Rethinking Address Translation in the Age of Containers, Serverless Computing, and Non-Volatile Memory

Josep Torrellas, Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu

Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

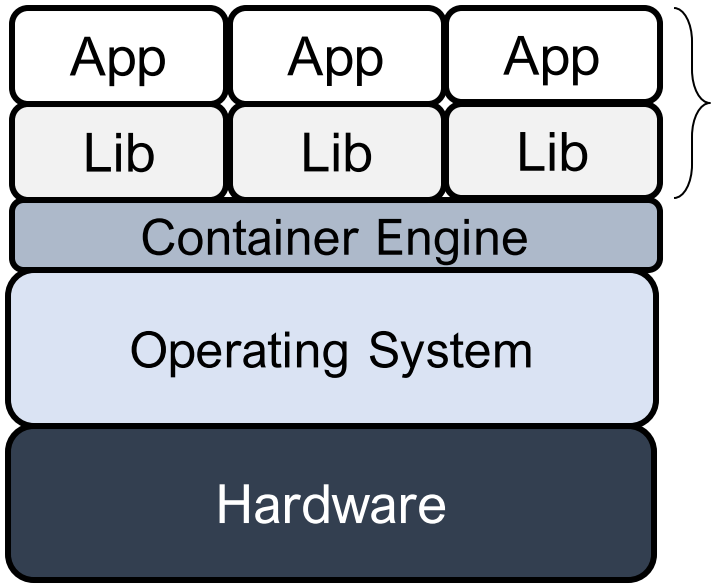
ARM Research Summit
September 2020

i-acoma group

ILLINOIS

New Era in Cloud Computing: Containers

1. Lightweight
2. Faster bring-up
3. Higher consolidation

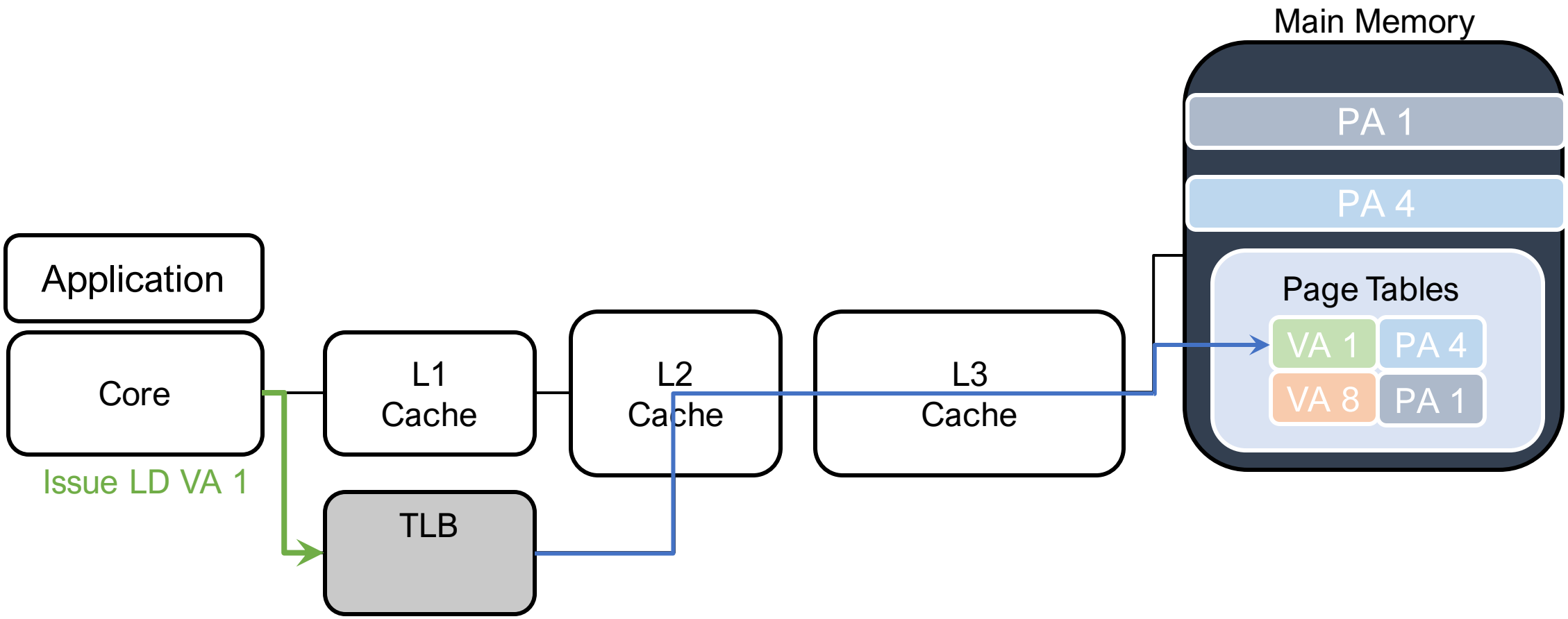


Containers

High number of containers

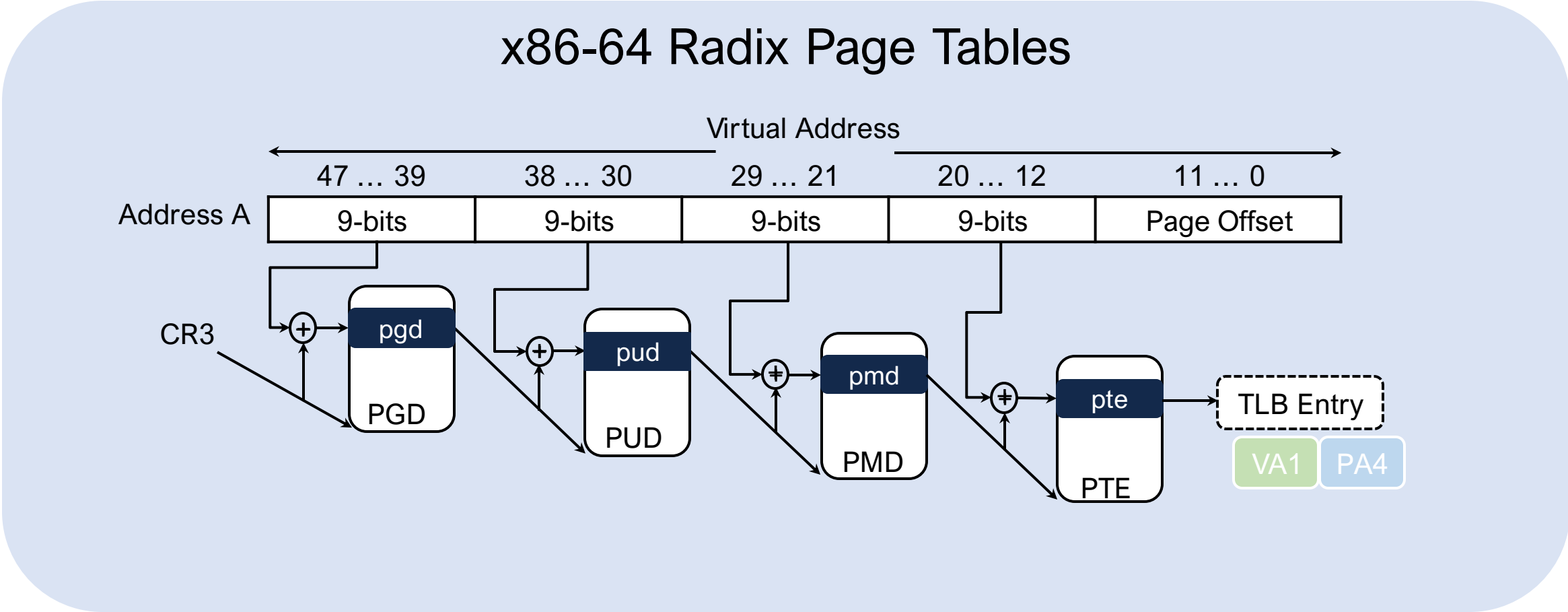
Very frequent context switches

Address Translation System

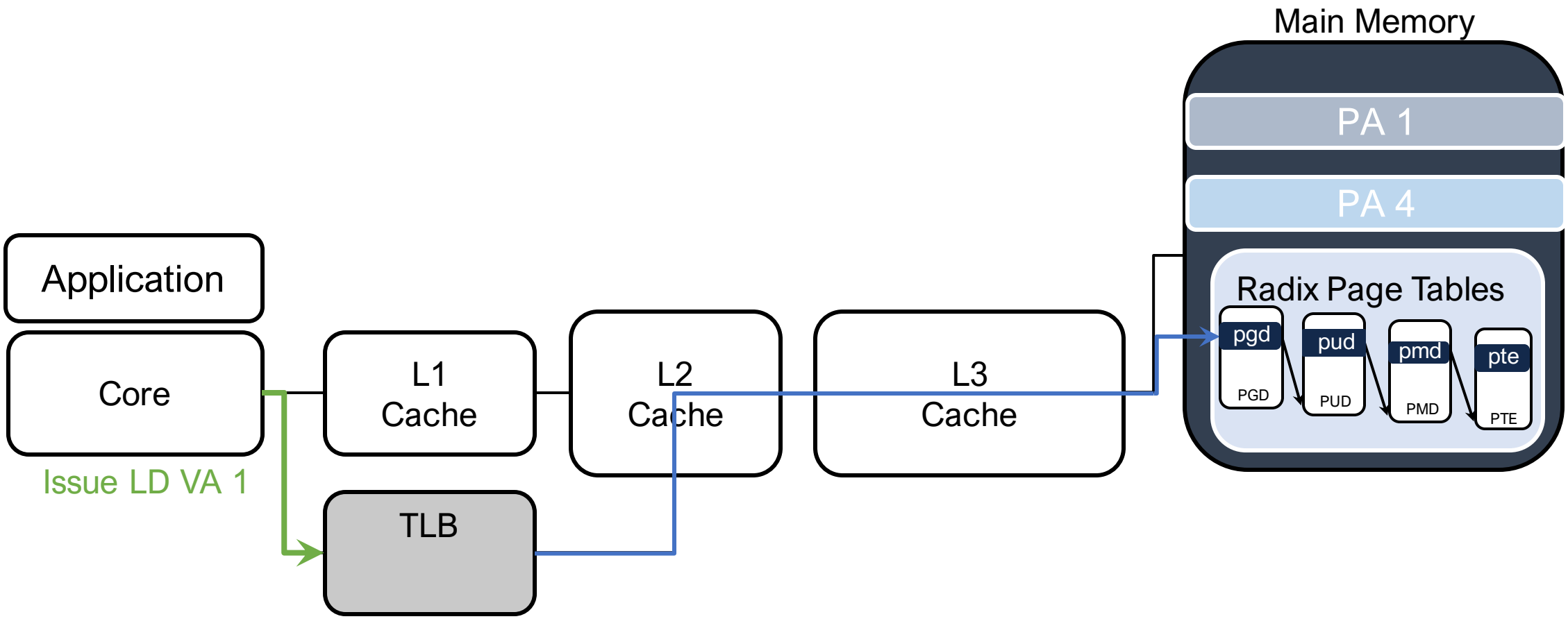


