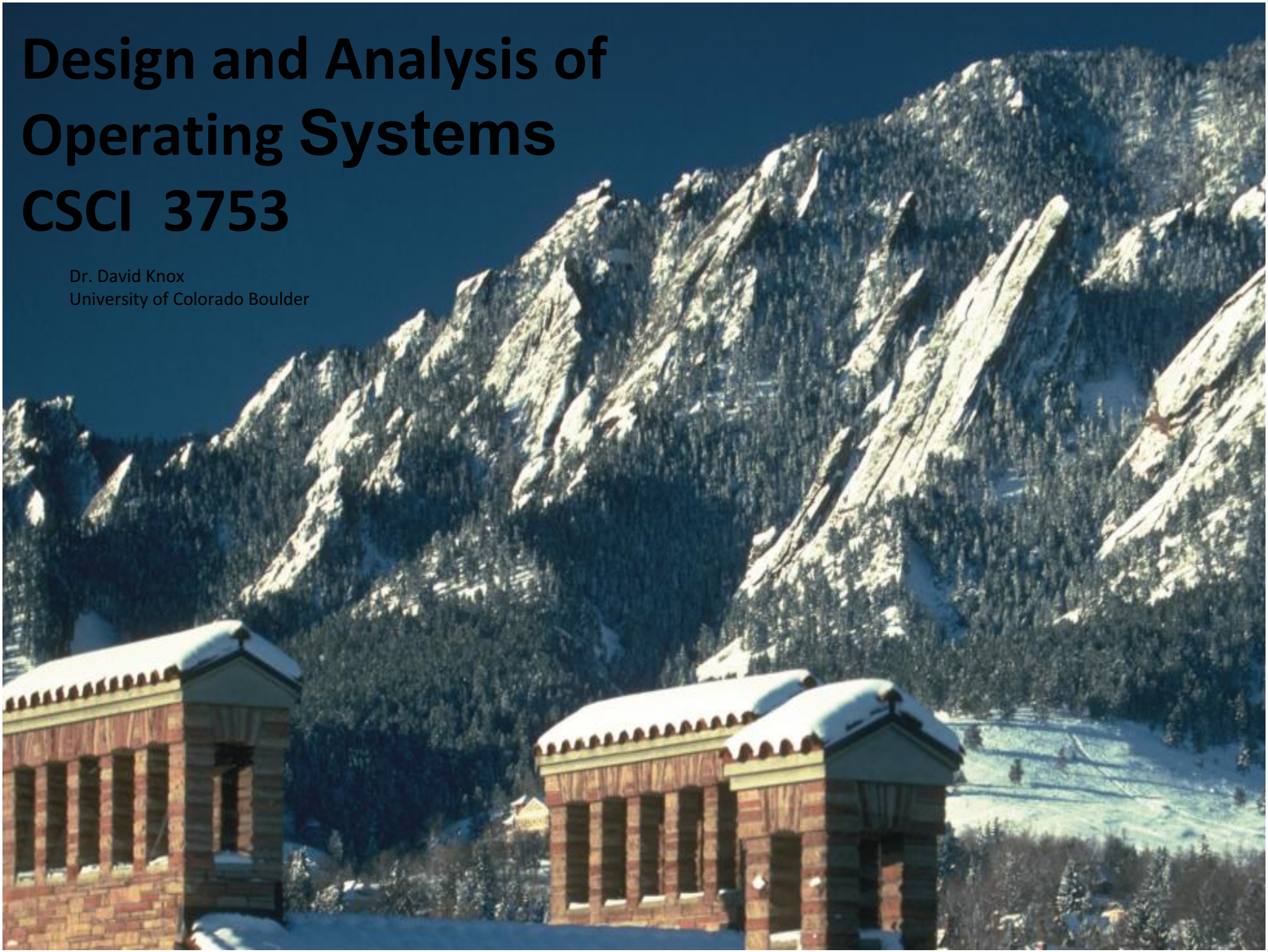
Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder



Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

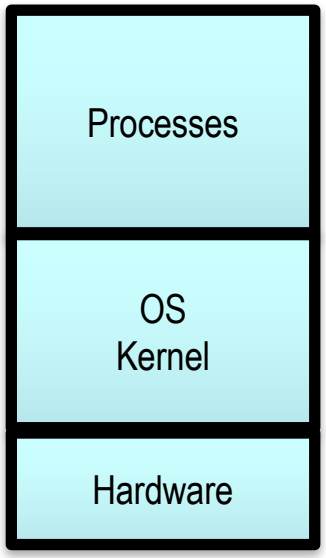
Design and Analysis of Operating Systems CSCI 3753

Virtual Machines

Dr. David Knox
University of
Colorado Boulder

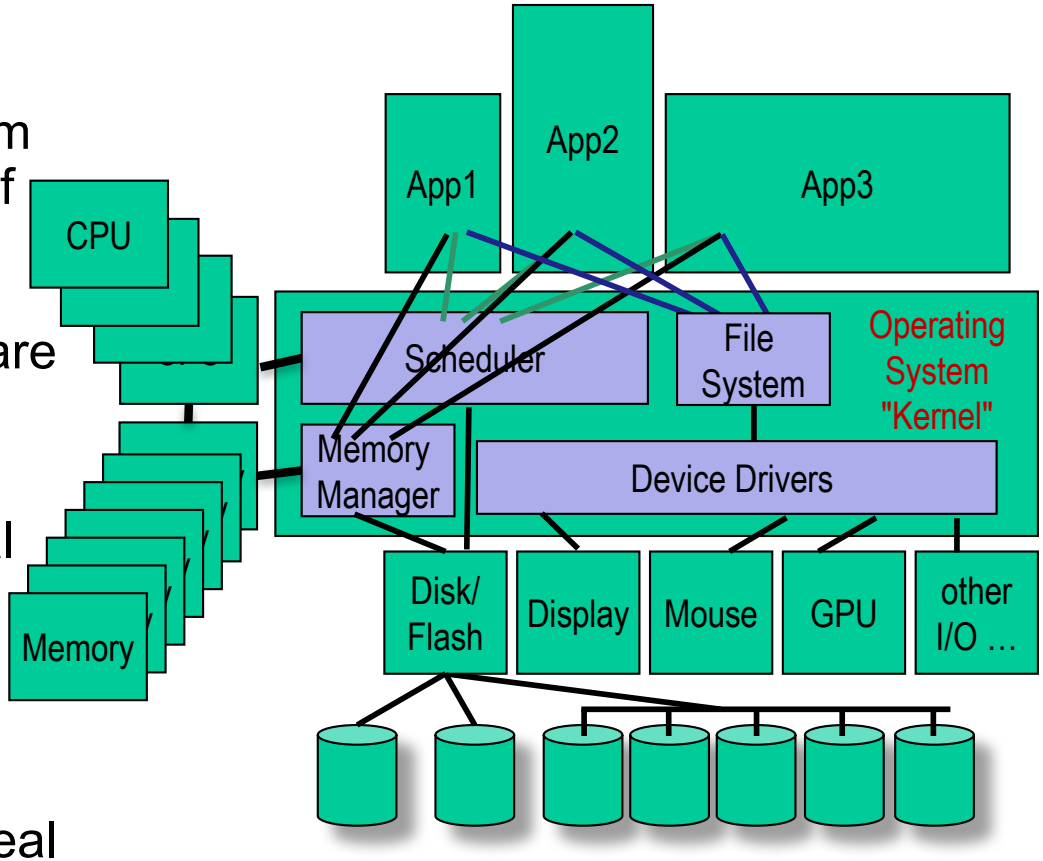
Virtual Machines

Operating System Components

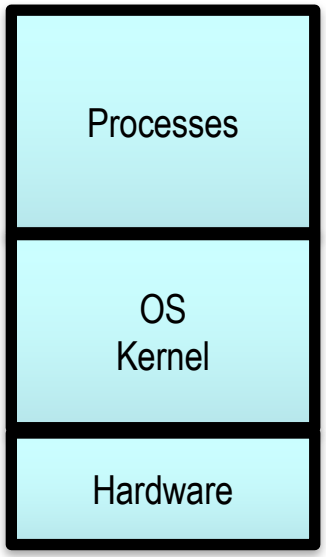


Traditional OS

- An operating system (kernel) is a layer of software between *many* applications and *diverse* hardware
- But, it is an abstraction of a real system with all its components
- Which is also an abstraction of the real hardware

