National Yang Ming Chiao Tung University
Computer Architecture & System Lab

Virtualization

IOC5226 Operating System Capstone

Tsung Tai Yeh
Department of Computer Science
National Yang Ming Chiao Tung University

Acknowledgements and Disclaimer

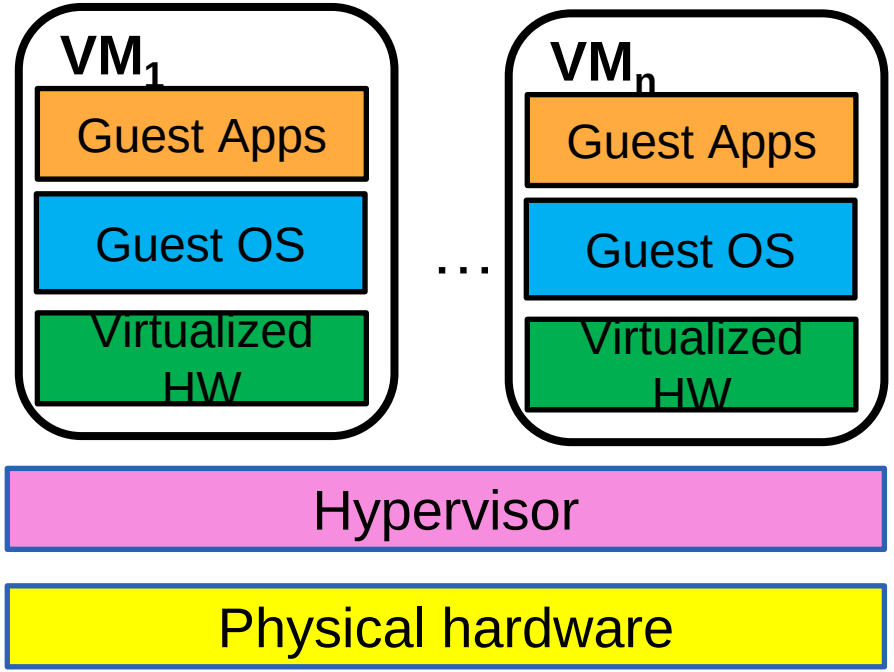
- Slides were developed in the reference with
  - MIT 6.828 Operating system engineering class, 2018
  - MIT 6.004 Operating system, 2018
  - Remzi H. Arpaci-Dusseau etl. , Operating systems: Three easy pieces. WISC

Outline

- Virtual machine
  - Full virtualization
  - Para-virtualization
- Hypervisors
- Xen
- Hardware-assisted virtualization

Virtual machines & Hypervisors

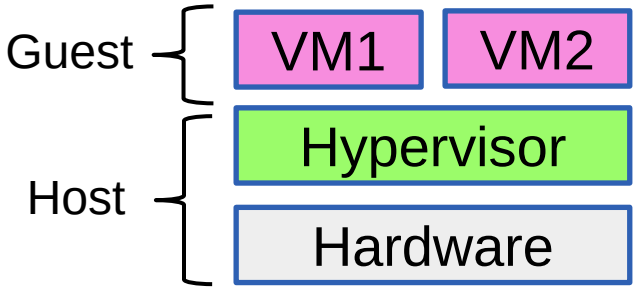
- **Virtual machine**
  - An abstraction of complete compute environment through virtualized hardware and I/O components of a computer
- **Hypervisor**
  - A system software (virtual machine manger(**VMM**)) manages and runs virtual machine