TLB Miss → "Page Walk" = Fetch entry from page table

Address Translation System

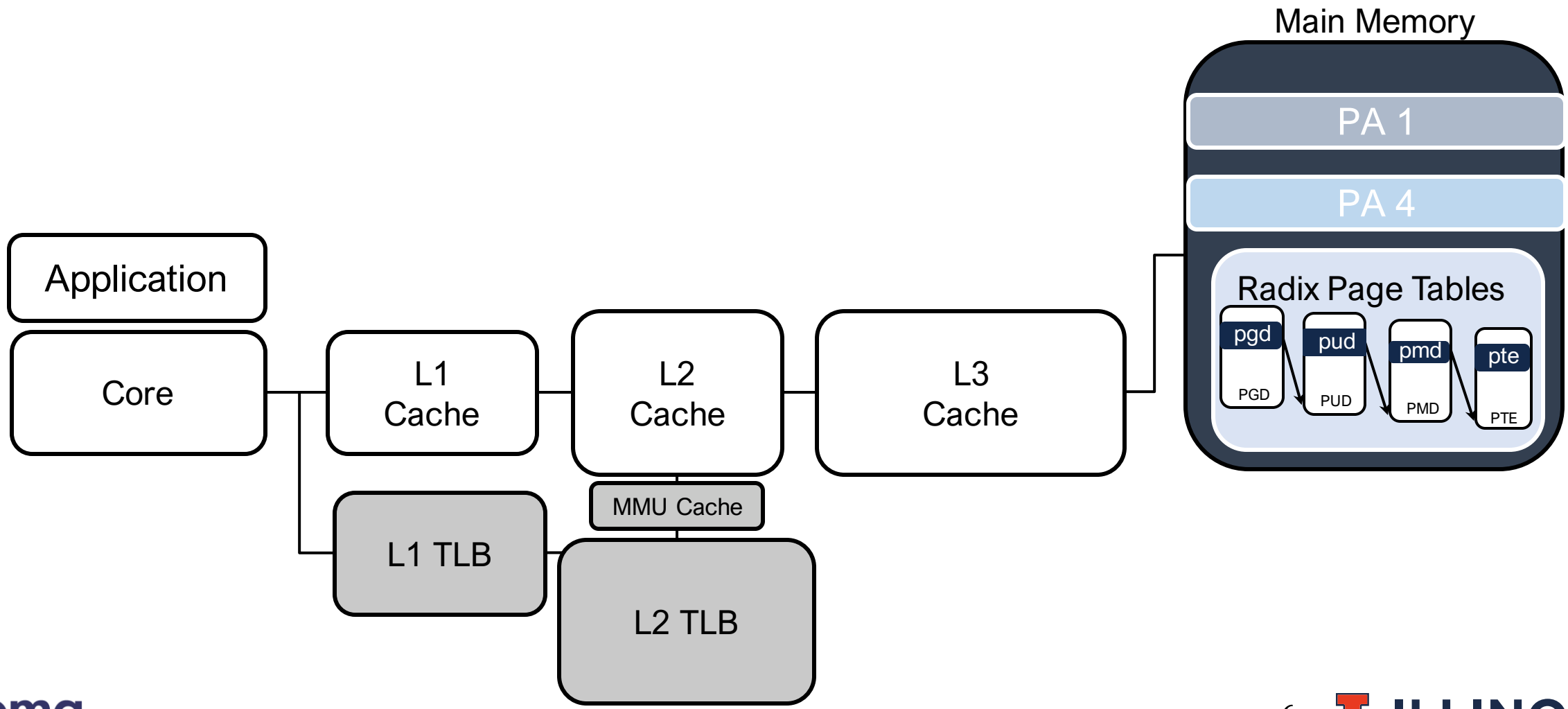


Address Translation System



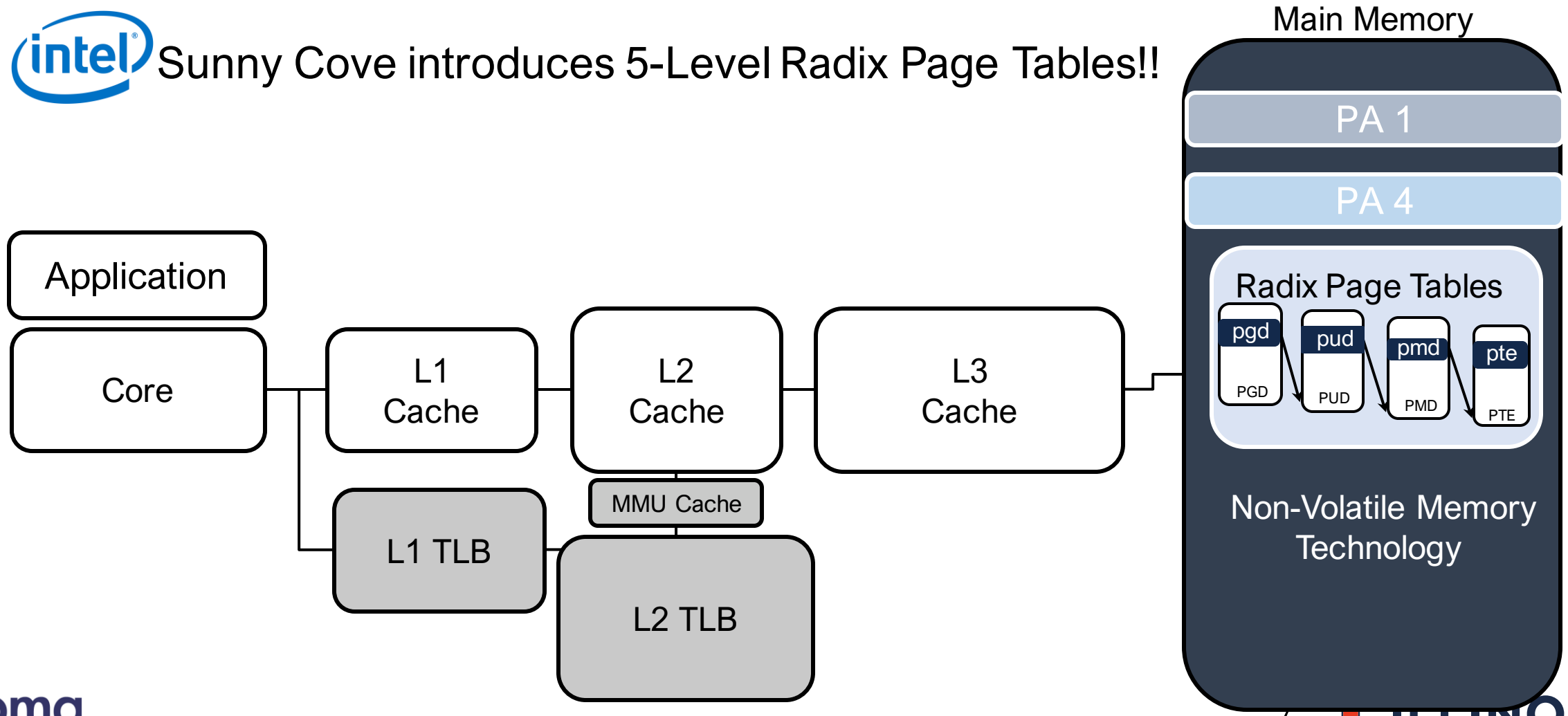
TLB Miss → "Page Walk" = Fetch entry from radix page table

Address Translation System



Even More Memory is Here!

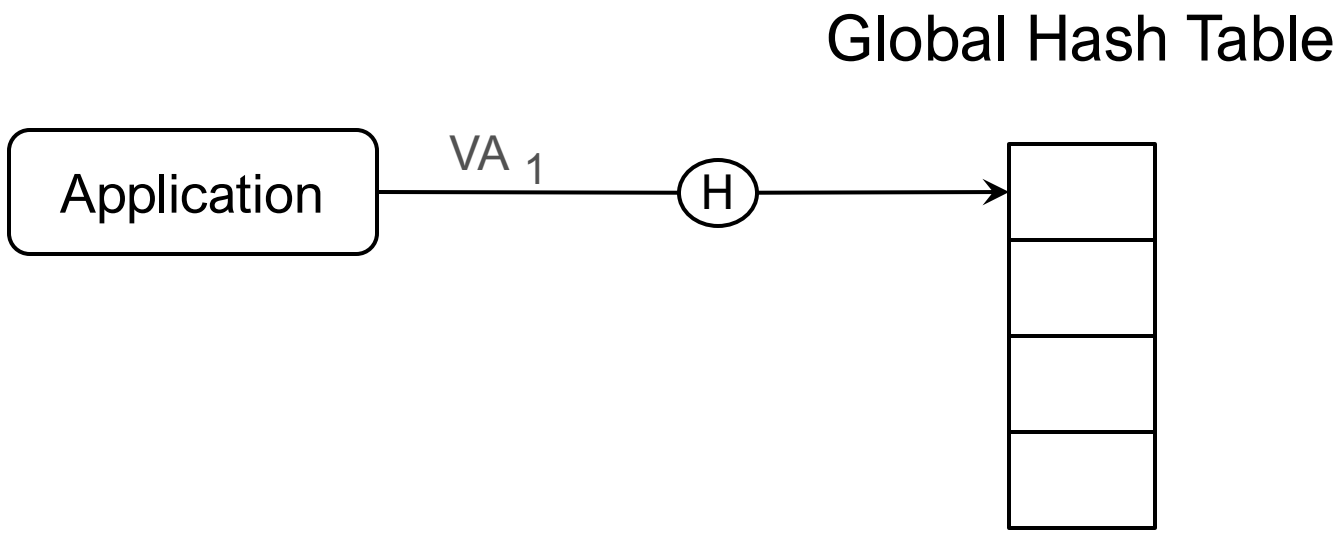
Sunny Cove introduces 5-Level Radix Page Tables!!



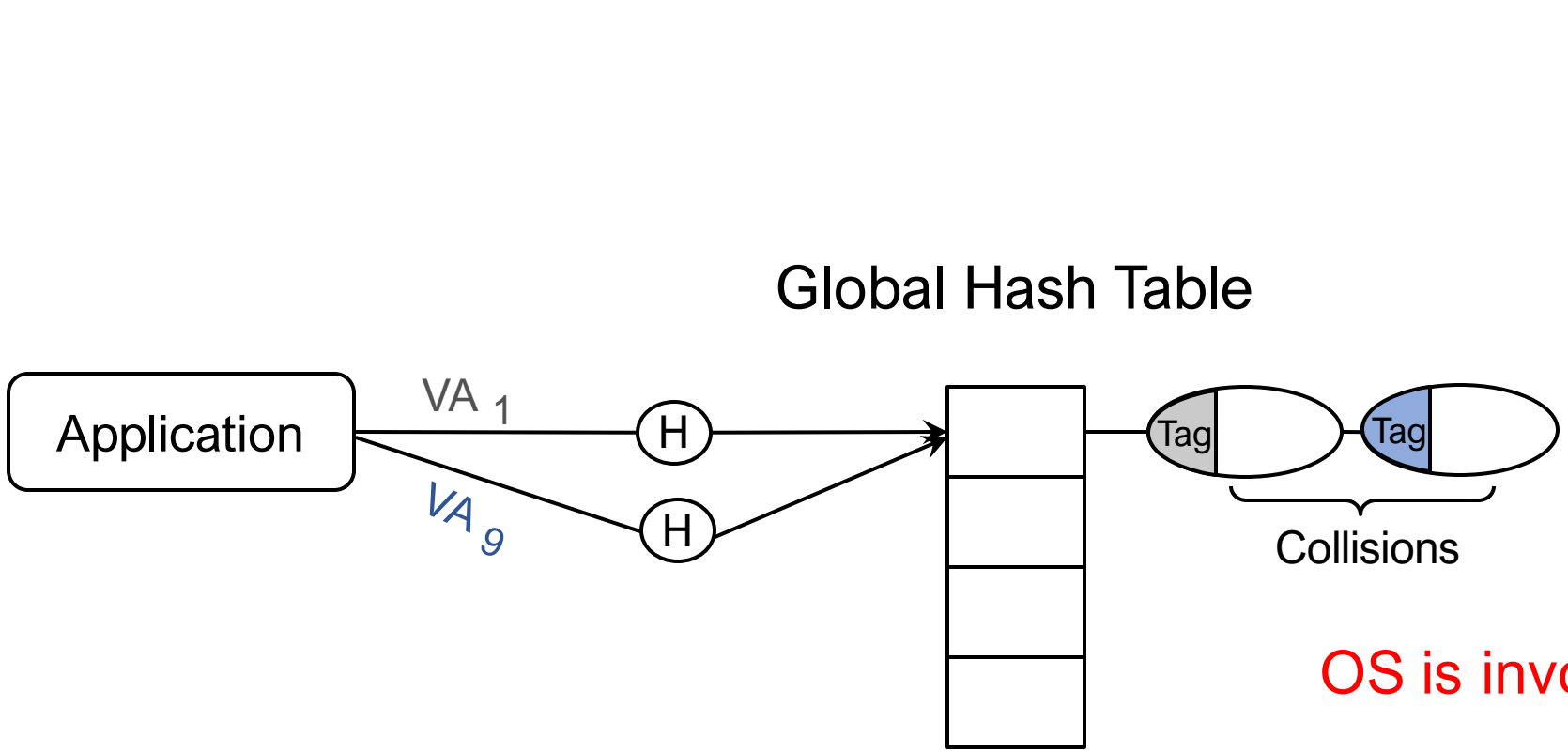
Time to Rethink Virtual Memory Translation

Current Radix Page Tables Are Not Scalable

Alternative: Global Hashed Page Table



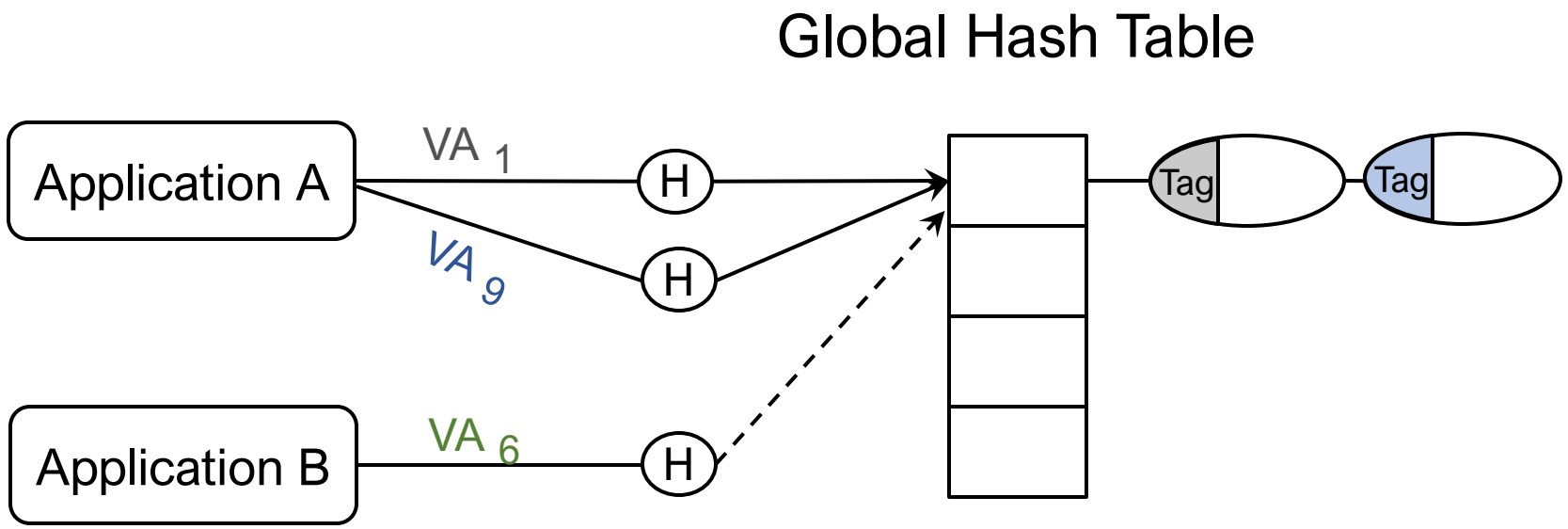
Alternative: Global Hashed Page Table



OS is invoked to resolve them!

