

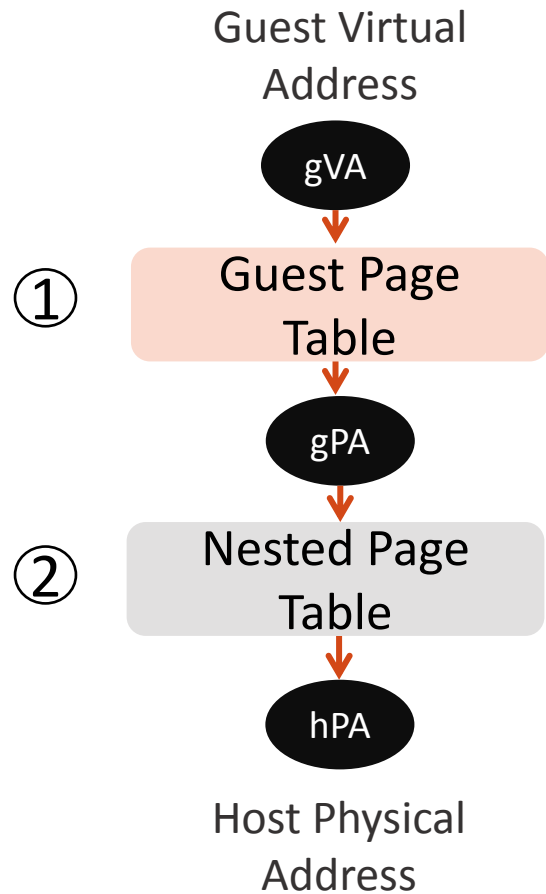
Enhancing and Exploiting Contiguity for Fast Memory Virtualization

CHLOE ALVERTI, STRATOS PSOMADAKIS, VASILEIOS KARAKOSTAS, JAYNEEL GANDHI*,
KONSTANTINOS NIKAS, GEORGIOS GOUMAS, NECTARIOS KOZIRIS

