Overview of System Virtualization: The most powerful platform for program analysis and system security

Zhiqiang Lin
Department of Computer Science
University of Texas at Dallas

April 4th, 2012
Outline

1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
Outline

1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
Why virtualization?

Today’s data center

Cloud computing

Fault resist
Cost Effect
Easy to maintain
What is Virtualization

Virtual Machine Monitor

Hardware

Linux
Linux (devel)
XP
Vista
MacOS
Virtualization Properties

- Isolation
- Encapsulation
- Interposition
Isolation

- Fault Isolation
  - Fundamental property of virtualization
- Performance Isolation
  - Accomplished through scheduling and resource allocation
Encapsulation

- All VM state can be captured into a file
  - Operate on VM by operating on file
  - `mv, cp, rm`

- Complexity
  - Proportional to virtual HW model
  - Independent of guest software configuration
Interposition

- All guest actions go through monitor
- Monitor can inspect, modify, deny operations
- Example:
  - Compression
  - Encryption
  - Profiling
  - Translation
Why Not the OS?

- It's about interfaces
  - VMMs operate at the hardware interface
  - Hardware interfaces are typically smaller, better defined than software interfaces
- Disadvantages of being in the monitor
  - Low visibility into what the guest is doing
Virtualization benefits

- Increased resource utilization:
  - Server consolidation
  - Multiplexing
- Mobility
- Enhanced Security
- Test and Deployment
- ...
Virtualization: server consolidation

Physical Servers

Virtual Server Host

Virtual Machines Guests
Mobility: load balance (Migration)

Server 1
CPU Utilization = 90%

Server 2
CPU Utilization = 50%

VM Guest 1
VM Guest 2
VM Guest 3
VM Guest 4
VM Guest 5
None
VM Guest 6
VM Guest 7
Mobility: load balance (Migration)

Server 1
CPU Utilization = 70%

Server 2
CPU Utilization = 70%

VM Guest 1
VM Guest 3
VM Guest 4
VM Guest 5
VM Guest 2
VM Guest 6
VM Guest 7
None
Enhanced Security

Conventional VM

Secure VM

Firewall

VMM
Testing and Deployment

- Develop VM
- Development
- QA VM
- Test
- Production VM
- Deploy
Outline

1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
Types of Virtualization

**Process Virtualization**

- Language construction
  - Java, .NET
- Cross-ISA emulation
  - Apple’s 68000-PowerPC to Intel X86 Transition
- Application virtualization
  - Sandboxing, mobility
## Types of Virtualization

### Process Virtualization
- Language construction
  - Java, .NET
- Cross-ISA emulation
  - Apple’s 68000-PowerPC to Intel X86 Transition
- Application virtualization
  - Sandboxing, mobility

### System Virtualization
- VMware
- Xen
- Microsoft’s Viridian
Taxonomy

**Process VMs**
- Same ISA
  - Multiprogrammed Systems
  - Dynamic Binary Optimizers
- Different ISA
  - Dynamic Translators
  - HLL VMs

**System VMs**
- Same ISA
  - Classic-System VMs
  - Hosted VMs
- Different ISA
  - Whole-System VMs
  - Co-designed VMs
## System Virtual Machine Monitor Architectures

<table>
<thead>
<tr>
<th>Virtualization Types</th>
<th>Key Techniques</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hosted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMware Workstation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMware ESX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypervisor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Traditional

Examples: IBM VM/370, Stanford DISCO
### Hosted Virtual Machines

**Goal:**
- Run Virtual Machines as an application on an existing Operating System (QEMU, VirtualPC, VirtualBox, VMware-workstation)

**Why**
- Application continuity
- Reuse existing device drivers
- Leverage OS support
  - File system
  - CPU Scheduler
- VM management platform
Hosted Monitor Architecture (VirtualBox)
Hosted Monitor Architecture (VirtualBox)

