Parallel Virtualized Memory Translation with Nested Elastic Cuckoo Page Tables

Jovan Stojkovic, Dimitrios Skarlatos‡, Apostolos Kokolis, Tianyin Xu, Josep Torrellas

jovans2@illinois.edu

University of Illinois at Urbana-Champaign ‡Carnegie Mellon university

ASPLOS, March 2022

i-acoma group

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Carnegie Mellon University Computer Science Department

Virtualization

Cloud computing virtualizes hardware for strong isolation and server consolidation

+ Virtual Machines multiplexed over hardware resources and offer a safe sand-boxing

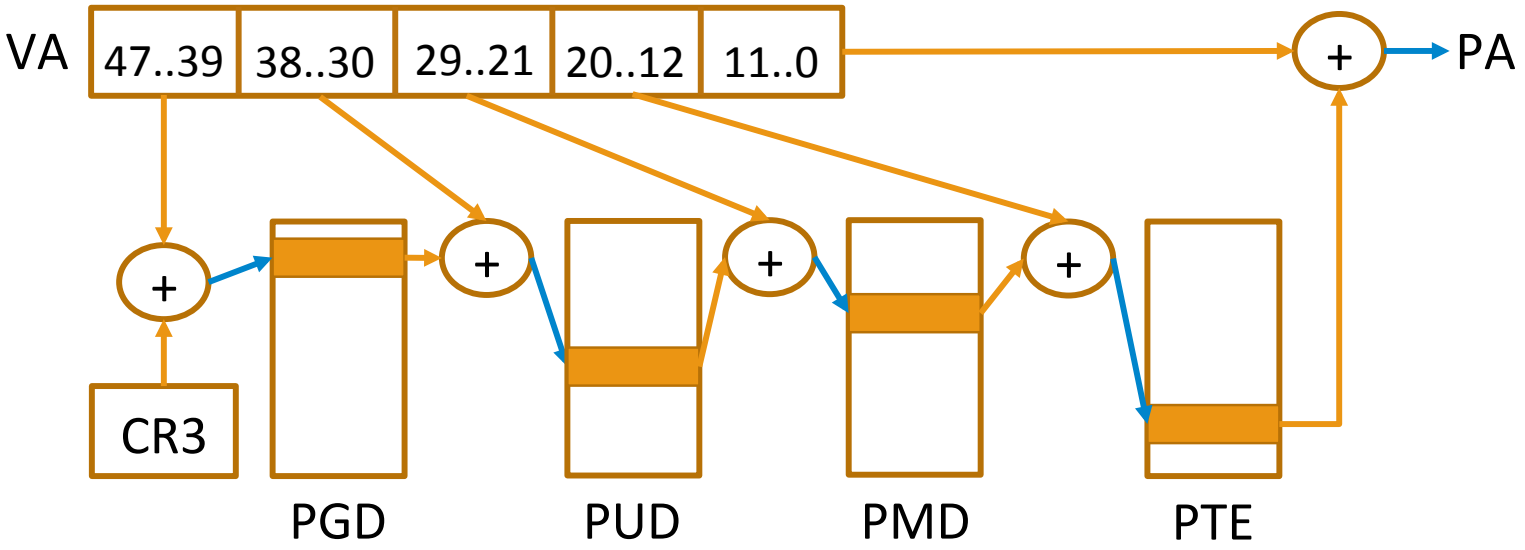
+ Lightweight virtualization frameworks that reduce long boot-up time

- Address translation becomes more complicated

Radix Page Tables

Organized as a tree data structure

Current systems support 4-level tree (soon 5)



Nested Paging

In virtualized environments → nested paging

Physical memory is managed by the hypervisor

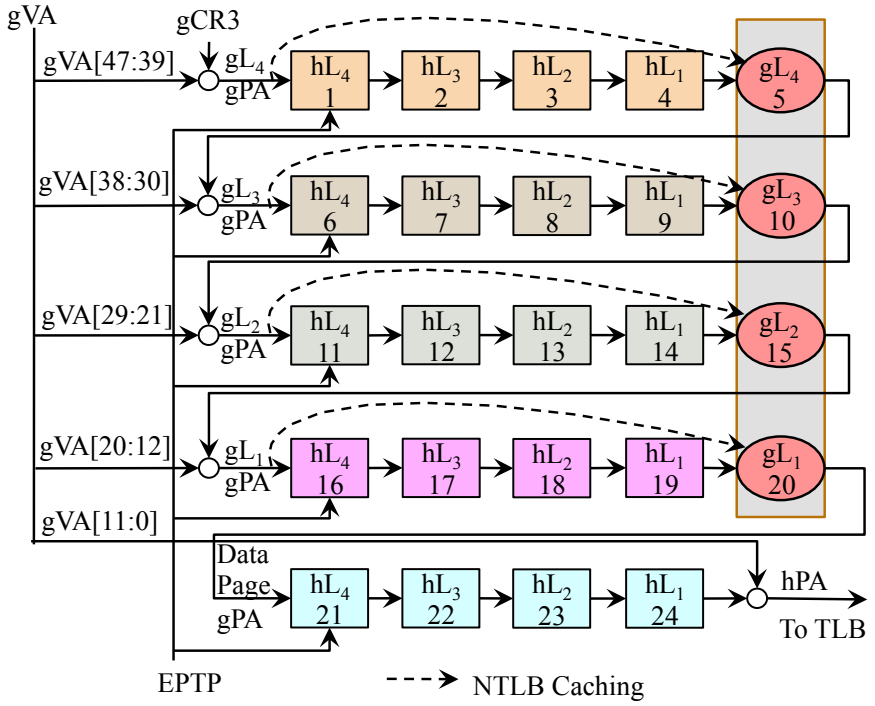
- Not exposed to guest OS

Guest Page Tables: gVA -> gPA

Host Page Tables: gPA -> hPA

Current systems support 4-level tree

- TLB miss: Up to 24 *sequential* memory references



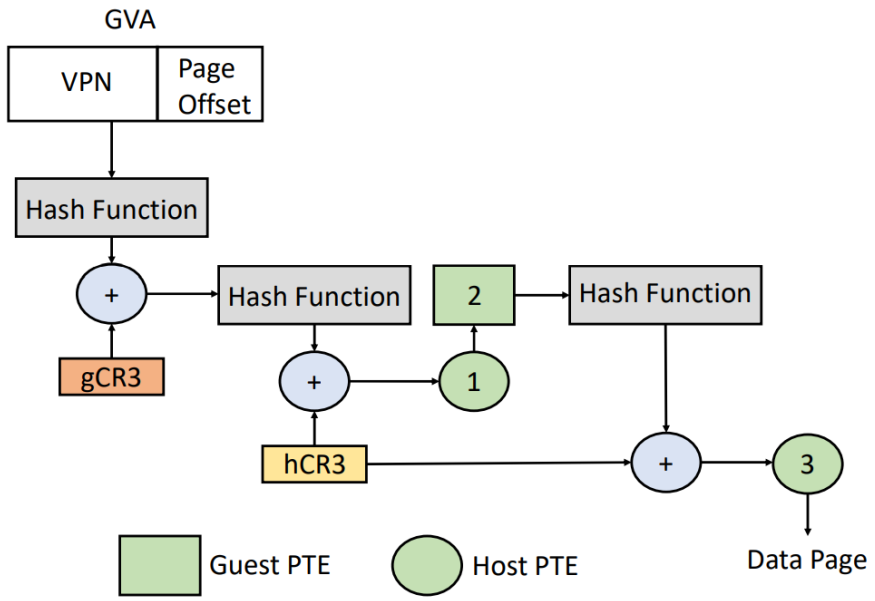
Hashed Page Tables

In native environments → 1 memory access

In virtualized environments → 3 memory accesses

Hash collisions
- Open addressing
- Chain walking
- Invoking OS

Single HPT shared by all processes
- No support for page sharing
- No support for multiple page sizes



Elastic Cuckoo Page Tables (ECPTs)

