Enhancing Virtualized Application Performance through Dynamic Adaptive Paging Mode Selection

Chang S. Bae, John R. Lange, Peter A. Dinda

Prescience Lab, Dept. of EECS, Northwestern Univ.

Dept. of CS, Univ. of Pittsburgh

Contributions of this work

- Minimizing cost of paging translation in virtualized environments
  - Generic applicability: enterprise, datacenter and etc.

Contributions of this work

- Minimizing cost of paging translation in virtualized environments

- **Dynamically adaptive scheme**
  - Selects between hardware-based and software-based translation depending on workload
  - "Best of both worlds" performance

Contributions of this work

- Minimizing cost of paging translation in virtualized environments

- Dynamically adaptive scheme
- **Near native performance**

Contributions of this work

- Minimizing cost of paging translation in virtualized environments

- Dynamically adaptive scheme
- Near native performance
- **Design and implementation on real system**
  - Our open source Palacios VMM

Outline

- Introduction
- Background and Motivation
  - Shadow paging versus Nested paging
  - Behaviors and metrics
- DAV^{2}M policy
- Evaluation
- Conclusion

Virtualization model

- Trap and emulate operation
  - Privileged instructions/events are trapped by VMM through hardware mechanism (VM exit)
  - Emulation in VMM

- Full system virtualization
  - *Applicable* to other model such as paravirtualization

(most widely used virtualization model)

Virtualized virtual memory

- Additional layer of indirection
  - Guest Virtual Address (GVA)

    → Guest Physical Address (GPA)

      → Host Physical Address (HPA)

- Software-based vs. Hardware-based

Virtualized virtual memory

- Additional layer of indirection
  - Guest Virtual Address (GVA) ⟸ *Virtual Address*

    → Guest Physical Address (GPA)

      → Host Physical Address (HPA)

- Software-based vs. Hardware-based

Virtualized virtual memory

- Additional layer of indirection
  - Guest Virtual Address (GVA) ⟸ *Virtual Address*

    → Guest Physical Address (GPA) ⟸ *Virtual Address*

    → Host Physical Address (HPA)

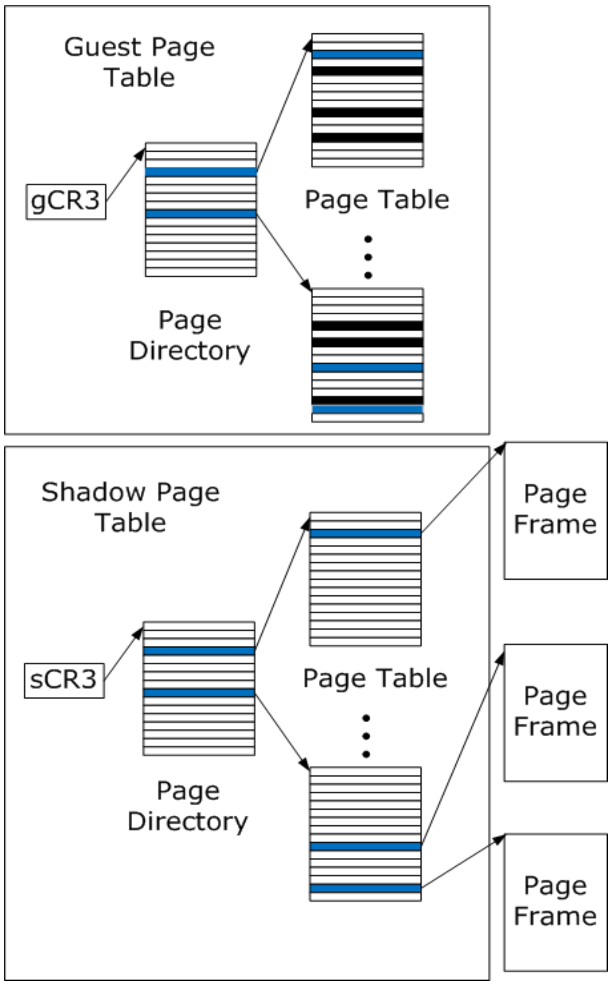
- Software-based vs. Hardware-based

Virtualized virtual memory

- Additional layer of indirection
  - Guest Virtual Address (GVA) $\Leftarrow$ *Virtual Address*
    → Guest Physical Address (GPA) $\Leftarrow$ *Virtual Address*
    → Host Physical Address (HPA) $\Leftarrow$ *Physical Address*

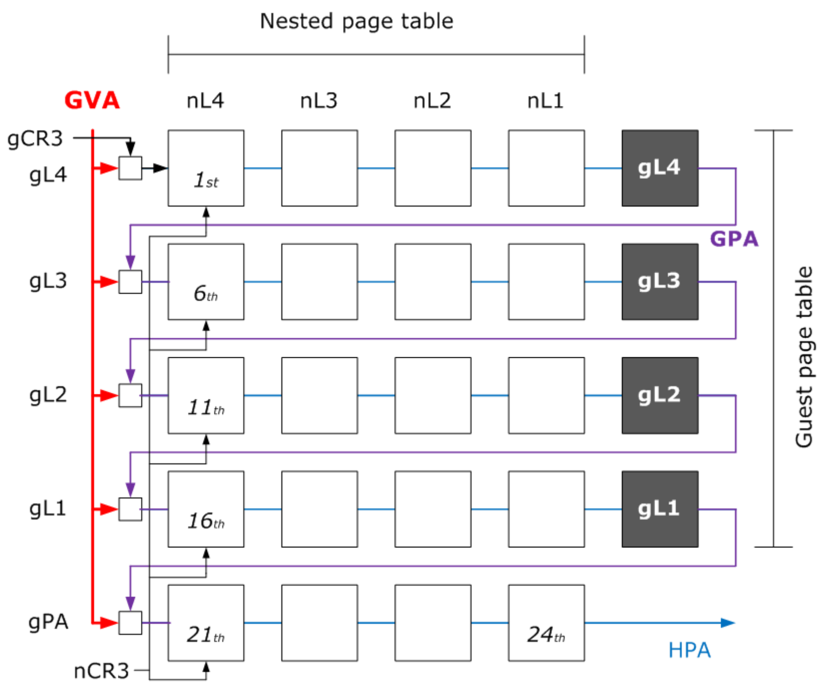
- Software-based vs. Hardware-based

Software: shadow paging with caching



- Software managed
  - VMM addresses missing entry in shadow page table at every trap
- Cached shadow page tables
  - Allow reuse of page table even after guest context switches
  - Need to be synchronized with every modification made by guest OS

Hardware: nested paging



ASPLOS'08 (Bargava et al)

- Hardware page walker addresses TLB misses
  - No VMM intervention
    - Except for nested page table allocations
- 2-dimensional page walk
  - Much longer than shadow
    - O(n^2): n is level of page table
  - Increased memory accesses

Insight from two approaches

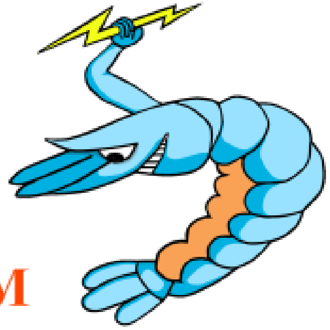
- Software-based approach
  - Good: short one dimensional page walk
  - Bad: many exits on guest page table edits
- Hardware-based approach
  - Good: no exits due to guest page table edits
  - Bad: long 2-dimensional page walk

Palacios VMM

- OS-independent embeddable virtual machine monitor

- Open source and freely available
- Virtualization layer for Kitten
  - Lightweight supercomputing OS from Sandia National Labs

- Successfully used on supercomputers, clusters (Infiniband and Ethernet), and servers



Palacios

An OS Independent Embeddable VMM

http://www.v3vee.org/palacios

Application benchmarks

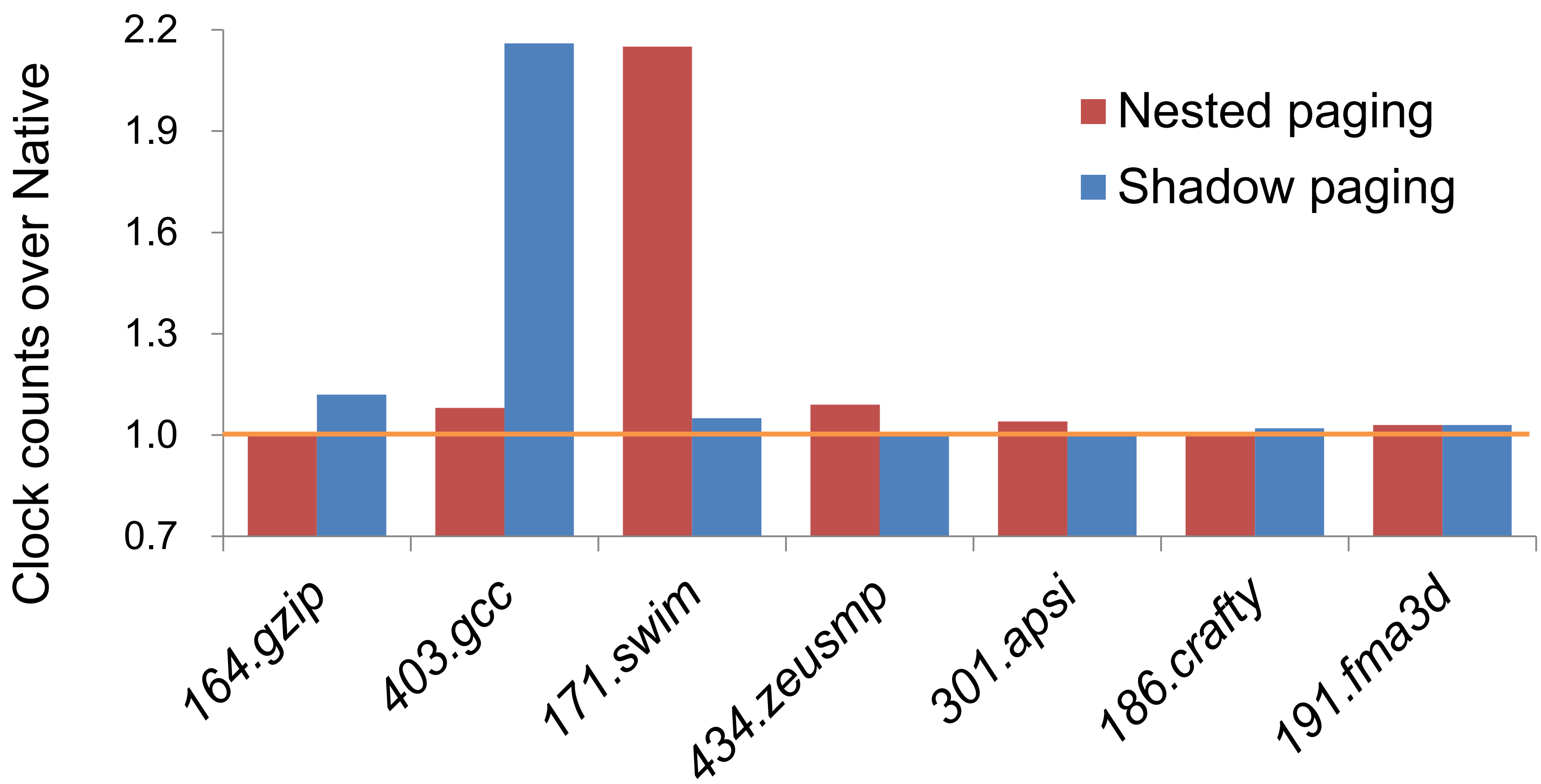
- SPEC CPU 2000/2006[1]
- PARSEC 2.1[2]

