Introduction to Virtual Machines

Michael Jantz
Acknowledgements

- Slides adapted from Chapter 1 in *Virtual Machines: Versatile Platforms for Systems and Processes* by James E. Smith and Ravi Nair
- Credit to Prasad A. Kulkarni – some slides were borrowed from his course on *Virtual Machines* at the University of Kansas
Introduction to Virtual Machines

• Abstraction and interfaces
• Virtualization
• Computer system architecture
• Process virtual machines
• System virtual machines
Abstraction

- Mechanism to manage complexity in computer systems.
- Partitions design of a system into levels.
- Allows higher levels to ignore the implementation details of lower levels.
Interfaces

• Define the communication b/w two entities
  • Hierarchical relationship
  • Linear relationship

• Software can run on any machine supporting a compatible interface
Interfaces: Advantages

• Allows de-coupling of design tasks
  • Work on different components can progress independently

• Helps manage system complexity
  • Each component provides an abstraction of itself to the outside world
Interfaces: Disadvantages

- Limit flexibility
  - Developers must work within the constraints of the interface
- Can be confining
  - ARM binaries on x86 machine?
  - Windows applications on Linux?
Interfaces: Disadvantages

• Inhibits innovation
  • Hard to change instruction sets

• Application software cannot directly exploit microprocessor features
  • Software is supposed to be architecture independent!
Virtualization

• Map the interface of one system to a real system that actually implements it

• Removes constraints imposed by interfaces
  • Improves availability of application software
  • Removes the assumption of a single regime, improves security and failure isolation

• Provide a *different* view to a particular computer resource
  • Not necessarily a *simpler* view
Virtualization

• Creates an isomorphism that maps a virtual *guest* to a real *host*
Virtualization vs. Abstraction

• Virtualization does not necessarily hide details
Virtual Machines

• Virtualization applied to the entire machine
• Adds a layer of software to a real machine to support the desired architecture
• Process of virtualization
  • Maps virtual resources or state to real resources
  • Uses real machine instructions to carry out actions specified by the VM instructions
Benefits of VM's

- Flexibility
- Portability
- Isolation
- Security
Computer Architecture

• Architecture: functionality and appearance of a computer system
• Implementation: embodiment of the architecture
Computer Architecture

- Computer systems consist of:
  - Layers of abstraction
  - Well-defined interfaces
The ISA Interface

- Interface between hardware & software
- Two parts
  - System ISA
  - User ISA
The ISA Interface

• System ISA
  • Important for OS developers
The ABI Interface

- Application Binary Interface (ABI)
  - User ISA + system calls
  - Important for compiler writers
The API Interface

- Application Programming Interface (API)
  - User ISA + library calls
  - Important for application programmers
Major Program Interfaces

• ISA – supports all conventional software

• ABI – supports application software only
Process Virtual Machines

- Supports an individual *process*
  - Run SW for a different OS and different ISA
  - Couple at ABI level via *runtime system*
Process Virtual Machines

- Guest processes intermingle with host processes
- Binaries encapsulated by the runtime
- PVM does not include OS
- Examples:
  - Java
  - IA-32 EL
  - Dynamic optimizers (dynamo)
PVM: Multiprogramming

• OS provides PVMs for each application
• System calls + user ISA == PVM to execute multiple, concurrent processes
• Each process is given the illusion of having the entire machine to itself
PVM: Emulators

• Execute binaries compiled for one ISA on a machine with a different ISA

• Emulation methods
  • Interpretation
    • Fast startup, but slow steady-state
  • Dynamic binary translation
    • High startup overhead, faster steady-state
    • Uses a code cache to store translated blocks

• Examples: Java, IA32-EL
PVM: Binary Optimizers

- Same source and target ISAs
- Main task is optimization
  - ABI level optimization
  - May also collect performance profiles
  - May also enhance security
- e.g., HP's Dynamo
PVM: High-Level Language VMs

- Designed for VM execution
- Aim to minimize HW- and OS-specific features
PVM: High-Level Language VMs

• Distribution format is binary class files
  • Virtualized ISA (no real implementation)
• OS interaction via API
• Examples: Java, Microsoft CLI
System Virtual Machines

- Supports an OS with many user-level processes
  - Couple at the ISA level
  - Examples: VMWare, Transmeta Crusoe
Classic System Virtual Machine

- Original meaning of the term *virtual machine*
  - All guest and host software use the same ISA
  - VMM runs on bare hardware (privileged mode)
  - VMM intercepts all privileged operations for the guest OS's

![Diagram of Classic System Virtual Machine]

- Win process
- Win process
- Win process
- Linux process
- Linux process
- Linux process

- Guest OS (Windows)
- Guest OS2 (Linux)

- VMM

- HOST PLATFORM

- virtual network communication
Hosted VM

• VMM built on top of existing OS
• Advantages
  • Installs just like a user-level application
  • Host OS provides driver support
• Drawbacks
  • Less efficient
Whole System VMs

- Host and guest do not use a common ISA
  - Both app and OS code require emulation
- Typically implemented as a hosted VMM
- Example: VirtualPC
Co-designed Virtual Machines

• Designed to enable innovative ISA's and/or hardware implementations

• As if the VM software is part of the HW
  • Applications / OS never directly execute native ISA instructions

• Useful for backwards compatibility
  • Example: Transmeta Crusoe
VM Taxonomy

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

System VMs
- Same ISA
  - Classic-System VMs
  - Hosted VMs
- Different ISA
  - Whole-System VMs
  - Codesigned VMs