Efficient Synonym Filtering and Scalable Delayed Translation for Hybrid Virtual Caching

Chang Hyun Park, Taekyung Heo, and Jaehyuk Huh

KAIST School of Computing
Physical Caching

- Latency constraint limits TLB scalability
  - TLB size restricted
  - Limited coverage of TLB entry
- Missed Opportunities\(^\text{[1]}\)
  - Memory access misses TLB, hits in cache
  - TLB miss delays cache hit opportunity

[1] Zhang et al. ICS 2010
Virtual Caching

- Delay translation: Virtual Caching
  - Access cache, then translate on miss
  - Cache hits do not need translation

- Problem: Synonyms
  - Synonyms are rare\cite{2}
  - Optimize for the common case

- TLB accesses reduced significantly
  - Loosen TLB access latency restriction
  - Possibility of sophisticated translation
  - Reduces power consumption

\cite{2} Basu et al. ISCA 2012
Hybrid Virtual Caching

Physical Caching

Virtual Address

Core

Virtual Address

Core

Virtual Address

Core

Virtual Address

Core

Physical Address

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Scalable Delayed Translation

Physical Address

Virtual Caching

EXPECT DELAYS

L1 $\rightarrow$ Core

L1 $\rightarrow$ Core

L1 $\rightarrow$ Core

Physical Address

Virtual Address

Core

Virtual Address

Core

Virtual Address

Core

Physical Address

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Scalable Delayed Translation

Physical Address

Virtual Caching

Virtual Address

Core

Virtual Address

Core

Virtual Address

Core

Physical Address

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Scalable Delayed Translation

Physical Address

Virtual Caching

Virtual Address

Core

Virtual Address

Core

Virtual Address

Core

Physical Address

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Last-Level $\rightarrow$ L1 $\rightarrow$ Core

Scalable Delayed Translation

Physical Address

Virtual Caching
Contributions

• Propose hybrid virtual physical caching
  • Cache populated by both virtual and physical blocks
  • Virtual cache for common case, physical for synonyms
  • Synonyms not confined to fixed address range, use entire cache

• Propose scalable yet flexible delayed translation
  • Improve TLB entry scalability by employing segments [2][3]
  • Provide many segments for flexibility of memory management
  • Propose efficient search mechanism to lookup segment

Hybrid Virtual Caching

• Virtual **and** physical cache
  • Each page **consistently** determined as physical or virtual
  • Cache tags hold either tags
  • **Challenge**: Choose address **before** cache access

• **Synonym Filter**: Bloom Filter that detects synonyms
  • HW managed by OS
  • Synonyms **always** detected, translated to physical address
Hybrid Virtual Caching Efficiency

- Pin-based simulation
- Baseline TLB
  - L1 TLB: 64 entries
  - L2 TLB: 1024 entries
- Hybrid Virtual Caching
  - 2x1Kb Synonym filters
  - Synonym TLB: 64 entries
  - Delayed TLB: 1024 entries
- Workloads
  - Apache, Ferret, Firefox, Postgres, SpecJBB
Hybrid Virtual Caching Efficiency

- **Synonym Filter**
  - 83.7~99.9% TLB accesses bypassed

- **Delayed Translation**
  - Up to 99.9% TLB access reduction
  - Up to 69.7% TLB miss reduction
Hybrid Virtual Caching Efficiency

**Hybrid Virtual Caching**

- **Synonym Filter**
  - Majority of accesses to virtual cache

- **Delayed Translation**
  - Cache hits remove TLB accesses and reduce TLB misses

- **Physical Address**
- **Virtual Address**
- **Core**
- **L1**
- **Last-Level**
- **Delayed TLB**
- **9.83.7~99.9% TLB accesses bypassed**
- **Up to 99.9% TLB access reduction**
- **Up to 69.7% TLB miss reduction**
Limitation of Delayed TLB

- TLB entries limited in scalability
  - Each entry maps fixed granularity
  - Increasing TLB size does not reduce miss as expected
Limitation of Delayed TLB