Alternative: Global Hashed Page Table

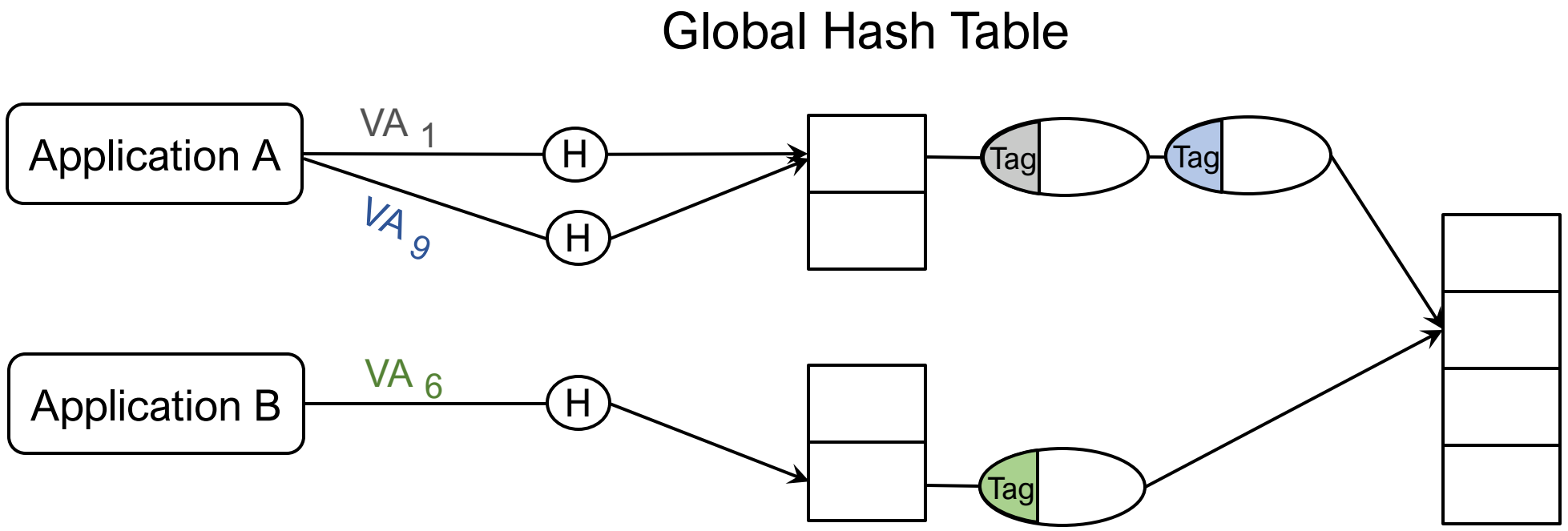
How to share pages?



Alternative: Global Hashed Page Table

How to share pages?

New level of indirection!!

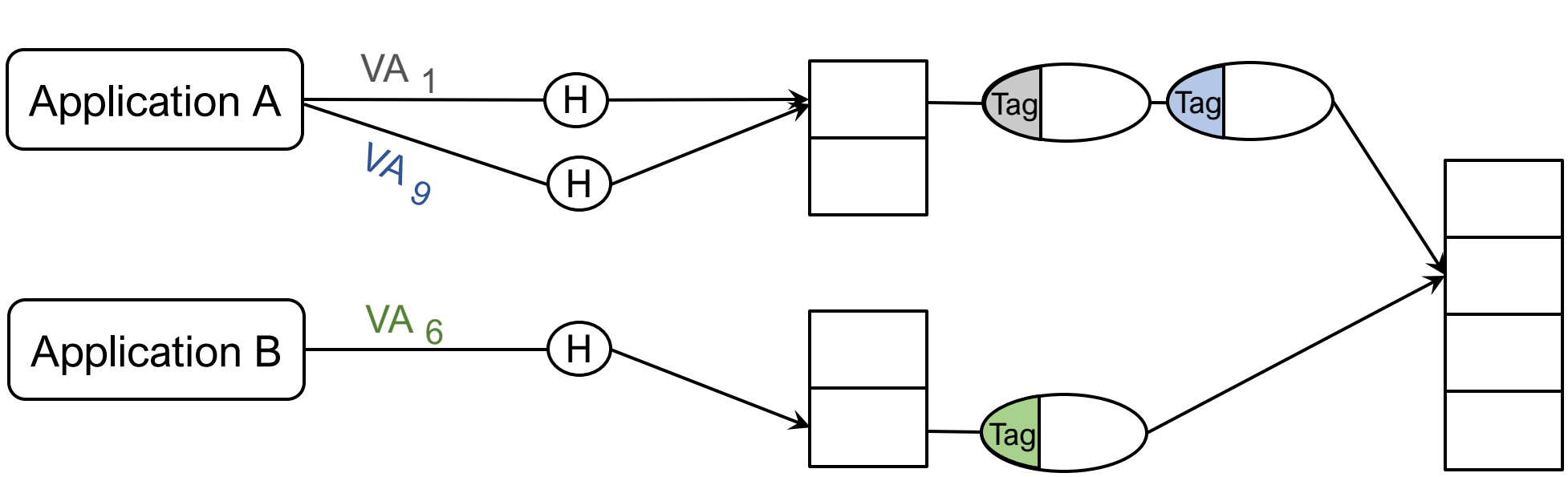


Alternative: Global Hashed Page Table

How to share pages?

Multiple page sizes?

New level of indirection!!

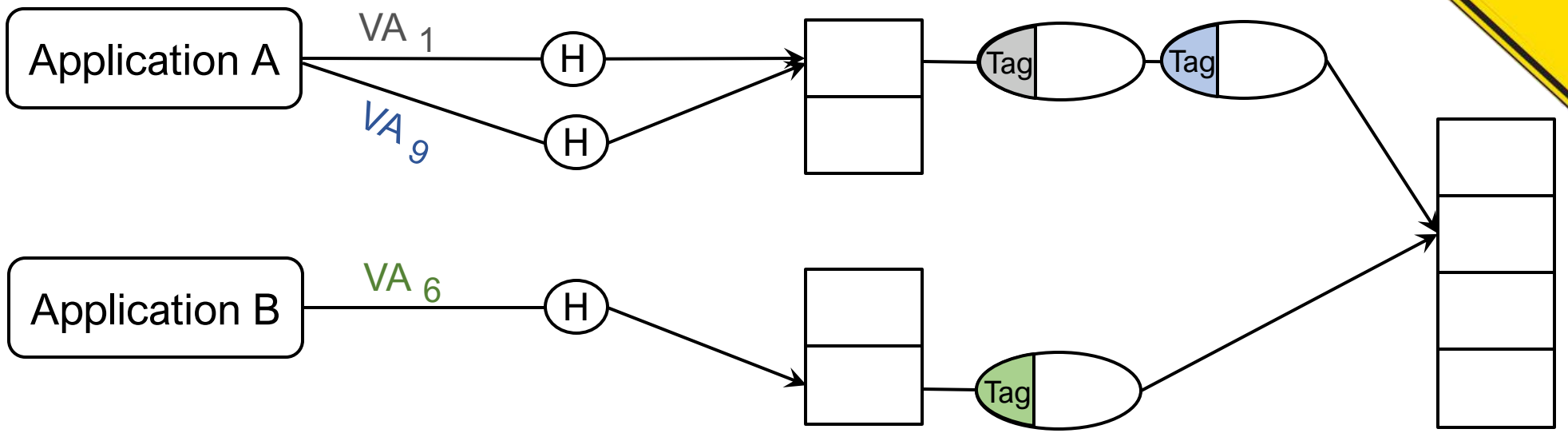


Alternative: Global Hashed Page Table

How to share pages?
Multiple page sizes?

New level of indirection!!

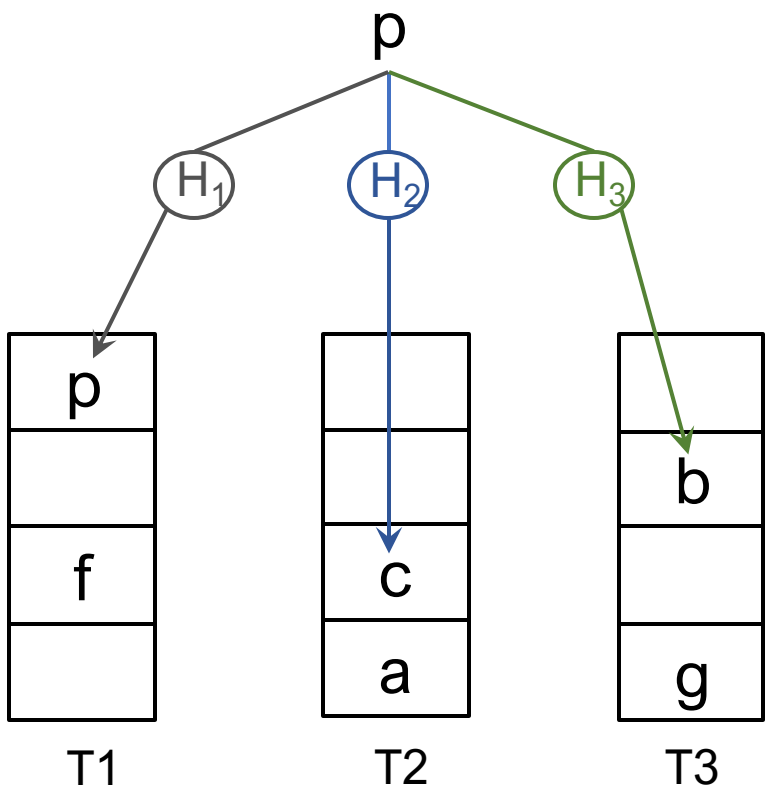
Global Hash Table



Elastic Cuckoo Page Tables

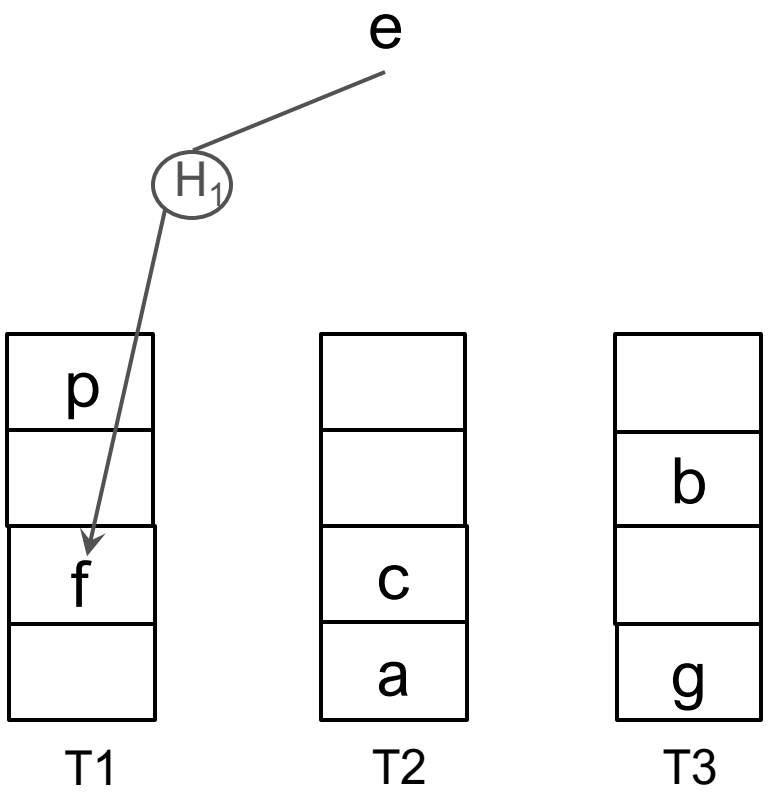
Rethinking virtual memory translation for parallelism

Cuckoo Hashing



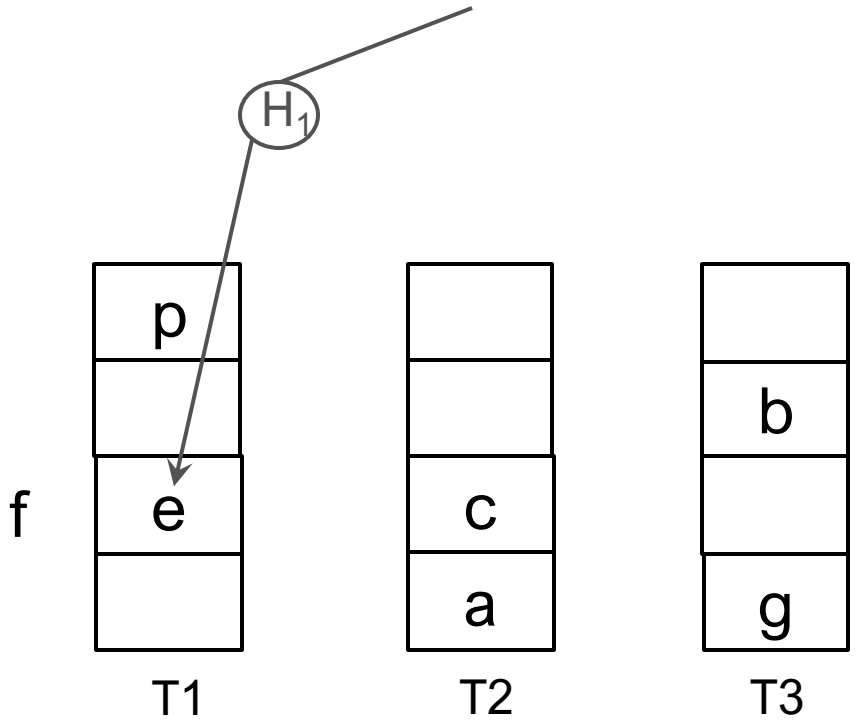
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