Hypervisors

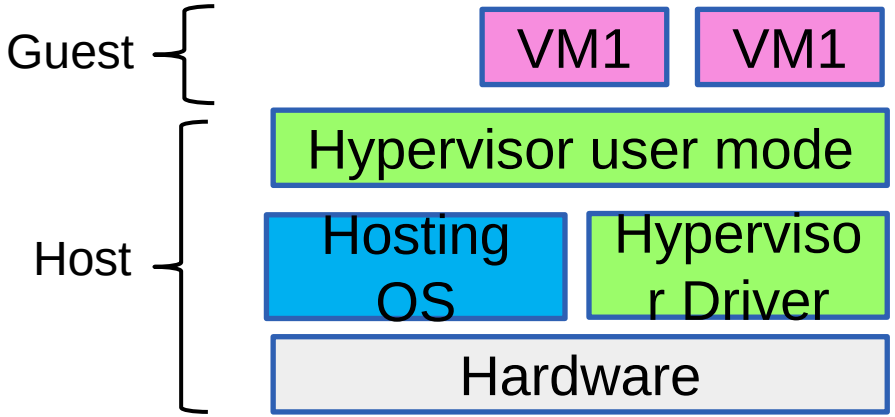
- **Type-1**: hypervisor controls the bare metal (e.g. servers)
- **Type-2**: hypervisor is hosted inside a host OS (e.g. desktops)

Type-1 (bare-metal)



Vmware ESX, Microsoft Hyper-V, Xen, Qemu

Type-2 (hosted)



Vmware Workstation, Microsoft Virtual PC, Sun virtualBox, KVM

Motivation of VM

- **Improve resource utilization**
  - Consolidate multiple virtual servers on a single physical server to improve utilization
- **Access to native applications on different platform**
- **Software testing and debugging**
  - Test codes on different platforms, even one that are not physically available
- **Better fault isolation**

Full virtualization

- **Hardware exposed to guest mimics a real hardware configuration**
  - No change required to the guest OS
- **Does not have to be exactly the same as underlying machine**
  - Can have smaller memory, fewer processor cores, different I/O devices, etc.
  - But should look and feel like some real machine

Problems of full virtualization

- **The guest kernel may not run in privileged mode**
  - To protect hypervisor
  - Let run the entire guest (both kernel and user) in unprivileged mode
  - However, the kernel will have to perform privileged operations
  - **Trap-and-emulate**
    - CPU will get the trap when the guest OS tries to execute a privileged operation
    - Hypervisor takes over, decodes the operation and emulates its effect

Problems of full virtualization

- Two problems with trap-and-emulate
  - 1) Too many traps will cause severe performance degradation
  - 2) Not all sensitive instructions will generate traps in all architectures
    - E.g. SIDT and POPF in x86
  - A solution to (2): **Binary translation**
    - The hypervisor will analyze all of guest code, and replace non-trapping sensitive instructions with explicit traps -> enable efficiently to virtualize x86

Para-virtualization

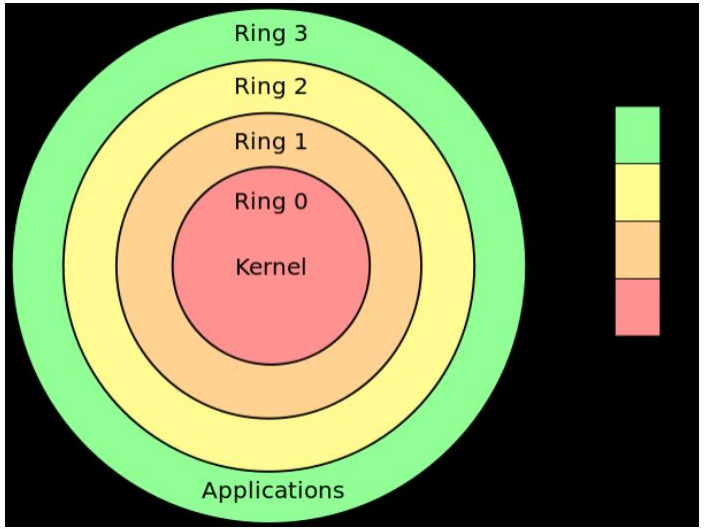
- Modify guest OS
- It knows about hypervisor
- Applications not modified
- Improved performance
  - Reduce redirections, allowing guest OS to use real hardware resource in a secure manner
- Allows to do virtualization without hardware support

