Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism

Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu, Josep Torrellas

University of Illinois at Urbana-Champaign

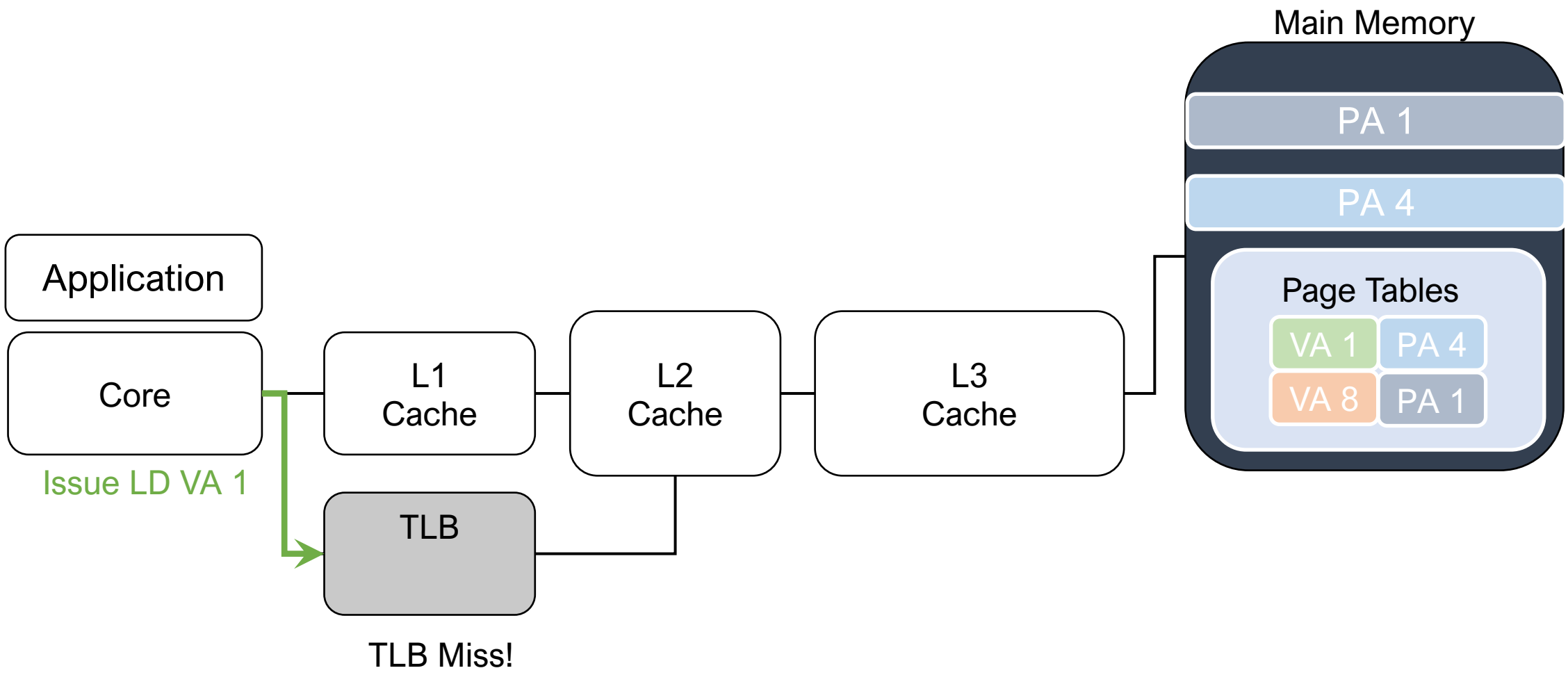
skarlat2.web.engr.illinois.edu

ASPLOS 2020

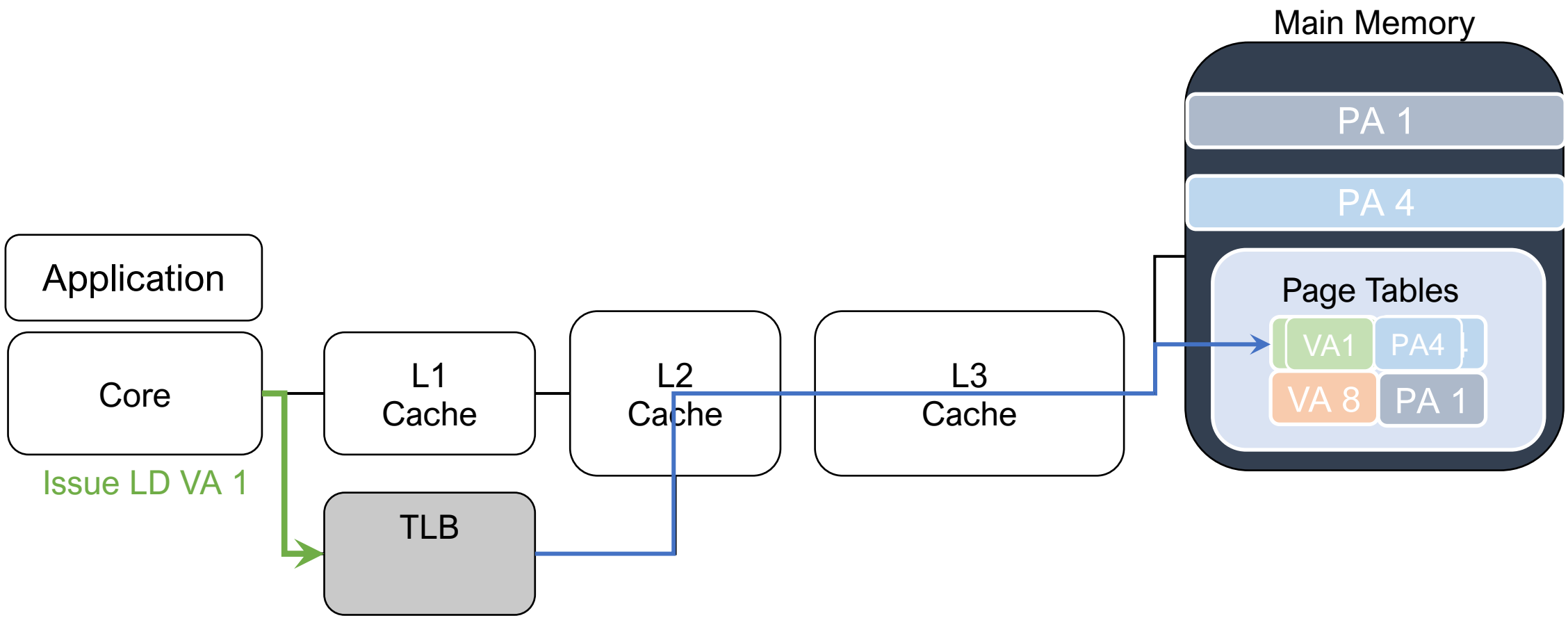
ILLINOIS

i-acoma group

Virtual Memory Translation is Expensive



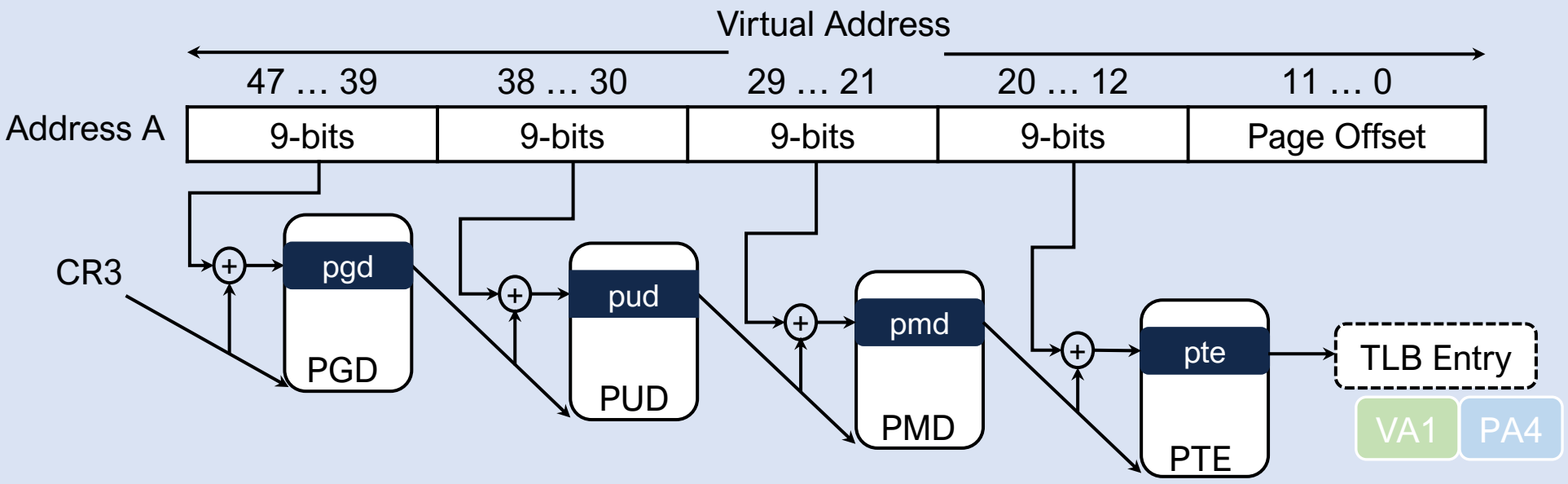
Virtual Memory Translation is Expensive



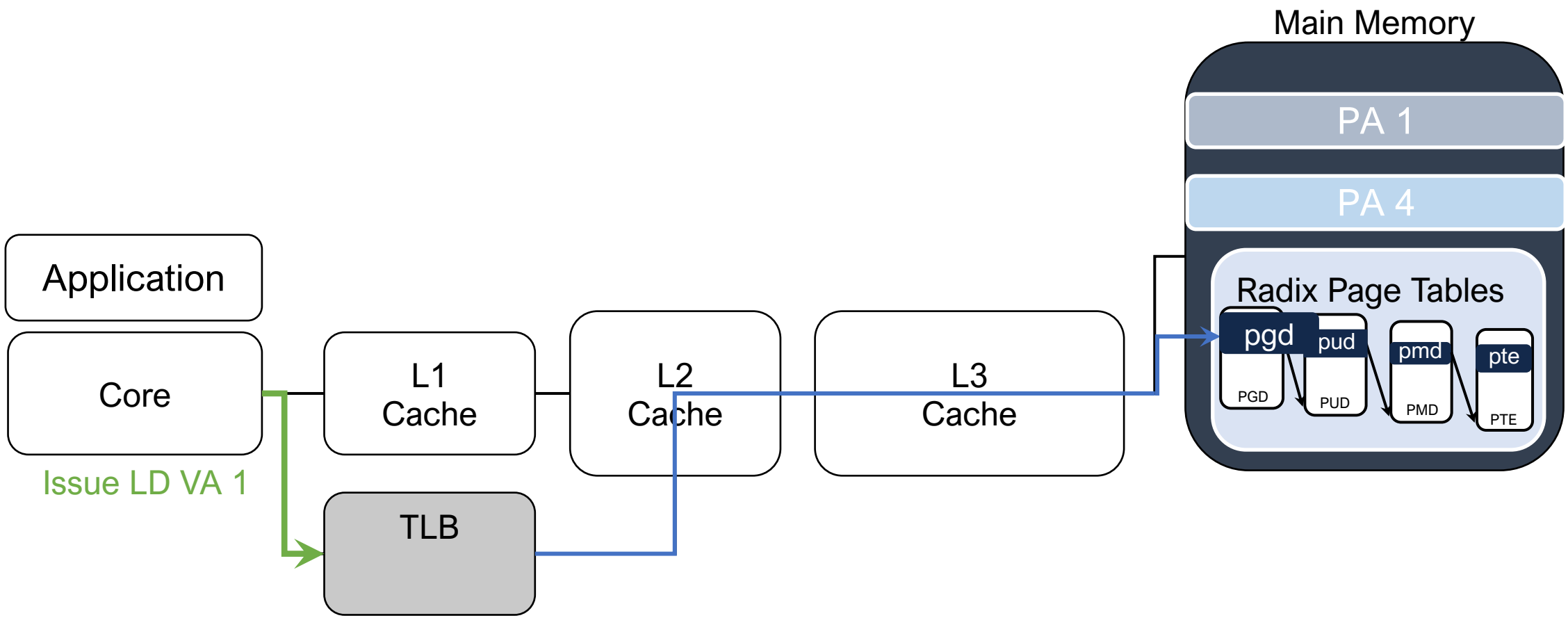
TLB Miss → "Page Walk" = Fetch entry from page table

x86-64 Radix Page Tables

x86-64 Radix Page Tables

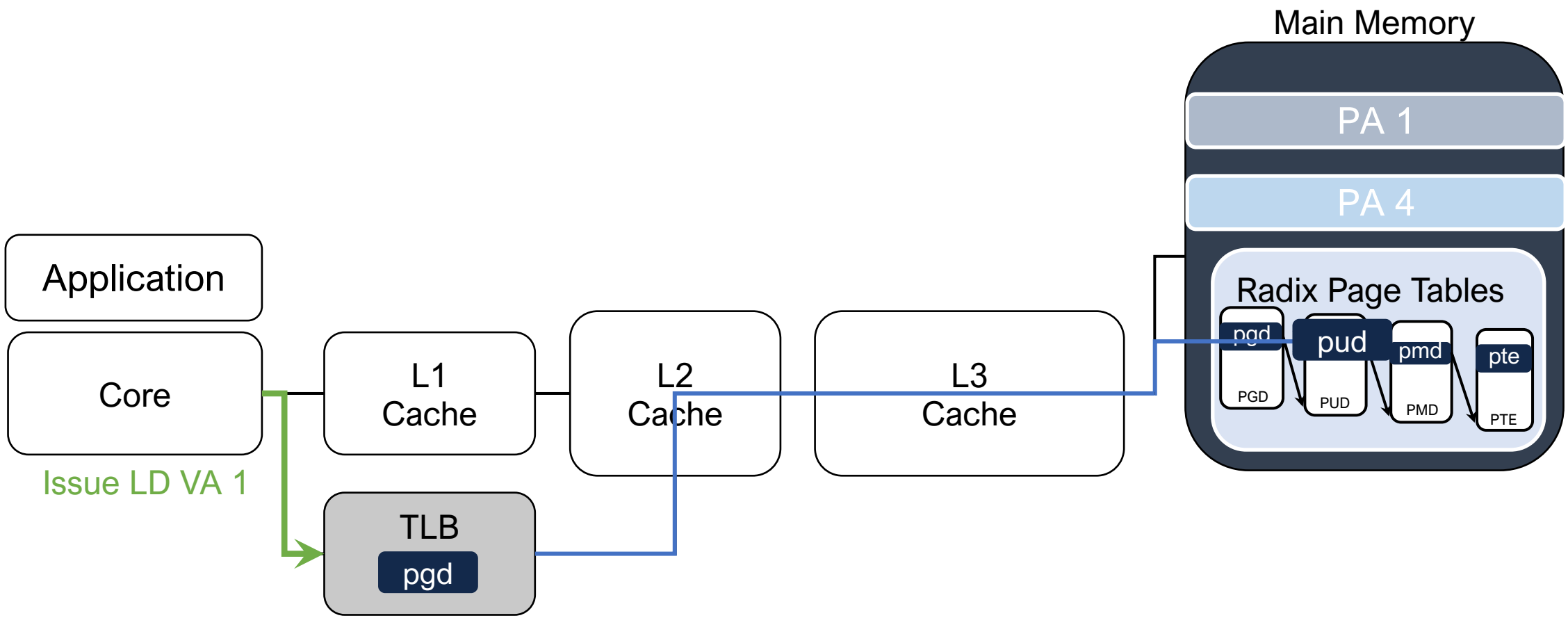


Virtual Memory Translation is Expensive



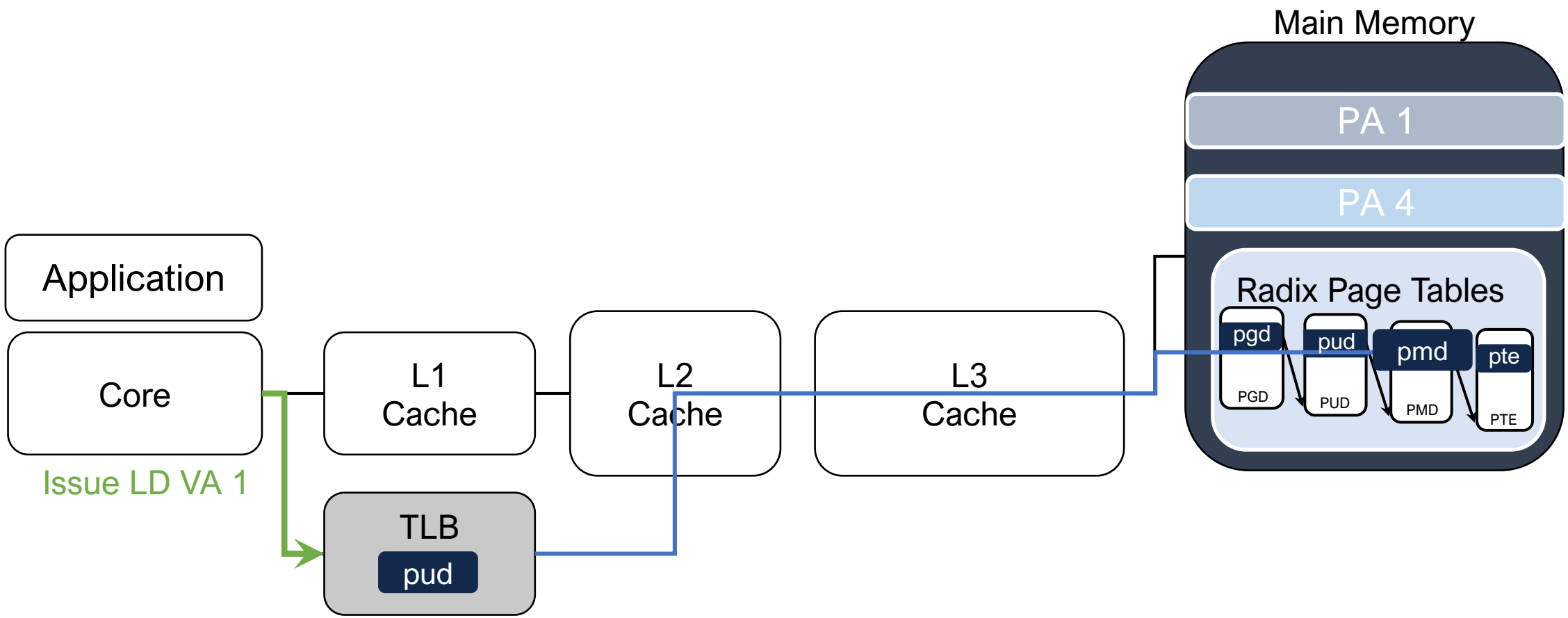
TLB Miss → "Page Walk" = Fetch entry from radix page table

Virtual Memory Translation is Expensive



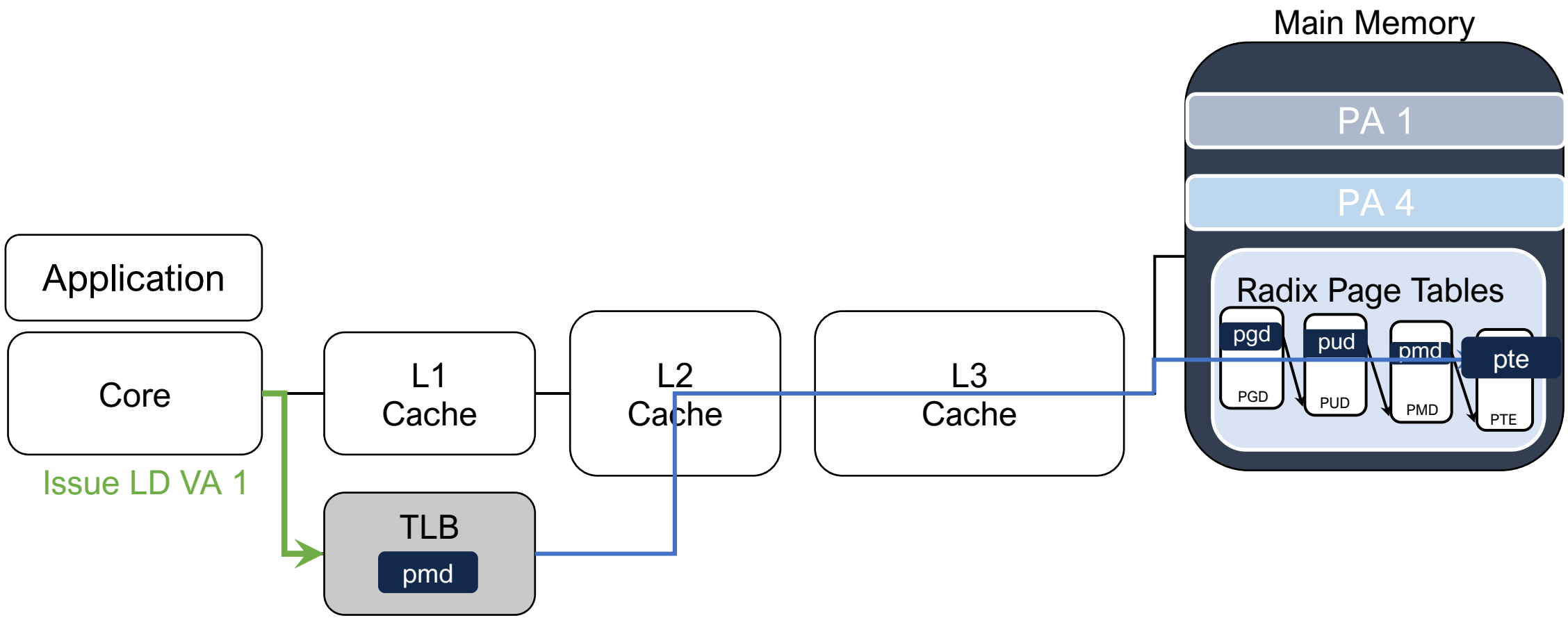
TLB Miss → "Page Walk" = Fetch entry from radix page table

