第 3 讲：Virtual Machine Monitor

第三节：Hardware-assisted Virtualization

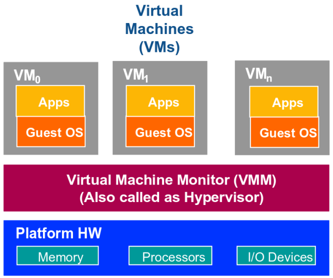
陈渝

清华大学计算机系

yuchen@tsinghua.edu.cn

2020 年 2 月 29 日

Problems



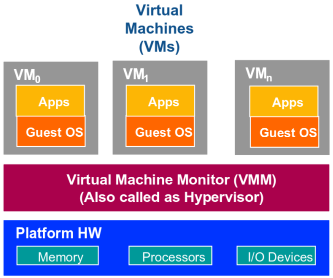
Virtualization holes

Conventional Intel® 64, or IA-32, is not virtualizable architecture

- Sensitive register instructions: read or change sensitive registers and/or memory locations such as a clock register or interrupt registers
  - SGDT, SIDT, SLDT
  - SMSW
  - PUSHF, POPF
- Protection system instructions: reference the storage protection system, memory or address relocation system
  - LAR, LSL, VERR, VERW
  - POP, PUSH
  - CALL, JMP, INT n, RET
  - STR, MOVE

Source from proceeding of 2000 USENIX ATC

Problems

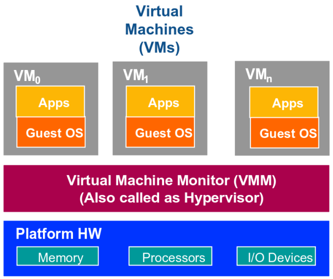


Controlling the CPU Resource

With an Intel® 64 CPU, a VMM must be able to retain control over

- Access to privileged state (CRn, DRn, MSRs)
- Exceptions (#PF, #MC, etc.)
- Interrupts and interrupt masking
- Address translation (via page tables)
- CPU access to I/O (via I/O ports or MMIO)

Problems

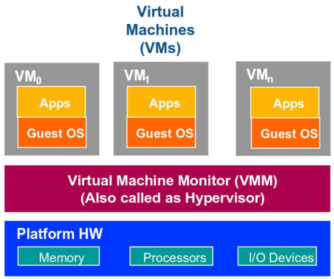


CPU Control via “Ring Deprivileging”

With an Intel® 64 CPU, a VMM must be able to retain control over

- Guest OS kernel runs in a less privileged ring than usual
- VMM runs in the most privileged ring 0
- prevent guest OS from accessing privileged instructions / state
- prevent guest OS from modifying VMM code and data

Problems

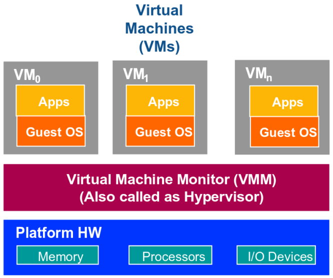


Problems with Ring Deprivileging

Ring deprivileging encounters various issues in current Intel® 64 architecture:

- Ring compression/aliasing
- Non-faulting reads of privileged state
- Excessive faulting
- Interrupt-virtualization issues

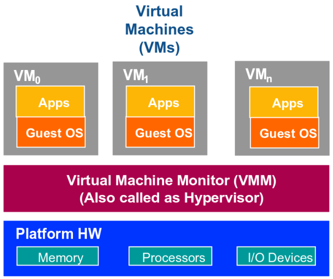
Solutions



Addressing "Virtualization Holes"

- Paravirtualization
- Binary translation (or patching)
- Hardware-assisted virtualization
  - Closing virtualization holes in hardware
  - Simplify VMM software
  - Optimizing for performance