- **Host OS (Window XP)**
- **Guest OS (Linux)**
- **User App**
- **Kernel Module**
- **Virtual Machine Monitor**
- **CPU / Memory Virtualization**

**Hardware**
Hosted Monitor Architecture (VirtualBox)

- **Host OS**: Window XP
- **Guest OS**: Linux
- **Kernel Module**
- **Device I/O**
  - Network, Disk, Display, Keyboard, Timer, USB

Diagram shows the relationships between the host OS, guest OS, user app, kernel module, and device I/O.
Hosted Monitor Architecture (VirtualBox)

- Host OS (Window XP)
- Hosted Monitor Architecture
- Hardware
- Virtual Machine Monitor
- Guest OS (Linux)
- User App
- Kernel Module
- Interrupts
Hosted Monitor Scheduling (VirtualBox)

- **Host OS (Window XP)**
- **User App**
- **Kernel Module**
- **Virtual Machine Monitor**

- **Guest OS (Linux)**

- **Hosted Monitor Scheduling**
- **Hardware**

- **Guest OS (Vista)**
- **Virtual Machine Monitor**
Hosted Monitor Scheduling (VirtualBox)

- Host OS (Window XP)
- Guest OS (Linux)
- Virtual Machine Monitor
- Guest OS (Vista)

1. CPU Scheduler
2. Kernel Module
3. Virtual Machine Monitor

Hardware
Hosted Monitor Scheduling (VirtualBox)

- Host OS (Windows XP)
- Guest OS (Linux)
- Virtual Machine Monitor
- User App
- CPU Scheduler

- Host OS (Windows Vista)
- Guest OS (Vista)
- Virtual Machine Monitor
Hosted Monitor Scheduling (VirtualBox)

Host OS (Window XP)

User App

User App

CPU Scheduler

Kernel Module

Guest OS (Linux)

Virtual Machine Monitor

Guest OS (Vista)

Virtual Machine Monitor

Hardware
Hosted Architecture Tradeoffs

Positives

- Installs like an application
  - No disk partitioning needed
  - Virtual disk is a file on host file system
  - No host reboot needed

- Runs like an application
  - Uses host schedulers

Negatives

- I/O path is slow
- Requires world switch
- Relies on host scheduling
- May not be suitable for intensive VM workloads
Hosted Architecture Tradeoffs

Positives
- Installs like an application
  - No disk partitioning needed
  - Virtual disk is a file on host file system
  - No host reboot needed
- Runs like an application
  - Uses host schedulers

Negatives
- I/O path is slow
  - Requires world switch
- Relies on host scheduling
  - May not be suitable for intensive VM workloads
VMware ESX 2.0

Figure 1: ESX Server architecture

Hybrid Ex 2 - Xen 3.0

- Para-virtualization
- Linux Guest
- Hardware-supported virtualization
- Unmodified Windows
- Isolated Device Drivers

Source: Ottawa Linux Symposium 2006 presentation.
http://www.cl.cam.ac.uk/netos/papers/
Hypervisor

- Hardware-supported single-use monitor

**Characteristics**
- Small size
- Runs in a special hardware mode
- Guest OS runs in normal privileged level

**Uses**
- Security
- System management
- Fault tolerance
1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
Key Techniques to implement Virtualization

- Instruction Interpretation
- Binary Translation
- Trap-and-emulate
- Para-virtualization (hardware virtualization)
Instruction Interpretation

- Emulate Fetch/Decode/Execute pipeline in software
- Positives
  - Easy to implement
  - Minimal complexity
- Negatives
  - Slow!

