Introduction

- Have been around since 1960’s on mainframes
  - used for multitasking
  - Good example – VM/370

- Have resurfaced on commodity platforms
  - Server Consolidation
  - Web Hosting centers
  - High-Performance Compute Clusters
  - Managed desktop / thin-client
  - Software development / kernel hacking
Why do we care?

- Manageability
  - Ease maintenance, administration, provisioning, etc.

- Performance
  - Overhead of virtualization should be small

- Isolation
  - Activity of one VM should not impact other active VMs
  - Data of one VM is inaccessible by another

- Scalability
  - Minimize cost per VM
Virtual Machine Monitor (VMM)

- Resides as a layer below the operating system
- Presents a hardware interface to an OS
- Multiplexes resources between several virtual machines (VMs)
- Performance isolates VMs from each other
VMM Types

Type I VMM

| guest application | guest operating system | virtual-machine monitor (VMM) | host hardware |

Type II VMM

| guest application | guest application | guest operating system |
| guest application | virtual-machine monitor (VMM) | host operating system |
| guest application | host hardware | host hardware |
Virtualization Styles

- Fully virtualizing VMM
- Para- virtualizing VMM
VMM Classification

Type I

Fully-virtualized
VMware ESX
Para-virtualized
Xen

Type II

VMware Workstation
User Mode Linux
VMM Implementation

Should efficiently virtualize the hardware
- Provide illusion of multiple machines
- Retain control of the physical machine

Subsystems
- Processor Virtualization
- I/O virtualization
- Memory Virtualization
Processor Virtualization

Popek and Goldberg (1974)

- Sensitive instructions: only executed in kernel mode
- Privileged instructions: trap when run in user mode
- CPU architecture is virtualizable only if sensitive instructions are subset of privileged instructions

- When guest OS runs a sensitive instruction, must trap to VMM so it maintains control
x86 Processor Virtualization

- x86 architecture is not fully virtualizable
  - Certain privileged instructions behave differently when run in unprivileged mode
  - Certain unprivileged instructions can access privileged state

- Techniques to address inability to virtualize x86
  - Replace non-virtualizable instructions with easily virtualized ones statically (Paravirtualization)
  - Perform Binary Translation (Full Virtualization)
I/O Virtualization

- Issue: lots of I/O devices
- Problem: Writing device drivers for all I/O device in the VMM layer is not a feasible option
- Insight: Device driver already written for popular Operating Systems
- Solution: Present *virtual* I/O devices to *guest* VMs and channel I/O requests to a trusted *host* VM running popular OS
I/O Virtualization

VMM + Device Drivers

Physical Devices

VMM

Physical Devices
Memory Virtualization

- Traditional way is to have the VMM maintain a shadow of the VM’s page table
- The shadow page table controls which pages of machine memory are assigned to a given VM
- When guest OS updates its page table, VMM updates the shadow
VMware ESX Server

- Type I VMM - Runs on bare hardware
- Full-virtualized – Legacy OS can run unmodified on top of ESX server
- Fully controls hardware resources and provides good performance
Most user code executes in Direct Execution mode; near native performance

Uses *runtime* Binary Translation for x86 virtualization

- Privileged mode code is run under control of a Binary Translator, which emulates problematic instructions
- Fast compared to other binary translators as source and destination instruction sets are nearly identical
ESX Server – Memory Virtualization

- Maintains shadow page tables with virtual to machine address mappings.
- Shadow page tables are used by the physical processor.
- ESX maintains the pmap data structure for each VM with “physical” to machine address mappings.
- ESX can easily remap a machine page.
ESX Server – Memory Mgmt

- **Page reclamation – Ballooning technique**
  - Reclaims memory from other VMs when memory is overcommitted

- **Page sharing – Content based sharing**
  - Eliminates redundancy and saves memory pages when VMs use same operating system and applications
ESX Server - Ballooning