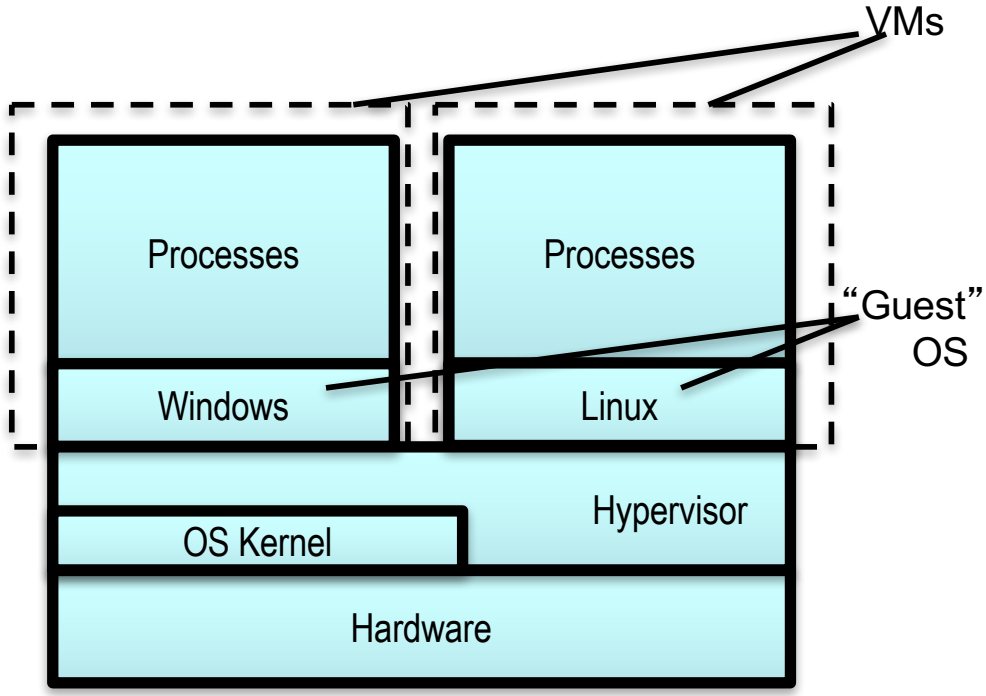


Virtual Machines



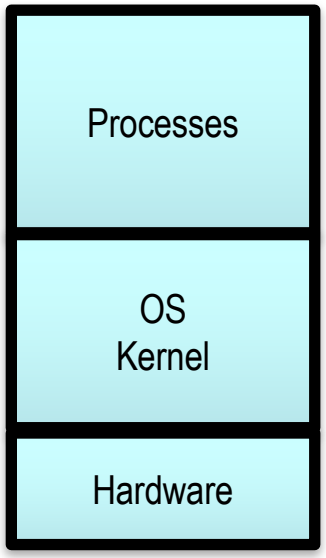
Traditional OS

- A process already is given the illusion that it has its
  - own memory, via virtual memory
  - own CPU, via time slicing
  - own File System
- Virtual machine extends this idea to give a process the illusion that it also has its own hardware
- Extends the concept from a process to an entire OS being given the illusion that it has its own memory, CPU, and I/O devices



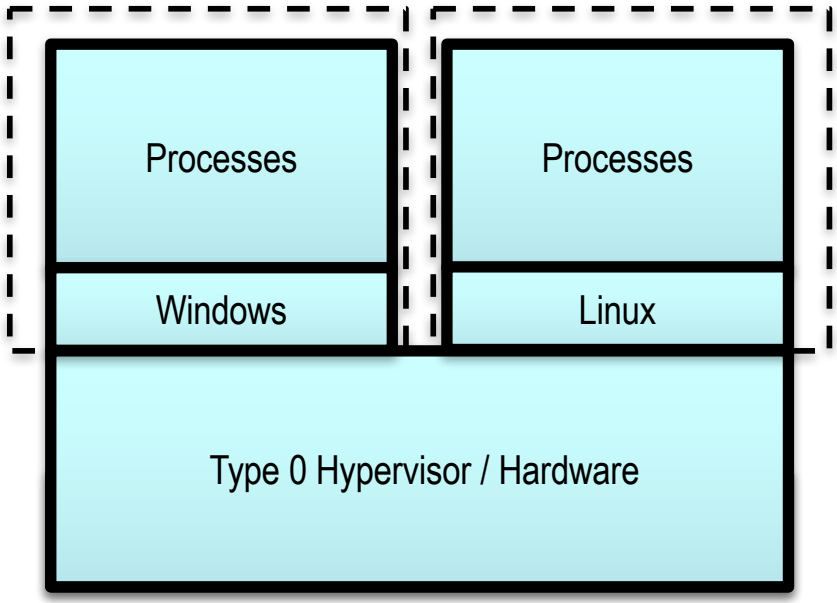
A *Hypervisor* provides a virtualization layer for guest OSs

