Reducing Memory Virtualization Overheads in Virtualized Datacenters

Jayneel Gandhi, Arkaprava Basu,
Michael M. Swift, Mark D. Hill

Executive Summary

- Problem: TLB misses in virtual machines
  - Hardware-virtualized MMU has high overheads
  - ➢ Up to 280% overhead
- Prior Work: **Direct Segments** – unvirtualized case
- Solution: segmentation to bypass paging
  - **Extend Direct Segments** for virtualization
  - **Three configurations** with different tradeoffs
- Results
  - **Near-** or **better-than**-native performance

Outline

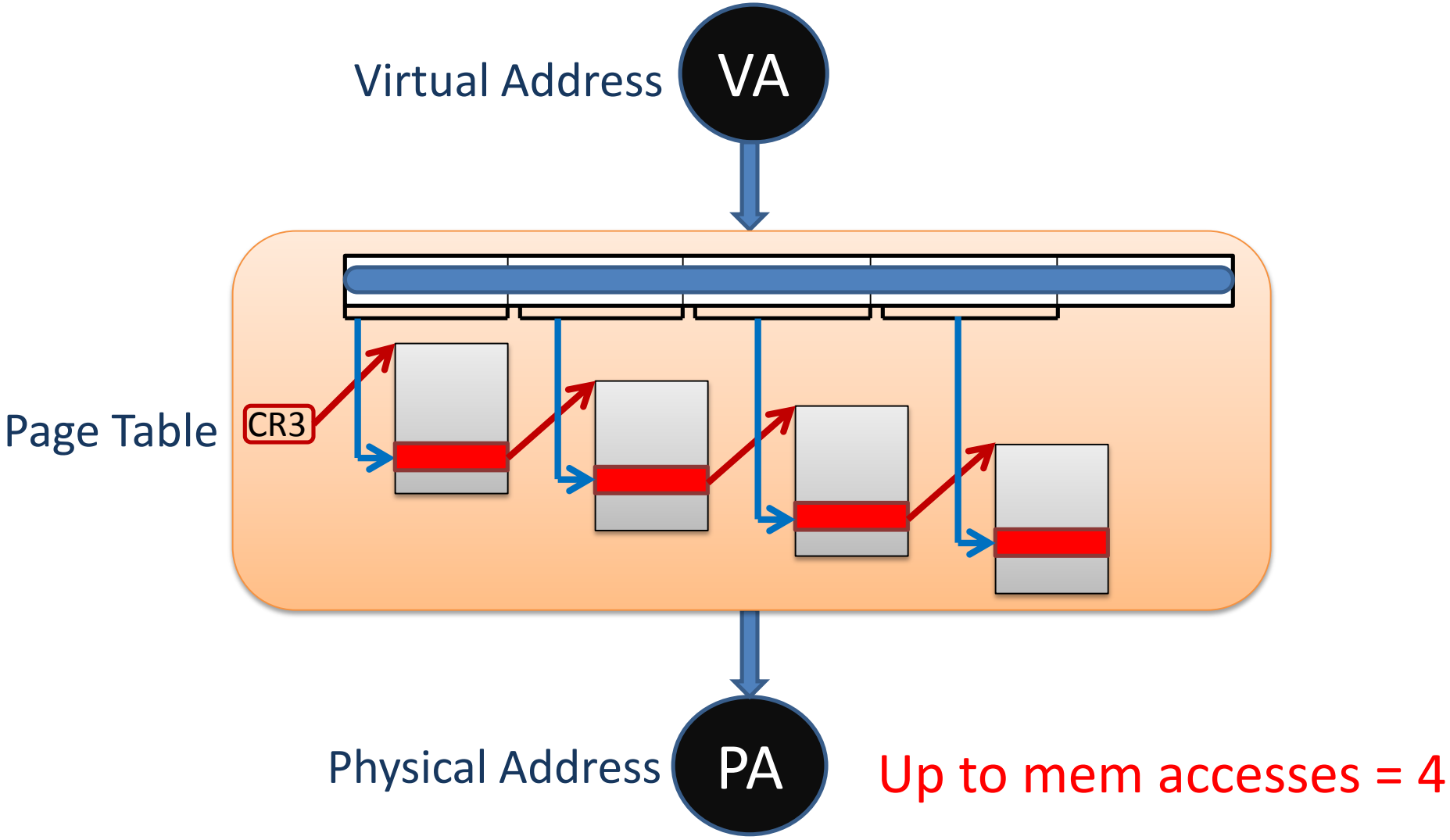
- Motivation ←
- Review: Direct Segments
- Virtualized Direct Segments
- Optimizations
- Evaluation
  - Methodology
  - Results
- Summary

Overheads of Virtualizing Memory

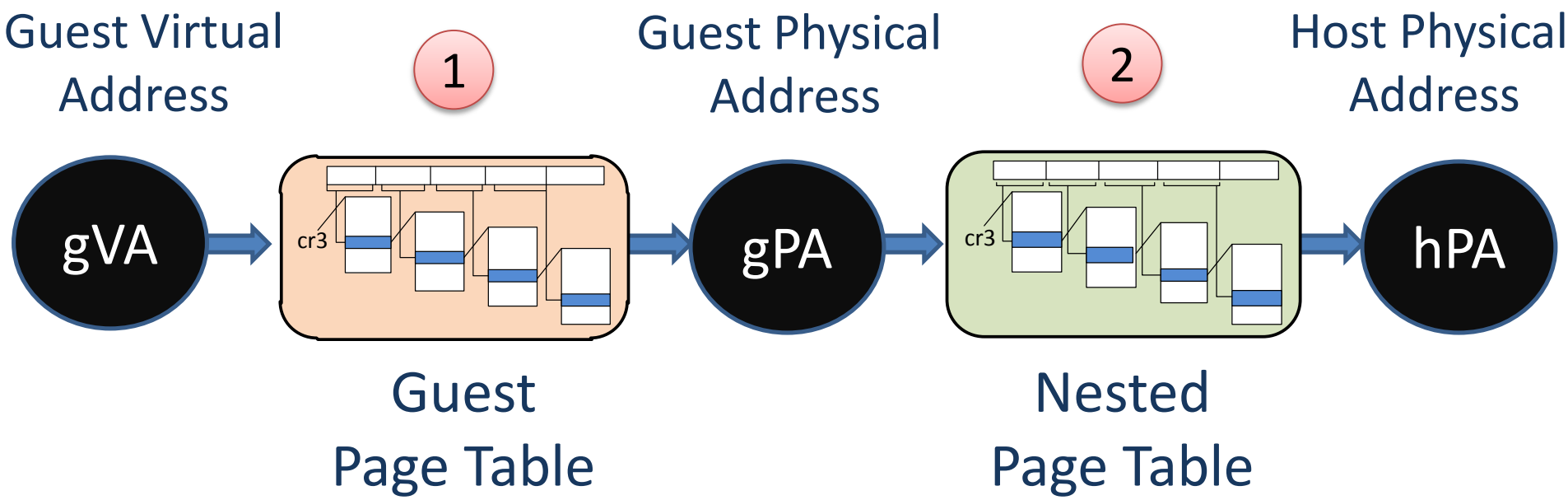
We show that that the increase in translation lookaside buffer (TLB) miss-handling costs due to hard-ware-assisted memory management unit (MMU) is the largest contributor to the performance gap between native and virtual servers.