- Inflate balloon (+ pressure)
- Deflate balloon (- pressure)
- Guest OS manages memory implicitly
- May page out to virtual disk
- May page in from virtual disk
ESX Server – Page Sharing

Diagram showing page sharing among VMs with a hash page contents process leading to a specific hash value (2bd806af). The diagram includes a machine memory block and a hint frame with hash information: Hash: 06af, VM: 3, PPN: 43f8, MPN: 123b.
Real World Page Sharing

<table>
<thead>
<tr>
<th>Workload</th>
<th>Guest Types</th>
<th>Total</th>
<th>Saved</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate IT</td>
<td>10 Windows</td>
<td>2048</td>
<td>673</td>
<td>32.9</td>
</tr>
<tr>
<td>Nonprofit Org</td>
<td>9 Linux</td>
<td>1846</td>
<td>345</td>
<td>18.7</td>
</tr>
<tr>
<td>VMware</td>
<td>5 Linux</td>
<td>1658</td>
<td>120</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Corporate IT – database, web, development servers (Oracle, Websphere, IIS, Java, etc.)
Nonprofit Org – web, mail, anti-virus, other servers (Apache, Majordomo, MailArmor, etc.)
VMware – web proxy, mail, remote access (Squid, Postfix, RAV, ssh, etc.)
ESX Server – I/O Virtualization

- Has highly optimized storage subsystem for networking and storage devices
  - Directly integrated into the VMM
  - Uses device drivers from the Linux kernel to talk directly to the device
- Low performance devices are channeled to special “host” VM, which runs a full Linux OS
I/O Virtualization

VMM + Device Drivers

Physical Devices
VMware Workstation

- Type II VMM - Runs on host operating system
- Full-virtualized – Legacy OS can run unmodified on top of VMware Workstation
- Appears like a process to the Host OS
Workstation - Virtualization

- CPU Virtualization and Memory Virtualization
  - Uses Similar Techniques as the VMware ESX server

- I/O Virtualization
  - Workstation relies on the Host OS for satisfying I/O requests
  - I/O incurs huge overhead as it has to switch to the Host OS on every IN/OUT instruction.
Workstation – I/O Virtualization

- VMM must be able to intercept all I/O operations issued by the Guest OS.
- These are trapped by the VMM and emulated either in VMM or VMAppl.
- Any access that interact with physical hardware have to be handled by VMAppl.
- I/O intensive workload performs poorly due to extra host switches between the Host and the VMM worlds.
Workstation – Virtualize NIC
Xen

- Type I VMM
- Para-virtualized
- Open-source
- Designed to run about 100 virtual machines on a single machine
Xen – CPU Virtualization

- Privileged instructions are para-virtualized by requiring them to be validated and executed with Xen
- Processor Rings
  - Guest applications run in Ring 3
  - Guest OS runs in Ring 1
  - Xen runs in Ring 0
Xen – Memory Virtualization(1)

- Initial memory allocation is specified and memory is statically partitioned.
- A maximum allowable reservation is also specified.
- Balloon driver technique similar to ESX server used to reclaim pages.
Xen – Memory Virtualization(2)

- Guest OS is responsible for allocating and managing hardware page table
- Xen involvement is limited to ensure safety and isolation
- Xen exists in the top 64 MB section at the top of every address space to avoid TLB flushes when entering and leaving the VMM
Xen – I/O Virtualization

- Xen exposes a set of clean and simple device abstractions
- I/O data is transferred to and from each domain via Xen, using shared memory, asynchronous buffer descriptor rings
- Xen supports lightweight event delivery mechanism used for sending asynchronous notifications to domains
VMMs the only way to Virtualize?

