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

Core

Issue LD VA 1

TLB

Main Memory

Page Tables

PA 1

PA 4

PA 1

PA 4

L1 Cache

L2 Cache

L3 Cache

TLB Miss → “Page Walk” = Fetch entry from page table
x86-64 Radix Page Tables
x86-64 Radix Page Tables
Virtual Memory Translation is Expensive

Application → Core → L1 Cache → L2 Cache → L3 Cache → TLB → "Page Walk" = Fetch entry from radix page table

Main Memory

Radix Page Tables

Application

Core

L1 Cache

L2 Cache

L3 Cache

TLB

Issue LD VA 1

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

Application

Core

Issue LD VA 1

L1 Cache

L2 Cache

L3 Cache

TLB

pgd

TLB Miss → “Page Walk” = Fetch entry from radix page table

Main Memory

Radix Page Tables

PUD

PMD

pte

pgd

pud

pmd

pte
Virtual Memory Translation is Expensive

Application

Core

L1 Cache

L2 Cache

L3 Cache

Main Memory

Radix Page Tables

TLB Miss → “Page Walk” = Fetch entry from radix page table

TLB
Virtual Memory Translation is Expensive

Application

Main Memory

PA 1

PA 4

Radix Page Tables

pmd

pgd

pud

pmd

pte

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

Application

Core

L1 Cache

L2 TLB

L1 TLB

L2 Cache

L2 TLB

L3 Cache

Main Memory

PA 1

PA 4

Radix Page Tables

pgd
PGD

pud
PUD

pmd
PMD

pte
PTE
Memory Management Unit (MMU) Cache
Translations in Data Caches

Application

Core

L1 Cache

L2 Cache

L3 Cache

L1 TLB

L2 TLB

MMU Cache

Radix Page Tables

Main Memory

PA 1

PA 4

pgd

pud

pmd

pte

PGD

PUD

PMD

PTE
NVM will Make the Problem Worse

Sunny Cove introduces 5-Level Radix Page Tables!!
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 ➔ VA₁ ➔ H ➔ Global Hash Table ➔ Tag ➔ Tag

VA₉ ➔ H

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

VA 1

VA 9

VA 6
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
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

Diagram:

Application A
- VA 1
- H
- VA 9

Application B
- VA 6
- H

Tags connected to pages.
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

Application B
Alternative: A Global Hashed Page Table

The old approach from Intel and IBM Switched to radix page tables!

Application A
VA 1
VA 9

Application B
VA 6

Global Hash Table
Tag H

DEAD END
Elastic Cuckoo Page Tables

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

d-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

d-ary Cuckoo Hash Table

T1

T2

T3

H_1
Insertions with Cuckoo Hashing

\[ H_1 \]

\[ \begin{array}{c}
\text{T1} \\
\text{d} \\
\text{e} \\
\text{T2} \\
\text{c} \\
\text{a} \\
\text{T3} \\
\text{b} \\
\text{g}
\end{array} \]

d-ary Cuckoo Hash Table
Insertions with Cuckoo Hashing

$d$-ary Cuckoo Hash Table

- T1: d
  - e
- T2: c
  - a
- T3: b
  - g
Insertions with Cuckoo Hashing
Insertions with Cuckoo Hashing

\[ d \rightarrow e \rightarrow a \rightarrow c \rightarrow f \rightarrow g \]

\[ H_2 \]

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

\[ \begin{array}{c}
    \begin{array}{c}
        d \\
        \hline \\
        e \\
        \hline \\
        \text{T1}
    \end{array} & \quad & \begin{array}{c}
        b \\
        \hline \\
        c \\
        \hline \\
        \text{T2}
    \end{array} & \quad & \begin{array}{c}
        f \\
        \hline \\
        g \\
        \hline \\
        \text{T3}
    \end{array}
\end{array} \]
Insertions with Cuckoo Hashing

d-ary Cuckoo Hash Table

- T1: d, e
- T2: b, c, a
- T3: f, g

COLLISIONS
PAGE SHARING
PAGE SIZES
Private Hashed Page Tables

PRIVATE PAGE TABLES

COLLISIONS
PAGE SHARING
PAGE SIZES

d-ary Cuckoo Hash Table

T1

T2

T3

d

e

b

c

f

g

a
Private page tables cannot be too big
Need to Dynamically Resize

Private page tables cannot be too big

Need to dynamically resize
Need to Dynamically Resize

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
Cannot Rehash All Entries at Once

PRIVATE PAGE TABLES

Main Memory

App A

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

COLLISIONS
PAGE SHARING
PAGE SIZES
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 $\rightarrow$ Rehash one element