Hypervisor



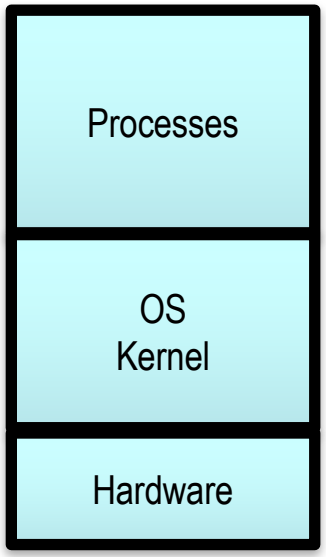
Traditional OS

- There are different styles of virtual machine hypervisors
- Type 0
  - Integrated with hardware



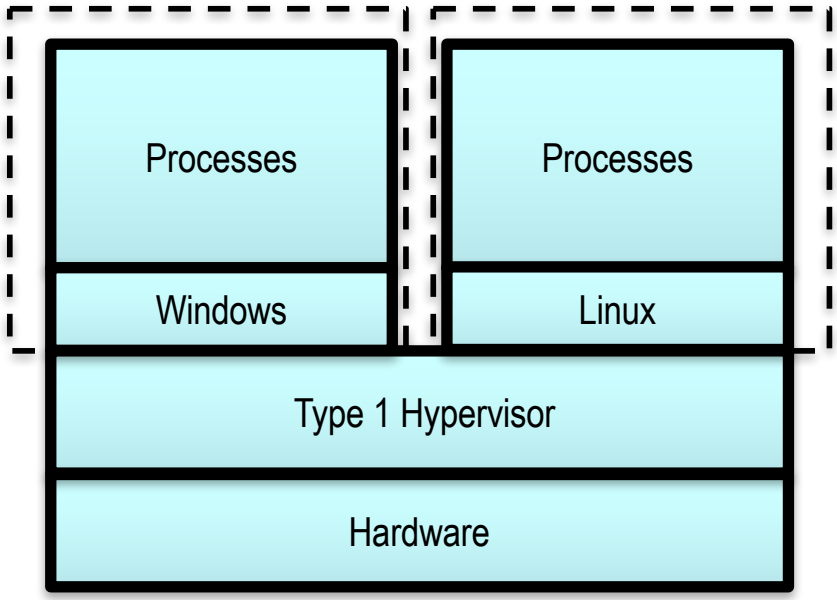
A Type 0 *Hypervisor is integrated with hardware*

Hypervisor



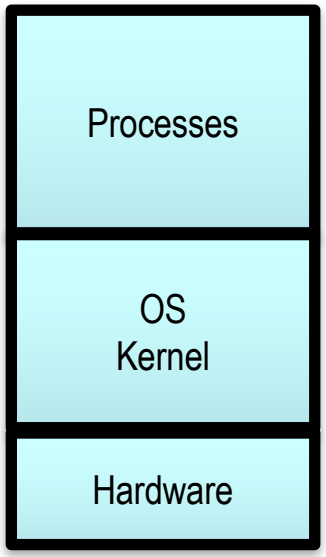
Traditional OS

- There are different styles of virtual machine hypervisors
- Type 1
  - Creates a an API for all access to hardware