Solution – hardware-assisted VT

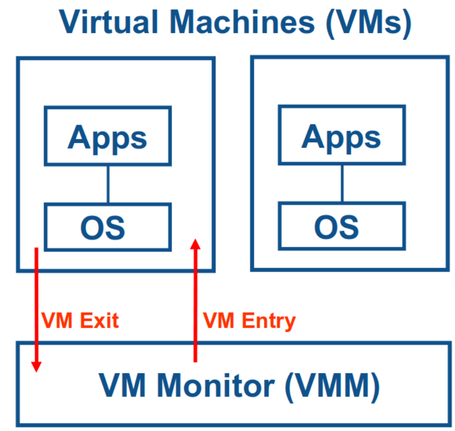


Intel® Virtualization Technology

A hardware-assisted virtualization technology, named as Intel® Virtualization Technology, or Intel® VT

- For Intel® 64, VT-x: CPU MEM
- For directed I/O, VT-d: DMA/Interrupt remapping
- For connectivity, VT-c: offload wor from CPU to IO device

VT-x – cpu



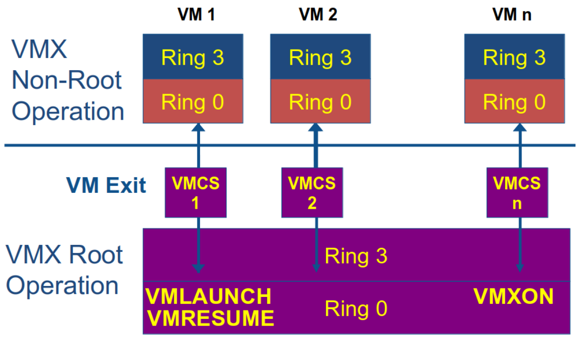
VM entry

- Transition from VMM to guest
- Enters VMX non-root operation
- VMLAUNCH used for initial entry
- VMRESUME used subsequently

VM exit

- Guest-to-VMM transition
- Enters VMX root operation
- Caused by external events, exceptions, some instructions

VT-x – cpu



Virtual Machine Control Structure (VMCS)

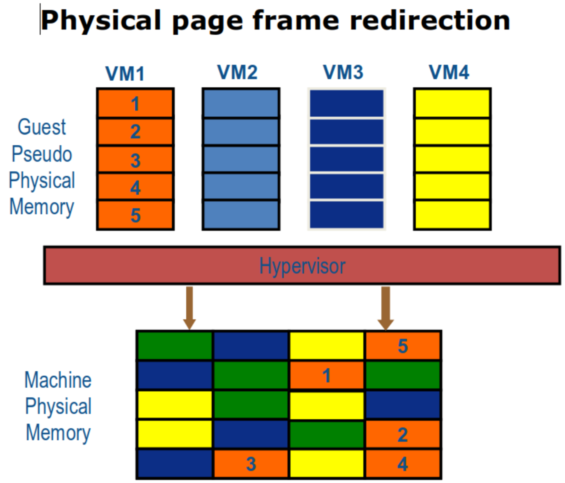
Controls VMX non-root operation/transitions

- Each virtual CPU should have its own VMCS
- Only one VMCS active at a time on a physical CPU

Each VMCS stored in a memory region

- Physical address set by new VMPTRLD instruction
- Need not reside in linear-address space
- VMPTRLD also used to switch VMCS