Para-virtualization

- **Virtualization-friendly abstraction to the guest OS**
  - Will require changes to the guest OS
  - Xen reported 1.5% code change for Linux and 0.04% for Windows XP
  - Guest OS knows that it is being virtualized and run with reduced privilege
  - Avoid most trap-and-emulate situations by using explicit **hypercalls** to the hypervisor

Discussion - Xen

- A Type 1 (bare-metal) hypervisor
  - Runs directly on the host hardware
- Providing a native-like perf.
- Protection
  - In x86, 4 levels of privilege
  - Level 3 for applications
  - Level 0 (highest) is for OS
  - Level 1 or 2 is for guest OS
  - Xen is at level 0



wikipedia

Discussion - Xen

- When exception (page fault) occurs
  - No back and forth between Xen and Guest OS shown in full virtualization
  - Fast handlers for system call
    - Directly goes to guest OS handler in ring 1, not to Xen
  - The page fault handler also goes through Xen
  - Handlers validated before installing in hardware exception table

Discussion - Xen

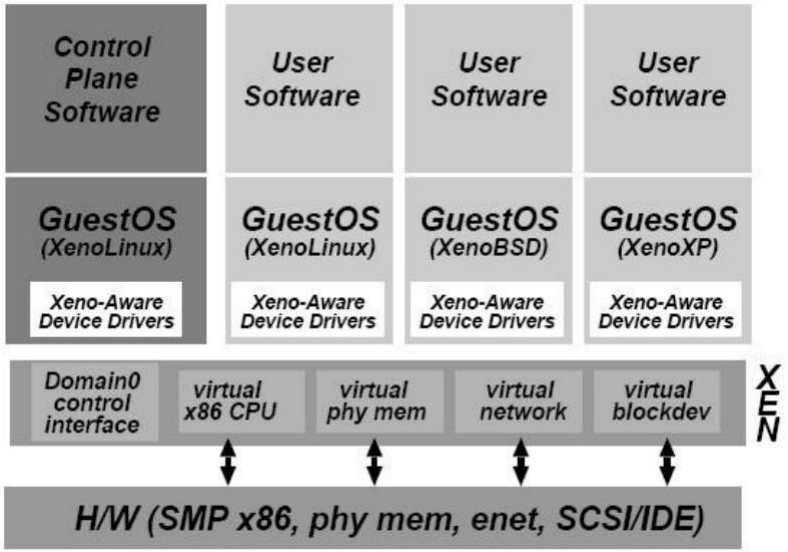
- **Memory management**
  - Want to avoid TLB flushing between switches (Guest OS - Xen)
  - In x86 architecture
    - Xen sits in the top of 64 MB of VM address space
    - Guest OS shouldn't access top 64 MB
    - Xen never paged out
    - When guest OS needs new page table, allocate from its own memory reservation
    - Xen gives up write-privileges, the guest OS can read directly
    - No shadow table here

Discussion - Xen

- **Hypercalls and Events**
  - The guest OS validates with Xen for every update
  - Batch updates together to minimizes the number of “Hypercalls”
  - **Hypercalls**: synchronous calls to Xen
  - In xen/include/public/xen.h
    - ~40 hypercalls
    - E.g. Set trap table, mmu update, etc.
  - **Events**: asynchronous notifications from Xen that is like device interrupts

Discussion - Xen

- **Guest OS**
  - OS hosted by Xen
  - Program, domain process
- **Domain**
  - Separate from Guest OS
  - Control Guest OS
  - More access to hardware hypervisor
  - Reduce hypervisor complexity
  - Memory reservation for new domains done statically



