Agenda

Virtualization and the Cloud

- What is virtualization?
- Examples/Demos
- Requirements for virtualization
- Advantages
- Type 1 and type 2 virtualization
- Approaches to emulating hardware: Page-tables, I/O
- Clouds

Virtualization

Idea: A **virtual machine (VM)** acts like a real computer system

- But runs on top of a “real” operating system
- A VM is a simulated computer

Software running on the virtual machine is separated from the underlying hardware

Hardware is emulated

Requires a Virtual Machine Monitor (Hypervisor)

Examples

Oracle VirtualBox, VMWare

Docker

Windows subsystem for linux (WSL)

Cloud Hosting: Digital Ocean/Dreamhost

Emulators: termux, Project64, Nand2Tetris

Example: VirtualBox

Idea: Runs any OS on top of an existing operating system

- The OS is installed from an ISO disk image, e.g. what is typically stored on a CD or a USB.

- From the perspective of the installed OS, it “thinks” it is the real OS
  - VirtualBox is the **host** OS. And the installed OS is the **guest**.

- Applications run on the guest OS in a sandboxed environment

Try it yourself: https://www.virtualbox.org/

NOTE: ISO stands International Standard Organization

Example: Docker

Idea: Docker is container platform that combines software with the dependencies and OS needed to run that software

- Makes it easier to share applications and ensure they still work, even when dependencies change

- Provides a sandboxed environment where the application runs as a guest on the host machine

- Sometimes also called a **virtual appliance**

Try it yourself:

- https://www.docker.com
- https://chtc.cs.wisc.edu/uw-research-computing/docker-build

Example: Windows Subsystem for Linux (WSL2)

Idea: Runs a parallel Linux OS with Windows

- WSL2 has direct access to the hardware, like the Windows OS

- WSL2 has its own file system. Each OS mounts the file system of the other so files can be moved/accessed between them

Cloud hosting: Digital Ocean, Dreamhost, AWS

Idea: Illusion of many machines on (potentially) the same

Each VM is independent. Failure of one VM doesn't bring down the others

Users can have 1+ VMs

Each VM has limited CPU/Memory resources (everyone shares)

Users let the cloud service providers handle backups and upgrades

The VMs run independent from the hardware, meaning that cloud service providers can do **live seamless migration** to new hardware

A **data center** is a large warehouses that stores many servers

Emulators: Project 64, termux, Nand2Tetris

Idea: The OS/Computer is simulated

Hardware is not accessed directly

System calls, or ISA instructions, are implemented on the host system

- Ex. Nand2Tetris simulated instructions on simulated hardware
- Ex. Project64 executes the Nintendo64 instruction set
- Ex. termux implements a lightweight Linux OS with file system on Android

Try it out:

https://github.com/termux/termux-app
https://github.com/project64/project64

Requirements for Virtualization

1. **Safety**: hypervisor should have full control of virtualized resources.
2. **Fidelity**: behavior of a program on a virtual machine should be identical to same program running on bare hardware.
3. **Efficiency**: much of code in virtual machine should run without intervention by hypervisor.

A **sensitive instruction** is an instruction that behaves differently depending on whether the process is in user mode or kernel mode

Exs. I/O, changing MMU settings, etc

A **privileged instruction** causes a trap when executed in user mode

A machine is virtualizable if the sensitive instructions are a subset of the privileged instructions(Popek and Goldberg, 1974). Why?

Advantages

A VM is **sandboxed,** meaning that errors in one machine does not affect the others

A VM container can be shared with others. Legacy applications can be run in their preferred enironment

Aids with porting software. We can test on multiple platforms without buying additional hardware

Prototype new hardware and OS features on existing hardware/OS

Multiple clients can share the same hardware.

Virtualization Types

Full virtualization

- Almost complete emulation of hardware to allow software (such as a guest OS) to run unmodified
- Exs. VirtualBox

Paravirtualization

- Guest software “knows” they are running on a VM
- **Hypercalls** make requests on the hypervisor (e.g. an API)

Process-level virtualization refers to situations where we don’t simulate an entire OS but just enough of one to let an application run

