Fetch A₃
DBCP Coverage with Unlimited Table Storage

- High average L1 miss coverage
- Low misprediction (2-bit counters)
Impractical On-Chip Storage Size

% of Achievable Coverage

average
worst-case

On-Chip Correlation Table Size

Needs over 150MB to achieve full potential!
Our Observation:
Signatures are Temporally Correlated

Signatures need not reside on chip

1. Last-touch sequences recur
   • Much as cache miss sequences recur [Chilimbi’02]
   • Often due to large structure traversals

3. Last-touch order ~ cache miss order
   • Off by at most L1 cache capacity

Key implications:

• Can record last touches in miss order
• Store & stream signatures from off-chip
Last-Touch Correlated Data Streaming (LT-CORDS)

• Streaming signatures on chip
  - Keep all sigs. in sequences in off-chip DRAM
  - Retain sequence “heads” on chip
  - “Head” signals a stream fetch

• Small (~200KB) on-chip stream cache
  - Tolerate order mismatch
  - Lookahead for stream startup

DBCP coverage with moderate on-chip storage!
DBCP Mechanisms

All signatures in random-access on-chip table
What LT-CORDS Does

... and only in order

Chips are subsets of off-chip "attatime"

Signatures stored off-chip
LT-CORDS Mechanisms

On-chip storage independent of footprint
Methodology

• SimpleScalar CPU model with Alpha ISA
  - SPEC CPU2000 & Olden benchmarks

• 8-wide out-of-order processor
  - 2 cycle L1, 16 cycle L2, 180 cycle DRAM
  - FU latencies similar to Alpha EV8
  - 64KB 2-way L1D, 1MB 8-way L2

• LT-CORDS with 214KB on-chip storage
• Apps. with significant memory stalls
LT-CORDS vs. DBCP Coverage

LT-CORDS reaches infinite DBCP coverage
LT-CORDS Speedup

LT-CORDS hides large fraction of memory latency
LT-CORDS Conclusions

- **Intuition:** Signatures temporally correlated
  - Cache miss & last-touch sequences recur
  - Miss order ~ last-touch order

- **Impact:** eliminates 75% of all misses
  - Retains DBCP coverage, lookahead, accuracy
  - On-chip storage indep. of footprint
  - 2x less memory stalls over best prior work
For more information

Visit our website:
http://www.ece.cmu.edu/CALCM