Remove hardware-assisted page walk overhead for virtualized systems using memory contiguity property of VMs

Alain Tchana & Boris Teabe

Insitut National Polytechnique de Toulouse - IRIT

2018

Plan

1. Memory address translation
2. Contributions:

Sous-section 1

1. Memory address translation
2. Contributions:

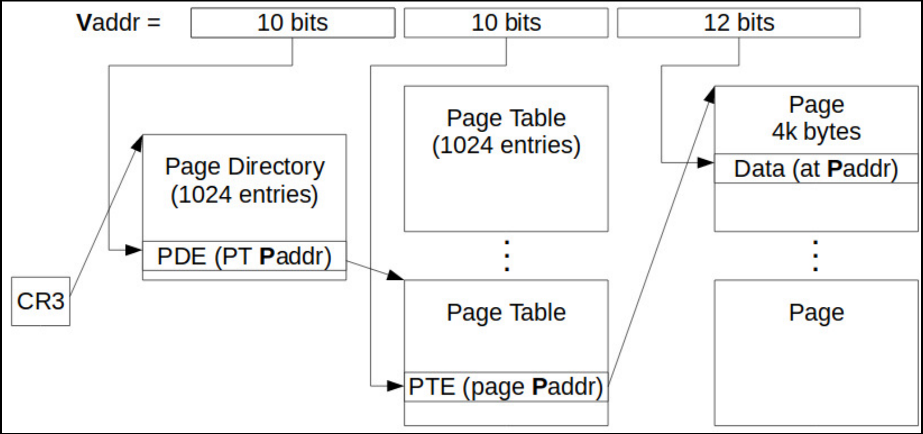
Paging

Principle

- Each process has a virtual address space
  - 0 to 2^{32} (in 32-bit machines)
  - 0 to 2^{64} (in 64-bit machines)
- Main memory addresses are called host physical address (hPA)
- Process addresses are called guest virtual address (gVA)
- Each process address space is organized in memory pages
  - 4KB, 2MB (large page)

Native systems

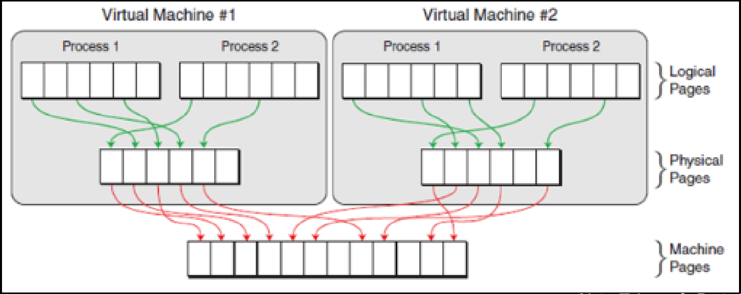
$gVA \rightarrow hPH$



Virtualized systems

Address Translation for VMs

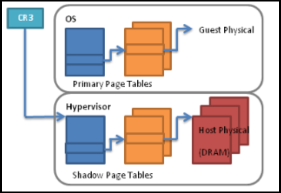
- Guest virtual address (gVA) $\rightarrow$ guest physical address (gPA) $\leftarrow$ host physical address (hPA)
- Two approaches: Shadow paging and Hardware assisted virtualization



Virtualized systems

Shadow Paging

- Hypervisor maintains a separate mapping from gVA $\rightarrow$ hPA
  - Hypervisor intercepts every attempt by guest to update or install a page table (performance)
  - Per process mapping (space)
- This approach incurs lots of page faults and VM exits



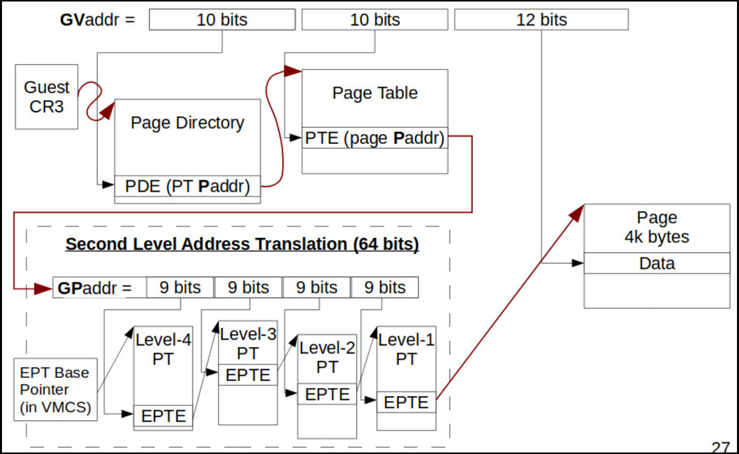
Virtualized systems

Hardware assisted virtualization

- This approach eliminates the need for shadow paging
- And provides architectural support for a new layer of address translation (also called Nested Level)
- A new hierarchyof paging: gPA $\rightarrow$ gPA
- Two dimensional Page Table
  - Nested Page Table (NPT) by AMD or Extended Page Table (EPT) by Intel

Virtualized systems

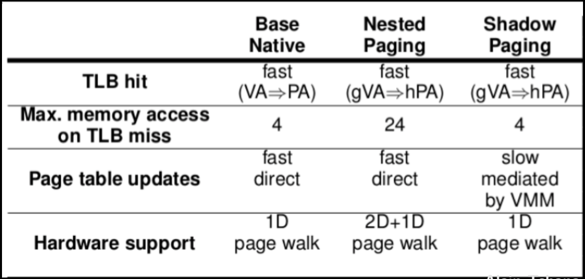
Hardware assisted virtualization: gPA → hPA