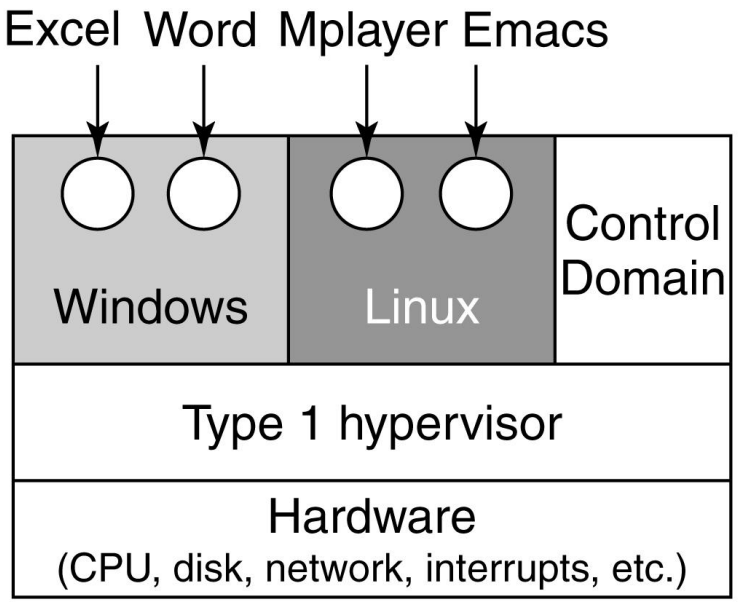
Type 1 and Type 2 Hypervisors

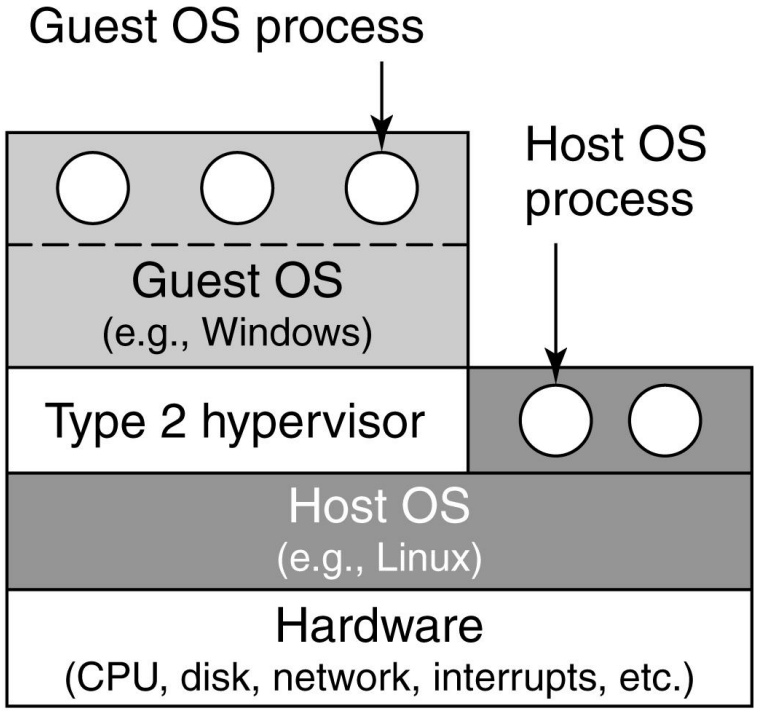
A **type 1 hypervisor** acts like an operating system that runs in a privileged mode with access to hardware

Supports multiple copies of the actual hardware, called a **virtual machine**

Exs. WSL2

A **type 2 hypervisor** runs like a regular process and accesses hardware through the host OS. Ex. VirtualBox, Emulators, Containers





Hypervisor Implementation Approaches

An **interpreter** executes each instruction in turn in terms of the host hardware

- Can be computationally expensive, so hypervisors usually try to execute instructions directly
- Safe

In a **trap-and-emulate** approach, the guest OS generates traps when it executes a privileged instruction. The hypervisor then takes over and executes the instruction (e.g. I/O)

With **binary translation**, instructions are either translated directly or converted to a series of equivalent instructions before they are executed.

Challenges: Emulating the host CPU, Keeping page tables consistent, Managing I/O

Memory Virtualization

- Each guest OS believes it is king and therefore is at liberty to map any virtual page to any physical page
- What does a hypervisor do if two VMs want to map the same physical memory page?
- Shadow page tables – a mapping that maps the virtual pages used by the VMs to the physical pages the hypervisor selected

Shadow Page Tables

A **shadow page table** maps the guest page table to real page table on the host OS

Must be updated every time a guest OS updates its page table

But how does the hypervisor know?

- keep track of the page tables and mark them read-only so that any attempts to write to them will cause a fault that traps to the hypervisor
- ignore it until the guest OS tries to actually access these new pages, which will cause a fault and trap

Lots of New Page Faults

Guest-induced page faults

- Normal guest OS page faults

Hypervisor-induced page faults

- the kind that are related to syncing the shadow page table and the guest OS's

Page faults are expensive

Nested/Extended Page Tables

Handle the additional pagetable manipulations in hardware without traps

Hypervisor still mains additional page tables

Hardware walks the all page tables to find these mappings

- guest virtual address – guest physical address
- guest physical address – host physical address

I/O Virtualization

Guest OSes must share a disk

Hypervisor creates a virtual disk (a file) for each VM

When a guest OS tries to access disk, block numbers are converted into an offset into the appropriate disk

Physical disk can differ from virtual disk (what the guest OS wants) via a VFS interface in the hypervisor

Clouds

National Institute of Standards and Technology defines characteristics of “cloud”

1. On-demand self-service: Users get access to resources without going through a person
2. Broad network access: Resources are available over standard network connections
3. Resource pooling: Resources are shared among multiple users
4. Rapid elasticity: Scale resource use with user demand
5. Measured service: Service use is metered and charged based on use

An **IAAS (Infrastructure as a Service)** cloud offers users direct access to a virtual machine. The same hardware might run different OSs for each user. Ex. Amazon EC2

An **PAAS (Platform as a Service)** cloud includes specific OS, databases, etc pre-installed. Ex. Dreamhost

Aside: A **SAAS (Software as a Service)** application provides software with a subscription. Ex. MS Office 360

NOTE: When it is difficult to leave a platform, it is called **vendor-lock-in**