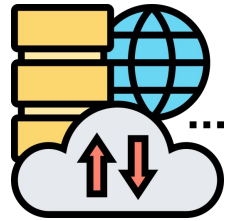
# COEN 241
# Introduction to Cloud Computing

## Lecture 2 - Virtualization



# Lecture 1 Recap

- Cloud Computing Primer

- Course Overview
  - Course Objectives
  - Course Structure
  - TODOs
  - Logistics
  - Instructor Information

- Readings
  - Recommended: CCSA 1.1 - 1.3
  - Optional: CCSA 1.5 - 1.15

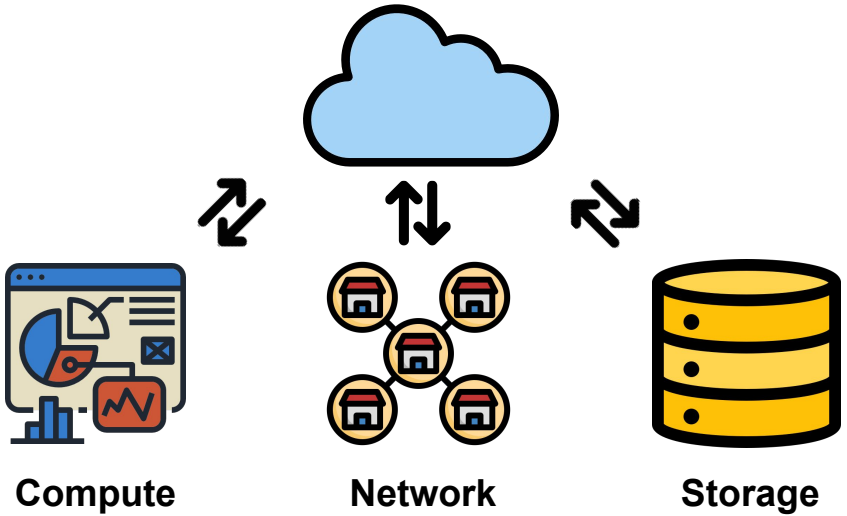
# What is Cloud Computing?

**Cloud computing** is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., server, network, storage & applications) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

* NIST(The National Institute of Standards and Technology)

# What is Cloud Computing?

On demand access to




over the internet

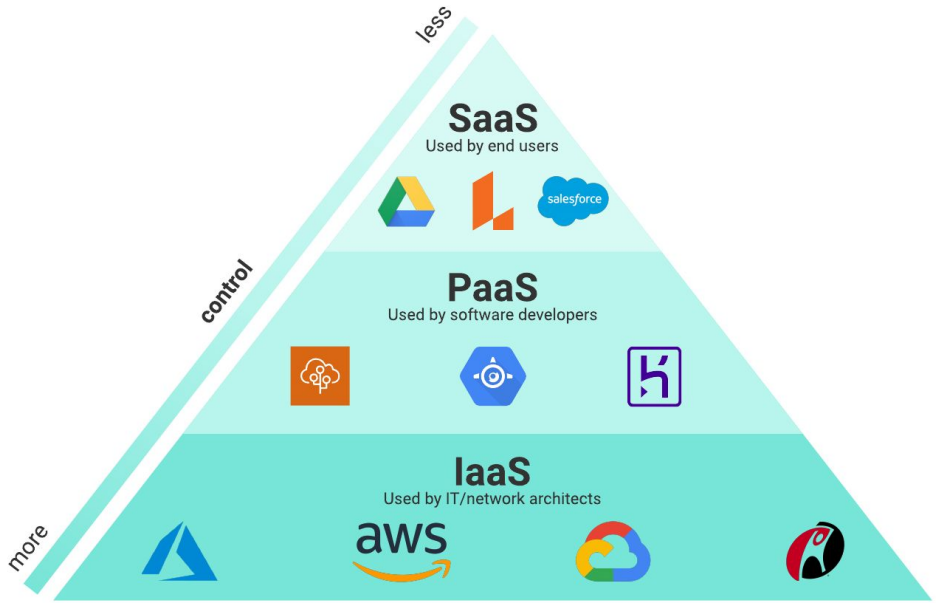
# Cloud Computing Enables:

- The illusion of “Ubiquitous” & infinite “Shared Pool” of computing resources available “On-Demand”.
  - Eliminates the need to plan far ahead for provisioning.
- Rapidly and automatically provisioned computing resources.
- Minimal hardware / infrastructure management with increased reliability.

# Cloud Computing Enables:

- Access to need-based performance.
- The elimination of an up-front commitment by Cloud users.
  - Allows companies to start small and expand only when needed.
- The ability to pay for use of computing resources on a short-term basis as needed (e.g., processors by the hour and storage by the day)
  - Rewards freeing resources when they are no longer useful.

