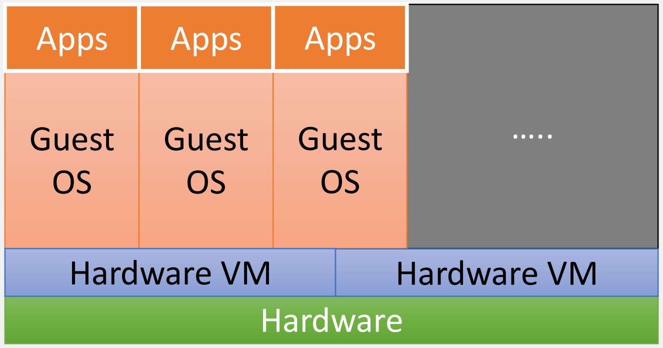
CLOUD COMPUTING APPLICATIONS

Virtualization: 1st Gen Hardware Virtualization

Prof. Reza Farivar

Hardware-Enabled Virtualization

- Intel VT (IVT)
- AMD virtualization (AMD-V)
- Allow “trapping” of sensitive instructions
  - Popek & Goldbreg → Trap and Emulate
- Examples:
  - VMWare Fusion, ESX
  - Parallels Desktop for Mac
  - Parallels Workstation



First Generation Hardware Virtualization

- First introduced in x86 in mid 2000s
- Intel VT-x, AMD-V
  - Virtual machine control block (`VMCB`).
    - in-memory data structure
  - The VMCB combines control state with a subset of the guest VCPU state
- A new, less privileged execution mode, guest mode, supports direct execution of guest code, including privileged kernel code

First Generation Hardware Virtualization

- A new instruction, `vmrun`, transfers from host to guest mode.
  - Upon execution of `vmrun`, the hardware loads guest state from the `VMCB` and continues execution in guest mode
  - Guest execution proceeds until some condition (set by VMM) is reached
    - The hardware performs an `exit` operation
    - `exit` is the inverse of `vmrun`
  - Guest state is saved to the `VMCB`, VMM state is loaded, and execution resumes in host mode, now in the VMM.

First Generation Hardware Virtualization

- First generation hardware support lacks explicit support for memory virtualization
  - The VMM must implement a software MMU using shadow page tables
  - → context switch on each `vmrun` and `exit`
    - VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, VMXON, INVEPT, INVVPID, and VMFUNC
- With hardware-assist, the guest runs at full speed, unless an exit is triggered
  - Virtualization overheads are determined as the product of the exit frequency and the average cost of handling an exit

MMU in First Generation Hardware Virtualization

- First gen hardware virtualization does not virtualize MMU
- The VMM has to get involved on MMU
  - VMM write-protects primary page tables to trigger `exits` when the guest updates primary page tables so that the VMM can propagate the change into the shadow page tables (e.g., invalidate).
  - the VMM must request `exits` on page faults to distinguish between hidden faults, which the VMM consumes to populate shadow page tables, and true faults, which the guest consumes to populate primary page tables.
  - the VMM must request `exits` on guest context switches so that it can activate the shadow page tables corresponding to the new context.
- First generation hardware support often did not outperform a BT- based VMM, often slower