—Buell, et al. VMware Technical Journal 2013

Unvirtualized x86 translation

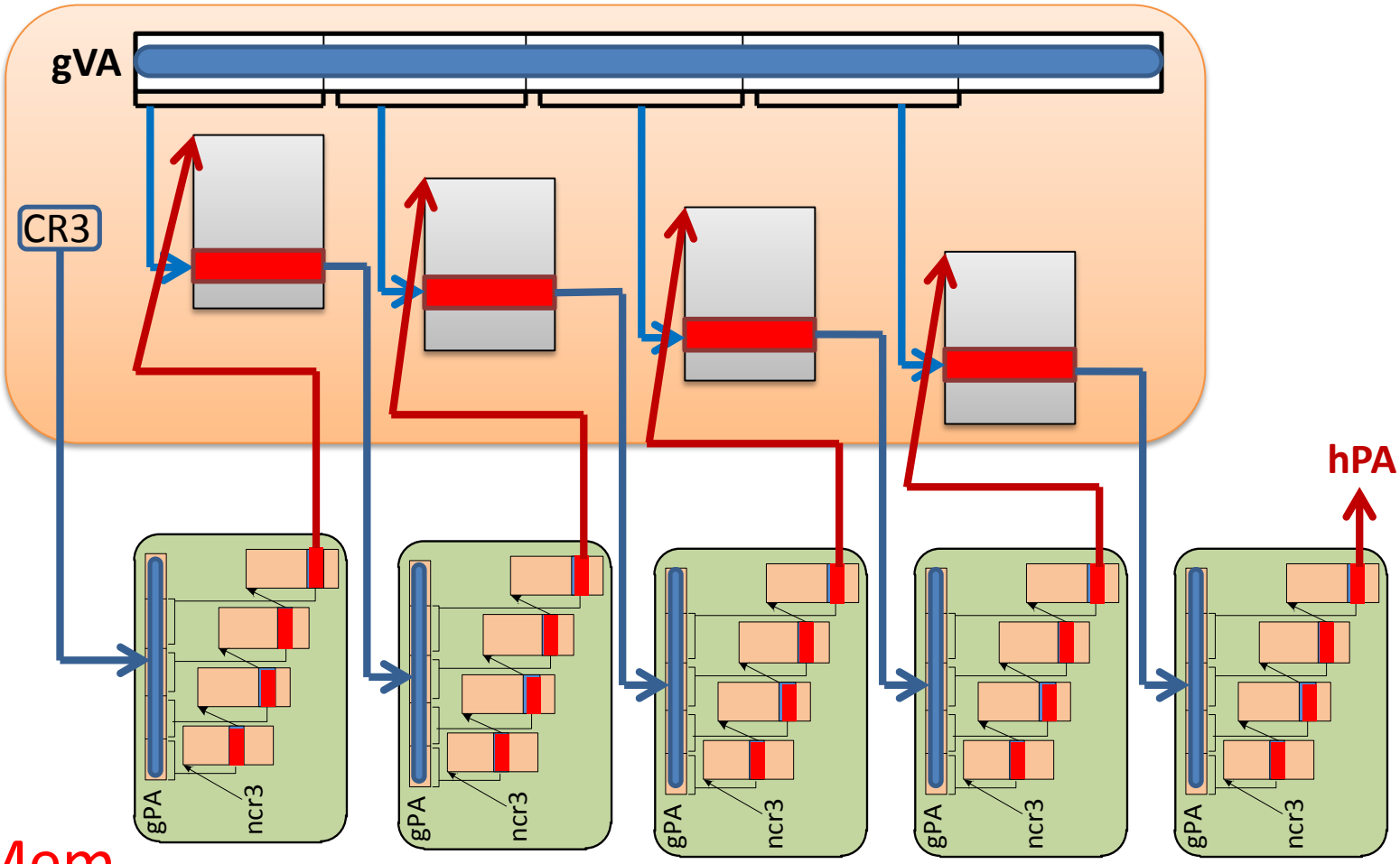


Two-levels of translation



Base Virtualized

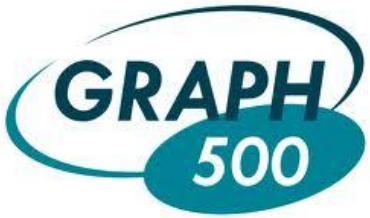
Support for Virtualizing Memory



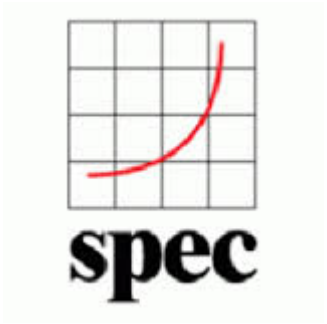
Up to Mem accesses

5 + 5 + 5 + 5 + 4 = **24**

Applications



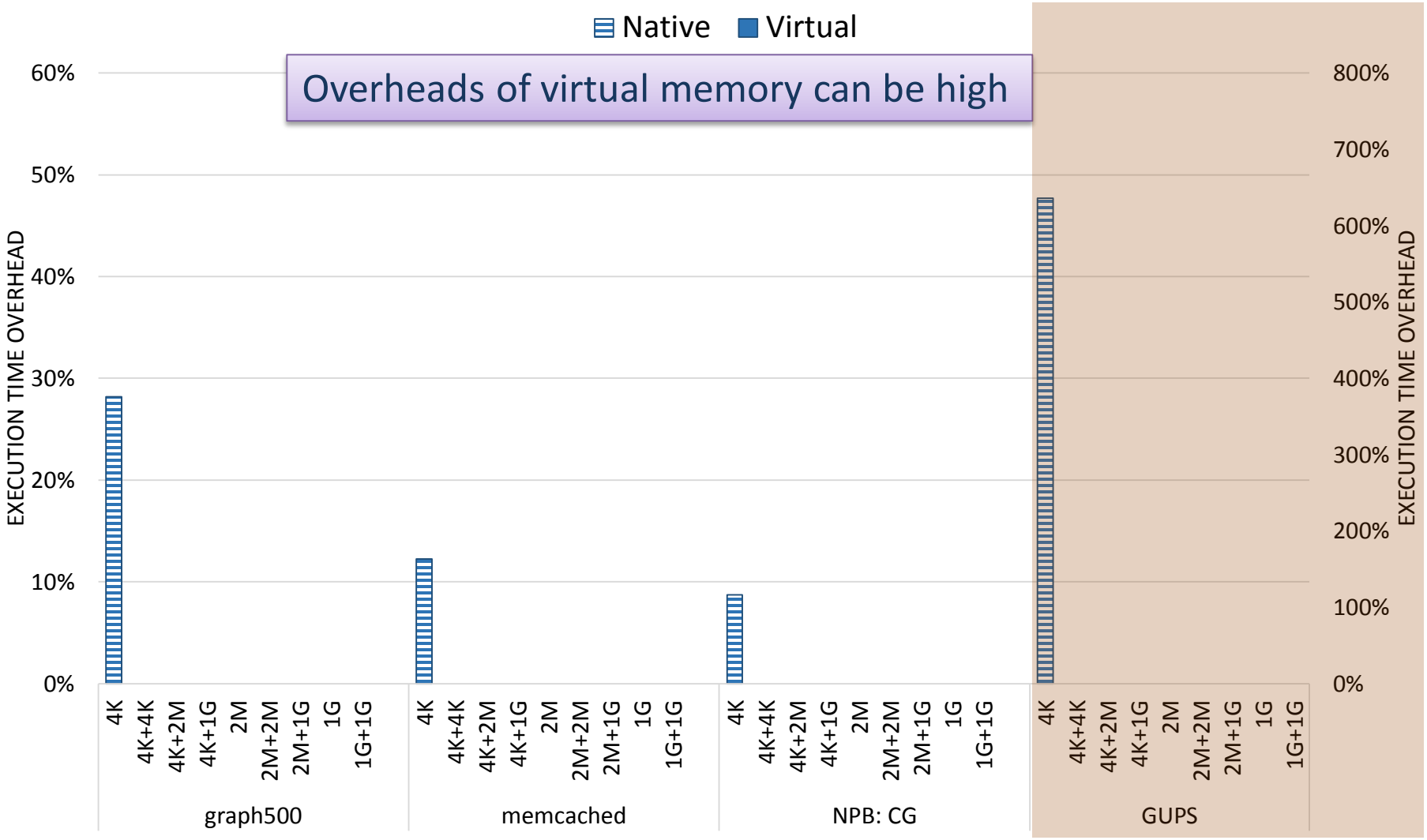
Database Graph-analytics Key-value store HPC Apps



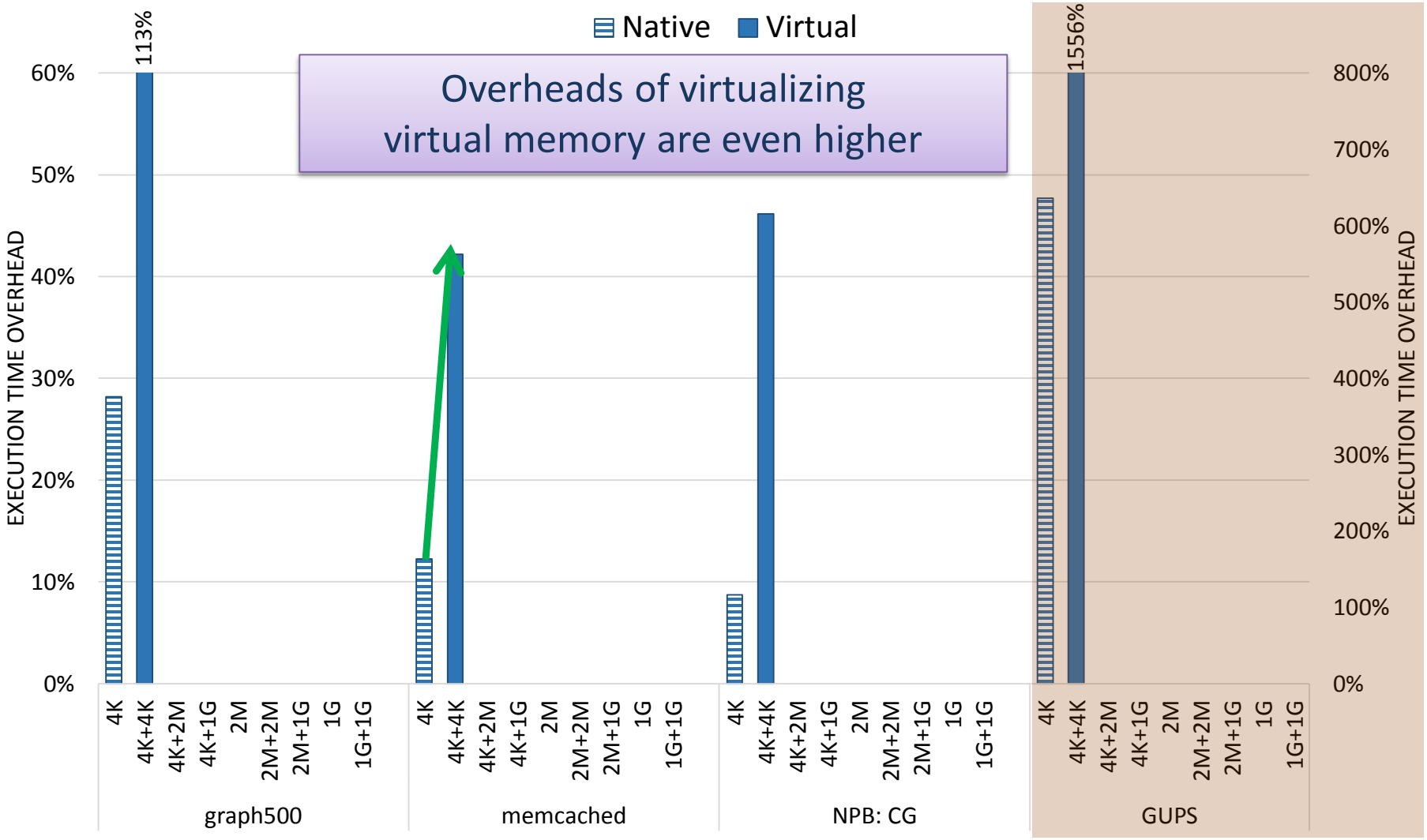
Standard Workloads

Big-memory applications
(in-memory apps)