# Cloud Service Models



# Cloud Deployment Models

# Agenda for Today

- HW 0 Preview & Project Overview
- Multitenancy
- Virtualizations
- History of Virtualization
- Readings
  - Recommended: CCSA 2.1-2.2
  - Optional:
    - http://www.firmcodes.com/memory-thrashing-in-operating-system/
    - https://www.ibm.com/cloud/learn/virtualization-a-complete-guide
    - http://www.leeandmelindavarian.com/Melinda/25paper.pdf

# HW 0 Preview

- Provided to give you some background context of Linux shell & git
- Keep the Shell tutorial for your reference
- Just need to submit short answers on Camino

# Project TODO Overview

- Form Teams by Friday! **Everyone** must submit.
- Project Proposal by next Wednesday!
  - Brief description of the goal, motivation, technologies you will use, etc.
  - More details soon
- Project Presentation (last two (or three) classes of the quarter)
  - 20 minutes of presentation + demo
  - You will get a signup sheet after forming the groups
- Final Project (due finals week)
  - Conference style paper 6~8 pages



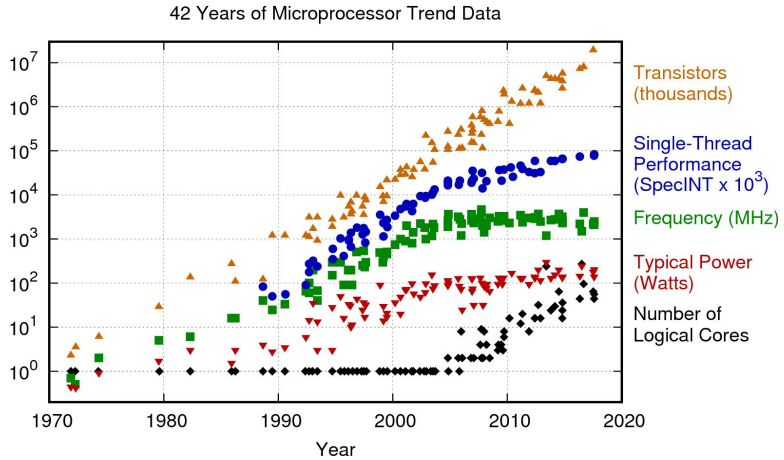
# Multitenancy

# Cloud Computing Enables:

- The illusion of “Ubiquitous” & infinite **“Shared Pool”** of computing resources available “On-Demand”.
  - Eliminates the need to plan far ahead for provisioning.
- Rapidly and automatically provisioned computing resources.
- Minimal hardware / infrastructure management with increased reliability.

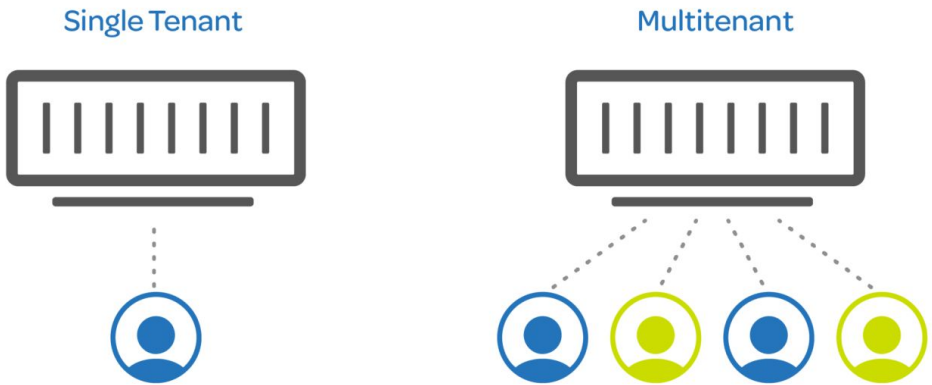
# The Need for a “Shared Pool”

- CPUs are getting more powerful (Higher speed, More cores)
- Often times, a single user does not fully utilize a single machine
  - Both in processing power and time
  - Average utilization is < 20%
- Often more cost efficient to “split” a large machine than having many small machines (due to fixed cost in wiring, networking and management)



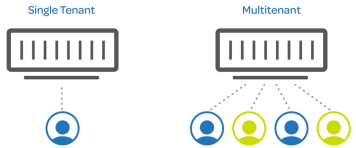
# Multitenancy

- Definition: An architecture where multiple users share a single (software or hardware) resource



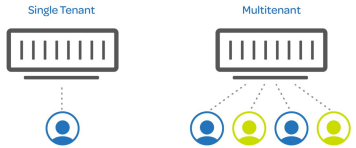
# Multitenancy

- Benefits
  - Cost efficient
  - Easier to monitor and manage
  - Easily scalable
- Drawbacks
  - Generally more complex architecture
  - Security and isolation
  - Allocation



# How to Enable Multitenancy?

- **Virtualization!**
  - A core part of this course
- Definition: Using **software** to create an abstraction layer over the **hardware** that allows the hardware elements of a single computer—processors, memory, storage and more—to be divided into **multiple virtual elements**.

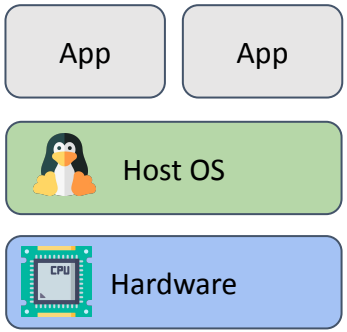





# Virtualization

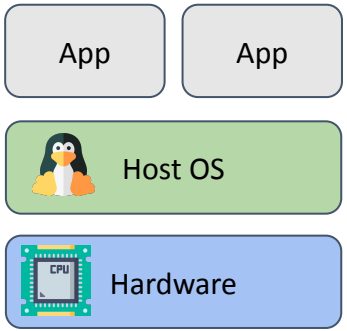
# Why can't we just have multiple processes?

- What are processes?
- How are they different from Threads?
- To make things simple, can't we just run a set of separate processes (or threads) for a given user?
  - We have multi-core systems now
  - Simple to manage
  - OS is now smart enough!
  - No need to learn virtualization! Bye bye
  - Why can't we do this?