Virtual Memory Translation is Expensive



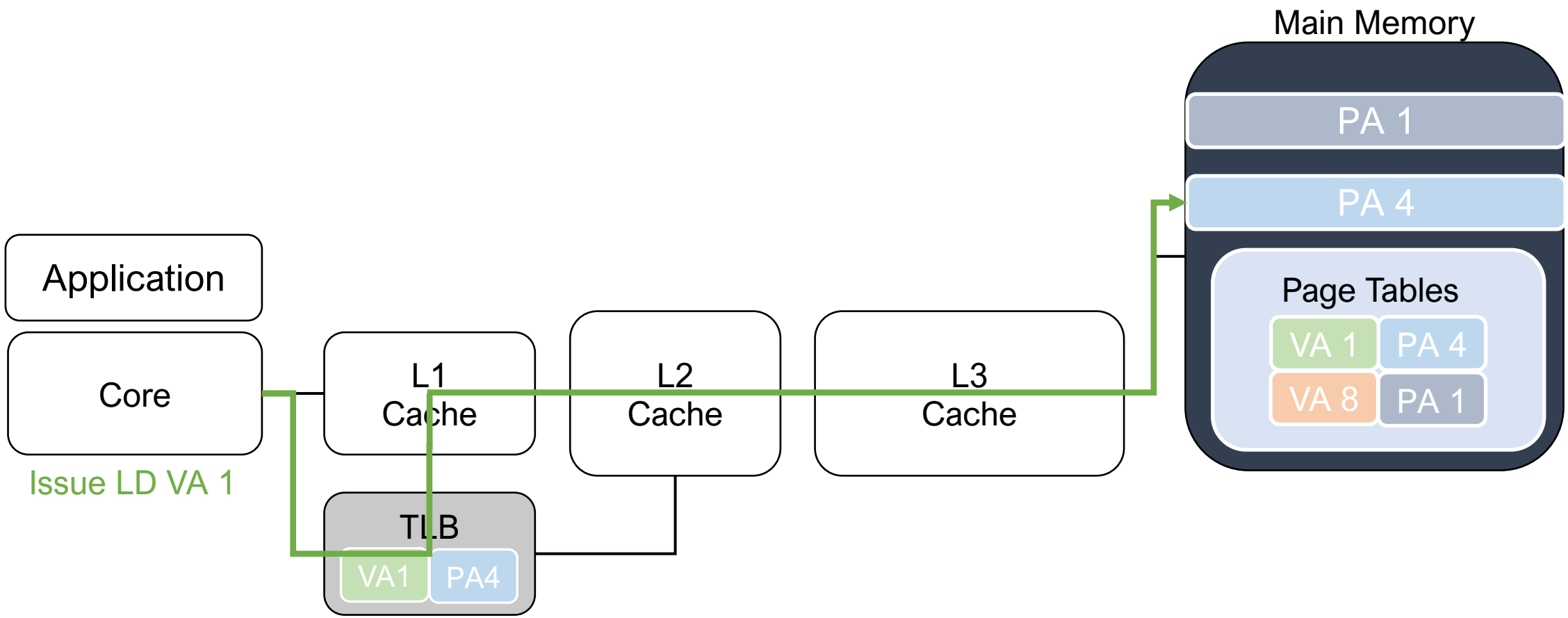
TLB Miss → "Page Walk" = Fetch entry from radix page table

Virtual Memory Translation is Expensive

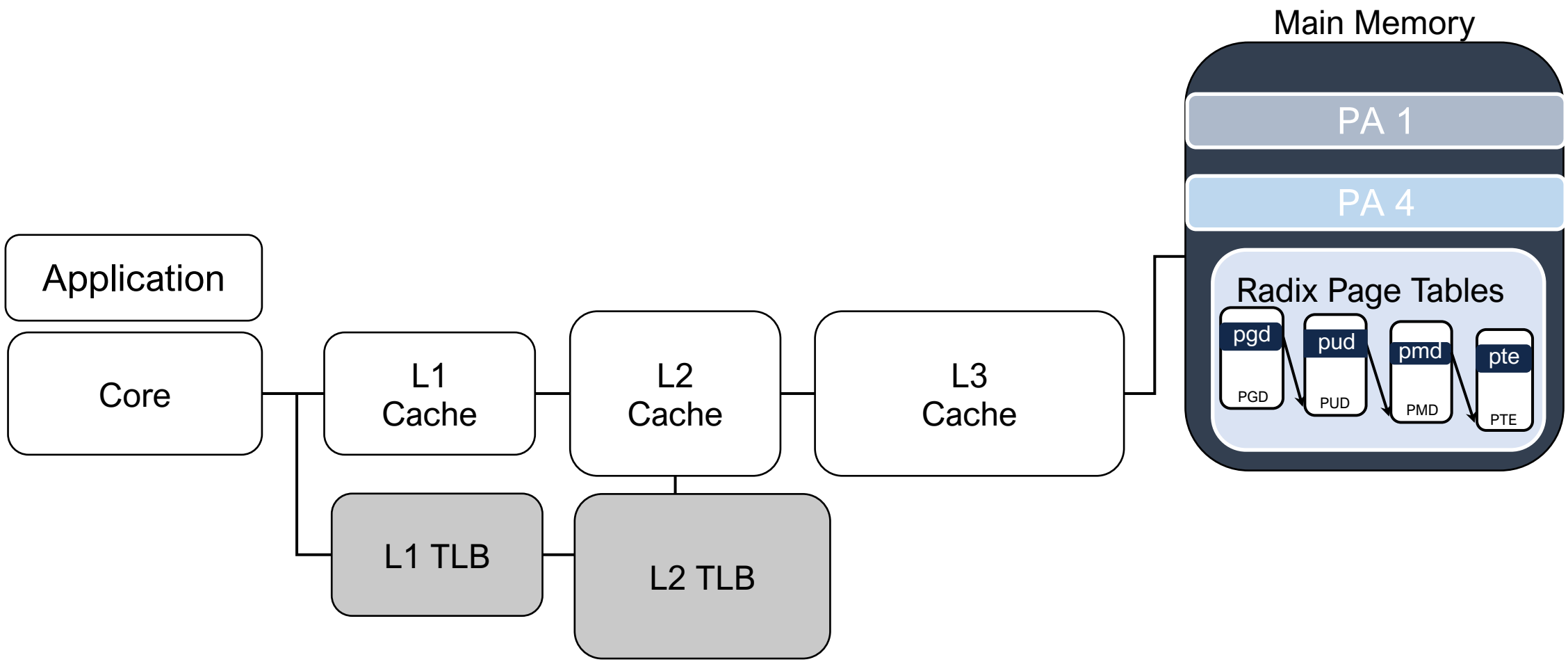


TLB Miss → "Page Walk" = Fetch entry from radix page table

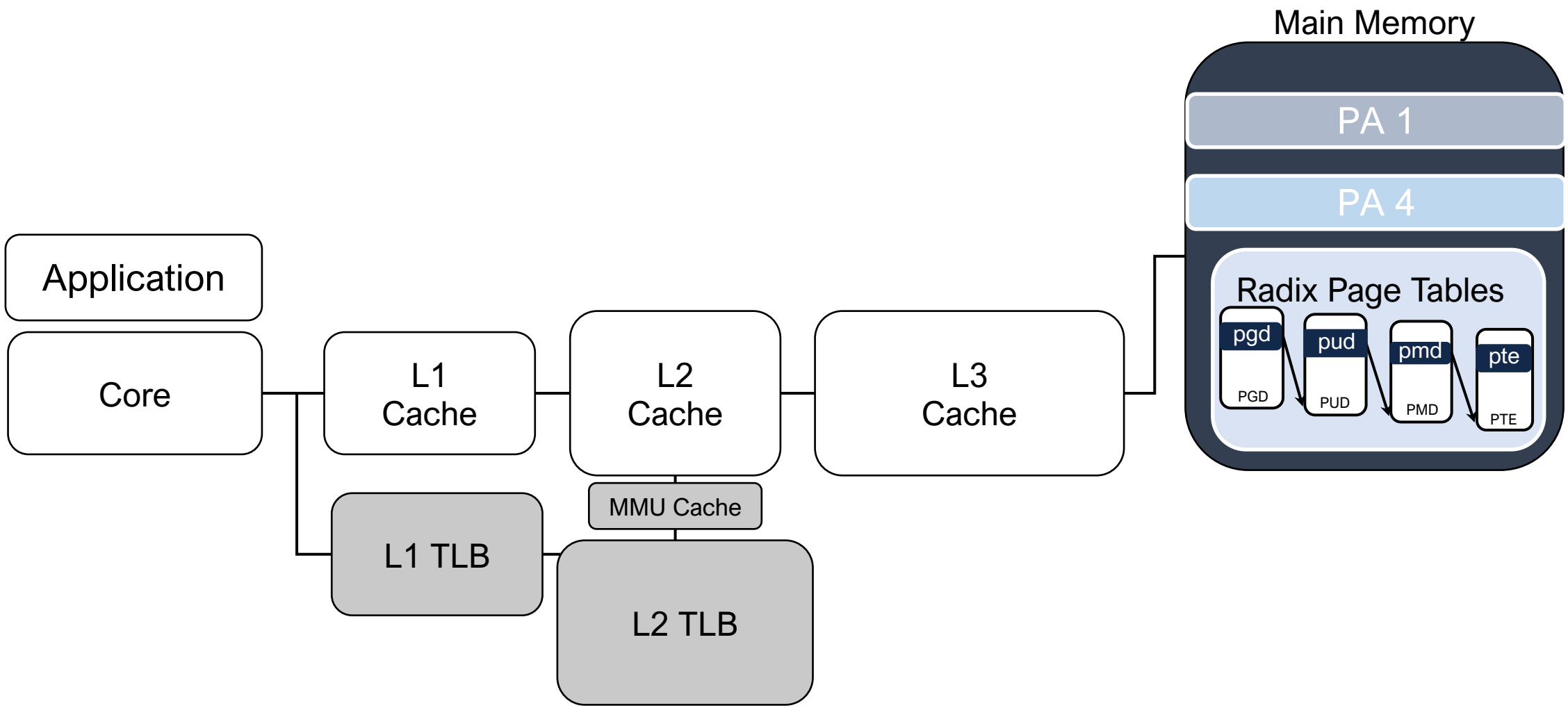
Virtual Memory Translation is Expensive



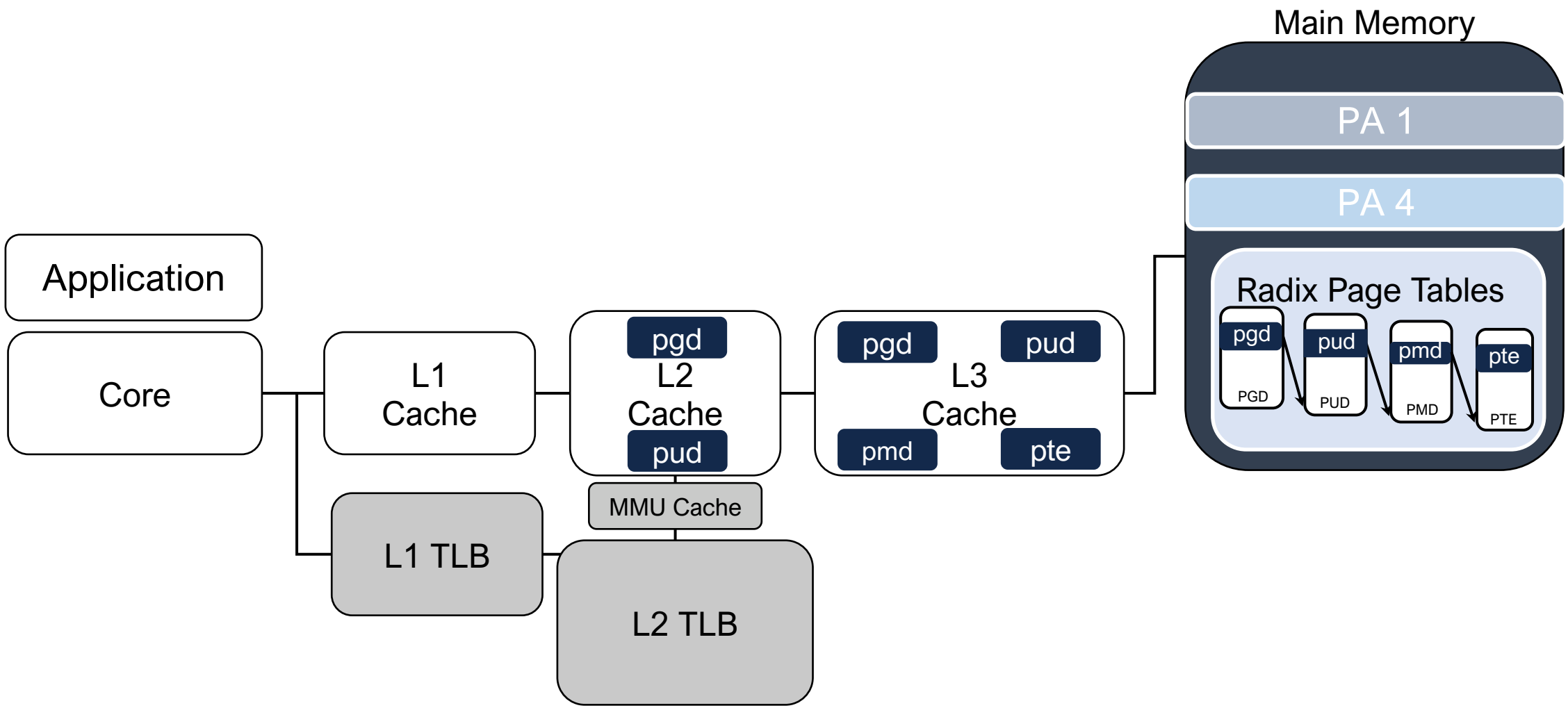
Multilevel TLBs



Memory Management Unit (MMU) Cache

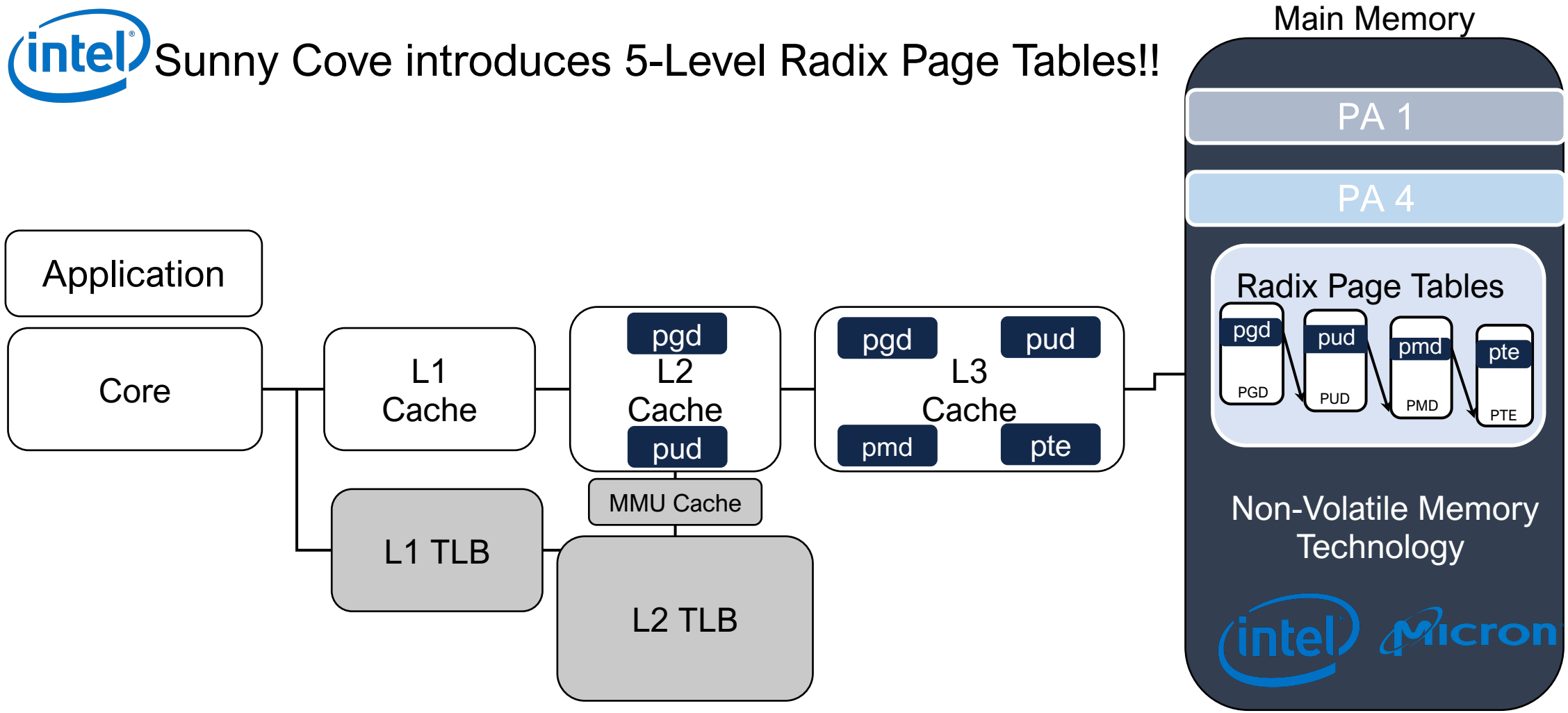


Translations in Data Caches



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

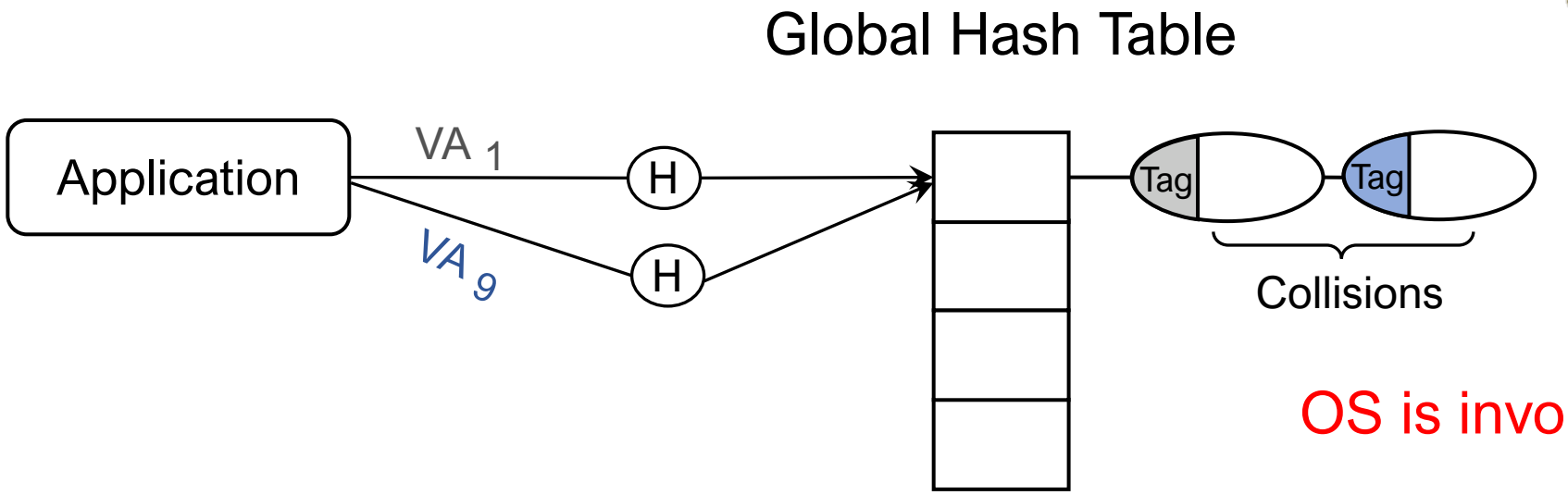
- Idea: Dynamically resizable page tables based on cuckoo hashing
- No sequential page table lookups → parallel single-step lookups

- 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



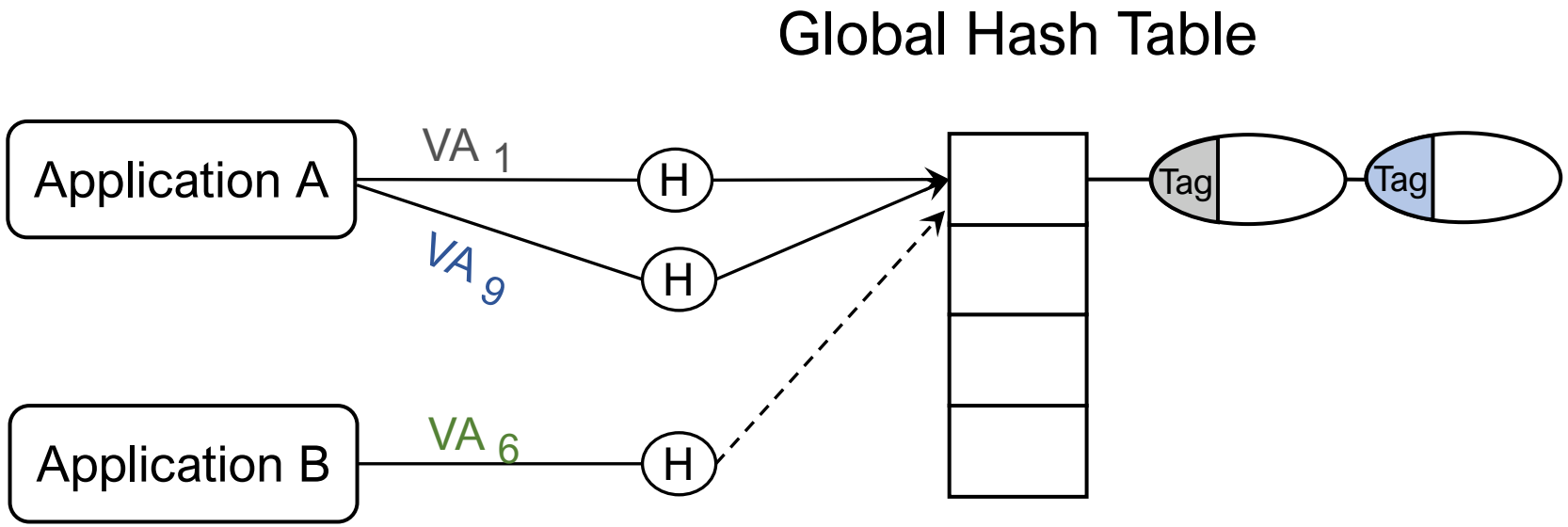
OS is invoked to resolve them!

Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

How to share pages?