Bochs: http://bochs.sourceforge.net/
Example: CPUSState

```c
static struct {
    uint32   GPR[16];
    uint32   LR;
    uint32   PC;
    int      IE;
    int      IRQ;
} CPUSState;

void CPU_CLI(void) {
    CPUSState.IE = 0;
}

void CPU_STI(void) {
    CPUSState.IE = 1;
}
```
Example: Virtualizing the Interrupt Flag w/ Instruction Interpreter

```c
void CPU_Run(void)
{
    while (1) {
        inst = Fetch(CPUState.PC);
        CPUState.PC += 4;
        switch (inst) {
            case ADD:
                CPUState.GPR[rd] = GPR[rn] + GPR[rm];
                break;
            ...
            case CLI:
                CPU_CLI();
                break;
            case STI:
                CPU_STI();
                break;
        }
        if (CPUState.IRQ && CPUState.IE) {
            CPUState.IE = 0;
            CPU_Vector(EXC_INT);
        }
    }
}

void CPU_CLI(void)
{
    CPUState.IE = 0;
}

void CPU_STI(void)
{
    CPUState.IE = 1;
}

void CPU_Vector(int exc)
{
    CPUState.LR = CPUState.PC;
    CPUState.PC = disTab[exc];
}
```
II. Binary Translator

- Guest Code
- Translator
- Translation Cache
- Callouts
- CPU Emulation Routines
- TC Index

Diagram: Flowchart showing the relationship between the components.
Basic Blocks

Guest Code

- mov ebx, eax
- cli
- and ebx, ~0xfff
- mov ebx, cr3
- sti
- ret

vPC

Straight-line code

Basic Block

Control flow
Binary Translator

<table>
<thead>
<tr>
<th>Guest Code</th>
<th>Translation Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov ebx, eax</td>
<td>mov ebx, eax</td>
</tr>
<tr>
<td>cli</td>
<td>call HANDLE_CLI</td>
</tr>
<tr>
<td>and ebx, ~0xffff</td>
<td>and ebx, ~0xffff</td>
</tr>
<tr>
<td>mov ebx, cr3</td>
<td>mov [CO_ARG], ebx</td>
</tr>
<tr>
<td>sti</td>
<td>call HANDLE_CR3</td>
</tr>
<tr>
<td>ret</td>
<td>call HANDLE_STI</td>
</tr>
<tr>
<td></td>
<td>jmp HANDLE_RET</td>
</tr>
</tbody>
</table>

vPC

start
III. Trap and Emulate

Guest OS + Applications

- Page Fault
- Undef Instr
- vIRQ
- MMU Emulation
- CPU Emulation
- I/O Emulation

Virtual Machine Monitor

Unprivileged

Privileged
Software VMM

Direct Exec (user)

Faults, syscalls, interrupts

VMM

IRET, sysret

Guest Kernel Execution

Translated Code (guest kernel)

Traces, faults, interrupts, I/O
IV. Hardware virtualization, Para-virtualization (e.g., Xen)

- Modify operating systems to let OS to cooperate with VMM
  - Let OS runs in Ring1 and user apps run in Ring3
  - Change sensitive instructions to explicit calls to the VMM
  - No need to trap and emulate
Traditional Address Spaces

- **Background Process**
- **Operating System**

**Virtual Address Space**
- Current Process
- Operating System

**Physical Address Space**
- RAM
- Frame Buffer
- Devices
- ROM
Shadow Page Tables

- Virtual CR3
- Real CR3

Diagram:
- Guest Page Table
- Shadow Page Table
- Virtual CR3
- Real CR3
Guest Write to CR3

Virtual CR3

Real CR3

Guest Page Table

Shadow Page Table

Guest Page Table

Shadow Page Table

Guest Page Table

Shadow Page Table
Guest Write to CR3

- Virtual CR3
  - Guest Page Table
  - Shadow Page Table

- Real CR3
  - Guest Page Table
  - Shadow Page Table
Outline

1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
Virtualization (or virtualisation), in computing, is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system, storage device, or network resources.

Impacts:
- Cloud computing
- Data centers
- Provides greater opportunities to security
References

- Xen and the art of virtualization
- Memory resource management for VMware ESX Server
- http://labs.vmware.com/courseware (great virtualization resources)
- CPU Virtualization- Basic Virtualization
  http://labs.vmware.com/download/76/
- Memory Virtualization
  http://labs.vmware.com/download/77/