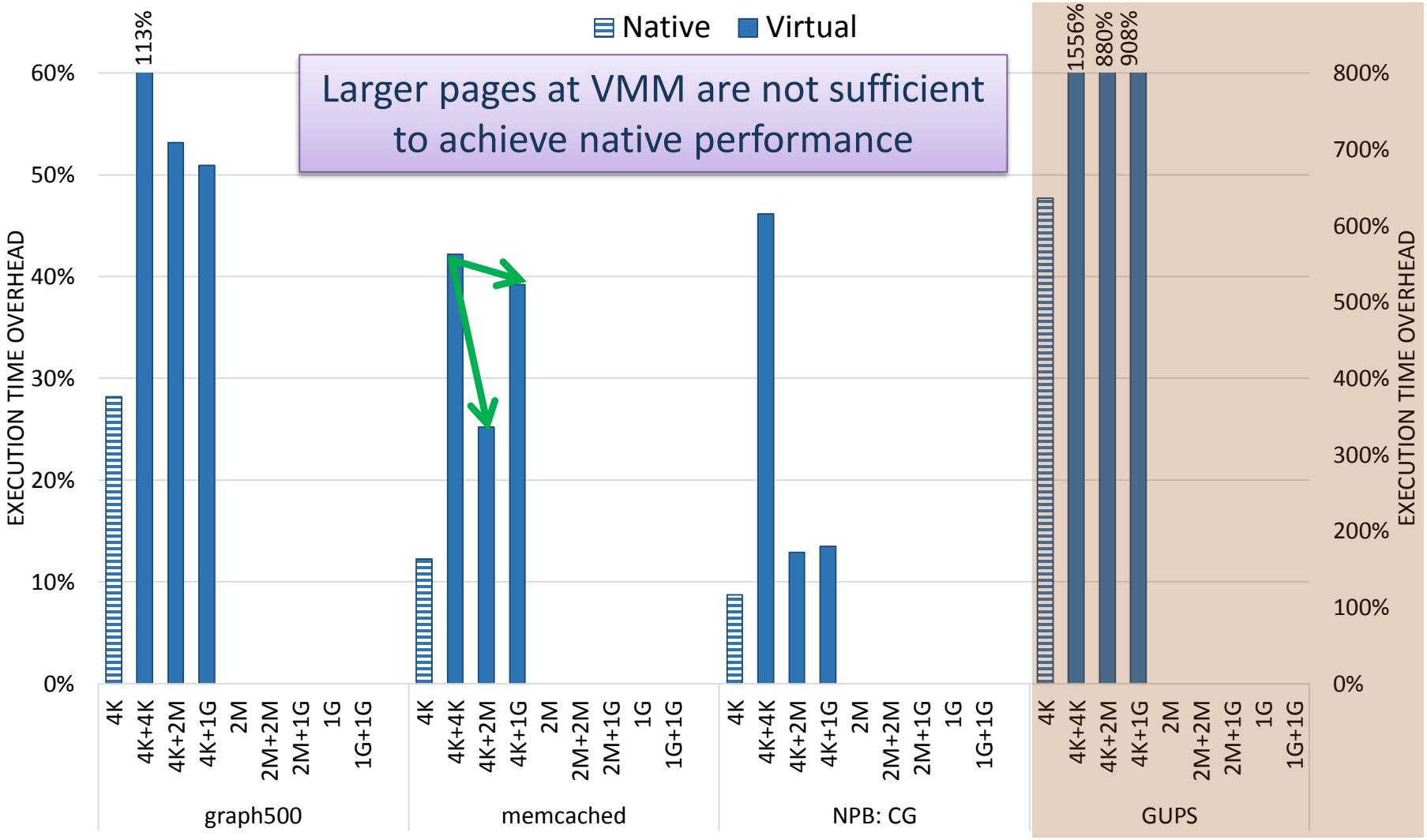
Cost of Virtualization



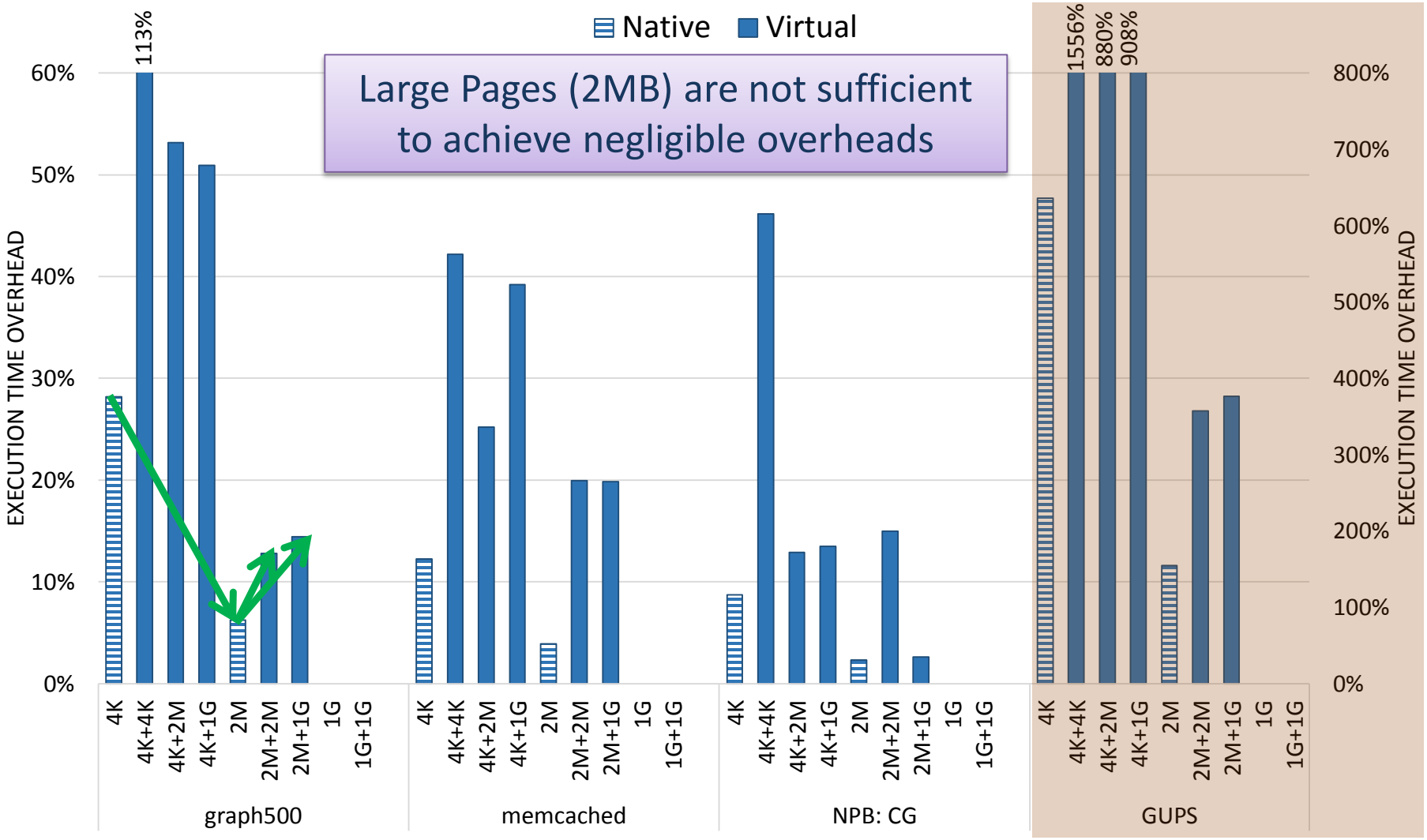
Cost of Virtualization



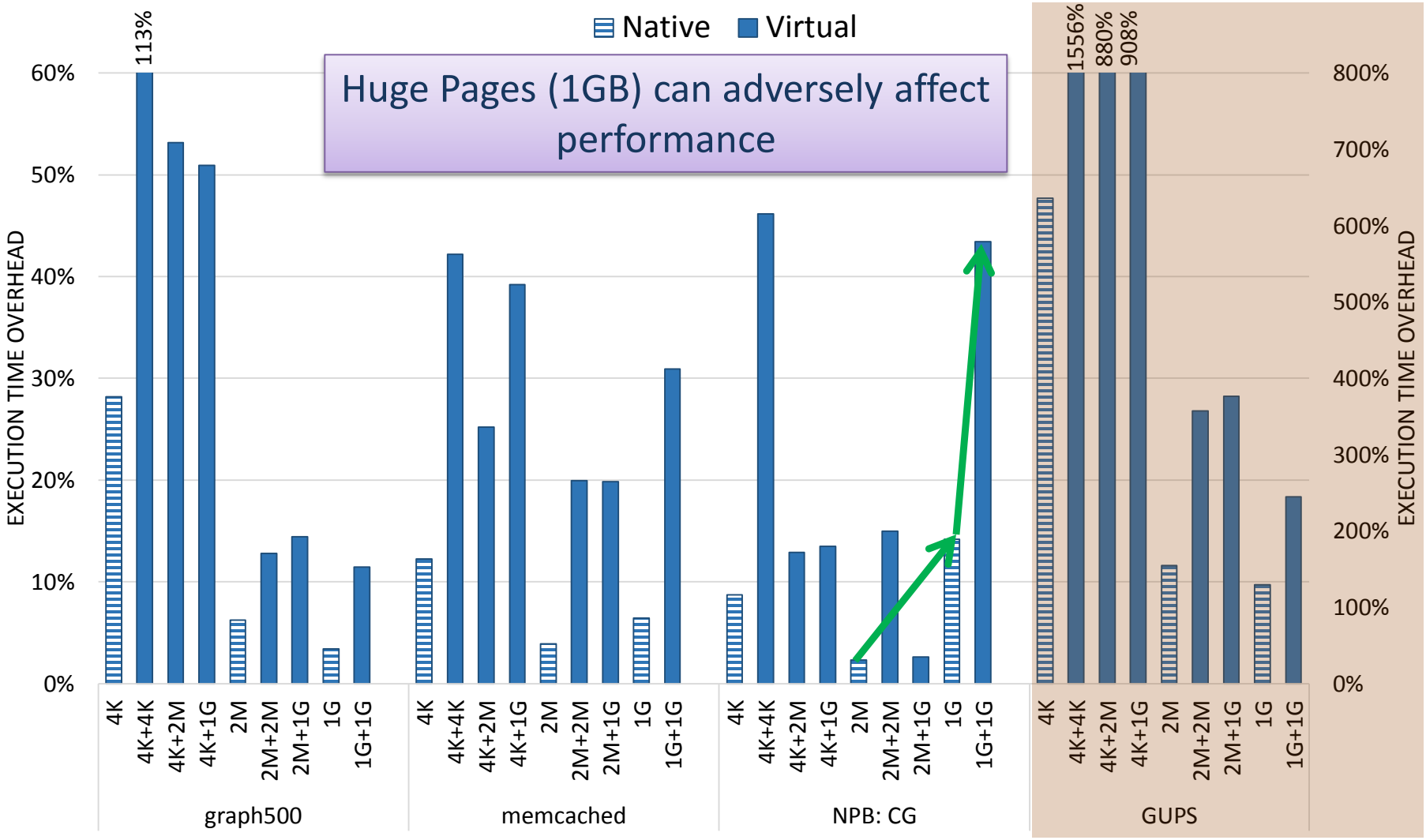
Cost of Virtualization



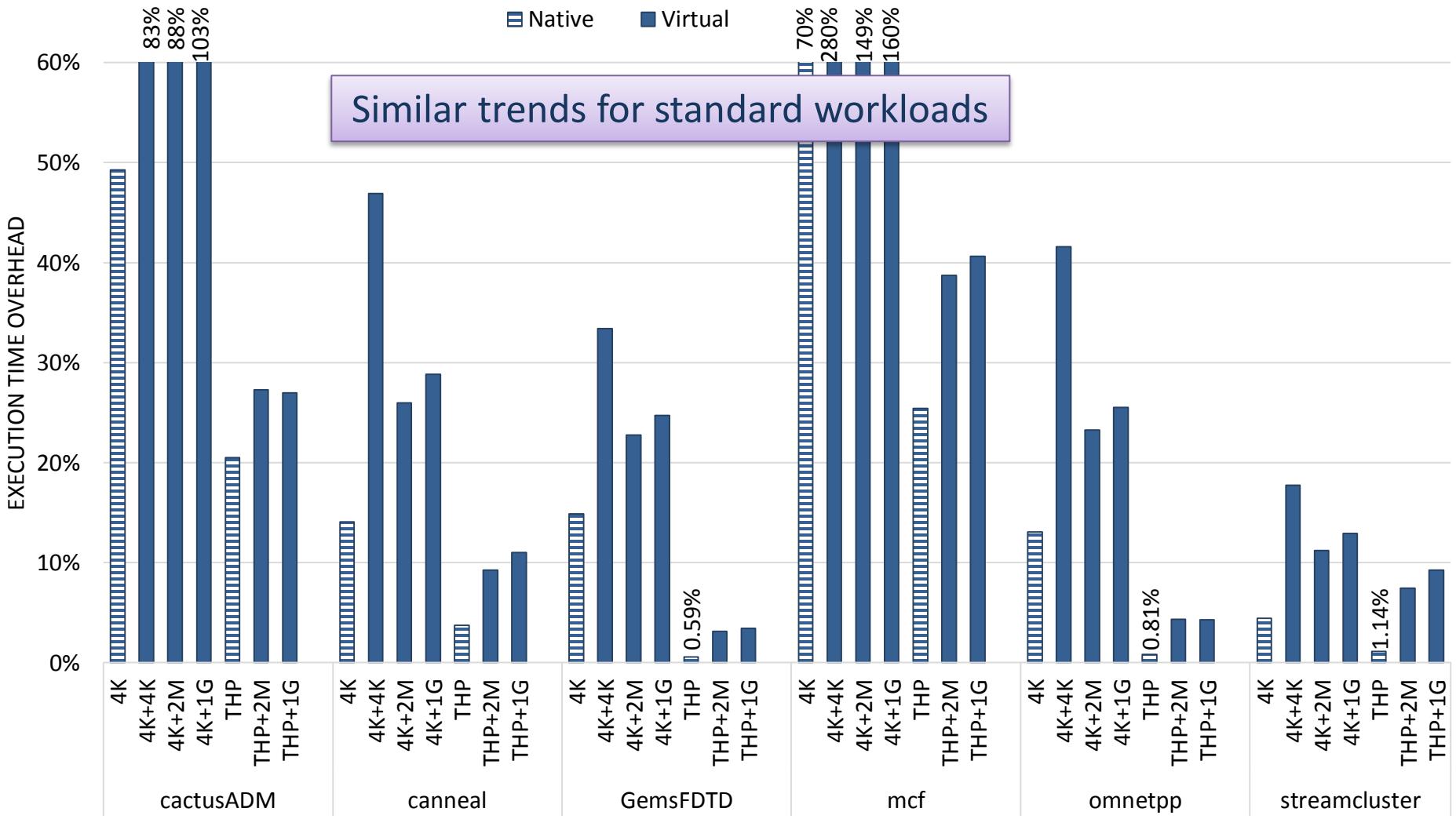
Cost of Virtualization



Cost of Virtualization



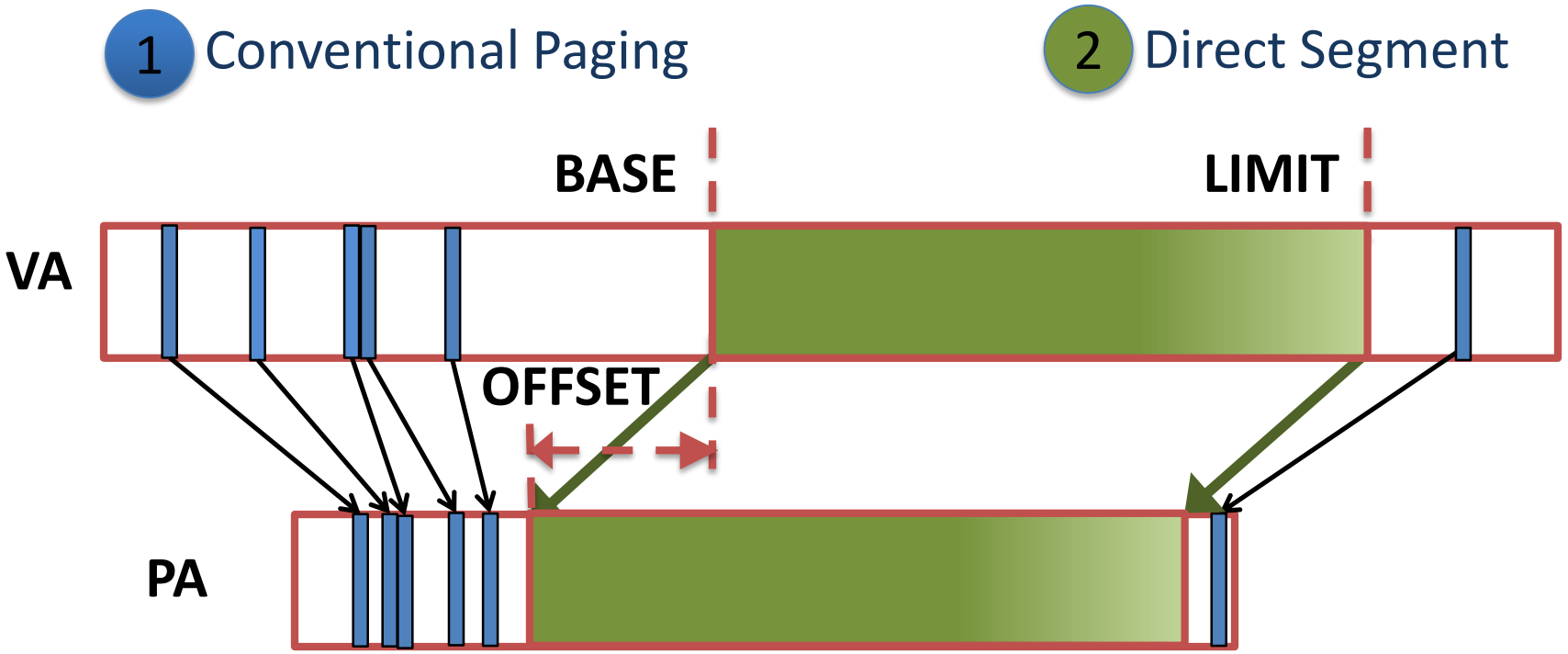
Cost of Virtualization



Outline