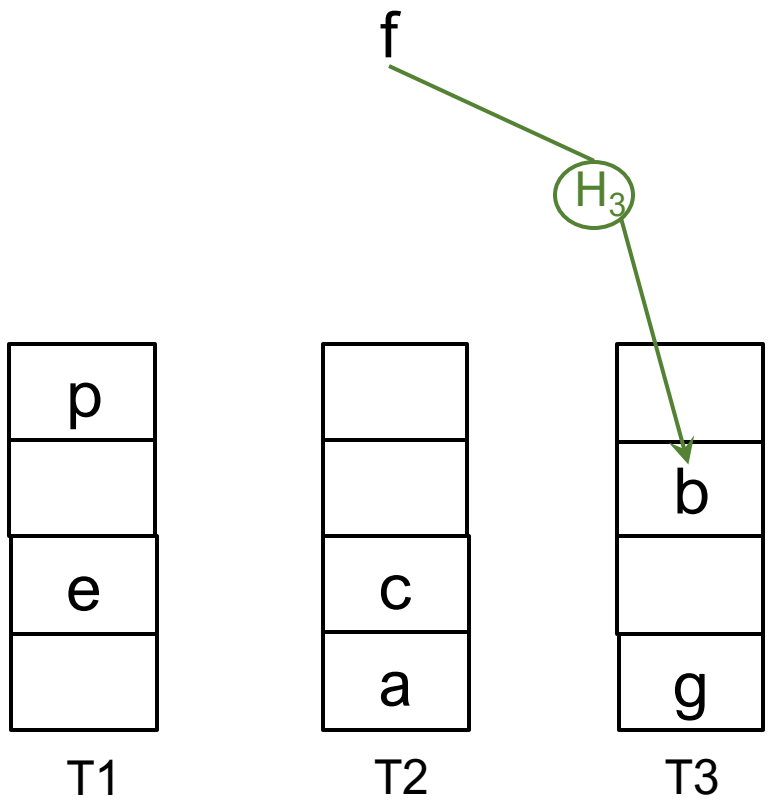
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



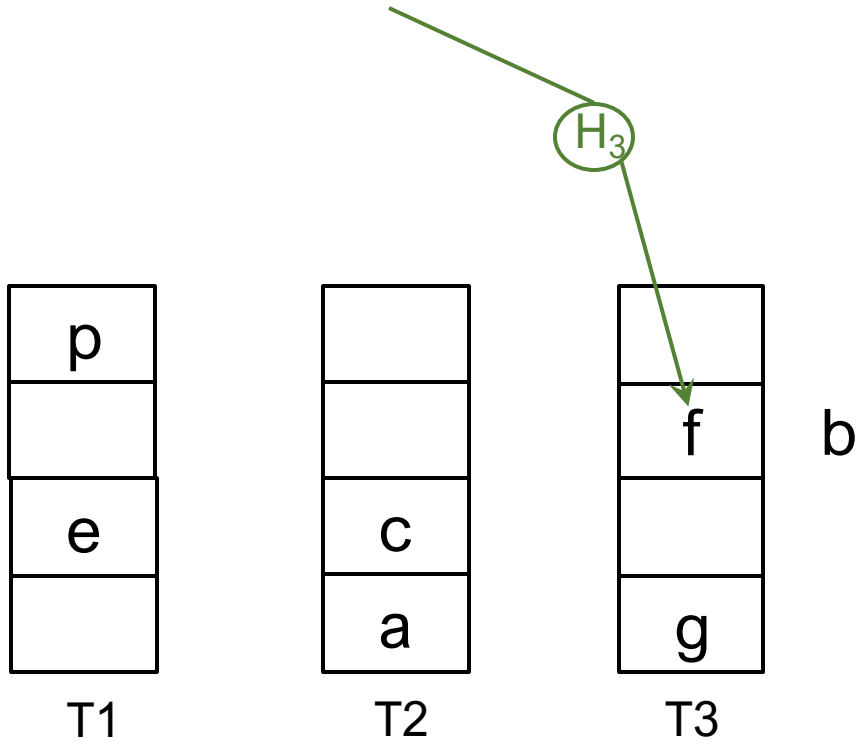
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



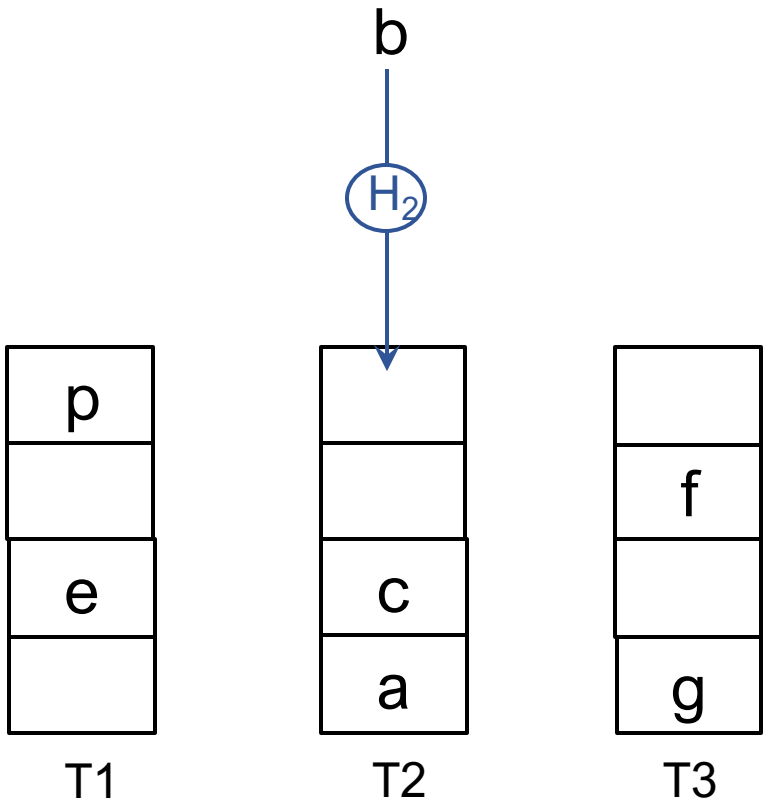
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



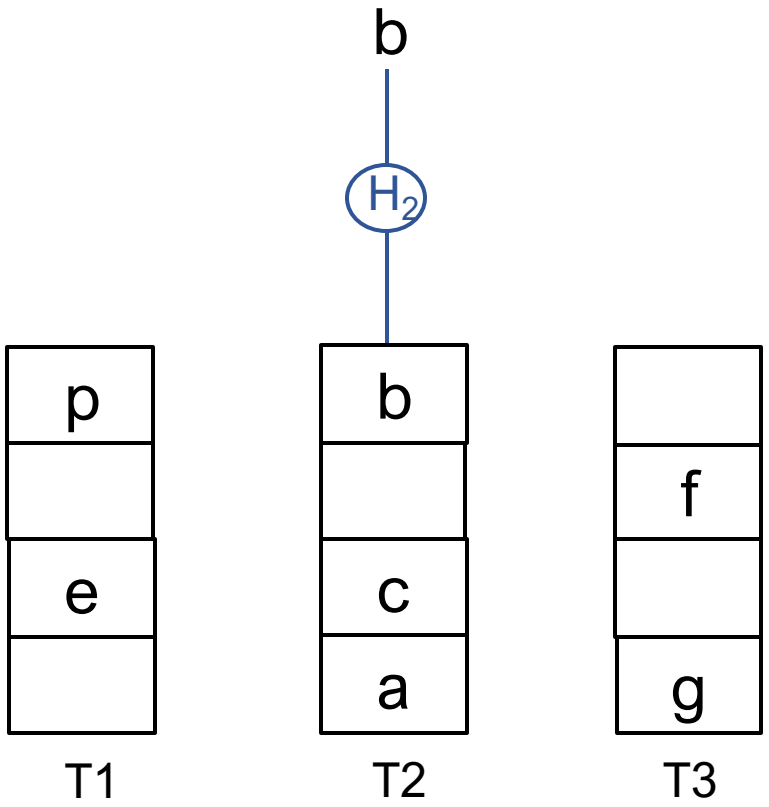
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