Multiple page sizes?



Alternative: A Global Hashed Page Table

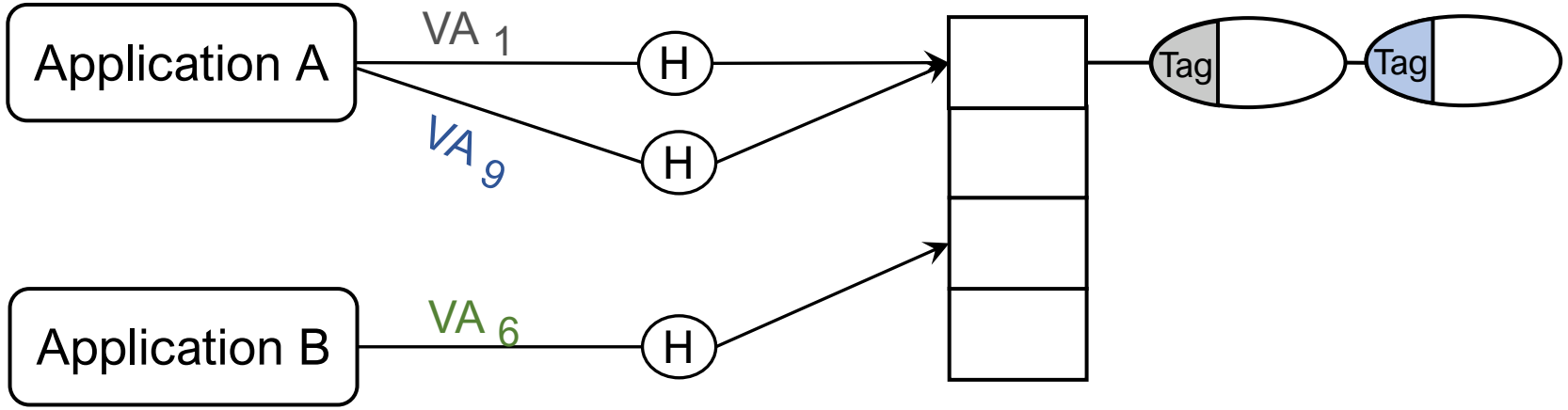
The old approach from Intel and IBM

How to share pages?

Multiple page sizes?

New level of indirection!!

Global Hash Table



Alternative: A Global Hashed Page Table

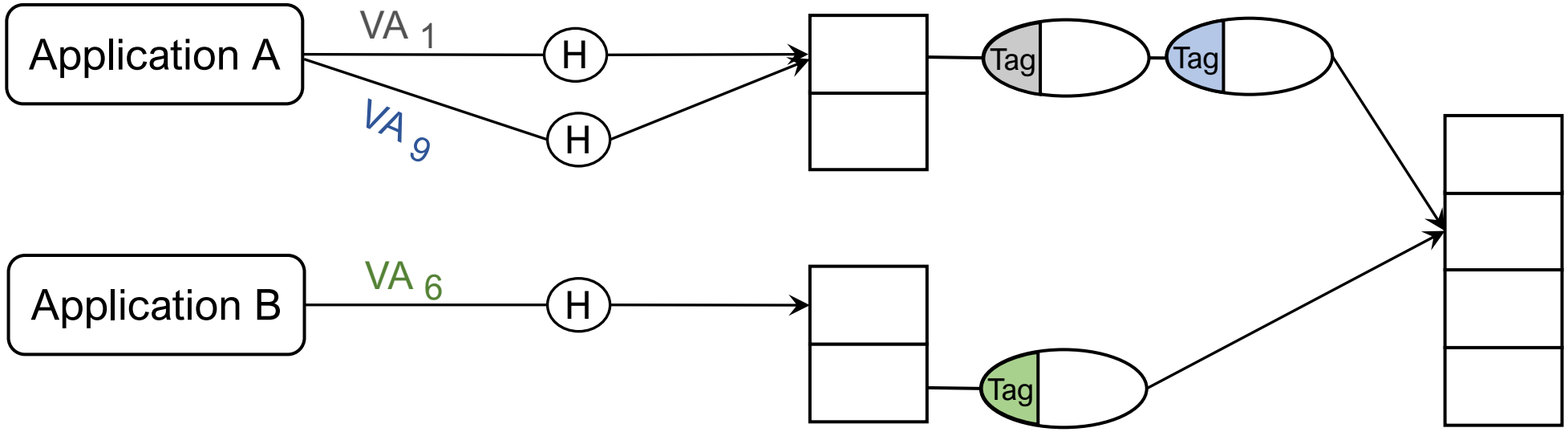
The old approach from Intel and IBM

How to share pages?

Multiple page sizes?

New level of indirection!!

Global Hash Table



Alternative: A Global Hashed Page Table

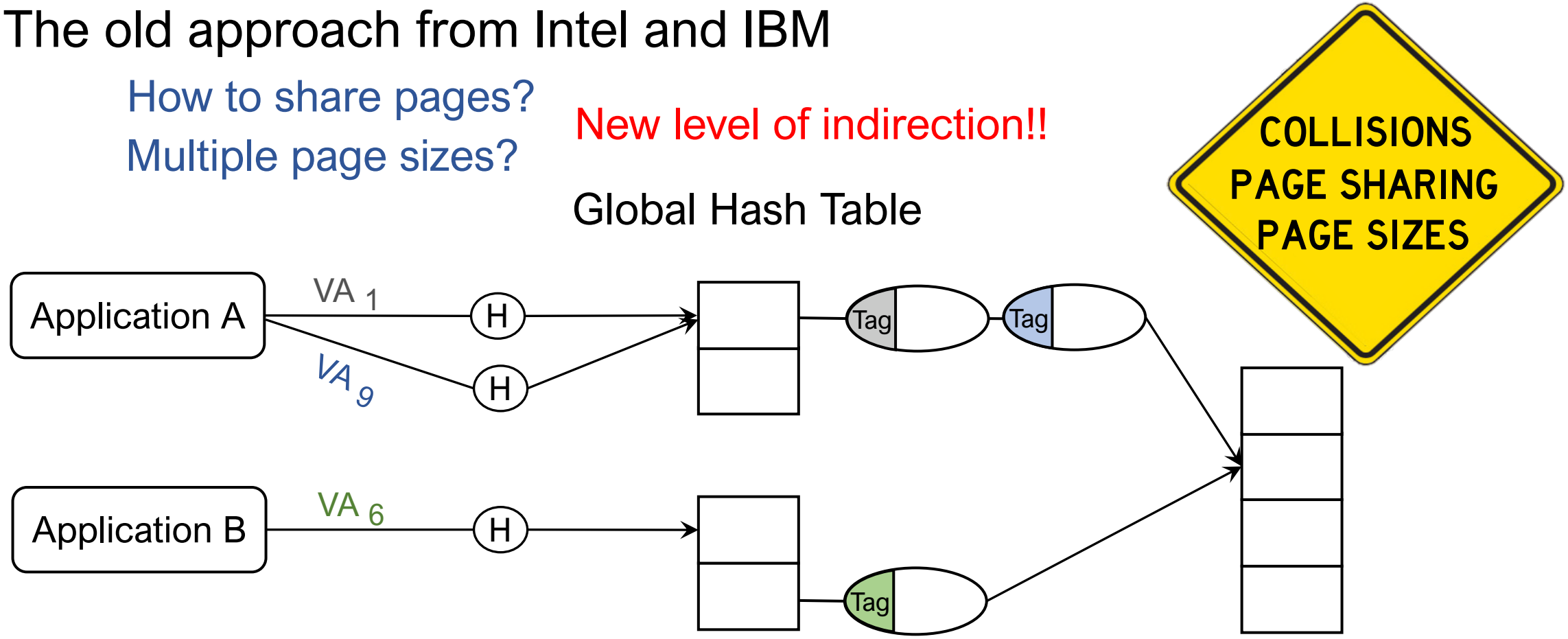
The old approach from Intel and IBM

How to share pages?

Multiple page sizes?

New level of indirection!!

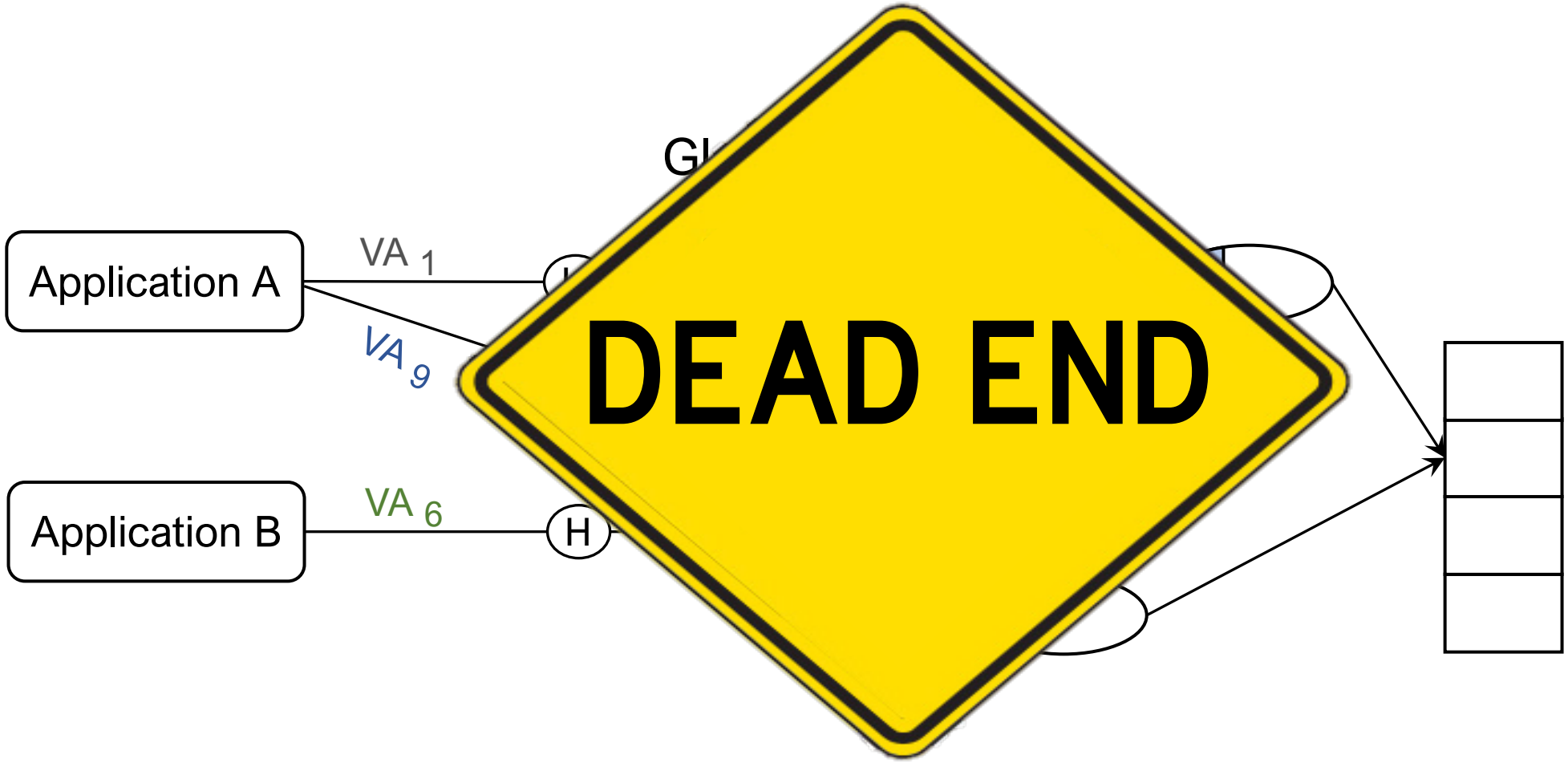
Global Hash Table



Alternative: A Global Hashed Page Table

The old approach from Intel and IBM

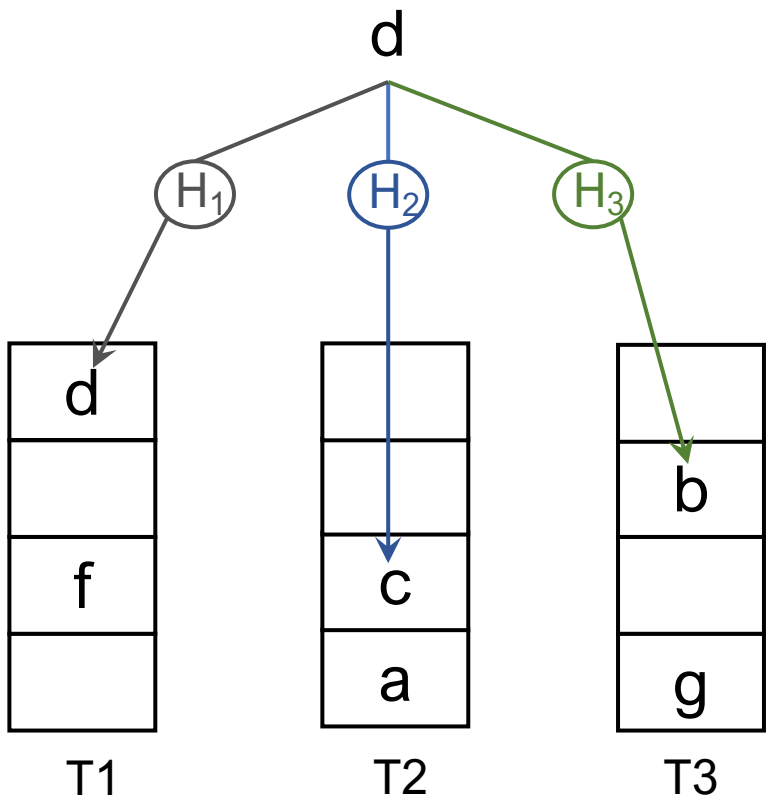
Switched to radix page tables!



Elastic Cuckoo Page Tables

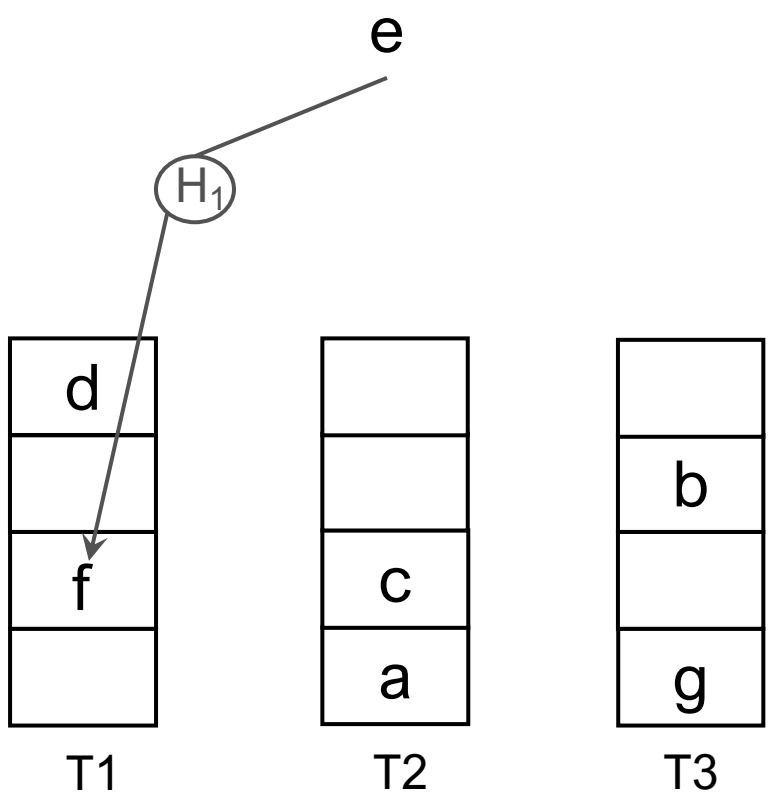
Rethinking virtual memory translation for parallelism

Cuckoo Hashing [Pagh 2001, Fotakis 2005]



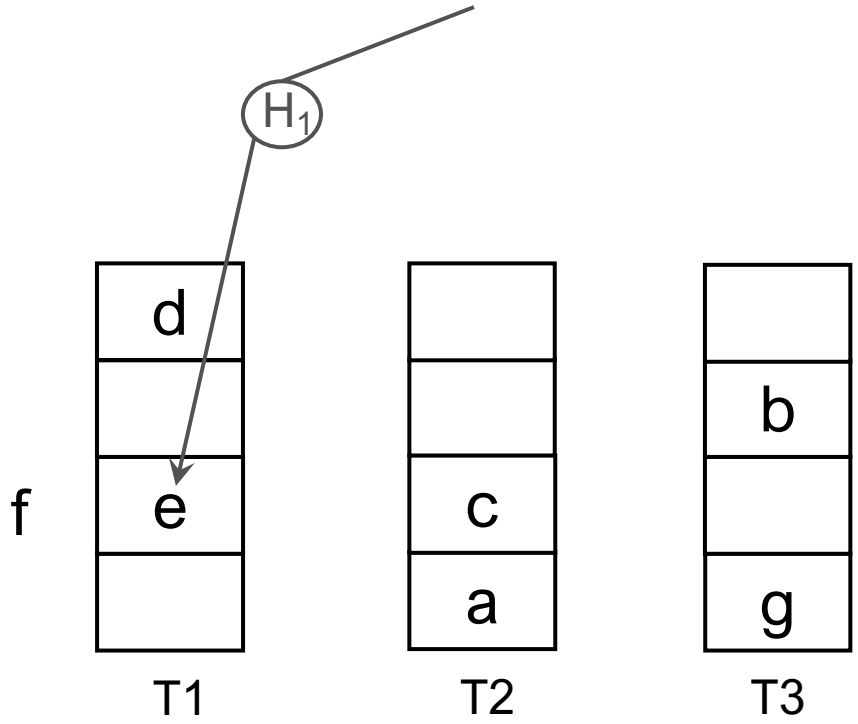
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



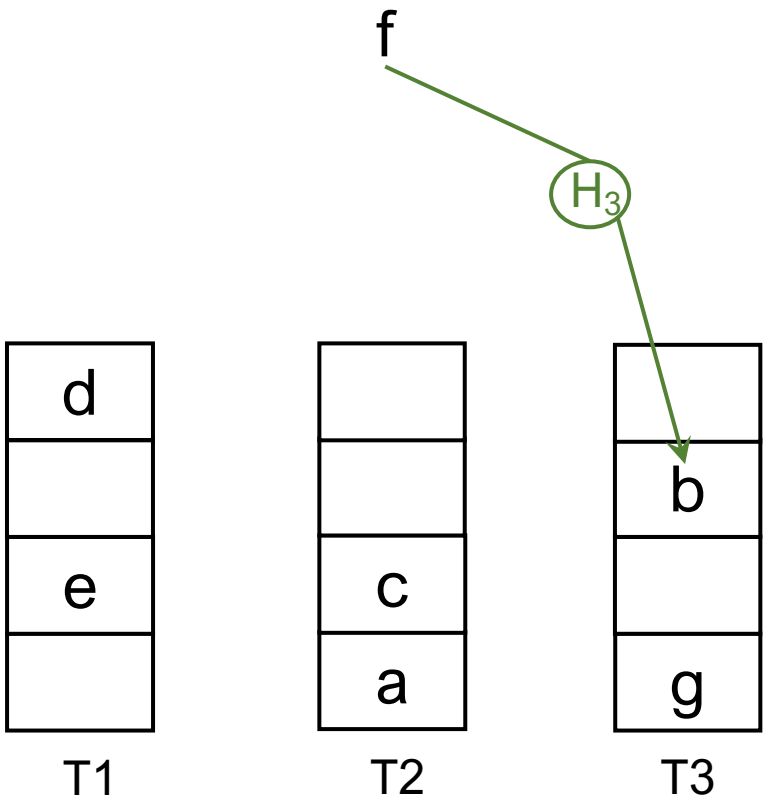
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



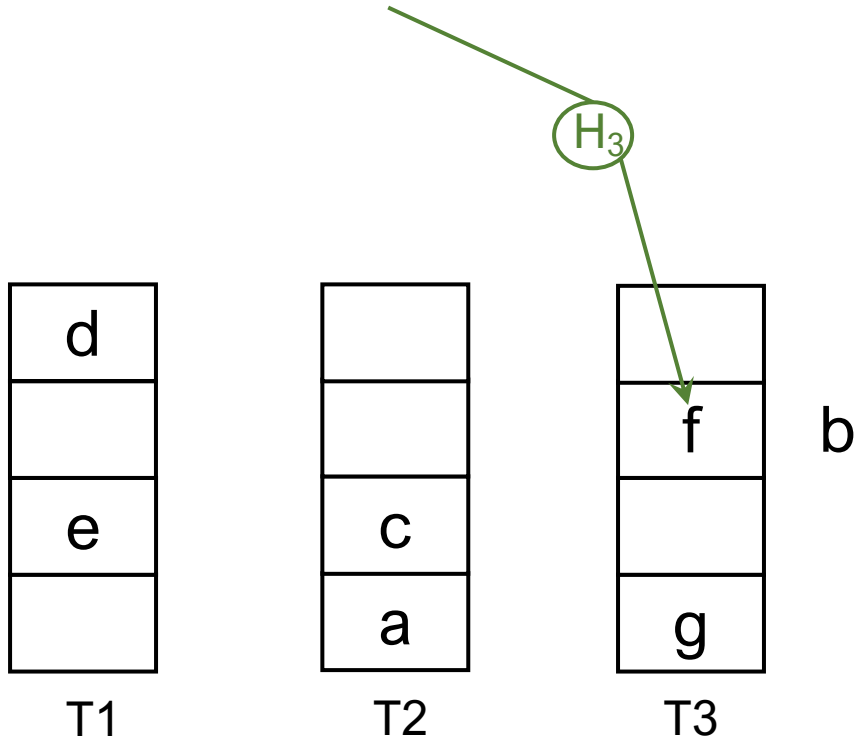
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