- Motivation
- Review: Direct Segments ←
- Virtualized Direct Segments
- Optimizations
- Evaluation
  - Methodology
  - Results
- Summary

Unvirtualized Direct Segments

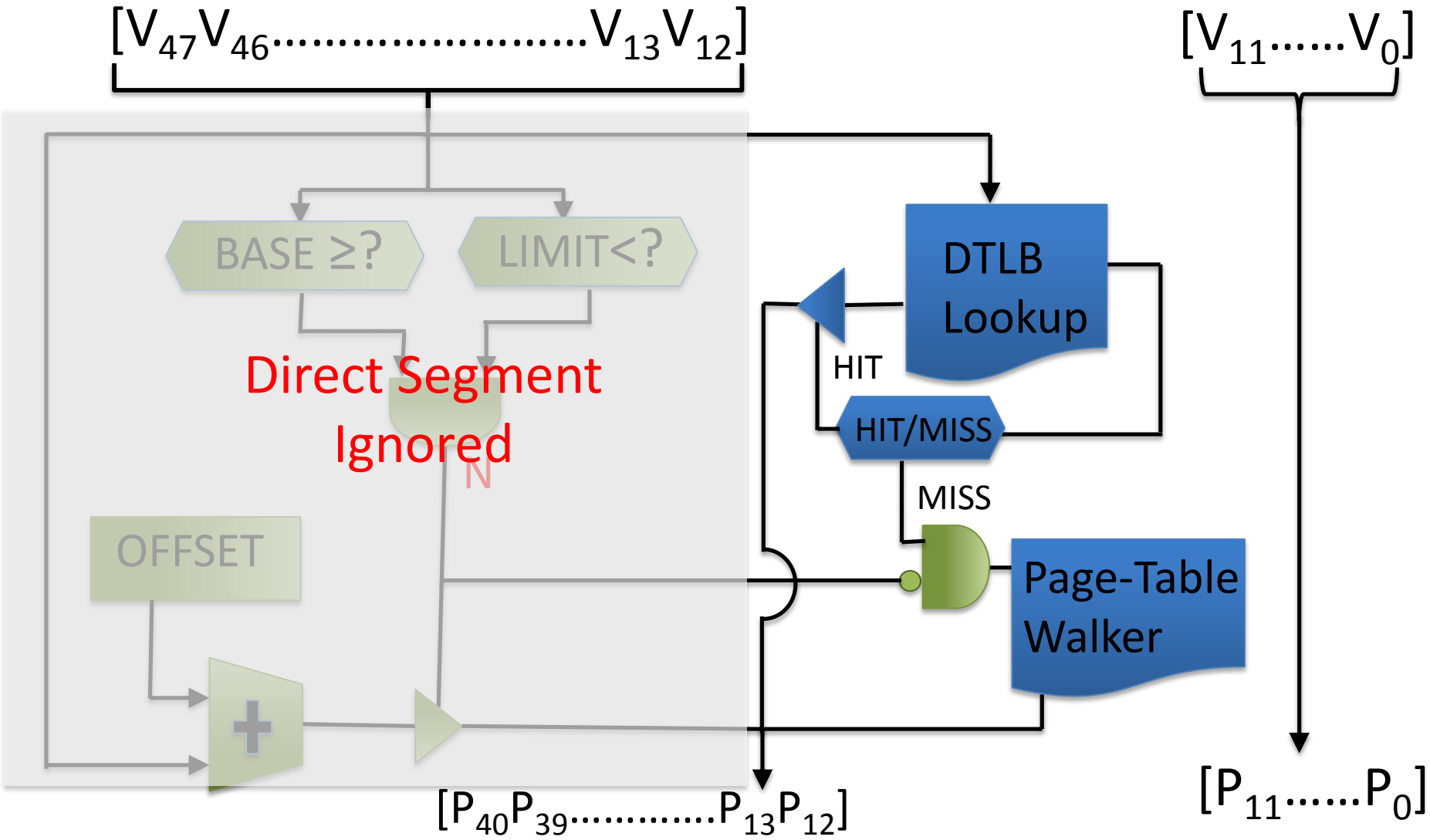


Why Direct Segment?

- Matches big memory workload needs
- NO TLB lookups => NO TLB Misses

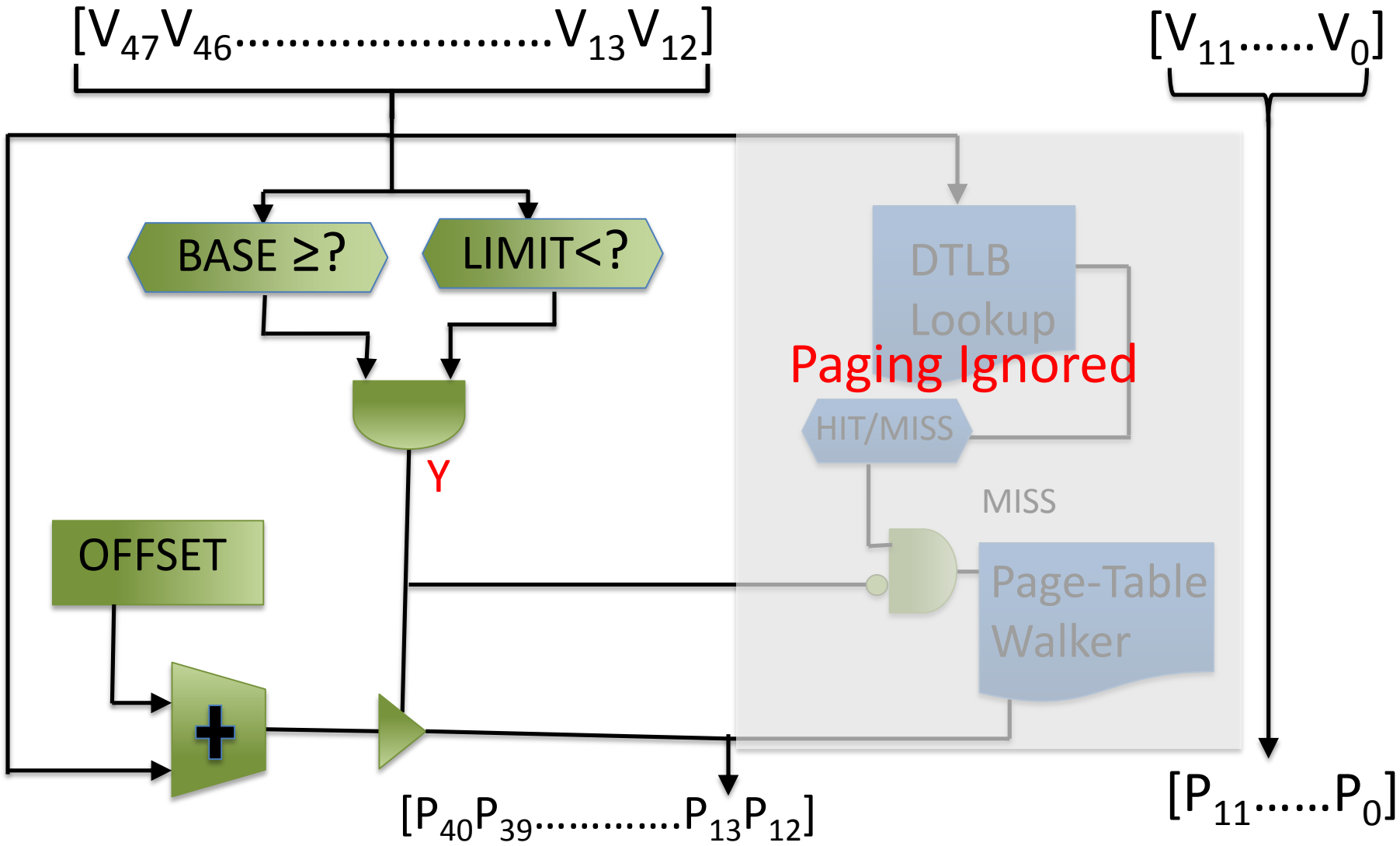
Basu et al. [ISCA 2013]