# Why can't we just have multiple processes?

- Hard to control resource contention
  - Need fine grained resource management for each process
- Hard to manage dependencies
  - All the users must be compatible with the shared OS
  - Cannot run different versions of the kernel
- Hard to guarantee performance!
  - Memory Thrashing
  - Unfair I/O access

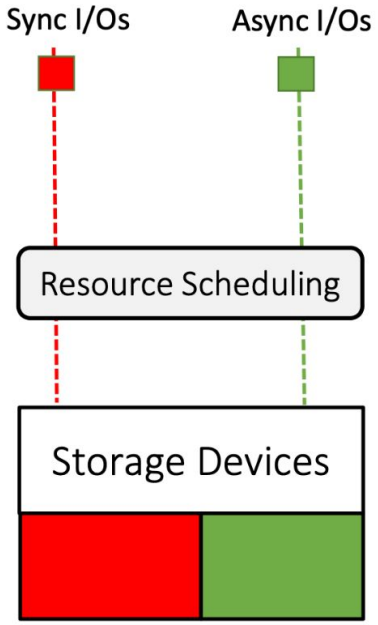


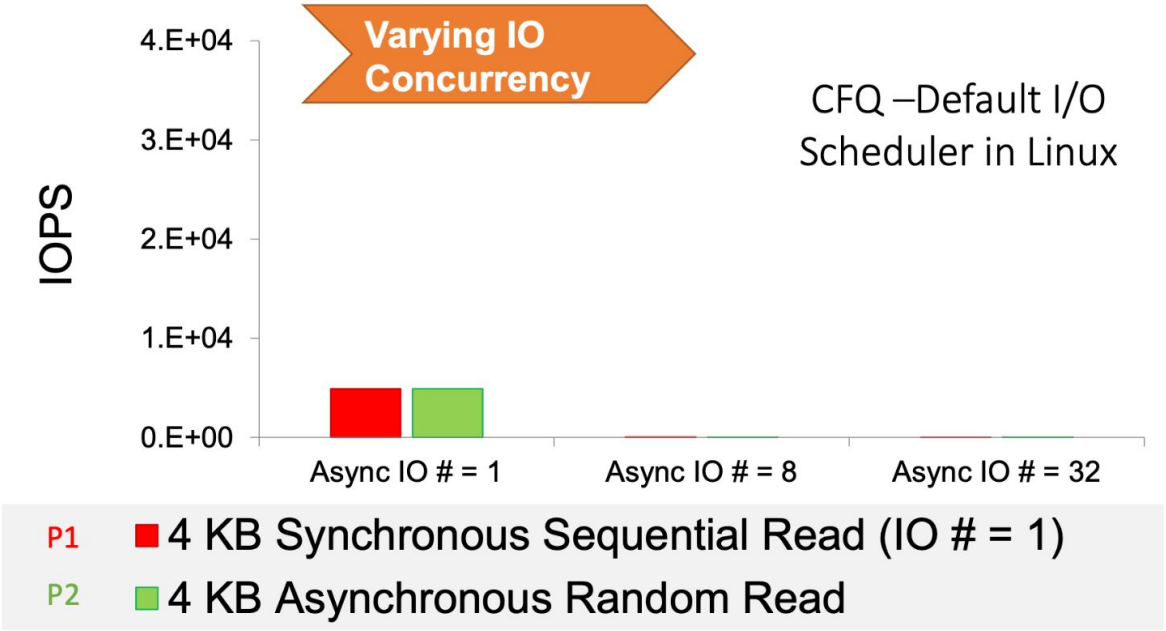
# Exercise: Memory Sharing

```
int main() {
        int i;
        char *p[2048];

        for(i=0;i<2048;i++) {
                p[i]=malloc(1024000); // allocate 1 MB
                memset(p[i],'\0',1024000); // set as 0s
        }

        while(1) {
                if (i>=2048)
                        i = 0;
                memset(p[i],'\0',1024000);
                i++;
        }
}
```

- What does this program do?
- What would happen if we run 10 processes with the same code given on the right?
- What happens if we have machines with following amounts of RAM?
  - 2GB
  - 10GB
  - 20GB

# Exercise: Unfairness of I/O Resource Sharing



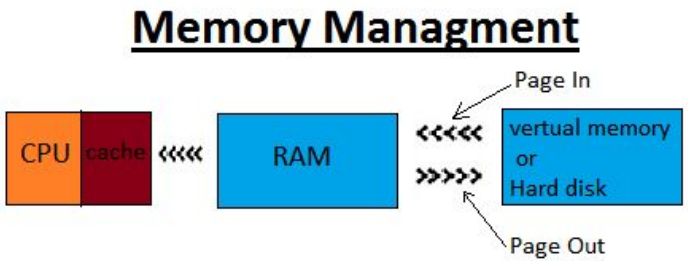



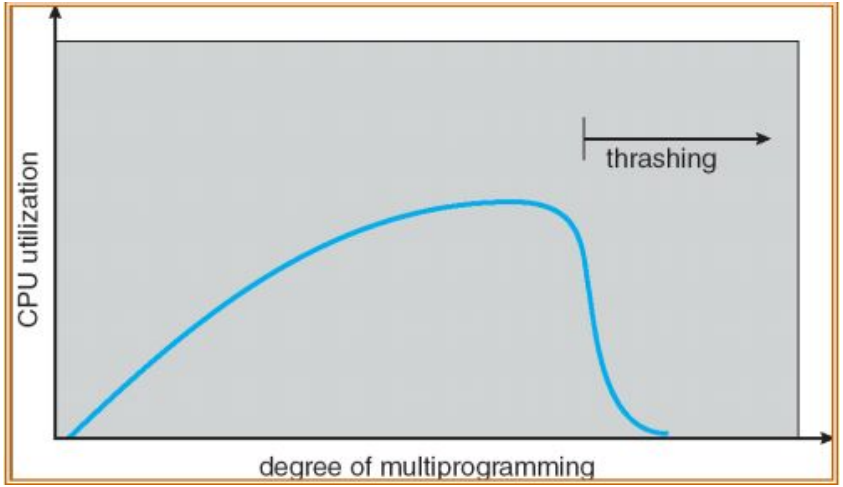
# Exercise: Unfairness of I/O Resource Sharing

- What happens if we increase the number of asynchronous IOs to 4?
- What happens if we increase the number of asynchronous IOs to 32?

