02 Virtualization

EN0746 Computer Network Implementation

David Kendall

Northumbria University

What is virtualization?

virtual, adj. (and n.)
Computing: *Not physically existing as such but made by software to appear to do so from the point of view of the program or the user* (OED)

Virtual systems

- Abstract physical components using logical objects
- Dynamically bind logical objects to physical configurations

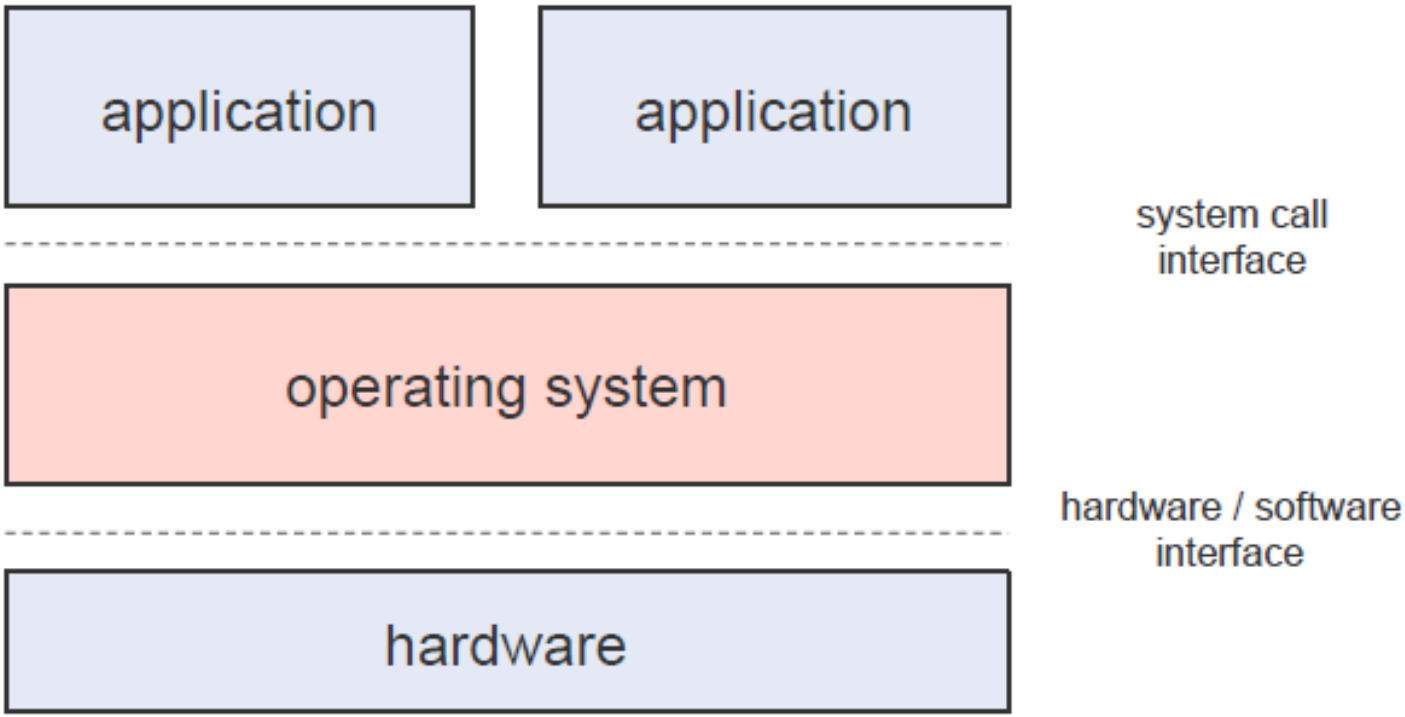
Examples

- Network – Virtual LAN (VLAN), Virtual Private Network (VPN)
- Storage – Storage Area Network (SAN)
- Computer – Virtual Machine (VM), simulator

Outline

- Virtual machines
- Virtualization approaches
- Applications of virtualization

Starting point: a traditional machine



Hardware

A collection of devices:

- CPU
  - defines the instruction set of the machine
  - provides registers, processes instructions, handles interrupts
  - defines privilege modes (e.g., supervisor, user)
- Memory hierarchy
  - physical memory words accessible via load/store instructions
  - Memory Management Unit (MMU) provides paging / segmentation, and therefore virtual memory support
  - MMU controlled via special registers and page tables
- I/O devices
  - disks, NICs, etc., controlled by programmed I/O (inb, outb) or by DMA (load/store)
  - events delivered to software via polling or interrupts
- Other devices
  - graphics cards, clocks, USB controllers, etc.

Operating system

Just a program!

- you write it in some language (C/C++), and compile it into a program image
- it runs like any other program, but in a privileged (supervisor) CPU mode
  - this allows it to interact with hardware devices using “sensitive” instructions

Looking downwards:

- an OS issues instructions to control hardware devices
- it does so to allocate and manage hardware resources on behalf of programs

Looking upwards:

- OS gives apps a high-level programming interface (system call interface)
- OS implements this interface using low-level hardware devices
  - file open / read / write close vs. disk block read / write

Application

A program that relies on the system call interface

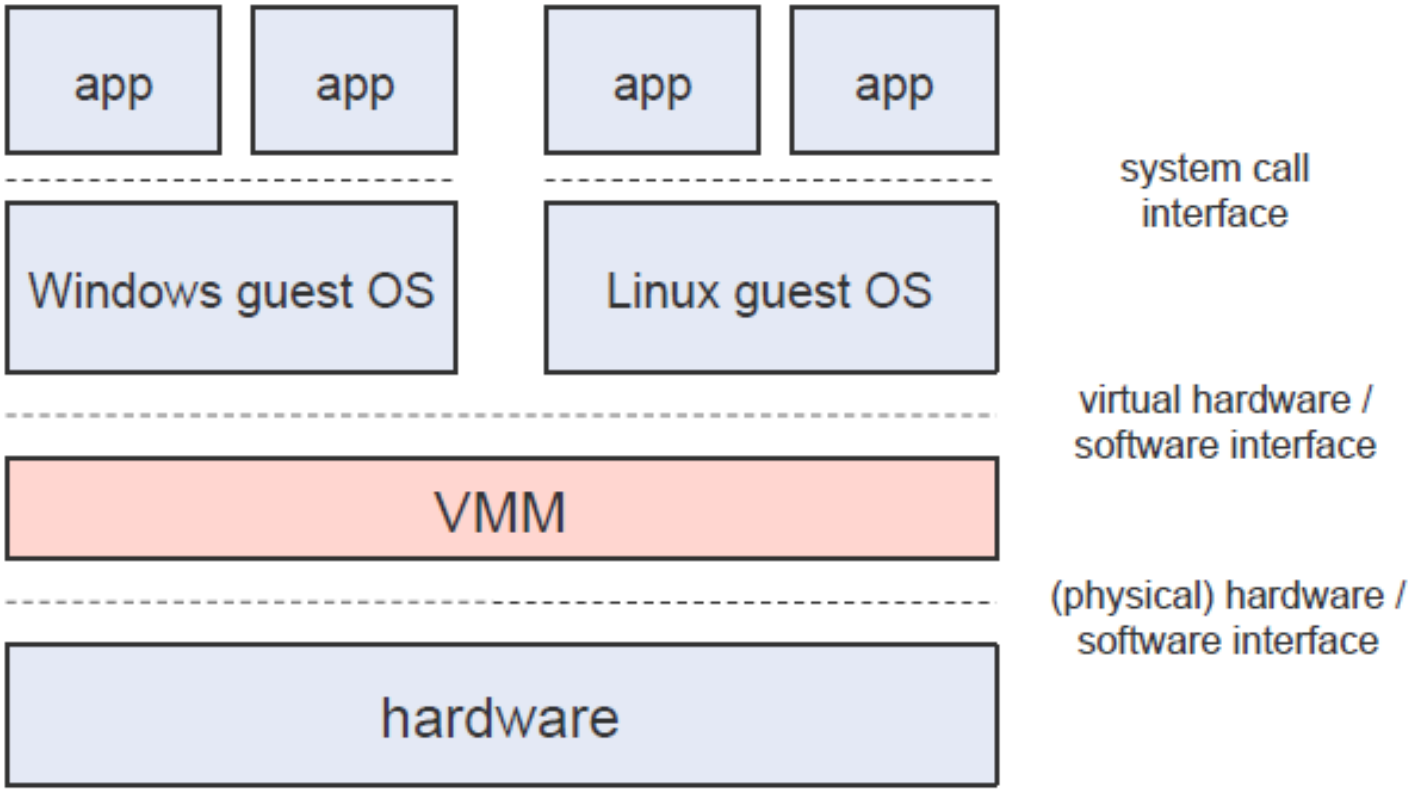
- While executing it, the CPU runs in unprivileged (user) mode
- a special instruction ("intc" on x86) lets a program call into the OS
  - the OS uses this to expose system calls
  - the program uses system calls to manipulate file system, network stack, etc.
- OS provides a program with the illusion of its own memory
  - MMU hardware lets the OS define the "virtual address space" of the program