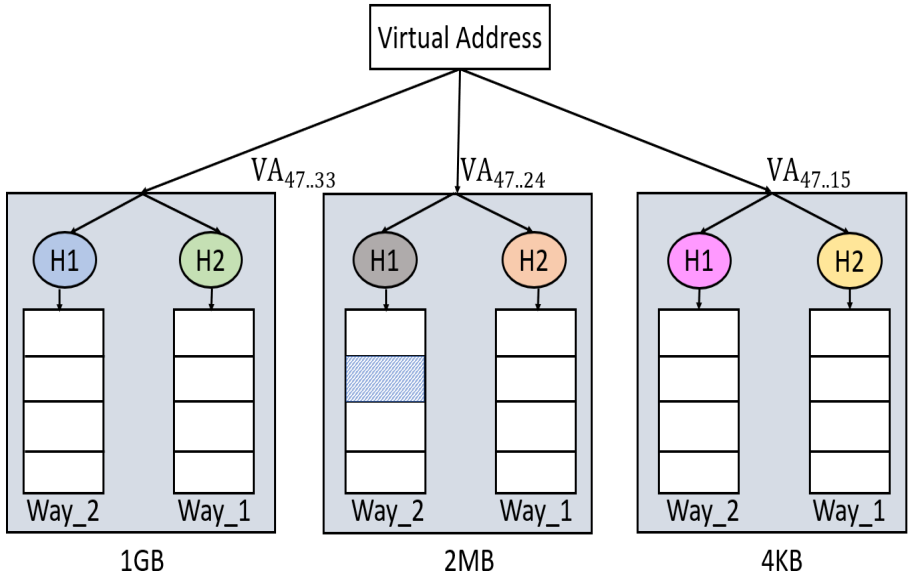
Hash collisions resolved via cuckoo hashing

Per-process private ECPT
- Multiple page sizes + page sharing

Dynamic ECPT resizing

Exploiting parallelism

Structures:
- Software: Cuckoo Walk Tables (CWTs)
  - Prune number of parallel requests
- Hardware: Cuckoo Walk Caches (CWCs)
  - Cache recently accessed CWT entries



Contributions

The first page table design for parallel nested address translation: Nested ECPTs
- Eliminate all but three of the potentially 24 sequential memory accesses of Nested Radix

Judiciously limits the number of parallel memory accesses issued
- Shortcut Translation Cache
- Caching some metadata (host PTE CWT), sometimes adaptively

Nested ECPTs outperform state-of-the-art Nested Radix tables
- Avg. 1.19X for 4KB pages
- Avg. 1.24X for 4KB + 2MB pages

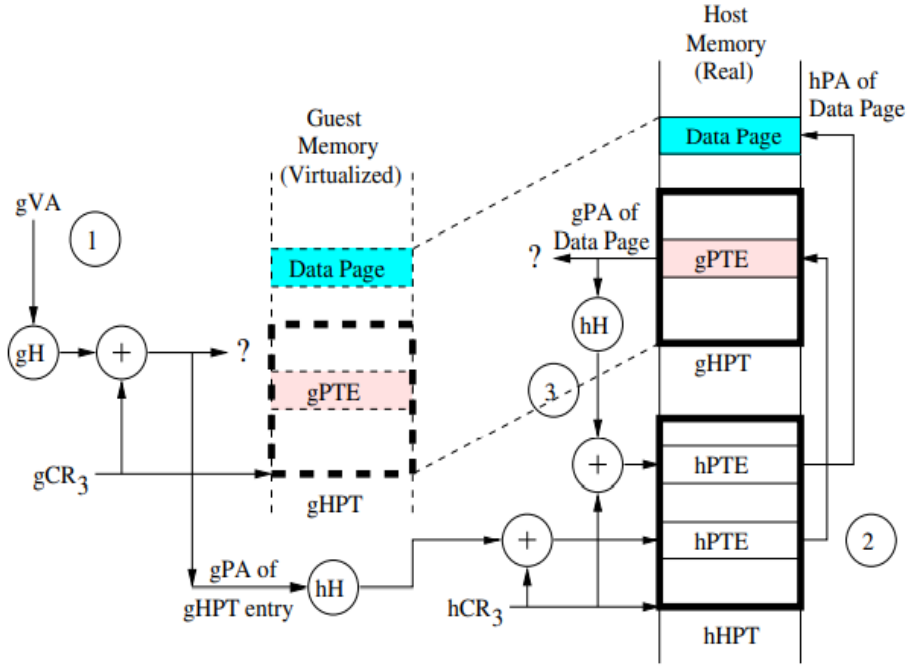
Possible migration path from Nested Radix to Nested ECPTs

Proposal: Nested ECPTs

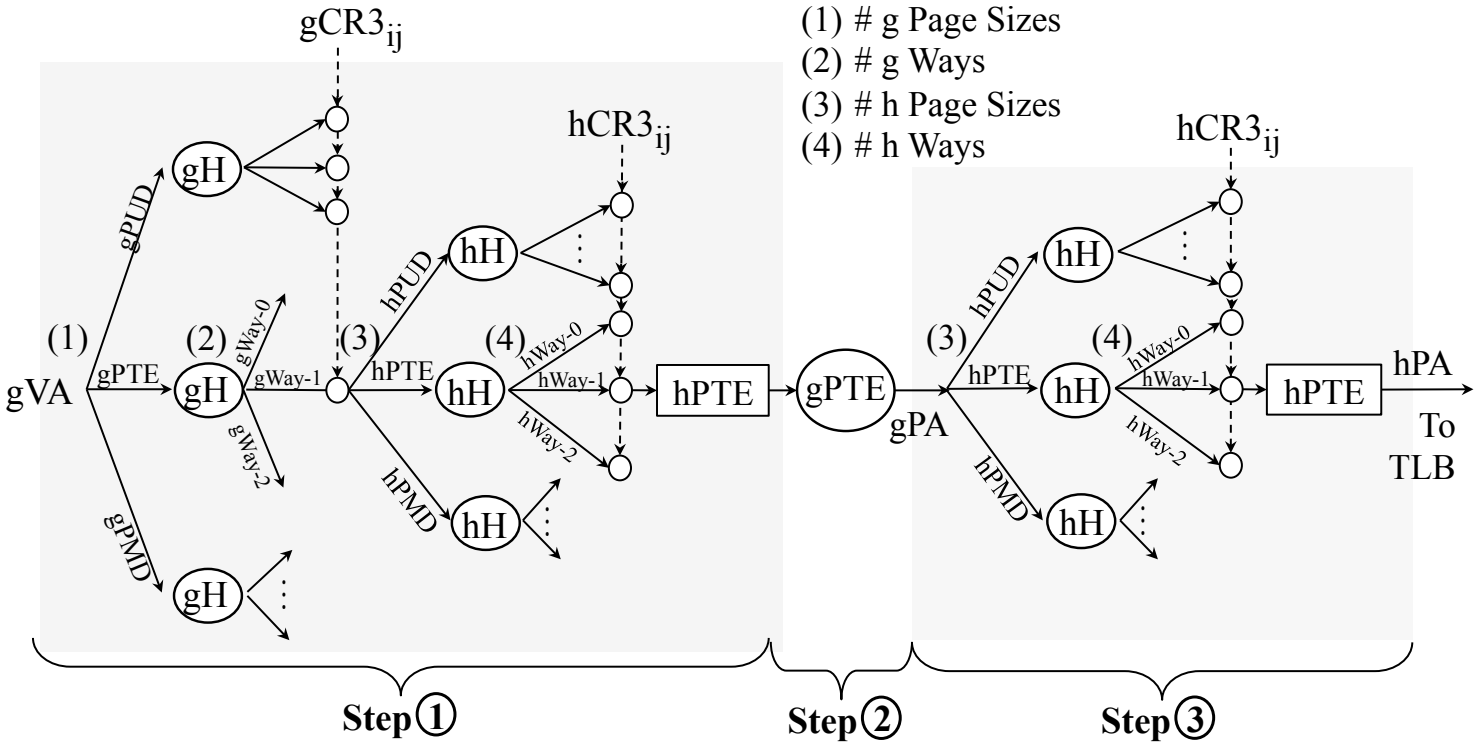
Goal: Speed-up the virtualized translation process by exploiting parallelism

Employ Elastic Cuckoo Page Tables (ECPTs)

Plain Nested ECPTs → Directly incorporate ECPT structures for both guest and host