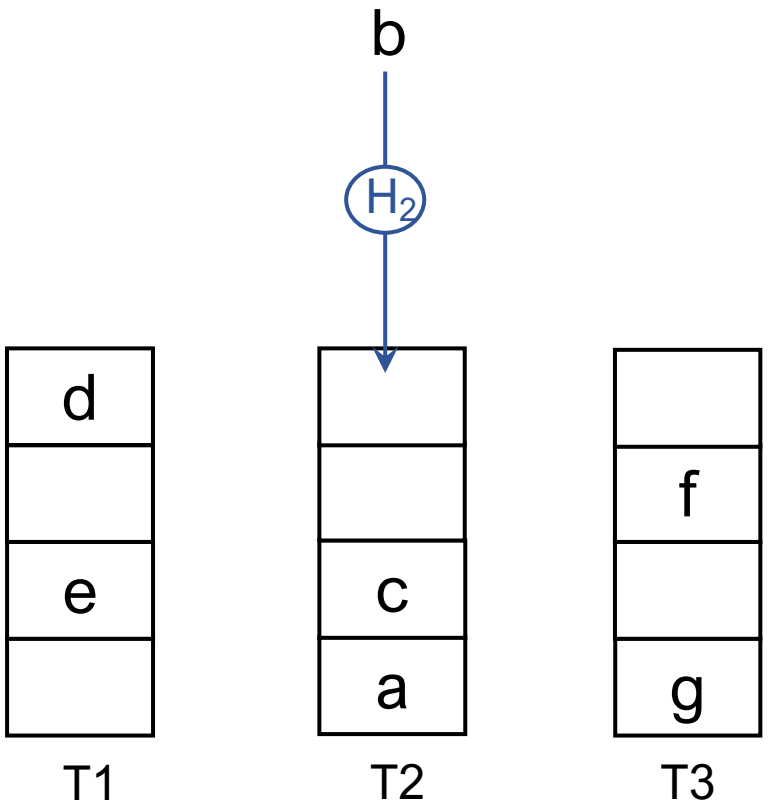
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



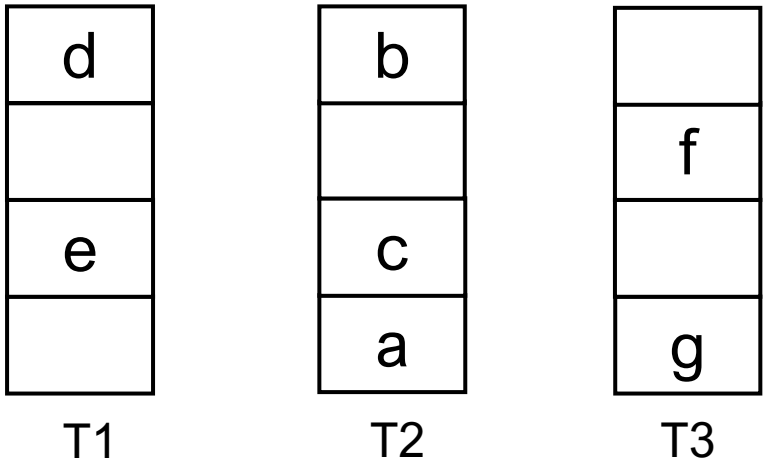
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



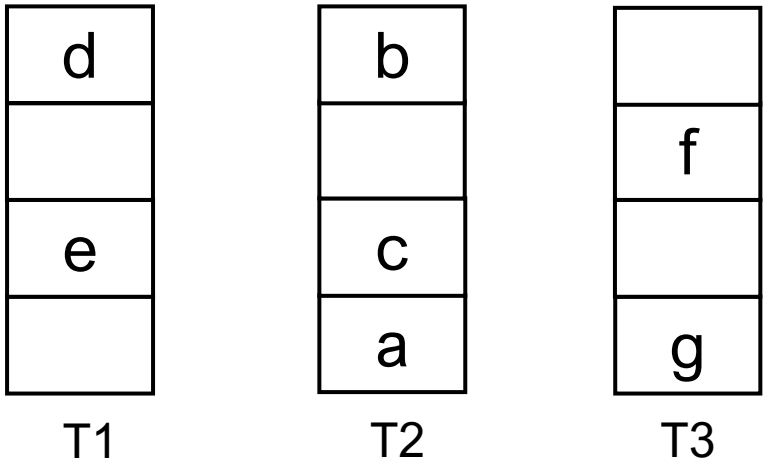
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing

T1	T2	T3
d	b	
		f
e	c	
	a	g

d-ary Cuckoo Hash Table

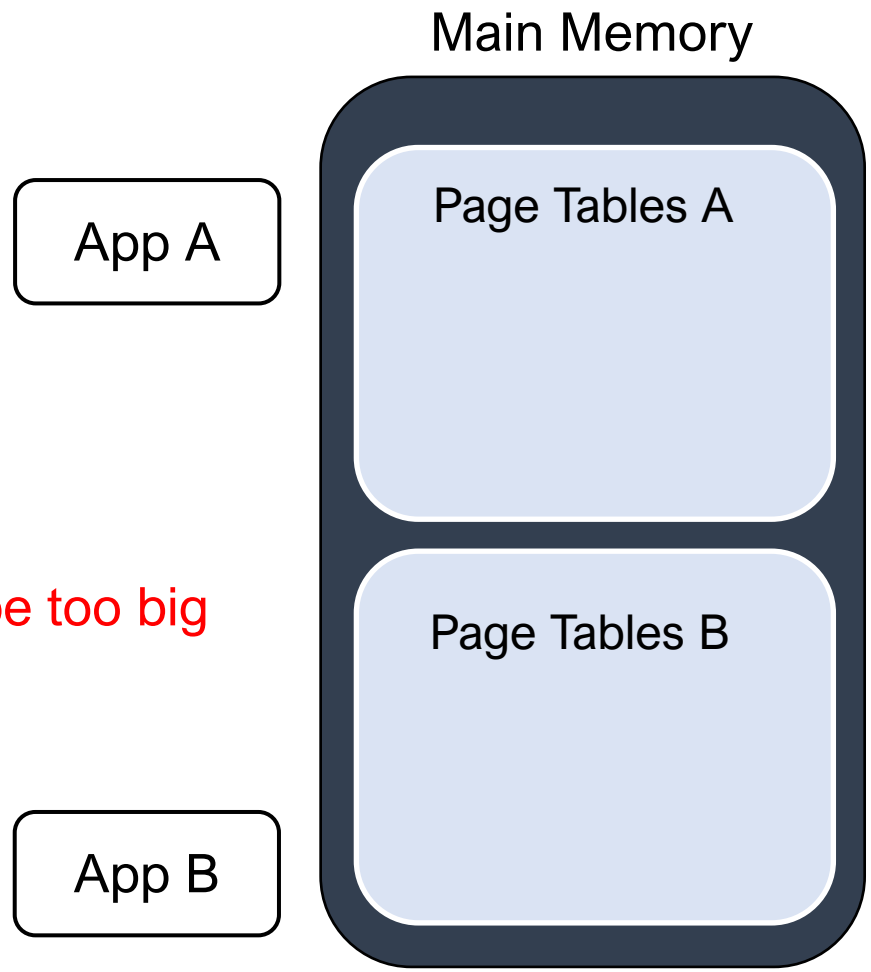
Private Hashed Page Tables



d-ary Cuckoo Hash Table

Cannot Be Too Big → Waste Memory

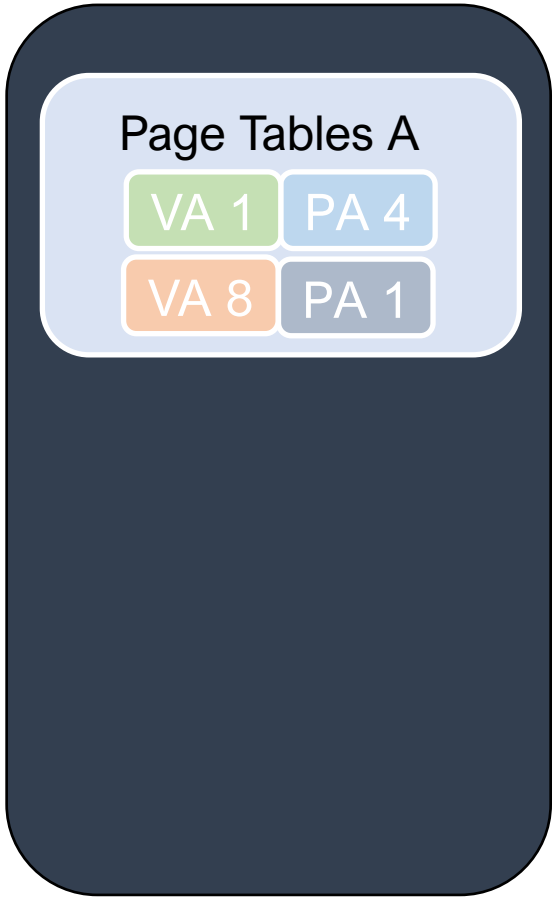
Private page tables cannot be too big



Need to Dynamically Resize

App A

Main Memory



Private page tables cannot be too big

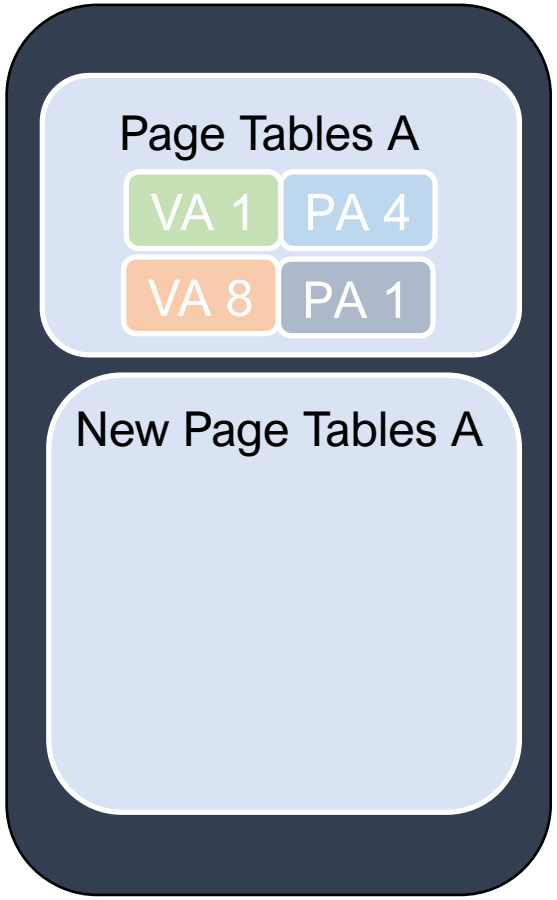
↓

Need to dynamically resize

Need to Dynamically Resize

App A

Main Memory

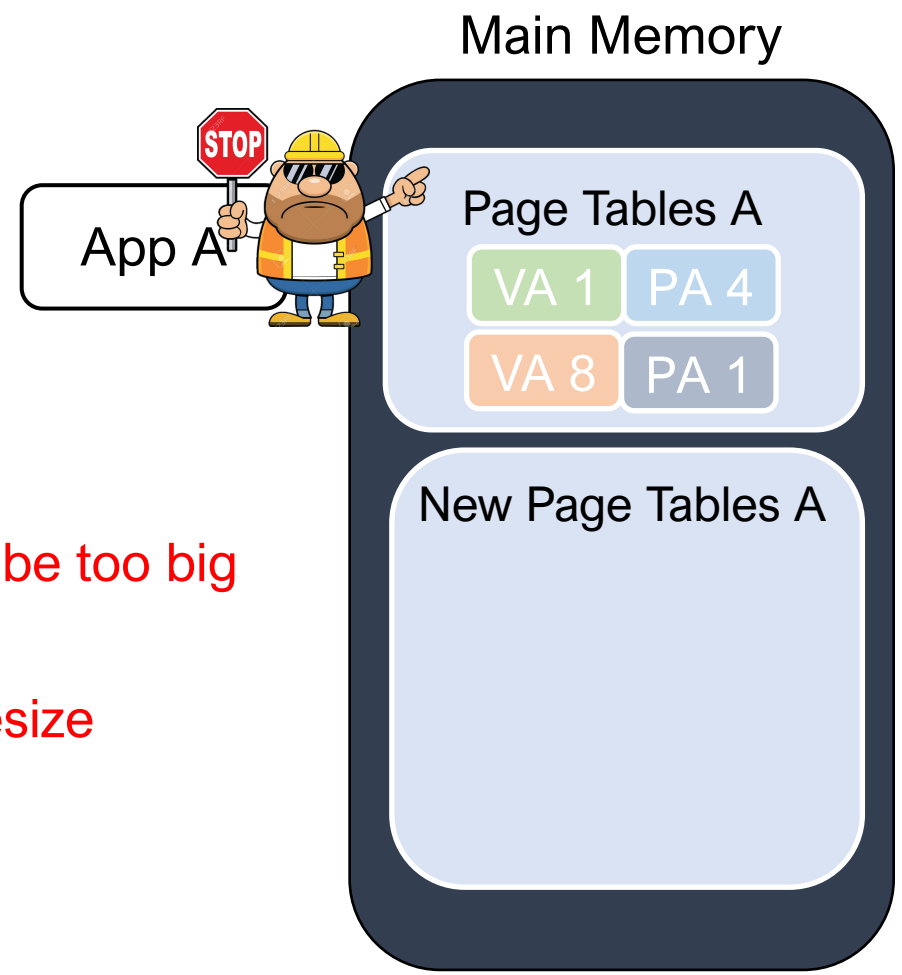


Private page tables cannot be too big

↓

Need to dynamically resize

Cannot Rehash All Entries at Once



Private page tables cannot be too big

↓

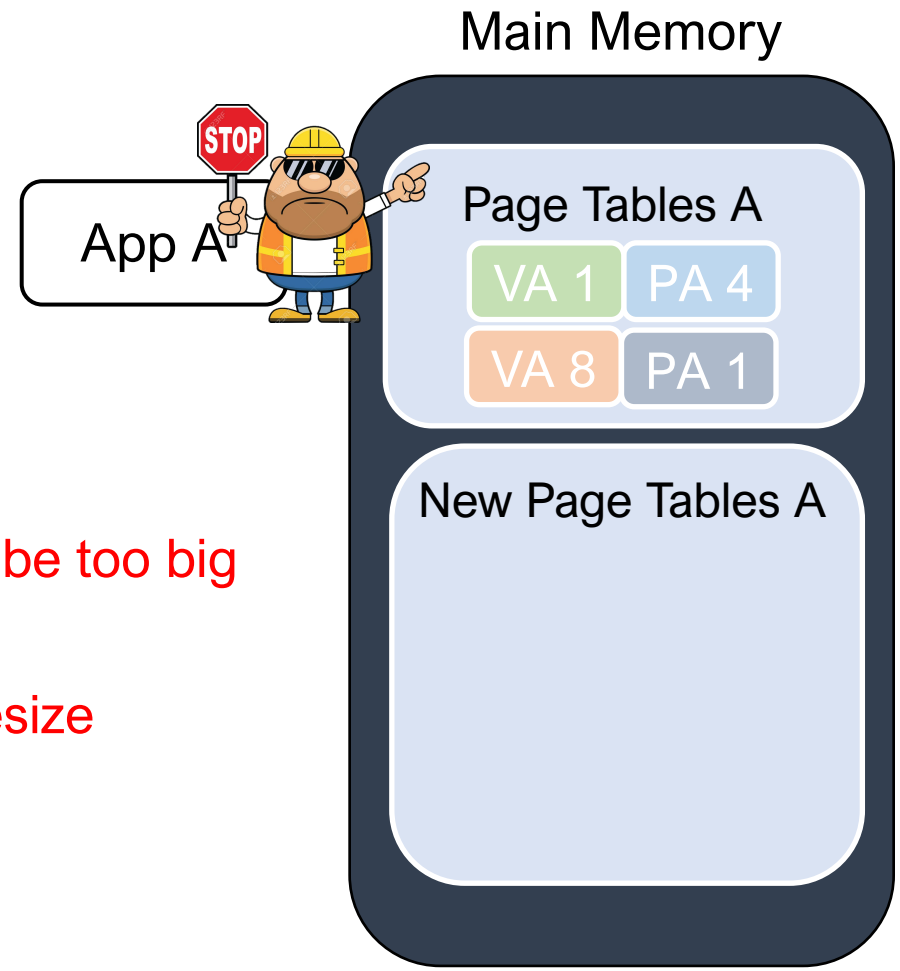
Need to dynamically resize

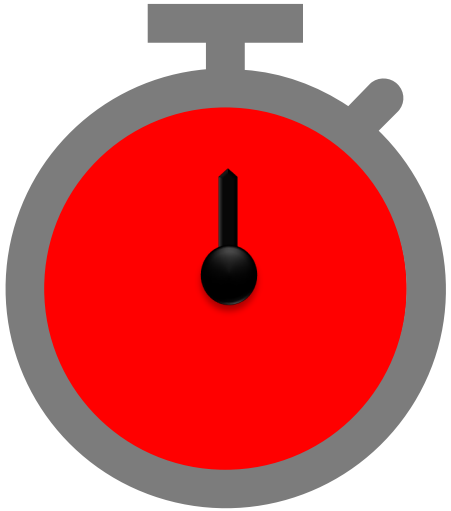
Cannot Rehash All Entries at Once

Private page tables cannot be too big

↓

Need to dynamically resize

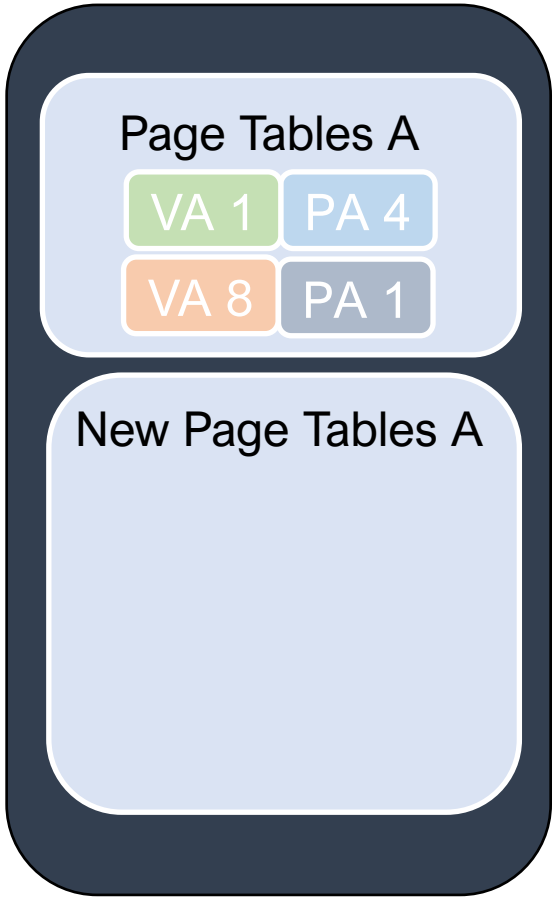




Cannot Rehash All Entries at Once

App A

Main Memory



Private page tables cannot be too big

↓

Need to dynamically resize

↓

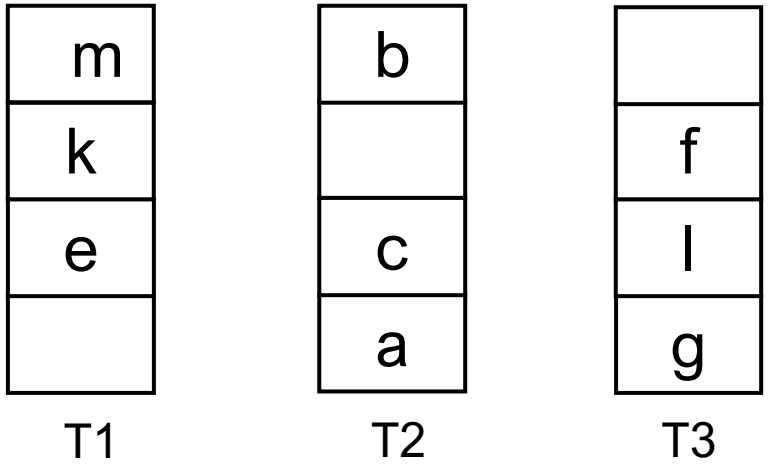
While the program is running

↓

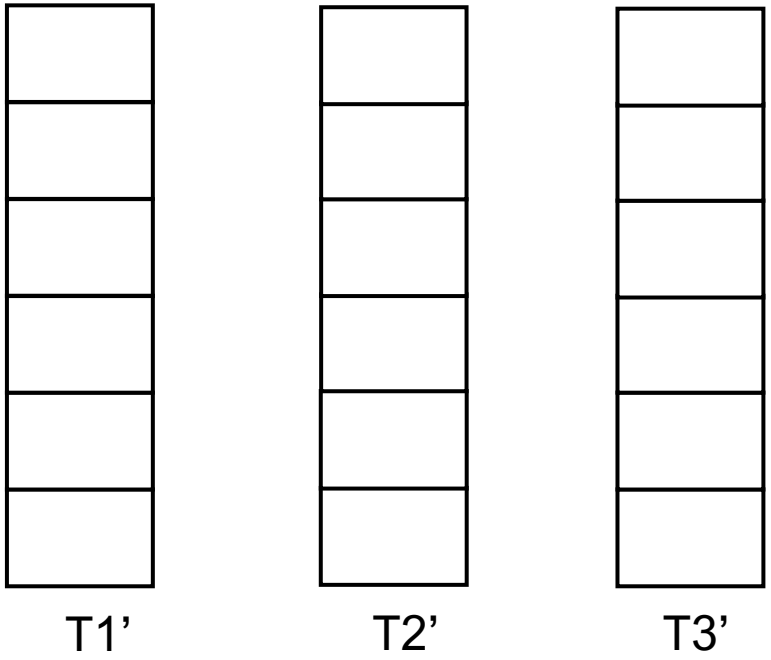
Gradual Resizing!