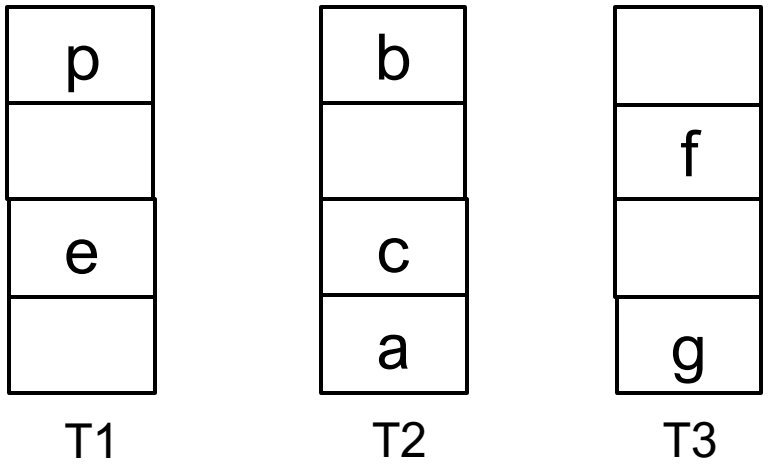
d-ary Cuckoo Hash Table

Insertions with Cuckoo Hashing



d-ary Cuckoo Hash Table

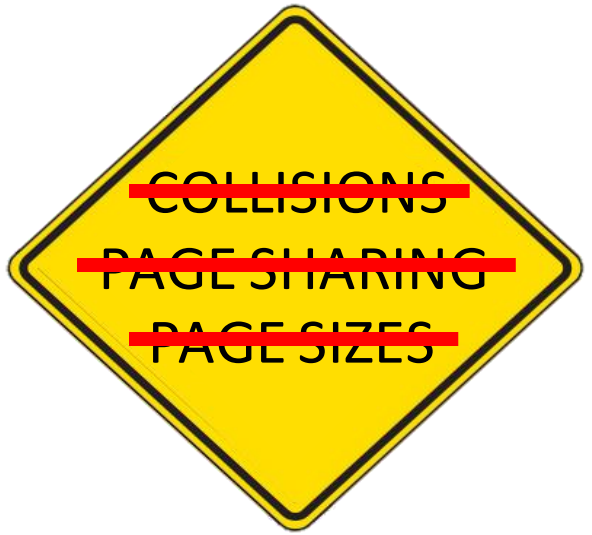
Insertions with Cuckoo Hashing

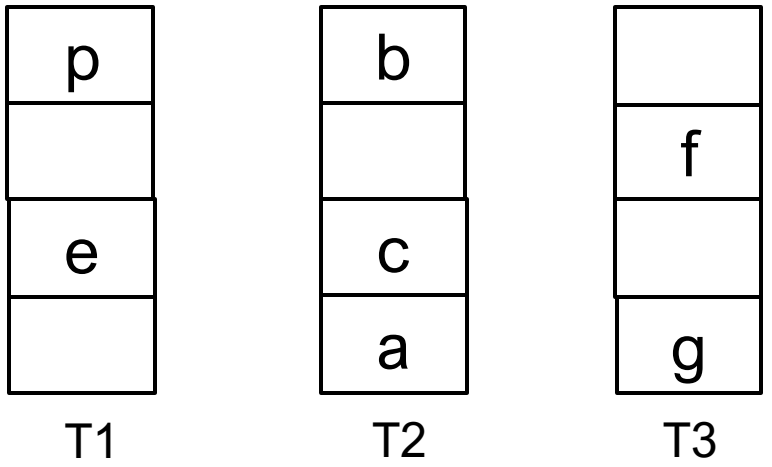


d-ary Cuckoo Hash Table

Private Hashed Page Tables

PRIVATE
PAGE TABLES





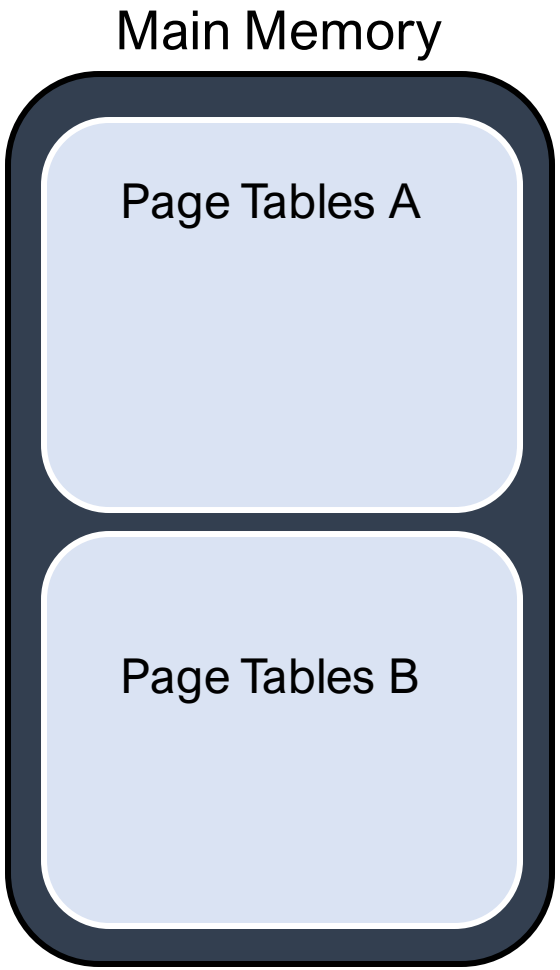
d-ary Cuckoo Hash Table

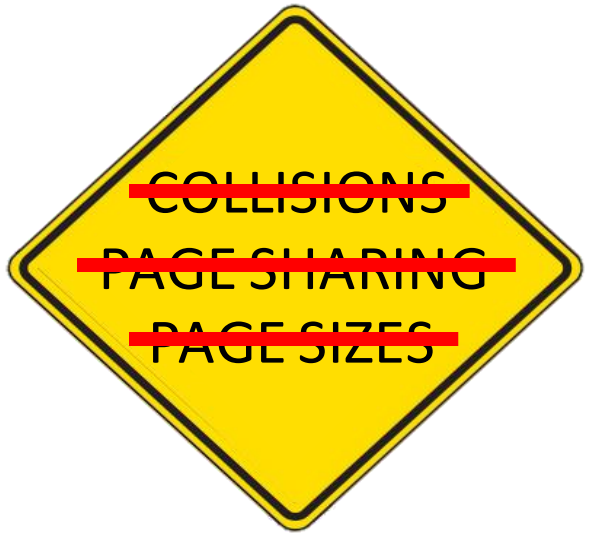
Cannot Be Too Big → Waste Memory

PRIVATE PAGE TABLES

App A

App B





Private page tables cannot be too big

Need to Dynamically Resize

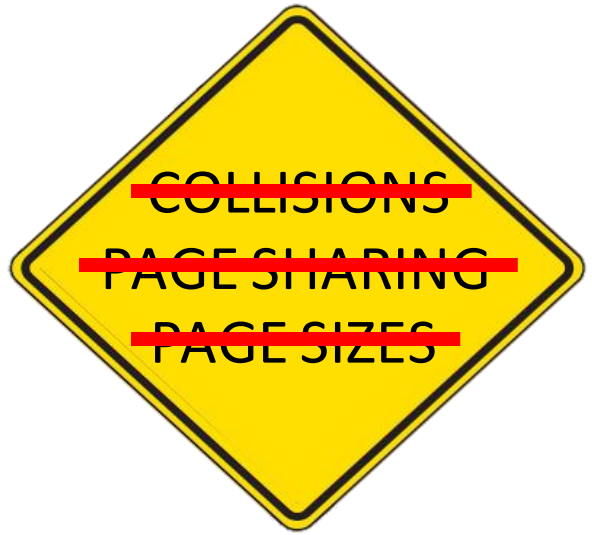
PRIVATE PAGE TABLES

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

Cannot Rehash All Entries at Once

PRIVATE PAGE TABLES

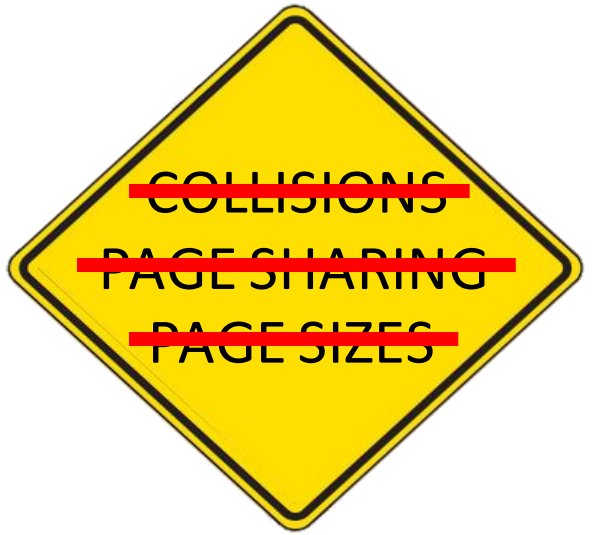
App A

Main Memory

Page Tables A

VA 1	PA 4
VA 8	PA 1

New Page Tables A



Private page tables cannot be too big

↓

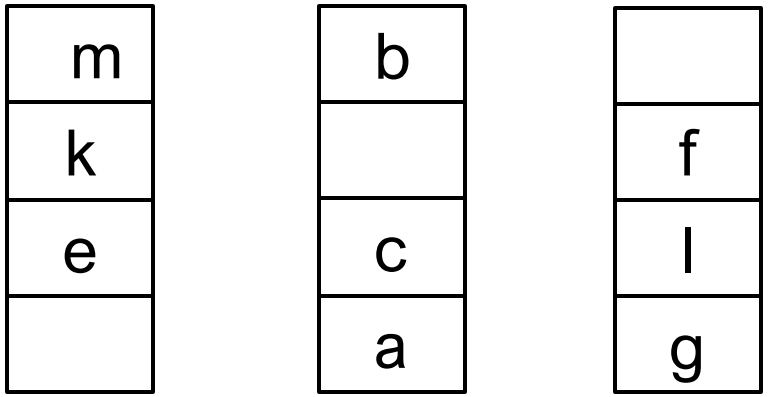
Need to dynamically resize

↓

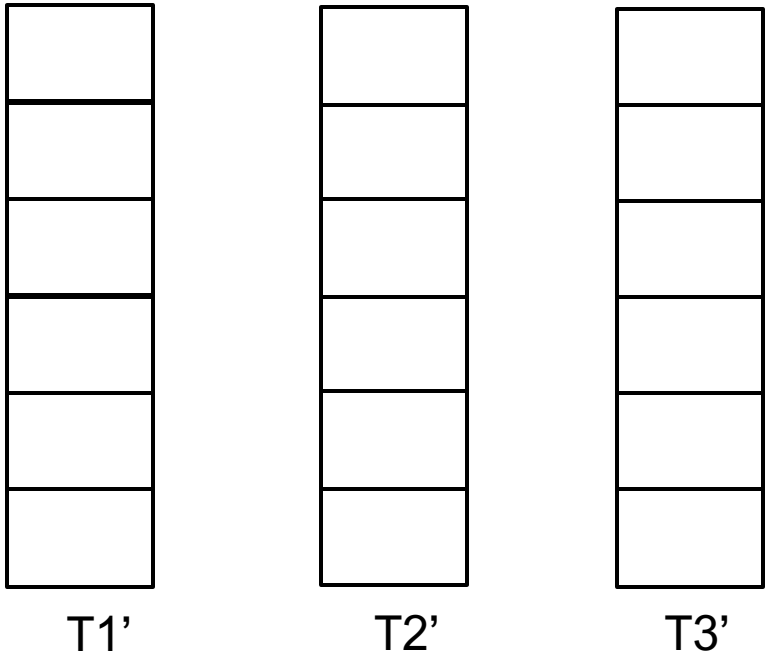
GRADUAL REHASHING
while the program is running