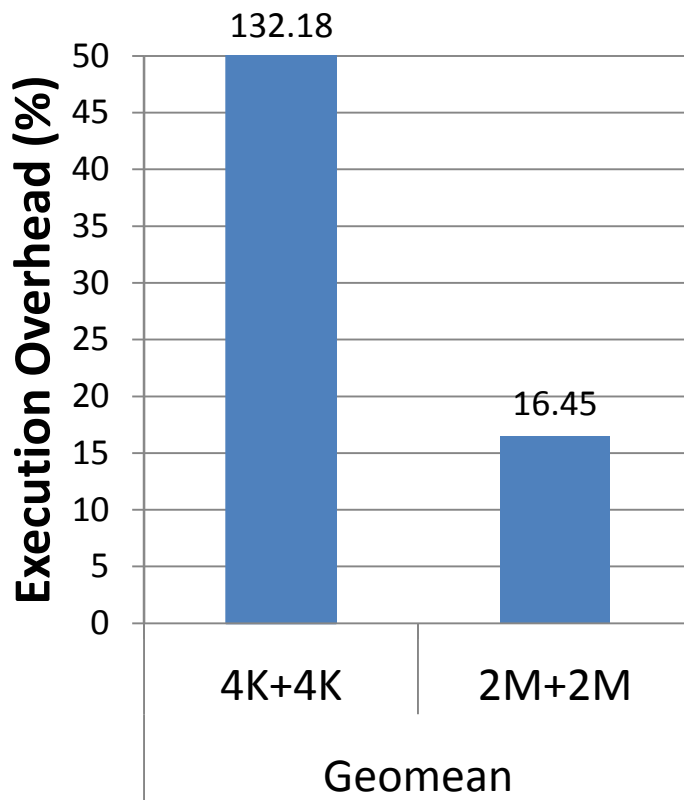
NATIONAL TECHNICAL UNIVERSITY OF ATHENS
* VMWARE RESEARCH



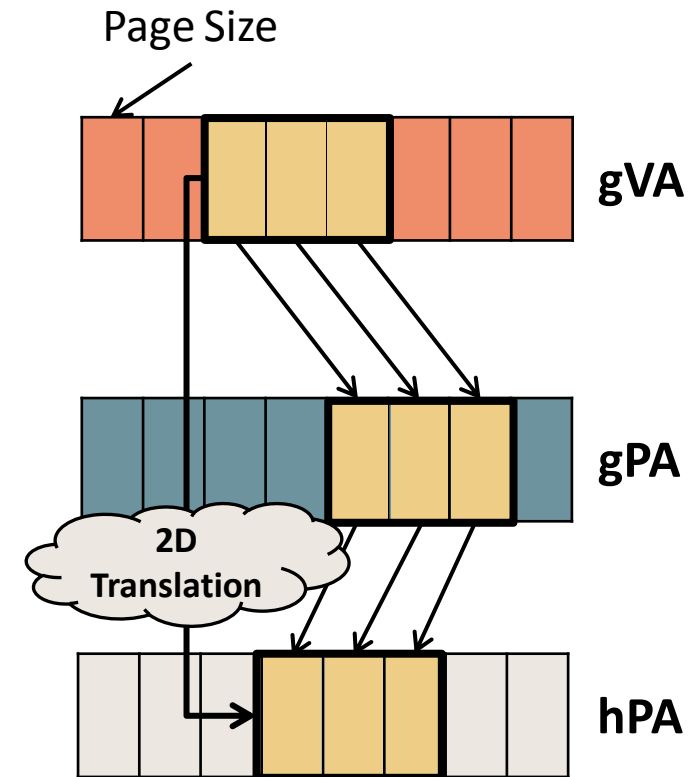
Motivation: 2D Address Translation in Virtualized Execution



Translation Overheads:
x86 nested page walk →
24 memory accesses



State of Practice:
Huge pages fail to eliminate
translation overheads



Large contiguously mapped pages

State of the Art:
Direct Segments [ISCA '13], RMM
[ISCA '15], TLB Coalescing [ISCA '17]

Contiguity-Aware Paging: Create Contiguous Mappings

Problem: Default paging allocates physical pages randomly → no contiguous page mappings

Solution: CA paging enhances the OS with **contiguous-aware page allocation** support →

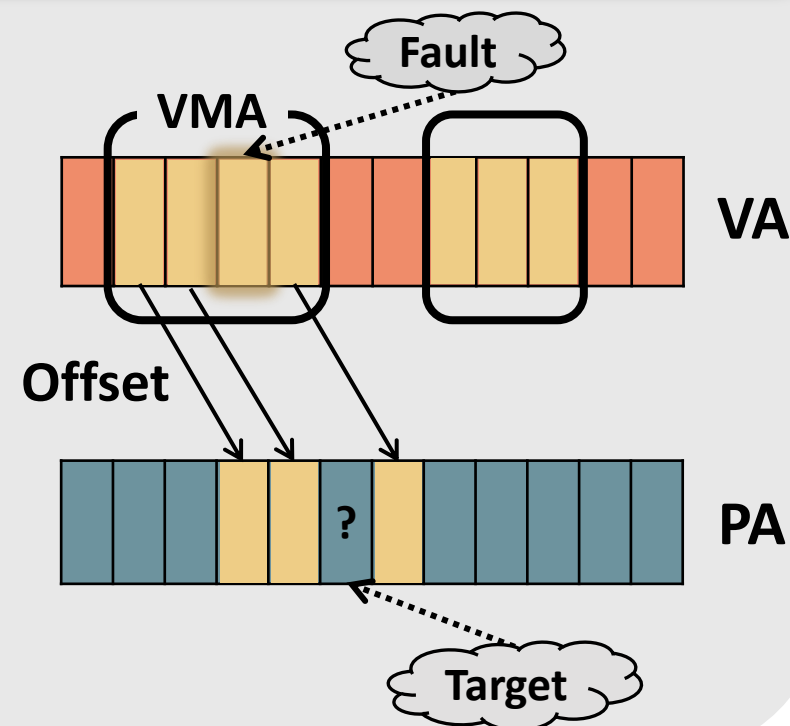
Create contiguous page mappings gradually across page faults, preserving allocation on demand