# Memory Thrashing

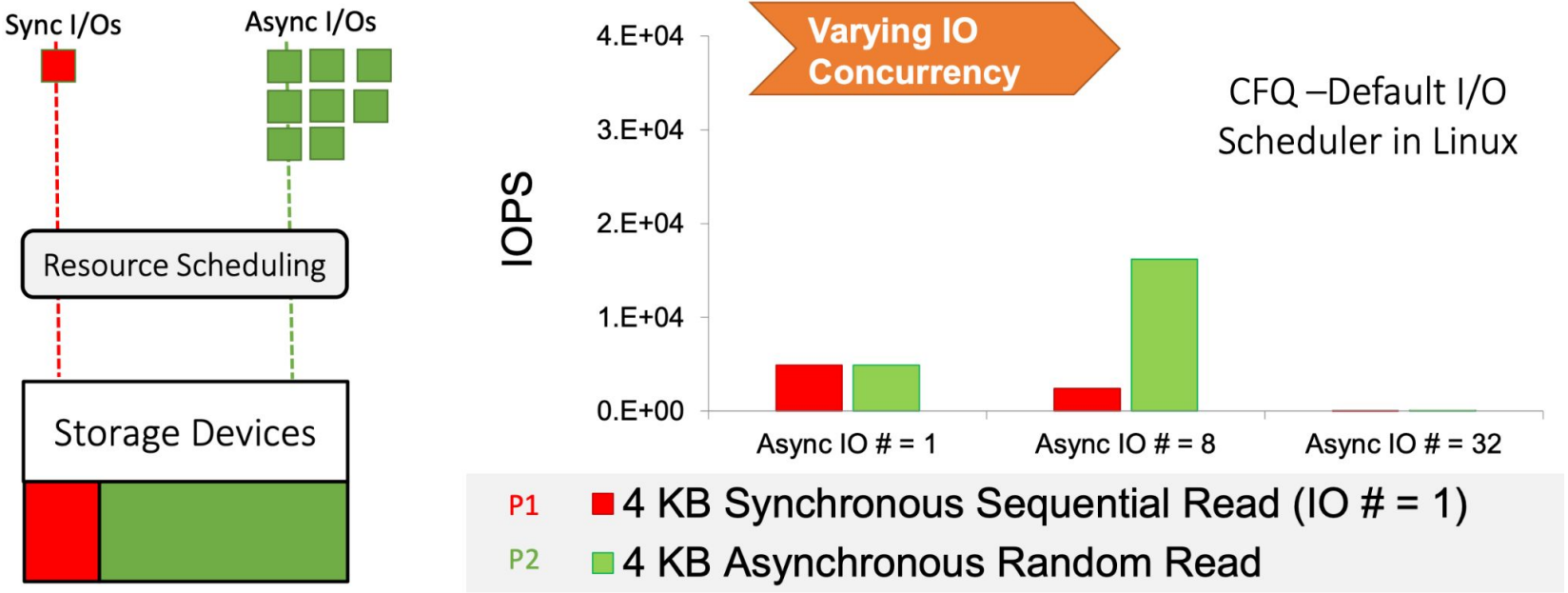
- Memory thrashing is a problem that arises when the memory being accessed is much more than the physical memory.
  - Unpredictive contention
  - Any solution?



