Types of system virtualization

- Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
- Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

Implementation techniques

- Paravirtualization (e.g., Xen) – static approach
  - OS is aware of virtualization
  - OS cooperates with VMM over resources (e.g., page tables)
  - Do not try to access resources, call VMM interface explicitly

- Detect & /uniFB01x interfaces – dynamic approach
  - Guarded resources (privilege): only VMM/hypervisor has access
  - Trap when guest OS tries to access resources
  - Hardware support or use dynamic binary translation
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
    - OS is aware of virtualization
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
    - OS is aware of virtualization
    - OS cooperates with VMM over resources (e.g., page tables)
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
    - OS is aware of virtualization
    - OS cooperates with VMM over resources (e.g., page tables)
    - Do not try to access resources, call VMM interface explicitly
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
    - OS is aware of virtualization
    - OS cooperates with VMM over resources (e.g., page tables)
    - Do not try to access resources, call VMM interface explicitly
  - Detect & fix interfaces – dynamic approach
Previous Lecture: System Virtualization

▶ Types of system virtualization
  ▶ Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  ▶ Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

▶ Implementation techniques
  ▶ Paravirtualization (e.g., Xen) – static approach
    ▶ OS is aware of virtualization
    ▶ OS cooperates with VMM over resources (e.g., page tables)
    ▶ Do not try to access resources, call VMM interface explicitly
  ▶ Detect & fix interfaces – dynamic approach
    ▶ Guarded resources (privilege): only VMM/hypervisor has access
Types of system virtualization
- Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
- Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

Implementation techniques
- Paravirtualization (e.g., Xen) – static approach
  - OS is aware of virtualization
  - OS cooperates with VMM over resources (e.g., page tables)
  - Do not try to access resources, call VMM interface explicitly
- Detect & fix interfaces – dynamic approach
  - Guarded resources (privilege): only VMM/hypervisor has access
  - Trap when guest OS tries to access resources
Previous Lecture: System Virtualization

- **Types of system virtualization**
  - Native (bare-metal) hypervisor virtualization (e.g., Oracle VM Server, VMware ESX)
  - Hosted virtualization (e.g., VMware player, VirtualBox, QEMU)

- **Implementation techniques**
  - Paravirtualization (e.g., Xen) – static approach
    - OS is aware of virtualization
    - OS cooperates with VMM over resources (e.g., page tables)
    - Do not try to access resources, call VMM interface explicitly
  - Detect & fix interfaces – dynamic approach
    - Guarded resources (privilege): only VMM/hypervisor has access
    - Trap when guest OS tries to access resources
    - Hardware support or use dynamic binary translation
Learning Objectives

What can we do to a VM?
Learning Objectives

What can we do to a VM?

- To understand the VM-handling mechanisms of a hypervisor
What can we do to a VM?

- To understand the VM-handling mechanisms of a hypervisor
- To understand how many different value-added services are constructed on top of VM-handling mechanisms
Starting a VM

Hypervisor

▶ gains control (e.g. clock tick)

Applications

Guest A
Operating System

Virtual Machine Monitor (VMM) / Hypervisor

Guest B
Operating System

Applications

Host Hardware

Virtualized
Starting a VM

Hypervisor
- gains control (e.g. clock tick)
- saves previous VM’s CPU registers
Starting a VM

Hypervisor
- gains control (e.g. clock tick)
- saves previous VM’s CPU registers
- loads next VM’s CPU registers

Applications
Guest A
Operating System
Virtual Machine Monitor (VMM) / Hypervisor
Guest B
Operating System
Host Hardware
Virtualized
Starting a VM

Hypervisor

- gains control (e.g. clock tick)
- saves previous VM’s CPU registers
- loads next VM’s CPU registers
- jumps to next VM’s next-PC (in correct privilege state)
Stopping a VM

- Save CPU registers into Hypervisor data area
- Hypervisor stops and starts VM all the time:
  - to share CPUs
  - to serialize access to resources
  - time multiplexing

Applications
Guest A
Operating System
Virtual Machine Monitor (VMM) / Hypervisor
Host Hardware

Applications
Guest B
Operating System
Virtualized
Stopping a VM

- Save CPU registers into Hypervisor data area

![Diagram of virtualization layers including Host Hardware, Virtual Machine Monitor (VMM) / Hypervisor, Guest A/B Operating Systems, and Applications. The diagram illustrates how virtualization can be used to share CPUs and serialize access to resources through time multiplexing.]
Stopping a VM

- Save CPU registers into Hypervisor data area
- Hypervisor stops and starts VM all the time:
  - to share CPUs
  - to serialize access to resources
Stopping a VM