Translation with Direct Segments



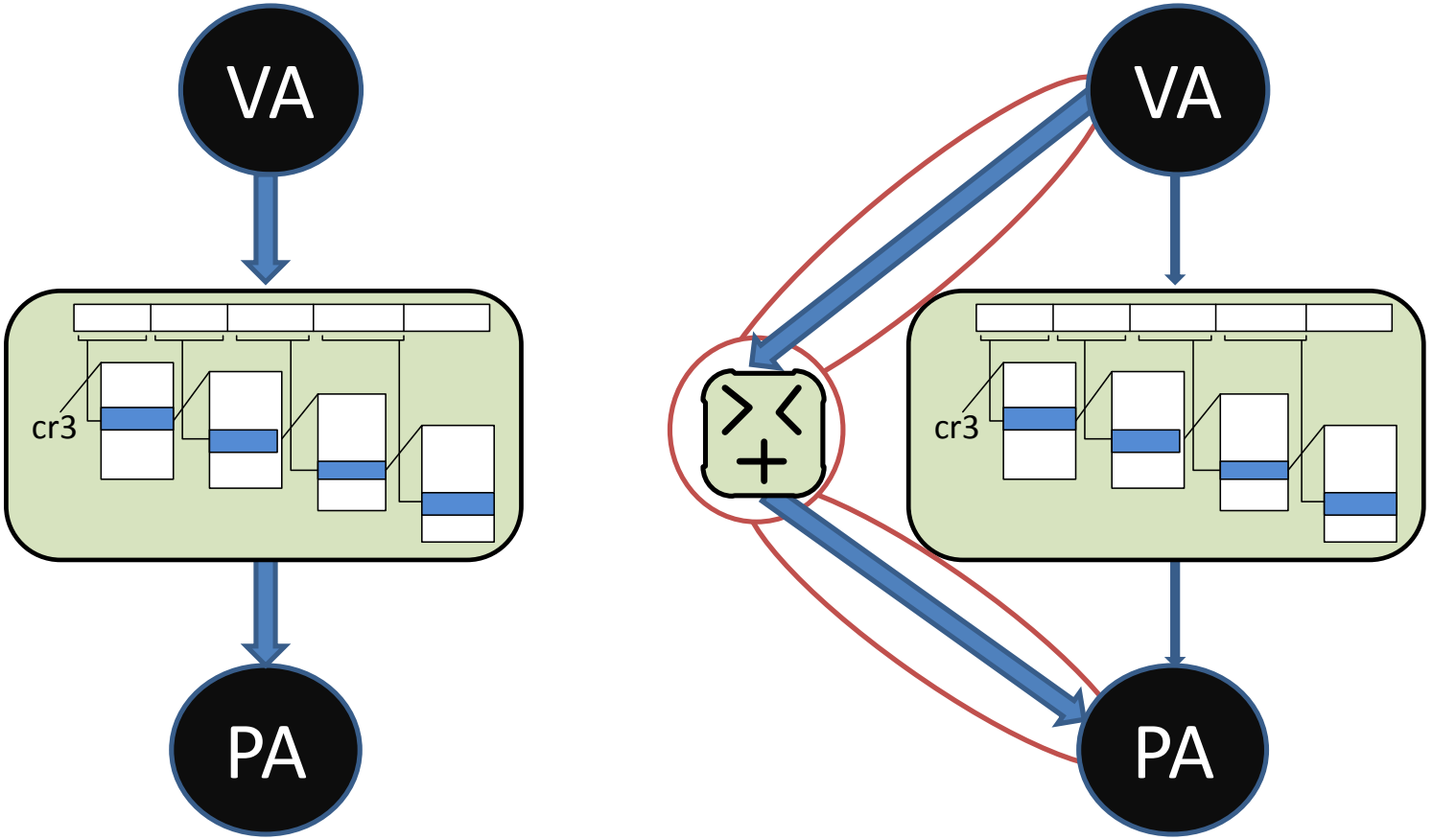
Basu et al. [ISCA 2013]

Translation with Direct Segments



Basu et al. [ISCA 2013]

Direct Segments

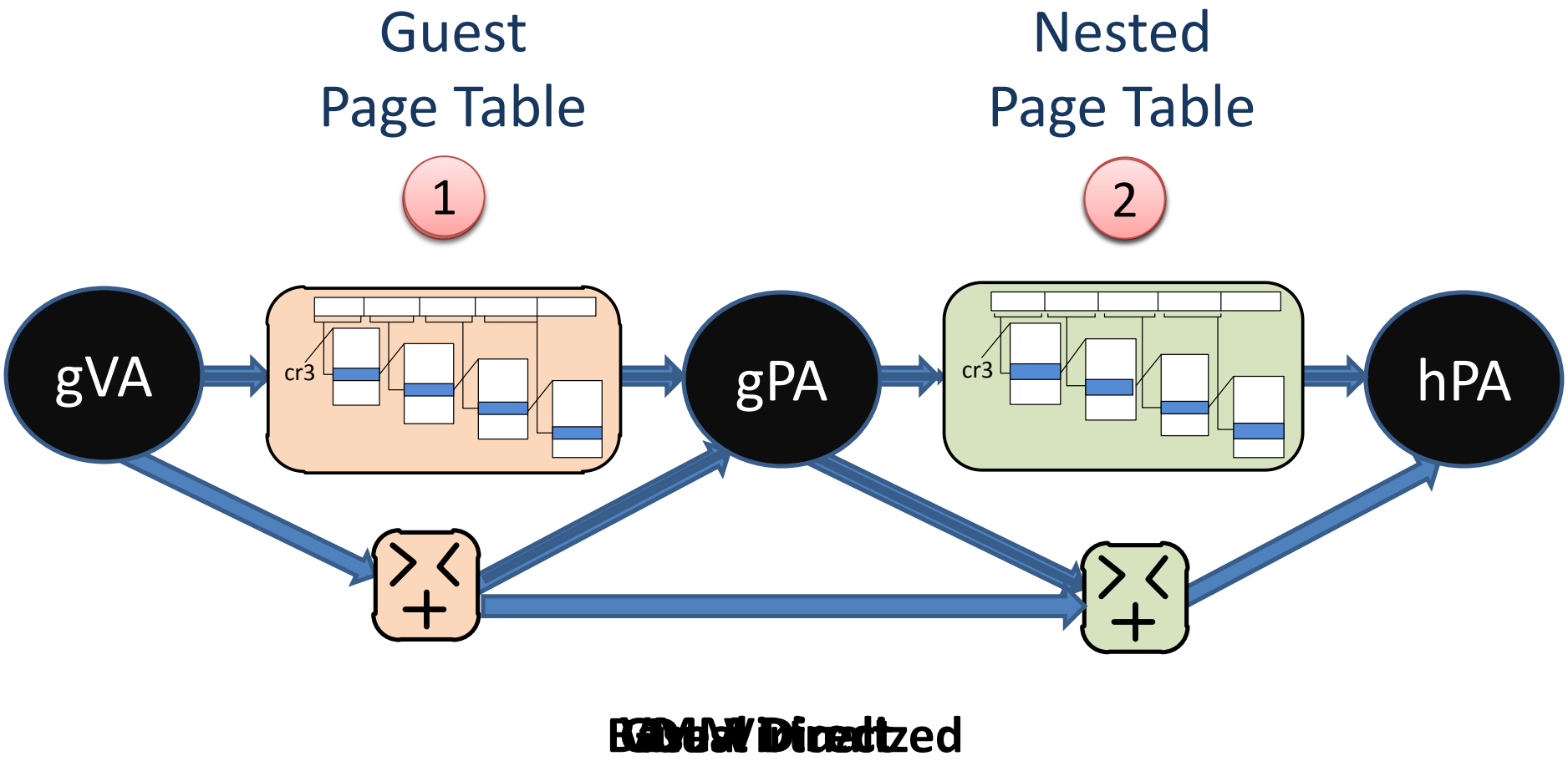


Base Native **Direct Segments**

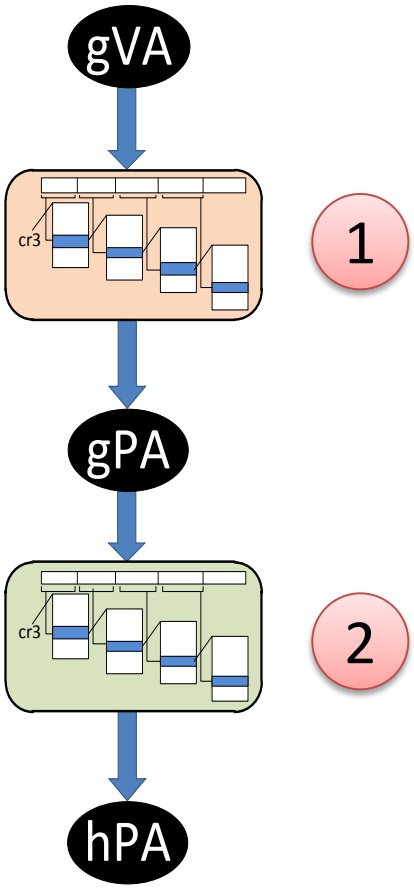
Outline

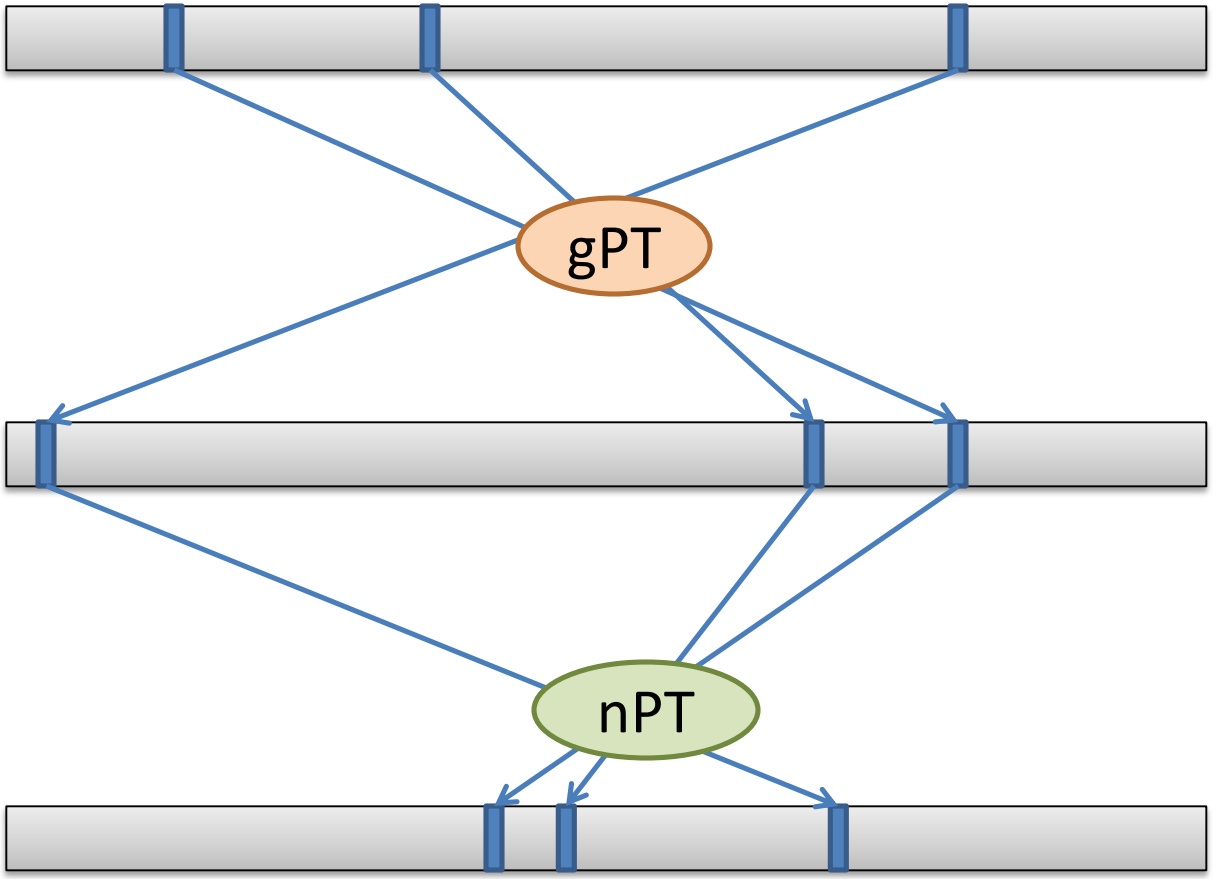
- Motivation
- Review: Direct Segments
- Virtualized Direct Segments ←
- Evaluation
  - Methodology
  - Results
- Summary

