OPERATING SYSTEMS

Virtual Machines

Introduction

Benefits

Types & Implementation techniques

Virtualization concerns

VM & OS components

Introduction

- Single computer hardware into several different execution environments
  - Layered approach
  - Creates virtual system (virtual machine, or VM) on which operation systems or applications can run
- Multiple operating systems executes concurrently on single machine(host) as dedicated machine (virtual)
- Lifecycle
  - Created by Virtual Machine Monitor (VMM also known as Hypervisor)
  - Resources assigned to it (number of cores, amount of memory, networking details, storage details)
  - Resources usually dedicated, share resources, or combination
  - VM can be deleted if no more required, freeing resources

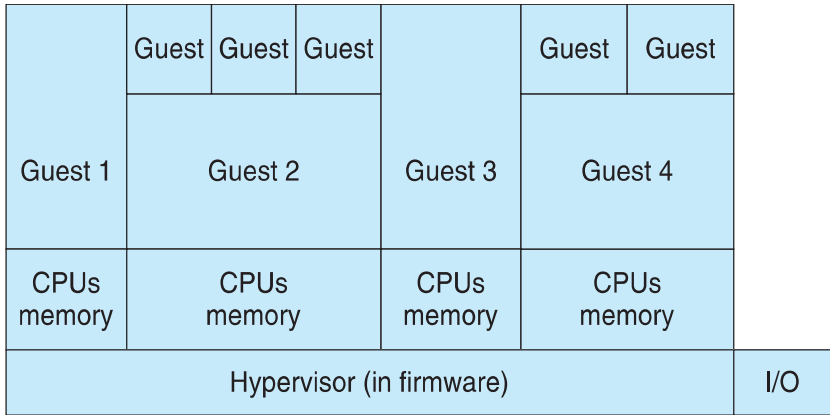
Introduction

- Advantages
  - Security
  - Reusability
    - Cloning
    - Creating image
  - Resource utilization
  - System Consolidation
  - Stop, suspend and Run VM
  - Templating
  - Live Migration
  - Ease of management
- Introduces the new era of computing i.e. Cloud Computing

Types of Virtual Machines

Type 0 Hypervisor

- Hardware-based solutions
- Provide support for virtual machine creation and management via firmware
  - Hardware feature implemented by firmware
  - OS need to nothing special, VMM is in firmware
  - Smaller feature set
  - Each guest has logical dedicated H/W



- Difficult to have enough I/O devices, controllers to dedicate to each guest
- VMM uses control partition which allows guests to communicate for shared I/O
- Provide virtualization-within-virtualization easily
- e.g. IBM LPARs(Logical PARtitions) and Oracle LDOMs(Logical Domains)

Types of Virtual Machines

Type 1 Hypervisor

- Operating-system-like software built to provide virtualization
- Commonly used for datacenters
  - Become “datacenter operating systems”
    - Consolidation, Cloning and Live Migration
- Special purpose operating systems
  - OS provides services such as CPU and memory management
  - Also create, run and manage guest OS
  - Can run on Type 0 hypervisors
  - Run in kernel mode (kernel level virtualization)
  - Implement device drivers for host H/W
  - Guest OS are unaware about VMs
  - e.g. VMware ESX, Joyent SmartOS, and Citrix XenServer

Types of Virtual Machines

Type 1 Hypervisor

- Another variation is a general purpose OS that also provides VMM functionality
  - Perform normal duties as well as VMM duties
  - Treats guest OS as an process
  - Provides less feature than dedicated Type 1 hypervisors
  - Nested virtualization
  - e.g. Red Hat Enterprise Linux (RHEL) with KVM(Kernel based Virtual Machine), Windows with Hyper-V, Oracle Solaris

Types of Virtual Machines

Type 2 Hypervisor

- Application-level VMM
  - Less OS involvement in virtualization
  - Another process run and managed by host
  - Poorer overall performance
  - Host is unaware about VMs
  - Runs on general purpose OS
  - No special host H/W or OS requirement
  - Used for educational or experimental purpose
  - e.g. VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

Types of Virtual Machines

Para-virtualization

- Guest OS is modified to work in cooperation with VMM
- Increases performance by proper H/W optimization
- Shared host is visible to modified guest OS
  - Each device might be shared via circullar buffer or via other techniques
  - Efficient I/O
- Guest OS can mostly perform read-only operations, other required operations are performed via hypercall
    - E.g. Xen VMM

Types of Virtual Machines

Programming Environment Virtualization

- VMMs do not virtualize real hardware
- Create an optimized virtual system
- Programming language is designed to run within custom-built virtualized environment
- Virtualization provides APIs that define a set of features available to a language
- And provide an improved execution environment for programs written in those particular languages
- Similar to interpreted languages
  - e.g. Oracle Java Virtual Machine, Microsoft .Net framework

