Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism

Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu, Josep Torrellas
University of Illinois at Urbana-Champaign
skarlat2.web.engr.illinois.edu

ASPLOS 2020
Virtual Memory Translation is Expensive

Application

Main Memory

Page Tables

PA 1

PA 4

VA 1

PA 4

VA 8

PA 1

Core

L1 Cache

L2 Cache

L3 Cache

Issue LD VA 1

TLB

TLB Miss!
Virtual Memory Translation is Expensive

Application → Core

Issue LD VA 1

Core → L1 Cache → L2 Cache → L3 Cache → Main Memory

Main Memory: PA 1, PA 4, Page Tables: VA 8, PA 1, VA 1, PA 4

TLB: VA 1, PA 4

TLB Miss → “Page Walk” = Fetch entry from page table
x86-64 Radix Page Tables
x86-64 Radix Page Tables

Virtual Address

<table>
<thead>
<tr>
<th>47 … 39</th>
<th>38 … 30</th>
<th>29 … 21</th>
<th>20 … 12</th>
<th>11 … 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Address A

CR3

pgd

PUD

pte

PTE

TLB Entry

VA1  PA4

Page Offset

9-bits  9-bits  9-bits  9-bits
Virtual Memory Translation is Expensive

Application" Page Walk" = Fetch entry from radix page table

Main Memory

Radix Page Tables

pgd pud pmd pte

L3 Cache

Core

Issue LD VA 1

L2 Cache

L1 Cache

Application

Core

TLB
Virtual Memory Translation is Expensive

Application ➔ Core ➔ L1 Cache ➔ L2 Cache ➔ L3 Cache ➔ TLB ➔ Main Memory

TLB Miss ➔ “Page Walk” = Fetch entry from radix page table
Virtual Memory Translation is Expensive

TLB Miss \( \rightarrow \) “Page Walk” = Fetch entry from radix page table
Virtual Memory Translation is Expensive

TLB Miss → “Page Walk” = Fetch entry from radix page table
Virtual Memory Translation is Expensive

Application

Core

L1 Cache

L2 Cache

L3 Cache

TLB

Issue LD VA 1

Main Memory

Page Tables

PA 1

PA 4

VA 1

PA 4

VA 8

PA 1
Multilevel TLBs
Memory Management Unit (MMU) Cache

- Application
- Core
- L1 Cache
- L2 Cache
- L3 Cache
- L1 TLB
- L2 TLB
- MMU Cache

Main Memory
- PA 1
- PA 4

Radix Page Tables
- pgd
- pud
- pmd
- pte
Translations in Data Caches

Application

Core

L1 Cache

L2 Cache

pgd

pud

L3 Cache

pgd

pud

pmd

pte

L1 TLB

MMU Cache

L2 TLB

Radix Page Tables

Main Memory

PA 4

PA 1

pgd

PUD

PMD

pte
Sunny Cove introduces 5-Level Radix Page Tables!!

NVM will Make the Problem Worse
Contribution: **Elastic Cuckoo Page Tables**

- Rethinking virtual memory translation for parallelism

- Idea: Dynamically resizable page tables based on cuckoo hashing
  - No sequential page table lookups → parallel single-step lookups

- Application speedup over state-of-the-art:
  - 3-28% with 4KB pages
  - 3-18% with Huge pages
Alternative: A Global Hashed Page Table
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

Global Hash Table

Application

VA_1
VA_9

H
H

Tag
Tag

Collisions

OS is invoked to resolve them!
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

- How to share pages?
- Multiple page sizes?

Global Hash Table

Application A

Application B
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

How to share pages?  New level of indirection!!
Multiple page sizes?

Global Hash Table
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

How to share pages?
Multiple page sizes?

New level of indirection!!

Global Hash Table

Application A

VA 1

H

VA 9

H

Application B

VA 6

H

Tag

Tag

Tag
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

How to share pages? Multiple page sizes?

New level of indirection!!