Design without Limiting Memory Accesses



Nested ECPTs With Caches

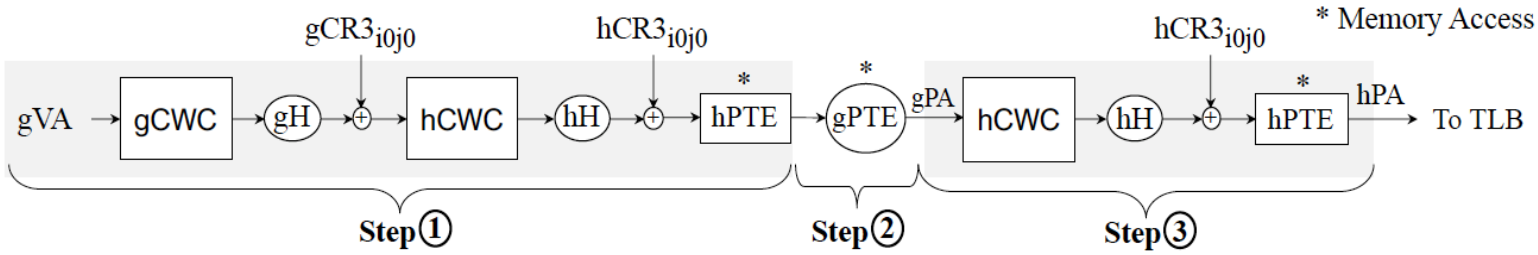
Previous design can result in many parallel memory requests

- $n \uparrow 2 \times d \uparrow 2$ (Step 1), $n \times d$ (Step 2), $n \times d$ (Step 3)

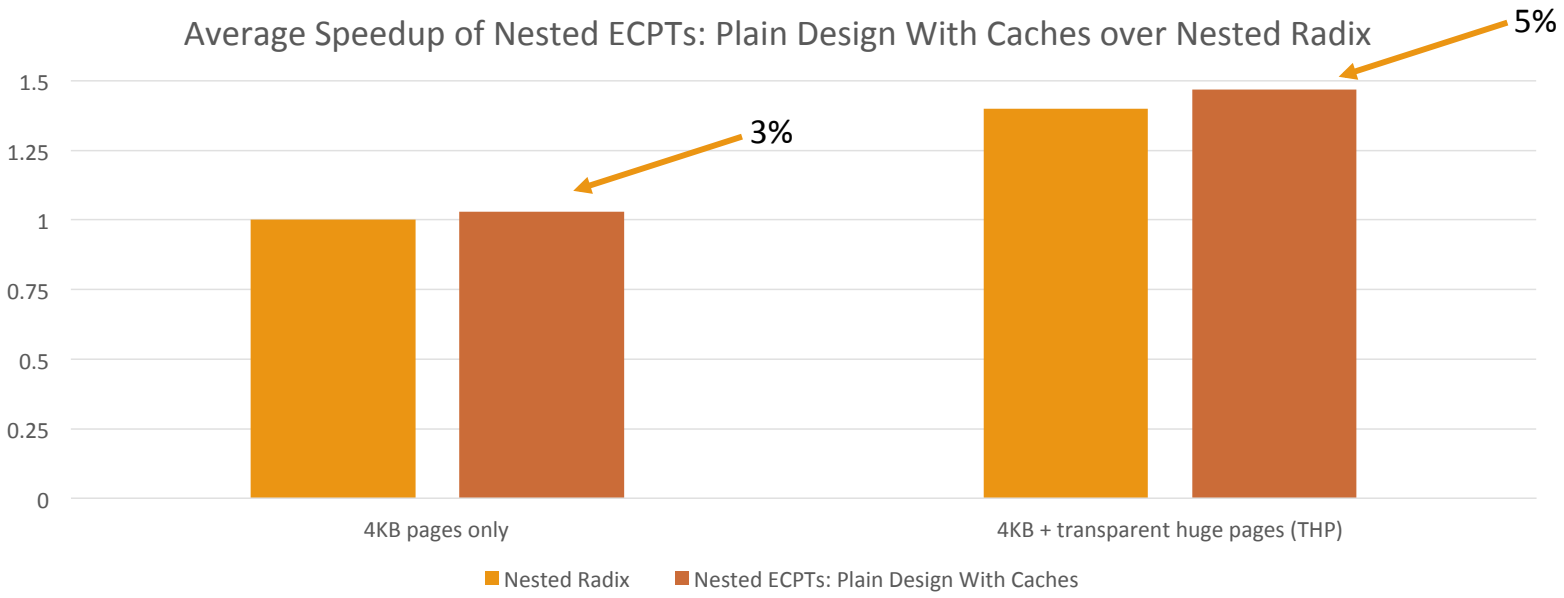
Cuckoo Walk Table (CWT): one per way and page size

Guest and Host Cuckoo Walk Tables (gCWT and hCWT)

gCWT and hCWT cached in the Cuckoo Walk Caches (gCWC and hCWC)



Performance Improvement



Minor performance improvement
Many parallel memory accesses

Need to analyze and redesign translation mechanisms

Proposal: Advanced Design for Nested ECPTs

To improve Nested ECPT performance need to redesign the translation mechanisms in the MMU

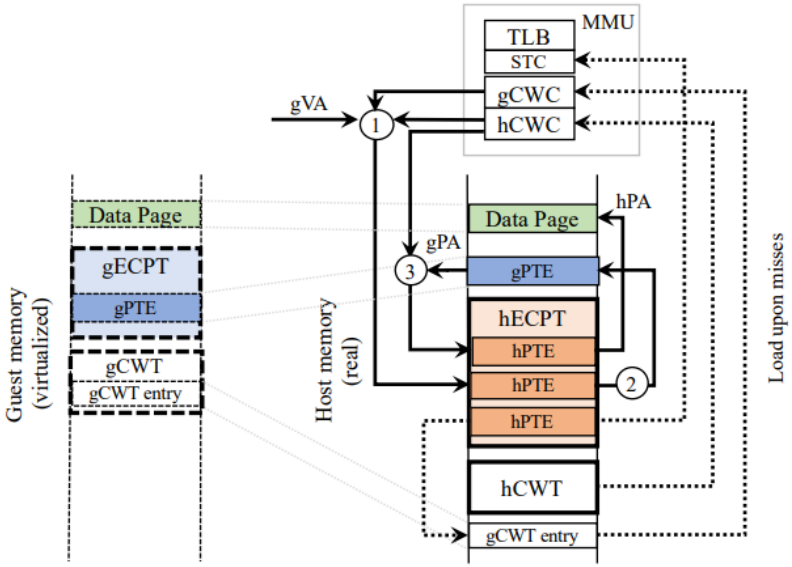
Goal is to minimize number of parallel memory requests

1. Shortcut Translation Cache (STC)

2. Caching some metadata (host PTE CWT), sometimes adaptively

3. Not caching some of the metadata

4. Page tables stored in 4KB pages



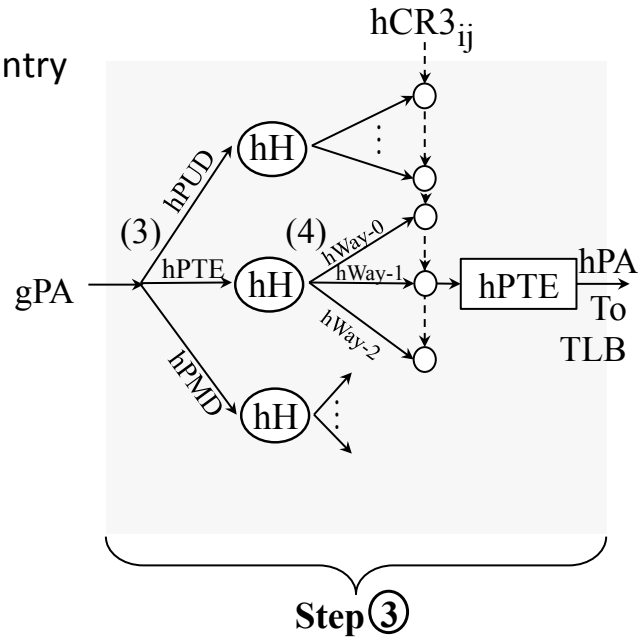
New: Shortcut Translation Cache (STC)

On a CWC miss hardware needs to fetch CWT entries in the background

- hCWC miss → Hardware can directly access hCWT

- gCWC miss → Need to translate gPA of gCWT entry to hPA of gCWT entry
  - Operations and memory traffic in the background
  - Hurts performance

Proposed solution:
- Cache gPA to hPA translations of gCWT entries in a small cache
- Called Shortcut Translation Cache (STC)
- Conceptually similar to the NTLB in Nested Radix page tables