Modes



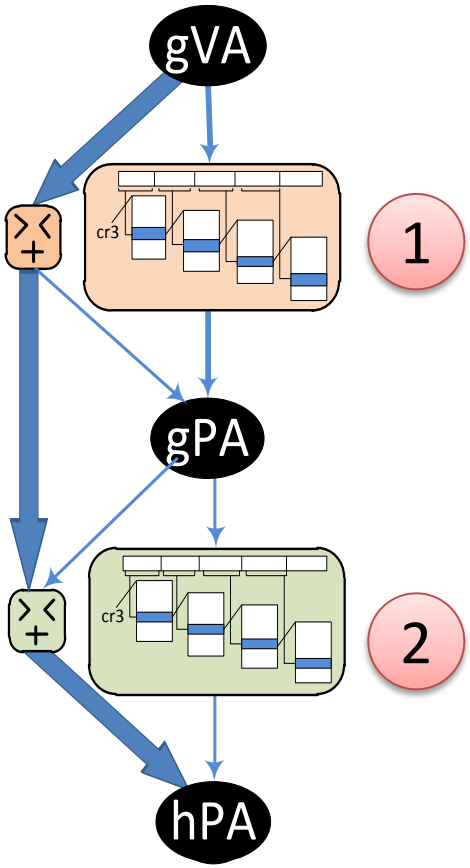
Base Virtualized: Translation

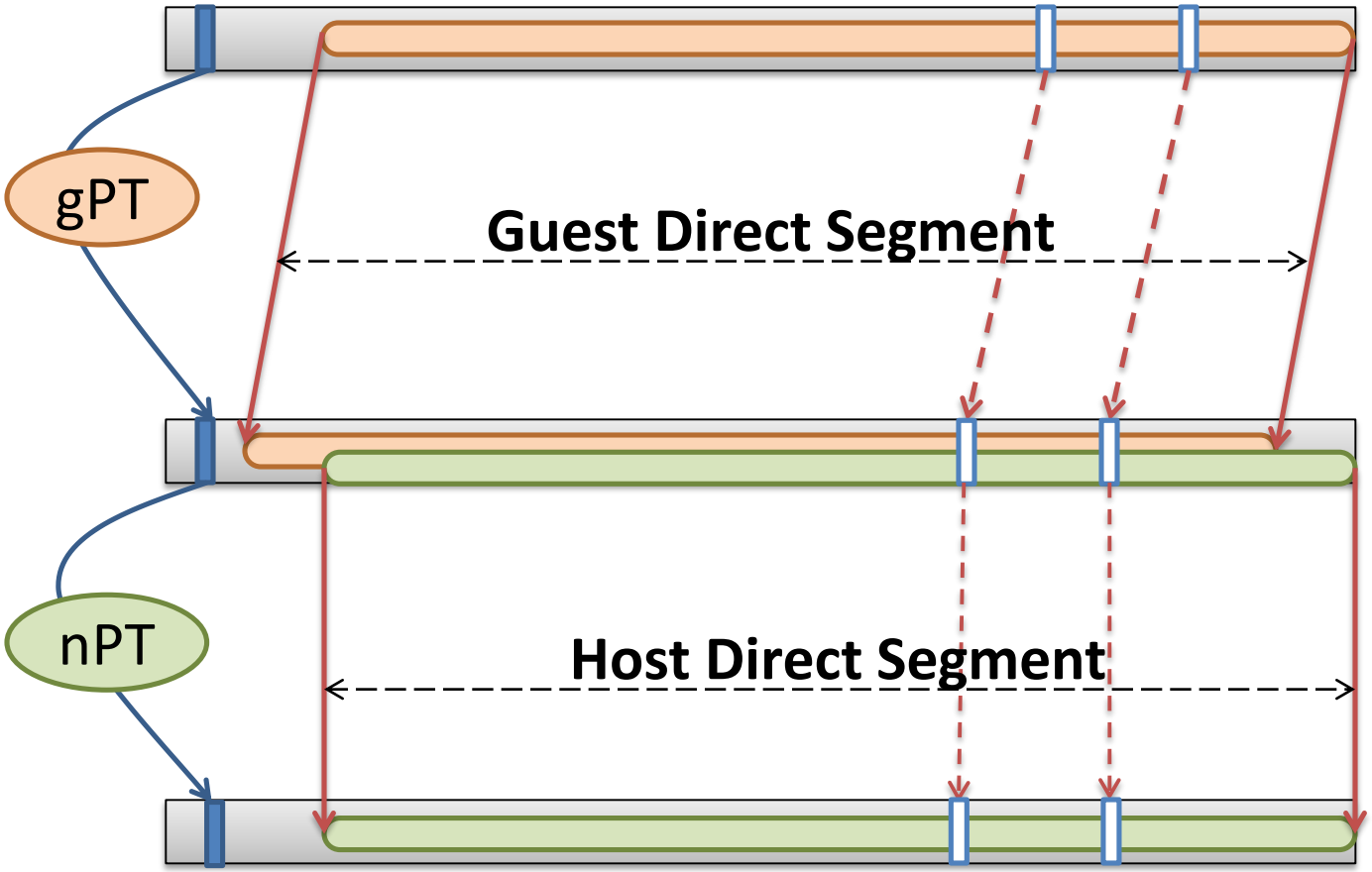




Base Virtualized

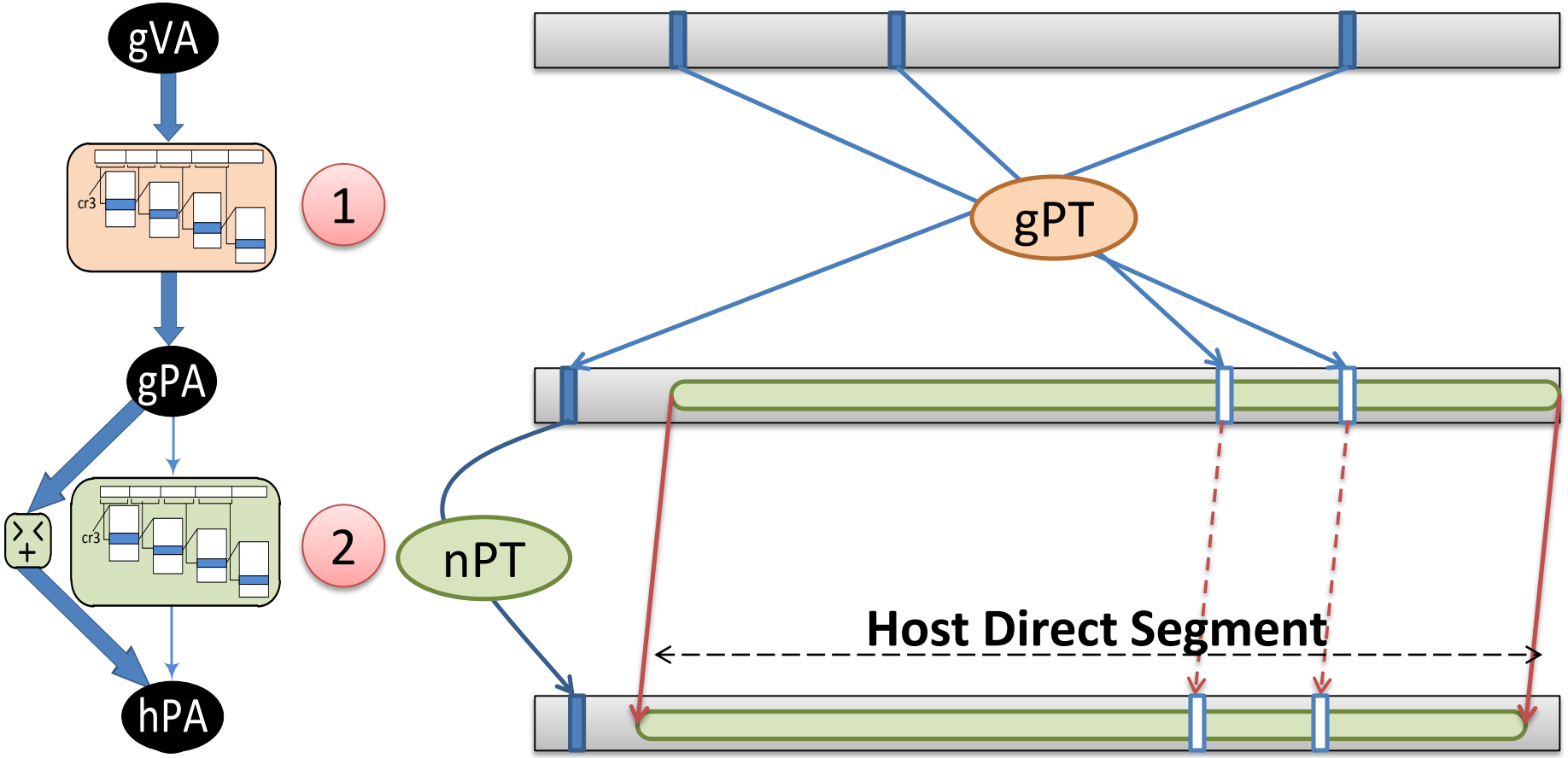
Dual Direct: Translation





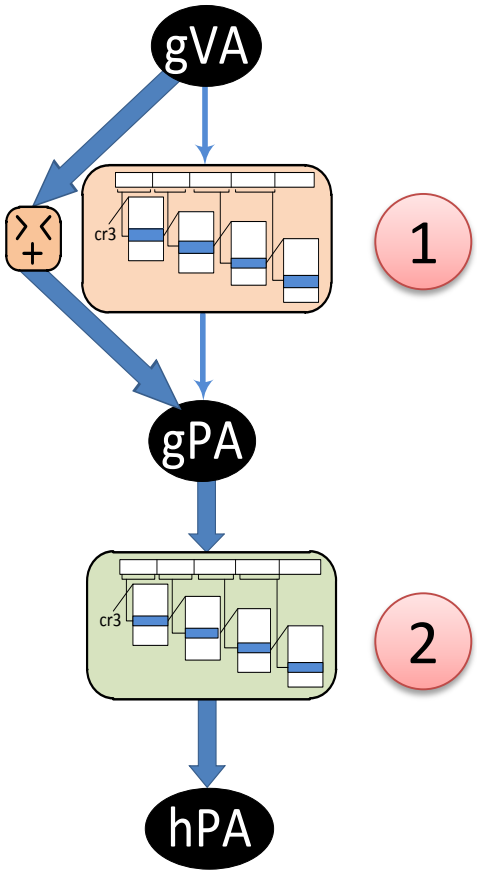
Dual Direct

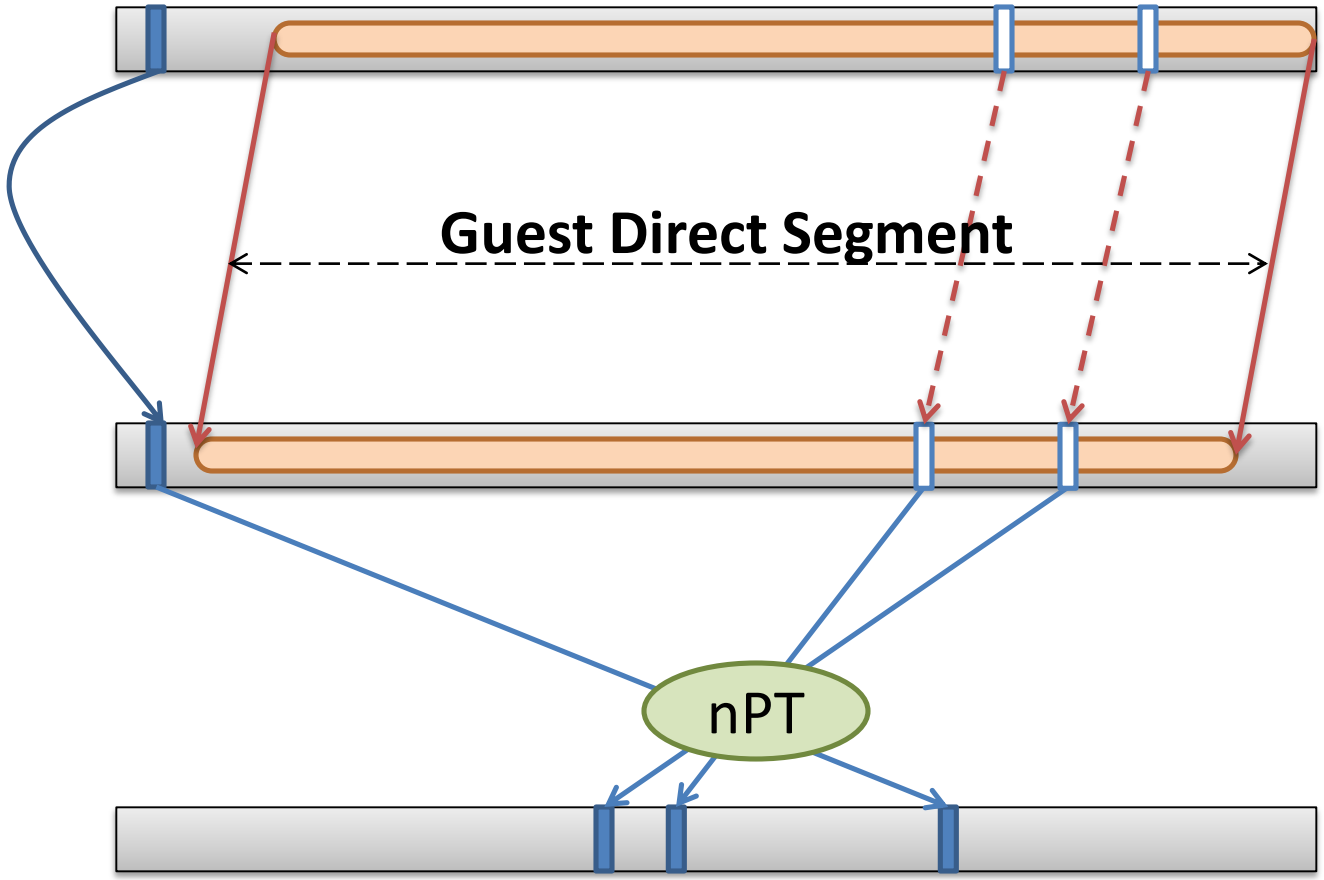
VMM Direct: Translation



VMM Direct

Guest Direct: Translation





Tradeoffs: Efficiency

Properties	Base Virtualized	Dual Direct	VMM Direct	Guest Direct
Dimension of page walk	2D	0D	1D	1D
# of mem. accesses for most page walks	24	0	4	4
# of base-bound computation for page walks	0	1	5	1

Tradeoffs: Compatibility

Properties	Base Virtualized	Dual Direct	VMM Direct	Guest Direct
Guest OS modifications	✗	✓	✗	✓
VMM modifications	✗	✓	✓	minimal
Application Category	Any	Big-memory	Any	Big-memory

Tradeoffs: Memory Overcommit

Properties	Base Virtualized	Dual Direct	VMM Direct	Guest Direct
