CS695 Lecture 4

cpu virtualization for VMs.

designs : trap-and-emulate — ✗

: binary translation / scan-and-patch ✓

avoid critical instructions via explicit scan & replace.

full virtualization ~ did not need any modifications to the OS.

int → save state on stack, s/w modes, jmps to the interrupt handler (?) VMM stack → copied onto guest Kstack.

Design2: PV approach (para-virtualized soln). Xen

- os is aware of being in a VM.
- avoids making any critical updates
- all system config access/update is explicit and via an interface.
- hypercalls ~ interfaces & impl^n's of services of the hypervisor.

popf :→ hc_cpuconfig(⟶);

hc_update_pgtable (va, pa, pgd);

hc_lcr3( );

Design 3: Hardware-assisted CPU virtualization

- Intel VT-x, AMD-V
- vmx mode of CPU operation
  - non-root — Ring 0 — Ring 3
  - root — Ring 0 — Ring 3

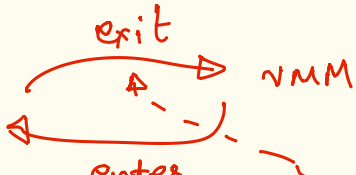
} introduce two levels of privileged modes.

(high PL)

vmx non-root } vmx modes
vmx root

Ring 3 Apps

Ring 0 OS



path was configurable

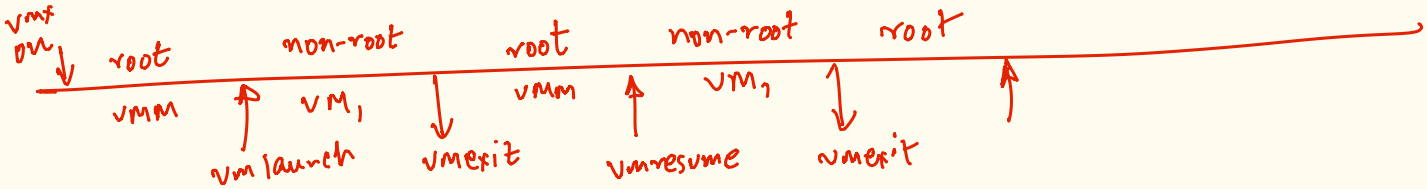
VMCS — virtual machine configuration set

- guest area } save & restore context
- host area }
- VM execution control fields ~ when to exit?
- VM entry control fields ~ config. on VM schedule
- VM exit control fields — what exit info. should be stored?
- VM exit information } actual exit info.