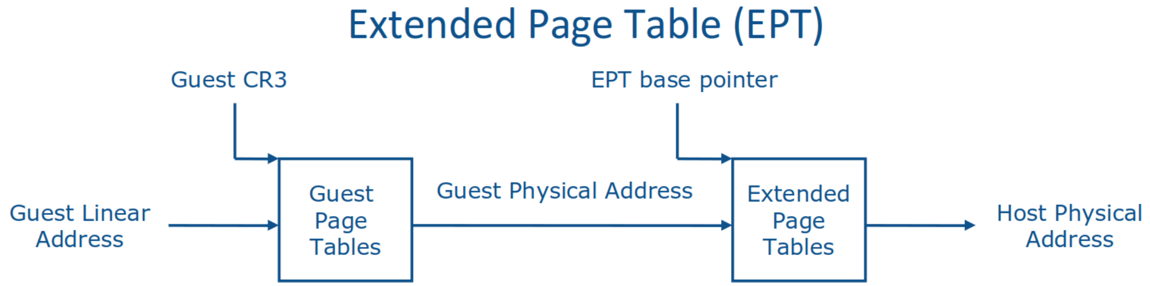
VT-x – mem



Memory virtualization challenges

- OS expect to see physical memory starting from 0
- OS expect to see contiguous memory in address space
- BIOS/Legacy OS are designed to boot from address low 1M
- DMA, TLB, etc.

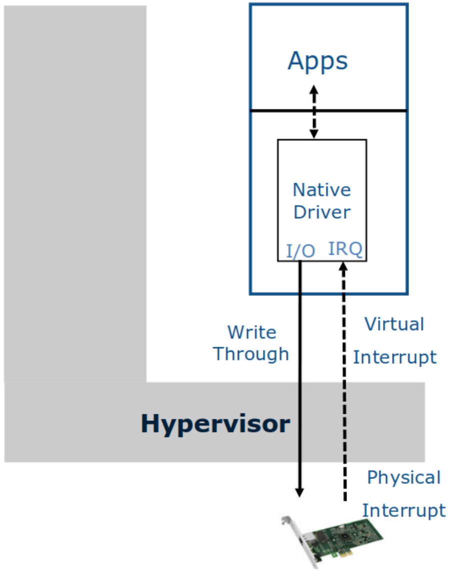
VT-x – mem



Memory virtualization challenges

- Guest can have full control over its page tables and events:
    - CR3, INVLPG, page fault
- VMM controls Extended Page Tables:
    - BIOS/Legacy OS are designed to boot from address low 1M
    - DMA, TLB, etc.