- Widely used and *representative* workloads
- In this talk, we focus on benchmarks with *the greatest variations* in a virtualized system

[1] SPEC CPU Benchmark Suites
www.spec.org/cpu

[2] PARSEC Benchmark Suite
parsec.cs.princeton.edu

No single best approach



Lower is better

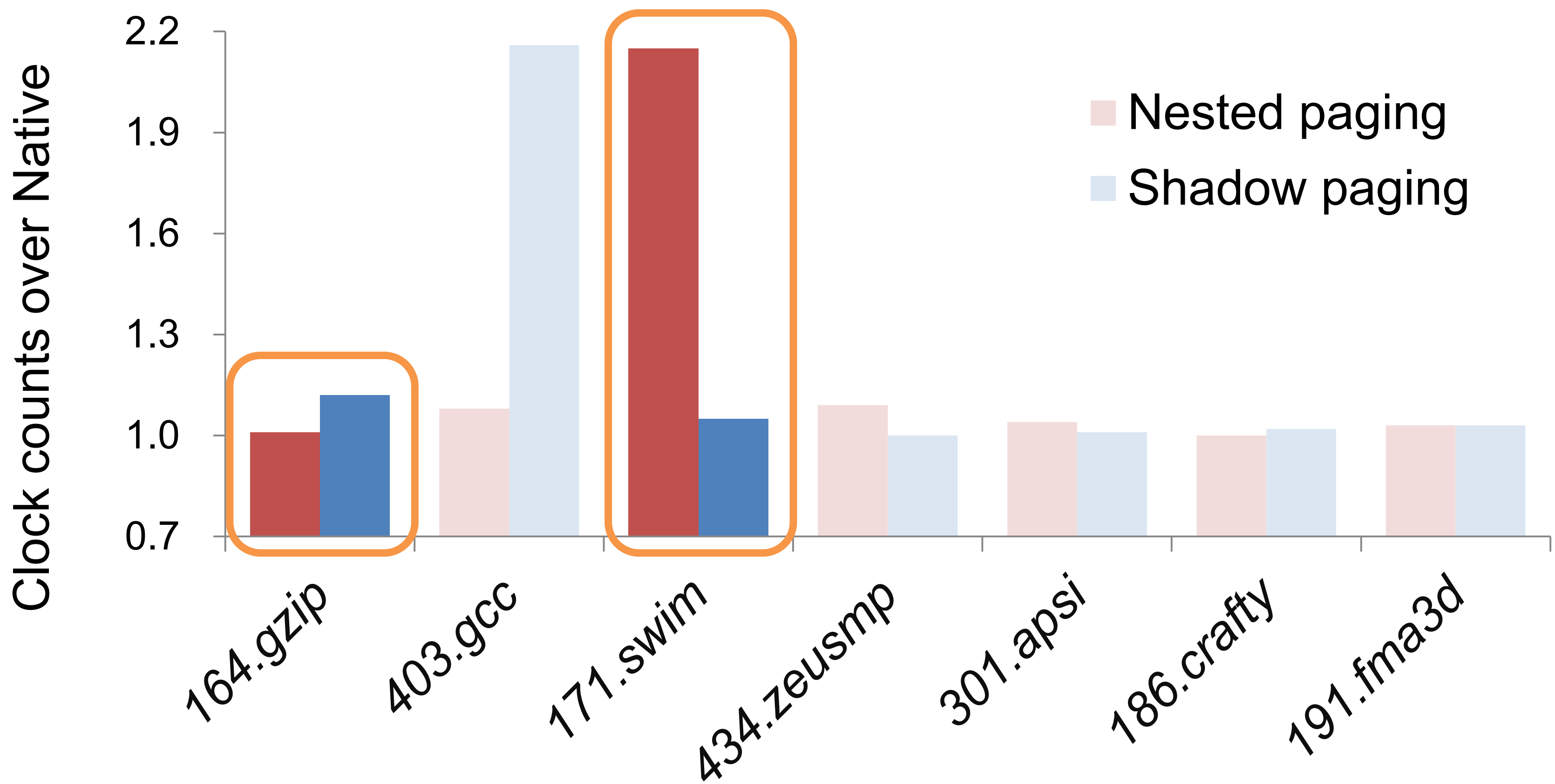
Performance metrics with low overhead at runtime

- Application performance
  - ***Cycles per instruction* (CPI)**
  - Distinct from overall runtime

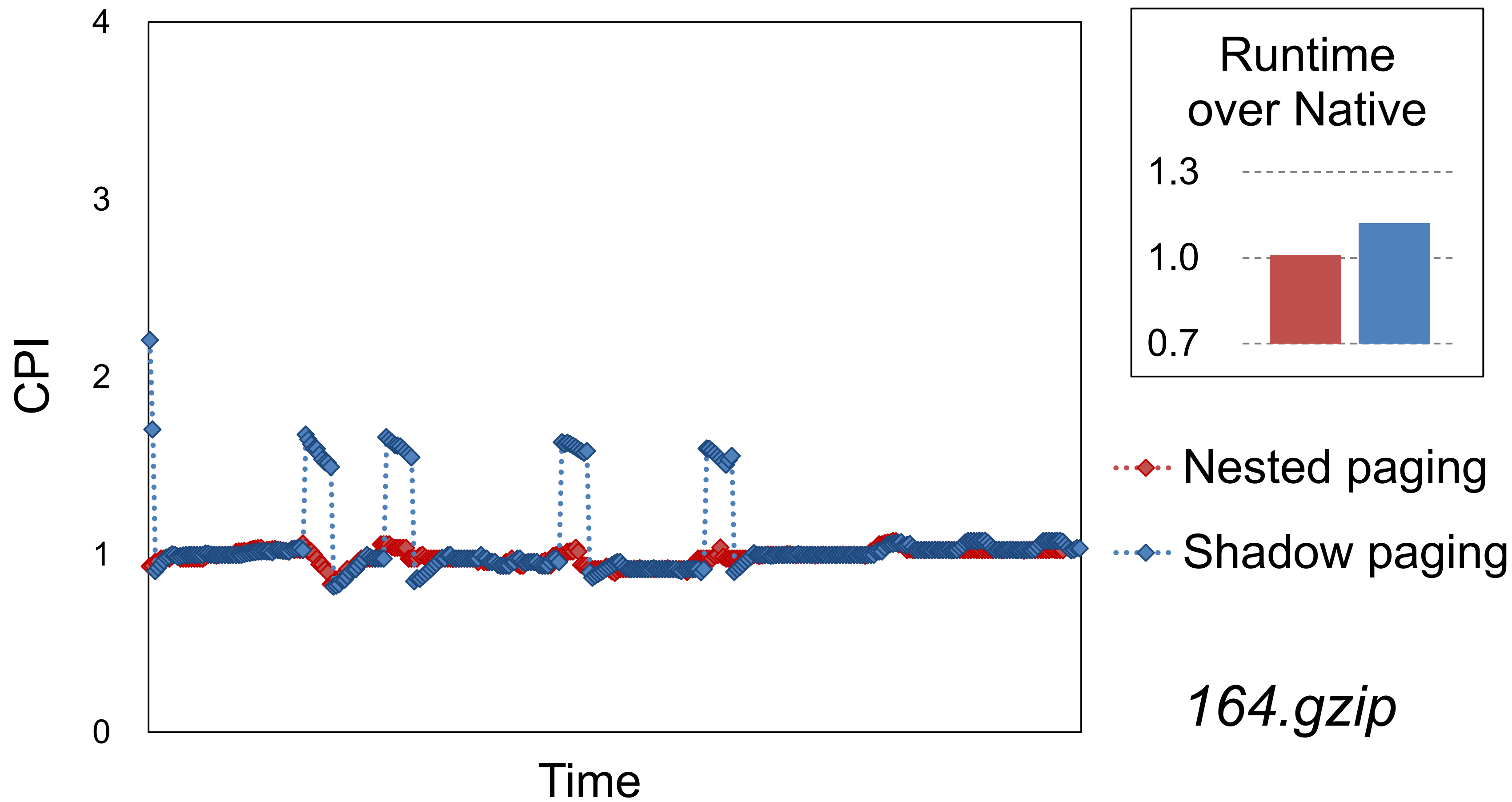
- Nested paging performance
  - ***TLB miss frequency***