Gradual Resizing Cuckoo Hash Tables

At every insert → Rehash one element



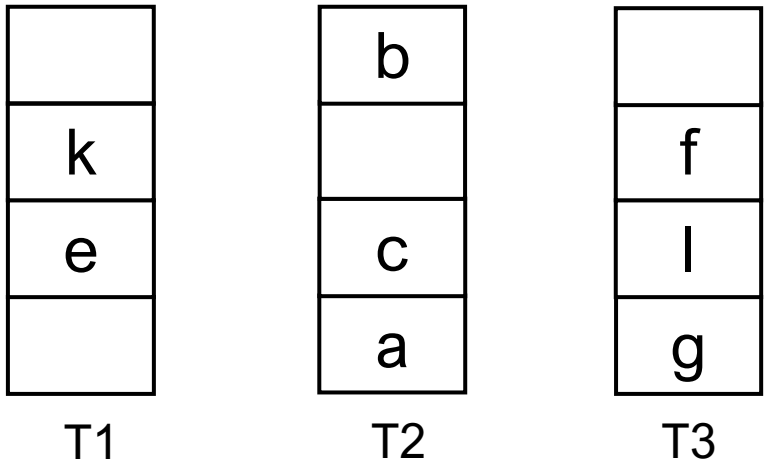
Old *d*-ary Cuckoo Hash Table



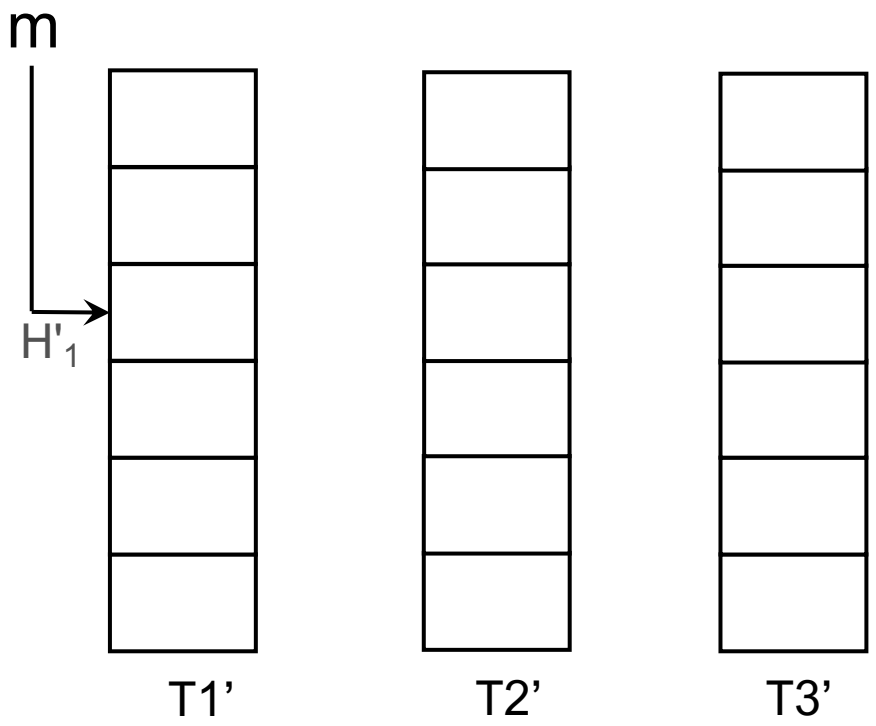
New *d*-ary Cuckoo Hash Table

Gradual Resizing Cuckoo Hash Tables

At every insert → Rehash one element

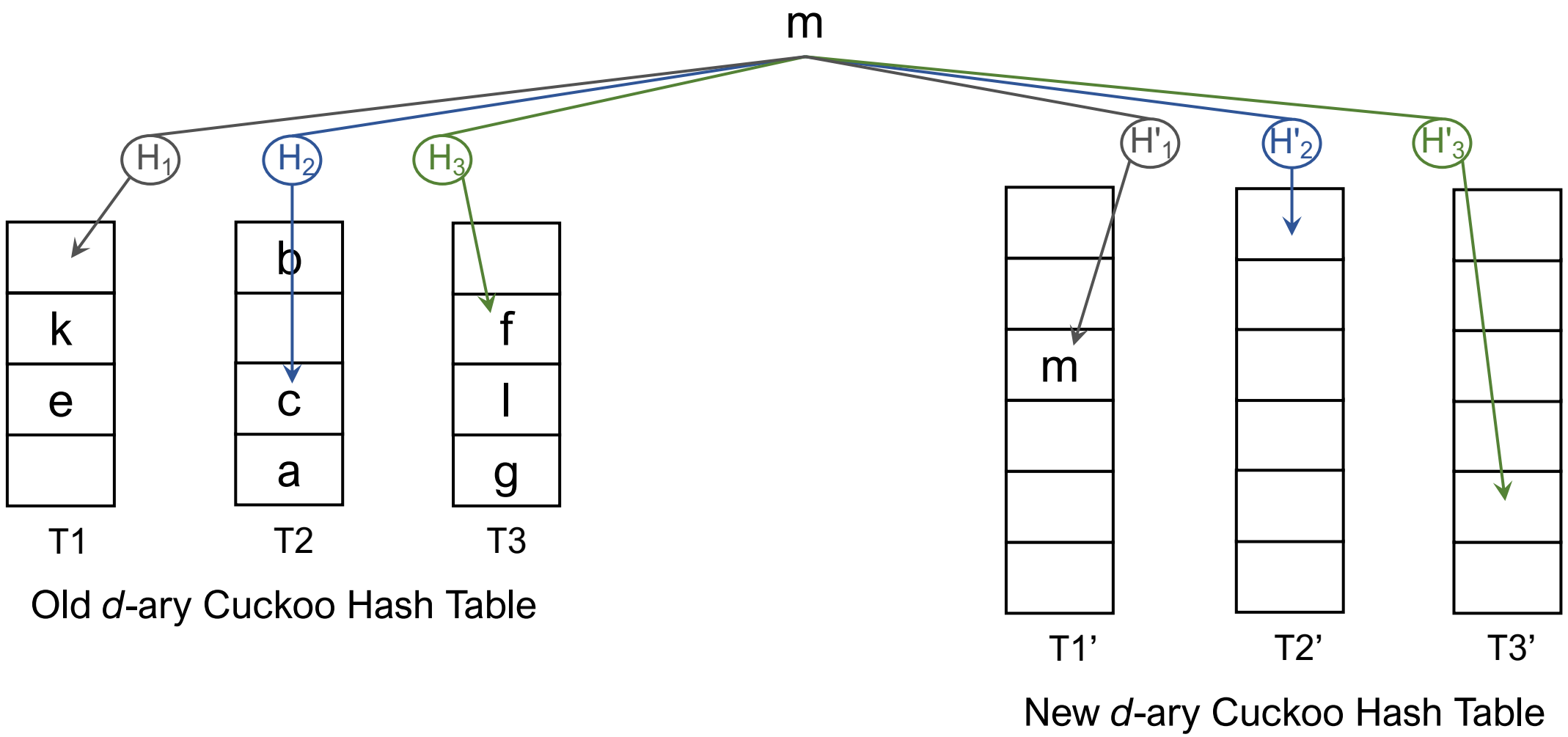


Old *d*-ary Cuckoo Hash Table

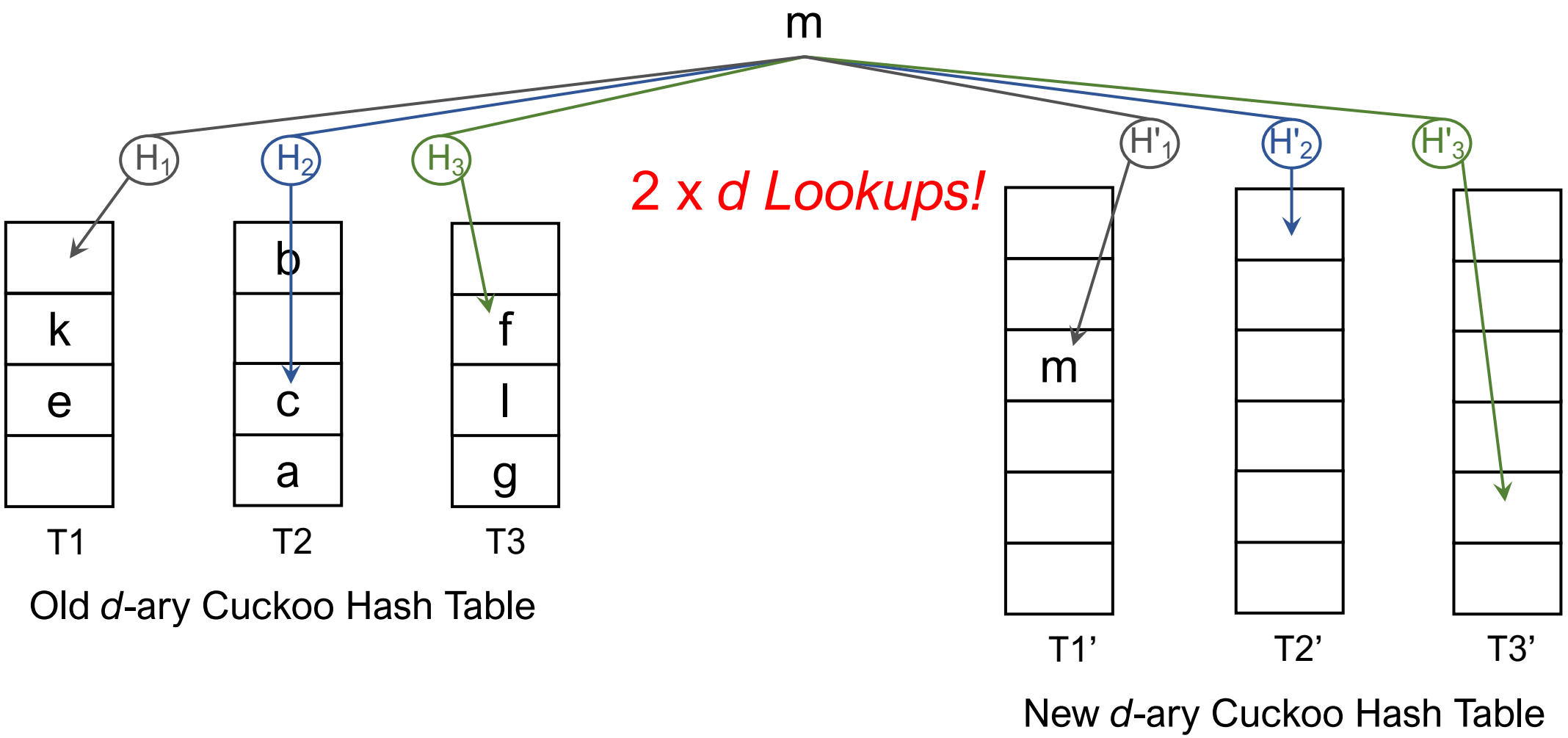


New *d*-ary Cuckoo Hash Table

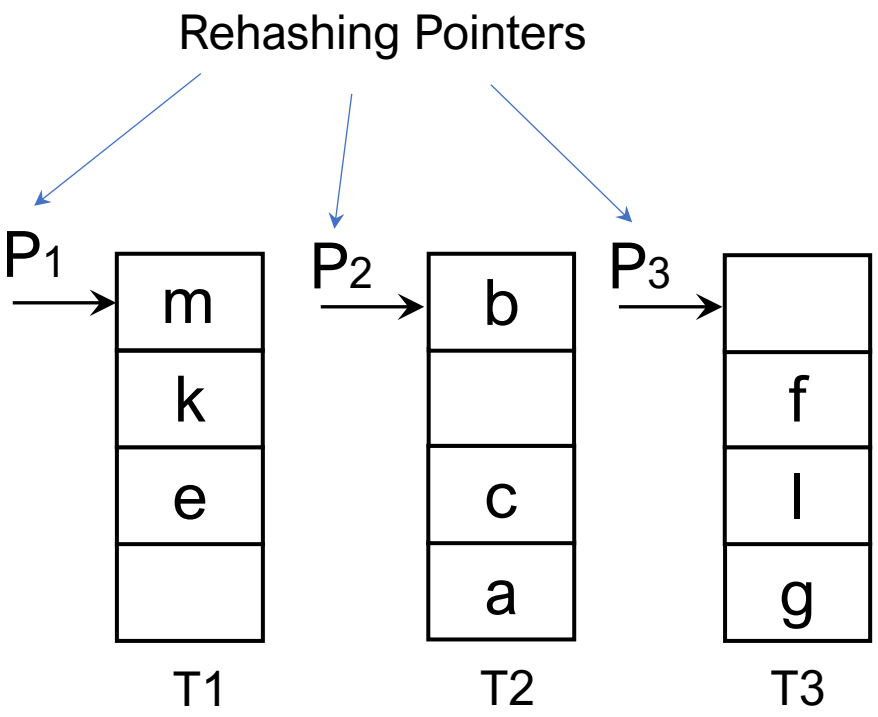
Lookup During Gradual Resizing



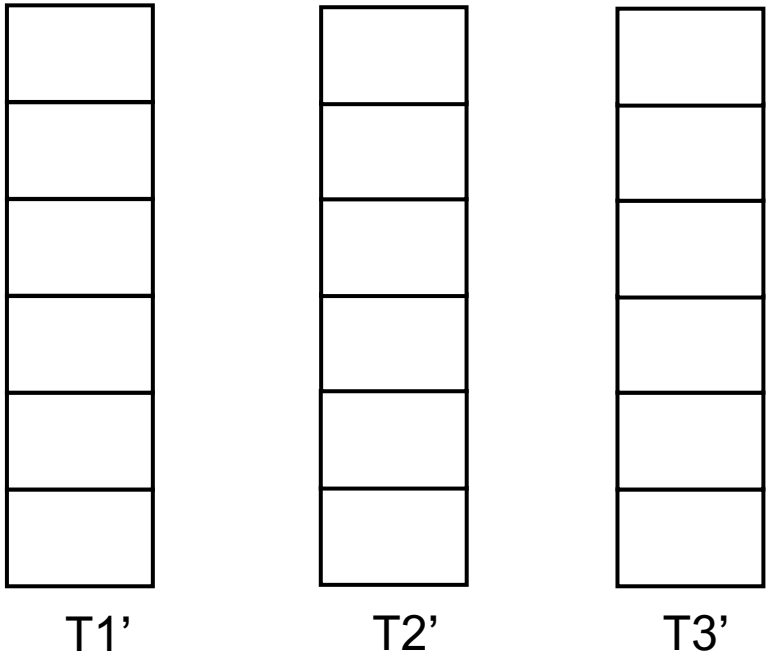
Problem of Resizing: Double #Lookups



Contribution: Elastic Cuckoo Hashing

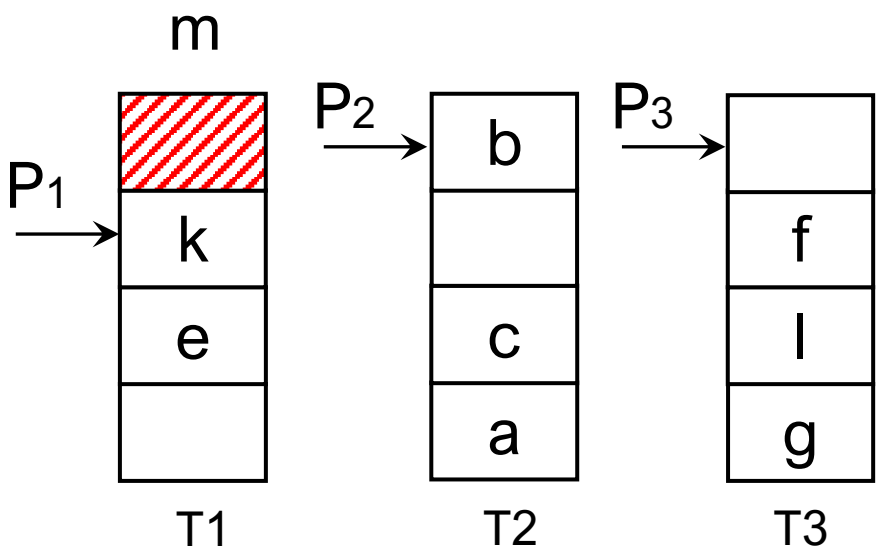


Old *d*-ary Cuckoo Hash Table

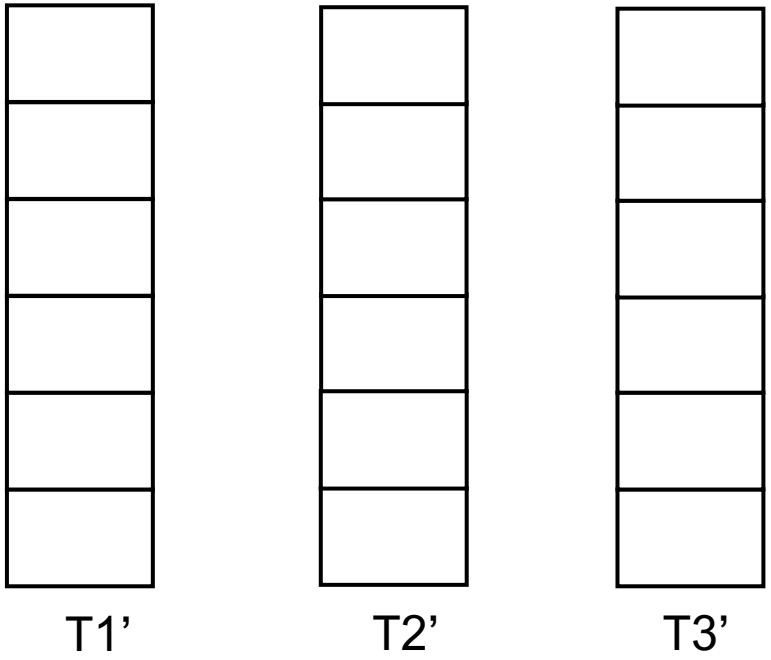


New *d*-ary Cuckoo Hash Table

Elastic Cuckoo Migration



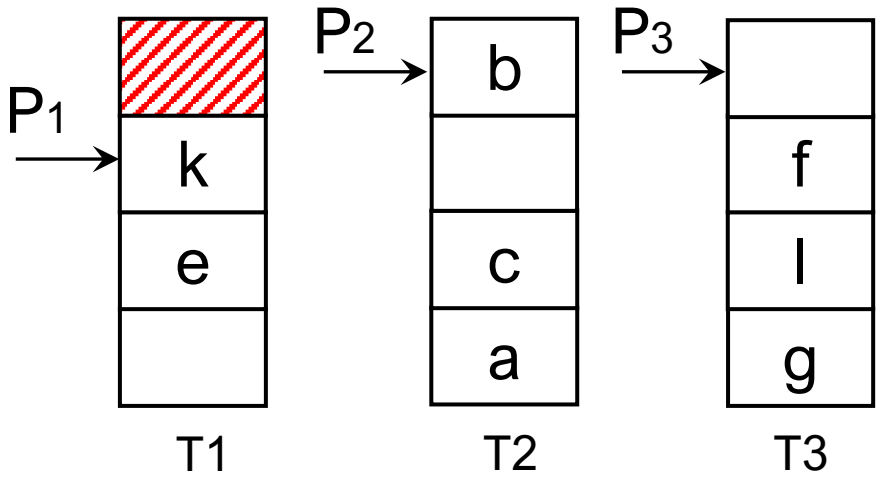
Old *d*-ary Cuckoo Hash Table



New *d*-ary Cuckoo Hash Table

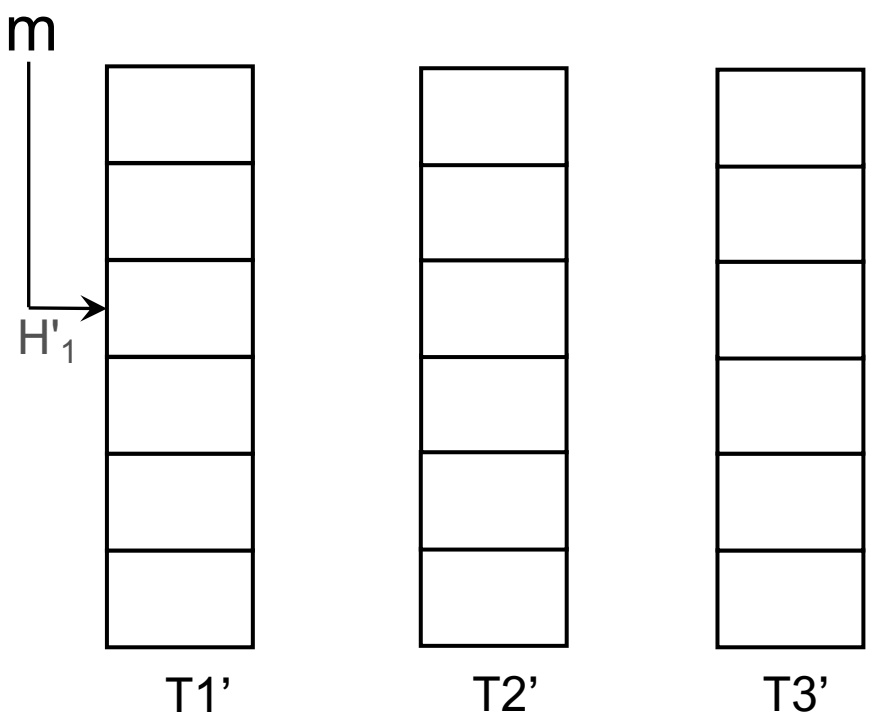
Migrated Region

Elastic Cuckoo Migration



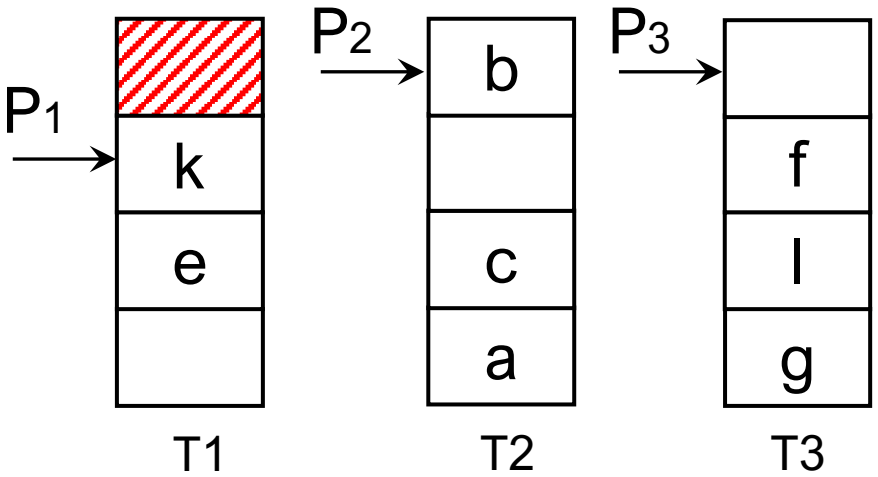
Old *d*-ary Cuckoo Hash Table

Migrated Region

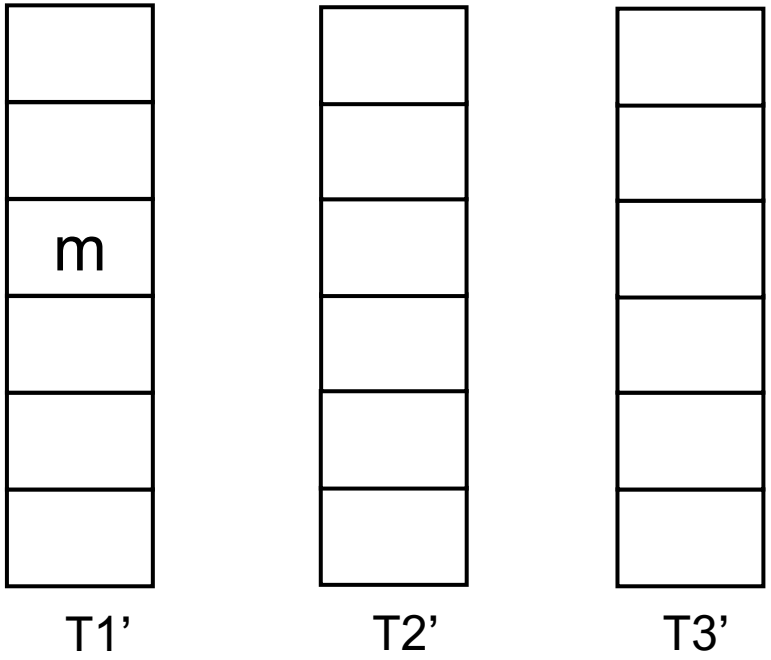