Global Hash Table

Application A

VA 1
VA 9

Application B

VA 6

COLLISIONS
PAGE SHARING
PAGE SIZES
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM Switched to radix page tables!
Elastic Cuckoo Page Tables

Rethinking virtual memory translation for parallelism
Cuckoo Hashing [Pagh 2001, Fotakis 2005]

$d$-ary Cuckoo Hash Table

T1: d, f

T2: c, a

T3: b, g
Insertions with Cuckoo Hashing

$d$-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

d-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

d-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing
Insertions with Cuckoo Hashing

\[ d \quad e \quad c \quad a \]

\[ b \]  \[ H_2 \]

\[ f \quad g \]

\[ T1 \quad T2 \quad T3 \]

d-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

\[ T_1 \]
\[ d \] \\
\[ e \] \\
\[ \] \\

\[ T_2 \]
\[ b \] \\
\[ c \] \\
\[ a \] \\

\[ T_3 \]
\[ \] \\
\[ f \] \\
\[ g \] \\

\( d \)-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

d-ary Cuckoo Hash Table

T1

T2

T3

d e
b c
f g
Private Hashed Page Tables

PRIVATE PAGE TABLES

$\begin{array}{ccc}
\text{T1} & \text{T2} & \text{T3} \\
d & b & \text{f} \\
e & c & \\
a & g & \\
\end{array}$

$d$-ary Cuckoo Hash Table
Private page tables cannot be too big
Need to Dynamically Resize

PRIVATE PAGE TABLES

App A

Main Memory

Page Tables A

VA 1  PA 4
VA 8  PA 1

Private page tables cannot be too big

Need to dynamically resize
Need to Dynamically Resize

PRIVATE PAGE TABLES

App A

Main Memory

Page Tables A
- VA 1: PA 4
- VA 8: PA 1

New Page Tables A

Private page tables cannot be too big

Need to dynamically resize
Cannot Rehash All Entries at Once

Private page tables cannot be too big
Need to dynamically resize

Main Memory
Page Tables A
- VA 1 → PA 4
- VA 8 → PA 1

New Page Tables A

PRIVATE PAGE TABLES

COLLISIONS
PAGE SHARING
PAGE SIZES
Private page tables cannot be too big
Need to dynamically resize
Cannot Rehash All Entries at Once

Private page tables cannot be too big

Need to dynamically resize

While the program is running

Gradual Resizing!
Gradual Resizing Cuckoo Hash Tables

At every insert → Rehash one element

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table
Gradual Resizing Cuckoo Hash Tables

At every insert → Rehash one element

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table
Lookup During Gradual Resizing

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table
Problem of Resizing: Double #Lookups

2 x d Lookups!
Contribution: Elastic Cuckoo Hashing

Rehashing Pointers

Old \(d\)-ary Cuckoo Hash Table

\[ T1 \]
- m
- k
- e

\[ T2 \]
- b
- c
- a

\[ T3 \]
- f
- l
- g

New \(d\)-ary Cuckoo Hash Table

\[ T1' \]

\[ T2' \]

\[ T3' \]
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table

Migrated Region
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table

Migrated Region
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

- T1
  - e
  - c
  - a

- T2
  - f
  - l
  - g

- T3

Migrated Region

New $d$-ary Cuckoo Hash Table

- T1'
  - m
  - k
  - b

- T2'

- T3'
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

Old迁移到新

New $d$-ary Cuckoo Hash Table

Migrated Region
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table

Only need $d$ lookups during resizing
Exploiting Parallelism in Virtual Translation

d-ary Elastic Cuckoo Hash Table
Exploiting Parallelism in Virtual Translation

d-ary Elastic Cuckoo Page Table
Exploiting Parallelism in Virtual Translation

- No sequential page walk (unlike radix)
- At most $d$ accesses always
- Leverages multiple issue out-of-order processors

