Virtual machines

What are virtual machines?

- Abstract hardware of a single computer into several different execution environments(VM)
- Users are given their own VM, they can run any of the OS or software packages that are available on the underlying machine
- The resources of the host are shared
- Components:
    - Several components
        - **Host** – underlying hardware system
        - **Virtual machine manager (VMM) or hypervisor** – creates and runs virtual machines by providing an interface that is identical to the host • (Except in the case of paravirtualization)
        - **Guest** – process provided with a virtual copy of the host • Usually an operating system

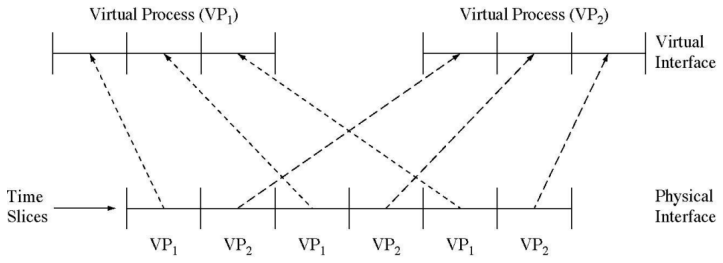
Illusion created using:

- CPU scheduling
- Memory management
- partitioning the disk (mini-disk - partitioning the tracks on physical discs)

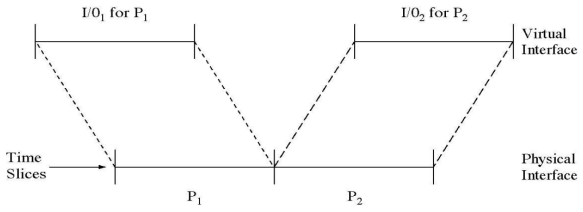
How to share resources - Methods?

- **Time multiplexing**: scheduling a serially reusable resource among several users

Time-multiplexing the processor

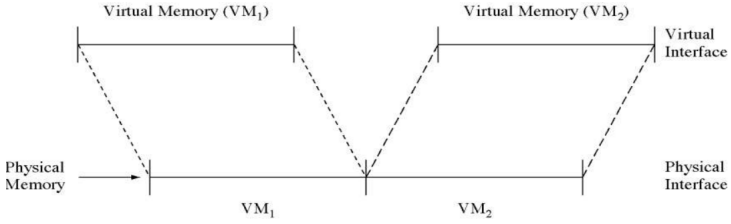


Time-multiplexing I/O devices



- **Space multiplexing**: dividing a multiple-use resource up among several users

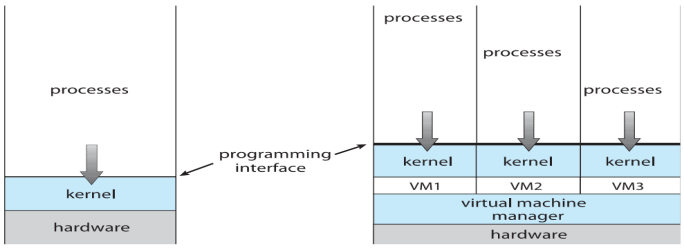
Space-multiplexing memory



Types of virtual computers

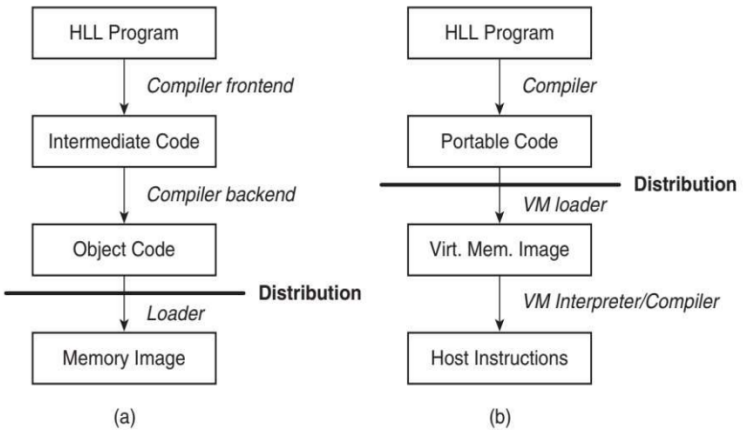
➔ Processor virtualized to processes • *mainly time-multiplexing* •
➔ Memory virtualized to address spaces • *space and time multiplexing* •
➔ Disks virtualized to files • *space-multiplexing* • *transforming*