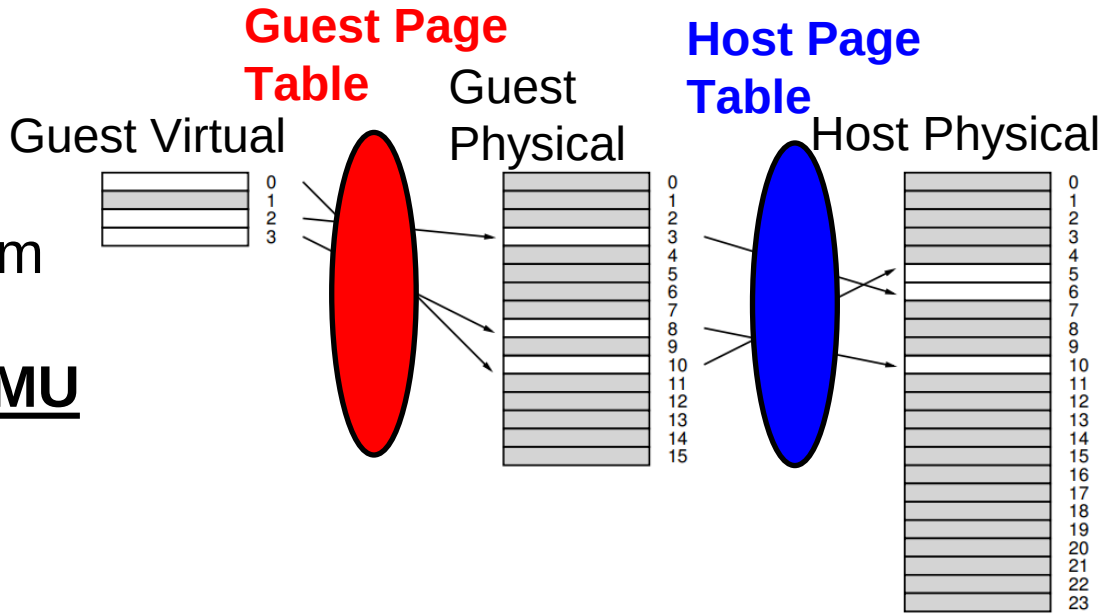
https://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf

What to virtualize ?

- Three things
  - 1) CPU (including the privileged state)
  - 2) Memory
  - 3) I/O devices
- 1) and 3) are simpler
  - Just trap on relevant accesses, or binary translate them, or have the guest OS make hypercalls
- 2) is more complicated
  - Requires **shadow page table**

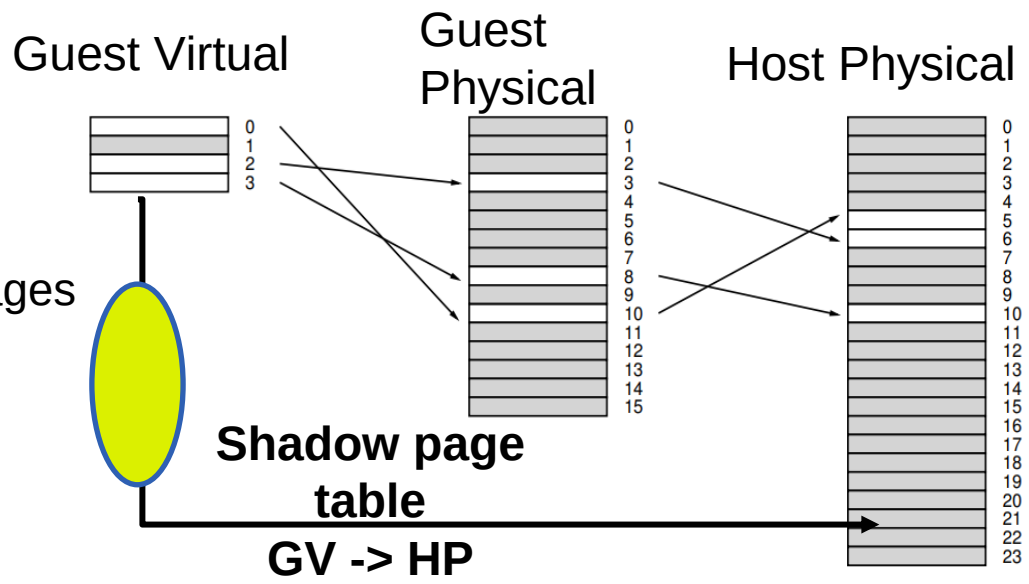
MMU virtualization problem

- The VMM handles three types of memory spaces in a virtualized environment
  - **Guest Virtual (GV)**
  - **Guest Physical (GP)**
  - **Host Physical (HP)**
- Need one page table from GV to GP and GP to HP
- **<u>But older processor MMU can only deal with one page table</u>**