- Shadow paging performance
  - ***Page fault VM exit frequency***

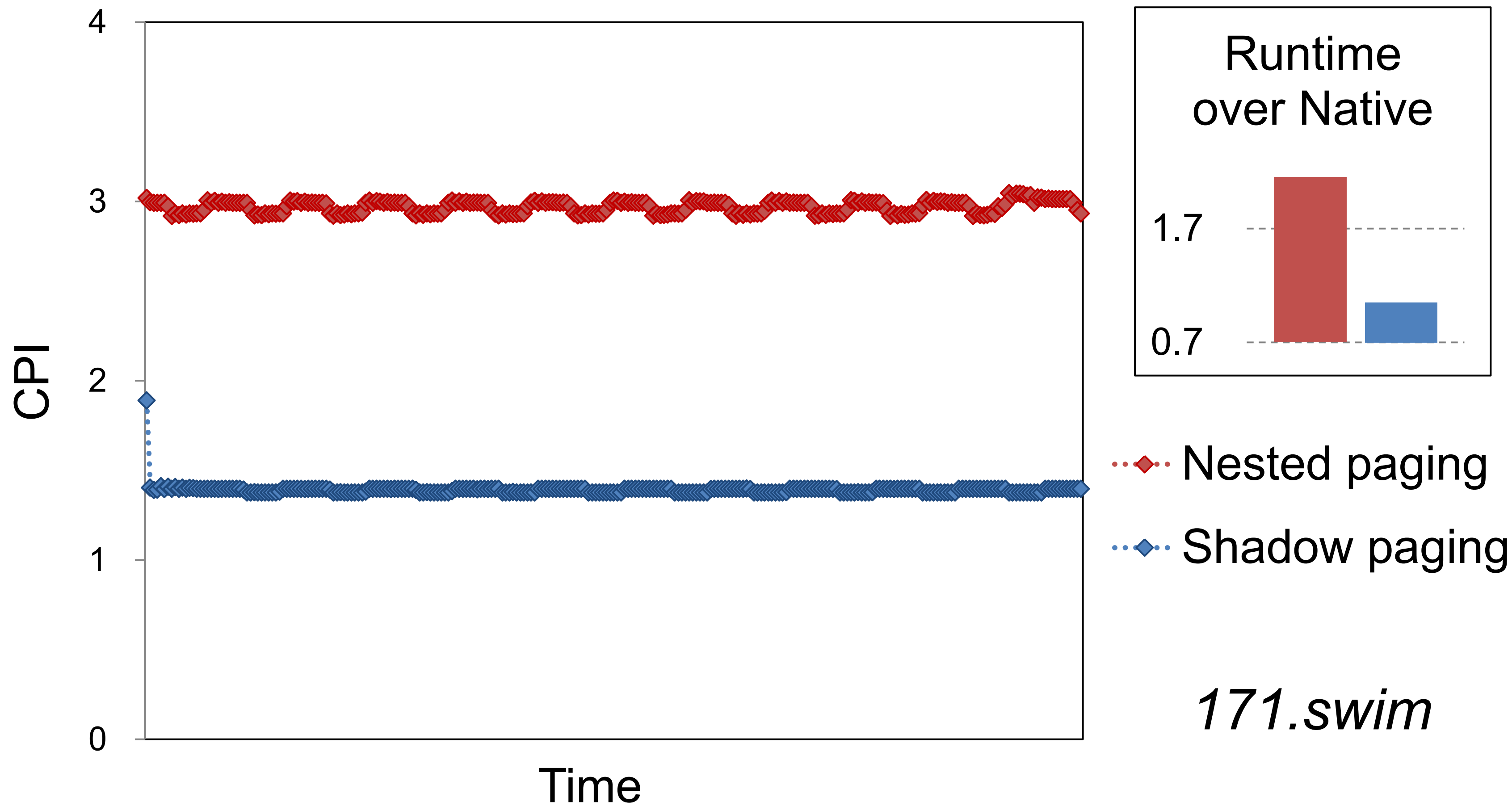
Deeper look with metrics



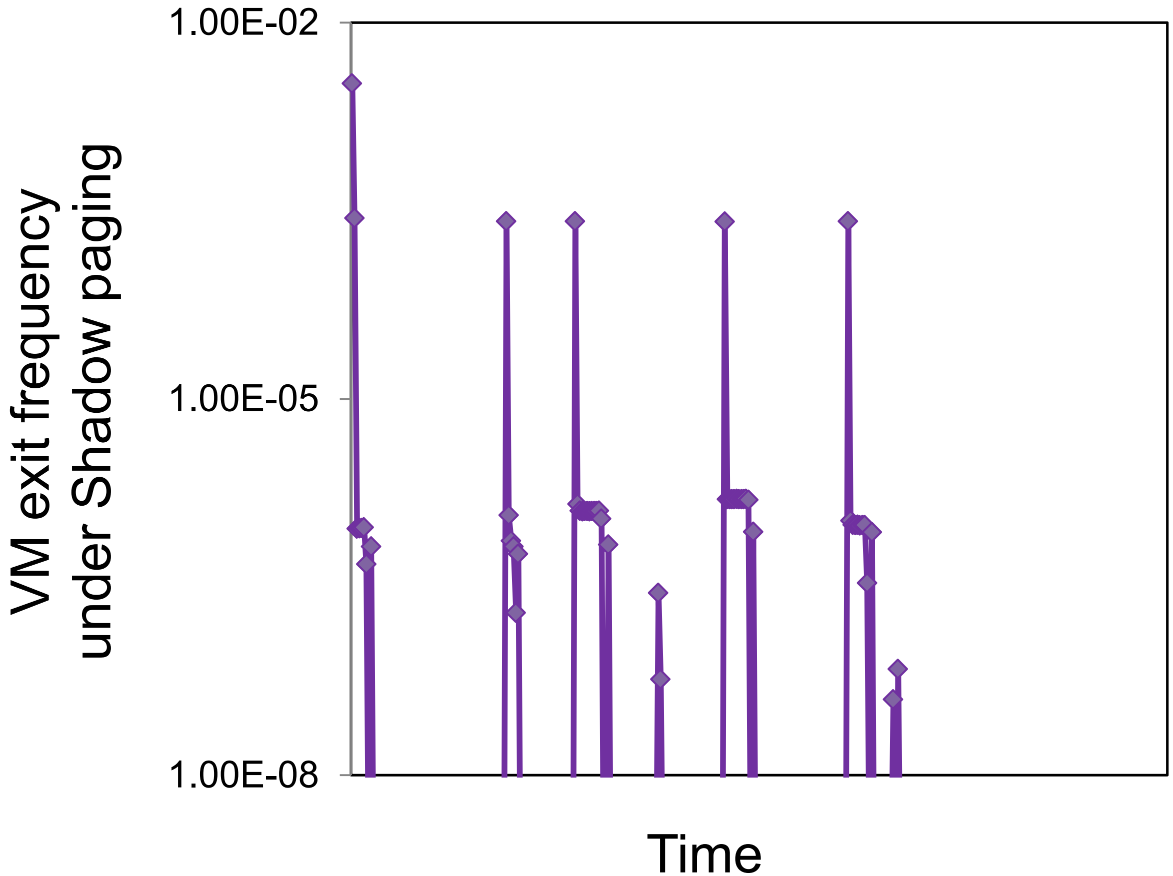
CPI as a performance measure

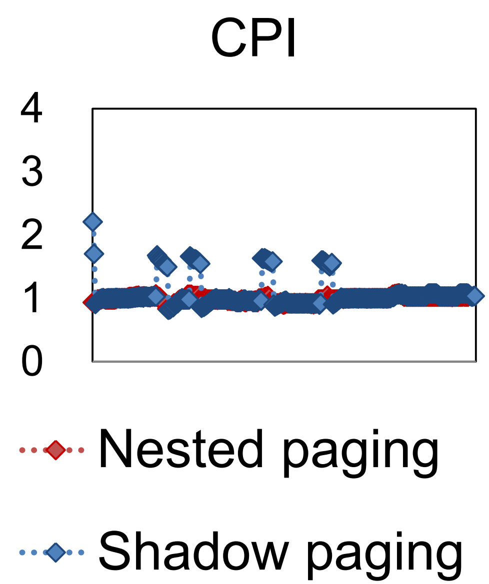


CPI as a performance measure



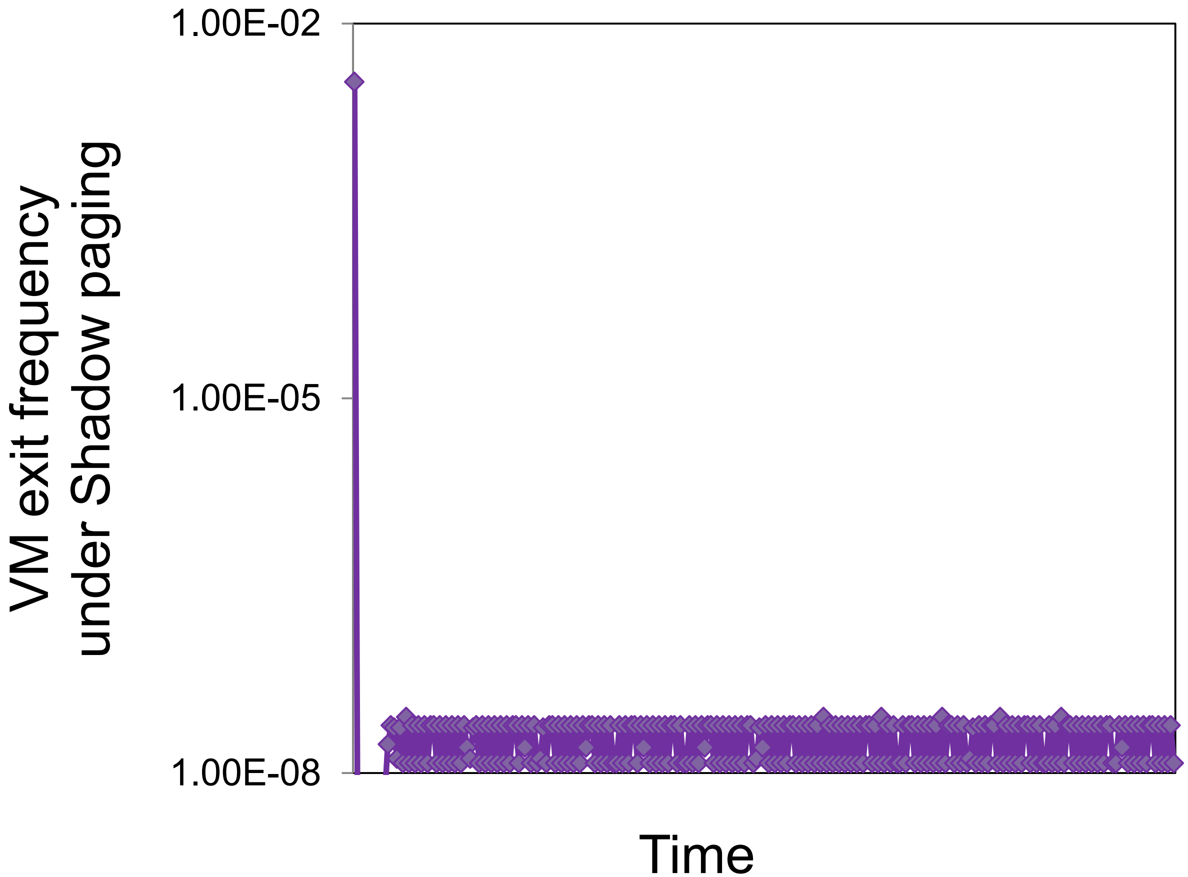