$d$-ary Elastic Cuckoo Page Table

VA$_K$

H$_1$  H$_2$  H$_3$

pte d  pte c  pte a

pte k

T1  T2  T3

Leverages multiple issue out-of-order processors
Exploiting Parallelism in Virtual Translation

- No sequential page walk (unlike radix)
- At most $d$ accesses always
- Leverages multiple issue out-of-order processors

$d$-ary Elastic Cuckoo Page Table
Lookup Multiple Page Sizes in Parallel

- $d$-ary Elastic Cuckoo Page Table
  - 4KB PTE Entries
  - 2MB PMD Entries
  - 1GB PUD Entries

Elastic Cuckoo Page Table
- 1GB PUD Entries

T1, T2, T3
New MMU Cache to Prune Parallelism

- MMU Cache
- VA
- MMU Cache
- VA
- MMU Cache
- VA

T1 T2 T3

pte d pte c pte a
pte k

pte f

T1 T2 T3

pmd d pmd a

Elastic Cuckoo Page Table
2MB PMD Entries

Elastic Cuckoo Page Table
1GB PUD Entries

Elastic Cuckoo Page Table
4KB PTE Entries

d-ary Elastic Cuckoo Page Table
New MMU Cache to Prune Parallelism

Elastic Cuckoo Page Table
2MB PMD Entries
New MMU Cache to Prune Parallelism

Elastic Cuckoo Page Table
2MB PMD Entries
New MMU Cache to Prune Parallelism

Elastic Cuckoo Page Table
2MB PMD Entries
Evaluation
Application Speedup

- BC
- BFS
- CC
- DC
- DFS
- GUPS
- MUMmer
- PR
- SSSP
- SysBench
- TC
- Mean
Application Speedup

![Bar chart showing application speedup for various benchmarks. The chart includes applications such as BC, BFS, CC, DC, DFS, GUPS, MUMmer, PR, SSSP, SysBench, TC, and Mean. The y-axis represents speedup, and the x-axis represents different benchmarks.]
Application Speedup

![Bar chart showing speedup for various applications using Radix 4KB and Radix THP.]
Application Speedup

- Radix 4KB
- Radix THP
- Cuckoo 4KB

Speedup

- BC
- BFS
- CC
- DC
- DFS
- GUPS
- MUMmer
- PR
- SSSP
- SysBench
- TC
- Mean

Values:
- Radix 4KB: 3.31
- Radix THP: 1.64
- Cuckoo 4KB: 64
Application Speedup

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Radix 4KB</th>
<th>Radix THP</th>
<th>Cuckoo 4KB</th>
<th>Cuckoo THP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>3.31</td>
<td>3.45</td>
<td>1.64</td>
<td>1.82</td>
</tr>
<tr>
<td>BFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUMmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysBench</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Application Speedup

Elastic Cuckoo Page Tables over Radix
3-28% (only 4KB pages)
3-18% (+Huge pages)
Time Spent in Translations

- Radix 4KB
- Radix 2MB
- Cuckoo 4KB
- Cuckoo THP

Bar chart showing the time spent in translations for different benchmarks and configurations.
Time Spent in Translations

Elastic Cuckoo Page Tables
Reduce Time Spent in Translation by 41% on Average
More in the Paper

- Elastic cuckoo hashing operation
- Design of MMU cache and other structures

- More evaluation:
  - MMU and cache-subsystem characterization
  - Cuckoo walks characterization
  - Memory consumption of page tables
Takeaway: Elastic Cuckoo Page Tables

Better alternative to existing radix page tables

• Exploits parallelism in virtual translation for the first time
• Reduces the cost of dynamic resizing of hash tables
• Application speedup over state-of-the-art:
  • 3-28% with 4KB pages
  • 3-18% with Huge pages

• Expected high performance impact on:
  • Virtualized environments and nested page tables (ongoing work)
Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism

Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu, Josep Torrellas
University of Illinois at Urbana-Champaign
skarlat2.web.engr.illinois.edu

ASPLOS 2020