Gradual Rehashing of Cuckoo Hash Tables

At every insert → Rehash one element



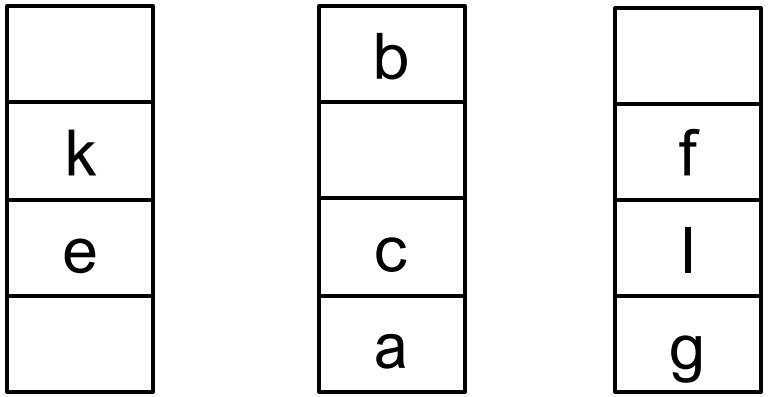
Old *d*-ary Cuckoo Hash Table



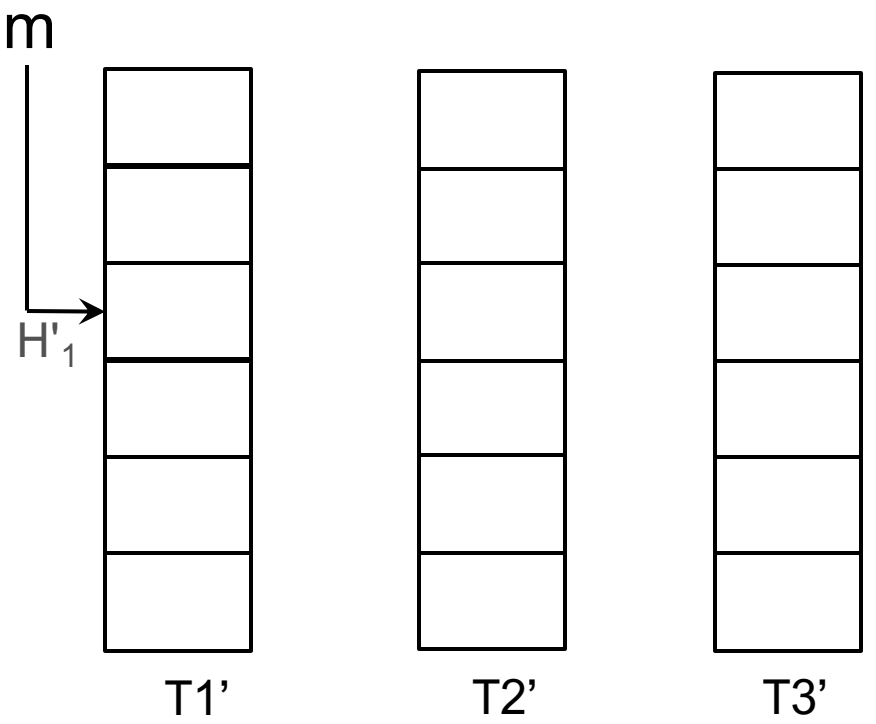
New *d*-ary Cuckoo Hash Table

Gradual Rehashing of Cuckoo Hash Tables

At every insert → Rehash one element

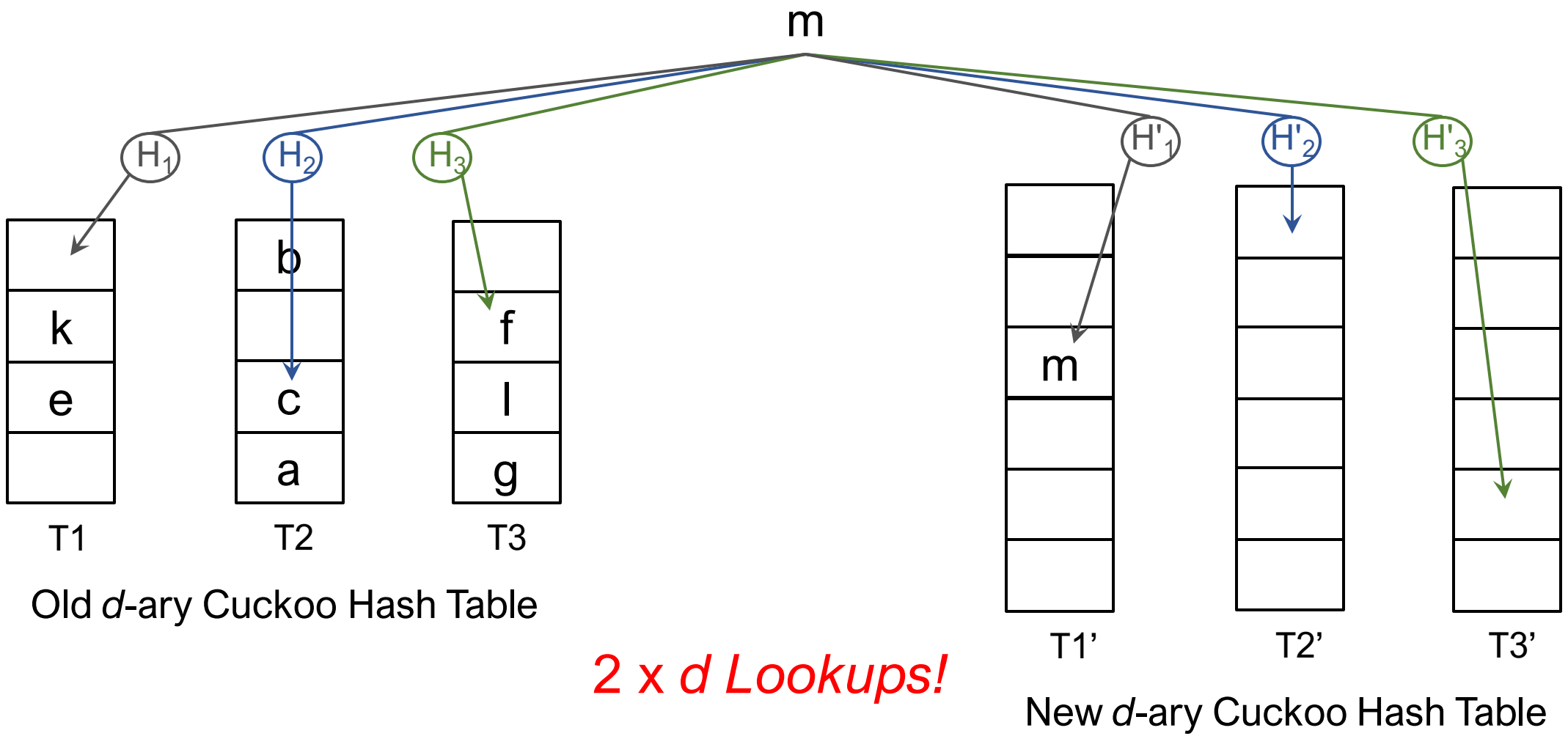


Old *d*-ary Cuckoo Hash Table

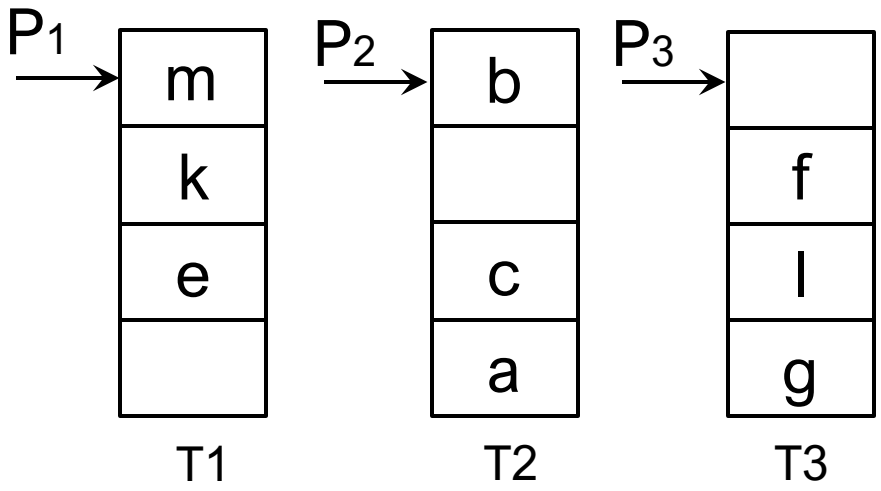


New *d*-ary Cuckoo Hash Table

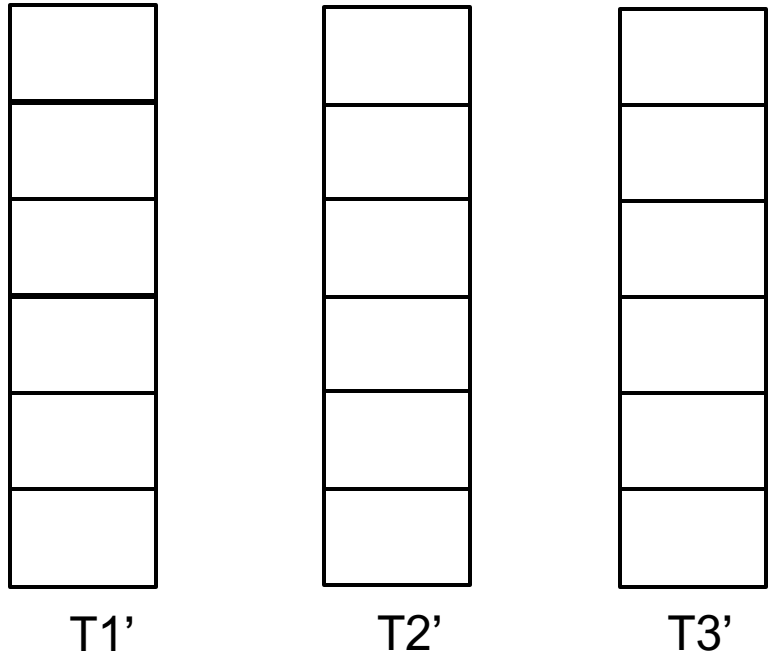
Lookup During Gradual Rehashing



Solution: Rehashing Pointers

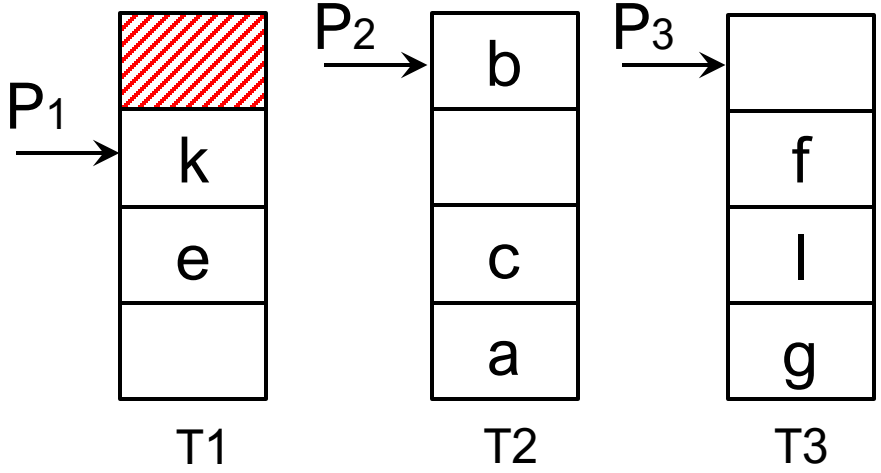


Old *d*-ary Cuckoo Hash Table

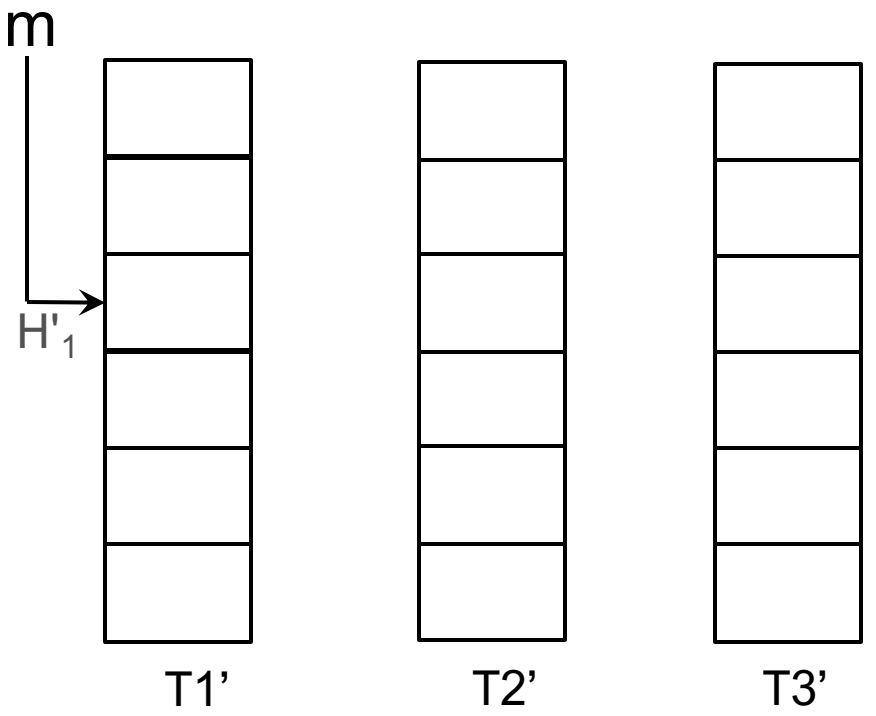


New *d*-ary Cuckoo Hash Table

Solution: Rehashing Pointers



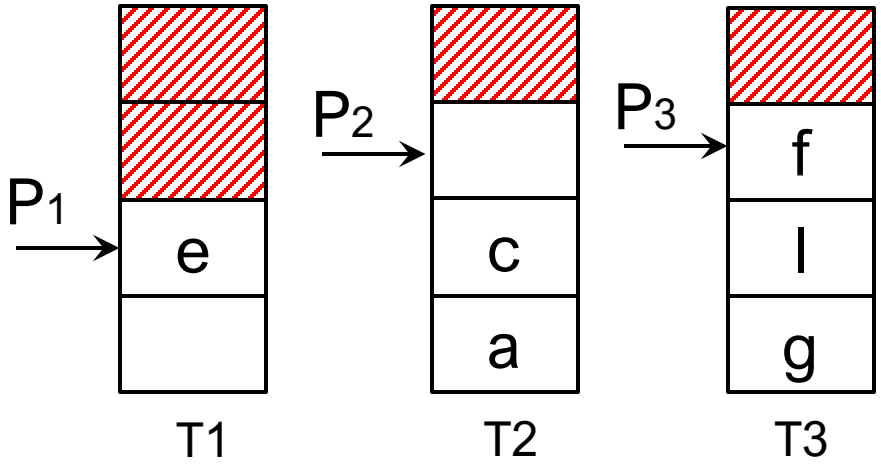
Old *d*-ary Cuckoo Hash Table



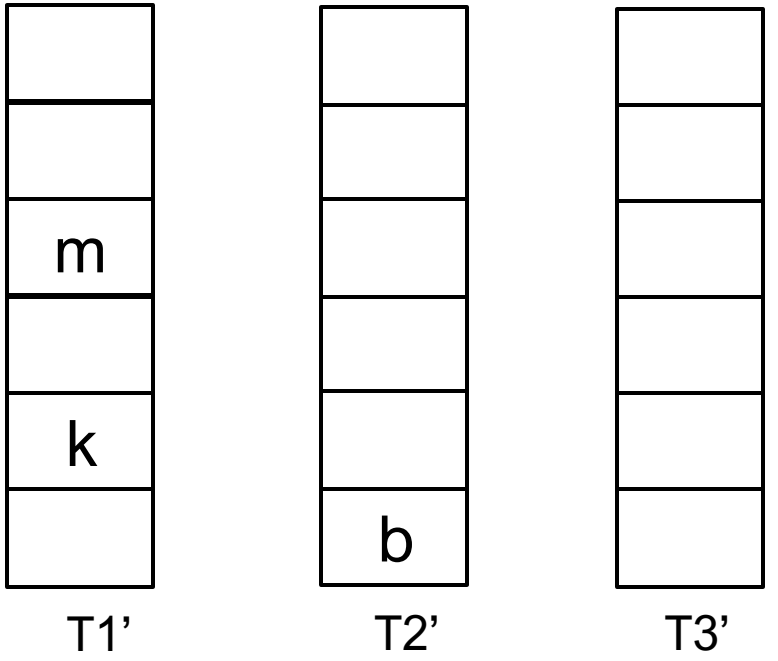
New *d*-ary Cuckoo Hash Table