Virtualized systems

Hardware assisted virtualization: Overhead

- Each TLB miss leads to 24 memory access
  - Guest page table walk
  - Nested or Extended page table walk
- TLB miss intensive applications are penalized, in comparison to software based MMU (Shadow paging)



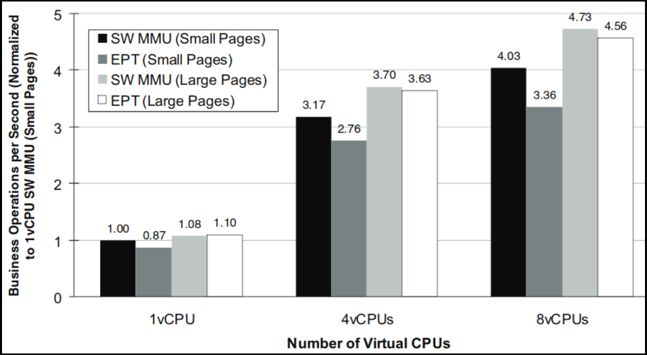
	Base Native	Nested Paging	Shadow Paging
TLB hit	fast (VA⇒PA)	fast (gVA⇒hPA)	fast (gVA⇒hPA)
Max. memory access on TLB miss	4	24	4
Page table updates	fast direct	fast direct	slow mediated by VMM
Hardware support	1D page walk	2D+1D page walk	1D page walk

Virtualized systems

Hardware assisted virtualization: Overhead

from https://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf

- Higher is Better



Related work

Existing solutions

- To increase TLB size, using the CPU cache (ISCA '17)
- Direct-segment (ISCA '12, ISCA '17)
- Large pages (ASPLOS '08)
- Caching nested page walk (MICRO '14)
- Switch between Nested or Extended page table $\Leftrightarrow$ Shadow page table (ISCA '14)

Sous-section 2

1. Memory address translation

2. Contributions:

Basic idea

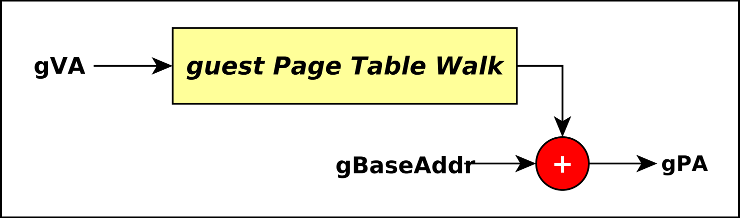
Single page table walk

- Remove EPT
- The guest's page table is sufficient when the host memory of a VM is contiguous
- gVA $\rightarrow$ hPH is computed as follows
  - gVA $\rightarrow$ gPA
  - gPA + gBaseAddr = hPA
  - gBaseAddr is the host address start for the VM in the host memory
- This idea is inspired by "*Do-It-Yourself Virtual Memory Translation*", ISCA'17

Basic idea

Single page gable walk

- For a contiguous memory VM:
  - The VM OS is in charge of handling his page tables without hypervisor interfering
  - Single page walk after a TLB miss



A favorable context: Contiguous memory VM

Existence of VMs with contiguous Memory

- Simulate the hypervisor VM memory allocator at boot (Xen hypervisor)
  - allocates contiguous memory region
- Simulate two memory placement algorithms for VMs
  - naive placement which only takes into account resource availability
  - contiguity aware placement which prioritises machines with contiguous memory regions
- Run the simulation on Microsoft Azure traces
  - take into account VM arrival rate
  - count the number of VMs having contiguous memory

Challenges

Memory isolation

- (C_1) Each VM must be confined within its memory region
  - How to ensure isolation, knowing the hypervisor does not interfere?
- (C_2) How to enforce contiguity?
- (C_3) Cohabitation of our solution with EPT
  - Contiguous memory VMs use our solution
  - Other VMs use EPT

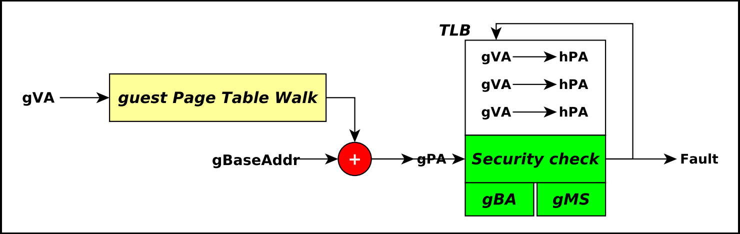
Contributions

Memory isolation

- Introducing a base address (noted **gBA**) for each VM (included within the VMCS)
- Once scheduled, the VM's **gBA** is loaded within a dedicated processor register (guest base address register).
- Also, the VM's memory size (noted **gMS**) is stored in a dedicated processor register .
- To ensure isolation (no memory violation), the TLB is enhanced with a security module
  - during insertion in TLB, the security module checks if **gBA** $\leq$ hPA $\leq$ **gBA+gMS**

Contributions

Memory isolation



Contributions

Memory contiguity enforcement: in the datacenter

- We improve the VM placement system within the datacenter in order to prioritize machines offering contiguity to the VM
- In addition to classical monitoring informations (CPU, memory), each compute node sends the size of the contiguous memory regions
- Placement system rely on these information to place

Contributions

Memory contiguity enforcement: on a server

- We change a VM from discontinuous memory to contiguous memory at runtime
  - The VM boot with discontinuous memory
  - Once possible, the hypervisor reorganizes the VM's memory to be contiguous
  - This can be done locally (migrating the VM's pages during idle period), or migrate the VM to another host