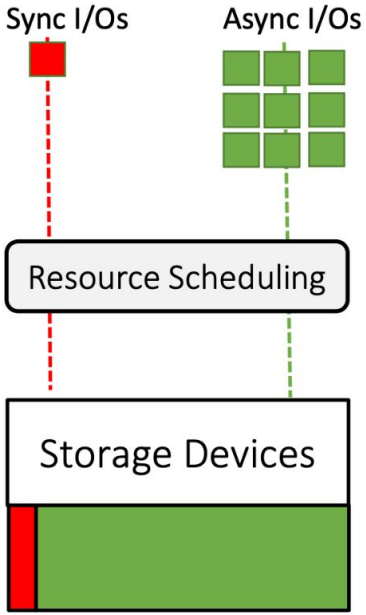



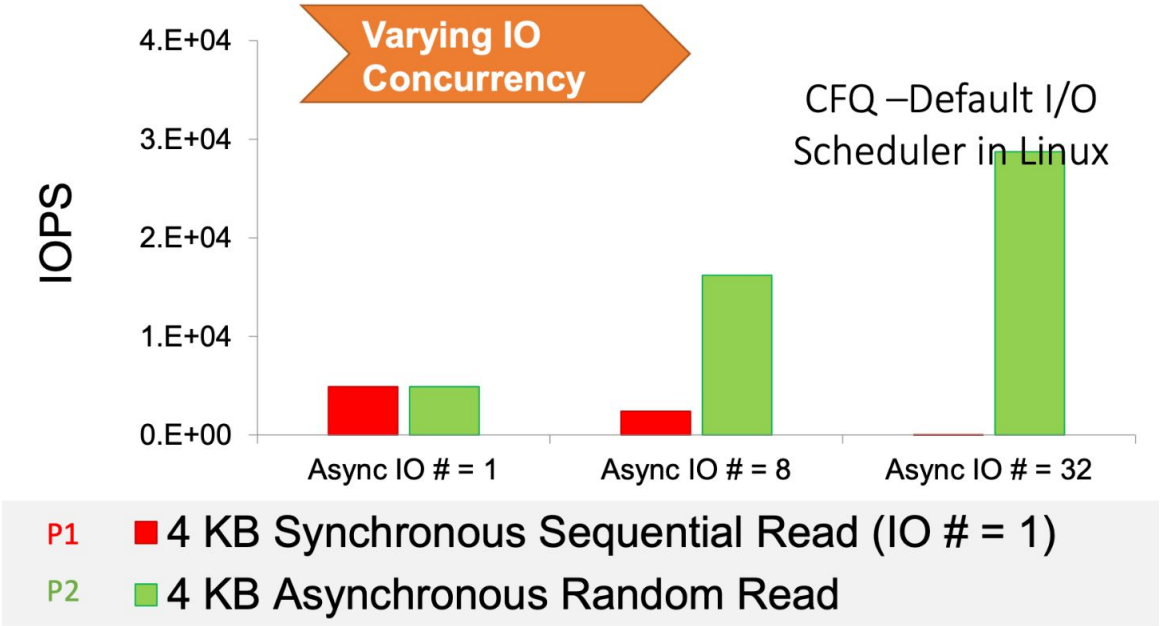

http://www.firmcodes.com/memory-thrashing-in-operating-system/

# Example: Unfairness of I/O Resource Sharing



# Example: Unfairness of I/O Resource Sharing

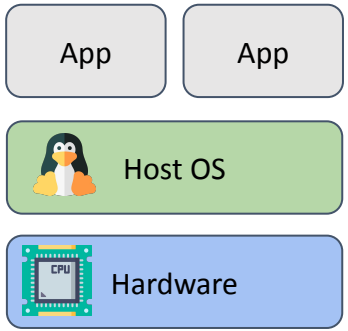


# Pros & Cons of Having Multiple Processes

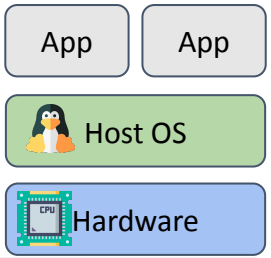
- Pros:
  - Multiple applications (from different users) can run “simultaneously”.
  - Maximize resource utilization
    e.g., multiple processes on a multi-core system

- Cons
  - Contention
  - Dependency/Compatibility Issues
    - The application’s version and the OS version should (strictly) match one another other



# Potential Solutions

- Contention
  - Strong regulation, such as limiting (via metering)
  - More advanced resource management: limit, share, and reservation for multiple resources, such as CPU, memory, block I/O, and network
- Dependency/compatibility
  - Provide processes with their own view of the system
  - With their own running environments (i.e., library versions)
- **Virtualization!**

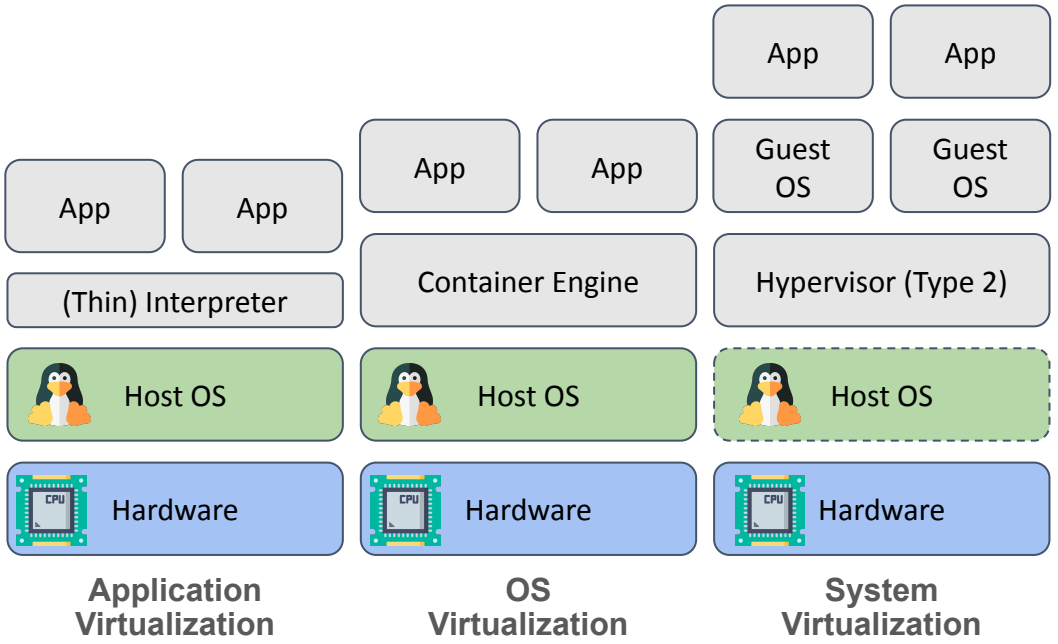


# What is Virtualization?

- The act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.
- Virtualization is about adding a layer for manageability
  - Virtualization support is increasingly being added into various software and hardware

# Types of Virtualization

- **Application Virtualization**
  - E.g., JVM, PVM
- **OS Virtualization**
  - Containers
- **System Virtualization**
  - Virtual Machines
- Network Virtualization
- Storage Virtualization
- Many more...