- TLB entries limited in scalability
  - Each entry maps fixed granularity
  - Increasing TLB size does not reduce miss as expected

TLB size is restricted, Improve coverage of TLB entry
Segments: Scalable Translation

- Direct Segment\cite{2} improves TLB entry coverage
  - Represented by three values (base, limit, offset)
  - Translates contiguous memory of any size

\cite{2} Basu et al. ISCA 2013
\cite{3} Karakostas, Gandhi et al. ISCA 2015
Segments: Scalable Translation

• Direct Segment\(^2\) improves TLB entry coverage
  • Represented by three values (base, limit, offset)
  • Translates contiguous memory of any size

• OS benefits from more available segments
  • Memory sharing among processes fragment memory
  • OS can offer multiple smaller segments

• Number of segments\(^3\) limited by latency
  • Segment lookup between Core and L1 cache
  • Fully-associative lookup of all segments required

\(^2\) Basu et al. ISCA 2013
\(^3\) Karakostas, Gandhi et al. ISCA 2015
Scalable Delayed Translation

• Exploit reduced frequency of delayed translation
  • Prior work limited to 10s of segments
  • Provide 1000s of segments for OS Flexibility

• Efficient searching of owner segment required
  • OS managed tree that locates segment in a HW table
  • HW walker that traverses tree to acquire location
  • Use location (index) to access segment in HW table
Scalable Delayed Translation

*Segment Table*: register values for many segments

Infeasible to search all Segment Table entries
Scalable Delayed Translation

**Index Tree**: B-tree that holds following mapping
- **key**: virtual address
- **value**: index to Segment Table

LLC Miss (Non-synonym) → Index Tree → Segment index → Segment Table → Memory Access

<table>
<thead>
<tr>
<th>Index</th>
<th>Base</th>
<th>Limit</th>
<th>Offset</th>
<th>etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scalable Delayed Translation

*Index Cache*: caches index tree nodes on-chip

*Hardware Walker*: searches through the index tree to produce a segment table index
Address Translation Procedure

*Segment Cache*: caches many segment translation

- LLC Miss (Non-synonym)
- Segment Cache: caches many segment translation
- Index Cache
- Segment Table
- Index Tree
- Traverse tree
- HW Walker
Address Translation Procedure

*Segment Cache*: caches many segment translation

**Reduces latency and power consumption**
Evaluation

- Full system OoO simulation on Marssx86 + DRAMSim2
  - Hosts Linux with 4GB RAM (DDR3)
- Three level cache hierarchy (based on Intel CPUs)
- Baseline TLB configurations (based on Intel Haswell)
  - L1 TLB: 1 cycle, 64 entry, 4-way
  - L2 TLB: 7 cycle, 1024 entry, 8-way
  - Delayed TLB configurations range 1K - 16K entry
- Many segment translation configurations
  - Segment Table: 2K entries
  - Index Cache: 32KB
  - Segment Cache: 128 entry
- Benchmarks: SPECCPU, NPB, biobench, gups
Results

Normalized IPC to Baseline TLB (%)

- Delayed TLB 1K entries
- 4K
- 16K
- Many Segment Translation

21
Cache hits reduce TLB accesses & misses
Improving Performance
Results

Normalized IPC to Baseline TLB (%)

- Delayed TLB 1K entries
- 4K
- 16K
- Many Segment Translation

Delayed TLB offers some scalability
Results

Scalable Delayed Translation improves performance by 10.7% on average.

Power consumption is reduced by 60% on average.

Increased translation scalability significantly reduces TLB misses.
Conclusion

• **Hybrid Virtual Cache** allows **delaying** address translation
  • Majority of memory accesses use virtual caching, synonyms use physical caching
  • Synonym Filter consistently and quickly identifies access to synonym pages
  • Reduces up to 99.9% of TLB accesses, 69.7% of TLB misses

• **Scalable** delayed translation
  • Exploits reduced translations
  • Provides many segments and efficient segment searching
  • Average 10.7% performance improvement, 60% power saving