Old $d$-ary Cuckoo Hash Table

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

At every insert $\rightarrow$ 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

Old $d$-ary Cuckoo Hash Table

New $d$-ary Cuckoo Hash Table

2 x $d$ Lookups!
Contribution: Elastic Cuckoo Hashing

Old \( d \)-ary Cuckoo Hash Table

New \( d \)-ary Cuckoo Hash Table
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

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

Old $d$-ary Cuckoo Hash Table

- Migrated Region

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

Old $d$-ary Cuckoo Hash Table

- T1: k, e
- T2: b, c, a
- T3: f, l, g

Migrated Region

New $d$-ary Cuckoo Hash Table

- T1':
- T2': m
- T3':
Elastic Cuckoo Migration

Old $d$-ary Cuckoo Hash Table

- T1
  - e
  - P1

- T2
  - c
  - a

- T3
  - f
  - l
  - g

New $d$-ary Cuckoo Hash Table

- T1'
  - m

- T2'
  - k

- T3'
  - b

Migrated Region
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

- $T1$: $e$
- $T2$: $c$
- $T3$: $f$

Migrated Region

New $d$-ary Cuckoo Hash Table

- $T1'$
- $T2'$
- $T3'$
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

- T1
- T2
- T3

Migrated Region

New $d$-ary Cuckoo Hash Table

- T1'
- T2'
- T3'

P1 → P2 → P3

H1 < H2 < H3

H'1

H'3

m

Migrated Region
Elastic Cuckoo Lookup

Old $d$-ary Cuckoo Hash Table

- $T_1$: Migrated Region
- $T_2$: $c$
- $T_3$: $a$

New $d$-ary Cuckoo Hash Table

- $T_1'$
- $T_2'$
- $T_3'$

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

No sequential page walk (unlike radix)

At most $d$ accesses always

Leverages multiple issue out-of-order processors

$d$-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

No sequential page walk (unlike radix)

Leverages multiple issue out-of-order processors

VA$_k$
Lookup Multiple Page Sizes in Parallel

- 4KB PTE Entries
  - T1: d-ary Elastic Cuckoo Page Table
  - T2: 4KB PTE Entries
  - T3: 4KB PTE Entries

- 2MB PMD Entries
  - T1: Elastic Cuckoo Page Table
  - T2: 2MB PMD Entries
  - T3: 2MB PMD Entries

- 1GB PUD Entries
  - T1: Elastic Cuckoo Page Table
  - T2: 1GB PUD Entries
  - T3: 1GB PUD Entries
New MMU Cache to Prune Parallelism

- 4KB PTE Entries
- 2MB PMD Entries
- 1GB PUD Entries

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

Elastic Cuckoo Page Table

VA MMU Cache

VA2MB

VA1GB
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

Speedup

0.4 0.6 0.8 1

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

<table>
<thead>
<tr>
<th>Application</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>0.4</td>
</tr>
<tr>
<td>BFS</td>
<td>0.6</td>
</tr>
<tr>
<td>CC</td>
<td>0.8</td>
</tr>
<tr>
<td>DC</td>
<td>1.0</td>
</tr>
<tr>
<td>DFS</td>
<td>1.2</td>
</tr>
<tr>
<td>GUPS</td>
<td>1.4</td>
</tr>
<tr>
<td>MUMmer</td>
<td>1.0</td>
</tr>
<tr>
<td>PR</td>
<td>1.0</td>
</tr>
<tr>
<td>SSSP</td>
<td>1.0</td>
</tr>
<tr>
<td>SysBench</td>
<td>1.0</td>
</tr>
<tr>
<td>TC</td>
<td>1.0</td>
</tr>
<tr>
<td>Mean</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Radix 4KB
Application Speedup

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

- Radix 4KB
- Radix THP
- Cuckoo 4KB

<table>
<thead>
<tr>
<th>Application</th>
<th>Radix 4KB</th>
<th>Radix THP</th>
<th>Cuckoo 4KB</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>BFS</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>CC</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>DC</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>DFS</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>GUPS</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>MUMmer</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PR</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>SSSP</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>SysBench</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>TC</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Mean</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Application Speedup

- Radix 4KB
- Radix THP
- Cuckoo 4KB
- Cuckoo THP

<table>
<thead>
<tr>
<th>Application</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>3.31</td>
<td>3.45</td>
<td>1.64</td>
<td>1.82</td>
</tr>
<tr>
<td>MUMmer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>3.31</td>
<td>3.45</td>
<td>1.64</td>
<td>1.82</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>3.31</td>
<td>3.45</td>
<td>1.64</td>
<td>1.82</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
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