New *d*-ary Cuckoo Hash Table

Elastic Cuckoo Migration



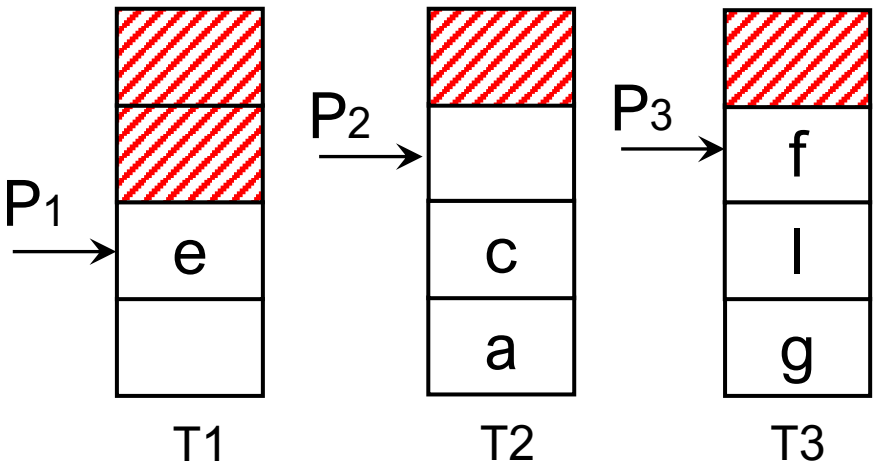
Old *d*-ary Cuckoo Hash Table



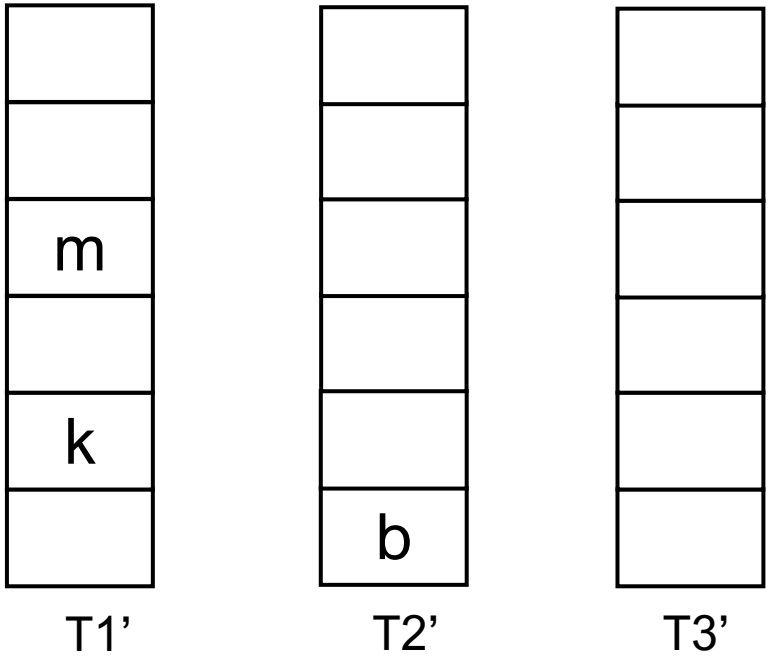
New *d*-ary Cuckoo Hash Table

Migrated Region

Elastic Cuckoo Migration



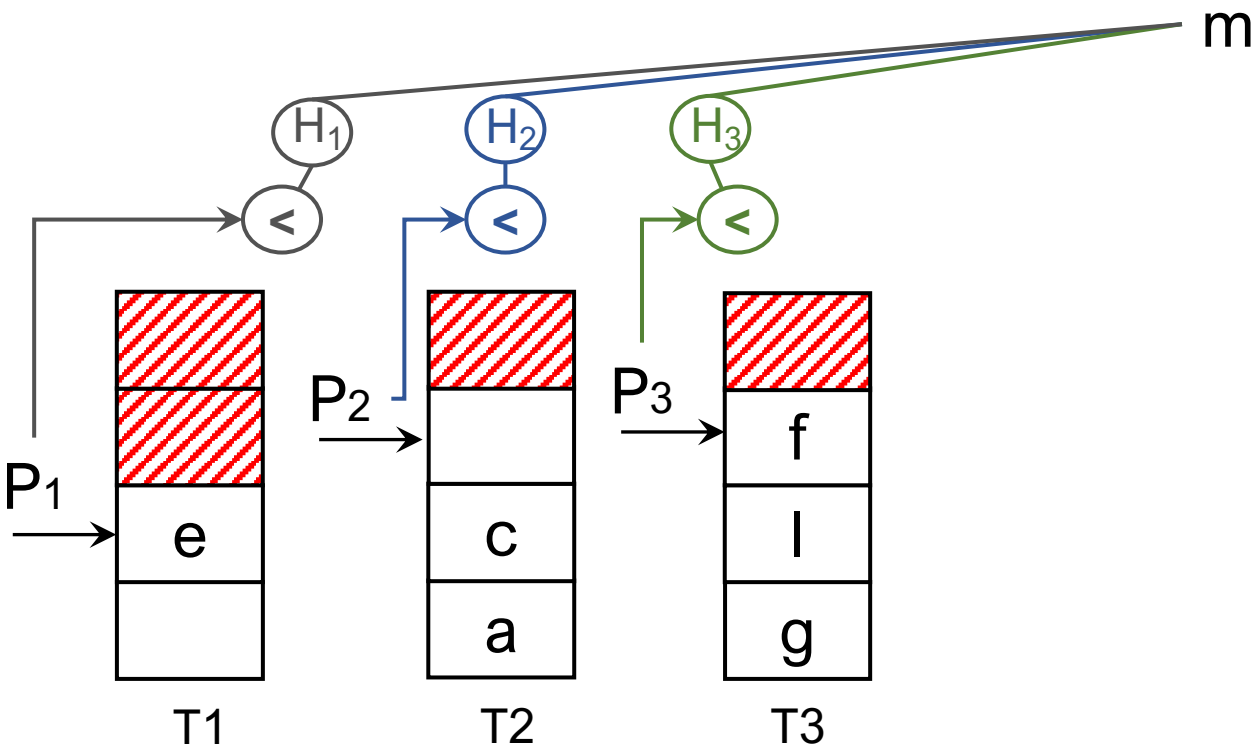
Old *d*-ary Cuckoo Hash Table



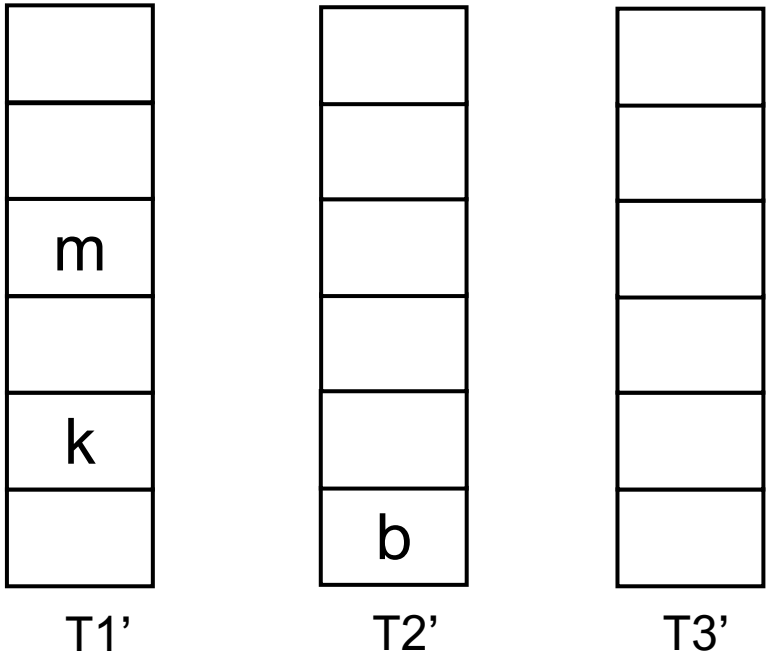
New *d*-ary Cuckoo Hash Table

Migrated Region

Elastic Cuckoo Lookup



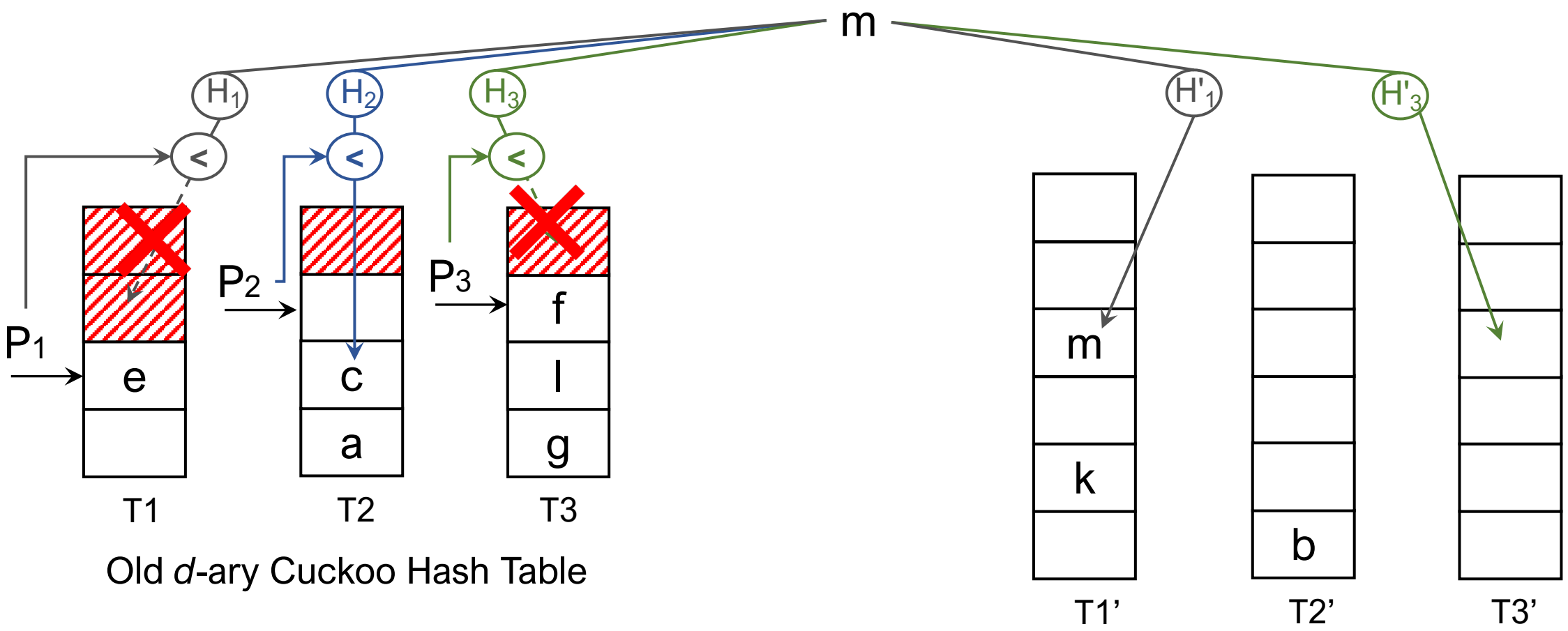
Old *d*-ary Cuckoo Hash Table



New *d*-ary Cuckoo Hash Table

Migrated Region

Elastic Cuckoo Lookup

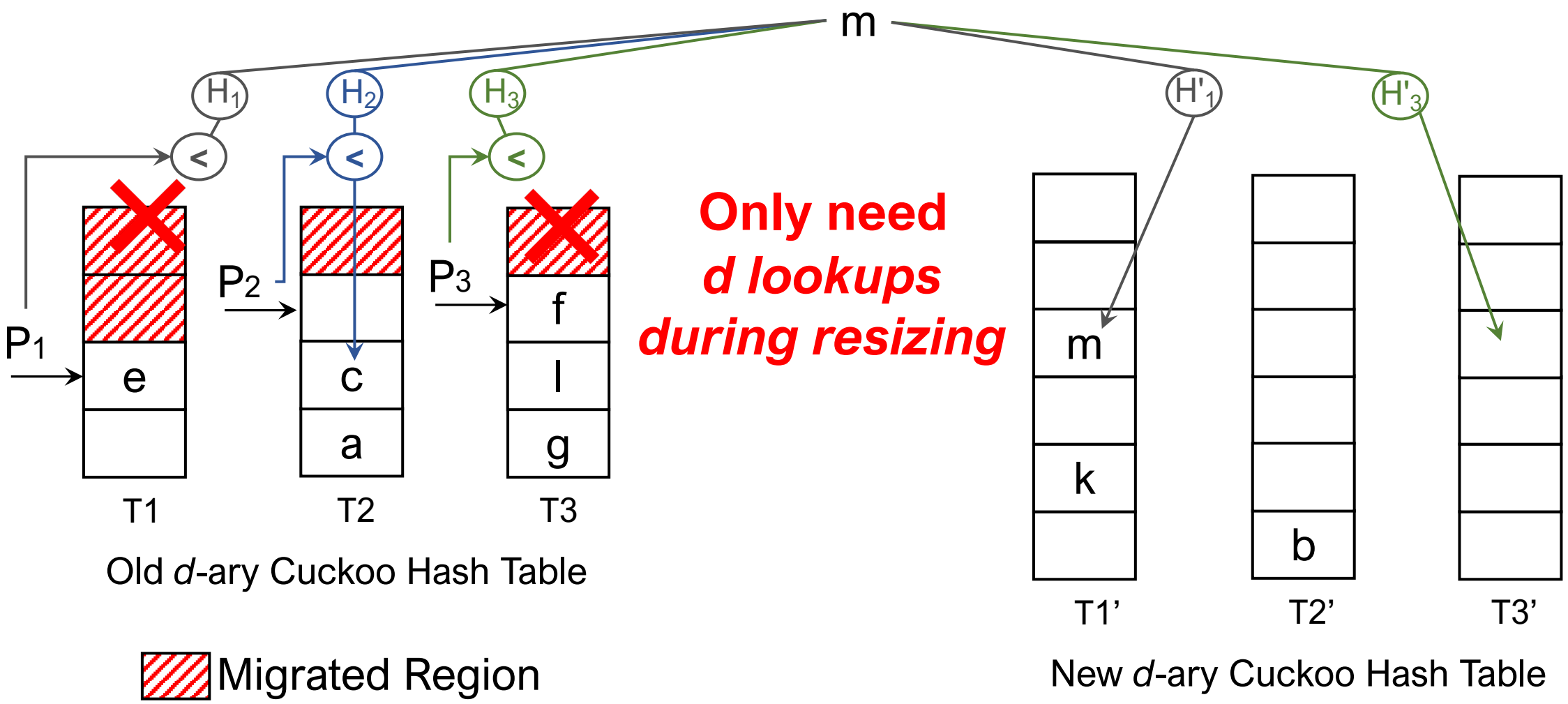


Old *d*-ary Cuckoo Hash Table

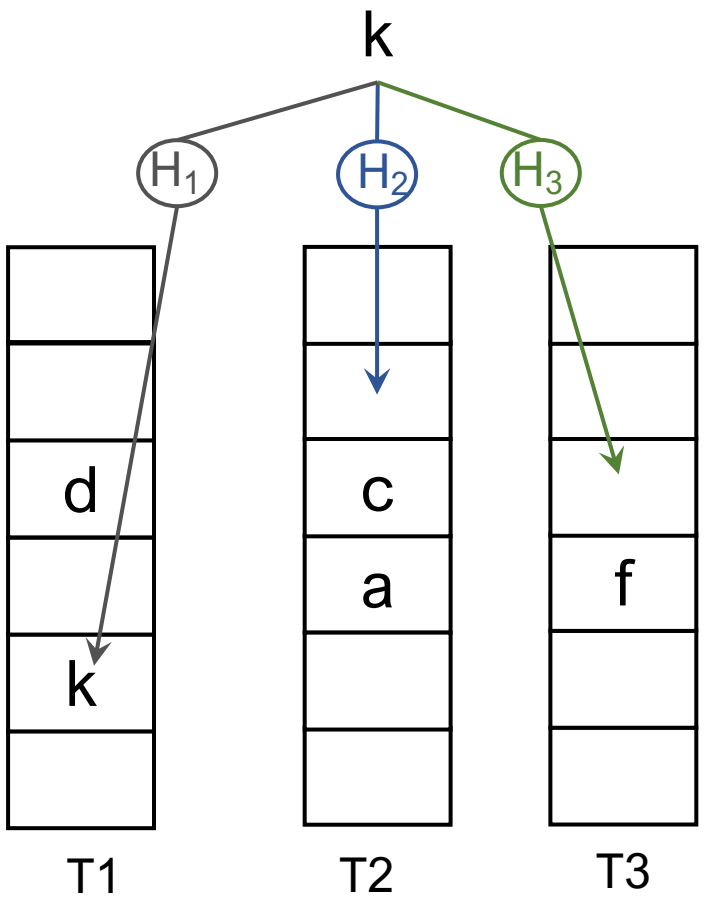
New *d*-ary Cuckoo Hash Table

Migrated Region

Elastic Cuckoo Lookup

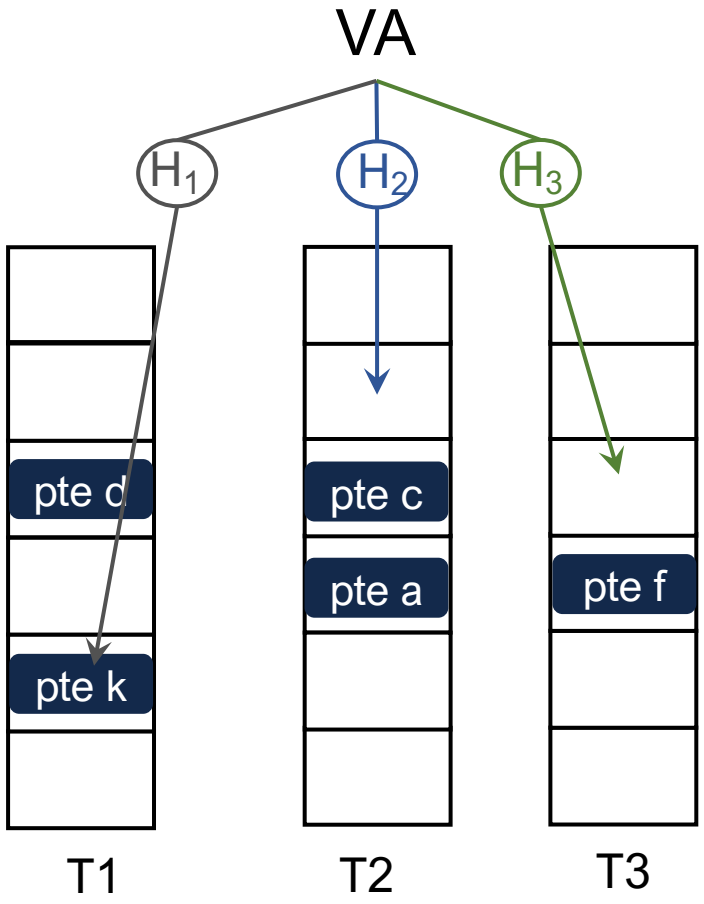


Exploiting Parallelism in Virtual Translation



d-ary Elastic Cuckoo Hash Table

Exploiting Parallelism in Virtual Translation



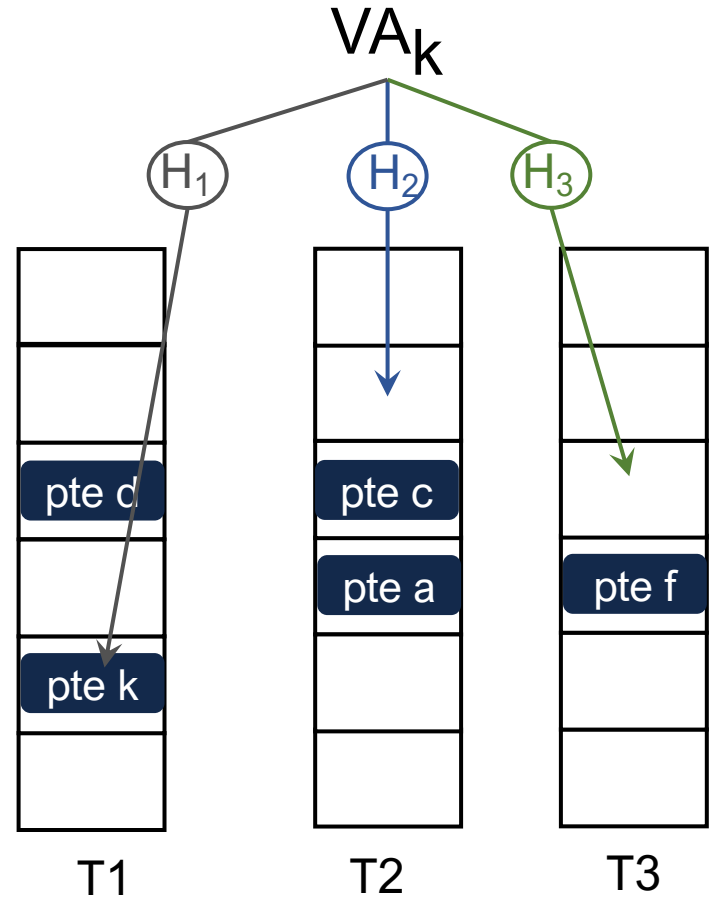
d-ary Elastic Cuckoo Page Table

Exploiting Parallelism in Virtual Translation

No sequential page walk (unlike radix)