Peak page faults hurt shadow performance

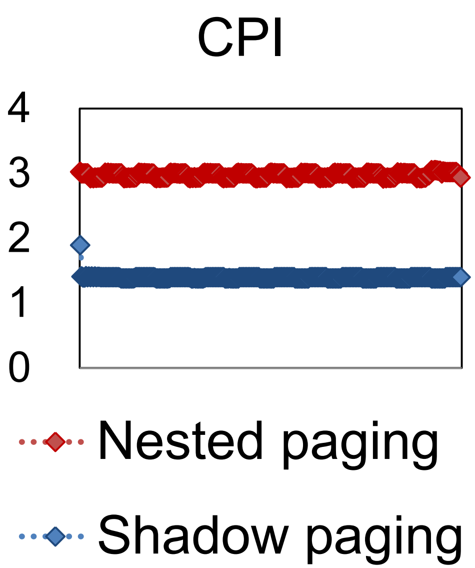




164.gzip

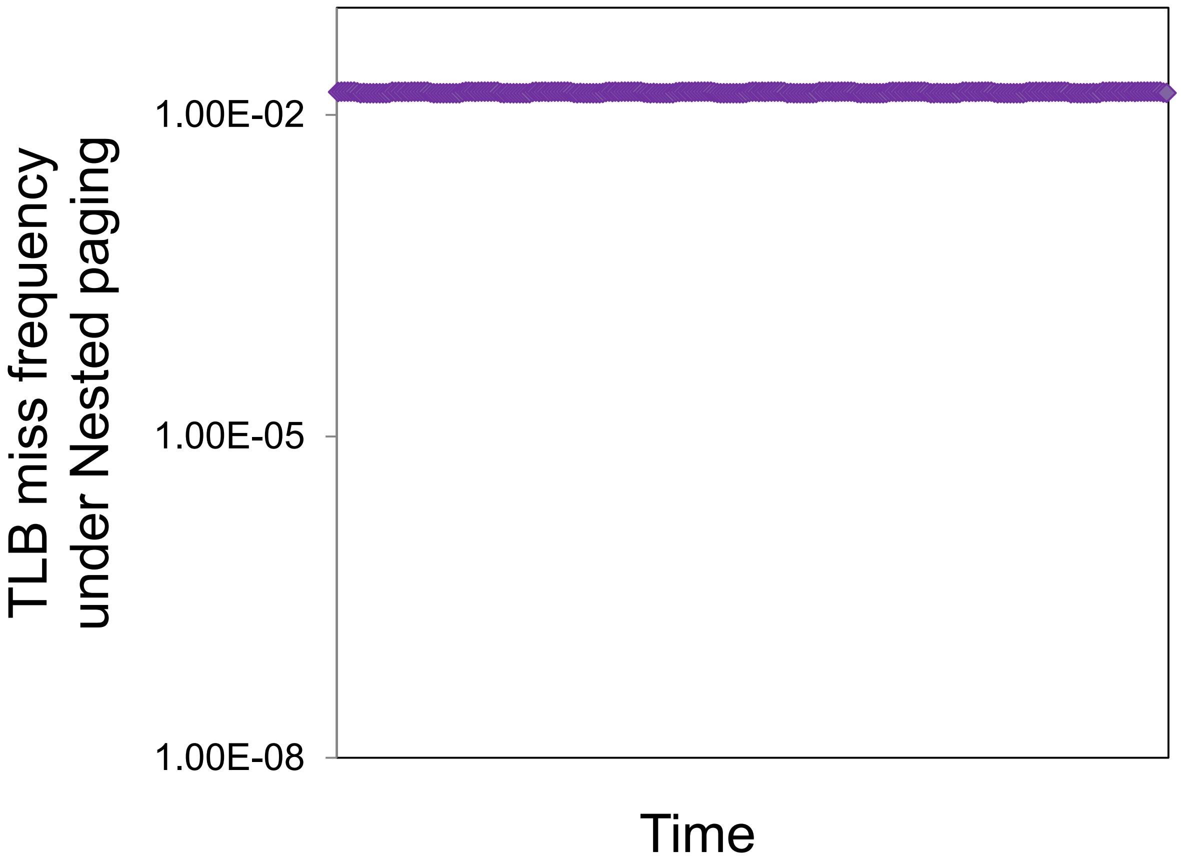
Otherwise, shadow should be fine

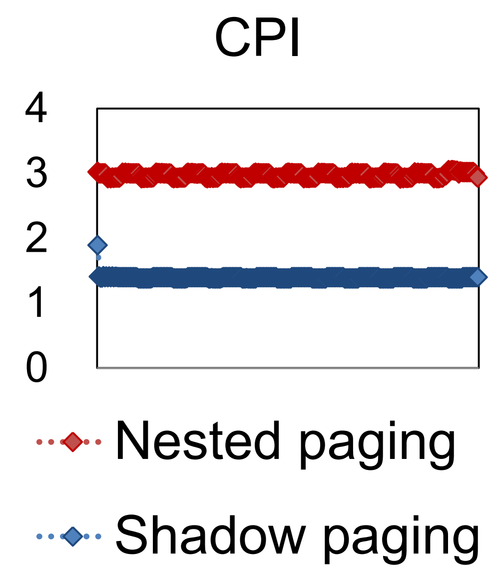




171.swim

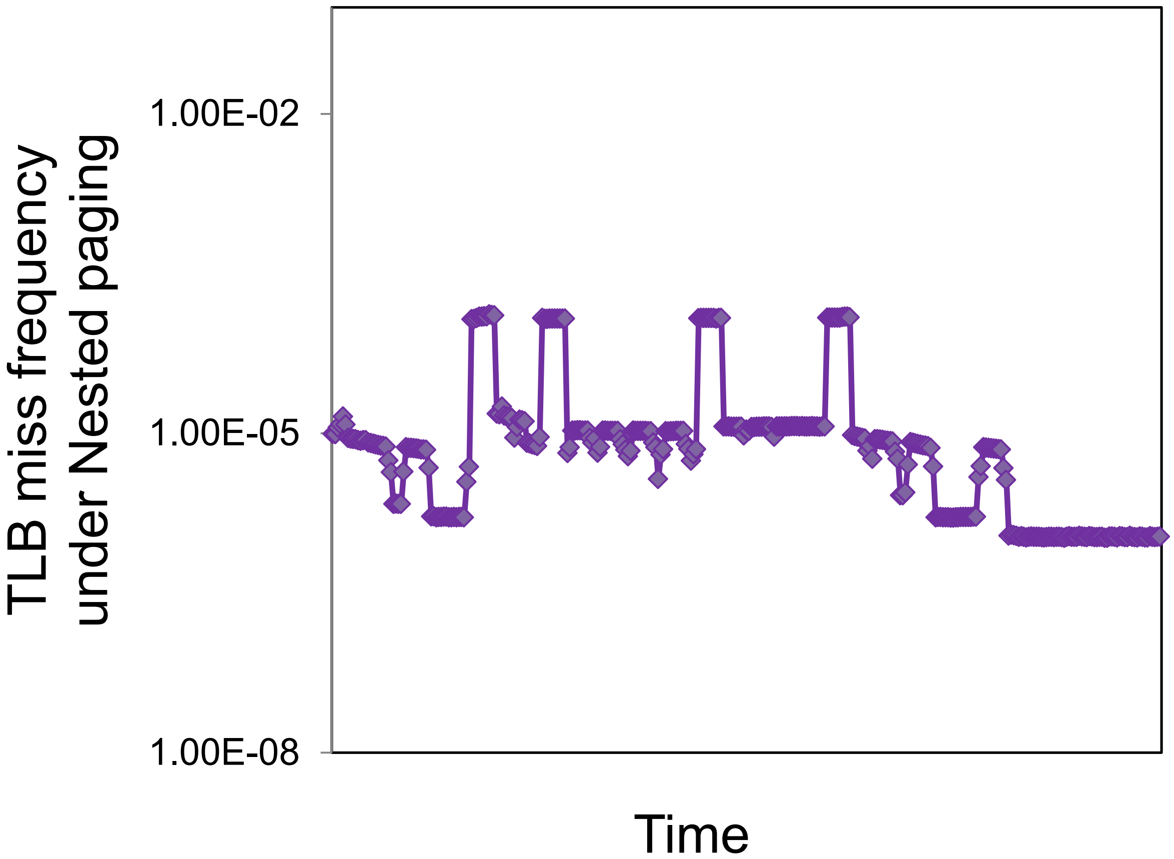
High TLB miss rate degrades nested performance