CA
Paging

VMA Offset → target page identification

Contiguity Map → track free contiguity

Placement → avoid fragmented memory



Speculative Offset Address Translation (SpOT)

Problem: Hard to track the mappings boundaries in 2 dimensions and cache their intersection

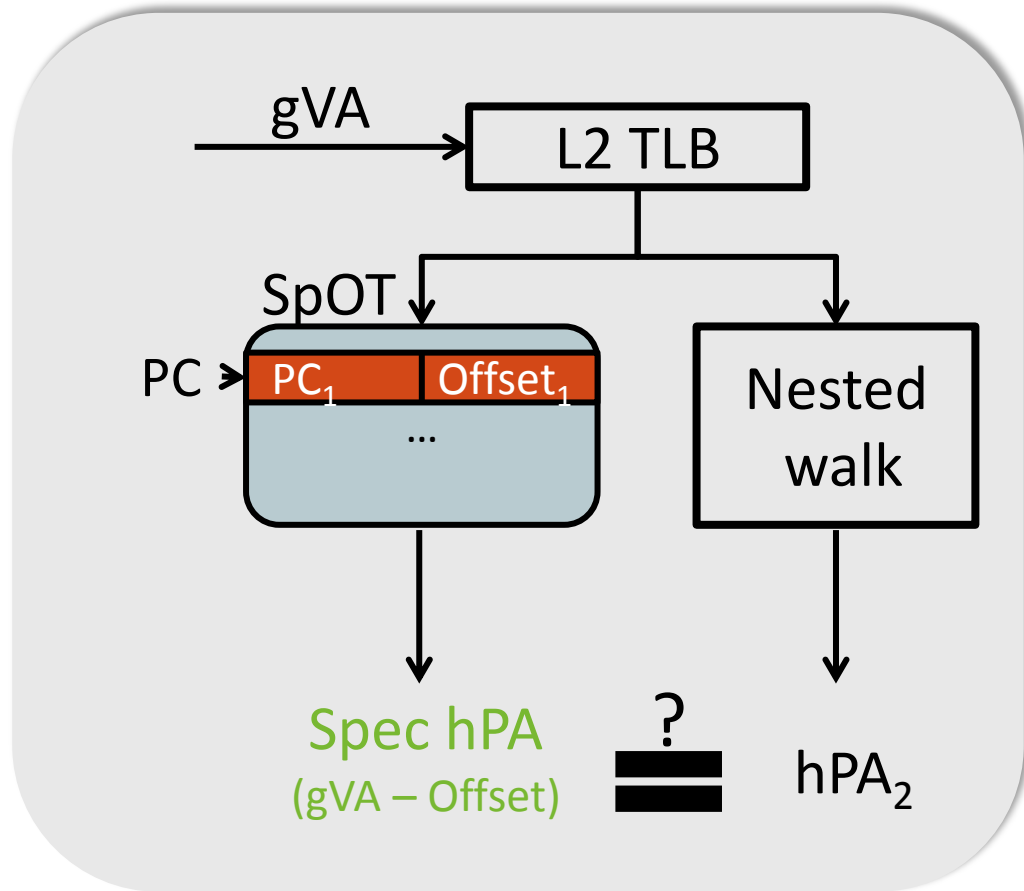
Solution: SpOT exploits CA to **predict translations** →
Hides nested page walk latency with speculative execution