- Save CPU registers into Hypervisor data area
- Hypervisor stops and starts VM all the time:
  - to share CPUs
  - to serialize access to resources
  - time multiplexing
VM State While Stopped

VM State

Applications
Guest A
Operating System

Applications
Guest B
Operating System

Virtual Machine Monitor (VMM) / Hypervisor

Host Hardware

Virtualized
VM State

- Memory (all guest physical memory)
  - Includes: Application state, OS state

Applications
Guest A
Operating System
Virtual Machine Monitor (VMM) / Hypervisor
Host Hardware

Virtualized
VM State

- Memory (all guest physical memory)
  - Includes: Application state, OS state
- CPU state (registers)

Virtualized

Applications

Guest A
Operating System

Guest B
Operating System

Virtual Machine Monitor (VMM) / Hypervisor

Host Hardware
VM State

- Memory (all guest physical memory)
  - Includes: Application state, OS state
- CPU state (registers)
- Small amount of I/O state
VM State

- Memory (all guest physical memory)
  - Includes: Application state, OS state
- CPU state (registers)
- Small amount of I/O state
  - Let’s stop VM when I/O is quiescent!

Virtualized
Once suspended, the VM image is self-contained

- Applications
- Guest Operating System
- CPU Registers
- I/O State
“Freeze” a VM

- Once suspended, the VM image is self-contained
  - VM can be (e.g.) copied to a file
“Freeze” a VM

- Once suspended, the VM image is self-contained
  - VM can be (e.g.) copied to a file
  - (LARGE file!)
“Freeze” a VM

- Once suspended, the VM image is self-contained
  - VM can be (e.g.) copied to a file
  - (LARGE file!)

What else can we do with this?
Move a VM

Applications

Guest A
Operating System

Virtual Machine Monitor

Host Hardware X

Virtualized

Freeze

Applications

Guest B
Operating System

CPU Registers

I/O State

Restart

Applications

Guest B
Operating System

CPU Registers

Virtual Machine Monitor (VMM) / Hypervisor

I/O State

Host Hardware Y

Virtualized

Copy

Applications

Guest C
Operating System

Antoniu Pop – Virtualization 3
Snapshot and Rollback a VM

- Applications
- Guest A
  - Operating System
- Virtual Machine Monitor (VMM) / Hypervisor
- Host Hardware

Virtualized

Why?
Can this process be optimized?

Antoniu Pop – Virtualization
Snapshot and Rollback a VM

Virtualized

Why?
Snapshot and Rollback a VM

Virtualized

Why?
Can this process be optimized?
Archive a VM
Rapid Provisioning

Applications
Guest Operating System
CPU Registers
I/O State

Applications
Guest Operating System
CPU Registers
I/O State

Applications
Guest Operating System
CPU Registers
I/O State

Virtual Machine Monitor (VMM) / Hypervisor

Host Hardware

Virtualized
Virtual Appliances

- http://www.vmware.com/appliances/directory/
- 1000+ downloadable appliances
- e.g., mail server, web server, hotel system, firewall, virus scanner, etc...

Diagram:

- Web Server
  - Guest Operating System
  - CPU Registers
  - I/O State

- Mail Server
  - Guest Operating System
  - CPU Registers
  - I/O State

- PABX
  - Guest Operating System
  - CPU Registers
  - I/O State

- ApplicationX
  - Guest Operating System
  - CPU Registers
  - I/O State
Virtual Appliances

- 1000+ downloadable appliances
- e.g., mail server, web server, hotel system, firewall, virus scanner, etc...

<table>
<thead>
<tr>
<th>Web Server</th>
<th>Mail Server</th>
<th>PABX</th>
<th>ApplicationX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Guest Operating System</strong></td>
<td><strong>Guest Operating System</strong></td>
<td><strong>Guest Operating System</strong></td>
<td><strong>Guest Operating System</strong></td>
</tr>
<tr>
<td>CPU Registers</td>
<td>CPU Registers</td>
<td>CPU Registers</td>
<td>CPU Registers</td>
</tr>
<tr>
<td>I/O State</td>
<td>I/O State</td>
<td>I/O State</td>
<td>I/O State</td>
</tr>
</tbody>
</table>