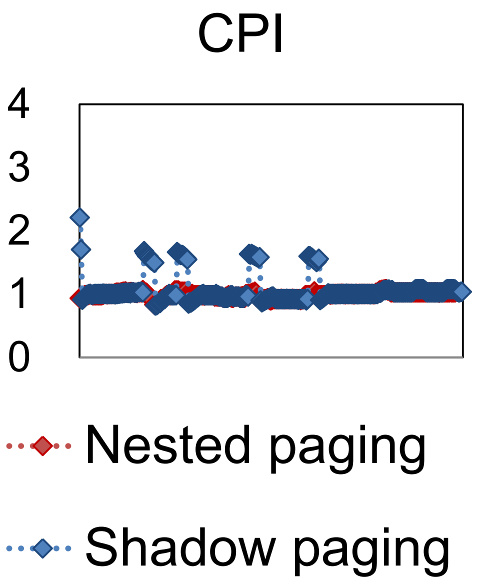




171.swim

Otherwise, nested should be fine





164.gzip

Outline

- Introduction
- Background and Motivation
- DAV^{2}M policy
  - Threshold-based heuristics
  - Threshold value control
- Evaluation
- Conclusion

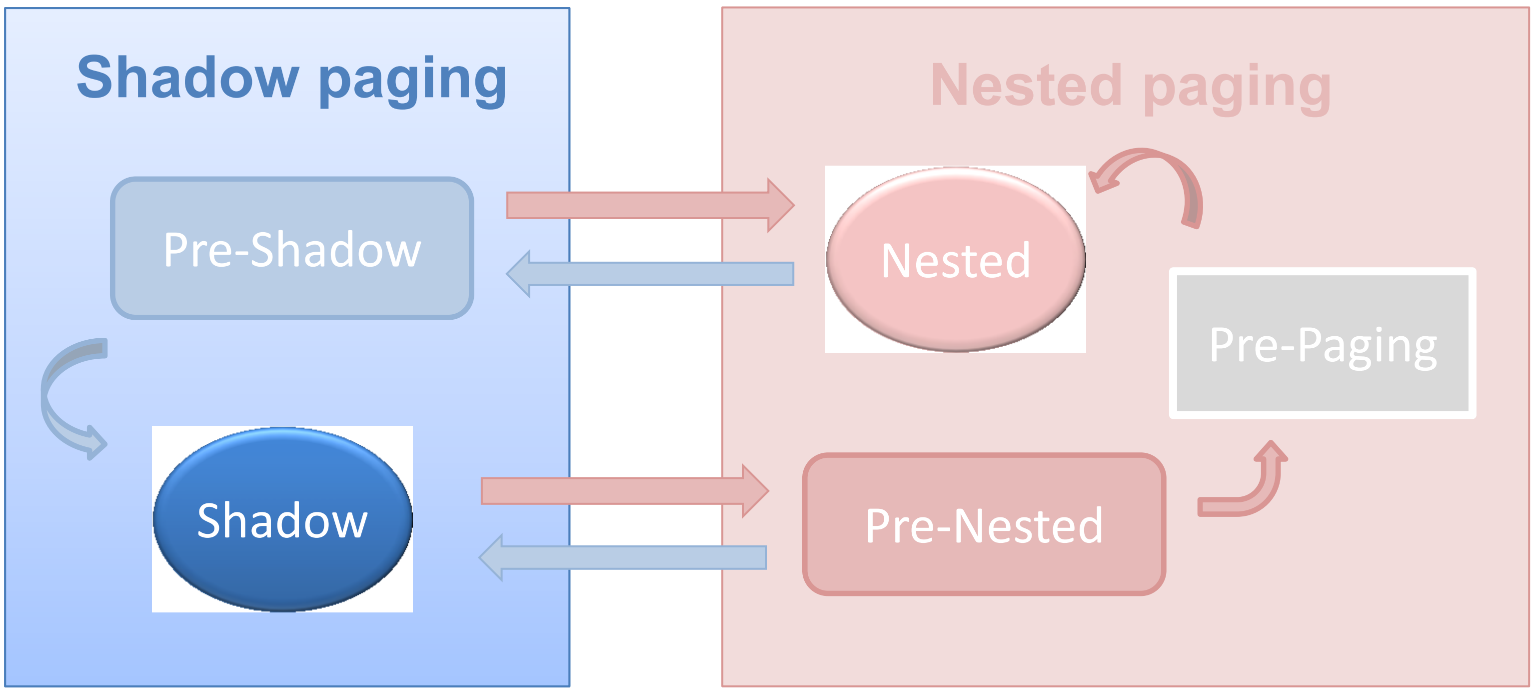
Threshold-based heuristics

- Threshold triggered mode transition

- States
  - Shadow: *monitoring VM exit frequency*
  - Nested: *monitoring TLB miss frequency*
  - Pre-Shadow: *probing shadow performance*
  - Pre-Nested: *probing nested performance*
  - Pre-Paging: *hysteresis during switch to nested paging*

