Improving the Effectiveness of Context-based Prefetching with Multi-order Analysis

Yong Chen, Oak Ridge National Laboratory
Huaiyu Zhu, Illinois Institute of Technology
Hui Jin, Illinois Institute of Technology
Xian-He Sun, Illinois Institute of Technology
Outline

- Data-access bottleneck problem in high-end computing
- Context-based data prefetching and limitations
- Multi-order context-based prefetching
- Simulation evaluation and results
- Conclusion and future work
High-End Processor Architecture Evolution

- Multicore/manycore architecture
- GPGPU architecture
- Integrate multiple processing units into a single chip
- Conquer limitations of uncore architecture
  - Pipeline depth (limited ILP)
  - Frequency
  - Power consumption
- Cost-effective architecture
  - Pollack’s Rule
- Substantially increase computing capability on chip
Growing Gap between Computing and Data Access

- **Data-Access Wall**: huge gap between computing and data access performance
- **Memory access**:
  - 0.3 cycles, 1980
  - 200 cycles, 2010
- Data-access speed is “hundred times slower”, in CPU cycles, than their ancestor in 1980!
- **Limits the sustained performance**
Bridging Gap: Memory Hierarchy + Prefetching

- Memory hierarchy model
  - Principle of locality, works well if spatial or temporal locality exist
  - Large amount of off-chip accesses exist due to
    - Limited cache capacity, esp. divided by more and more number of cores
    - Many noncontiguous access patterns exist in HEC applications
  - Off-chip accesses cause long data-access latency

- Data prefetching
  - Predict and fetch required data in advance
  - Effectively reduce data-access latency
  - Especially when applications lack temporal/spatial locality
Context-based Data Prefetching

- Conventional data prefetching
  - Sequential/stream prefetching
  - Stride prefetching
  - Markov prefetching
  - Distance/delta prefetching
- Context-based data prefetching
  - Predicts next value based on *k* number of preceding values (context)
    - *k*: “order”, the length of context
  - A more general data prefetching mechanism
  - Representative examples
    - Finite Context Method (FCM) prefetcher
    - Differential Finite Context Method (DFCM) prefetcher
    - Distance/delta prefetching: use strides as context, order-1

Value Sequence: $a a b c a a a b c a a a$

<table>
<thead>
<tr>
<th>Context</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Prediction: $a$
Limitations of Existing Context-based Prefetcher

- Consider **single order only**
- A single-order approach limits the prefetching coverage
  - Prefetching accuracy: the percentage of correct prefetches
  - Prefetching coverage: the percentage of reduced misses
  - Prefetching effectiveness: the production of accuracy and coverage
- Limited coverage leads to limited prefetch effectiveness
Multi-Order Context-based (MOC) Prefetching

- Explore **multiple order analysis** for context-based prefetching
- Goal: **increase coverage to achieve better overall effectiveness**
- MOC prefcher: three-level table organization
  - PC (Program Counter)/Address index table: locate DAH table with PC/Addr
  - Data Access History (DAH) table: data referencing history
  - Data Access Prediction (DAP) table: predictions and confidence
Three-Level Table Organization

- **PC/Address Index Tables (PIT/AIT)**
  
<table>
<thead>
<tr>
<th>PC</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addr</td>
<td>Index</td>
</tr>
</tbody>
</table>
  
  PIT Table Entry
  
  AIT Table Entry

- **Data-Access History (DAH) table**
  
  - A FIFO structure and stores the detailed data access history information
  - The index tables are used to provide quick access to DAH with a “key”

<table>
<thead>
<tr>
<th>PC</th>
<th>ADDR</th>
<th>PC_CHAIN</th>
<th>ADDR_CHAIN</th>
</tr>
</thead>
</table>

  DAH Table Entry

- **Data-Access Prediction (DAP) table**
  
  - Stores the prediction associated with the context analysis
  - Table entry is located with a hashed form of context (FS-5 hash)

<table>
<thead>
<tr>
<th>PREDICTION</th>
<th>CONFIDENCE</th>
</tr>
</thead>
</table>