- Alternative: Container-based OS (COS)
  - Eg., Solaris 10, Linux-Vserver, OpenVZ

<table>
<thead>
<tr>
<th>Features</th>
<th>VMM</th>
<th>COS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple kernels</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Administrative power (root)</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Manageability</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Scalability</td>
<td>✔</td>
<td>✔✔</td>
</tr>
<tr>
<td>Isolation</td>
<td>✔✔</td>
<td>✔</td>
</tr>
<tr>
<td>Efficiency</td>
<td>✔</td>
<td>✔✔</td>
</tr>
</tbody>
</table>
PlanetLab (circa 2005) Usage

- Typical Node (2.4GHz, 1GB, 100GB disk)
- ~250-300 configured VM file systems on disk
- 40-90 resident VMs with ≥ 1 process
- 5-20 active VMs using CPU
Container vs. Hypervisor Virtualization: What is the Trade-Off?

Container Design
## Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Hypervisor</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Kernels</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Load Arbitrary Modules</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Local Administration (root)</td>
<td>✓</td>
<td>✓ All</td>
</tr>
<tr>
<td>Live Migration</td>
<td>✓</td>
<td>✓ OpenVZ</td>
</tr>
<tr>
<td>Cross Version Migration</td>
<td>X</td>
<td>✓ Zap</td>
</tr>
</tbody>
</table>
Linux-VServer Overview

**Resource Control**
- Map Container to
  - HTB for Network
  - CFQ for Disk
- Logical Limits
  - Processes
  - Open FD

**Scheduler**
- Single Level
- Token Bucket Filter
  - preserves O(1) scheduler

**Context ID Filter**
- User IDs
- SHM & IPC address
- File system Barriers

**Optimizations**
- File-level Copy-on-write
COS vs. VMM Summary

- COS=Linux-Vserver VMM=Xen
- Performance
  - COS 1.25x – 2x more efficient than VMM
- Scalability
  - COS scales ~10x better
- Isolation
  - COS almost as good as VMM
Summary

- Classifying Virtual Machine Monitors
  - Type I vs. type II
  - Full vs. para-virtualization
- Processor virtualization
- Memory virtualization
- I/O virtualization
- Containers vs. VMM
Review Topics

- OS structure
- Process management
- CPU scheduling
- Virtual memory
- Disks and file systems
- General concepts
Operating System Structure

- Abstraction
- Protection and security
- Kernel structure
  - Layered
  - Monolithic
  - Micro-kernel
- Virtualization
  - Virtual machine monitor
Process Management

- Implementation
  - State, creation, dispatching, context switch
  - Threads and processes

- Synchronization
  - Race conditions and inconsistencies
  - Mutual exclusion and critical sections
  - Semaphores: P() and V()
    - Producer & Consumer problems
    - Scheduling problems
  - Semaphore implementations
  - Monitors and Condition Variables
  - Deadlock detection and prevention
CPU Scheduling

- Allocation -- Non-preemptible resources
- Scheduling -- Preemptible resources
  - FIFO
  - Round-robin
  - STCF
  - Lottery
Virtual Memory

- Mechanisms
  - Base and bounds
  - Paging
  - Segmentation
  - Page and segmentation
  - TLBs

- Page replacement
  - LRU and clock
  - Thrashing, working sets and WSClock
Disks and File Systems

- Disks
  - Disk behavior
  - Disk scheduling
  - RAID
  - Volume manager
- File access pattern and layout
- Directories and implementation
- File system performance
  - Layout for performance
  - Buffer cache
- File system reliability
  - Crash recovery and logging
- NFS and NetApp file system
- Deduplication file system
Major Concepts

- Locality
  - Spatial, temporal and working set
- Scheduling
  - Optimal algorithms know future, but we use past instead
- Layering
  - Synchronization, transactions, file systems, etc
- Caching
  - Translation look aside buffer, VM, buffer cache, etc
Operating System as Illusionist

Physical reality
◆ Single CPU
◆ Interrupts
◆ Limited memory
◆ No protection
◆ Raw storage device

Abstraction
◆ Infinite number of CPUs
◆ Cooperating sequential threads
◆ Unlimited virtual memory
◆ Each address has its own machine
◆ Organized and reliable storage system

Future courses
Networking: COS 461
Security: COS 429
Advanced OS: COS 518
Parallel Arch & Prog. COS 598A