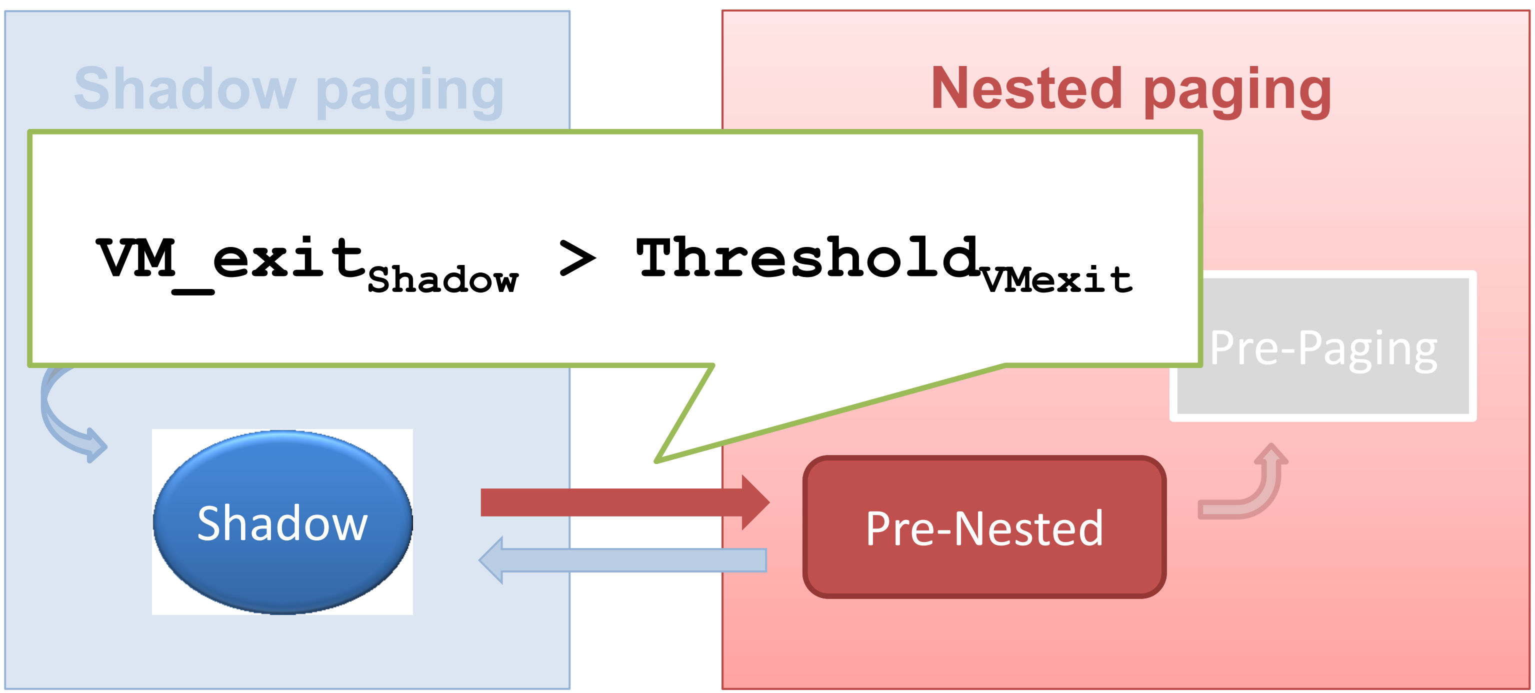
Example: begin with Shadow

- Monitoring VM exit frequency under Shadow paging



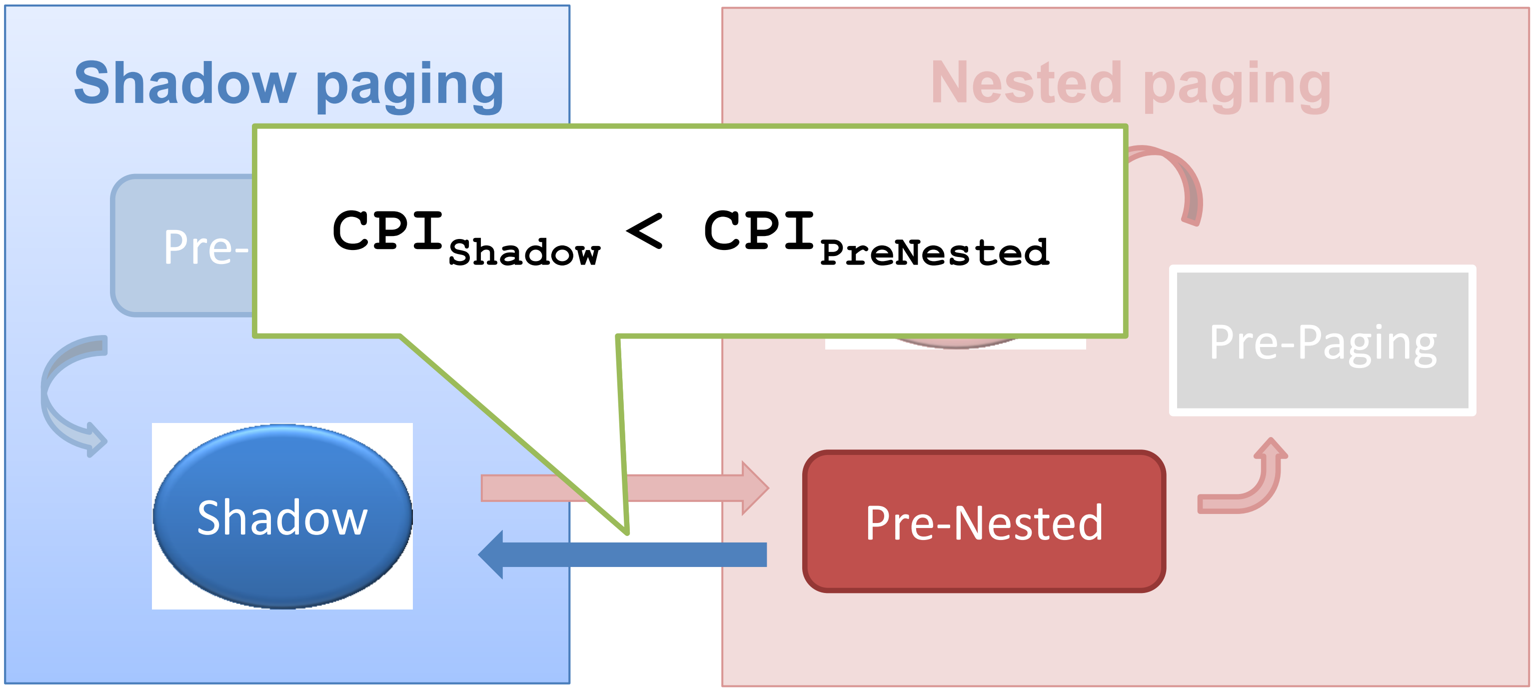
Example: Shadow to PreNested

- PF VM exit threshold triggers the transition



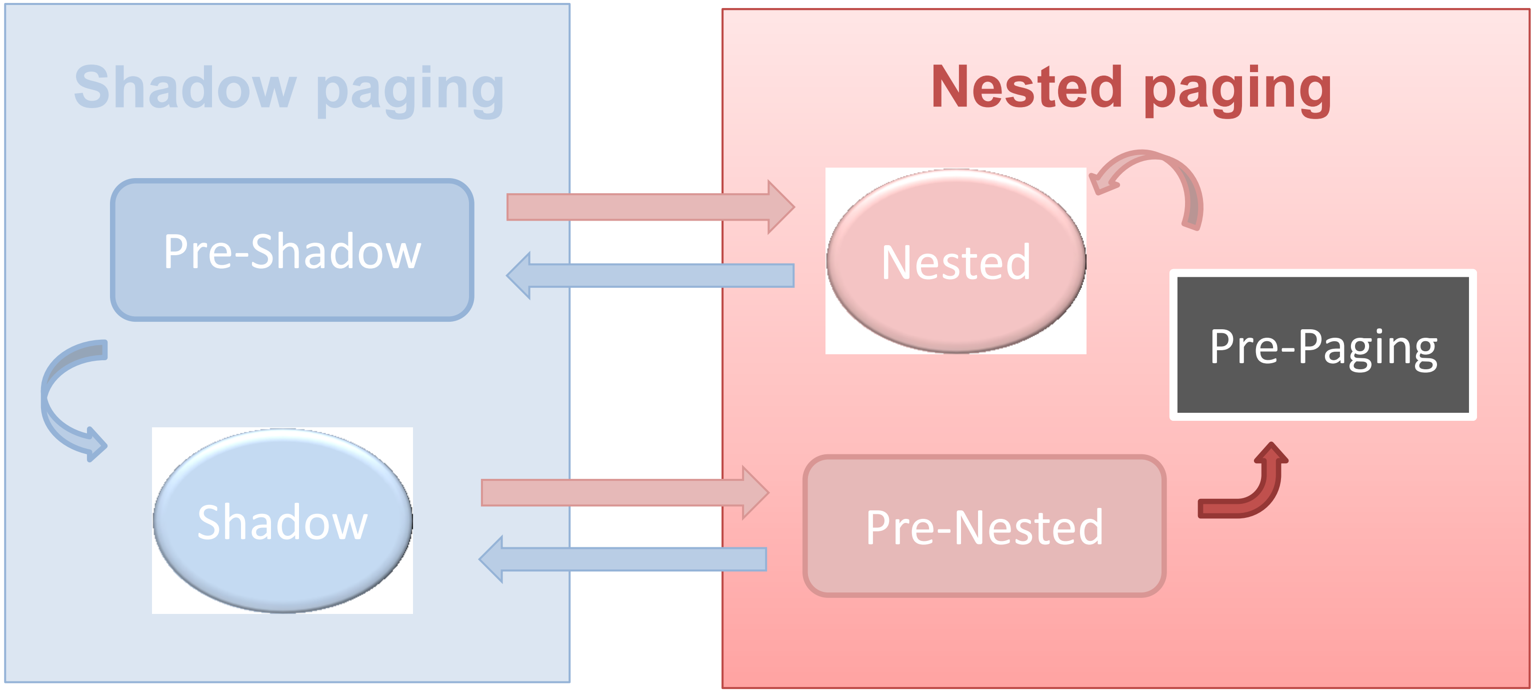
Example: PreNested to Shadow

- But, it is possible to turn back to Shadow state



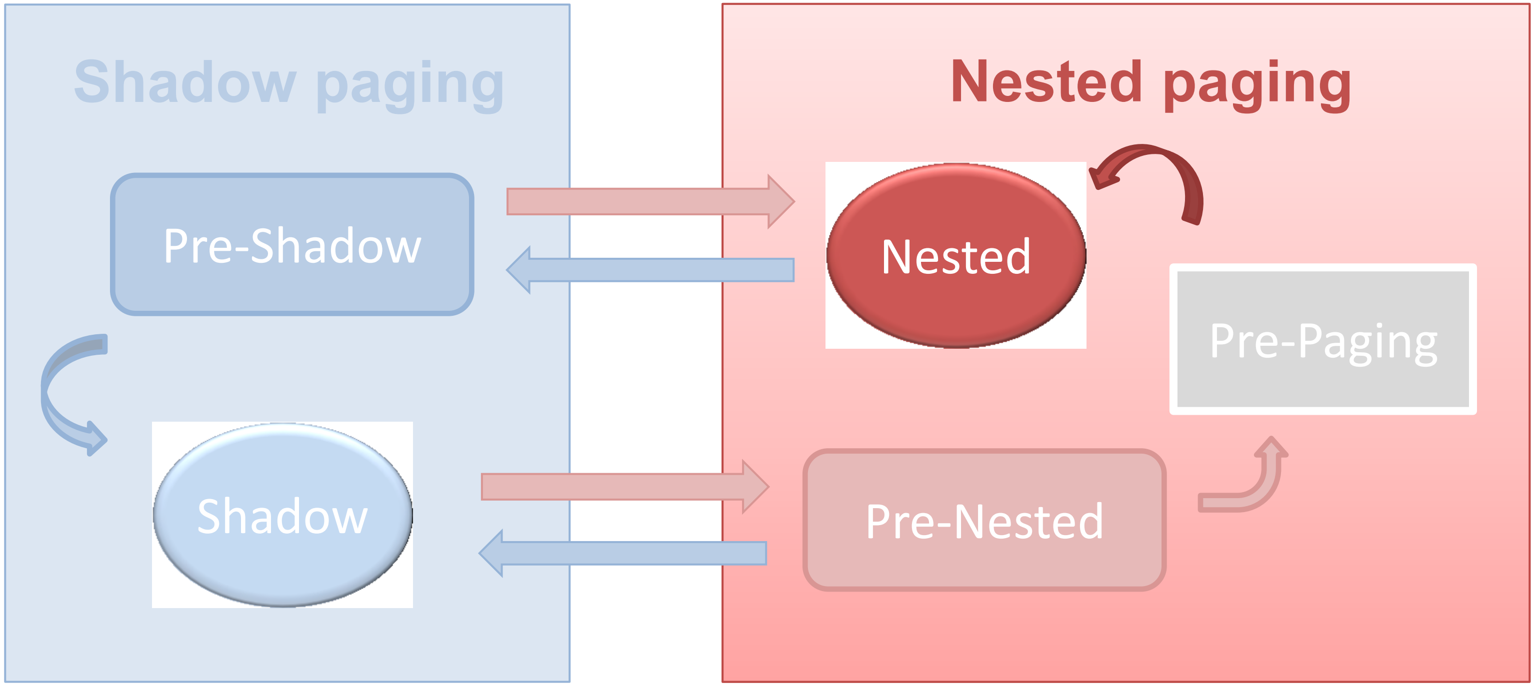
Example: Prepaging