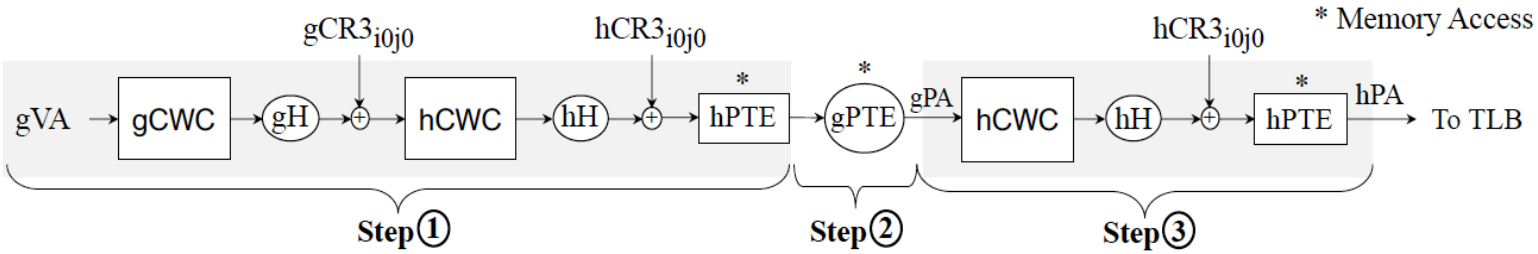
Caching PTE hCWT Entries

PTE guest CWT exhibits poor locality → thus, no caching

PTE host CWT has more locality → Opportunity to cache it in hCWC

Step 1: translate gPA to hPA of gECPT entries
- Small size of gECPT and large coverage per CWC entry
- Always cache PTE hCWT

Step 3: translate gPA to hPA of data pages
- Locality is application dependent
- Adaptively cache PTE hCWT by monitoring hit rates hCWCs



Leverage Page Size used by Page Tables

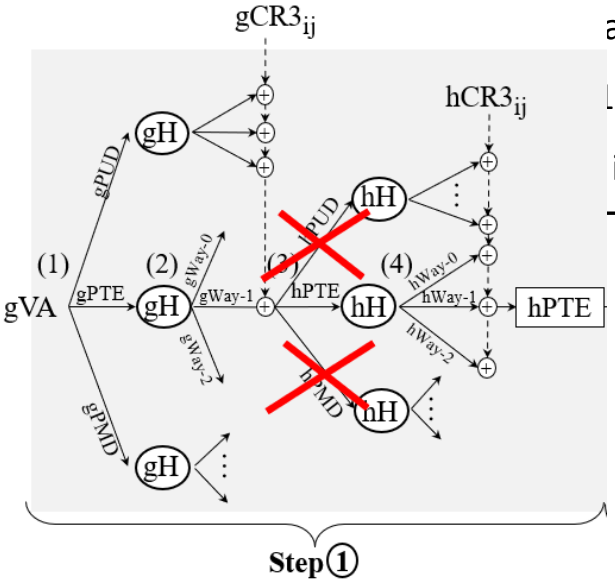
By knowing page size used I able walk

Used to trim number of par

KVM (host) and Linux (guest in 4KB pages
- Page tables are usually sma → avoid internal fragmentation
- 4KB pages are more flexible
- Legacy reasons

Guest page table always stc

→ No need to issue request



Helps with the tail latency during application warm up

Migration Path: Hybrid Design

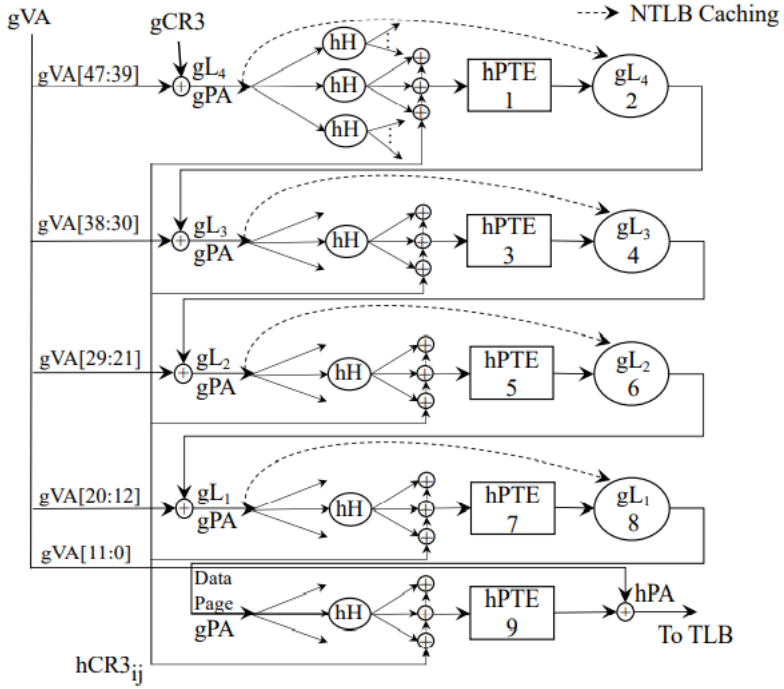
Nested ECPTs radical change

Hybrid design

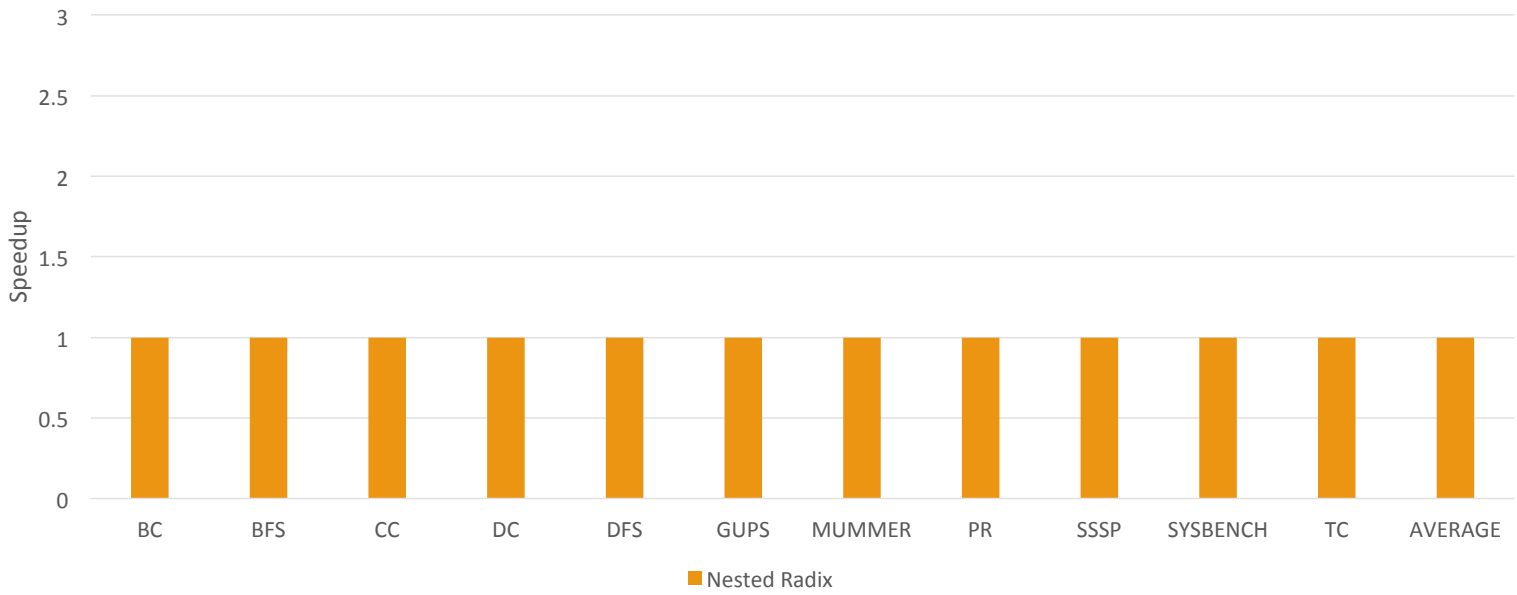
- Guest OS unmodified → Radix Page Tables
- Hypervisor modified → ECPTs