VT-d

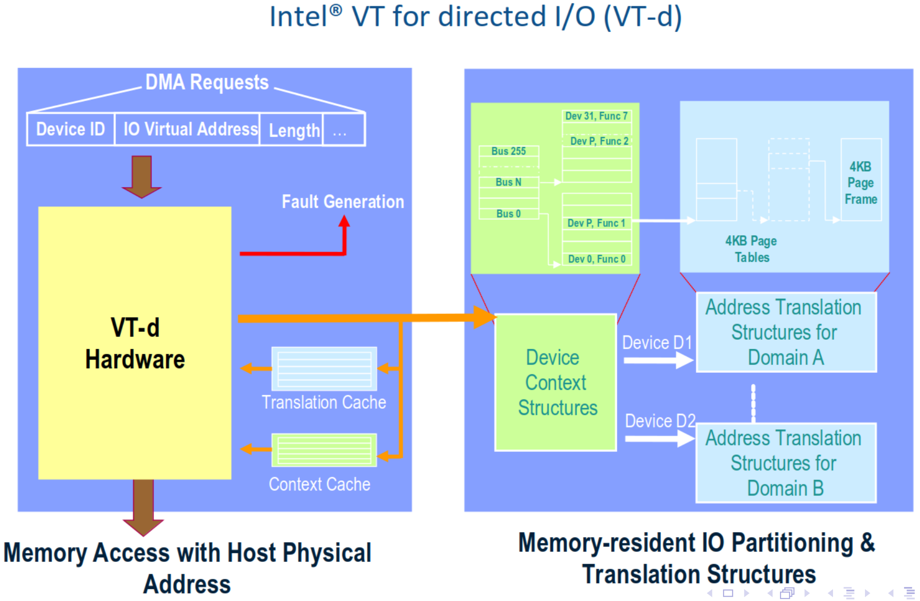


Direct assignment

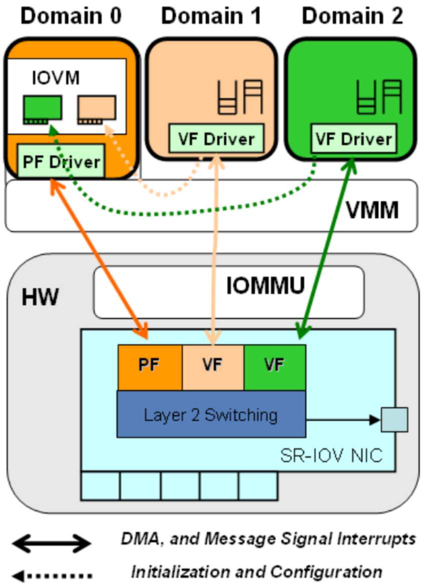
- Guest runs native driver
- I/O is written through
- Physical interrupt is captured by hypervisor (pIRQ)
- Virtual interrupt is signaled for guest (vIRQ)
- Remapping from pIRQ <-> vIRQ in hypervisor

Maximizing performance, but sacrificing sharing

VT-d


Intel® VT for directed I/O (VT-d)
DMA Requests
Device ID
IO Virtual Address
Length
...
Fault Generation
VT-d Hardware
Translation Cache
Context Cache
Memory Access with Host Physical Address
Dev 31, Func 7
Dev P, Func 2
Bus 255
Bus N
Bus 0
Dev P, Func 1
Dev 0, Func 0
4KB Page Frame
4KB Page Tables
Device Context Structures
Device D1
Device D2
Address Translation Structures for Domain A
Address Translation Structures for Domain B
Memory-resident IO Partitioning & Translation Structures

VT-c



Single Root I/O Virtualization (SR-IOV)

- Close to native performance
- Limited CPU overhead
- Flexible sharing
- Security control through PF

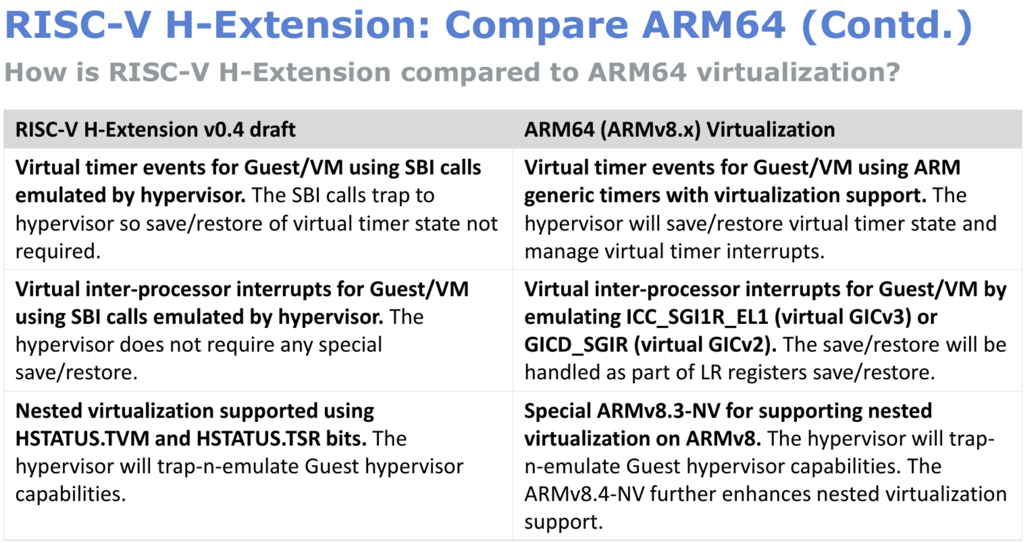
Maximizing performance

RISC-V H-Extension: Compare ARM64

How is RISC-V H-Extension compared to ARM64 virtualization?

