Multi-Core Memory Hierarchies

Lecture 5: LLC
Prefetching and Eager Write

Tong Dong

http://mprc.pku.edu.cn/~tongdong/MMH
In last lecture

- Wire delay
- Shared (manycore) vs. Private (few cores) LLC
- Multithreading vs. Multiprogramming
  - Instruction
  - Private data
  - Shared data
- Shared distributed cache: long term impact.
  - OS-based policies
  - Reactive NUCA
  - Cache partition -> Private cache
- Cache Coherence Directory
Model of cache performance

- For N cores and T thread per core, the miss ratio of the $i$-level cache:
  $$m_{Li} = \left(\frac{C_{Li}}{I(NT)\beta_{Li}}\right)^{1-\alpha_{Li}}$$
  - $\alpha, \beta$: locality (Replacement/prefetching)
  - $I(NT)$: interference
  - $C$: cache size in level $i$

- Average Access Time in $i$-level cache:
  - $T$: Access Time
    - Hit Time (NUCA)
    - Miss Penalty (Prefetching)
  $$T_{Li} = (1 - m_{Li})T_{Li}^{hit} + m_{Li}T_{Li-1}$$
Memory Hierarchy in Server and Mobile Device

Long Memory Latencies

(a) Memory hierarchy for server

- CPU
- Registers
- L1 Cache
  - Register reference
  - Size: 1000 bytes
  - Speed: 300 ps
- Level 1 Cache reference
  - Size: 64 KB
  - Speed: 1 ns
- Level 2 Cache reference
  - Size: 256 KB
  - Speed: 3–10 ns
- Level 3 Cache reference
  - Size: 2–4 MB
  - Speed: 10–20 ns
- Memory reference
  - Size: 4–16 GB
  - Speed: 50–100 ns
- I/O bus
- Disk storage
  - Size: 4–16 TB
  - Speed: 5–10 ms

(b) Memory hierarchy for a personal mobile device

- CPU
- Registers
- L1 Cache
  - Register reference
  - Size: 500 bytes
  - Speed: 500 ps
- Level 1 Cache reference
  - Size: 64 KB
  - Speed: 2 ns
- Level 2 Cache reference
  - Size: 256 KB
  - Speed: 10–20 ns
- Memory reference
  - Size: 256–512 MB
  - Speed: 50–100 ns
- Memory bus
- Storage
  - Flash memory reference
  - Size: 4–8 GB
  - Speed: 25–50 us
Reducing Miss Penalty: Long Latencies

- Latency Tolerance
  - Out-of-order Pipeline
  - Speculative Execution
  - Multithreading: SMT, GPU, …
  - Non-blocking Cache: MSHR
  - Bank-level parallelism

- Latency Reduction
  - Prefetching
  - DRAM scheduling
    - Row-buffer Hit Ratio
    - DRAM Command Buffer Reordering
Prefetcher basics

- Ideal Cache: when a core need a data or instruction, it’s in the cache yet! How? Prefetch it
- Data accesses patterns in programs
  - Spatially predictable/Temporally predictable
- Software Prefetching vs **Hardware Prefetching**
- Regular vs Irregular access patterns
  - Conservative, confirmation-based prefetchers (stream prefetcher)
  - Aggressive, immediate prefetchers (next-line prefetcher)
- Prefetching Timeliness
- Recently, **instruction prefetcher** is more critical
  - More complex software stack and infrastructure
  - Poor instruction latency tolerance ability
Stride-based Prefetching

- **Next-line prefetcher**
  - For every cache line A that is fetched, prefetch A+1

- **Stream Prefetcher**
  - Prefetch multiple +1 lines ahead
    - Stream stared on access A
    - Stream direction determined on access A+1
    - Stream confirmed on access A+2
    - Begin prefetching A+3
  - Prefetch degree
    - How many cache lines are prefetched at a time

- **Stream Buffer, Jouppi, ISCA 1990**

## Stream Prefetcher

<table>
<thead>
<tr>
<th>First Access</th>
<th>Second Access</th>
<th>Stride</th>
<th>Next Expected</th>
<th>Next Prefetch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accesses: 0, 1, 2, 10, 11, 3, 12, 4, 5
Prefetched:
Stride Prefetcher

- Like a stream prefetcher, but with variable access stride (not always +1)
- More bookkeeping to determine stride, also requires confirmation before prefetching
  - Allocate stream on access A
  - Determine direction and stride on access A+X
  - Confirm stream on access A+2*X
  - Begin prefetching A+3*X

Feedback-Directed Prefetcher Throttling

- Idea: Release interference to other core
  - Dynamically monitor prefetcher
    - accuracy, timeliness, and cache pollution
  - Throttle the prefetcher **aggressiveness**: 5 levels
    - Prefetch distance and prefetch degree
  - Change the location prefetches are inserted

Srinath et al., Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers, HPCA 2007. (251)
Irregular Access Prefetching

- Irregular Access Pattern
  - Indirect array accesses
  - Linked data structures
  - Multiple regular strides (1,2,3,1,2,3,1,2,3,…)
  - Random patterns?
  - Generalized prefetcher for all patterns?