SpOT

Micro-architectural extension

PC-indexed small prediction table

Tracks 2D Effective Offsets on the fly



Performance Results

CA Paging: 128 largest mappings cover

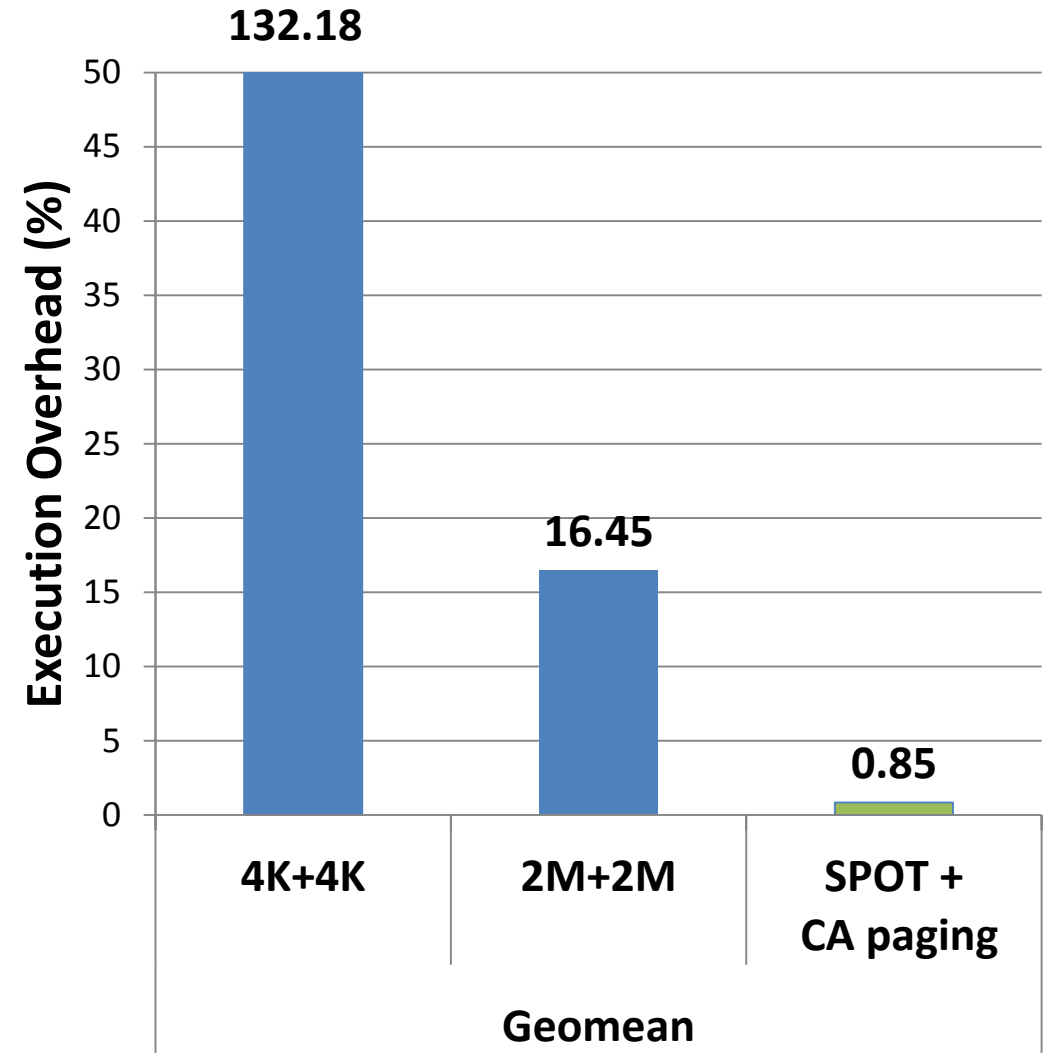
- ~97% footprint when 0% fragmentation
- ~94% footprint when 50% fragmentation

Avoids pre-allocation or unnecessary page migrations

SpOT: Combined with CA paging serves

- >90% TLB misses → correct predictions
- <1% TLB misses → incorrect predictions

SpOT avoids all the hardware complexity of a deterministic scheme



Enhancing and Exploiting Contiguity for Fast Memory Virtualization

<https://github.com/cslab-ntua/contiguity-isca2020>

**Thank you
Q&A**