Types of Virtual Machines

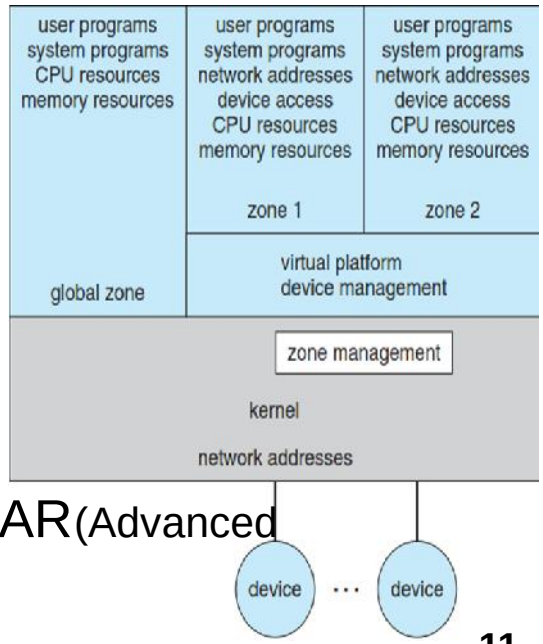
Emulators

- Intends to run software (application or OS) on different machines regardless of their different instruction set
- Allows guest to run on different CPUs
- Translates guest instruction from guest CPU to native CPU
- Correct Emulator should have complete CPU instruction set, difficult for emulators' creators
- Slower execution due to mapping, requires much faster hardware then the mapped one
- Used for gamming and platform porting purpose
  - e.g. ePSXe(for PS), Project 64(for Nentindo), Snes 9x EX(for Android) etc.

Types of Virtual Machines

Application Contentment

- There is requirement for isolation of apps, performance, users and resource management
- Which is easy to start, stop, move, and manage
- If applications compiled for the host operating system, don't need full virtualization
- Creates virtual layer between OS and apps
  - One host OS kernel
  - OS and devices are virtualized for each zone
  - Devices & resources are shared or divided
  - Each zone is independent from other
  - e.g. Oracle Solaris Zones, BSD jails, IBM AIX WPAR(Advanced Interactive eXective - Workload Partitions)



Virtualization Concerns

- Difficult to provide exact duplicate of underlying machine
- Specially those resources which involves in concurrent execution of VMMs, such as CPU & Main Memory
  - H/W features and support for VMM
- Other H/W such as I/O can be shared & scheduled among several VMs
- Two major virtualization concerns
  - Virtual CPU
  - Nested Page Tables
- These issues are not concerned with Type 0 Hypervisor virtualization

Virtualization Concerns

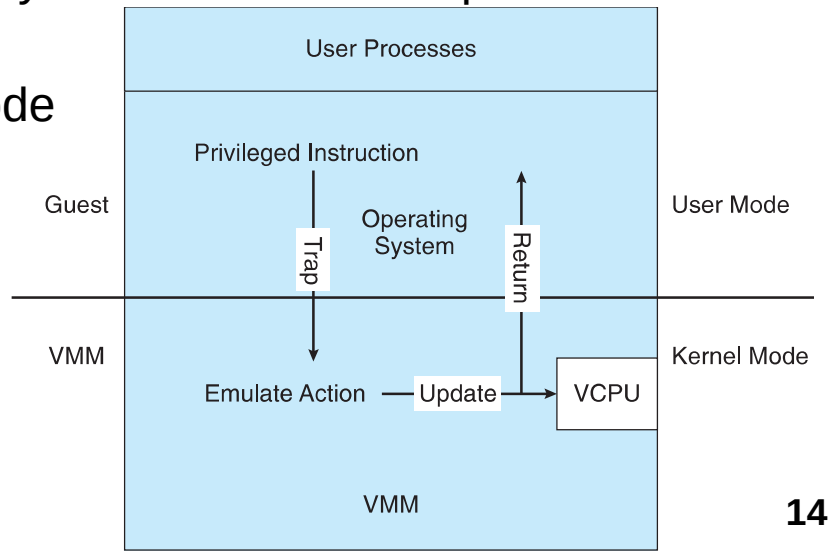
Virtual CPU

- VCPU is maintained by VMM which contains current CPU state of VM
- When CPU executes VM instruction, the data is fetched from VCPU
- Techniques
  1. Trap-and-Emulate
     - Designed for Dual mode CPU
       - Guest executes in user mode
       - Host kernel runs in kernel mode
     - Not feasible to run guest kernel run in host kernel mode
     - VM needs two modes – virtual user mode and virtual kernel mode
       - Both of which run in real user mode

