Outline

1. Background
   - AMD64 Page Walks and Caching
   - Virtualization Terminology
   - Memory Management in Virtualized Systems

2. Two-Dimensional Page Walks
   - Nested Paging + Current Paging = 2D Page Walk
   - 2D Page Walk Caching
   - Hardware and Software 2D Page Walk Acceleration

3. KVM Implementation and Results
   - KVM Software Implementation
   - Results
Four-Level AMD64 Page Walk

Virtual Address → L₄ PT Base

L₄ data PA = Data VA + PT Base

L₃ data PA = Data VA + L₄ value

L₂ data PA = Data VA + L₃ value

L₁ data PA = Data VA + L₂ value

Data PA = L₁ value

Cache Translation in TLB

Page Tables
AMD64 Processor Page Walk Caching

**Page Walk Cache (PWC)**
- In all generations of AMD64 processors
- Stores intermediate page table values
- Low-latency access

![Diagram of memory hierarchy showing page walk cache and page table walk logic](image-url)
Address Terminology

Addresses
- GVA: guest virtual address
- GPA: guest physical address
- SPA: system physical address

![Diagram showing address terminology with GUEST, GVA, GPA, and Hypervisor]
Virtualization Memory Management: No hardware support: Shadow Paging

- Guest OS can only convert GVA → GPA
- Hypervisor creates shadow page table to convert GVA → SPA
- GVA → SPA with Shadow Page Table
- Trap Guest Page Table Accesses

Diagram:
- Guest OS
- Hypervisor
- Guest Page Table
- Shadow Page Table
- GVA
- SPA
Virtualization Memory Management: Hardware support: Nested Paging

- **Benefits**: No more traps on Guest Page Table accesses
- **Drawback**: Extra page table steps add latency to TLB miss

\[
gVA \rightarrow gPA \quad \text{with unmodified guest page table}
\]

\[
gPA \rightarrow sPA \quad \text{with nested page table}
\]
Outline

1. Background
   - AMD64 Page Walks and Caching
   - Virtualization Terminology
   - Address Translations in Virtualized Systems

2. Two-Dimensional Page Walks
   - Nested Paging + Native Paging = 2D Page Walk
   - 2D Page Walk Caching
   - Hardware and Software 2D Page Walk Acceleration

3. KVM Implementation and Results
   - KVM Software Implementation
   - Results
Two-Dimensional (2D) Page Walk

- Nested page table
- Guest page table
- Cache Translation in TLB

GVA_{D} → Nested page table → GPA_{L4} → 1 → GPA_{L3} → 6 → GPA_{L2} → 11 → GPA_{L1} → 16 → GPA_{D} → 21

nL_{4} → nL_{3} → nL_{2} → nL_{1} → G
Two-Dimensional Page Walk Caching

Average 2D Walk

- **14** PWC hits
  - 5% of PWC misses
- **6** mixed PWC hit/miss
  - 25% of PWC misses
- **4** PWC misses
  - 70% of PWC misses

**BLUE** = PWC MISS

**PURPLE** = PWC MIXED

**RED** = PWC HIT

High Reuse
4 reds in a row = translation reuse

Difficult to cache
Map small memory regions
Two-Dimensional Page Walk Caching: with the Nested TLB (NTLB)

- **gL₄** NTLB Hit: Skip Nested Page Walk
- **gL₃** NTLB Hit: Skip Nested Page Walk
- **gL₂** NTLB Hit: Skip Nested Page Walk
- **gL₁** NTLB Miss: Perform Nested Page Walk
- **gPA** NTLB Miss: Perform Nested Page Walk

Simulated Results of not-exactly-real hardware – see ASPLOS08 paper
Sources of 2D Walk Overhead: L2 Cache Misses

- Many PWC misses become L2 cache misses
- 1 of 4 PWC misses also miss in L2 cache

Simulated Results of not-exactly-real hardware – see ASPLOS08 paper
Large Nested Pages & Large Guest Pages

Large Nested Pages
- Created by hypervisor
- Map 2MB instead of 4KB
- Eliminate nL₁ column
- Fewer PWC misses

Large Guest Pages
- Created by guest OS
- Map 2MB instead of 4KB
- Eliminate gL₁ row
- Fewer PWC misses

Large pages are good!
Outline

1. Background
   - AMD64 Page Walks and Caching
   - Virtualization Terminology
   - Address Translations in Virtualized Systems

2. Two-Dimensional Page Walks
   - Nested Paging + Native Paging = 2D Page Walk
   - 2D Page Walk Caching
   - Hardware and Software 2D Page Walk Acceleration

3. KVM Implementation and Results
   - KVM Software Implementation
   - Results
Nested Paging Support in KVM

- Direct mapped page table is the same as the nested page table
- Shadow Paging Code for Real Mode creates a direct mapped page table
- Nested Paging support utilizes Shadow Paging Code
- This kept implementation very simple (5 files changed, 190 insertions(+), 12 deletions(-))
- Live Migration and Guest Swapping work out of the box
- Real performance boost for KVM on AMD processors
Benchmarking - Environment

- Hardware: AMD Phenom™ 9550 2.2 GHz B3 silicon with 4GB RAM
- Host OS: Redhat Enterprise Linux 5.2
  - KVM-69
  - Xen-unstable 17731
- Guest OS: Redhat Enterprise Linux 5.1
- Guest: 2 VCPUs and 2 GB Memory
- For benchmarks on bare metal (Native) host was booted with “maxcpus=2 mem=2G”
Kernbench Performance: Shadow Paging vs. Native

Relative Performance

Native

KVM Shadow Paging
Kernbench Performance: Nested paging
KVM Nested 4KB pages

Relative Performance

Native | KVM Shadow Paging | KVM Nested 4KB pages

100 | 0 | 0
Kernbench Performance: Nested paging
Performance benefits from large pages

Relative Performance

Native | KVM Shadow Paging | KVM Nested 4KB pages | KVM Nested 2MB pages
LMBench Performance: Shadow Paging vs. Native

- Protection Fault
- Page Fault
- mmap latency

Slowdown relative to native

- Native
- KVM Shadow Paging
- KVM Nested 4KB pages
- KVM Nested 2MB pages
LMBench Performance: Nested paging
KVM Nested 4KB pages

- Protection Fault
- Page Fault
- mmap latency

Slowdown relative to native

- Native
- KVM Shadow Paging
- KVM Nested 4KB pages
- KVM Nested 2MB pages
LMBench Performance: Nested paging

Performance benefits from large pages

- Protection Fault
- Page Fault
- mmap latency

- Native
- KVM Shadow Paging
- KVM Nested 4KB pages
- KVM Nested 2MB pages
Conclusion

Nested Paging

- A HW solution to reduce memory management overhead
- Also introduces overhead on TLB misses

Hardware overhead can be significantly reduced

- Nested TLB to skip nested page walks and Page walk cache
- Approach native speed with these techniques

Overhead elimination more difficult

- Some 2D walk references always miss in PWC and L2 cache
- Exclusive use of 2MB pages in hypervisor is difficult

KVM Implements Nested Paging

- Performance improves and memory footprint shrinks
- Best performance from use of large nested page sizes
Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2008 Advanced Micro Devices, Inc. All rights reserved.