- Probes are temporally limited
  - To avoid potential oscillations



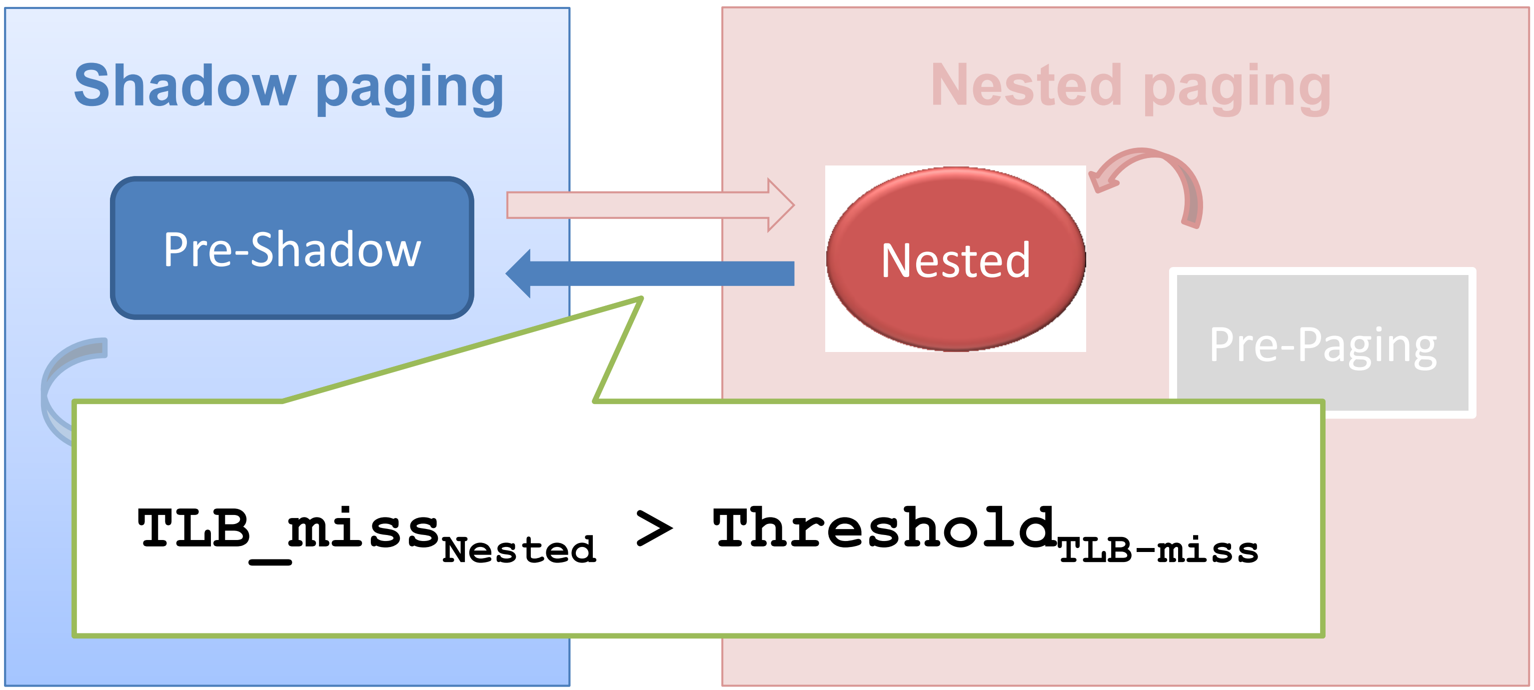
Example: Nested

- Monitoring TLB miss frequency under Nested paging



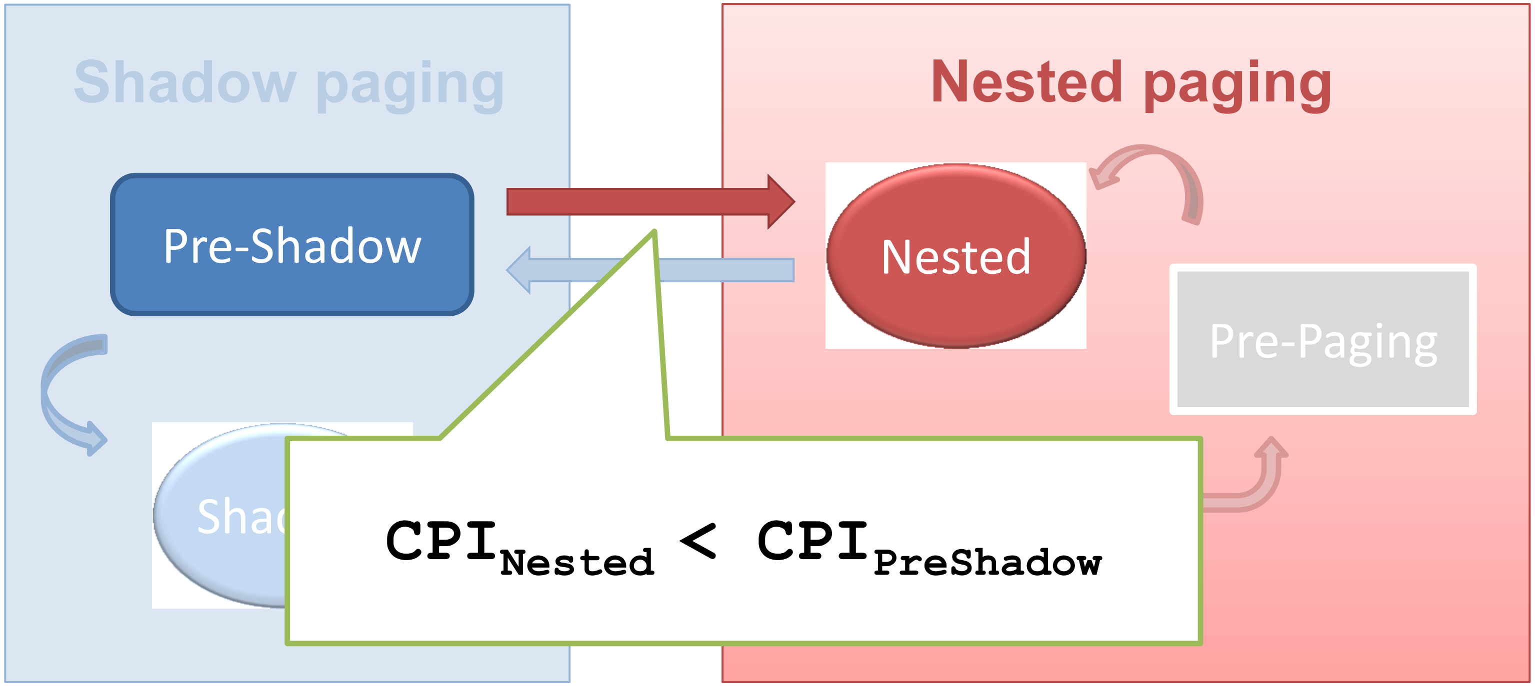
Example: Nested to PreShadow

- TLB miss threshold triggers the transition



Example: PreShadow to Nested

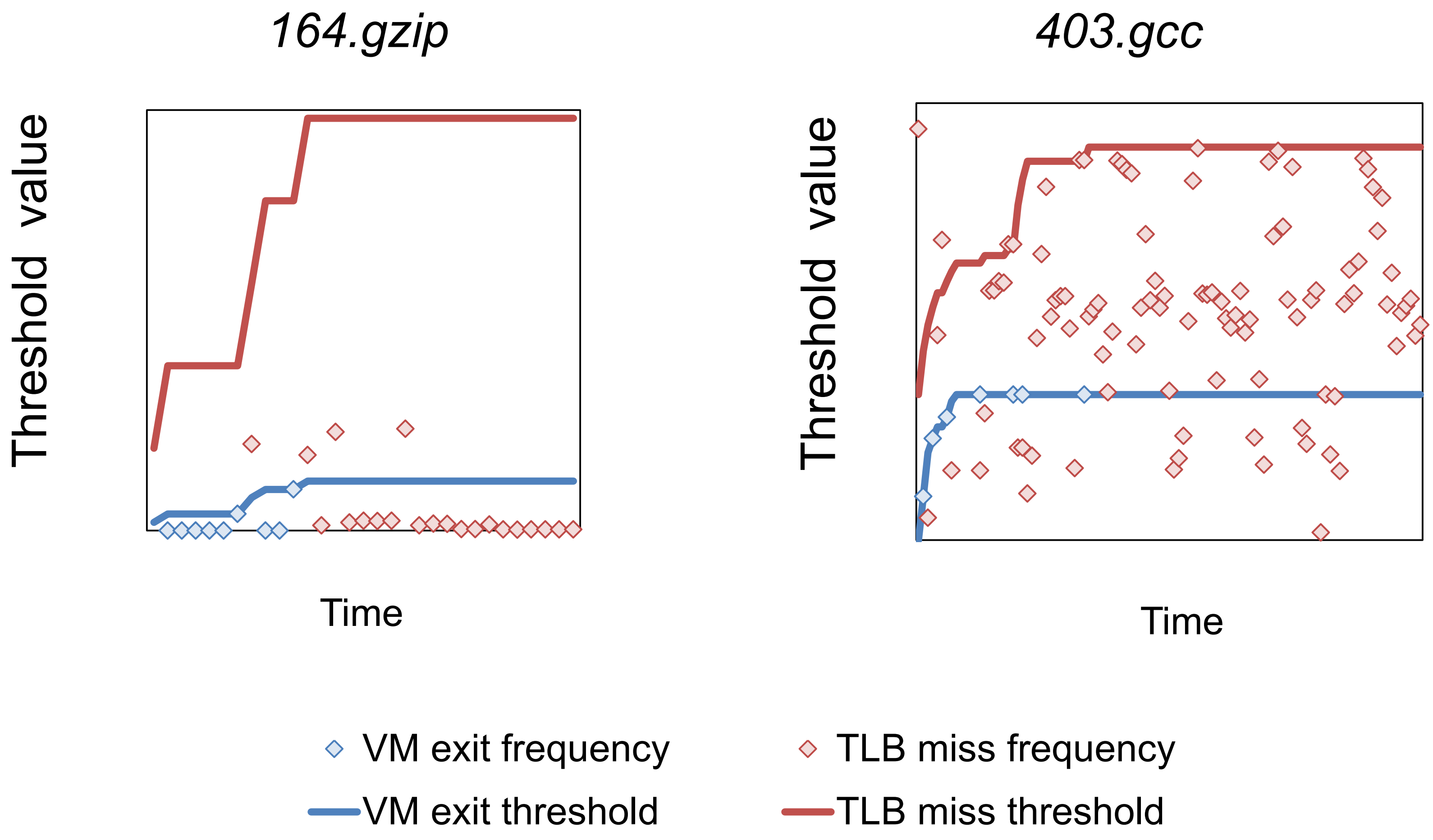
- Also, possible to turn back to Nested state