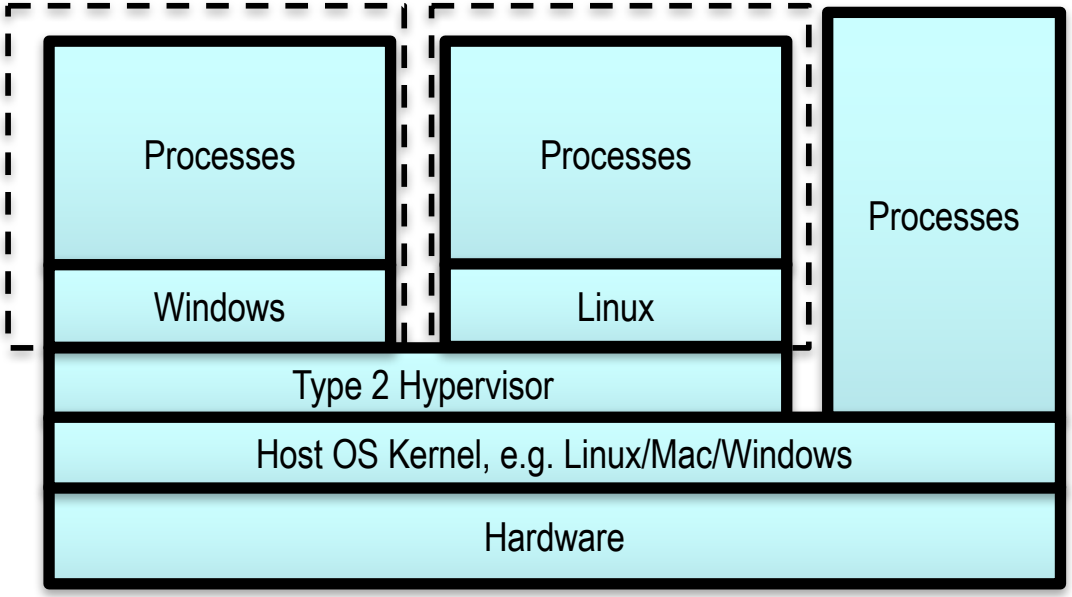
A Type 1 *Hypervisor* provides a virtualization layer for guest OSs and resides just above the hardware.

Hypervisor



Traditional OS

- There are different styles of virtual machine hypervisors
- Type 2
  - Creates a an API for all hardware access via a host OS



A Type 2 Hypervisor essentially runs like an application process on top of the host OS

What is a Virtual Machine?

- An simulated computer running within a real computer
- The virtual computer runs an operating system that can be different than the host operating
- All the requests to access real hardware are routed to the appropriate host hardware, then virtual operating system or applications don't know they are virtual

Virtual Machines Benefits

- **Can run multiple OS's simultaneously on the same host**
- **Copy a VM file to clone the current state of the machine**
- **Fault isolation if an OS fails – doesn't crash another VM.**
  This is also useful for debugging a new OS
- **Easier to deploy applications**
  - Java Virtual Machine (JVM)
  - Deploy app within a customized VM instance
  - Cloud Computing
  - Containers (such as Docker)

Java Virtual Machines

- **Process VMs, e.g. Java VMs**
  - Differ from System VMs in that the goal is not to try to run multiple OSs on the same host, but to provide portable code execution of a single application across different hosts

- **Java applications are compiled into Java byte code that can be run on any Java VM**
  - Java VM acts as an *interpreter* of byte code, translating each byte code instruction into a local action on the host OS