- Prefetchers for irregular access patterns
  - Pointer based prefetchers
  - Content-directed prefetchers
  - Correlation based prefetchers
  - Precomputation or execution-based prefetchers
  - Prefetch filters
Prefetching Based on Temporal Locality

Markov Prefetcher, ISCA’97
- Address correlation
- Large amount of storage are needed

General structure for prefetching streams of temporally correlated memory request.
- GHB: Global history buffer (FIFO with pointer)
- Index table
  - Address correlation
  - Delta correlation
  - PC correlation
Prefetchers before GHB

Figure 2: Arbitrary Stride Prefetching Table

Figure 3: Markov Prefetching

Figure 4: Distance Prefetching.

Figure 1a: Basic Prefetch Table
Global History Buffer, Nesbit and Smith, HPCA 2004

- Instead of just one history table, GHB uses an index table and a global history buffer
  - Index table is accessed by directly indexing into it
  - GHB is a FIFO with pointers between entries

- Advantages
  - FIFO history buffer can improve the accuracy of correlation prefetching
  - GHB contains more complete (and contact) history

- Disadvantages:
  - storage overhead/useless prefetch


GHB Prefetcher

GHB Global / Address correlation

GHB Global / Delta correlation
Temporal Streaming, Wenisch, ISCA 2005

- Temporal address correlation
  - Shared data typically were accessed in the same sequence of addresses (stream) by different cores.

- Temporal stream locality
  - When a node in CMP has a miss, it checks to see if another node recently had the same miss.
  - If then prefetches the same miss stream.

- Implementation
  - Large circular buffer that is stored in memory
  - Directory record the stream
  - Good for commercial applications
Temporal Streaming

FIGURE 1: Temporal streaming.
Prefetching Based on Spatial Locality

Spatial Memory Streaming
- In commercial applications, memory accesses within a region can be sparse, un-strided.
- But, they are repeatable and then predictable.

FIGURE 3. Pattern History Table and prediction process. Upon a trigger access that matches in the PHT, the region base address and spatial pattern are transferred to a prediction register, beginning the streaming process.
Spatio-Temporal Prefetching, Somogyi, ISCA’09

- Spatio-Temporal Memory Streaming (STeMs)
- Hybrid Spatio-Temporal Prediction

FIGURE 2. Motivating example for spatio-temporal streaming: a database index scan. The order of page accesses is arbitrary but repetitive (temporal behavior). Accesses within pages repeat (spatial behavior). Overall, the scan consists of a global access sequence that is predictable using temporal and spatial correlation together.
Execution-based prefetching

Run-ahead, HPCA 2003

- When the oldest instruction is a long-latency cache miss: Checkpoint architectural state and enter run-ahead mode
- In run-ahead mode: Speculatively pre-execute instructions
- Checkpoint is restored and normal execution resumes
- Continuous Run-Ahead, MICRO’16

Helper-threading

- Pre-execute a piece of the (pruned) program that lead to cache misses solely for prefetching data/instruction
- On Multi-core and Multi-threading processor

Cold Data Prefetching

- Virtualized desktop infrastructure (VDI)
  - 6-8 VMs per core
  - Context switch performance

- Cache Restoration, HPCA 2012
Instruction Prefetcher

- Frontend stalls due to large instruction working sets of server workloads account for up to 40% of execution time in server processors.
- Temporal streaming instruction prefetcher
  - Temporal instruction fetch streaming, MICRO 2008
  - Proactive instruction fetch, MICRO 2011
- Storage and bandwidth overhead
  - Shared history. SHIFT, MICRO 2013
- Return-address correlation
  - RDIP, MICRO 2013
- OS-event: pTask, MICRO 2016
Advanced Write-Back Policies

- Cooperative Main Memory and LLC
- Virtual write queue, ISCA 2010
- Last-write prediction, ISCA 2012

SBP: Sandbox Prefetching, HPCA’14

- SBP: global confirmation with immediate action
  - Access Map Pattern Matching Prefetcher, JILP’11
  - Bloom Filter

- Best-Offset Prefetcher, HPCA’16
  - Offset prefetcher, prefetch timeliness.
TEMPO: translation-enabled memory prefetching optimizations

- A substantial fraction (20-40%) of DRAM references are devoted to accessing page tables
- The vast majority of them (98%+) also look up DRAM for the subsequent data access.

Figure 5. Timeline of events for a memory reference that misses in the TLB. Page table walks are shown in blue the memory replay is shown in green.

Figure 6. Timeline of events when TEMPO prefetches the data that the replayed instruction will use into the DRAM row buffer and LLC. Subsequent LLC and row buffer hits improve performance.
Holistic cache management

PC is not valuable to prefetcher

KPC-P generates a confidence value which cache level to insert the prefetched blocks

KPC-R quickly adapts to the dynamic program phase using two small global counters

KPC outperforms a prior unified memory architecture by 8.1%

Figure 6: Design overview of the KPC system.
Summary

- Prefetching: reducing miss penalty (latency)
- For Regular Access Patterns
  - Stream and Stride prefetchers
- For Irregular Access Patterns
  - Global history Buffer (GHB)
  - Pointer based prefetchers
  - Content-directed prefetchers
  - Precomputation or execution-based prefetchers
- Cold data prefetchers
- Instruction and TLB prefetchers
- Eager Write: coordination with DRAM