

Intel VT-x and VT-i overview

Issues with traditional VMM designs

- ❑ Ring depriving:
 - A technique that runs all guest software at a privilege level greater than 0.
 - 0/1/3 model : guest OS runs at level 1
 - Can't be used for IA-32 processors in 64-bit mode
 - 0/3/3 model : guest OS runs at level 3
 - Level 3 shared with apps
 - → Ring compression
- ❑ Ring aliasing:
 - Problems due to not running a guest OS at its intended privilege level
 - Guest OS can tell it is not at level 0

Issues with current VMM designs

- ❑ Address space compression
 - VMM needs to occupy part of the guest's virtual address space
 - How to protect this part of virtual address space from guest?
- ❑ Non-faulting accesses to privileged state
 - Some accesses to privileged resources do not fault
 - E.g. LGDT, LIDT, LLDT etc will fault,
 - But SGDT, SIDT, SLDT etc will *not* fault
- ❑ System calls
 - SYSENTER will transition to ring 0, where VMM executes
 - SYSEXIT will fault if executed outside ring 0
 - Can end up with too many transitions via VMM at ring 0

Issues with current VMM designs

- ❑ Interrupt virtualization
 - Guest attempts to control interrupt masking will fault due to ring depriving
 - That is fine, BUT
 - Guest OSes frequently mask/unmask interrupts.
 - So it's a big performance overhead
 - VMM intervention not needed for all such accesses
- ❑ Hidden state problem
 - Some processor state hidden from all software
 - Not accessible via registers
 - E.g. hidden descriptor caches
 - Not a problem if only a single OS executes
 - BUT, for better performance, one may want to save this hidden state when switching between guest OSes

VTx and VTi Overview

VT-x: Ring Aliasing and Ring Compression

- ❑ Applications execute in ring 3
- ❑ Guest OS executes *deprivileged* in ring 0
- ❑ VMM executes in a new mode with highest privilege

VT-x: Two operating modes

- ❑ VMX root operation:
 - Fully privileged mode for VMM execution
- ❑ VMX non-root operation:
 - Not fully privileged, intended for guest OS
 - Deprivileges guest OS without using up rings for the VMM
- ❑ These modes are orthogonal to rings

VT-x: VM Entry and VM Exit

- ❑ New Data Structure: Virtual Machine Control Structure (VMCS)
 - Guest State Area
 - Host State Area
 - Analogous in some respects to Task State Structure (TSS) for processes and used during context switches.
- ❑ VM Entry:
 - Transfers control from VMM to guest
 - Via two instructions:
 - VMLAUNCH on first entry
 - VMRESUME on subsequent entries
 - Loads processor state from guest state area in VMCS
- ❑ VM Exit:
 - Transfer control from guest to VMM
 - Via VMEXIT instruction
 - Save processor state to guest state area in VMCS
 - Load VMM state from the host state area in VMCS

VT-x: Virtual Machine Control Structure (VMCS)

- ❑ One control structure per virtual processor
- ❑ Located in physical memory
- ❑ Contains control information for
 - VM execution
 - EXIT conditions and exit information
 - ENTRY conditions
- ❑ VMCS format not defined (?)
 - Perhaps dependent on the implementation of hypervisor

VT-i: Two operating modes

- ❑ Determined by PSR.vm (processor status register)
 - PSR.vm = 0 → VMM operation
 - PSR.vm = 1 → Guest operation
- ❑ Controls the number of virtual address bits available to the guest OS.
 - Uppermost address bit not available to guest OS
 - → gives VMM a dedicated virtual address space
- ❑ Mode switching:
 - Access to privileged resources by the guest software results in *intercepts* to the VMM. (similar to VM Exit in VT-x)
 - Privileged resources: TLB, Privileged registers (PSR, control), etc.
- ❑ Virtual Processor Descriptor
 - One per virtual processor
 - Similar purpose to VMCS in VT-x

Solutions to virtualization challenges

Address space compression

- ❑ VT-x
 - Two additional modes mean that guest OS can run at level 0 in non-root mode
 - But does VMM's code still map to guest address space?
- ❑ VT-i
 - PSR.vm bit serves a similar purpose
 - VMM has a virtual address bit that guest software cannot use

Ring aliasing

- ❑ Allows VMM to run at its intended privilege level
 - Ring 0
 - VT-x → in non-root mode
 - VT-i → with PSR.vm set to 1

Non-faulting access to privileged state

- ❑ Two mechanisms
 - Support to have such instructions fault to VMM
 - No more scanning and patching needed!
 - Support that causes the state to become unimportant to VMM
 - E.g. GDT/LDT/IDT operations in IA-32

System calls

- ❑ Guest runs at level 0
- ❑ So SYSENTER/SYSEXIT transition via VMM not required

Interrupt virtualization

- ❑ Ability to control instructions and events that cause VM exits
- ❑ VT-x
 - External interrupt exiting control
 - When set, all external interrupts cause VM exits.
 - Guest unable to mask these interrupts
 - Interrupt window exiting
 - VM exit occurs whenever guest software is ready to receive interrupts
- ❑ VT-i
 - Virtualization acceleration field
 - Can allow/prevent guest from controlling interrupt masking
 - Reduce transitions to VMM on each access to interruption control registers.
 - Virtualization disable field
 - Allows VMM to disable virtualization of a particular instruction or resource.
 - Reduces the number of intercepts that VMM handles

Access to hidden state

- ❑ VT-x
 - Guest state area of VMCS includes hidden state information
- ❑ VT-i
 - Similar mechanisms to save necessary hidden state information

Other more recent features

- ❑ Extended Page Tables
 - Support for nested paging
- ❑ Virtual Processor IDs
 - To tag TLB entries
 - Avoids TLB flush on each VM Entry and VM Exit