Advantages

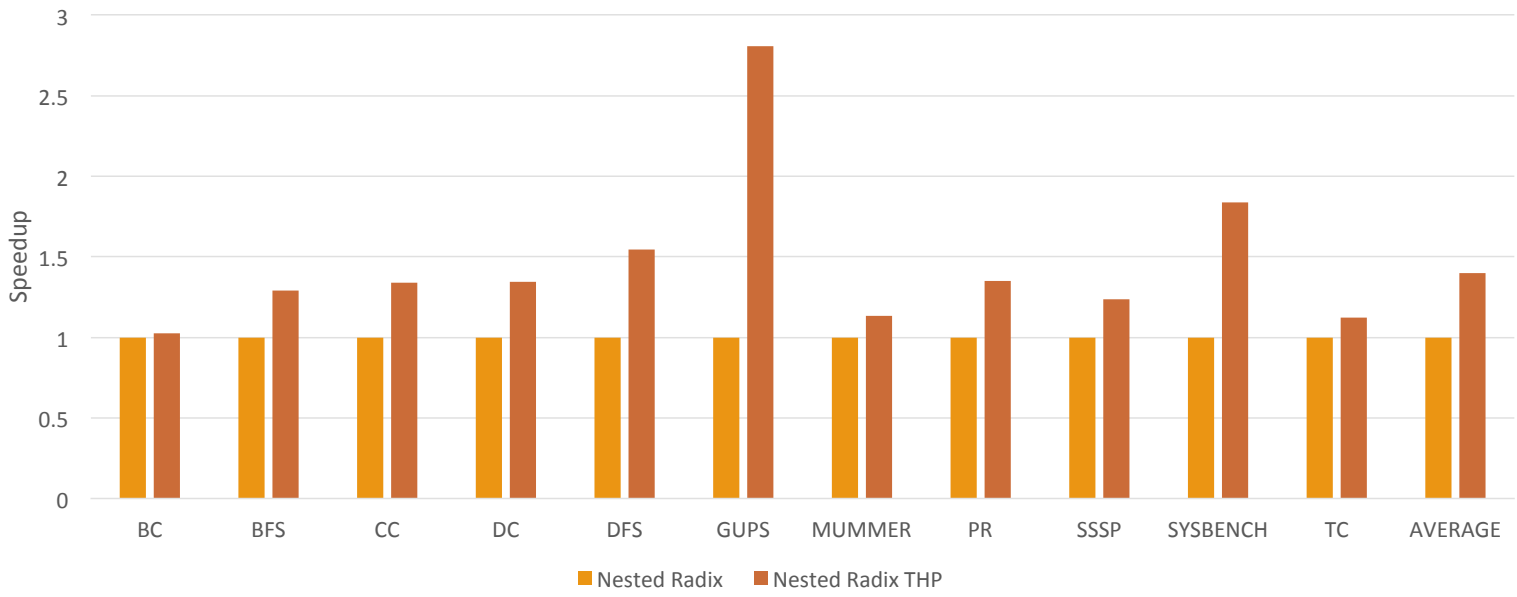
- Legacy guest OS
- Hypervisor tuned for high-performance