Page Sharing	✓	limited	limited	✓
Ballooning	✓	limited	✓	limited
Guest OS Swapping	✓	limited	✓	limited
VMM Swapping	✓	limited	limited	✓

Outline

- Motivation
- Review: Direct Segments
- Virtualized Direct Segments
- Optimizations ←
- Evaluation
  - Methodology
  - Results
- Summary

Optimizations

- **Issue 1:** Guest/Host memory fragmentation
  - Guest physical memory: *Self-ballooning*
  - Host physical memory: *Compaction*
- **Issue 2:** Permanent "hard" memory faults
  - *Escape filter* to provide alternate translation
  - Bloom filter stores small number of faulty pages
  - Filter checked in parallel with segment registers

For more details: Come to the poster

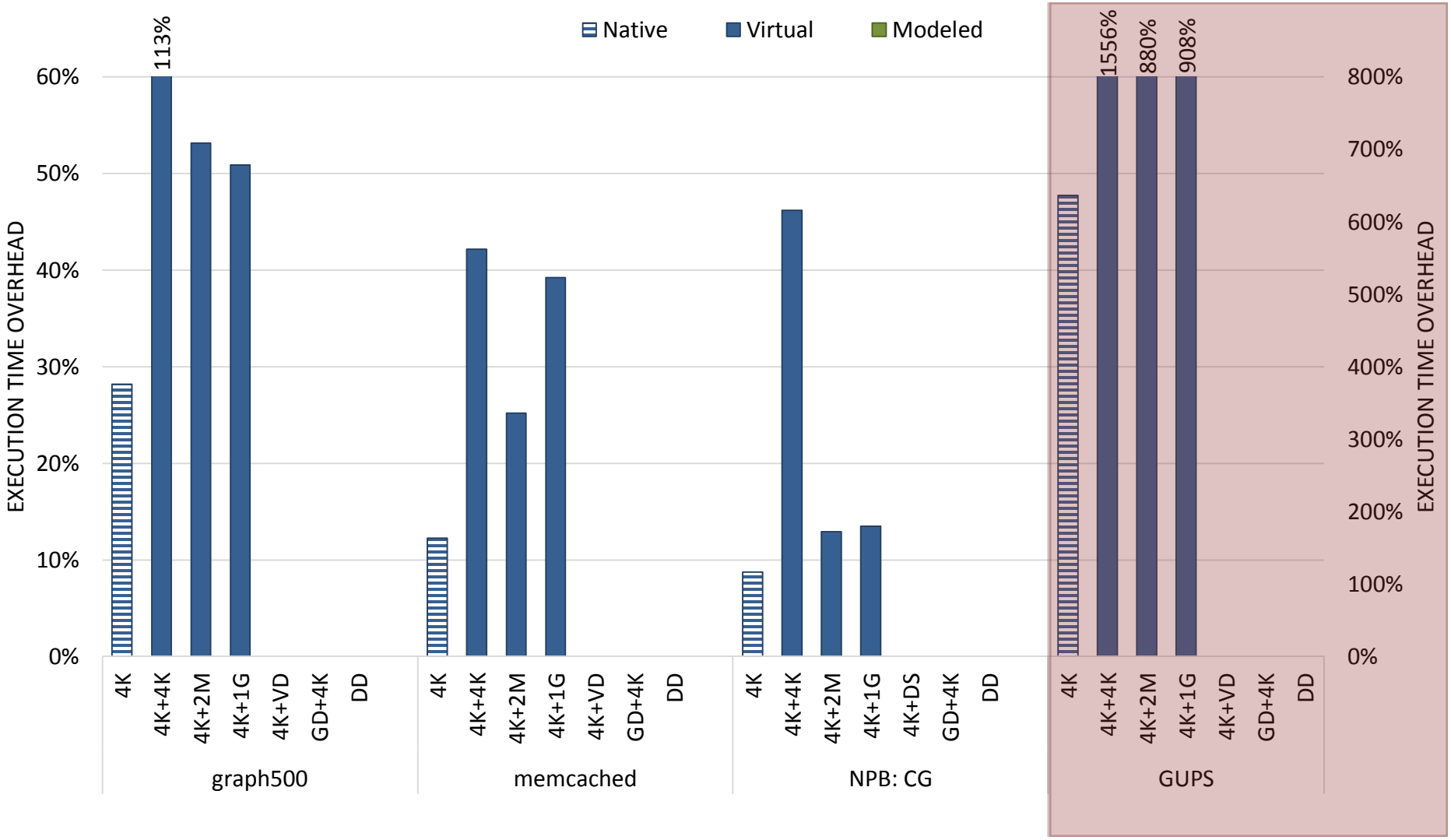
Outline

- Motivation
- Review: Direct Segments
- Virtualized Direct Segments
- Optimizations
- Evaluation ←
  - Methodology
  - Results
- Summary