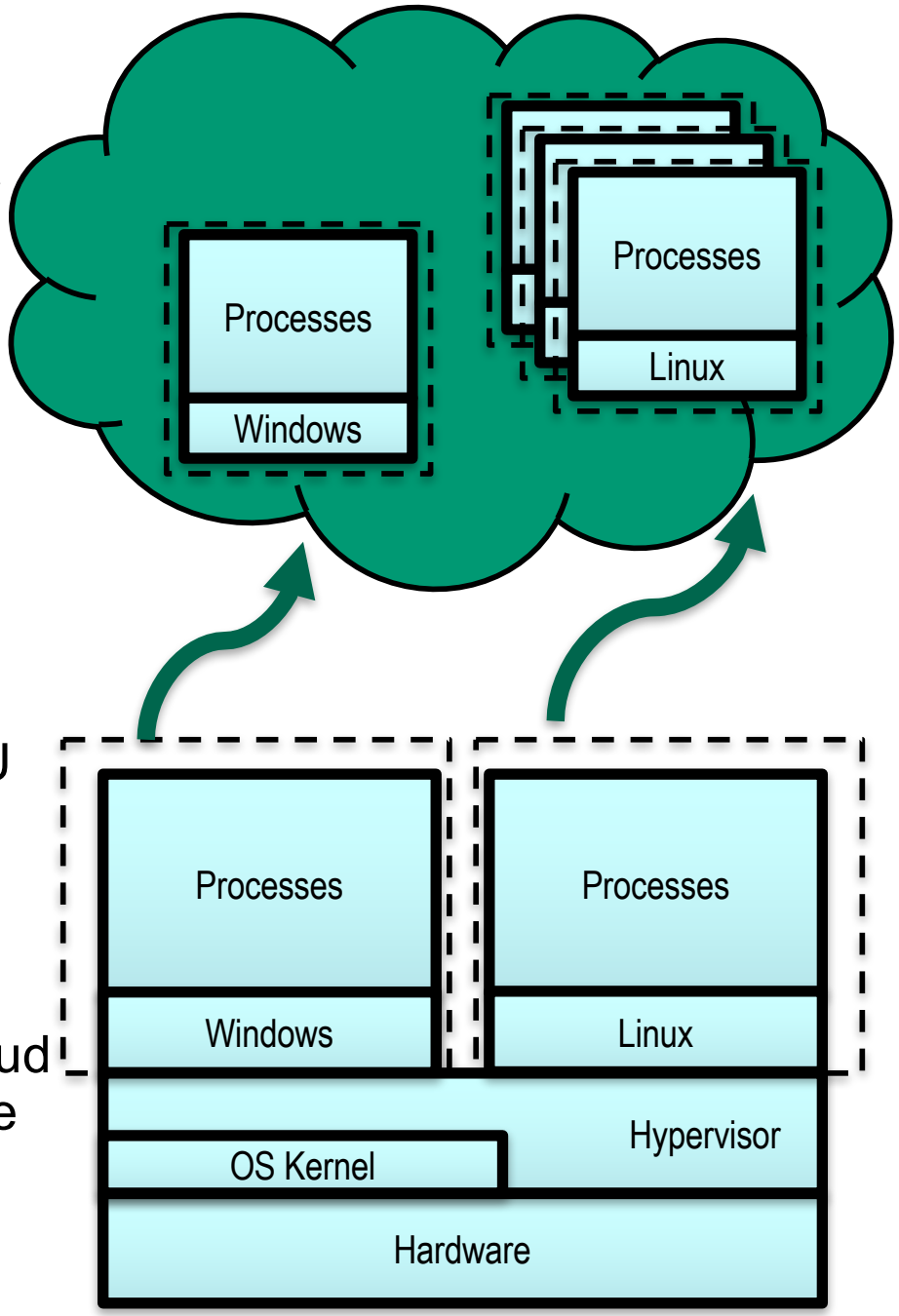
Java Virtual Machines

- **Just in time compilation can be used to speed up execution of Java code**
  - Java byte code is compiled at run time into native machine code that is executed directly on the hardware, rather than being interpreted instruction by instruction
- **Note Java VMs virtualize an abstract machine, not actual hardware, unlike system VMs**
  - i.e. the target machine that Java byte code is being compiled for is a software specification

Cloud Computing

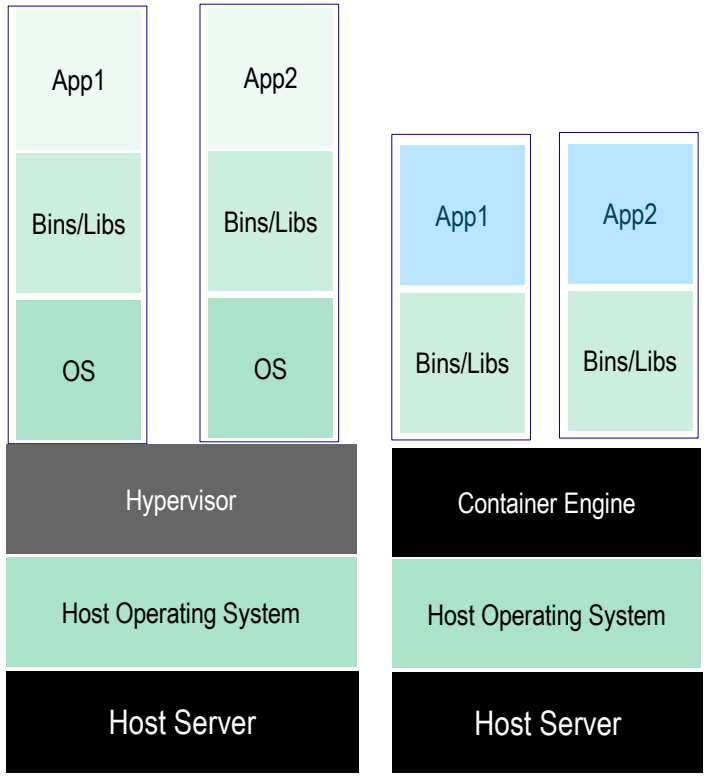
- **Very easy to provision and deploy VM instances on the cloud**