System Models



Non-virtual machine

Virtual machine



High-Level Language Environments. *(a) A conventional system, where platform-dependent object code is distributed; (b) an HLL VM environment, where portable intermediate code is "executed" by a platform-dependent virtual machine.*

Benefits/ Advantages/ Features of Virtual Machines:

- Run multiple, different OSes on a single machine
- Host system protected from VMs, VMs protected from each other i.e. A virus less likely to spread
- Freeze, suspend, and run VM:
  - move or copy somewhere else and resume
  - Clone and run both the original and copy.
- Great for OS research, better system development efficiency
- Steps are simpler and faster than with a physical machine install

---

VM lifecycle:

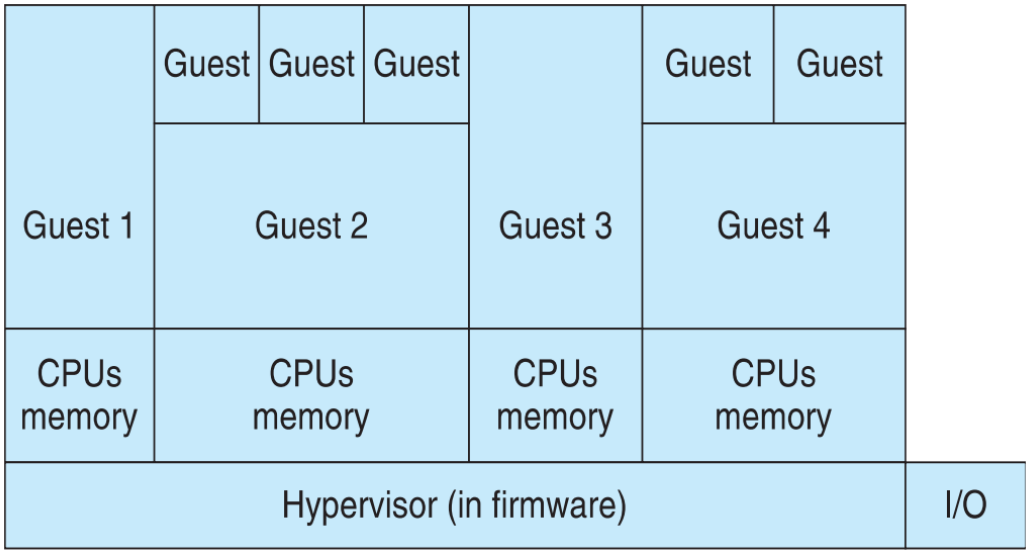
1. Created by VMM
2. Resources assigned to it (number of cores, amount of memory, networking details, storage details)
   -> In type 0 hypervisor, resources are usually dedicated
   -> Other types dedicate or share resources, or a mix
3. When no longer needed, the VM can be deleted, freeing resource

---

Types of VMs:

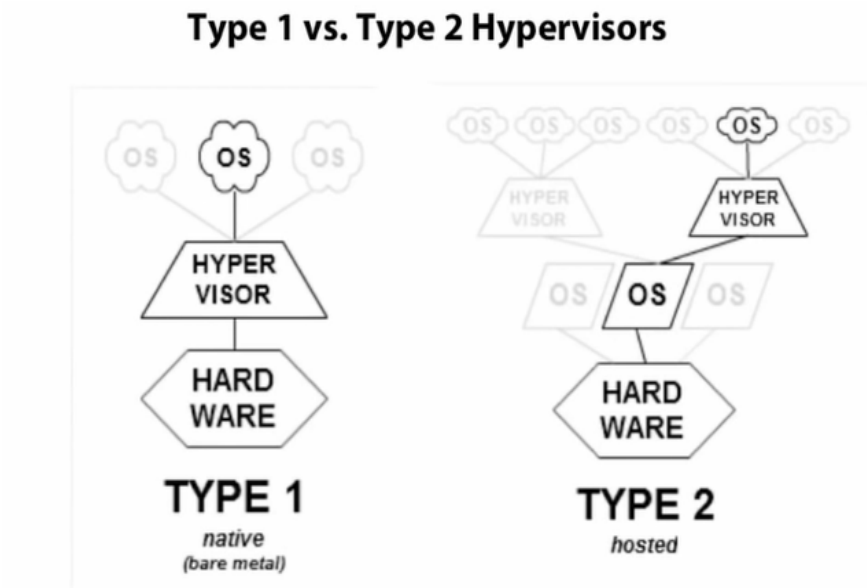
★ **Type 0 hypervisors** - Hardware-based solutions that provide support for virtual machine creation and management via firmware (Firmware is a microcode or program that is embedded into the memory of hardware devices to help them operate).

