Virtualizing Memory

CMPE283
Slide Presentation 7

Agenda

- Review
- Memory Virtualization

But First...

- A review...
- Hardware assisted virtualization (VT-x)
  - Interception – how?
  - Redirection – how?

Managing Memory

- It's the VMM's job to manage memory
  - After all, it's just another hardware resource
- The VMM's responsibilities include
  - Tracking which pages have been assigned to which VM
  - Reclaiming / Swapping pages as needed
  - Ensuring memory maps are correct

Managing Memory

- How do these responsibilities compare to the MM responsibilities of an OS?
  - Same?
  - How did the OS keep a process from mismanaging its memory map / layout?

Managing Memory

- Can we let the VM “manage” its own memory map?
  - Eg, direct control over CR3
  - What does this do to the memory layout from the processor's view?
  - When does the VM need to manipulate CR3 anyway?

Managing Memory

- Two modes of VMM memory management
  - Shadow paging
  - Nested paging

- Shadow paging
  - First implementation
  - Virtual TLB and “brute force” methods
  - VMM needs to be more involved

Managing Memory

- Nested Paging
  - EPT / RVI
  - Introduced in later CPUs (~2009)
  - VMM is less involved
  - Guest manages own page tables (CR3)

Shadow Paging

- Two approaches
  - Brute force
  - Virtual TLB

- Described in Intel SDM Vol. 3 Ch. 32.3.3 – 32.3.5

Brute Force Approach

- The approach nobody uses
  - Intel SDM Vol. 3 Ch. 32.3.3
  - Intercept and fake everything involving the MMU

- “All this implies considerable overhead that should be avoided”

Virtual TLB

- The other mode of shadow paging is the virtual TLB mode
  - Guest software modifies its own page tables
  - Exits taken on CR3 manipulation and #PF (and various other paging-related things)
- The *actual, in-use* CR3 and paging structure is managed by the VMM

Virtual TLB

- Intially, the VMM starts with a page table with all entries marked as not present
  - This forces a #PF exit on each guest memory access
- On each #PF, various parts of the guest page table hierarchy is loaded into the active (VMM-managed) table
  - Side-by-side walk of each table

Virtual TLB

- Failing instruction is then re-executed
  - This results in a TLB load
  - No further access to the table is needed until this entry is flushed

Virtual TLB

- Flushes
  - Guest VMs can flush their TLB (virtually) several ways
  - INVLPG instruction
  - MOV CR3
  - MOV CR4 (some bits trigger TLB flushes)
  - … etc …

Virtual TLB

- Flushes are handled by the VMM
  - Exit on MOV CR3 / MOV CR4 / INVLPG
  - Mark the corresponding entry in the active table as not present (INVLPG)
  - Reload the entire active table as a fully-empty one (MOV CR3/4)
  - Next access causes a new #PF, simulating a TLB flush

Virtual TLB

- Intel SDM Vol. 3, Ch 32.3
  - Figure 32-1 for more info

Nested Paging

- A newer (better?) model of MMU virtualization
- Present in newer CPUs
- Called EPT (Intel), RVI (AMD)
  - Sometimes called “SLAT”
  - Second Level Address Translation

Nested Paging

- In a traditional system …
  - CR3 points to a page directory
  - … which maps virtual addresses to physical addresses (VAs → PAs)
  - … using a 3 or 4-level table lookup
- Translations from this directory are cached after lookup in the TLB

Nested Paging

- In a nested paging system …
  - Two separate page directories are used
  - … one to map guest virtual addresses to guest physical addresses
  - … and a second to map guest physical addresses to host physical addresses
- Using a second directory hides the underlying physical memory layout from the VM

Nested Paging

- Using two directories provides a few benefits
  - … the VM can manage its own directory without interference (or even monitoring) by the VMM
  - Easier implementation inside the VMM
- … and a few drawbacks

Nested Paging

- Concerns
  - Each memory access by the guest may now need to go through the page table walk *twice*
  - … for *each access*
- Think about what this does to a simple instruction like:
  - MOV %RAX, mem_loc
  - How many memory accesses could this produce?
    - Does NP make this worse?

Nested Paging

- More memory accesses
  - … with each of those accesses also consuming a TLB entry
- More TLB pressure

- Can we make it better?

Nested Paging

- CPU improvements
  - VPIDs / ASIDs
    - Help TLB utilization, avoid excessive flushing
- EPT pointer switching
- #VE

Shadow Paging vs. Nested Paging

- Shadow paging models are complex
  - More code, more MMU interactions
- Nested paging models are easier to implement
  - … but in a select few cases, not the best choice, performance-wise

Shadow Paging vs. Nested Paging

- Most new VMMs just focus on NP
  - NP is as good or better in most workloads
- Some even require it
  - Eg, client Hyper-V (Windows 8 and later)
- “Legacy” VMMs may still have SP around for workloads that benefit from it

Reading

- Intel SDM Vol. 3
  - Ch 32.1, 32.3

- Next time:
  - Devices and VT-d / IOMMUs