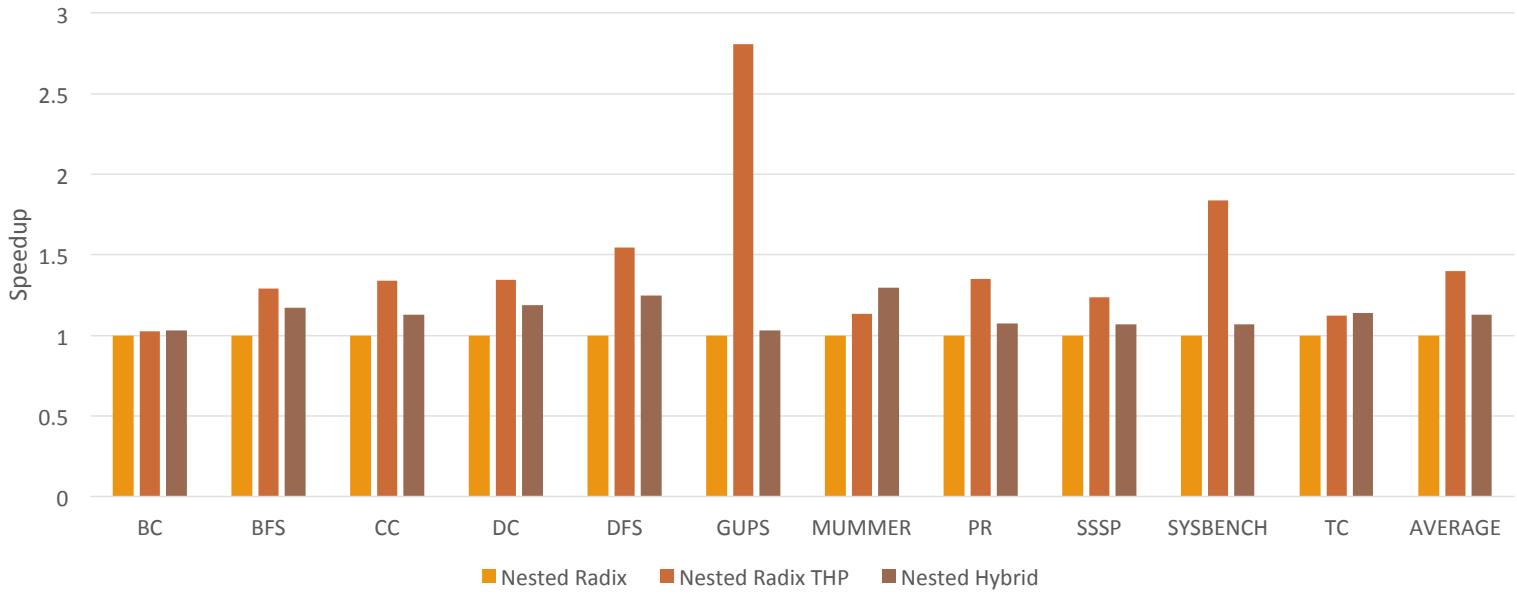
Evaluation Results: Application Speedup



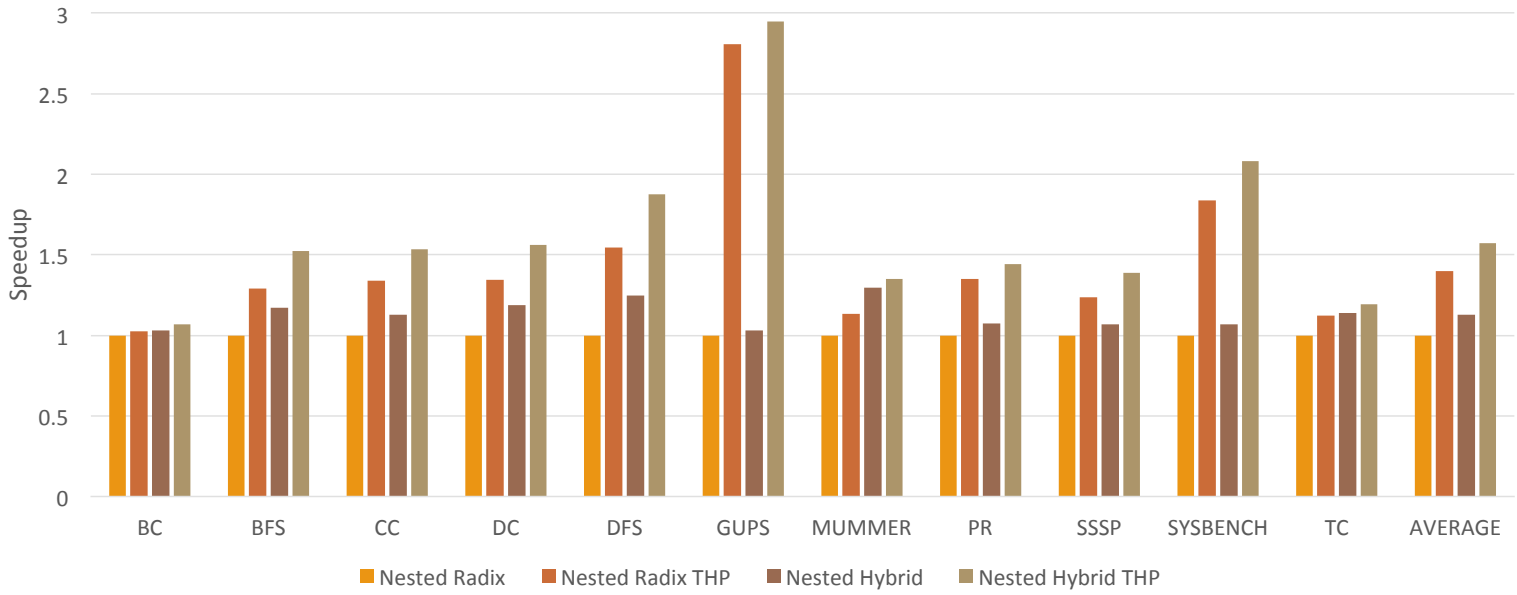
Evaluation Results: Application Speedup



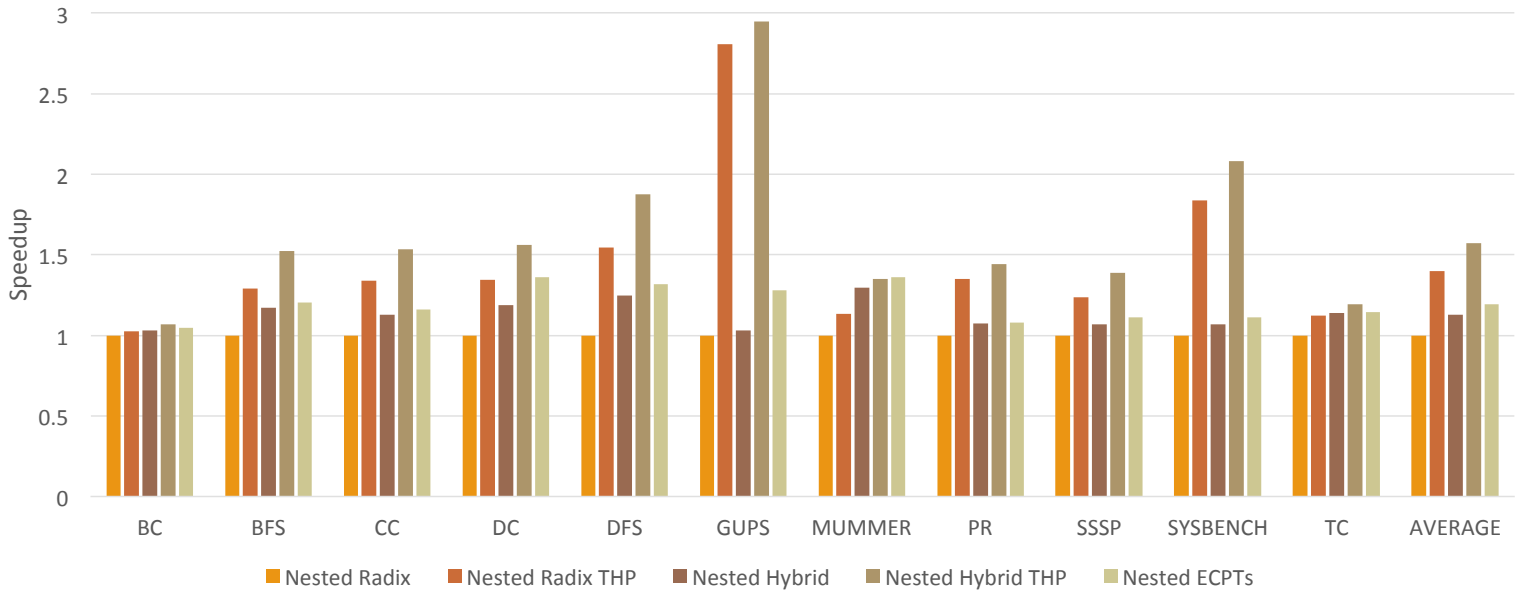
Evaluation Results: Application Speedup



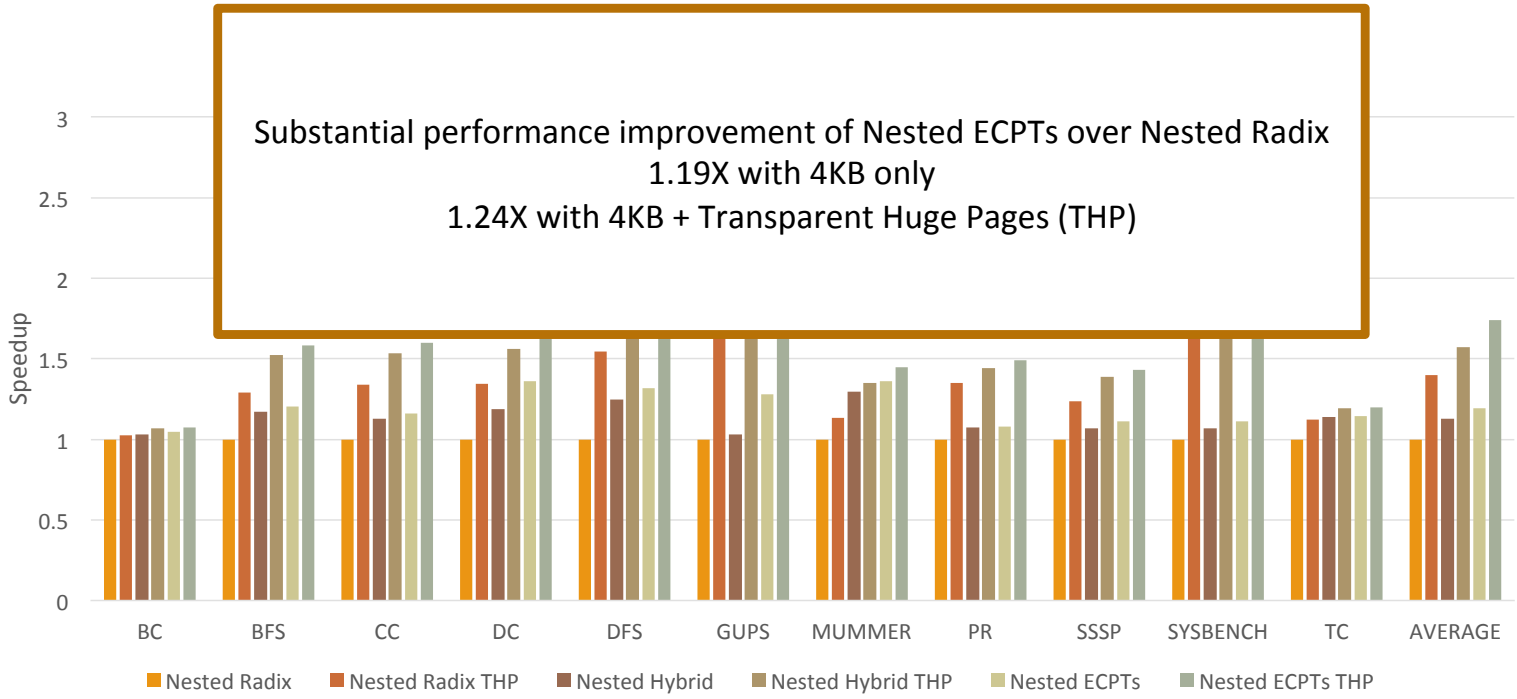
Evaluation Results: Application Speedup



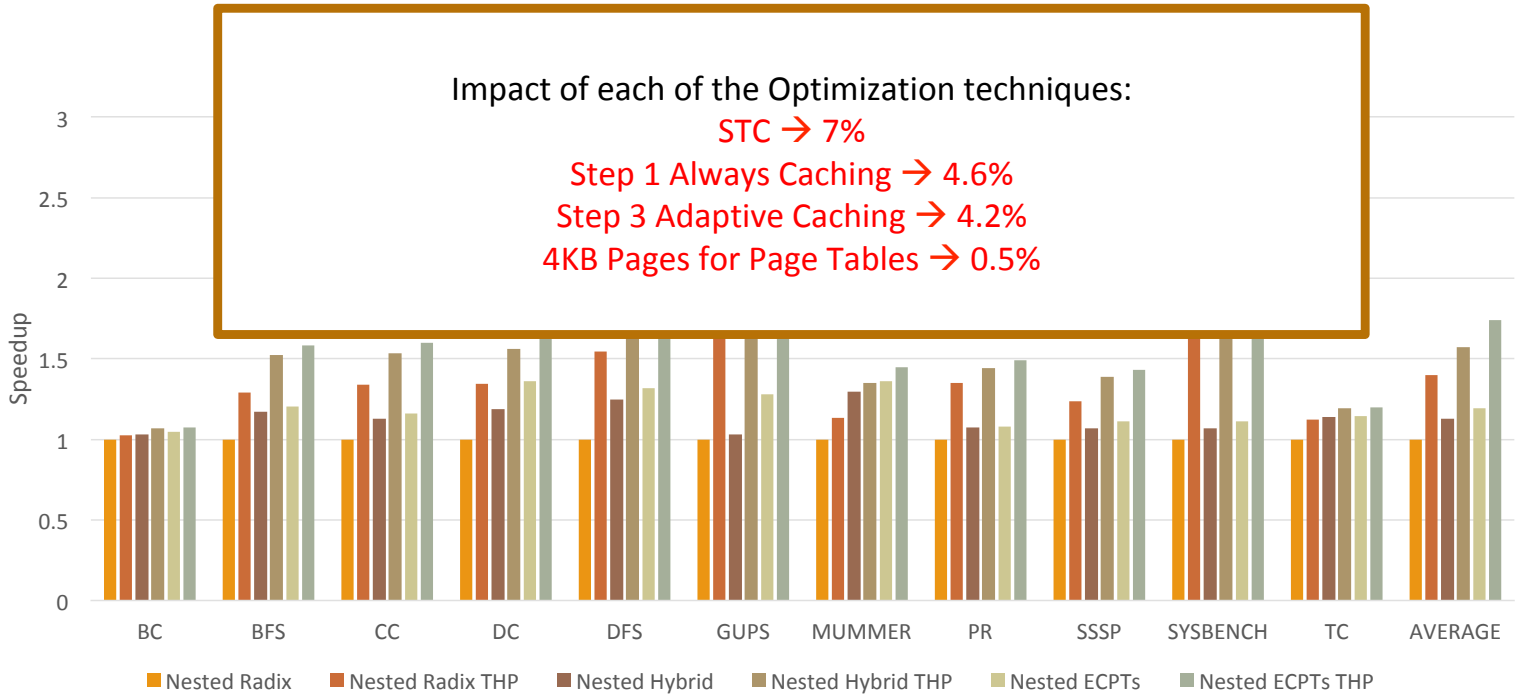
Evaluation Results: Application Speedup



Evaluation Results: Application Speedup



Evaluation Results: Application Speedup



More in the Paper

OS/Hypervisor support requirements

Evaluation results

- MMU and Cache characterization
- Characterization of Nested ECPT walks
- Memory Consumption and Hardware cost

Comparison to other advanced designs

- Agile Paging
- POM-TLB
- Flat Nested Page Tables

Conclusion

The first page table design for parallel nested address translation: Nested ECPTs

- Eliminate all but three of the potentially 24 sequential memory accesses

Judiciously limits the number of parallel memory accesses issued

- Shortcut Translation Cache
- Caching some metadata (host PTE CWT), sometimes adaptively
- Page tables always stored in 4KB pages

Nested ECPTs outperform state-of-the-art Nested Radix tables

- Avg. 1.19X for 4KB pages
- Avg. 1.24X for 4KB + 2MB pages

Possible migration path from Nested Radix to Nested ECPTs

Parallel Virtualized Memory Translation with Nested Elastic Cuckoo Page Tables