At most *d* accesses *always*

Leverages multiple issue out-of-order processors



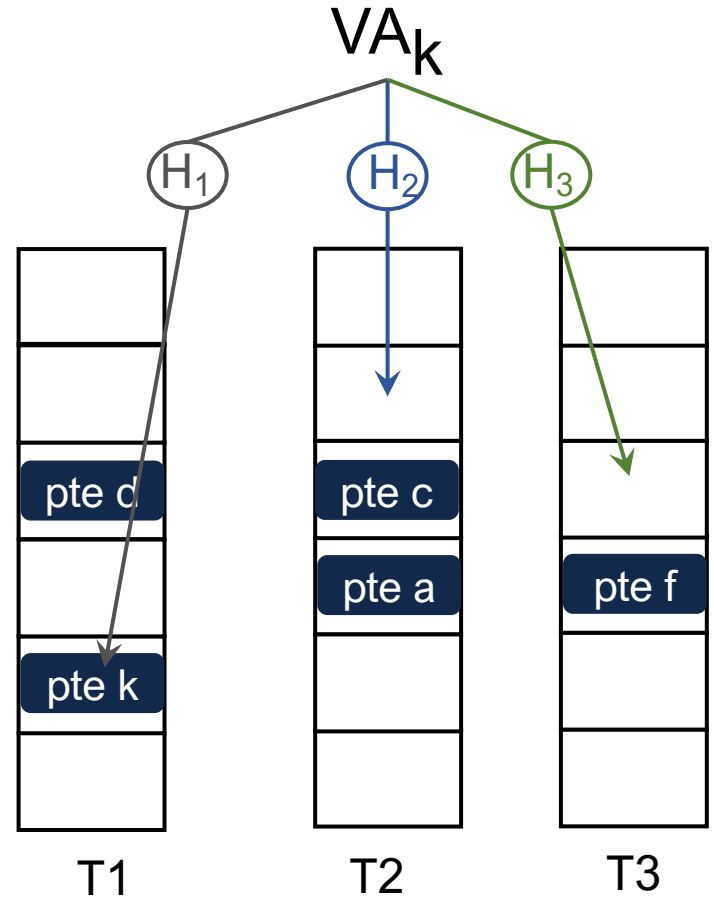
d-ary Elastic Cuckoo Page Table

Exploiting Parallelism in Virtual Translation

No sequential page walk (unlike radix)

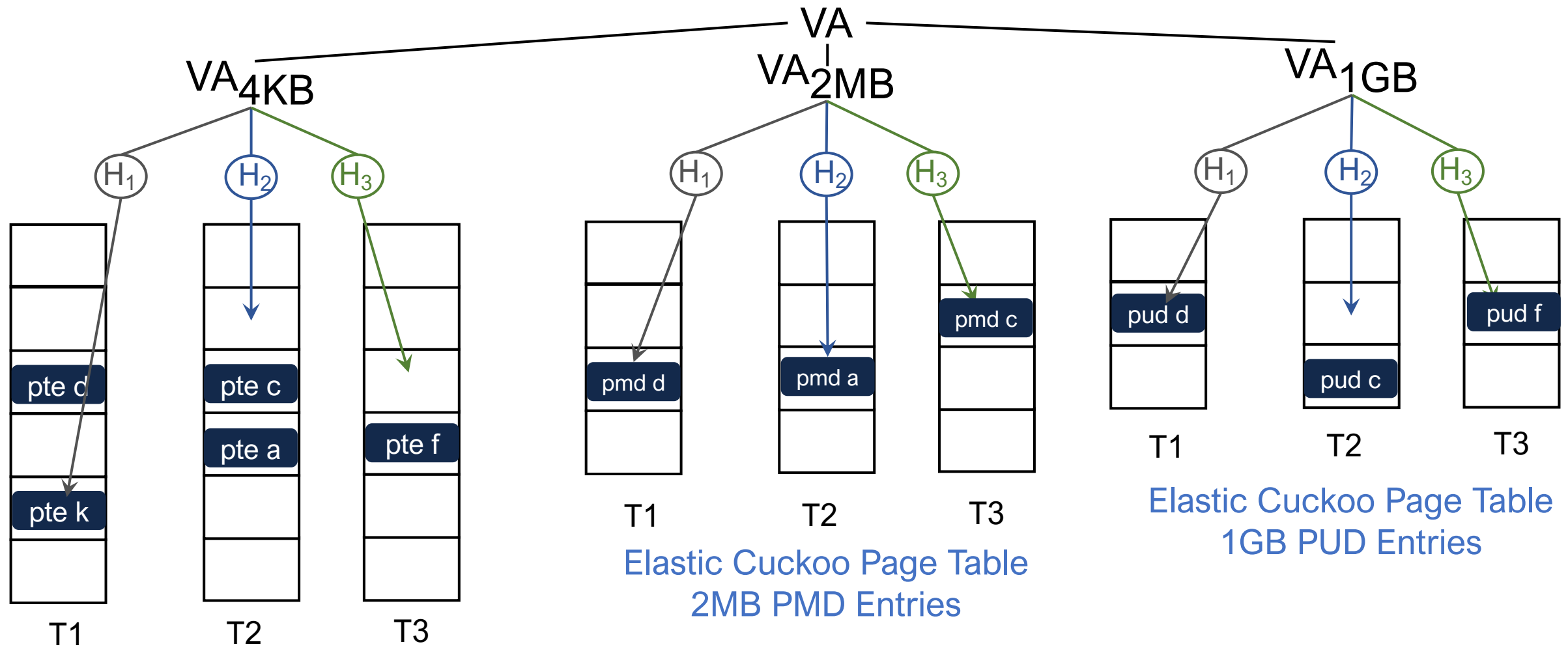
At most *d* accesses *always*

Leverages multiple issue out-of-order processors



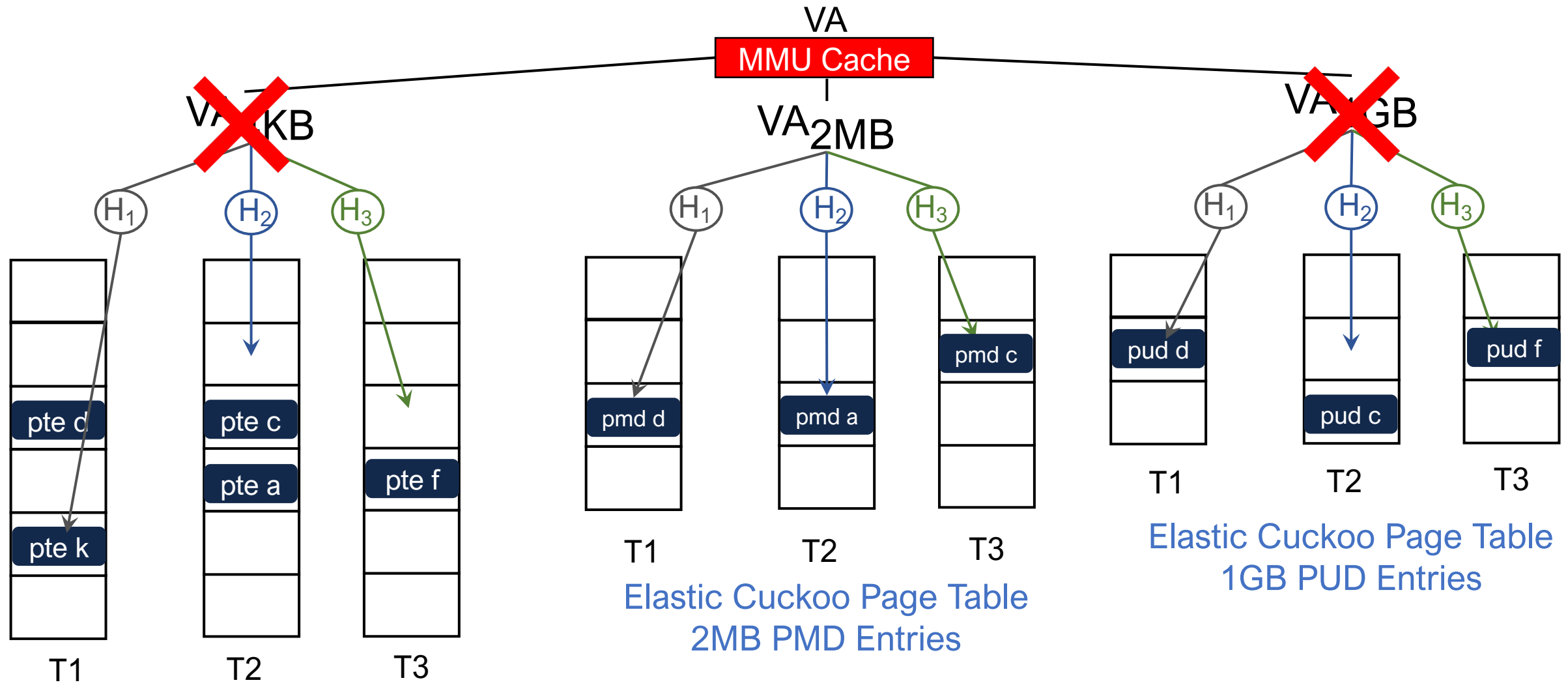
d-ary Elastic Cuckoo Page Table

Lookup Multiple Page Sizes in Parallel



d-ary Elastic Cuckoo Page Table
4KB PTE Entries

New MMU Cache to Prune Parallelism

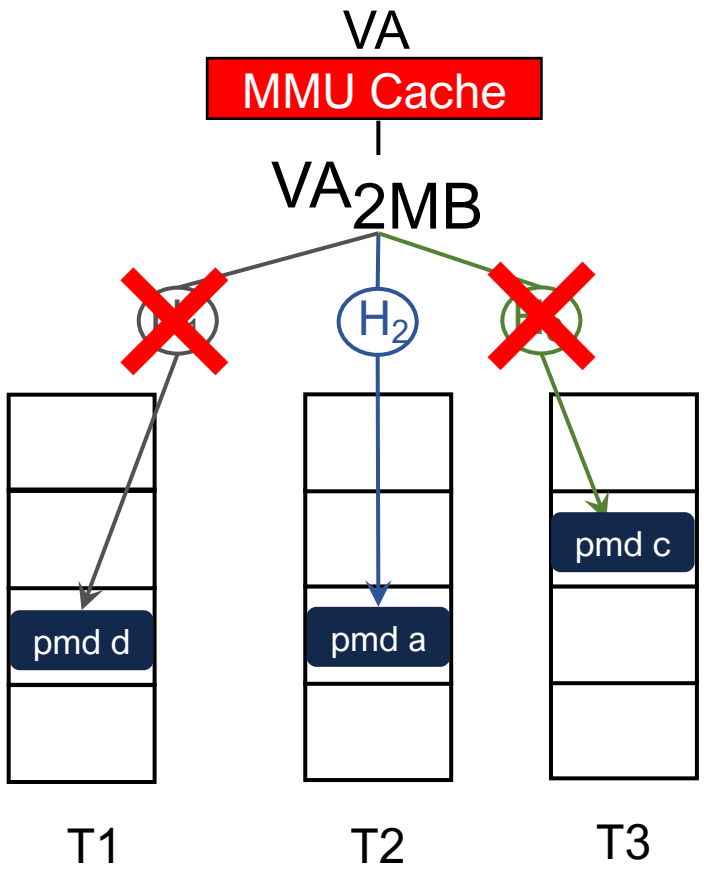


d-ary Elastic Cuckoo Page Table
4KB PTE Entries

Elastic Cuckoo Page Table
2MB PMD Entries

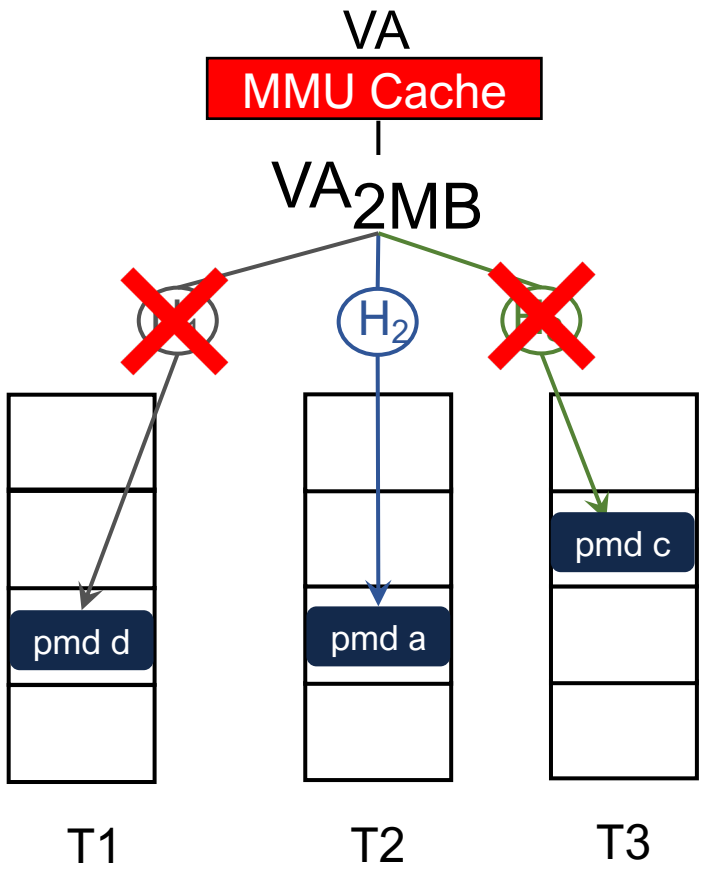
Elastic Cuckoo Page Table
1GB PUD Entries

New MMU Cache to Prune Parallelism



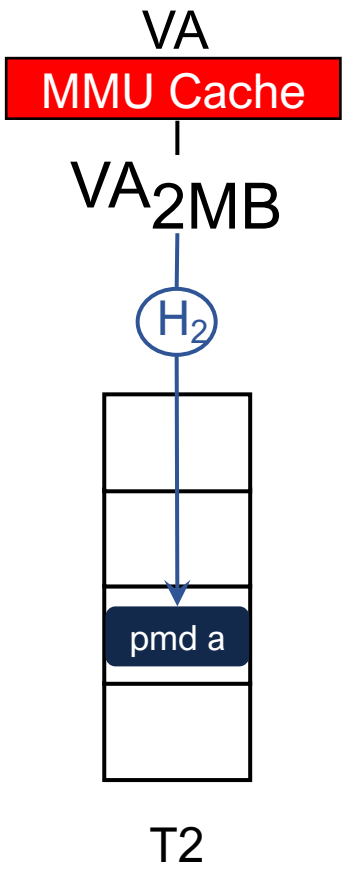
Elastic Cuckoo Page Table
2MB PMD Entries

New MMU Cache to Prune Parallelism



Elastic Cuckoo Page Table
2MB PMD Entries

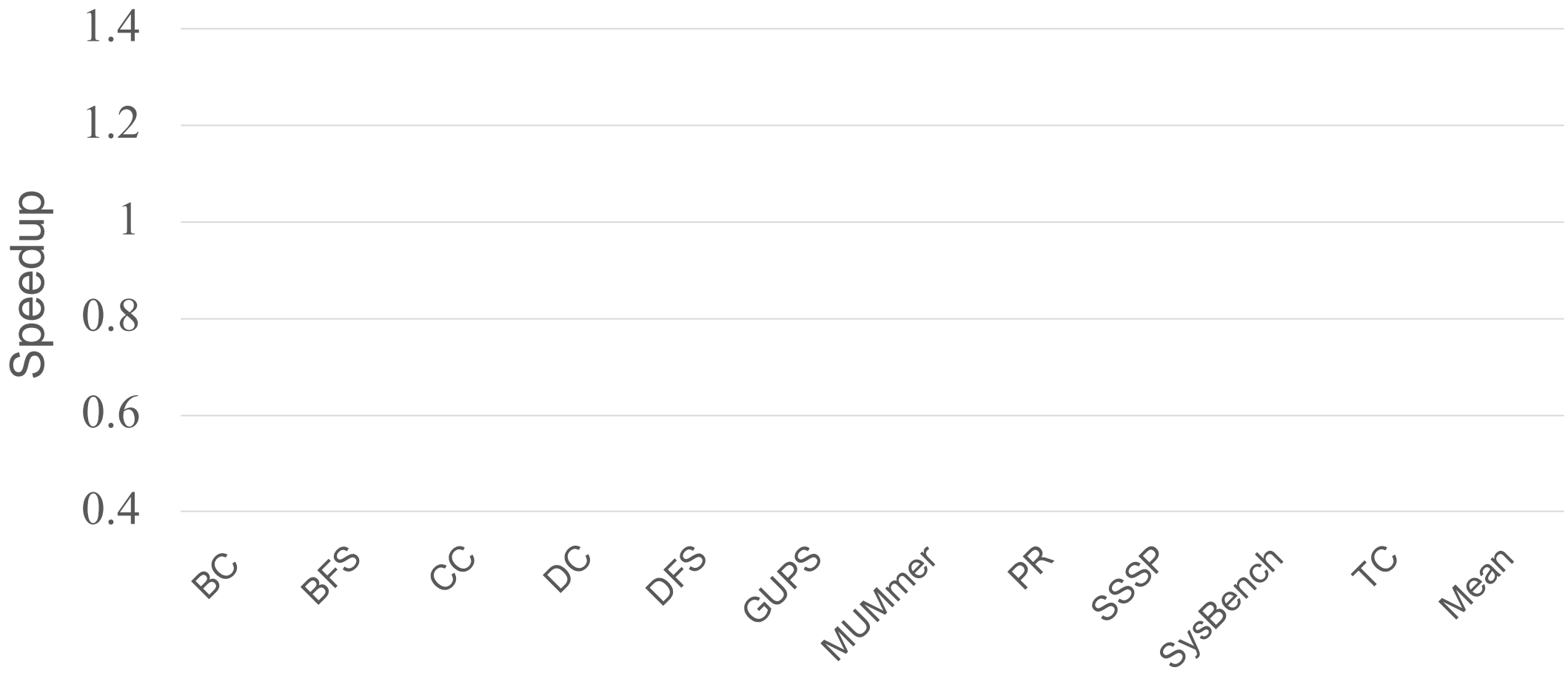
New MMU Cache to Prune Parallelism



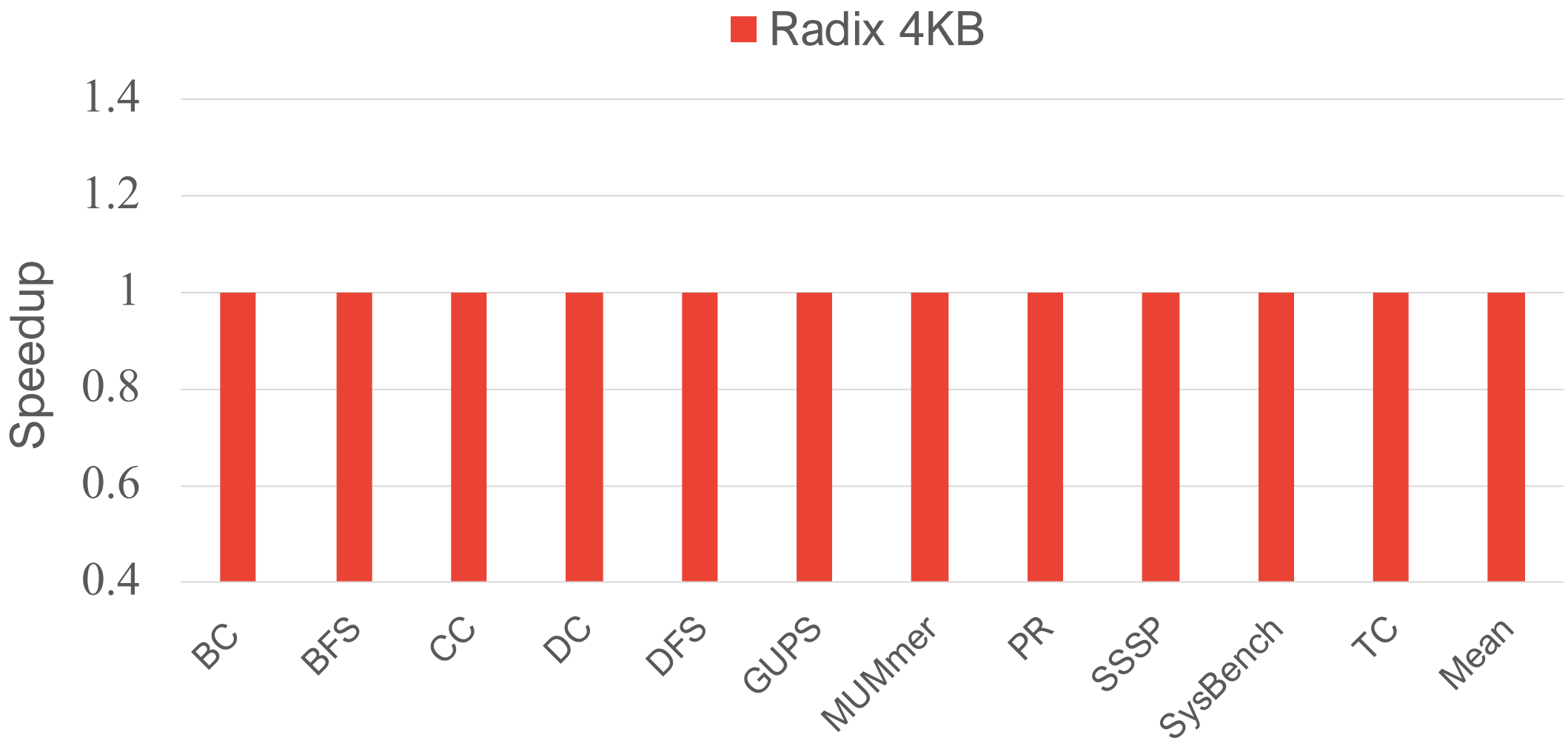
Elastic Cuckoo Page Table
2MB PMD Entries