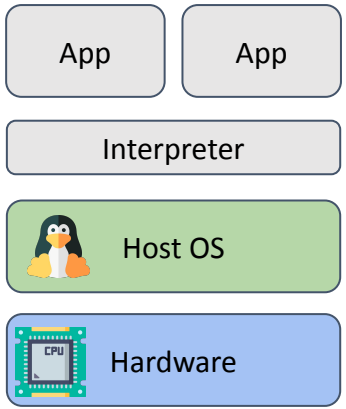


# Application Virtualization

- Purpose: Provide a **platform-independent programming environment** that abstracts the underlying hardware or OS and allows a program to execute in the same way on any platform

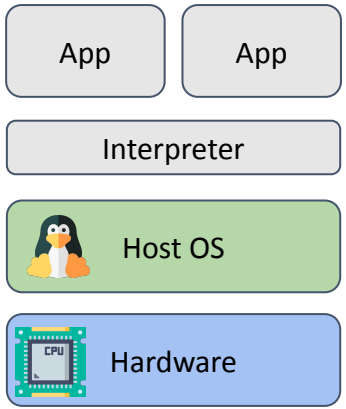
- What is virtualized?
  - Processor
  - System calls
  - Memory management

- Each application is run as a **separate process**
- Examples: Java Virtual Machine



# Application Virtualization

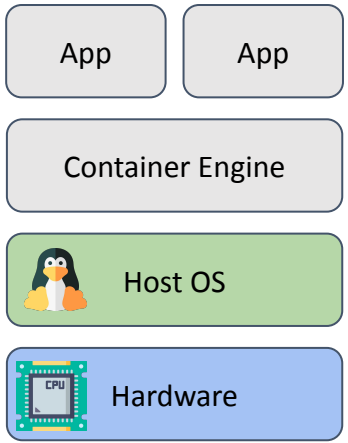
- Benefit
  - Portability
- Drawbacks
  - Language dependent, i.e, not flexible



# OS Virtualization

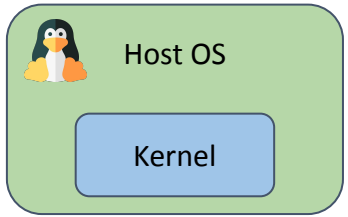
- Virtualizing everything above the Host OS **Kernel**
  - Userspace

- We will revisit this concept!



# Digression: What is a Kernel?

- **Operating System** is system program that provides interface between user and computer.
- **Kernel** is the core component of an operating system. It is also a system program and a part of of OS which converts user commands into machine language.

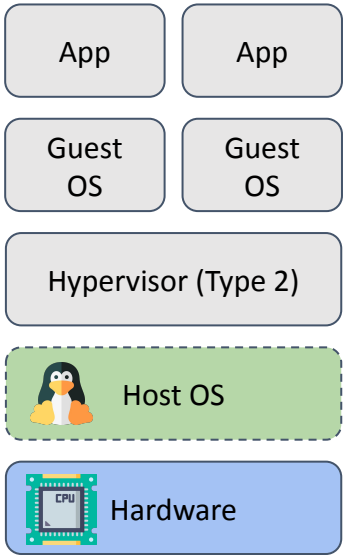


# Digression: What is a Kernel?

| | KERNEL | OPERATING SYSTEM |
|---|---|---|
| Definition | Kernel is part of an OS | Operating System is a system program. |
| Interface | Kernel is an interface between software and hardware | Operating System is an interface between user and hardware. |
| Type | Monolithic and Microkernels | Single and Multiprogramming batch system, Distributed, Realtime |
| Purpose | Manages kernel memory, processes, task and disk | In addition to the responsibilities of Kernel, OS provides added security, UI, window manager and etc. |

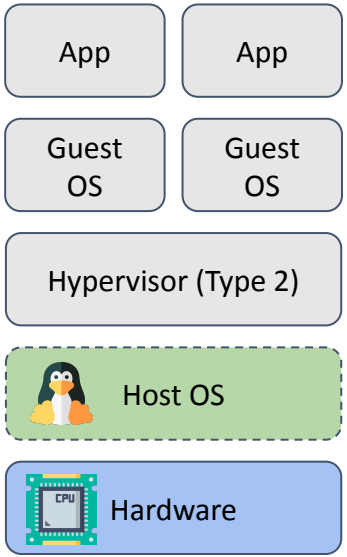
# System Virtualization

- What is virtualized?
  - CPU: vCPU
  - vMem
  - virtual network card
  - virtual hard disks
  - Even usb ports and more
- Creates a virtual computer system, called **virtual machine (VM)** consisting of the set of (virtual) hardware, upon which a full (guest) OS runs.
- **Hypervisor** manages the virtual machines



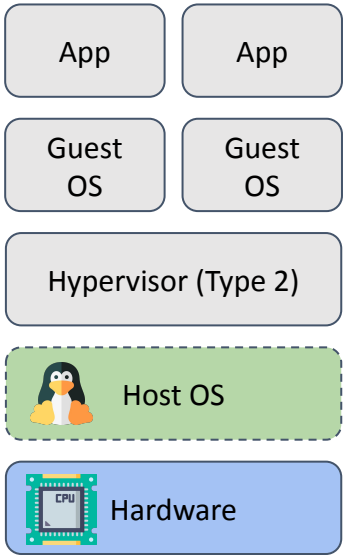
# Hypervisor (Virtual Machine Monitor)