Jovan Stojkovic, Dimitrios Skarlatos‡, Apostolos Kokolis, Tianyin Xu, Josep Torrellas

jovans2@illinois.edu

University of Illinois at Urbana-Champaign ‡Carnegie Mellon university

ASPLOS, March 2022

i-acoma group

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Carnegie Mellon University Computer Science Department

Backup Slides

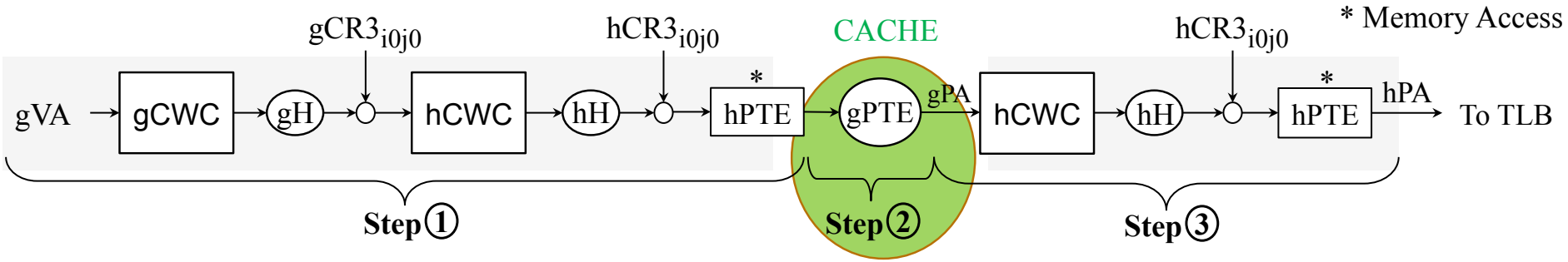
Caching gECPT-to-hECPT translations?

Due to its importance applied to both Plain and Advanced designs

NTLB in Nested Radix caches address translation of a level of the guest page table

In Nested ECPTs parallel to NTLB would be caching hECPT to gECPT translations in Step 2

- Could eliminate one of the three sequential steps of the translation process



Avoid Stale hECPT entries

hPA of gPTE changes often

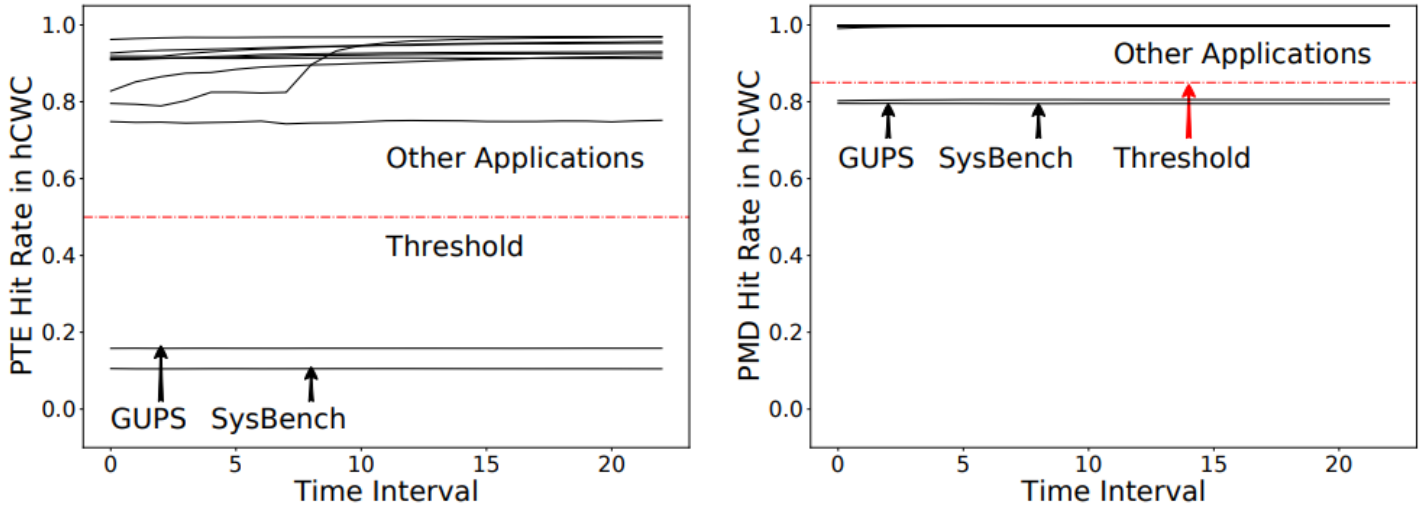
- Due to cuckoo rehashing, inserting an entry may cause shuffling of existing entries
- Due to dynamic resizing of a gECPT, entries migrate from old to new gECPT

On a change of hPA of a gPTE

→ hPTE that maintained the original pointer to the gPTE becomes stale

To avoid flushing such translations, neither the Plain nor the Advanced Nested ECPT design caches the mapping of hPTEs-to-gPTEs in Step 2

Adaptive Caching Monitoring



Area and Power Comparison

Table 3: Area and power of the hardware caches in the MMU.