Deploying Secure Desktops

- Increased security and flexibility

---

**User X**
- Applications
- Guest Operating System
- Virtual Machine Monitor (VMM) / Hypervisor
- Host Hardware

**User Y**
- Applications
- Guest Operating System
- Virtual Machine Monitor (VMM) / Hypervisor
- Host Hardware

Virtualized Desktop Machine

Antoniu Pop – Virtualization 3

14 / 18
Deploying Secure Desktops

- Increased security and flexibility
- Better isolation between users

![Diagram showing a virtualized desktop machine with two users, User X and User Y, each with their applications, guest operating system, virtual machine monitor (VMM)/hypervisor, and host hardware.]
Deploying Secure Desktops

- Increased security and flexibility
  - Better isolation between users
  - Users can have “admin” privileges within their Guest OS

![Diagram of virtualized desktop machine with User X and User Y]
Deploying Secure Desktops

- Increased security and flexibility
  - Better isolation between users
  - Users can have “admin” privileges within their Guest OS

Is this common? Where?
Live Migration

Optimizing live migration from source to destination VMM
Live Migration

Optimizing live migration from source to destination VMM

▶ Copy every page from source to destination machine

▶ Reset dirty bit in VMM's page table for every page copied

▶ Repeat:
  ▶ Find next dirty page in source machine
  ▶ Copy to destination machine and reset dirty bit

▶ Until only minimal subset of pages left

▶ Suspend VM on source

▶ Copy remaining pages to destination

▶ Resume VM on destination
Live Migration

Optimizing live migration from source to destination VMM
▶ Copy every page from **source** to **destination** machine
  ▶ reset *dirty* bit in VMM’s page table for every page copied
▶ Repeat:
  ▶ Find next *dirty* page in source machine
  ▶ Copy to destination machine and reset *dirty* bit
▶ Until only minimal subset of pages left
▶ Suspend VM on source
▶ Copy remaining pages to destination
▶ Resume VM on destination
Live Migration

Optimizing live migration from source to destination VMM

▶ Copy every page from source to destination machine
  ▶ reset dirty bit in VMM’s page table for every page copied

▶ Repeat:
  ▶ Find next dirty page in source machine
  ▶ Copy to destination machine and reset dirty bit
Live Migration

Optimizing live migration from source to destination VMM

► Copy every page from **source** to **destination** machine
  ► reset *dirty* bit in VMM’s page table for every page copied

► Repeat:
  ► Find next *dirty* page in **source** machine
  ► Copy to **destination** machine and reset *dirty* bit

► Until only minimal subset of pages left
Live Migration

Optimizing live migration from source to destination VMM

- Copy every page from source to destination machine
  - reset dirty bit in VMM’s page table for every page copied
- Repeat:
  - Find next dirty page in source machine
  - Copy to destination machine and reset dirty bit
- Until only minimal subset of pages left

- Suspend VM on source
- Copy remaining pages to destination
- Resume VM on destination
Load Balancing

- Management software monitors load on all physical machines
Load Balancing

- Management software monitors *load* on all physical machines
- If loads are mismatched, migrate a VM from a loaded to a less-loaded machine
Management software monitors load on all physical machines.
If loads are mismatched, migrate a VM from a loaded to a less-loaded machine.

Independent of Application!
Load Balancing

- Management software monitors *load* on all physical machines
- If loads are mismatched, migrate a VM from a loaded to a less-loaded machine
- Independent of Application!
- Independent of Operating System!
High Availability

- For critical applications, keep a standby VM available on a different hardware system
High Availability

- For critical applications, keep a standby VM available on a different hardware system
- Regularly copy active VM image to standby VM (but don’t activate it)
High Availability

- For critical applications, keep a standby VM available on a different hardware system
- Regularly copy active VM image to standby VM (but don’t activate it)
- Activate standby VM if active VM stops responding (VM crashes? VMM crashes? Hardware system fails?)
High Availability

- For critical applications, keep a standby VM available on a different hardware system
- Regularly copy active VM image to standby VM (but don’t activate it)
- Activate standby VM if active VM stops responding (VM crashes? VMM crashes? Hardware system fails?)
- Independent of Application!
High Availability

- For critical applications, keep a standby VM available on a different hardware system
- Regularly copy active VM image to standby VM (but don’t activate it)
- Activate standby VM if active VM stops responding (VM crashes? VMM crashes? Hardware system fails?)

- Independent of Application!
- Independent of Operating System!
Goals of System Virtualization

- Multiple OS running on the same hardware
Goals of System Virtualization

- Multiple OS running on the same hardware
- Pre-configured virtual machines
Goals of System Virtualization

- Multiple OS running on the same hardware
- Pre-configured virtual machines
- Load balancing
Goals of System Virtualization

- Multiple OS running on the same hardware
- Pre-configured virtual machines
- Load balancing
- High availability