Migrated Region

Solution: Rehashing Pointers



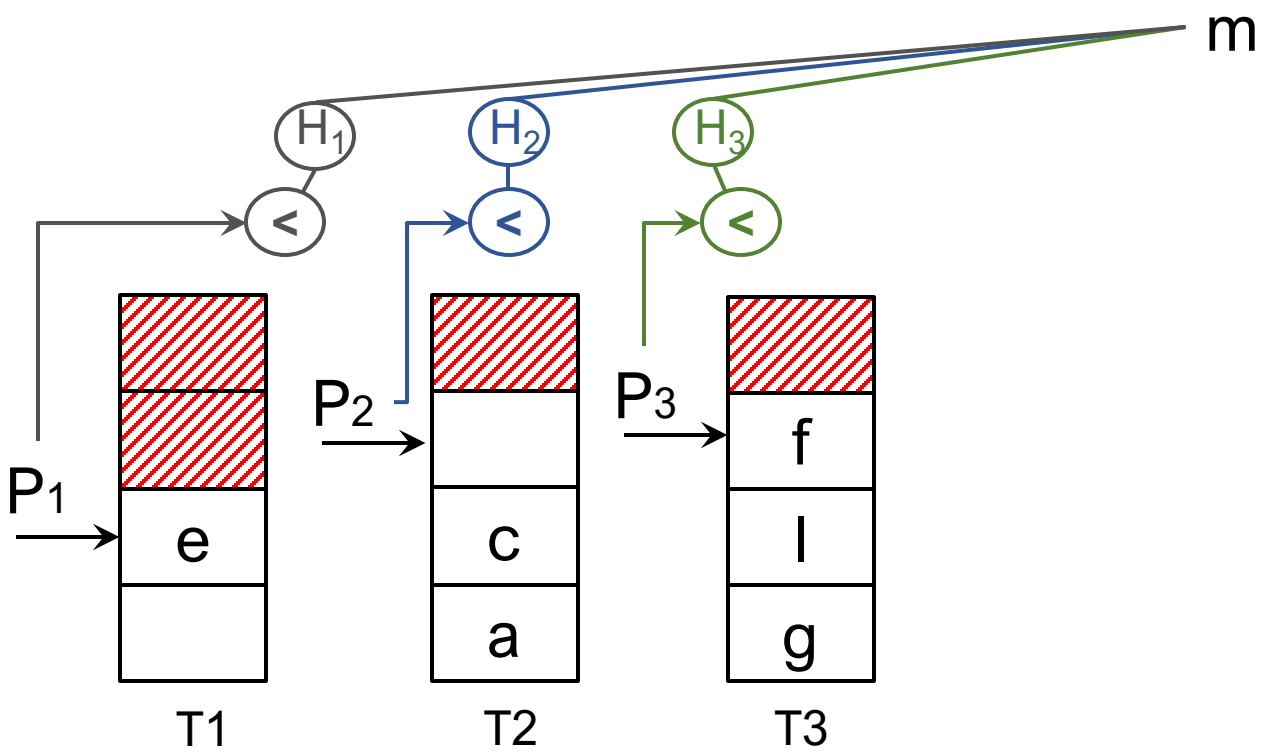
Old *d*-ary Cuckoo Hash Table



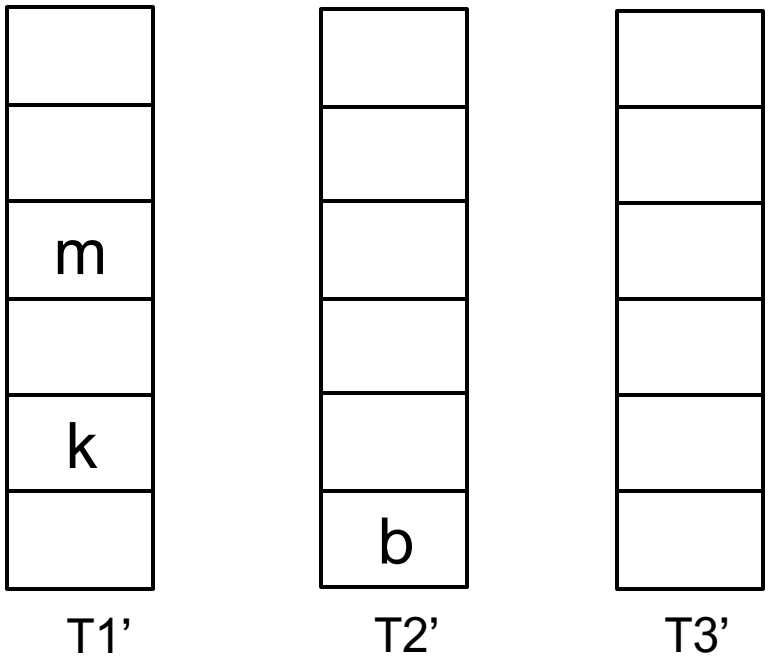
New *d*-ary Cuckoo Hash Table

Migrated Region

Lookup with Rehashing Pointers



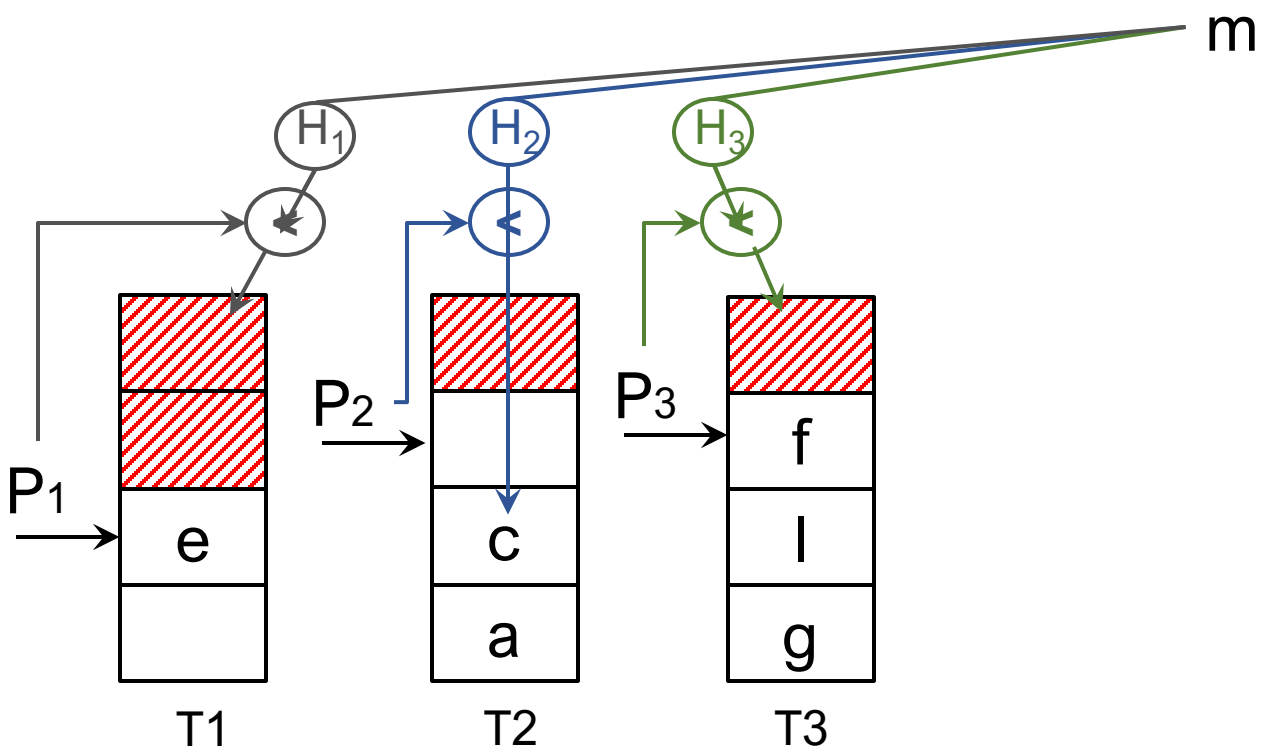
Old *d*-ary Cuckoo Hash Table



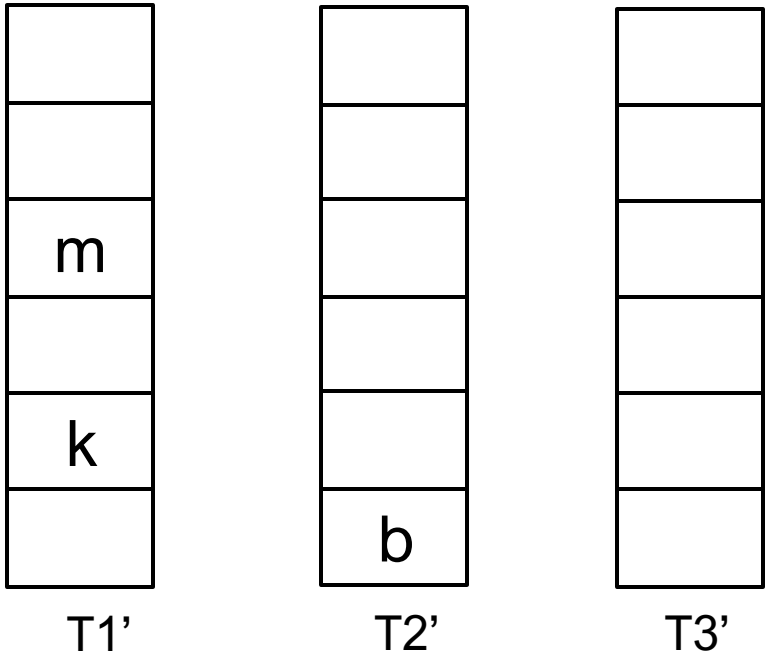
New *d*-ary Cuckoo Hash Table

Migrated Region

Lookup with Rehashing Pointers



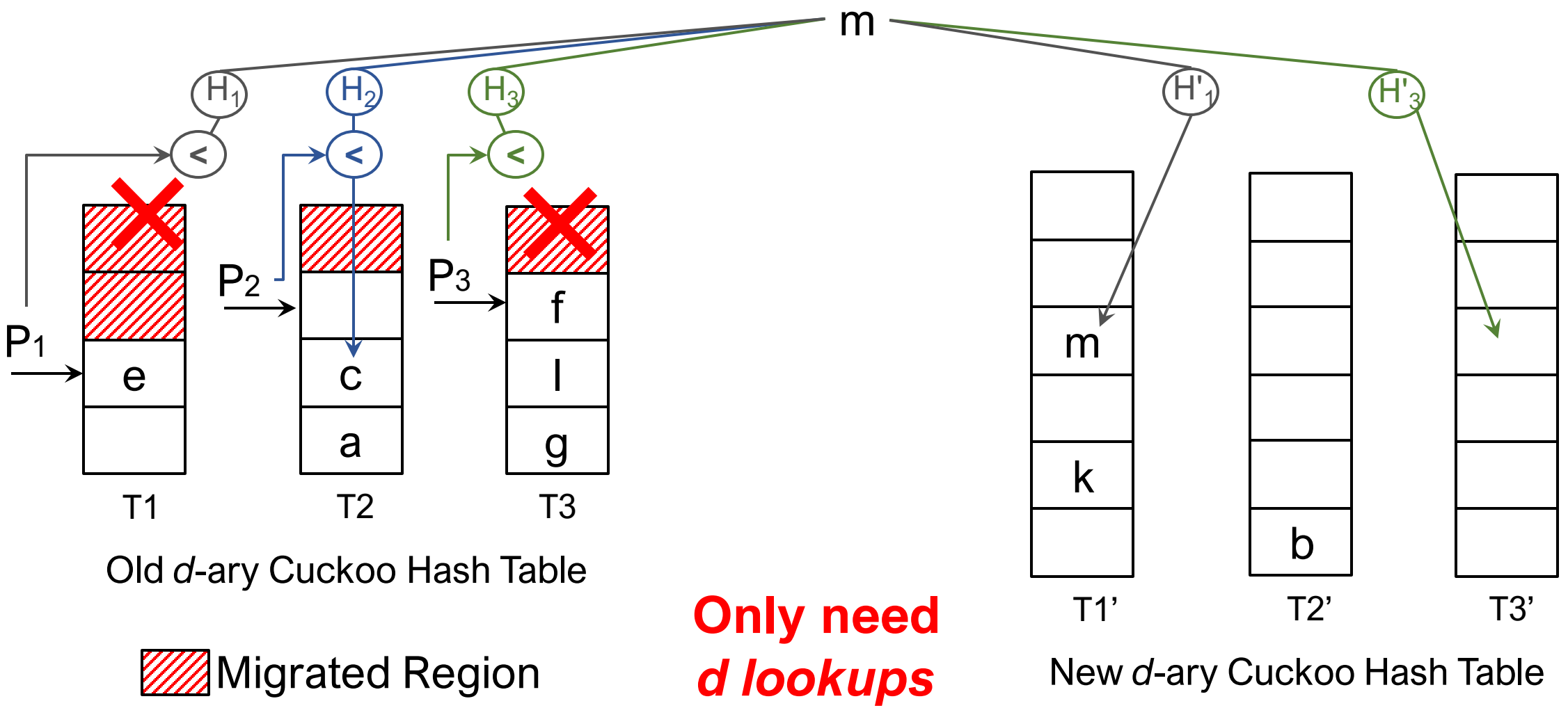
Old *d*-ary Cuckoo Hash Table



New *d*-ary Cuckoo Hash Table

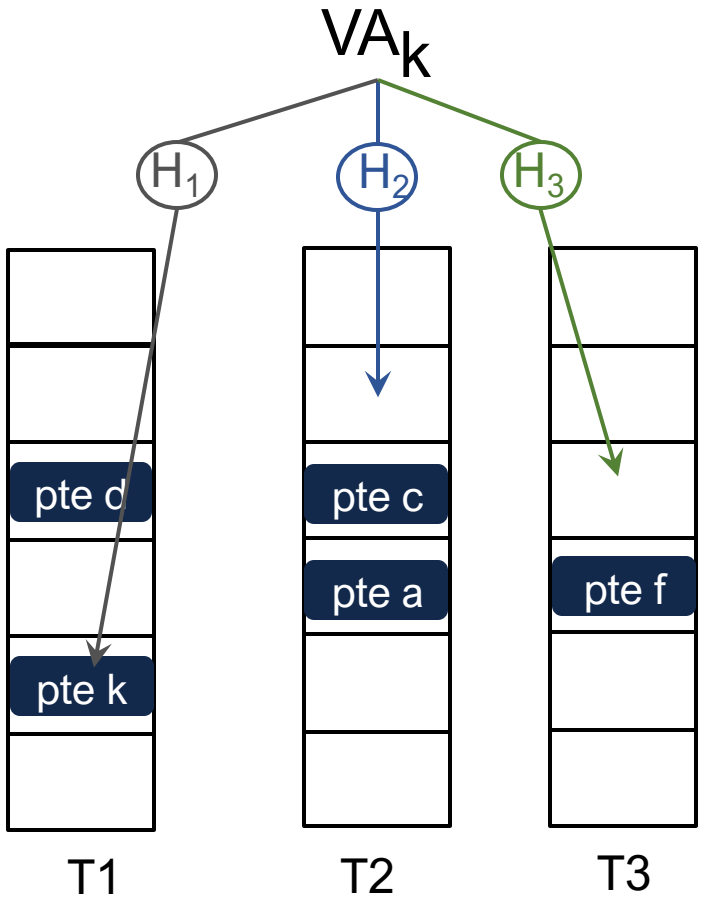
Migrated Region

Lookup with Rehashing Pointers



Only need *d lookups* *during resizing*

Exploiting Parallelism in Virtual Translation



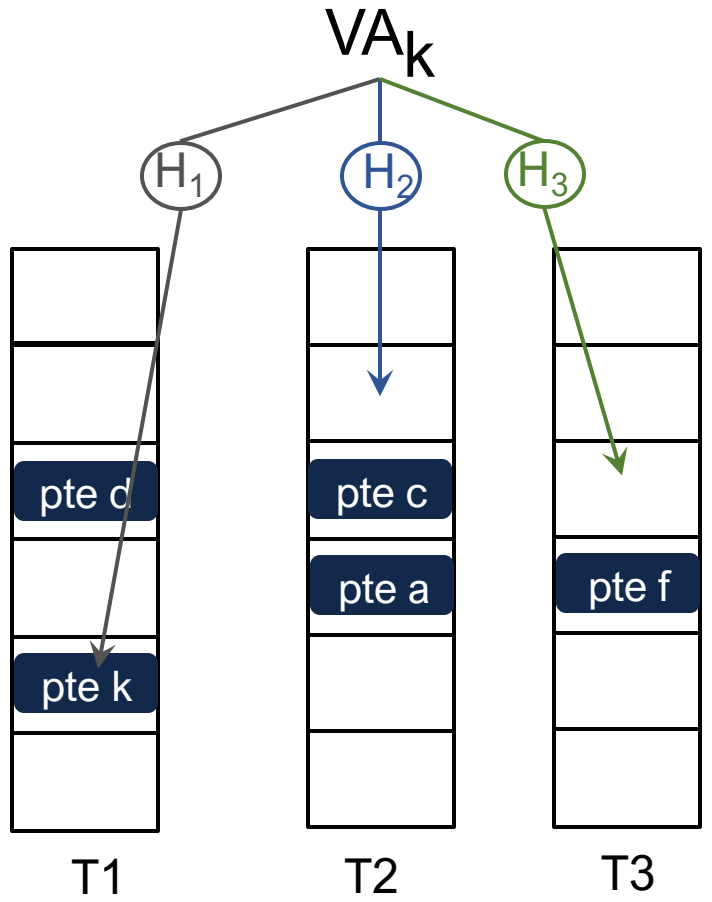
d-ary Elastic Cuckoo Page Table

Exploiting Parallelism in Virtual Translation

No sequential page walk (*unlike radix*)

