Hybrid TLB Coalescing: Improving TLB Translation Coverage under Diverse Fragmented Memory Allocations

Chang Hyun Park, Taekyung Heo, Jungi Jeong, and Jaehyuk Huh
Introduction

• Virtual memory provides rich features
  • Requires an address translation

• Workloads have grown in size pressuring TLB

• Contiguous memory allocations to the rescue!
Past Proposals: Large pages

- Large pages represent larger mappings (2MB)
  - Strict alignment required
  - Exact size match required
Past Proposals: Large pages

- Large pages represent larger mappings (2MB)
  - Strict alignment required
  - Exact size match required

Efficient when large pages provided by OS
Past Proposals: Cluster TLB

• HW oriented clustering\[5\]

• Cluster TLB represents flexible mapping within cluster
  • Provides flexible mapping within cluster block
  • However cluster size is fixed at design time

[5] Pham et al. HPCA ’14
Past Proposals: Cluster TLB

- HW oriented clustering\(^5\)
- Cluster TLB represents flexible mapping within cluster
  - Provides flexible mapping within cluster block
  - However cluster size is fixed at design time

Efficient with small clustering opportunities

\(^5\) Pham et al. HPCA ’14
Past Proposals: Direct Segments

- Segment based translation\textsuperscript{[1]}
  - Three values represent \textit{contiguous} translation of any size
  - Fully assoc. lookup for multiple segments (limits size of TLB)
    - Redundant Memory Mappings (RMM)\textsuperscript{[6]} \to 32 Fully-associative TLB

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
Virtual Pages & Direct Segment & Physical Pages \\
\hline
Base & Offset & Direct Segment \\
\hline
Limit & Base & Limit & Offset \\
\hline
\end{tabular}
\end{center}

\textsuperscript{[1]} Basu et al. ISCA ’13
\textsuperscript{[6]} Karakostas et al. ISCA ’15
Past Proposals: Direct Segments

• Segment based translation\[^1\]
  • Three values represent *contiguous* translation of any size
  • Fully assoc. lookup for multiple segments (limits size of TLB)
    • Redundant Memory Mappings (RMM)\[^6\] -> 32 Fully-associative TLB

Efficient with small number of big memory chunks

---

\[^1\] Basu et al. ISCA ’13
\[^6\] Karakostas et al. ISCA ‘15
Past Proposals: Summary

• Large pages
  • Affinity for large pages (2MB)

• Cluster TLB
  • Affinity for clustering of mapping of up to 8 pages

• Segment translations
  • Affinity for small number of large chunks (32 entry TLB)

Prior proposals efficiently support specific memory mapping scenarios
Large Contiguity vs. Memory Non-Uniformity

- Conflicting goals of NUMA systems and large pages\cite{2}
  - Memory traffic balance vs. efficient address translation

- Heterogeneous memory worsens non-uniformity \cite{3}\cite{4}

\cite{2} Baptiste et al. ATC ’14
\cite{3} Lee et al. ISCA ‘15
\cite{4} Agarwal et al. ASPLOS ‘17
Large Contiguity vs. Memory Non-Uniformity

• Conflicting goals of NUMA systems and large pages\textsuperscript{[2]}
  • Memory traffic balance vs. efficient address translation

Different systems have different memory mapping needs

• Heterogeneous memory worsens non-uniformity \textsuperscript{[3][4]}

[3] Lee et al. ISCA ‘15
[4] Agarwal et al. ASPLOS ‘17
Need for an All-Rounder Solution

- Contiguity distribution varies among workloads
- Also varies within the same workload\textsuperscript{[7]}

\textsuperscript{[7]} Kwon et al. OSDI ’16
Need for an All-Rounder Solution

- Contiguity distribution varies among workloads
- Also varies within the same workload\textsuperscript{[7]}

Can we make a TLB scheme that works well for diverse scenarios?

