Opening: Virtual Machines

Outline

- Historical perspective
- Motivations and examples
- Software support
- Hardware support

Historical Perspective

IBM VM

- Starting point 1967: IBM VM System/370
- Offers long-term portability of System/360 applications over a wide range of machines and peripherals, although the processor's machine instructions where quite different

Implementation Challenges

How Does it Work?

- Main requirement: intercept any *guest*-specific action to redirect it to the *host*'s interface
  - Machine language instructions
  - Memory-mapped control registers
  - Exceptions and interrupts
- Good example: IBM System/370
- Bad example: x86, where many exceptions, port I/O, accesses to system buses and memory-mapped PC motherboard registers *were* impossible to "trap" (be reconfigured to trigger exceptions selectively)... until Core Duo 2 processor

References

- First Attempt to Formalize and Classify: Popek and Goldberg 1974
- James E. Smith and Ravi Nair: *Virtual Machines: Versatile Platforms for Systems and Processes*, 2005

Implementing a Virtual Machine Monitor

Software

- Most general case, when native virtualization is not possible, or to provide *sandboxing*
- Emulation and full-system simulation: e.g., *QEMU*, *VMware*
- Binary translation: e.g., Dynamo (HPLabs), Transmeta Code Morphing, Rosetta (Apple), IA32EL (Intel)
- Code caching
- Dynamic optimization (similar to just-in-time compilation of Java bytecode)
- Paravirtualization: e.g., *Xen* or *User Mode Linux*

Hardware

- Native virtualization (lightweight hypervisor): "traps" selectively on some instructions or address ranges, e.g., *Parallels*
- Reduce exception overhead and cost of switching to kernel mode
- Support to accelerate instruction decoding (PowerPC)

Sophisticated Example

Virtualization Stack

Java application

↓ JVM just-in-time compilation

Linux x86

↓ VMWare full system emulation

Windows x86

↓ Code Morphing

Transmeta Crusoe/Efficeon VLIW processor

Thank You!