- A HW feature implemented by firmware
- OS needs nothing special, VMM is in the firmware
- Smaller feature set than other types
- Each guest has a dedicated HW
- Can provide virtualization-within-virtualization (guest itself can be a VMM with guests
- I/O a challenge as difficult to have enough devices, and controllers to dedicate to each guest



Type 1 vs. Type 2 Hypervisors

- **Type 1 Hypervisor**
  - Loaded directly on the hardware
    - Hyper-V
    - ESXi
    - KVM
- **Type 2 Hypervisor**
  - Loaded in an OS running on the hardware
    - Workstation
    - Oracle VM (Virtual Box)
    - Parallels
    - Fusion



Type 1 hypervisors (V1) -

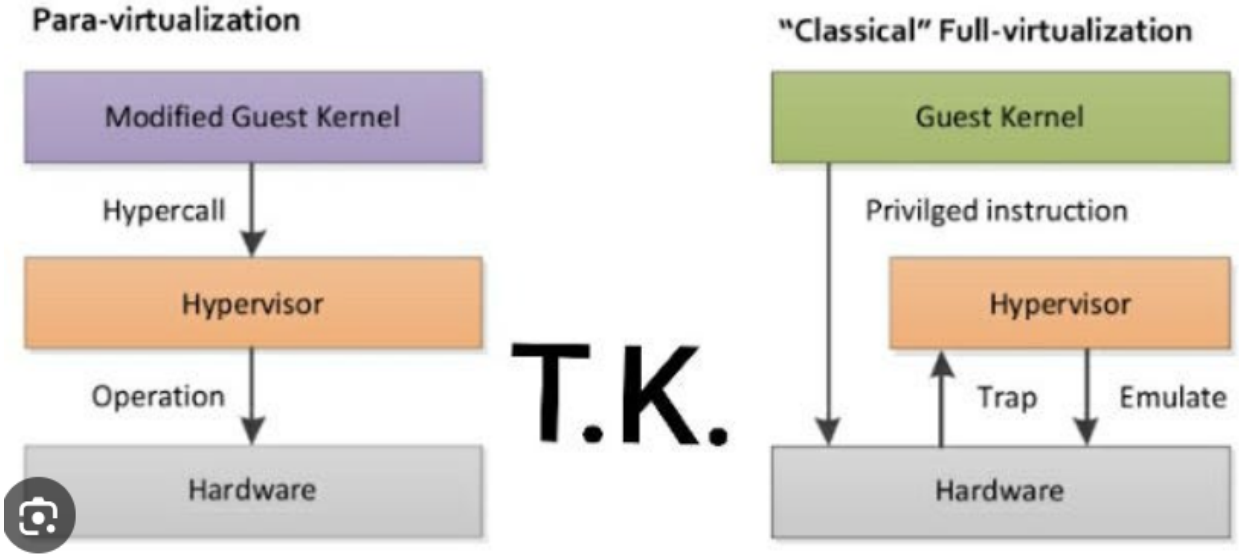
- Special-purpose operating systems that run natively on HW to provide virtualization
- Commonly found in company data centers
- Can Move guests between systems to balance performance
- Refer figure

Type 2 hypervisors - Applications that run on standard operating systems but provide VMM features to guest operating systems

- VMM is simply another process, run and managed by the host
- Even the host doesn't know they are a VMM running guests
- Tend to have poorer overall performance because can't take advantage of some HW features
- require no changes to host OS

Paravirtualization -

- Paravirtualization is a computer hardware virtualization technique that improves the performance of virtual machines (VMs).
- It does this by modifying the guest operating system (OS) to allow VMs to have an interface similar to the underlying hardware



