Virtual Machines: Concepts & Applications

Lecture 9: System VMs – I

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Disclaimer: Many slides of this lecture are based on the slides of authors of the textbook from Elsevier. All copyrights reserved.
System VMs

- Support multiple guest OSes on single hardware platform; all running the same ISA
System VMs

- Support multiple guest OSes on single hardware platform; all running the same ISA
Applications (partial list)

• Simultaneous support for multiple OSes/Apps
  – Easy way to implement multiprogramming without requiring complete multiprogramming OS.
• Legacy applications
• Simultaneous support for different OSes/Apps
  – E.g. Windows and Unix
• Error containment
  – sandboxing
  – If a VM crashes, the other VMs can continue to work
    Assumes VMM is correct (smaller/simpler)
• Operating System debugging
  – Can proceed while system is being used for normal work
Applications, contd.

- **Operating System Migration**
  - Can proceed while “old” OS continues to be used
Applications, contd.

• Retrofitting new features
  – Have VMM transform new device into a virtual device
• Support for multiple networked machines on one physical machine
  – Allows debug of network software
• Event monitoring
  – traces of execution
  – replay
• Education
System VMs

- **Virtual Machine Monitor (VMM)** manages real hardware resources
- **All Guest systems must be given** logical hardware resources
- All resources are **virtualized**
  - By partitioning real resources
  - By sharing real resources
- **Guest state must be managed**
  - By using indirection
  - By copying
State Management: Indirection

- Hold guest state in VMM memory
- Change pointer on guest switch
- Example: registers
State Management: Copying

- Hold guest state in VMM Memory
- Copy state on guest switch
System VMs: Processor Mgmt/Protection

- VMM runs in **system mode**
  - VMM manages/protects processor through conventional mechanisms
- **Guest OSes** run in **user mode**
  - Guest OSes do not have direct control over hardware resources
    - All attempts to interact w/ hardware resources are intercepted by VMM
- VMM manages shadow copies of Guest System state (incl. control registers)
- VMM schedules and runs Guest Systems
VM Timesharing

- VMM Timeshares resources among guests
  - Similar to OS timesharing applications

- Guest OS must not be allowed to set timer interrupt
- Guest OS must not know the real timer value set by VMM
- VMM can provide guest OS with emulated virtual interval timer
Native and Hosted VMs

- **Applications**
  - OS
  - Hardware

- **Virtual Machine**
  - VMM
  - Hardware

- **Virtual Machine**
  - VMM
  - Host OS
  - Hardware

- **Virtual Machine**
  - VMM
  - Host OS
  - Hardware

- **Traditional uniprocessor system**
- **Native VM system**
- **User-mode Hosted VM system**
- **Dual-mode Hosted VM system**
Virtualizing the processor

Execution of the guest instructions

- Emulation
- Direct native execution
Privileged Instructions

• Trap if executed in user mode; not in supervisor mode
• Privileged instructions are required to trap
  – No-op in user mode is not enough
Control Sensitive instructions:

All instructions that attempt to change the configuration of resources
- e.g. page table in general, timer
Behavior Sensitive instructions:

All instructions whose behavior or results depend on the configuration

• Examples:
  – Load physical address
  – POPF (Intel x86)
Instruction Types -- Summary

**Innocuous Instructions**: Those that are not control or behavior sensitive
VMM components

These instructions desire to change machine resources, e.g. Load Relocation Bounds Register

These instructions do not change machine resources, but access privileged resources, e.g. IN, OUT, Write TLB

Interpreter Routine 1

Interpreter Routine 2

Interpreter Routine n
VMM components

- **Dispatcher**
  - Top level control module for VMM
  - Decides which of other components to call

- **Allocator**
  - Decides which system resources should be provided and to manage shared resources among VMs

- **Interpreters**
  - One per privileged instructions
  - Emulate the effects of privileged instructions when operating on virtual resources

- **VMM runs in supervisor mode; all other software in user mode**
Privileged Instruction Handling

LPSW: Load Program Status Word
Includes Mode Bit and PC (among other things)

**Guest OS code in VM**
*(user mode)*

- Privileged instruction (LPSW)
- ...
- ...
- ...
- Next instruction (target of LPSW)

**VMM code**
*(privileged mode)*

- Dispatcher
- 
- 
- LPSW Routine:
  - Change mode to privileged
  - Check privilege level in VM
  - Emulate instruction
  - Compute target
  - Restore mode to user
  - Jump to target

Diagram showing the flow of execution between the guest OS code in VM (user mode) and the VMM code (privileged mode) when a privileged instruction (LPSW) is encountered.

Dispatcher

LPSW Routine:
- Change mode to privileged
- Check privilege level in VM
- Emulate instruction
- Compute target
- Restore mode to user
- Jump to target
Virtual Machine “requirements”

1. **Efficiency**: All innocuous instructions are executed by the hardware directly

2. **Resource control**: The allocator must be invoked when any program attempts to affect system resources

3. **Equivalence**: Any program executes exactly as on real hardware except
   - Performance
   - Availability of system resources

• VMM must satisfy all three requirements

• Alternative definition: VMM satisfies 2 and 3. Efficient VMM satisfies also 1.
Virtual Machines: Main Theorem

A virtual machine monitor can be constructed if the set of sensitive instructions is a subset of the set of privileged instructions.

Proof shows:

- Equivalence by interpreting privileged instructions and executing remaining instructions natively.
- Resource control by having all instructions that change resources trap to the VMM.
- Efficiency by executing all non-privileged instructions directly on hardware.

A key aspect of the theorem is that it is easy to check.
Virtual Machines: Main Theorem

A virtual machine monitor can be constructed if the set of sensitive instructions is a subset of the set of privileged instructions.
Recursive Virtualization

- Virtual Machine
- Virtual Machine
- Virtual Machine
- Virtual Machine
- 2nd level VMM
- VMM
- Hardware

Privileged Mode
Recursive Virtualization

Running a VMM as a VM on a VM on a VM....

**Theorem:** A conventional third generation computer is recursively virtualizable if it is (a) virtualizable, and (b) a VMM without any timing dependences can be constructed for it.

**Proof** - A VMM is a program and from the VM theorem will be “identically performing” except for timing dependences and resource constraints.
Timing is excluded in the theorem; Resource constraints only limit the depth of recursion.
Conclusions

• System VMs must control all resources
• There are two types of resources:
  – replicated (keyboard, ...)
  – shared (processor, memory, storage, and some I/O)