RISC-V H-Extension v0.4 draft	ARM64 (ARMv8.x) Virtualization
No separate privilege mode for hypervisors. Extends S-mode with hypervisor capabilities (HS-mode) and Guest/VM run in virtualized S-mode/U-mode (VS-mode or VU-mode).	**Separate EL2 exception-level for hypervisors** with it's own <xyz>_EL2 MSRs. The Guest/VM will run in EL1/EL0 exception levels.
Well suited for both Type-1 (baremetal) and Type-2 (hosted) hypervisors. The S<xyz> CSRs access from VS-mode map to special VS<xyz> CSRs which are only accessible to HS-mode and M-mode.	**Special ARMv8.1-VHE Virtualization Host Extension for better performance of Type-2 (hosted) hypervisor.** Allows Host kernel (meant for EL1) to run in EL2 by mapping <xyz>_EL1 MSRs to <abc>_EL2 MSRs in Host mode.
Virtual interrupts for Guest/VM injected using VSIP CSR. The hypervisor does not require any special save/restore but it will emulate entire PLIC in software.	**Virtual interrupts for Guest/VM injected using LR registers of GICv2/GICv3 with virtualization extension.** The hypervisor will save/restore LR registers and emulate all GIC registers in software except GIC CPU registers.

Summary



RISC-V H-Extension: Compare ARM64 (Contd.)

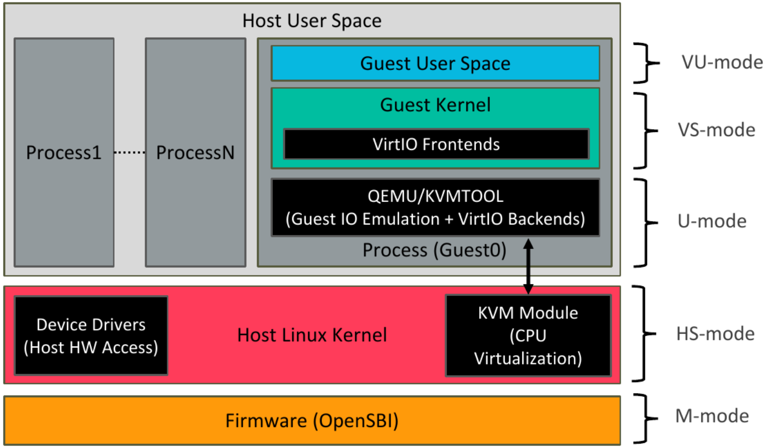
How is RISC-V H-Extension compared to ARM64 virtualization?

RISC-V H-Extension v0.4 draft	ARM64 (ARMv8.x) Virtualization
Virtual timer events for Guest/VM using SBI calls emulated by hypervisor. The SBI calls trap to hypervisor so save/restore of virtual timer state not required.	**Virtual timer events for Guest/VM using ARM generic timers with virtualization support.** The hypervisor will save/restore virtual timer state and manage virtual timer interrupts.
Virtual inter-processor interrupts for Guest/VM using SBI calls emulated by hypervisor. The hypervisor does not require any special save/restore.	**Virtual inter-processor interrupts for Guest/VM by emulating ICC_SGI1R_EL1 (virtual GICv3) or GICD_SGIR (virtual GICv2).** The save/restore will be handled as part of LR registers save/restore.
Nested virtualization supported using HSTATUS.TVM and HSTATUS.TSR bits. The hypervisor will trap-n-emulate Guest hypervisor capabilities.	**Special ARMv8.3-NV for supporting nested virtualization on ARMv8.** The hypervisor will trap-n-emulate Guest hypervisor capabilities. The ARMv8.4-NV further enhances nested virtualization support.

Summary

Linux KVM RISC-V

⬛	Hypervisor Component
🟧	M-mode Software
🟥	HS-mode Software
🟩	VS-mode Software
🟦	VU-mode Software
⬜	U-mode Software


Host User Space
Process1
ProcessN
Guest User Space
Guest Kernel
VirtIO Frontends
QEMU/KVMTOOL (Guest IO Emulation + VirtIO Backends)
Process (Guest0)
Device Drivers (Host HW Access)
Host Linux Kernel
KVM Module (CPU Virtualization)
Firmware (OpenSBI)
VU-mode
VS-mode
U-mode
HS-mode
M-mode