- **Amazon's Elastic Compute Cloud (EC2) uses Xen virtualization**
  - There are different types of VMs or instances that can be deployed:
    - Standard, High-Memory, High-CPU
  - Users can create and reboot their own VMs
  - To store data persistently, need to supplement EC2 with an additional cloud service, e.g. Amazon's Simple Storage Service (S3)



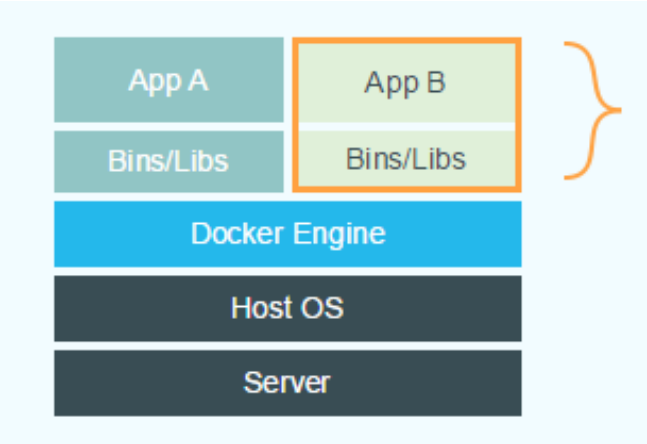
Containers

- Container virtualization isolates the guests
- Instead of virtualizing the hardware, use containers for each virtual environment
- A patched kernel and user tools to run the virtual environments
- The kernel provides process isolation and performs resource management
- Running under the same kernel, they effectively have their own file system, processes, memory, devices, etc.

