CS 355
Operating Systems
Virtualization and the Cloud

Virtualization

• Creation of a virtual machine that acts like a real computer with an OS
• Software running on the virtual machine is separated from the underlying hardware
• Hardware is emulated
• Requires a Virtual Machine Monitor (Hypervisor)
• Allows a single computer to host multiple VMs

Virtualization Types

• Full virtualization
  – Almost complete emulation of hardware to allow software (such as a guest OS) to run unmodified
• Paravirtualization
  – Guest softwares are executed in their own isolated domains
  – Modification necessary

Advantages

• Isolation
• Save on physical hardware, power, space, etc
• Quick prototyping of new ideas
• Running legacy applications in their preferred environment
• Cross-platform software development (porting softwares)
• Multiple clients to share the same hardware

Requirements for Virtualization

1. Safety: hypervisor should have full control of virtualized resources.
2. Fidelity: behavior of a program on a virtual machine should be identical to same program running on bare hardware.
3. Efficiency: much of code in virtual machine should run without intervention by hypervisor.

Virtualization in a Nutshell

• Create a container in which to run the VM
• A guest OS will run in the container until it causes an exception and traps to the hypervisor
• The set of operations that trap is controlled by a hardware bitmap set by the hypervisor
• Hypervisor then emulates appropriate hardware response
Type 1 and Type 2 Hypervisors

<table>
<thead>
<tr>
<th>Virtualization method</th>
<th>Type 1 hypervisor</th>
<th>Type 2 hypervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypervisor with HW support</td>
<td>ESXi Server 5.5</td>
<td>VMware Workstation 1</td>
</tr>
<tr>
<td>Parallel virtualization</td>
<td>Xen 4.0</td>
<td>VMware Fusion, KVM, Parallels</td>
</tr>
<tr>
<td>Hypervisor with HW support</td>
<td>vSphere, Xen, Hyper-V</td>
<td>VMware Fusion, KVM, Parallels</td>
</tr>
<tr>
<td>Process virtualization</td>
<td>vSphere, Xen, Hyper-V</td>
<td>VMware Fusion, KVM, Parallels</td>
</tr>
</tbody>
</table>

Guest OS process

Host OS process

Guest OS (e.g., Windows)

Type 1 hypervisor

Type 2 hypervisor

Host OS (e.g., Linux)

Hardware (CPU, disk, network, interrupts, etc.)

Techniques for Efficient Virtualization

- When the OS in a VM executes a kernel-only instruction, it traps to the hypervisor

Virtual machine

User process

Virtual user mode

Virtual kernel mode

User mode

Hardware

Memory Virtualization

- Each guest OS believes it is king and therefore is at liberty to map any virtual page to any physical page
- What does a hypervisor do if two VMs want to map the same physical memory page?
- Shadow page tables – a mapping that maps the virtual pages used by the VMs to the physical pages the hypervisor selected

Virtualizing the Unvirtualizable

- Binary translation rewrites the guest operating system running in ring 1, while the hypervisor runs in ring 0

User process

Guest operating system

(Rewrite binary prior to execution + emulate)

Type 1 hypervisor

ring 0

Hardware

Lots of New Page Faults

- Guest-induced page faults
  - Normal guest OS page faults
- Hypervisor-induced page faults
  - the kind that are related to syncing the shadow page table and the guest OS’s
- Page faults are expensive and lead to VM exit

Shadow Page Tables

- Must be updated every time a guest OS updates its page table
- But how does the hypervisor know?
  - keep track of the page tables and mark them read-only so that any attempts to write to them will cause a fault that traps to the hypervisor
  - ignore it until the guest OS tries to actually access these new pages, which will cause a fault and trap
Nested/Extended Page Tables

- Handle the additional page-table manipulations in hardware without traps
- Hypervisor still mains additional page tables
- Hardware walks the all page tables to find these mappings
  - guest virtual address – guest physical address
  - guest physical address – host physical address

Reclaiming Memory

- Overcommitment
- Deduplication
- Ballooning

I/O Virtualization

- Guest OSes must share a disk
- Hypervisor creates a virtual disk (a file) for each VM
- When a guest OS tries to access disk, block numbers are converted into an offset into the appropriate disk
- Physical disk can differ from virtual disk (what the guest OS wants) via a VFS interface in the hypervisor

Clouds

National Institute of Standards and Technology defines characteristics of “cloud”
1. On-demand self-service
2. Broad network access
3. Resource pooling
4. Rapid elasticity
5. Measured service

Challenges in Virtualizing the x86

Major Challenges:
1. The x86 architecture was not virtualizable
2. The x86 architecture was of daunting complexity
3. x86 machines had diverse peripherals
4. Need for a simple user experience

Virtualizing the x86

- High-level components of the VMware virtual machine monitor (no hardware support)

- Diagram showing the decision algorithm for direct execution, binary translation, and shared modules (shadow MMU, I/O handling, …)
Virtualizing the x86

Binary translation must be used if any of the following is true:

– Virtual machine is currently running in kernel mode
– Virtual machine can disable interrupts and issue I/O instruction
– Virtual machine is currently running in real mode

Virtual Hardware Configurations

• Early VMware, ca. 2000

<table>
<thead>
<tr>
<th>Virtual Hardware (front end)</th>
<th>Back end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiprocessor 1 virtual x86 CPU, with the same instruction set extensions as the underlying hardware CPU</td>
<td>Scheduled by the host operating system on either a uniprocessor or multiprocessor host</td>
</tr>
<tr>
<td>Up to 512 MB of contiguous DRAM</td>
<td>Allocated and managed by the host OS (page by page)</td>
</tr>
</tbody>
</table>

Virtual Hardware Configurations

Role of the Host OS

• VMware hosted architecture and its three components: VMX, VMM driver and VMM

Role of the Host OS

• Difference between a normal context switch and a world switch