Evaluation

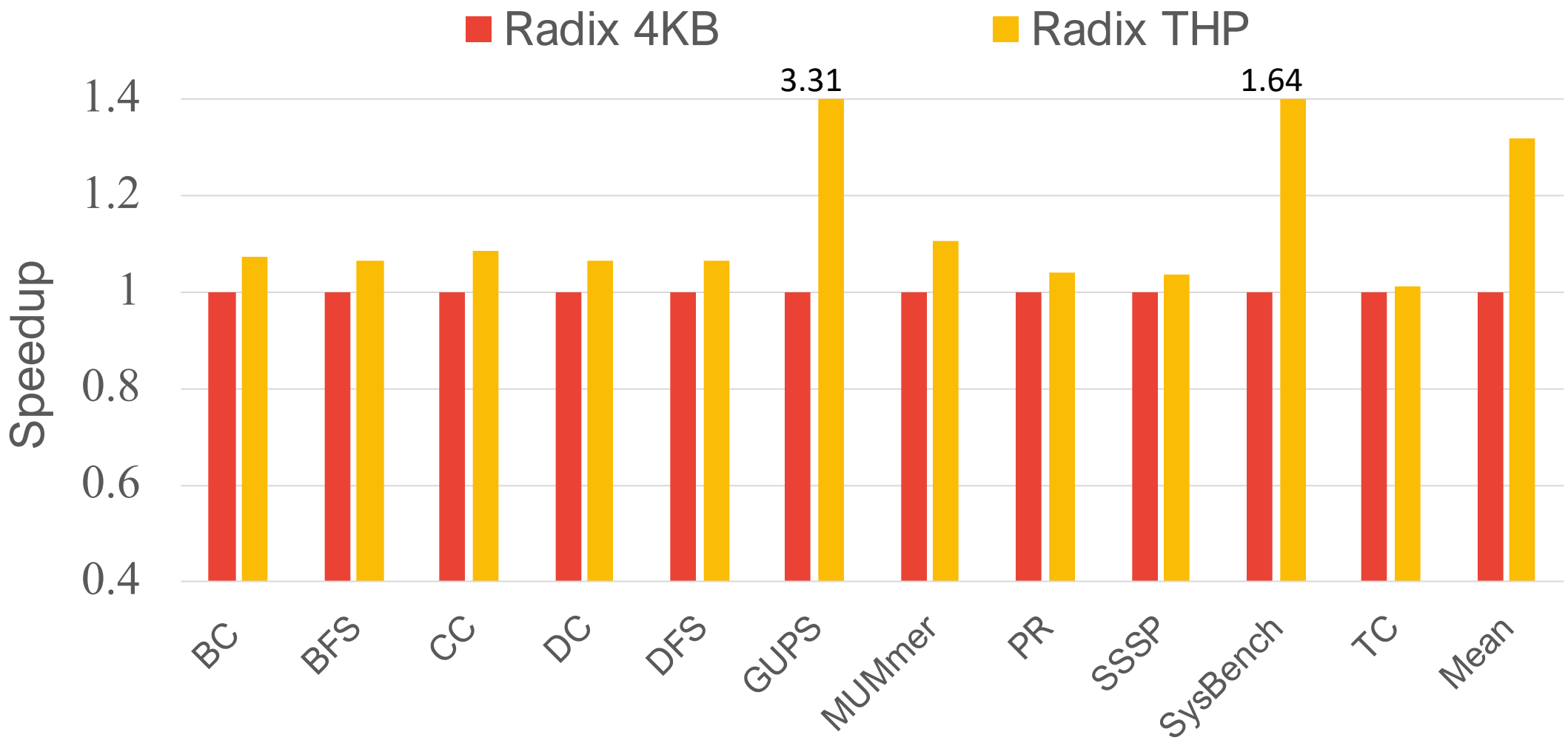
Application Speedup



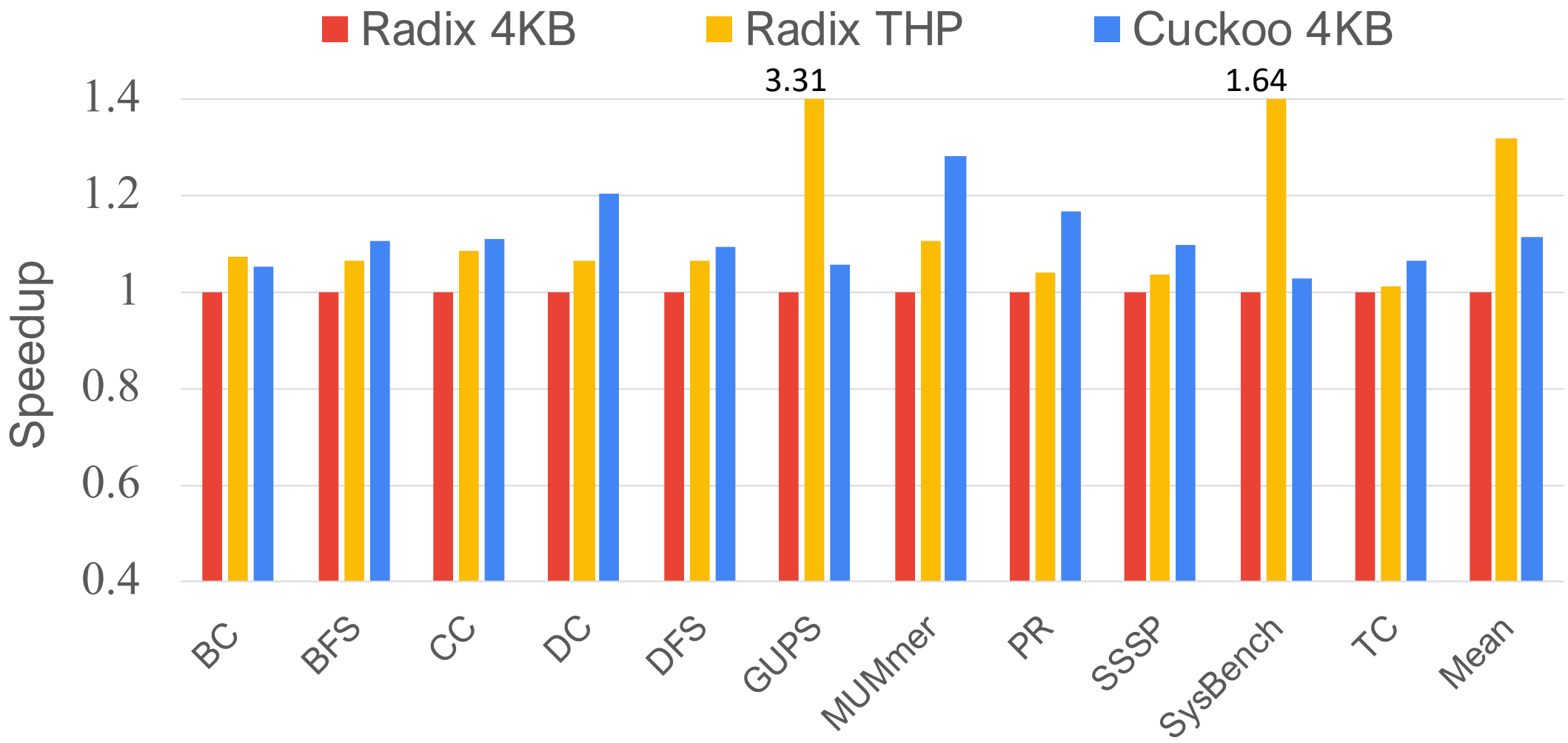
Application Speedup



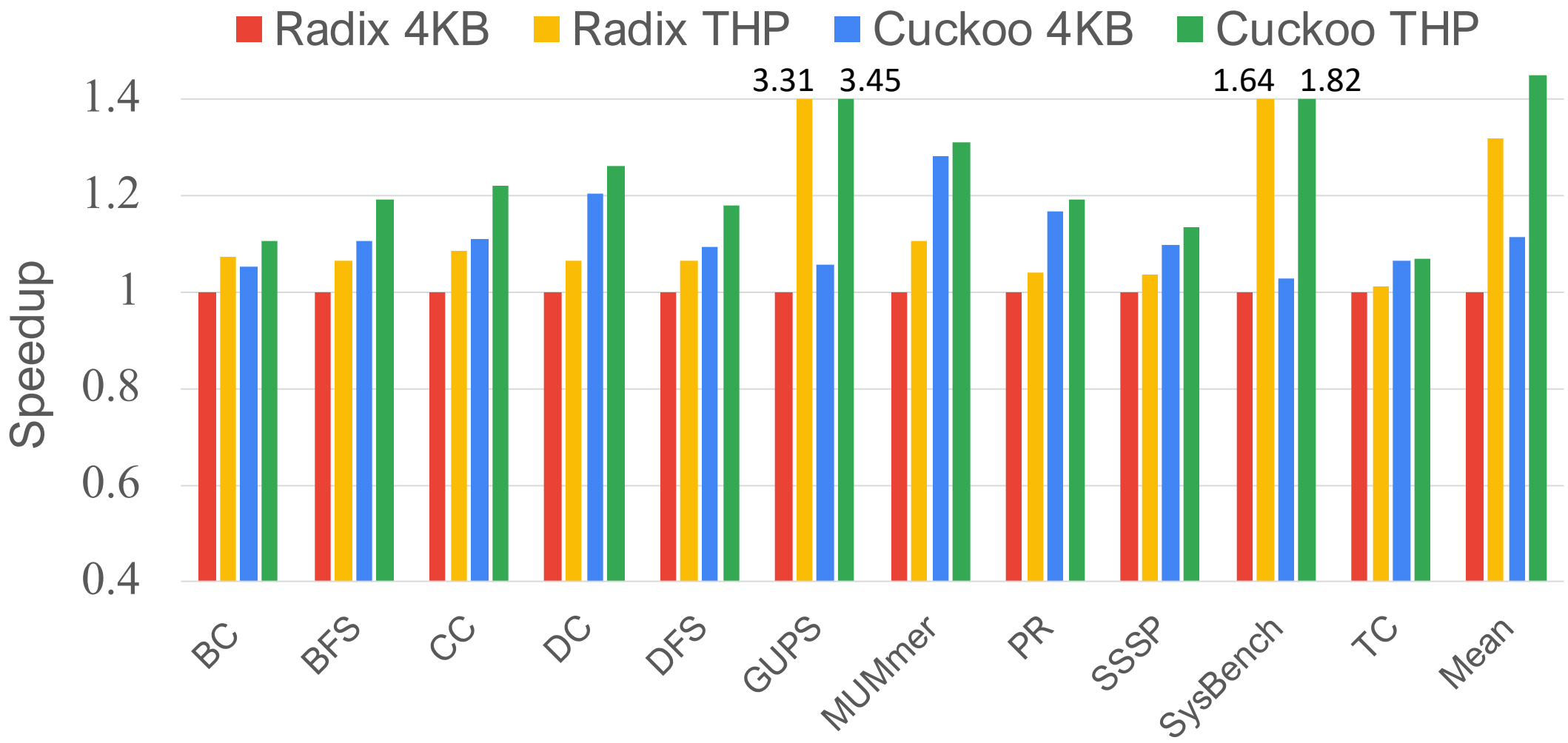
Application Speedup



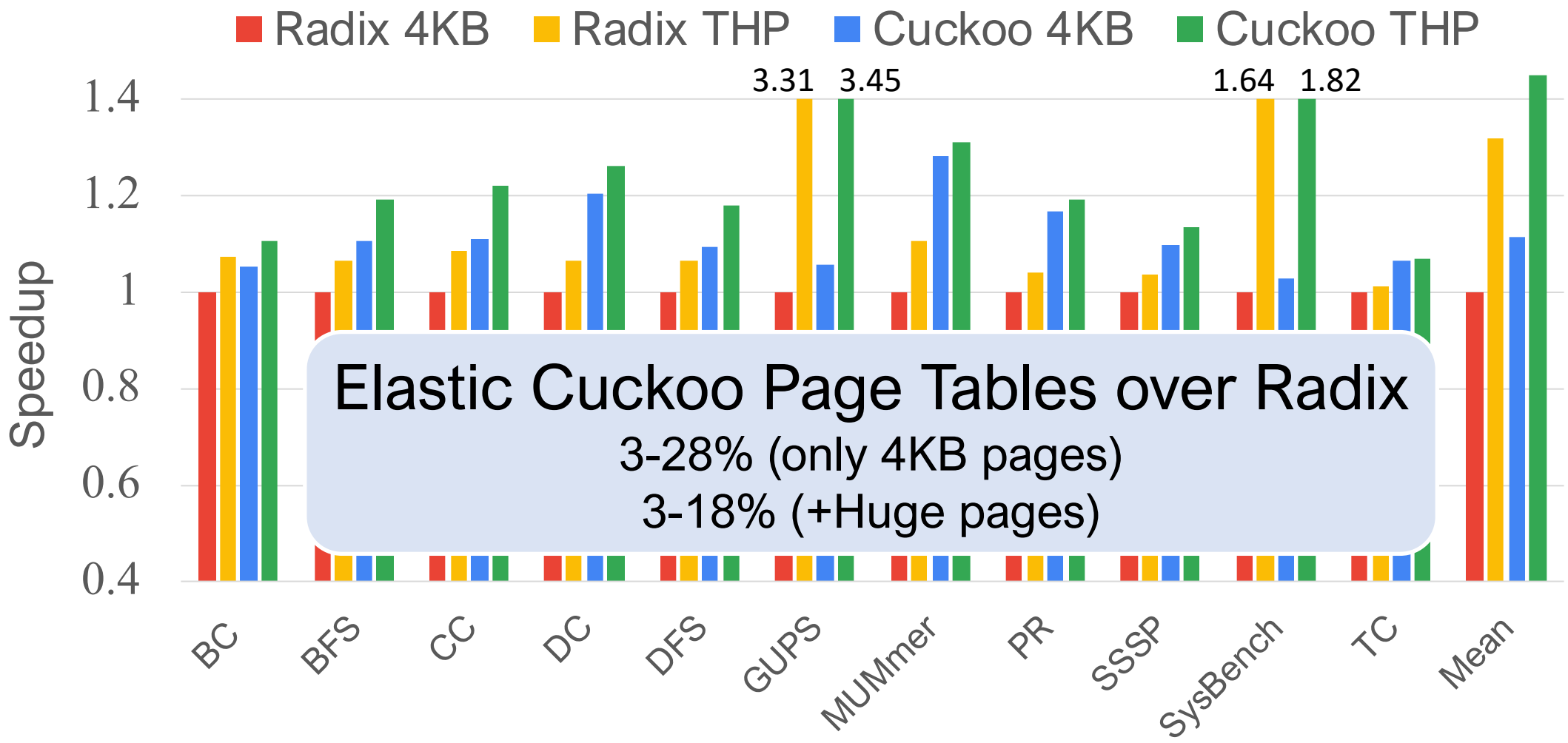
Application Speedup



Application Speedup



Application Speedup

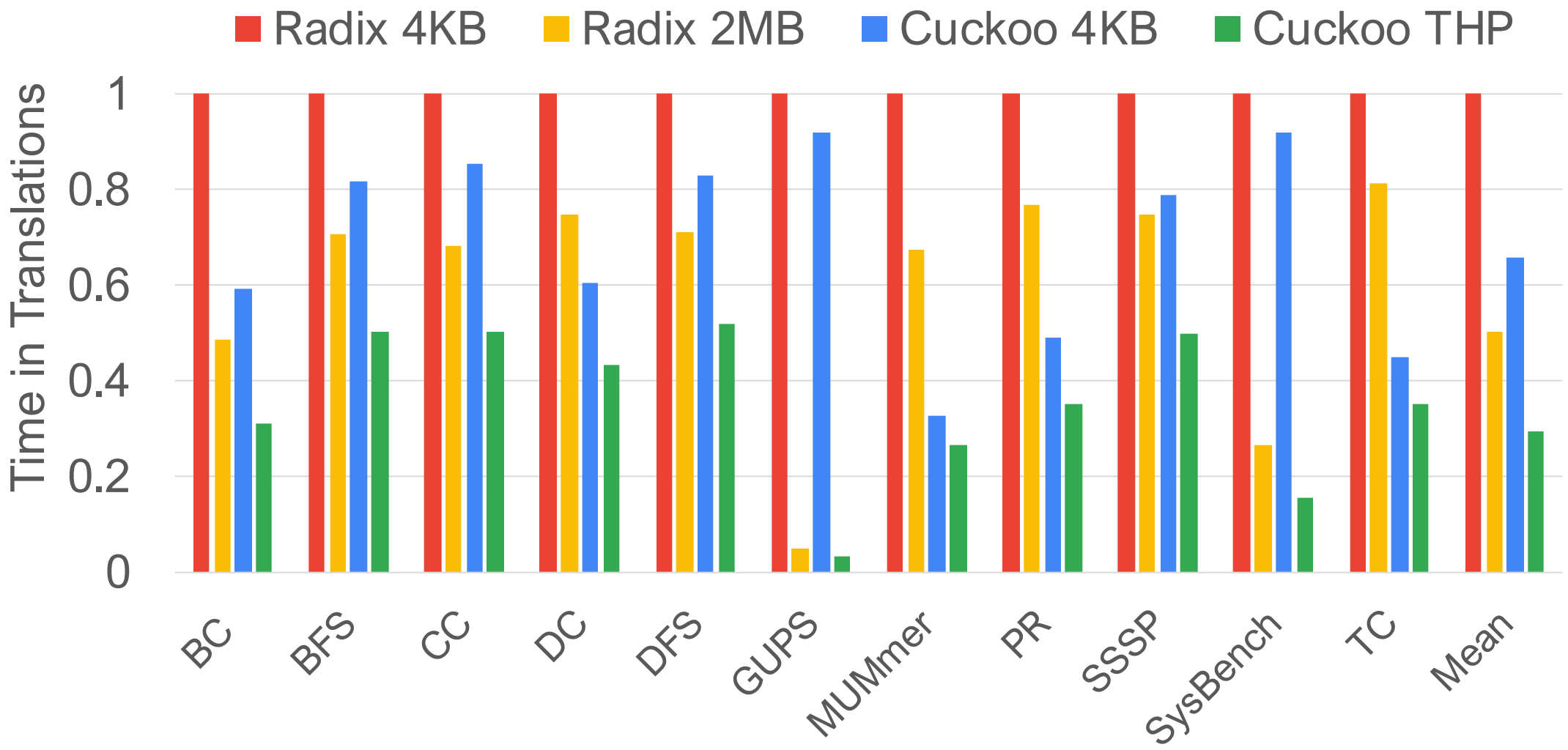


Elastic Cuckoo Page Tables over Radix

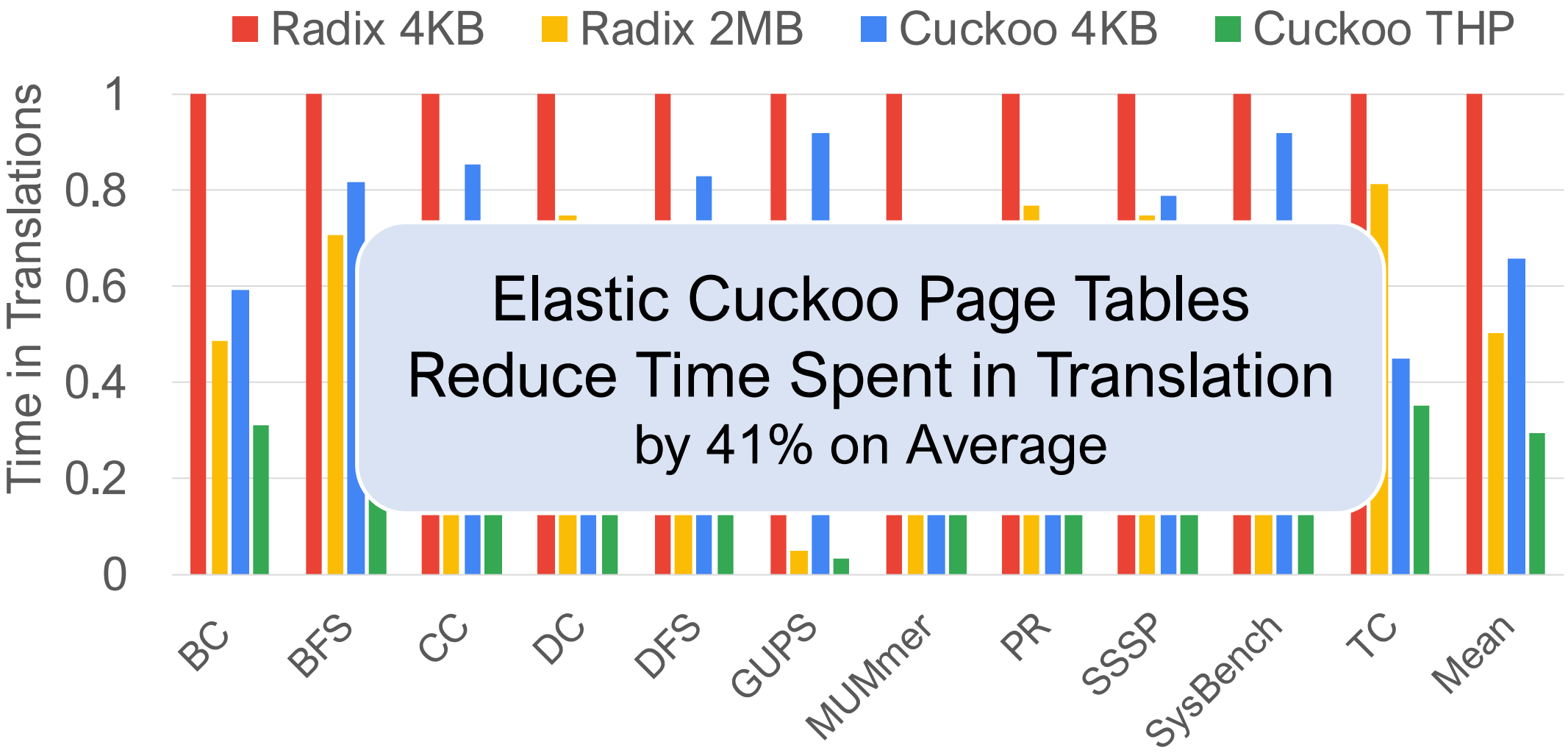
3-28% (only 4KB pages)

3-18% (+Huge pages)

Time Spent in Translations



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

ILLINOIS

i-acoma group