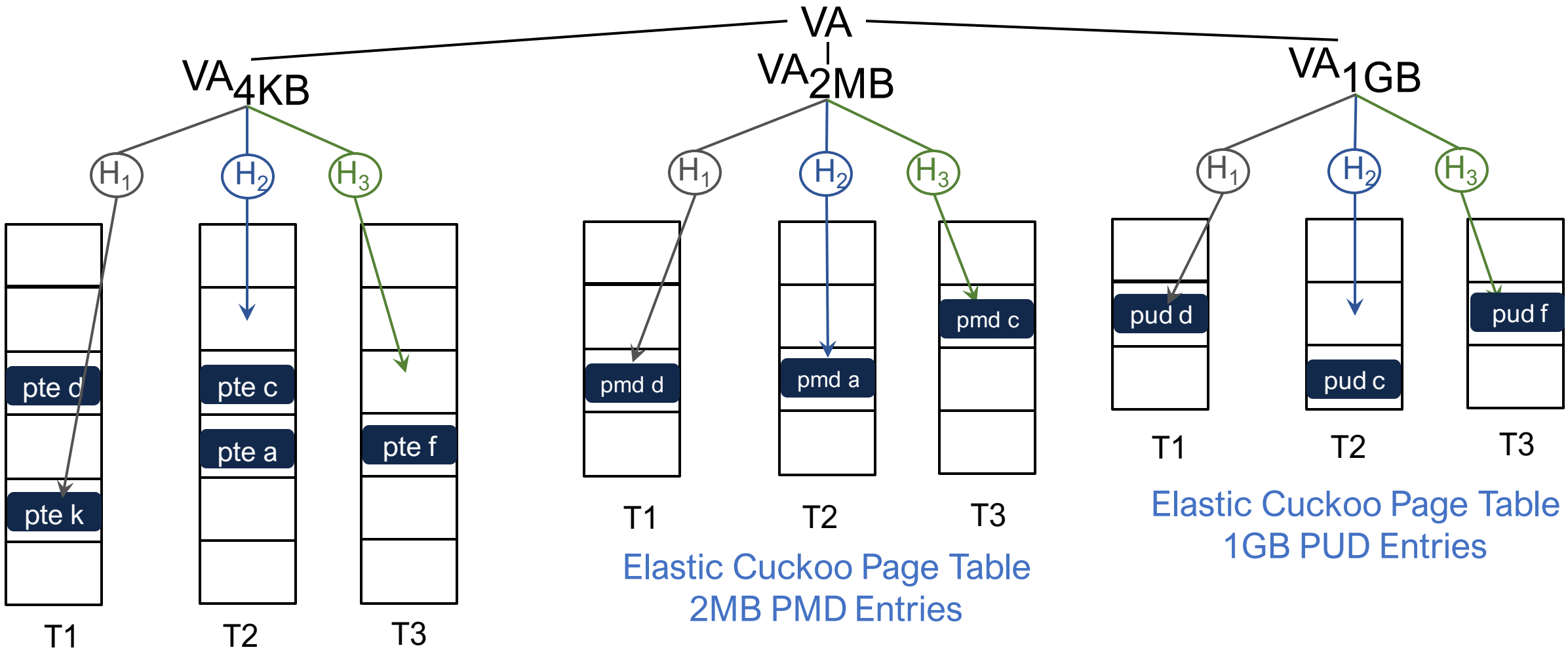
At most *d* accesses *always*

Leverages multiple issue out-of-order processors



d-ary Elastic Cuckoo Page Table

Lookup Multiple Page Sizes in Parallel



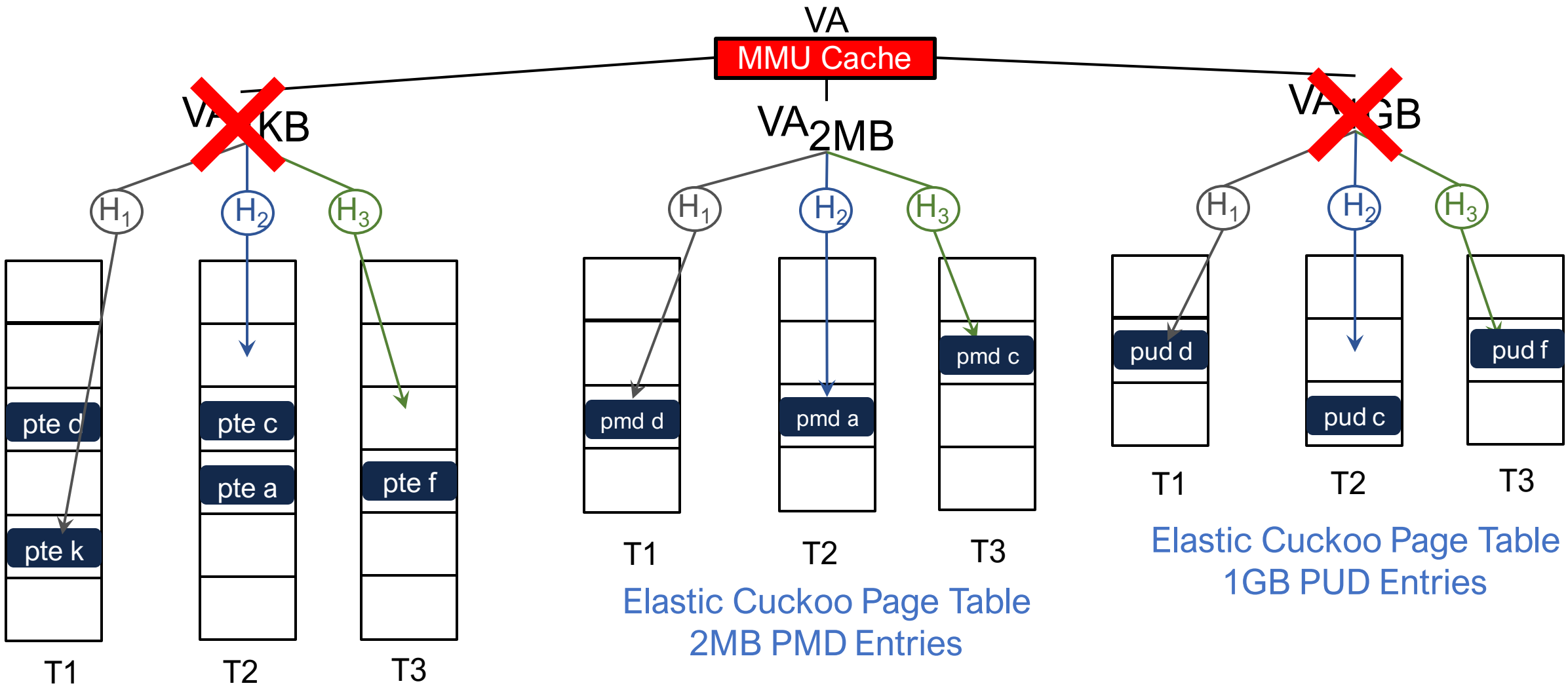
d-ary Elastic Cuckoo Page Table
4KB PTE Entries

Elastic Cuckoo Page Table
2MB PMD Entries

Elastic Cuckoo Page Table
1GB PUD Entries

ILLINOIS

New MMU Cache to Prune Parallelism



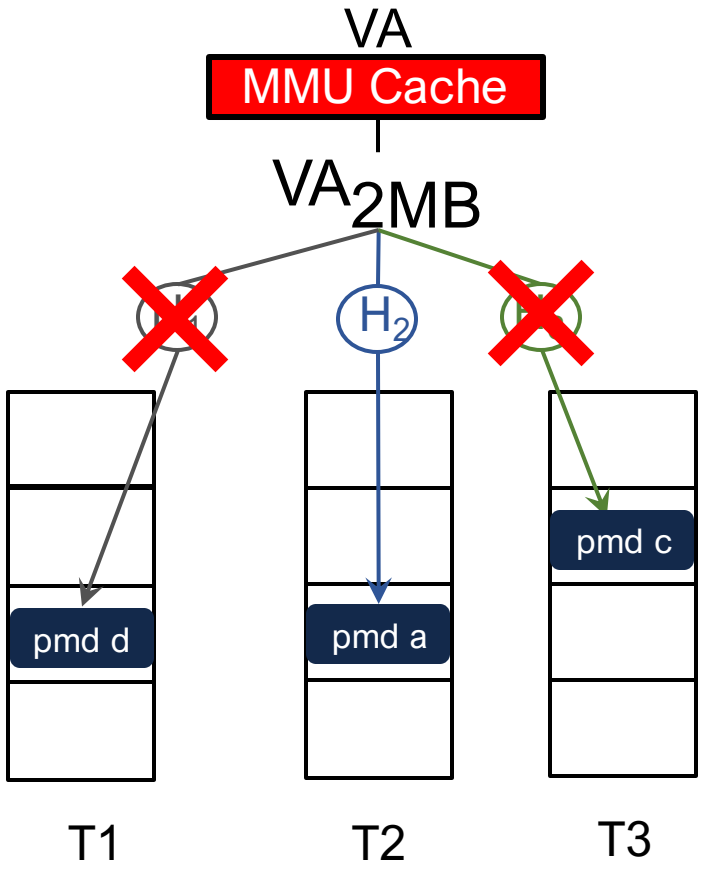
d-ary Elastic Cuckoo Page Table
4KB PTE Entries

Elastic Cuckoo Page Table
2MB PMD Entries

Elastic Cuckoo Page Table
1GB PUD Entries

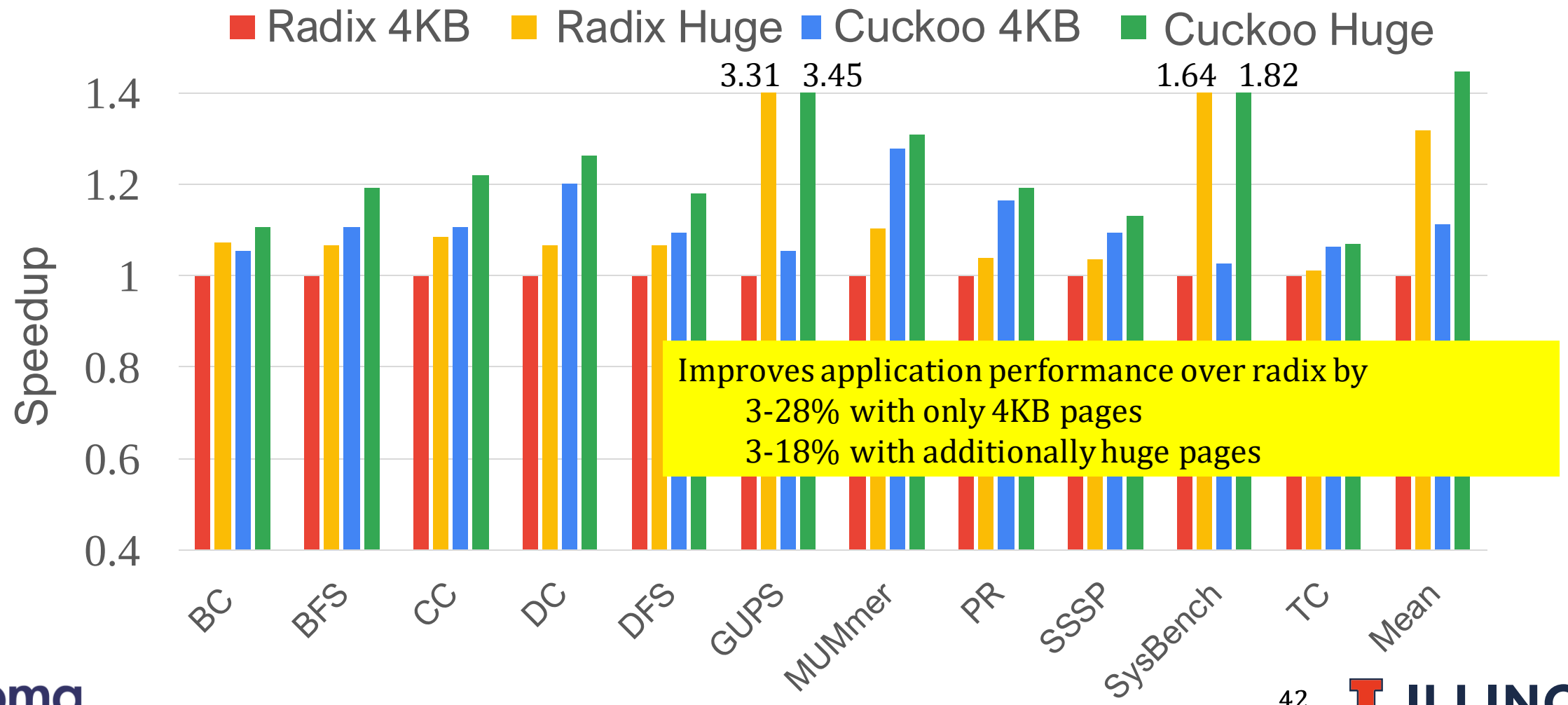
ILLINOIS

New MMU Cache to Prune Parallelism



Elastic Cuckoo Page Table
2MB PMD Entries

Application Speedup



Takeaway: Elastic Cuckoo Page Tables

Scalable alternative to existing radix page tables

- Exploits parallelism in virtual translation for the first time
- Reduces the cost of hash table resizing
- Improves application performance over radix by
  - 3-28% with only 4KB pages
  - 3-18% with additionally huge pages
- Expect even higher performance impact on virtualized environments
- Current work: graceful migration from current Radix designs to Elastic Cuckoo

Rethinking Address Translation in the Age of Containers, Serverless Computing, and Non-Volatile Memory

Josep Torrellas, Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu

Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

ARM Research Summit
September 2020