⊗ per VM VMCS

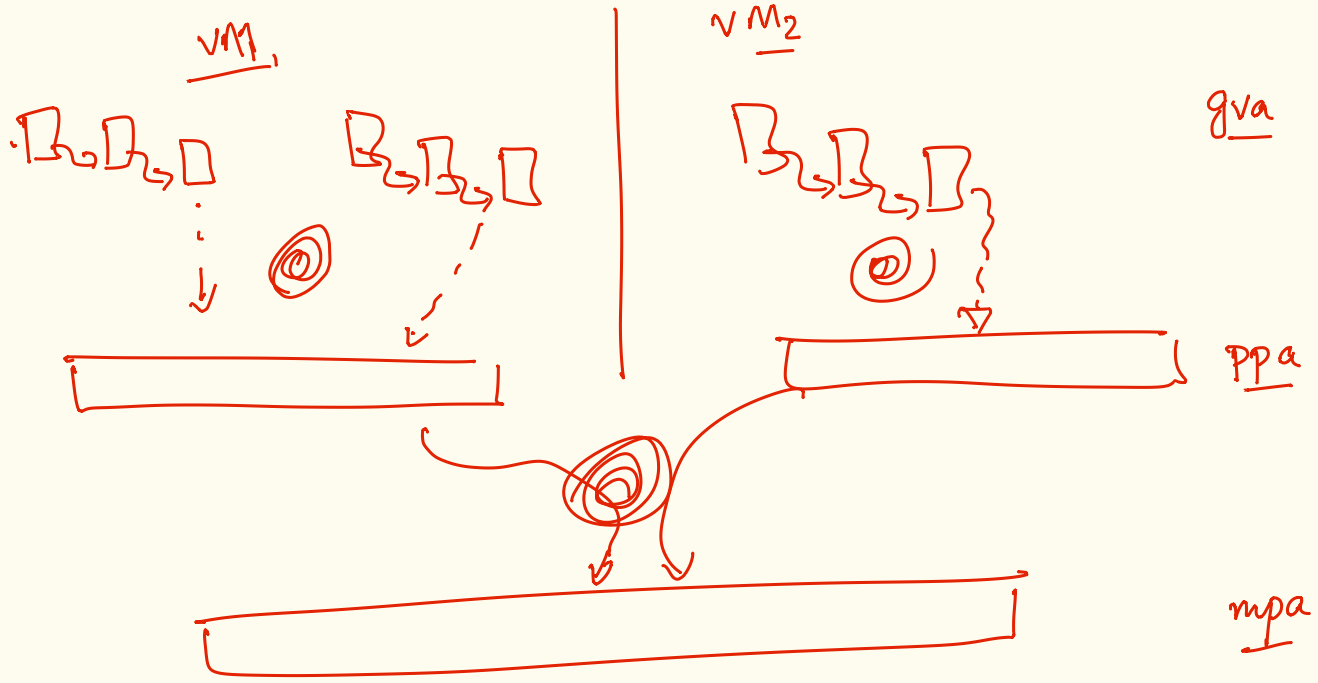
⊗ VMCALL — similar to hypercall.

the CPU uses an active VMCS per ~~int~~ scheduled VM.



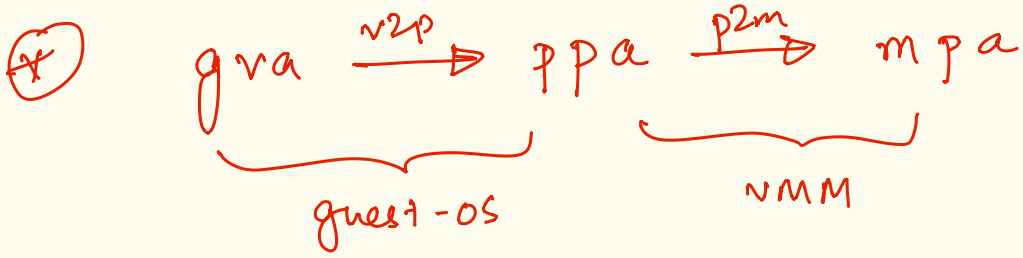
memory virtualization for VMs

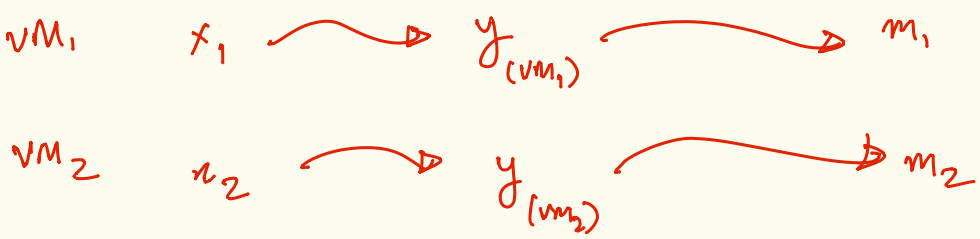
process-view	OS-view	hypervisor view
~ address space	- 0 starting	- 0 - starting
~ 0 - starting	- physical memory	- linear
- linear	- linear	- NUMA
	- NUMA	- physical memory



gva: uva ~ user virtual addresses
kva ~ kernel virtual address

x86. CR0 — bit 31 — paging
bit 0 — protected
} when enabled all CPU addresses are interpreted as virtual.





Q challenge!

↳ single MMM ⇒ single translation

but we need two translations.