Virtualization Concerns

Virtual CPU

- Guest OS executes system call towards virtual kernel which have to mapped into kernel mode
  - Attempting a privileged instruction in user mode causes an error (trap)
  - VMM gains control, analyzes error, executes operation as attempted by guest (emulate)
  - Returns control to guest in user mode
  - User level operations executes properly while kernel level operations executes much slower on Guest OS
  - Improved CPUs works on multiple mode which doesn't need VCPUs

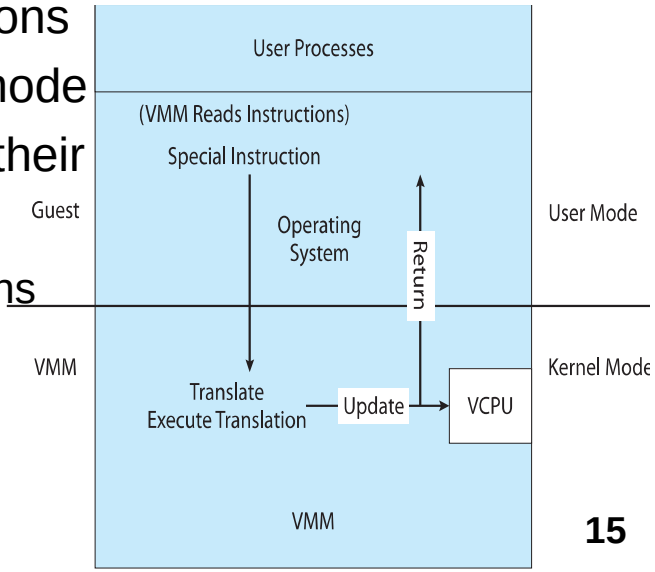


Virtualization Concerns

Virtual CPU

2. Binary Translation

- Machines don't have clean separation between privileged and non-privileged instructions
  - Earlier CPUs
- Some instructions results different in user mode and different in kernel mode (special instructions)
- Trap-and-emulate don't work with special instructions
- Non-privileged instructions are executed in user mode
- While, privileged instructions are detected before their execution via program counter of VMM
  - Special instructions are translated into new instructions that performs equivalent task
- Translation cache can increase performance



Virtualization Concerns

Nested Page Tables

- Memory management another challenge
- Guest OS manages its page tables virtually but Host OS is managing actually
- Mapping from virtual to real is concern
- Solution (for trap-and-emulate and binary translation) is nested page tables (NPTs)
  - Each guest maintains page tables to translate virtual to physical addresses
  - Physical address of guest is again mapped into host physical address
  - While, VMM maintains NPTs to represent guest's page-table state
  - VMM makes that guest's NPTs as active system page tables when Guest is active
  - When Guest requires a change page table, the VMM makes equivalent change to NPTs and its own page tables
  - Can result slower address translation

OS components with Virtualization

- OS is responsible to virtualize several components differently to VMs

CPU Scheduling

- Single-CPU systems act like multiprocessor ones when virtualized
  - One or more virtual CPUs per guest
- Generally VMM has one or more physical CPUs and number of threads to run on them
- Guests configured with certain number of VCPUs
- Multiple CPUs, each for VM
  - VMM can allocate dedicated CPUs, each guest manages its CPUs like host OS
- Not enough CPUs
  - CPU over-commitment
  - VMM can use standard scheduling algorithms to put threads on CPUs
  - Fairness aspect is considered

OS components with Virtualization

CPU Scheduling

- VMM itself needs CPU threads to perform its task
- Cycle stealing by VMM and oversubscription of CPUs results incorrect CPU cycles as Guests expect
  - A timesharing scheduler in a guest schedules 100ms time slices, each may take 100ms, 1 second, or longer
  - Each guest receive scheduled cycles and believes that it is having complete cycles
    - Poor response times for users of guest
    - Time-of-day clocks incorrect due to incomplete clock triggering cycles
  - Some VMMs provide application to run in each guest to fix time-of-day and provide other integration features

OS components with Virtualization

Memory Management

- Memory is required by guest, their applications & VMM itself
- Suffers from oversubscription
  - requires extra management efficiency from VMM
- Several methods of memory management
  1. Double-paging
     - Guest page table indicates a page in a physical frame but the VMM moves some of those pages to backing store
     - Less efficient so not preferable (VMM is unaware from access patterns)
  2. Pseudo-device driver
     - Each guest install the driver which looks like a device driver but only adds kernel-mode code to the guest kernel
     - Guest OS manages pages itself and Balloon memory manager communicates with VMM and to allocate or de-allocate memory
     - Pinned pages are not removed, if VMM/guest suffers from memory then VMM request guest OS to unpin some pages
  3. De-duplication by VMM determining the request of same page more than once then same page is mapped into multiple guests
     - Efficient if several guest are running same OS/applications