- A thin layer of software that's between the hardware and the operating system
- Virtualizes and manages all (or most) hardware resources for VMs
- Examples
  - Virtualbox
  - **QEMU**
  - VMWare
  - Parallels
  - Xen
  - KVM
- We will cover this more next lecture



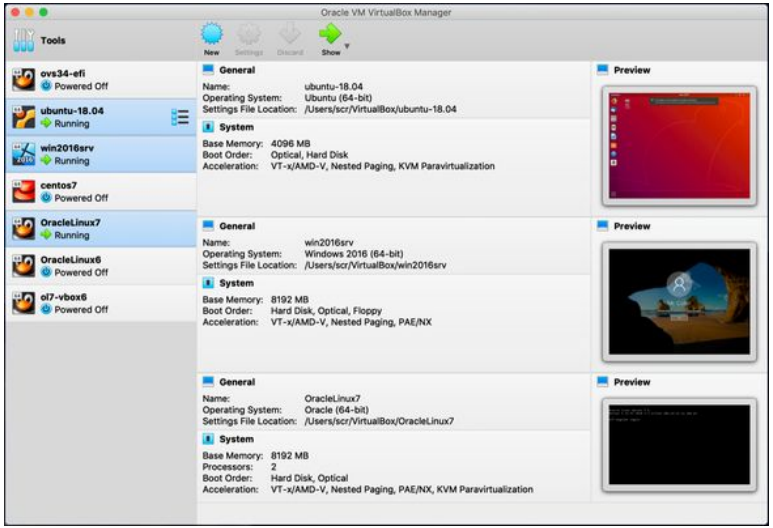
# Virtual Machines

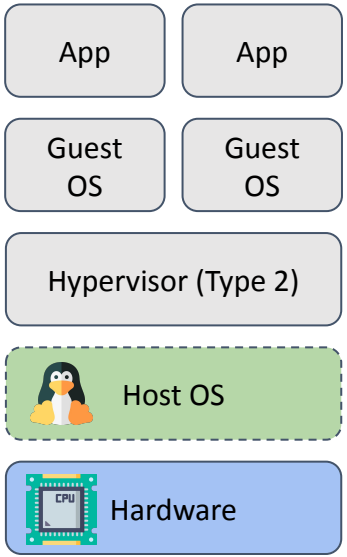
- A fully protected and isolated copy of the underlying physical machine's hardware (i.e., emulated by software)
- Must be created configured
- Portable and reconfigurable as needed
  - Open Virtualization Format (OVF)
  - Disk image



# Virtual Machines

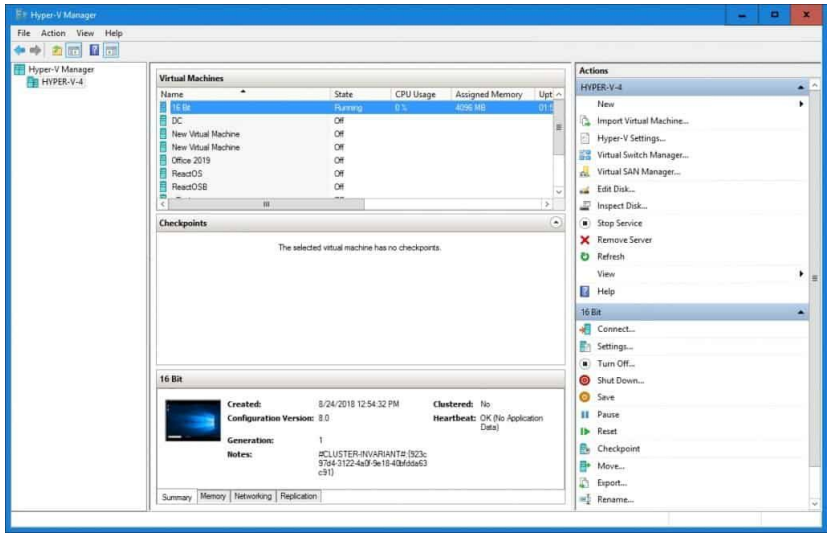
- Example of Virtual Machines
  - Virtual Disk Image file (VDI) for VirtualBox

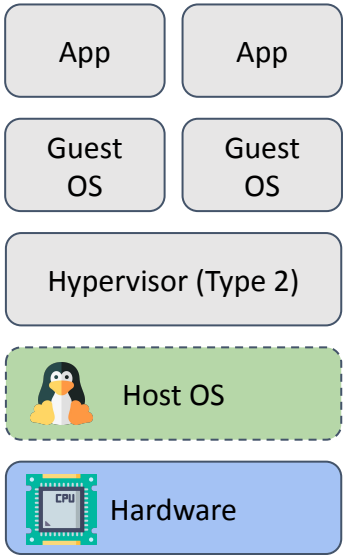


# Virtual Machines

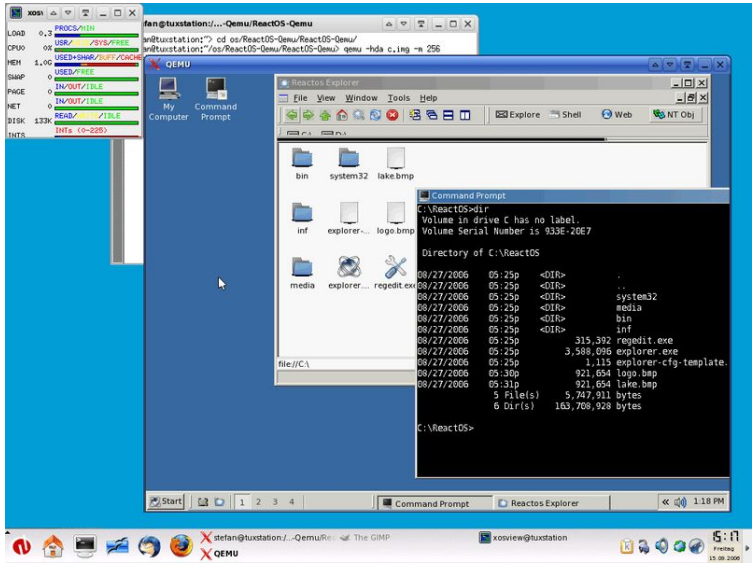
- Example of Virtual Machines
  - Virtual Hard Disk image (VHD) for Hyper-V