Shadow page table

- **Shadow page table**
  - Combine the two translations (GV->GP->HP) into a single translation
  - **The hypervisor needs to know of any change to the guest page table**
    - GV -> GP translation
  - **Full virtualization**
    - Mark guest page table pages
    - Trap on every page table change
  - **Para-virtualization**: change the shadow page table by hypercall



Hardware-assisted virtualization

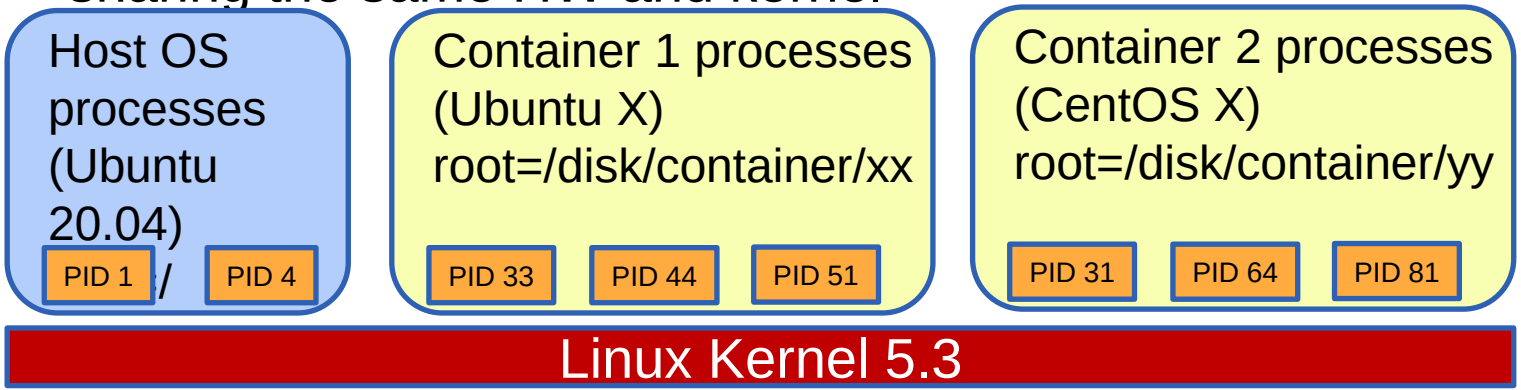
- **Motivation**
  - Trap-and-emulate is expensive
  - Too many changes to guest (para-virt) not always desirable
- **Change hardware to make it virtualization friendly**
  - E.g. Intel VT-x and AMD-V technologies
- **Hardware support**
  - Remove most trap-and-emulate situations

Hardware-assisted virtualization

- **Hardware-assisted virtualization**
  - **CPU:** duplicate the entire architecturally visible state of the processor in separate root (for hypervisor) and non-root (for guest) modes
    - Make most traps into hypervisor unnecessary
  - **MMU:** HW supports second layer of page table (managed by hypervisor)
    - Make shadow page tables unnecessary
  - **I/O:** add support for high-performance I/O through IOMMU and SR-IOV
    - Enable direct hardware access by guests

Lightweight Virtualization -- Container

- All processes on the machine share the same kernel
- Each container will use a different user-mode image of a compatible OS
  - Not real virtualization; just different user-mode OS images sharing the same HW and kernel



Takeaway Questions

- What is the primary purpose of a hypervisor in virtualization?
  - (A) To manage the guest operating systems
  - (B) To allocate hardware resources and manage virtual machines
  - (C) To create virtual hard disks
- In virtualization, what is the guest operating system?
  - (A) The operating system running on a remote server
  - (B) The main operating system running on the physical machine
  - (C) The operating system installed on a virtual machine

Takeaway Questions

- Which type of virtualization allows multiple virtual machines to share the same operating system kernel?
  - (A) Containerization
  - (B) Full virtualization
  - (C) Paravirtualization