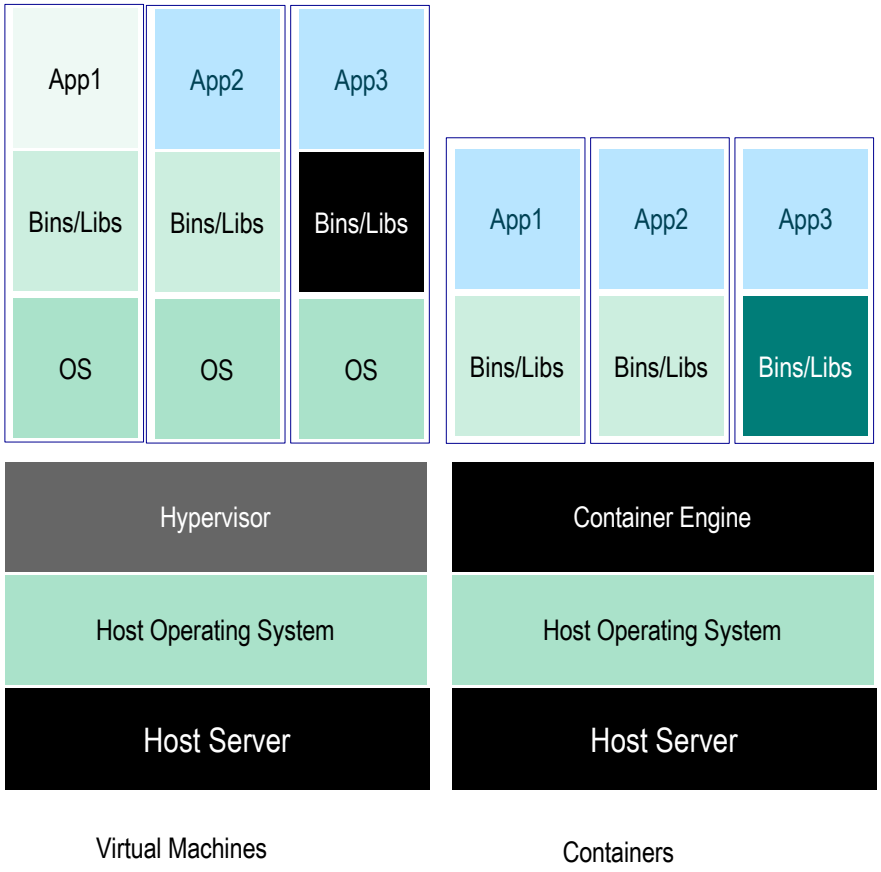


Containers

- **Docker**
  - just the application and its dependencies.
  - isolated process in user-space on the host OS
  - Shares the kernel with others
  - resource isolation and allocation benefits of VMs portable and efficient
  - can migrate the container and application



Containers and VMs Solve Different Problems



Containers are more lightweight and obtain better performance:

- **Portability: VM (GB) vs. Container (MB)**
- **Performance: Containers can boot and restart in seconds, compared to minutes for virtual machines**
- **Management cost: Each VM requires a full functional operating system, and then extra management for system**

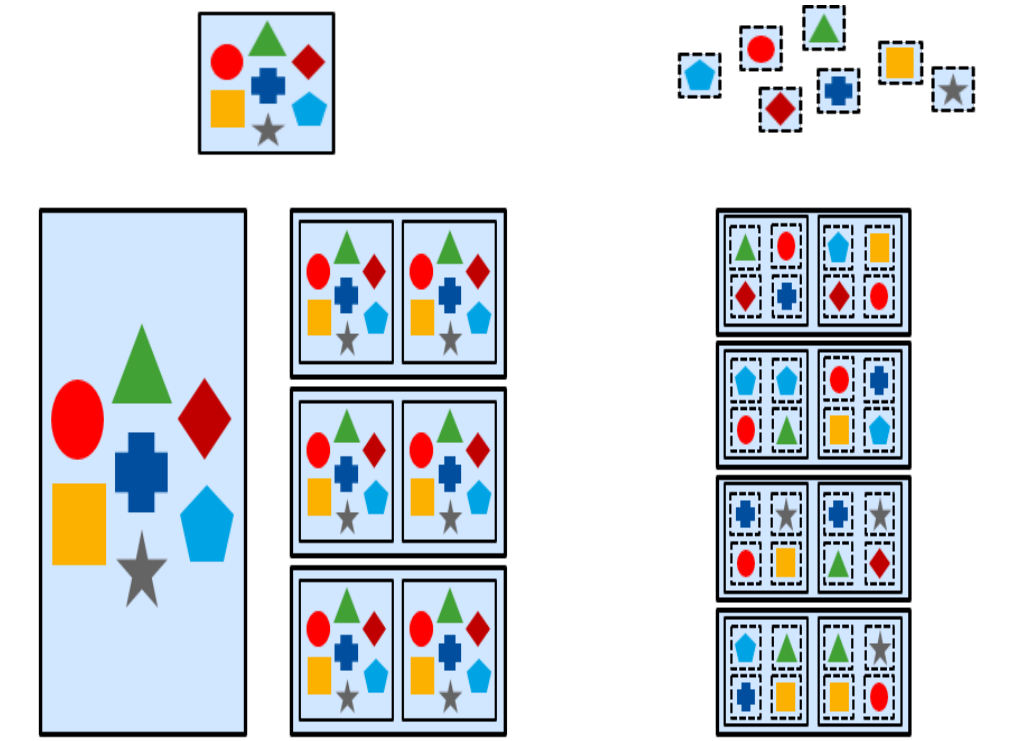
Great advantage to use containers in:

- **DevOps**
- **Batch computing**
- **Lightweight PaaS**
- **Microservices**

Next-Generation Application Development Requires Efficient Container Management

Microservices Architecture:

- **Decomposed into small pieces**
- **Loosely coupled**
- **Easier to scale development**
- **Improved fault isolation**
- **Each service can be developed and deployed independently**
- **Eliminates any long-term commitment to a technology stack**



Scales by size ... or monolithic replication.
Changes monolithically.

Scales by microservice replication.
Changes by microservices.

Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of
Colorado Boulder