[7] Kwon et al. OSDI ’16
Hybrid TLB Coalescing

We propose a TLB with adjustable coverage

- **HW-SW** Joint Effort
- **HW** offers adjustable TLB coverage
  - Number of TLB entries fixed
  - Coverage of entry adjustable

- **OS** decides best TLB coverage
  - Adjusts TLB coverage per process
- **OS** identifies contiguous chunks
  - Marks onto process page table
Anchor

- Anchors are special entries in the page table
  - Placed at every alignments of anchor distance
  - Anchor distance is a power of 2 (for encoding efficiency)
  - Anchor distance configurable by OS

Anchor Distance = 8

Page Table

0x00 0x04 0x08 0x0C 0x10
Anchor Page Table

• Uses the Page Table

• Anchor covers up to distance(4) contiguous pages
  • Each anchor represents contiguity that begins at anchor

• OS marks contiguity onto the anchor page table
Anchor TLB

- Integrated into the L2 TLB
  - L1 keeps regular entries
- Caches both regular and anchor page table entries
  - Regular and anchor indexed differently

![Diagram of Anchor TLB]

- Virtual Pages: 2, 3, 4, 0, 4
- TLB Entries: 0 | 2, 0 | 3, 0 | 4, 1 | 4
- Anchor Entry: 3 | X, 3 | X, 3 | X
- Regular Entry: 3 | X
Anchor TLB Lookup

• On L1 TLB Miss Anchor TLB looks up
  • Regular TLB first
  • Anchor TLB next

Offset (2) < Contiguity (3)

MISS
Start Page Walk

HIT
return Anchor PFN + offset

Virtual Pages
0 | 3

Anchor Entry
0 | 2
0 | 3
0 | 4
3 | X

Regular Entry
1 | 4
3 | X
3 | X

Anchor TLB
(4 sets)
Operating System Responsibilities

• OS periodically selects process anchor distance
  • Heuristic algorithm to minimize TLB entry count

• OS adjusts anchor distance
  • Anchor distance based on selection algorithm

• OS marks mapping contiguity
  • Memory mapping contiguity in anchor page table entry
## Simulation Methodology

- **Trace based TLB simulator (Based on Intel Haswell)**

<table>
<thead>
<tr>
<th></th>
<th>TLB Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common L1</strong></td>
<td>4KB: 64 entry, 4 way</td>
</tr>
<tr>
<td></td>
<td>2MB: 32 entry, 4 way</td>
</tr>
<tr>
<td><strong>Baseline L2 / THP</strong></td>
<td>4KB/2MB: 1024 entry, 8 way</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>Regular (4KB/2MB): 768 entry, 6 way</td>
</tr>
<tr>
<td></td>
<td>Cluster-8: 320 entry, 5 way</td>
</tr>
<tr>
<td><strong>RMM (Multiple segments)</strong></td>
<td>Baseline L2 TLB + RMM: 32 entry, fully-assoc.</td>
</tr>
<tr>
<td><strong>Anchor (Selected/Static Ideal)</strong></td>
<td>4KB/2MB/anchor: 1024 entry, 8 way</td>
</tr>
</tbody>
</table>
Memory Mapping Scenarios

- Two class of memory mapping scenarios
  - Two real system memory mappings
  - Four synthetic memory mappings

<table>
<thead>
<tr>
<th>Name</th>
<th>Trace information</th>
</tr>
</thead>
<tbody>
<tr>
<td>demand</td>
<td>Default Linux memory mapping</td>
</tr>
<tr>
<td>eager</td>
<td>‘Eager’ allocation</td>
</tr>
<tr>
<td>low</td>
<td>1– 16 pages (4KB – 64KB)</td>
</tr>
<tr>
<td>medium</td>
<td>1 – 512 pages (4KB – 2MB)</td>
</tr>
<tr>
<td>high</td>
<td>512 – 64K pages (2MB – 256MB)</td>
</tr>
<tr>
<td>max</td>
<td>Maximum contiguity</td>
</tr>
</tbody>
</table>
Evaluation – TLB Misses of demand mapping
Evaluation – TLB Misses of demand mapping

Relative TLB Misses (%)

- THP
- Cluster
- RMM
- Anchor Selected
- Anchor Ideal

Anchor TLB adjusted to satisfy small contiguities
Evaluation – TLB Misses of medium mapping
Evaluation –
TLB Misses of medium mapping

Anchor adjusted coverage to provide best TLB reduction
Evaluation – TLB Misses of all mapping

Relative TLB Misses (%)

- Baseline
- THP
- Cluster
- RMM
- Anchor Selected
- Anchor Ideal

Demand, eager, low cont., med cont., high cont., max cont.
Evaluation –
TLB Misses of all mapping

Anchor TLB performs well for diverse mapping scenarios
Conclusion

• Hybrid TLB Coalescing is a HW-SW joint effort
• Anchor TLB provides adjustable coverage
  • TLB entry coverage grows and shrinks dynamically
• OS provides contiguity hint using the page table
• OS picks adequate contiguity per-process

• Hybrid TLB Coalesce performs:
  • Best for Small-Intermediate contiguities
  • Similar to best prior scheme for Large contiguities