Virtual Machines

key concepts: virtualization, hypervisors

Lesley Istead

David R. Cheriton School of Computer Science
University of Waterloo

Spring 2021

Virtual Machines

- a **virtual machine** is a simulated or emulated computer system
  - Sys/161 is an emulation of a MIPS R3000
  - sys161.conf is used to configure the virtual hardware
- virtual machines provide the ability for one machine to act as many
- operating systems and programs can run on virtual machines in isolation
  - the OS and programs should not be aware of the virtualized hardware and operate normally —without modification, patching, etc.

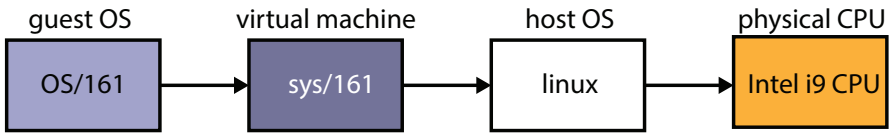
Virtual machines date back to the 1960s, but performance was typically very poor. They made a resurgence in 2005, when CPUs offered hardware support.

Role Review

- a system is comprised of a CPU, RAM, and other devices
- the CPU executes the instructions of both the operating system and user programs
- the operating system kernel:
  - creates execution environments for user programs
  - handles interrupts and exceptions raised by the CPU
    - interrupts are raised by devices requiring attention
    - i.e., a key was pushed, a packet has arrived, a timer has expired, ...
    - exceptions are raised by software
    - i.t., division by zero, illegal instructions, system calls, ...
  - manages memory
  - implements concurrency

A First Attempt

- the virtual CPU executes the instructions of its operating system (the **guest**) and user programs running within that OS; **but this CPU does not exist**
  - "capture" the instructions meant for this virtual CPU
  - **translate** these instructions into instructions for the real CPU, where they may be executed as a normal program
    - interrupts?
    - privilege?
    - virtual memory?



Interrupts, Privilege, Virtual Memory

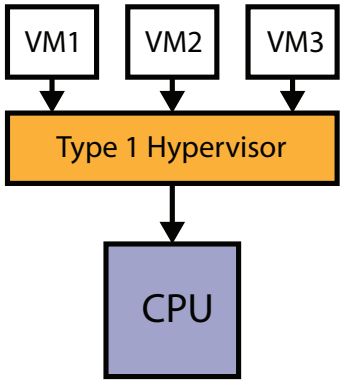
- An OS running on a virtual machine wants to disable interrupts on the CPU to perform some task without interruption, e.g., `spinlock_acquire`.
  - The VM translates and passes instructions through to the physical CPU for execution.
  - Can the VM disable interrupts on the physical CPU? Should it be able to?
- The letter "k" was pressed causing the keyboard to fire an interrupt on the physical CPU.
  - Whom should handle this interrupt? Which VM is it for?
- The OS running on the VM uses paging for its virtual memory implementation. A page table maps virtual page k to physical page 0x100. The host OS also maps a page to physical page 0x100.
  - collision!?

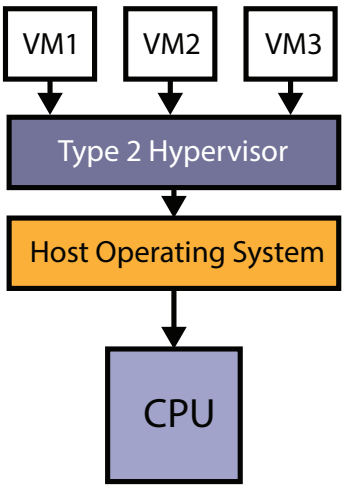
Hypervisor

- A **hypervisor** is a virtual machine manager
  - it creates and manages virtual machines
- There are two types:

Type 1: hypervisor runs on bare hardware
Type 2: hypervisor runs on the **host** operating system

- The operating system running on a VM is called the **guest**





Hypervisors

- Type 1 hypervisors run in privileged mode
  - VMs run less privileged modes; they cannot disable interrupts, etc.
  - unprivileged instructions may execute normally
  - privileged instructions trap to the hypervisor, where their behaviour is handled
    - the hypervisor may pass these instructions through for execution and emulate the behaviour of others
    - either way, to the VM, interrupts are turned off
- Type 2 hypervisors run in unprivileged mode
  - unprivileged instructions can be translated and executed as a normal user process
  - the hypervisor will emulate behaviour of privileged instructions and make system calls to the host OS as appropriate

Privilege

- Operating systems and user programs may make privileged calls.
  - user programs do so in error or maliciously
- How does the hypervisor differentiate between the OS and user program executing a privileged instruction?
  - "rings" of privilege at the CPU; not just privileged vs. unprivileged
  - the hypervisor runs in the highest privileged mode
  - the guest operating system runs in the next highest mode
  - guest user programs run in the lowest privilege mode
  - when execution of a privileged instruction causes a trap to the hypervisor; it can determine the CPU privilege to determine the source

> How might a Type 2 hypervisor differentiate between the operating system and the user process making privileged calls?

Virtual Memory

- Modern operating systems use multi-level paging
- multiple guest operating systems may be running
  - the guests are unaware of each other
  - each believe they are the "king of the castle"
- each guest operating system may map virtual pages to the same real physical page
  - but only **one** could technically use that page
- Type 1 hypervisors manage memory so that collisions on physical pages do not occur via **shadow page tables**
  - a **shadow page table** treats the guest physical pages as another level in the table, and will map it to a physical page that will not collide
  - but then **every** translation must go through the hypervisor, instead of just the MMU

A virtual addresses on a guest VM is called the **guest virtual address**, the physical address on the VM is called the **guest physical address**, and the the real physical address is called the **host or machine physical address**.

Extended Page Tables

- having every virtual address go through the hypervisor for translation is costly
- faster to perform translation **in hardware**
- modern CPUs, Intel Nehalem and on-wards, support **Extended Page Tables** in the MMU
  - the MMU can translate guest virtual addresses to host physical addresses directly
  - the entries in the guest physical tables are pointers to the shadow page tables containing host physical pages
  - the hypervisor should update the MMU's page table base register on a world swap (when the VM executing changes)

What if the sum of memory allocated to each guest is greater than the amount of physical memory? How can on-demand paging be used to solve this overcommitment issue?

I/O and Devices

- the guest operating system requires a disk to save data/programs
  - each guest could have it's own disk partition; but what if more VMs than feasible partitions?
  - the hypervisor creates a file on the disk and presents it to the VM as a file system
  - the hypervisor can present this file as any kind of disk—even a magnetic tape
- other devices required for input/output
  - **device pass through** lets devices be assigned to a specific VM
  - **device isolation** ensures that if the keyboard is assigned to a particular VM, that device will not affect other guests
    - if VM1 is running and VM2 is sleeping, pushing a key on the keyboard should not wake VM2
  - **interrupt redirection** lets interrupts from a particular device be directed to the specific VM they are assigned to
  - hardware support is required

Why Virtual Machines?

- provide safe, inexpensive sandbox environment for users and programmers
  - run sensitive or suspect applications without affecting the integrity of the real system
  - develop and test programs for a different architectures or operating systems
- resource utilization
  - permit multiple users to use a server or **cloud** in isolation
    - Amazon AWS
    - Microsoft Azure
    - Google Cloud
- checkpoints/snapshots: let users pause a virtual machine and continue later from the exact saved position

The End.

We hope that you have enjoyed this introduction to operating systems.