Programming Environment Virtualization -

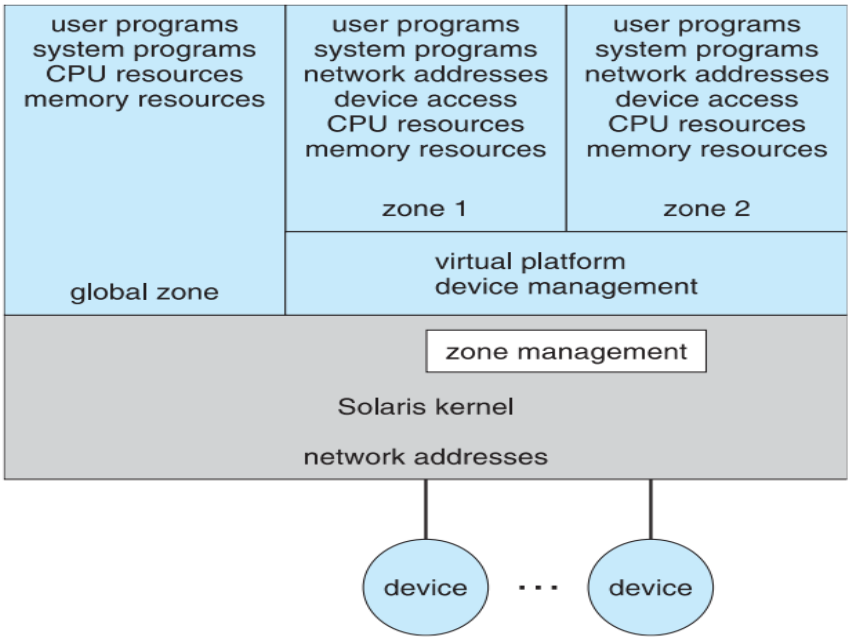
- Programming language is designed to run within a custom-built virtualized environment
- virtualization is defined as providing APIs
    - a set of features made available to a language and programs written in that language to provide an improved execution environment
- Programs written in Java run in the JVM no matter the underlying system

Emulation -

- hardware or software that enables one device (named Host) to function like other systems (named Guest)
- Necessary to translate all guest instructions from the guest CPU to the native CPU
- The company replacing outdated servers with new servers containing different CPU architectures, but still wants to run old applications
- Emulators in gaming

Application Containment/zones -

- goals of virtualization are segregation of apps, performance, and resource management, easy start, stop, move, and management
- If applications compiled for the host operating system, don't need full virtualization to meet these goals
- Only one kernel running – host OS
- OS and devices are virtualized, providing resources within the zone with the impression that they are only processes on system
- CPU and memory resources divided between zones



Q. How do VMMs schedule CPU use when guests believe they have dedicated CPUs?

1. When guests believe they have dedicated CPUs, it typically means they are allocated specific CPU resources for their use.
2. The VMM must ensure fair and efficient scheduling of these resources among multiple virtual machines (VMs) this can be achieved using
    a. The VMM allocates time slices to each VM, allowing them to execute on a physical CPU for a certain period. After the time slice expires, the VMM can switch to another VM, providing the illusion of dedicated CPU time

Q. How can memory management work when many guests require large amounts of memory?

Memory Page Sharing:

- Transparent Page Sharing: Identical memory pages among different VMs can be identified, and the hypervisor can share these pages, saving physical memory. This is particularly effective when multiple VMs are running similar operating systems or applications.

Memory Ballooning:

- Dynamic Memory Ballooning: The hypervisor can dynamically adjust the amount of memory allocated to a VM based on its current needs. This involves inflating or deflating a balloon inside the guest to reclaim or allocate memory.

Memory Reservation and Limits:

- Reservation: Administrators can reserve a specific amount of physical memory for a VM, ensuring that it always has access to at least that much memory.
- Limits: Upper limits on memory usage can be set for VMs to prevent them from consuming excessive resources.

Page Swapping:

- Memory Paging: In cases of extreme memory contention, the hypervisor can swap out less frequently used memory pages to disk. However, this can impact performance if not managed carefully.

OS components involved in Virtual machines

1. Scheduling

- Even single-CPU systems act like multiprocessor ones when virtualized
- Guests configured with certain number of VCPUs and Can be adjusted throughout life of VM
- VMM can allocate dedicated CPUs, each guest much like a native operating system managing its CPUs
- VMM can use standard scheduling algorithms to put threads on CPUs
- **Cycle stealing by VMM** and **oversubscription of CPUs** means guests don't get CPU cycles they expect
    - Problems
        - Poor response times for users of guest
        - Time-of-day clocks incorrect
    - VMMs provide application to fix this incorrect time issue