Threshold value control

- Pre-Nested
  - Increase $\mathtt{Threshold}_{\mathtt{VMexit}}$ if **`CPI`** increases
- Pre-Shadow
  - Increase $\mathtt{Threshold}_{\mathtt{TLB\text{-}miss}}$ if **`CPI`** increases
- Oscillating behavior
  - Increase both **`Thresholds`**
- Detailed algorithm in paper

Algorithm finds thresholds that result in stable behavior customized to the workload



Outline

- Introduction
- Background and Motivation
- DAV^{2}M
- Evaluation
  - Setup and Results
- Conclusion

Experimental setup

- Workload – SPEC CPU 2000/2006, PARSEC
- Software
  - Guest OS – Linux 2.6.18 (Puppy Linux 3.01)
  - VMM – Palacios
  - Host OS – Kitten
- Hardware
  - CPU – AMD Opteron 2350 2GHz
  - Memory – 2GB 667MHz (DDR2)

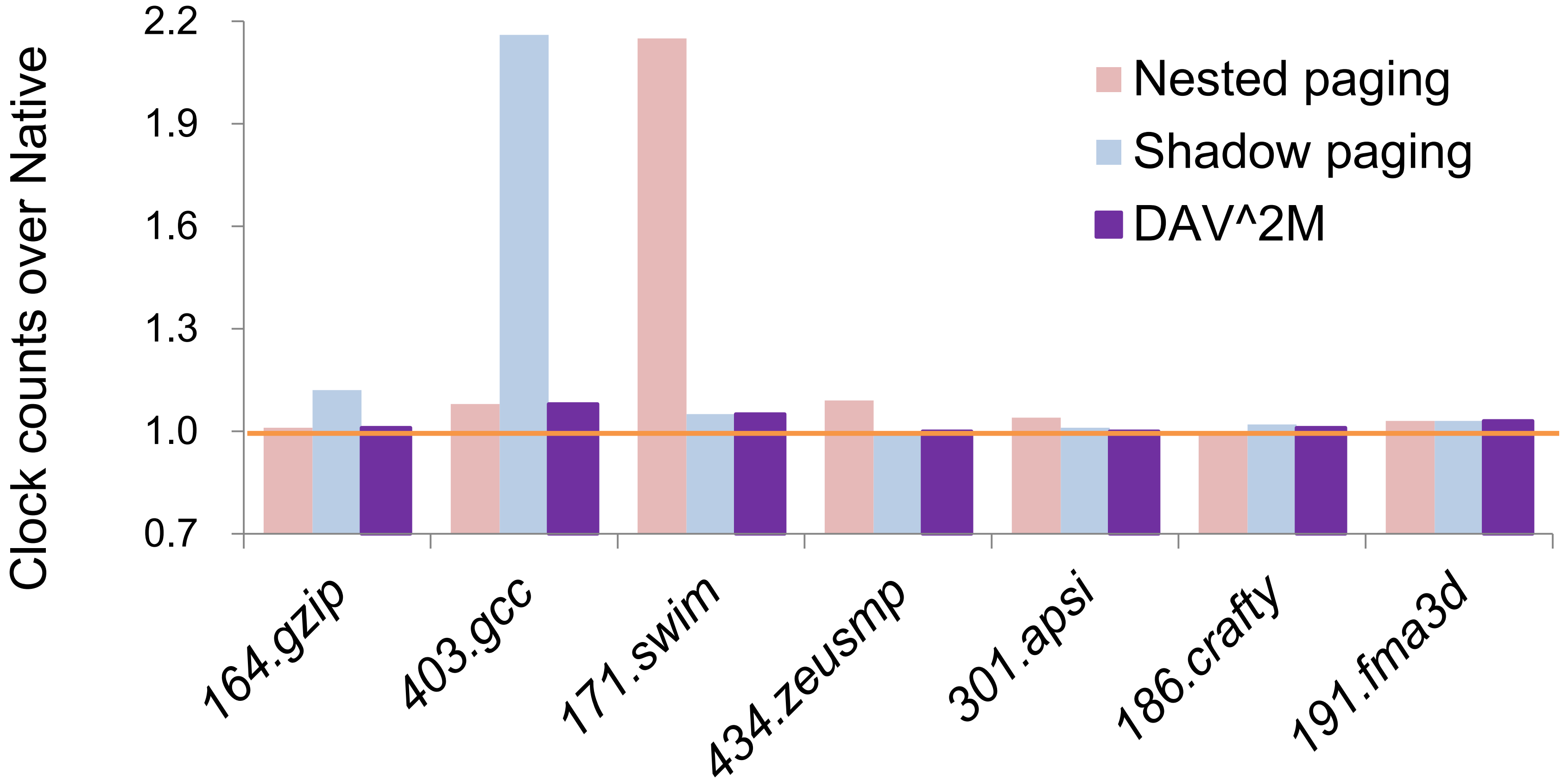
Mode switches are fast

- Worst observed case
  - 2GHz machine
  - Nested to Shadow paging: ~100ms *
  - Shadow to Nested paging: ~50ms *

* Nested page tables are *reusable*

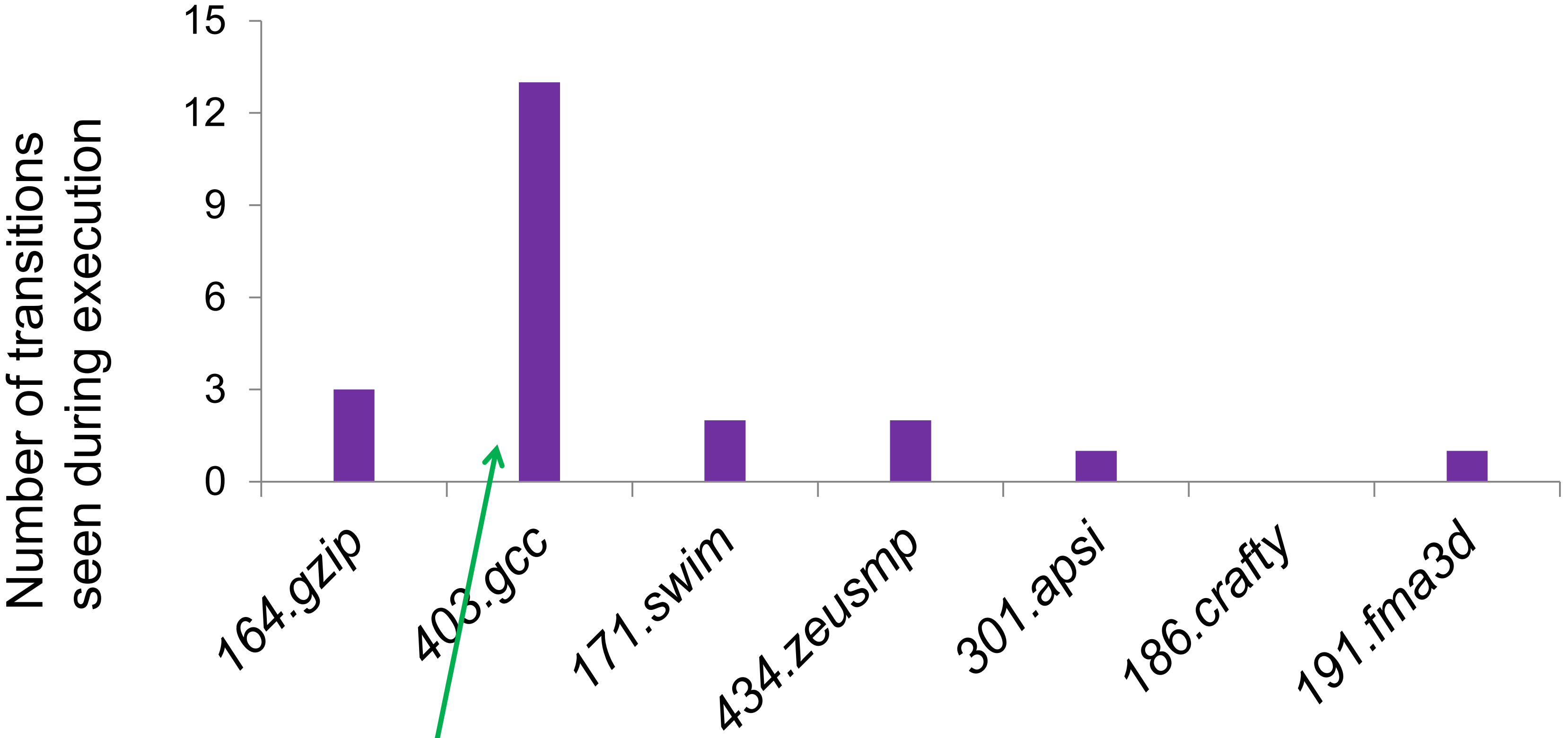
Shadow page tables must be *flushed and reconstructed*

Best of both worlds in performance



As good as the best statically chosen paging approach

Small adjustment cost



403.gcc: cost of switching is 1 sec over >3 minutes runtime

Related work

- Selective hardware/software memory virtualization
  (Xiaolin Wang et al, VEE'11)

- Enhancing nested paging
  - 2-dimensional nested page table caching
    (Bhargava et al, ASPLOS'08)
  - Hash based nested paging table (Hoang et al, CAL-Jan'10)
  - Various page table caching schemes (Barr et al, ISCA'10)

Conclusion

- No single best approach for virtualized virtual memory
  - Neither shadow paging nor nested paging
  - Choice is workload-dependent
- DAV^{2}M provides dynamic selection for the best of both worlds
  - The best paging approach for different workloads
  - Applicable to any VMM supporting multiple modes

Questions?

- Questions and Answers

- Contact information

  Chang.Bae@eecs.northwestern.edu

  http://www.changbae.org

- Project website

  http://v3vee.org