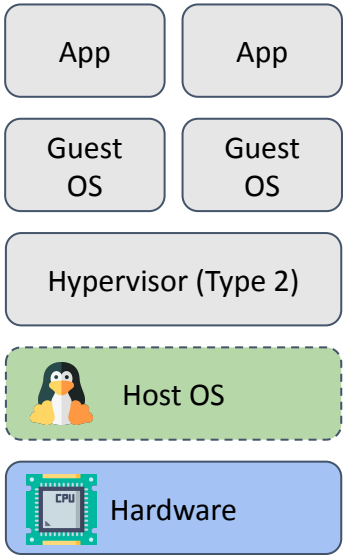


# Virtual Machines

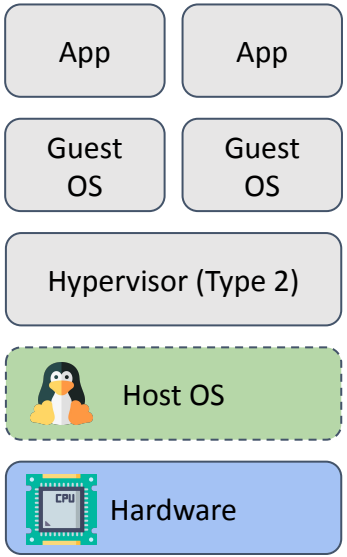
- Example of Virtual Machines
  - “QEMU Copy On Write" (qcow) disk image for QEMU



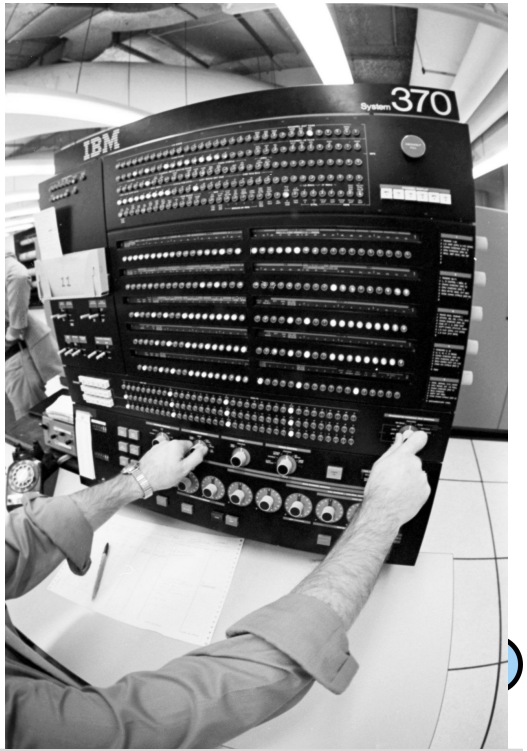
# Pros & Cons of System Virtualization

- Pros
  - Strong isolation (than containers)
  - More levels of protection – VM's kernel and host kernel
  - Full OS
- Cons
  - Performance overhead
  - Large memory & storage footprint
  - More configurations needed



# History of System Virtualization

- IBM VM/370 – A Hypervisor for IBM mainframe
  - Released in 1972
  - Build for IBM System/370 Mainframes
    - Model 145 priced from $705,775 to $1,783,000
  - Reimplementation of earlier CP/CMS OS.
    - Control Program/Cambridge Monitor System
    - Discontinued time-sharing OS of the late 1960s and early 1970s
  - Enables multiple OS environments
    - Full virtualization of the physical machine
- Feasible option when only few & costly machines are around



# History of System Virtualization

- VMs became popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitors
  - Hardware, Hypervisors & OS were often designed together
  -
- Interest died out in the 1980s and 1990s
  - Hardware got much cheaper
  - Operating systems got more powerful (e.g. multi-user)

# History of System Virtualization

- Early x86 computers were underpowered
  - Windows 95 with few applications often maxed-out resources
  - Server work left to expensive server-class computers
- PC computing power increased
  - X86 servers began to appear
  - Mostly under-utilized due to no virtualization
- x86 virtualization surged mid-2000s thanks to the Xen hypervisor

# History of System Virtualization

- Commercial virtual machines for x86 architecture
  - Connectx VirtualPC (now Microsoft)
  - VMware (1999-)
  - KVM (Linux)
- Research virtual machines for x86 architecture
  - **Xen (SOSP '03)**
  - Plex86
- Now widely used in all public cloud providers!

# Agenda for Today

- HW 0 Preview
- Multitenancy
- Virtualizations
- Readings
  - Recommended: CCSA 2.1-2.2
  - Optional:
    - http://www.firmcodes.com/memory-thrashing-in-operating-system/
    - https://www.ibm.com/cloud/learn/virtualization-a-complete-guide

# TODOs!

- HW 0
  - On Camino

- Start looking for teams ASAP!

# Questions?