OS components with Virtualization

I/O Management

- VMMs integrate I/O with guests
  - I/O are segregated / flexible via device drivers
  - VMM can provide virtual devices and device drivers
- Guest manages I/O easily if are dedicated to guest
- I/O is complicated for VMMs when are shared
  - Ideal device drivers are used to create virtualization but are used to communicate with VMM
  - VMM can use different sharing mechanisms such as circular buffer
  - May require H/W support
- Networking facility should be provided to all guests by VMM
  - VMM can bridge guest to network (allowing direct access)
  - And / or uses network address translation (NAT)

OS components with Virtualization

Secondary Storage Management

- Boot disk and general data access should be provided by VMM
- Standard Disk Partitioning will be not sufficient to support several guests
- Type 0 Hypervisor uses disk manager to manage boot disks for several partitions
- Type 1 Hypervisor stores guest root disks and configuration information in file(s) provided by VMM
- Type 2 hypervisor & others stores same information in a file as disk image
  - Duplicate file results duplicate virtual machines
  - Move file to another system to move guests
- More disk space is required by VM, handled as I/O request
  - Physical-to-virtual (P-to-V) convert native disk blocks into VM format
  - Virtual-to-physical (V-to-P) convert from virtual format to native or disk format
  - VMM also needs to provide access to other disk images, disk partitions etc.

OS components with Virtualization

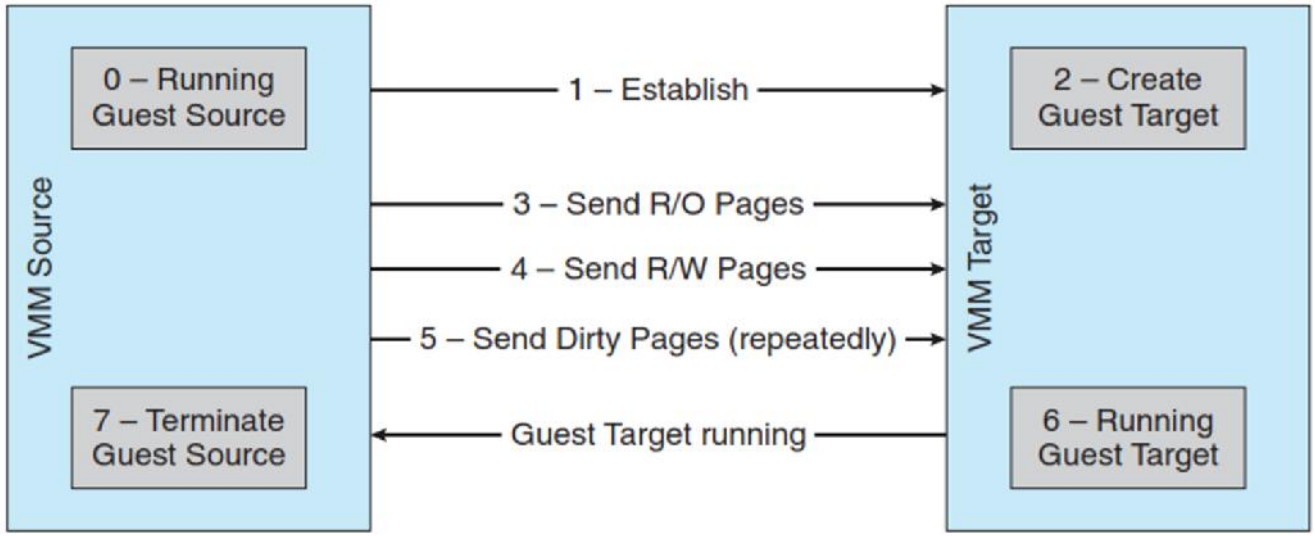
Live Migration

- VMM leads to functionality which is not provided by general OS
- Running guest can be moved between systems, without interrupting user access to the guest or its apps
- Steps:
  1. Source VMM establishes a connection with the target VMM
  2. Target creates new guest by creating new VCPU & other components
  3. Source sends all read-only guest memory pages to the target
  4. Source sends all read-write pages to the target, marking them as clean
  5. Source repeats step 4, as during that step some pages were probably modified by the guest (became dirty)
  6. When cycle of steps 4 and 5 becomes very short, source VMM freezes guest, sends guest state details, sends final dirty pages, and inform target VMM to start running the guest
     - Once target acknowledges that guest running, source terminates guest

OS components with Virtualization

Live Migration

- Very useful for resource management, maintenance etc.



- Limitations:
  - MAC address associated with Network
    - Modern switches understand and allow migration from MAC to MAC
  - Disk storage migration
    - Use of network storage devices

Questions