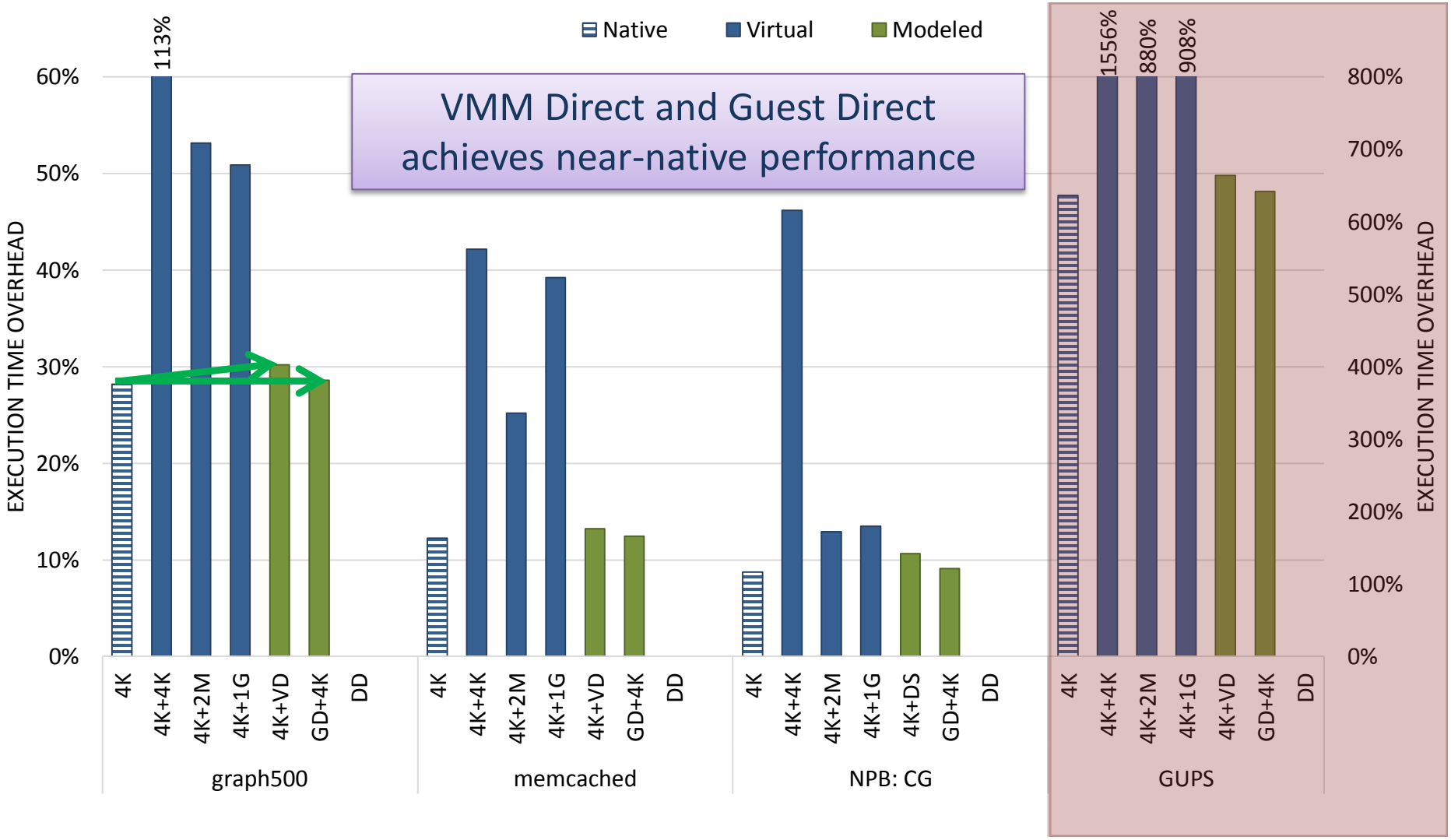
Methodology

- Estimate performance of future hardware
  - Measure fraction of TLB misses to segmented memory
  - Measure TLB miss cost with performance counters
  - Estimate performance gain with linear model
- Prototype
  - Linux v3.12.13 host/guest
  - Qemu-KVM hypervisor
- Intel 12-core Sandy-bridge with 96GB memory

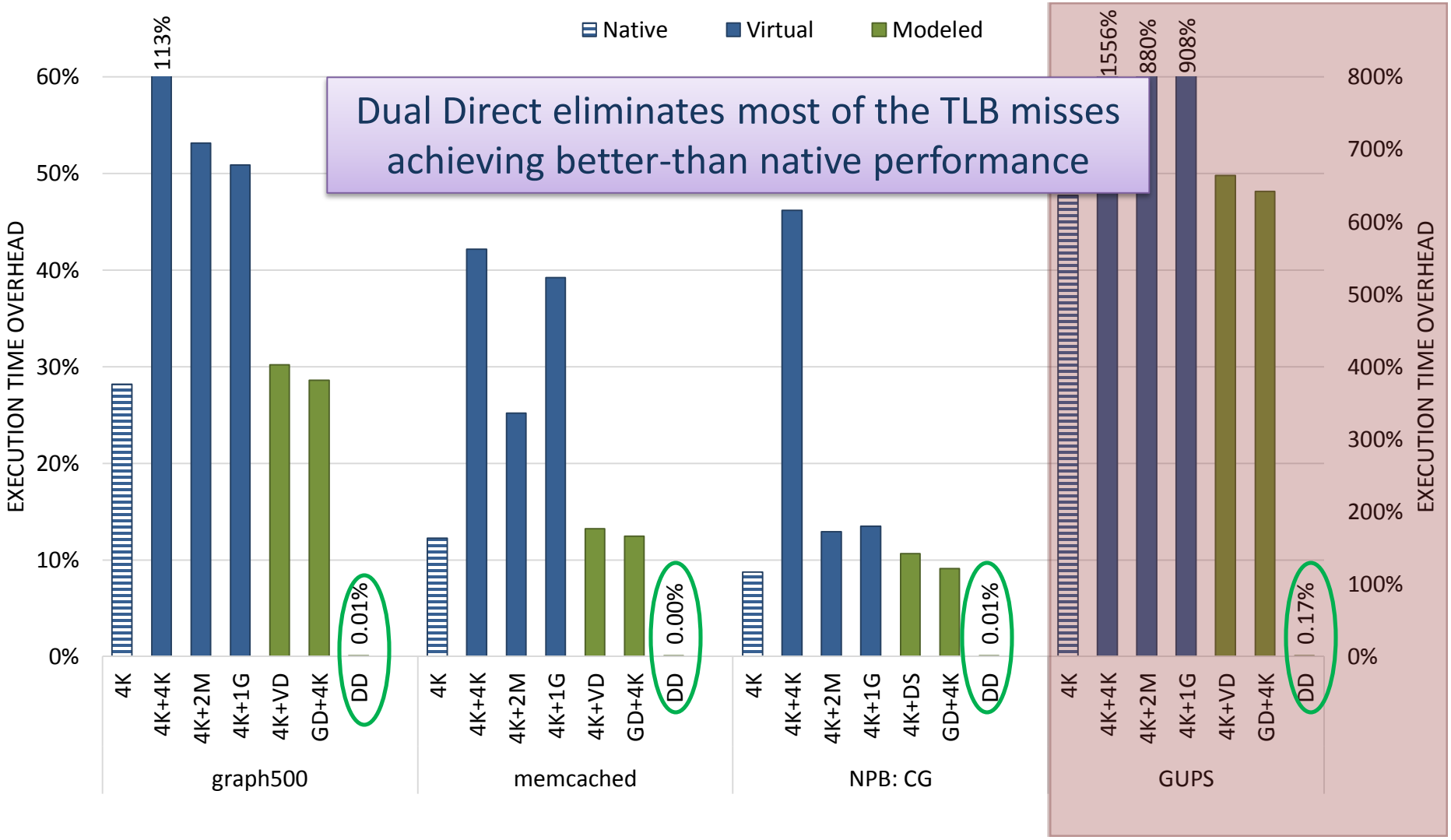
Results


Native
Virtual
Modeled
EXECUTION TIME OVERHEAD
60%
50%
40%
30%
20%
10%
0%
800%
700%
600%
500%
400%
300%
200%
100%
0%
113%
1556%
880%
908%
4K
4K+4K
4K+2M
4K+1G
4K+VD
4K+DS
GD+4K
DD
graph500
memcached
NPB: CG
GUPS

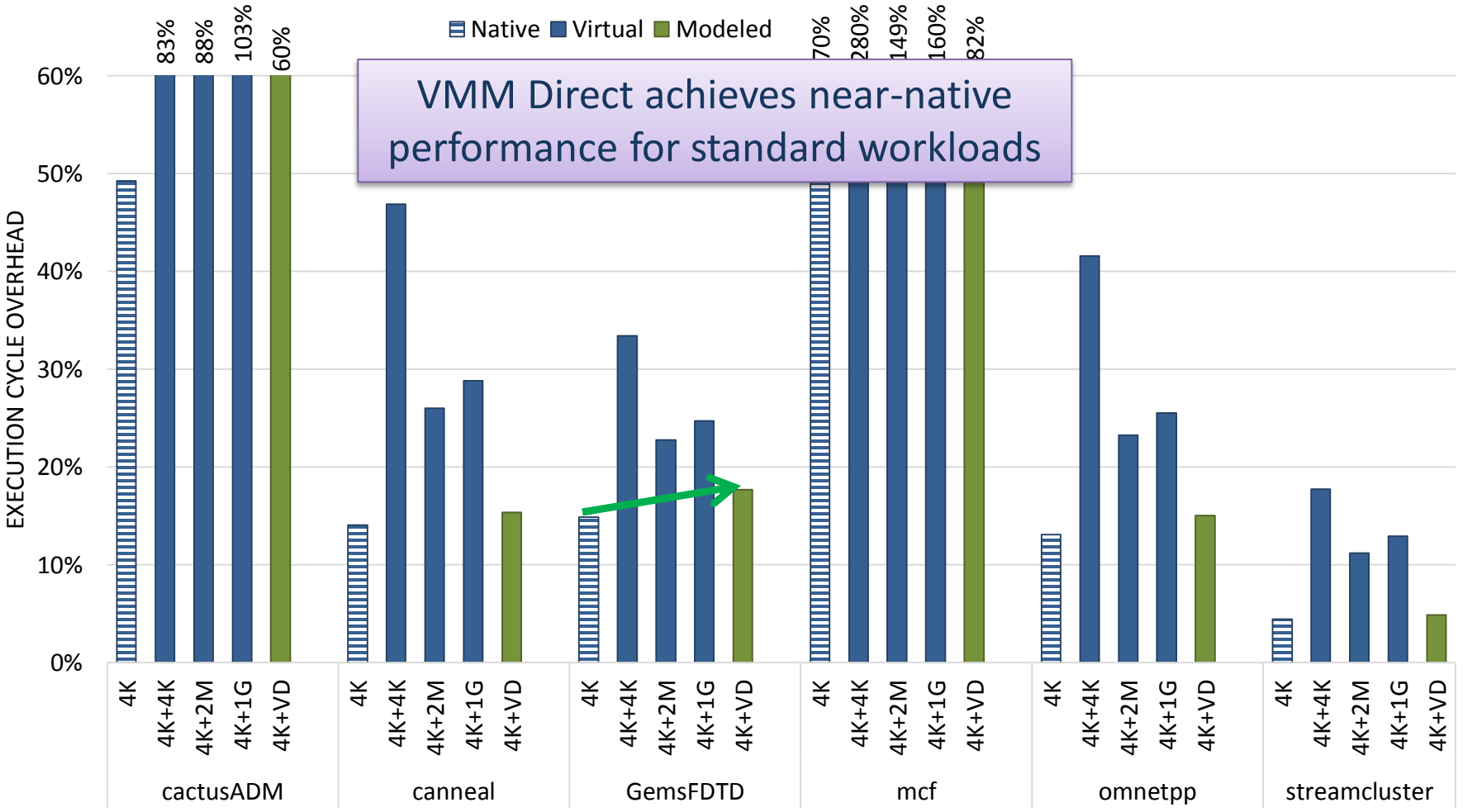
Results


Native
Virtual
Modeled
VMM Direct and Guest Direct achieves near-native performance
EXECUTION TIME OVERHEAD
60%
50%
40%
30%
20%
10%
0%
800%
700%
600%
500%
400%
300%
200%
100%
0%
113%
1556%
880%
908%
4K
4K+4K
4K+2M
4K+1G
4K+VD
GD+4K
4K+DS
DD
graph500
memcached
NPB: CG
GUPS

Results


Native
Virtual
Modeled
Dual Direct eliminates most of the TLB misses achieving better-than native performance
EXECUTION TIME OVERHEAD
60%
50%
40%
30%
20%
10%
0%
800%
700%
600%
500%
400%
300%
200%
100%
0%
113%
1556%
880%
908%
0.01%
0.00%
0.01%
0.17%
4K
4K+4K
4K+2M
4K+1G
4K+VD
GD+4K
4K+DS
DD
graph500
memcached
NPB: CG
GUPS

Results


Native
Virtual
Modeled
EXECUTION CYCLE OVERHEAD
60%
50%
40%
30%
20%
10%
0%
83%
88%
103%
60%
70%
280%
149%
160%
82%
4K
4K+4K
4K+2M
4K+1G
4K+VD
cactusADM
canneal
GemsFDTD
mcf
omnetpp
streamcluster
VMM Direct achieves near-native performance for standard workloads

Summary

- Problem: TLB misses in virtual machines
  - Hardware-virtualized MMU has high overheads
  - ➢ Up to 280% overhead
- Prior Work: **Direct Segments** – unvirtualized case
- Solution: segmentation to bypass paging
  - **Extend Direct Segments** for virtualization
  - **Three configurations** with different tradeoffs
- Results
  - **Near-** or **better-than**-native performance

Questions ?

For more details: Come to the poster