2. Memory Management

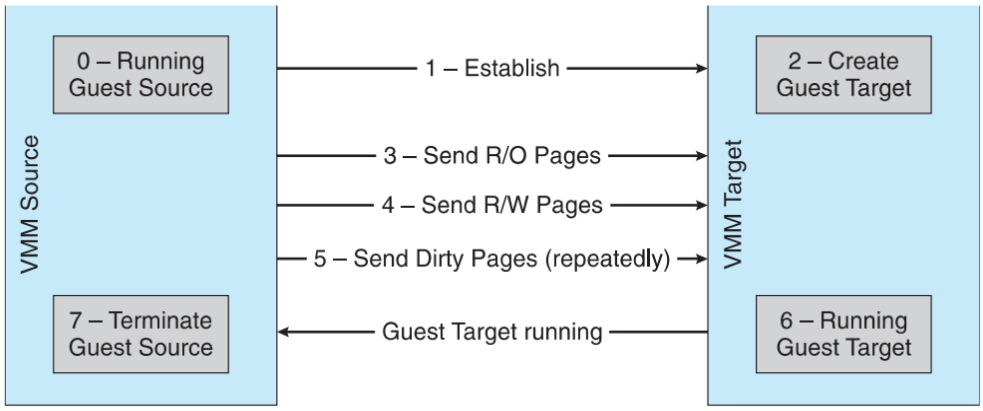
- Also suffers from oversubscription
    - requires extra management efficiency from VMM
- VMware ESX methods
    - Deduplication by VMM determining if same page loaded more than once, memory mapping the same page into multiple guests
    - Balloon memory manager communicates with VMM and is told to allocate or deallocate memory to decrease or increase physical memory use of guest

3. Live Migration

- Running guest can be moved between systems, without interrupting user access to the guest or its apps
- Very useful for resource management, maintenance downtime windows, etc
    - The source VMM establishes a connection with the target VMM
    - The target creates a new guest by creating a new VCPU, etc
    - The source sends all read-only guest memory pages to the target
    - The source sends all read-write pages to the target, marking them as clean

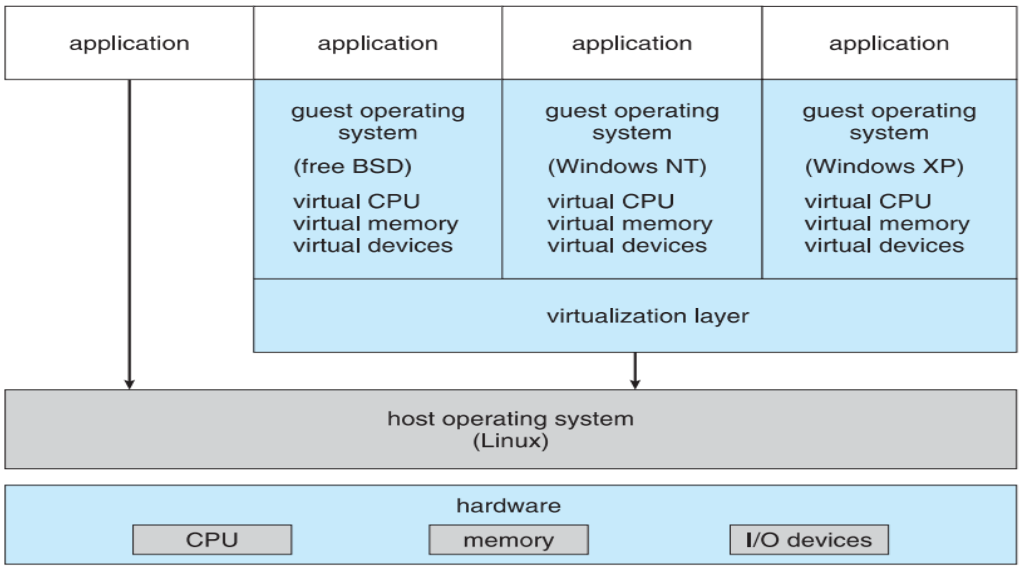
- The source repeats step 4, as during that step some pages were probably modified by the guest and are now dirty
- guest sends VCPU's final state, sends other state details, sends final dirty pages, and tells target to start running the guest
- Once target acknowledges that guest running, source terminates guest

Live Migration of Guest Between Servers



Examples:

1. VMware
    a. VMware Workstation runs on x86
    b. Runs as application on installed host operating system(Type 2)
    c. Lots of guests possible, including Windows, Linux
    d. all runnable concurrently
    e. Virtualization layer abstracts underlying HW, providing guest with is own virtual CPUs, memory, disk drives, network interfaces



2. Java Virtual Machine
    a. Example of programming-environment virtualization
    b. Write once, run anywhere
    c. API library used
    d. Each Java object is compiled into architecture-neutral bytecode output (.class) which the JVM class loader loads
    e. Includes garbage collection
    f. Made faster by just-in-time (JIT) compiler that turns bytecodes into native code and caches them