Configuration	Size (B)	Area (mm^2)	Power (mW)
Nested Radix	1680	0.01	2.9
Nested ECPTs	1488	0.03	5.2
Nested Hybrid	1408	0.02	2.8

MMU Busy Cycles

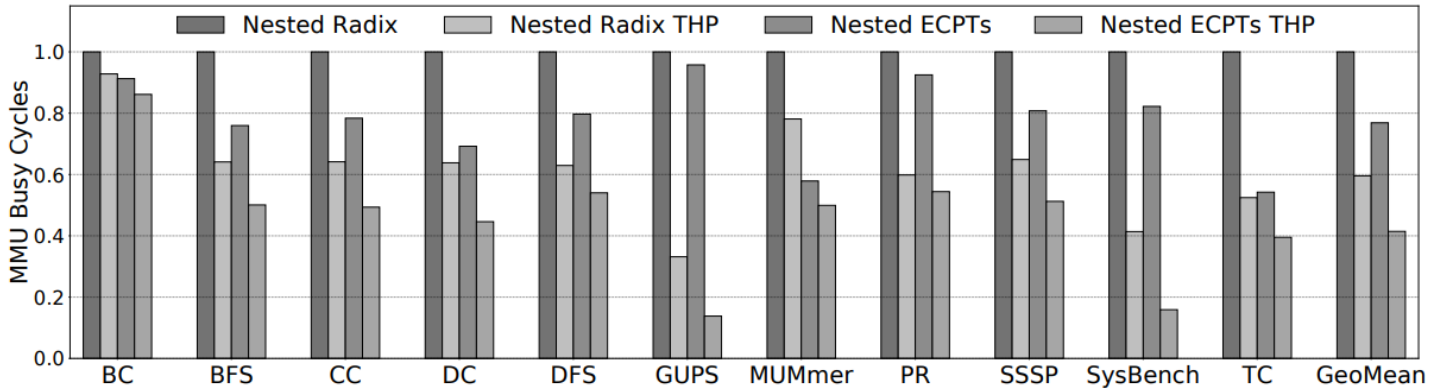


Figure 10: MMU busy cycles in nested configurations.

Cache Characterization

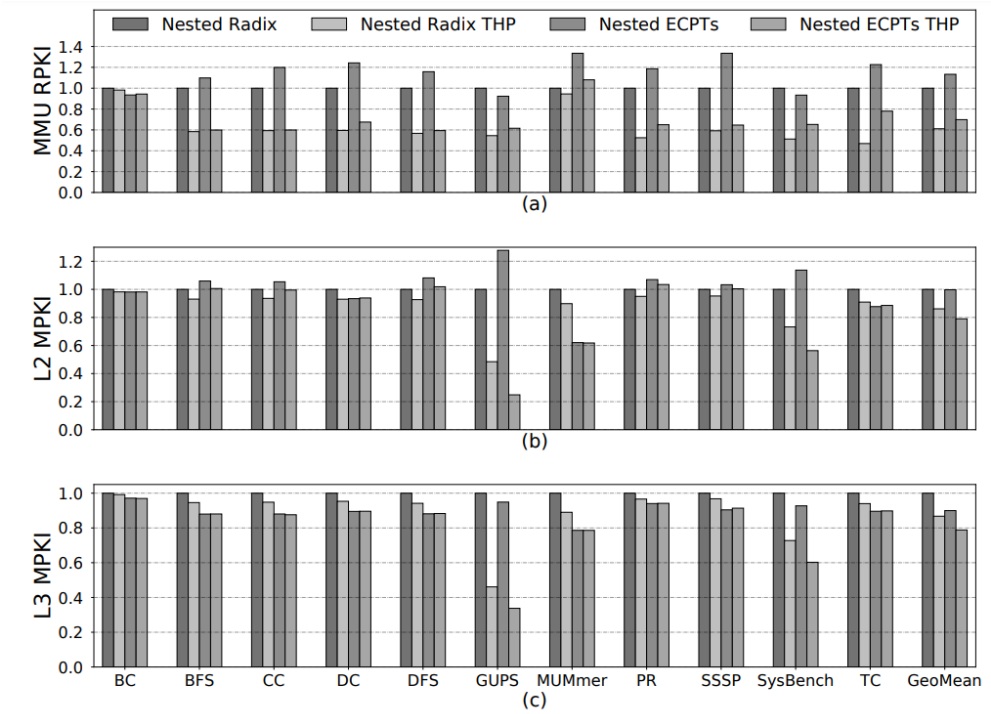


Figure 13: Characterizing the MMU and cache subsystem.

Walk-Type Distribution: guest and host

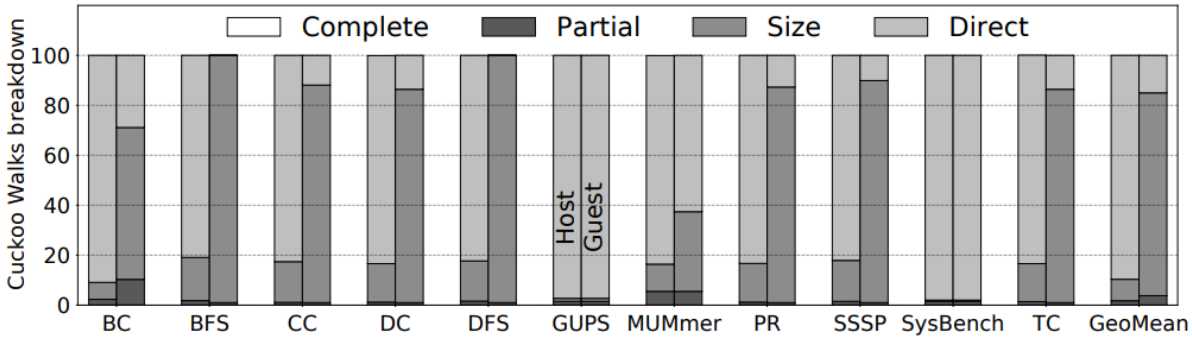


Figure 14: Breakdown of the types of host (left bar) and guest walks (right bar) for each application in Nested ECPTs THP.

Latency Histogram

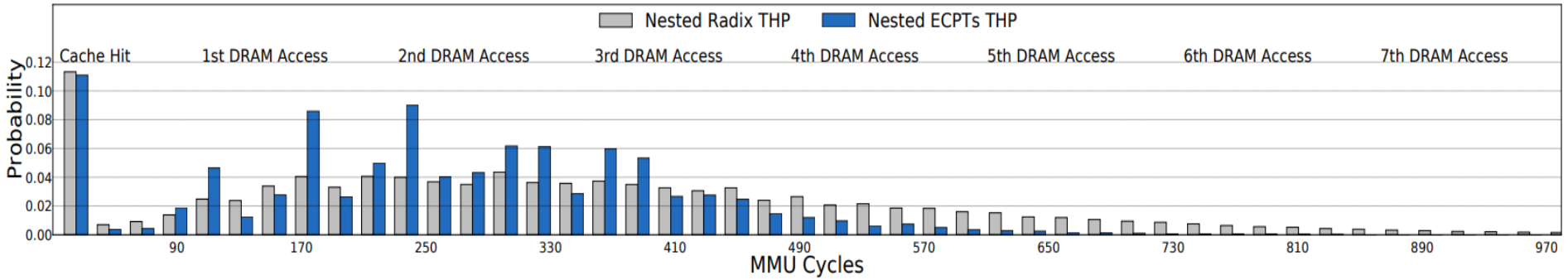


Figure 11: Histogram of the latency of the nested page walks in the MUMmer application.

Memory Consumption

On average 80MB needed (application data size * 8B page table entry size)

Nested Radix 84MB

- 56MB host
- 28MB guest

Nested ECPTs 97MB

- 61MB host
- 36MB guest

Comparison to Advanced Designs

Agile Paging – combines nested and shadow paging

- Idea: levels of upper-level page tables are unlikely to be changed
- Need 4 sequential requests at best case + hypervisor intervention cost
- We model Ideal Agile Paging: 4 sequential memory requests at most + caching structure + no host cost
- Nested ECPTs outperform Agile Paging by 16% on average

POM-TLB – large in-memory TLB

- Eliminates many page table walks
- L2 TLB miss needs to go to DRAM and can still miss there
- We model POM-TLB with perfect page size predictor
- Nested ECPTs outperform POM-TLB by 14% on average

Flat Nested Page Tables – combine guest radix page table with a host flat page table

- Reduces number of sequential memory accesses from 24 to 9
- Nested ECPTs outperform Flat Nested Page Tables by 12% (no THP) and by 15% (with THP)