  DAP Table Entry
MOC Prefetching Mechanism

- On a cache miss
  - Look up in PC/Addr index table
  - Follow PC/Addr chain to form order-k context
  - Feed into hash function unit and generates a hashed form
  - Look up local DAP table with the hashed context
  - If an entry is found, prefetch is generated
  - The confidence counter reflects the confidence of the prediction
  - The counter value increases upon a correct prediction and decreases upon a false prediction
Selection of Multiple Orders

- Major constraint of the order selection comes from storage
- Storage requirement can be characterized with table entry consumption in an ideal case
- Higher order results more table entries/hash conflicts
- Adopt order-0, order-1 and order-2 in current design
- DAH table supports order-0 and order-1 analysis natively

Table Entry Consumption for Different Orders in an Ideal Case
Multi-order Prefetching Support

- **Order-0 prefetching (without context):** similar as sequential prefetching
- **Order-1 prefetching:** use DAH table directly, similar as distance or Markov prefetching
- **Order-2 prefetching:** use DAP table for prediction
  - DAP entry is located with the hashed form of order-2 context

PC-based order-1: “distance prefetching”

Addr-based order-1: “Markov prefetching”
Experimental Setup

- Simulation: Pin + CMP$im
  - Pin: dynamic instrumentation tool, collect program traces
  - CMP$im: trace driven, characterizes memory system performance
    - Closed source but with prefetching interface provided by DPC
  - Simulation configuration
    - Out-of-order processor with a 15-stage, 4-wide pipeline
    - L1 cache: 32KB and 8-way set associative
    - L2 cache: 16-way set associative, capacity varies from 512KB to 2MB
    - Access latency: 20 cycles for L2 cache and 200 cycles for memory

- Benchmarks
  - SPEC CPU2006 benchmark suite
  - Compiled using GCC 4.1.2 with –O2 optimization
  - Traces are collected for all benchmarks by fast forwarding 40 billion instructions then running 100 million instructions
Analysis of Impact of Orders

- **Prefetching accuracy** analysis

  SPEC CINT2006 Benchmarks

  SPEC CFP2006 Benchmarks

- Conclusion: the accuracy **increases** with the increase of the order
Analysis of Impact of Orders

- Prefetching coverage analysis

<table>
<thead>
<tr>
<th>SPEC CINT2006 Benchmarks</th>
<th>SPEC CFP2006 Benchmarks</th>
</tr>
</thead>
</table>

Conclusion: the coverage decreases with the increase of the order
Analysis of Impact of Orders

- Even though a high order model can achieve high accuracy, its coverage is low and may not reduce considerable amount of misses.
- The prefcher needs to support multiple orders in order to have the merits of both high accuracy and wide coverage.
Performance Analysis

- L2 cache miss rate reduction

On average, MOC prefetching reduces misses by over 65%, which is more than two times of what the distance prefetching achieves.
Performance Analysis

- Instructions Per Cycle (IPC) speedup

![Graph showing IPC speedup for different benchmarks and MOC prefetcher comparison with conventional prefetcher.]

- The average IPC improvement for MOC prefetcher is 45%, while the conventional prefetcher is 9.1%
Conclusion

- Rapid advance of high-end processor architectures has put more pressure than ever on improving data-access performance

- Data prefetching is an effective solution to hiding latency and to mitigating the fast growing processor-memory performance gap

- Contributions
  - Demonstrate existing context-based prefetcher has limited prefetching effectiveness due to limited coverage with simulations
  - Propose Multi-Order Context-based (MOC) prefetching to address the identified limitation and to strengthen context-based prefetching
  - Carry out simulation testing and the results confirm MOC achieves promising prefetching effectiveness and overall IPC speedup
Ongoing and Future Work

- Reducing the storage cost with a signature concept
- Same high-order bits observed in many miss streams
- These same portions can be concisely represented with a signature
- Future work: explores the order selection intelligently and dynamically
Any Questions?

Thank you.

Welcome to visit: http://ft.ornl.gov
http://www.cs.iit.edu/~scs

This research was sponsored in part by