Is this safe?

- most instructions run directly on the CPU (fast)
- but sensitive instructions cause the CPU to throw an exception to the OS
- address spaces prevent program from interfering with memory of OS and other programs
- it's as though each program runs in its own, private machine (the "process")

The key idea of the Virtual Machine

Run the OS kernel as a user-level application:



VMM: Virtual Machine Monitor

Characteristics (Popek and Goldberg, 1974)

Fidelity the VMM provides an environment which is essentially identical with the original machine

Performance programs that run in this environment show at worst only minor decreases in speed

Isolation the VMM is in complete control of system resources

Outline

- Virtual machines
- Virtualization approaches
- Applications of virtualization

VMM: Two approaches

Hosted Architecture

- Install as application on existing x86 “host” OS, e.g. Windows, Linux, OS X
- Small context-switching driver
- Leverage host I/O stack and resource management
- Examples: VMware Player/Workstation/Server, Microsoft Virtual PC/Server, Parallels Desktop, Virtualbox

Bare-Metal Architecture

- “Hypervisor” installs directly on hardware
- Acknowledged as preferred architecture for high-end servers
- Examples: VMware ESX Server, Xen, Microsoft Viridian (2008)

A key problem for the VMM...

What happens when guest OS issues a sensitive instruction?

- What (virtual) hardware devices should OS see?
- How do you prevent apps running on OS from hurting OS?
- or apps from hurting the VMM...
- or one OS from hurting another OS...or the VMM...

Trap-and-emulate (Goldberg)

Answer: rely on CPU to trap sensitive instructions and hand off to VMM

- VMM emulates the effect of sensitive instruction on the virtual hardware that it provides to its guest OSs
- instead of OS providing high-level abstractions to process via system calls...
- VMM provides a virtual HW/SW interface to guest OSs by trapping and emulating sensitive instructions

Goldberg (1974): two classes of instructions

- privileged instructions: those that trap when CPU is in user-mode
- sensitive instructions: those that modify hardware configuration or resources, and those whose behavior depends on HW configuration

A VMM can be constructed efficiently and safely if the set of sensitive instructions is a subset of the set of privileged instructions.

Performance implications of trap-and-emulate

There is almost no overhead for non-sensitive instructions

- they execute directly on the CPU, and do not cause traps
- CPU-bound code (e.g., many SPEC benchmarks, some scientific programs) execute at the same speed on a VM as on a physical machine

There is a large potential performance hit for sensitive instructions

- they raise a trap and must be vectored to and emulated by VMM
- I/O or system-call intensive applications get hit hard
  - recent hardware extensions try to improve this by letting the hardware handle instructions that used to cause trap/emulate
  - in essence, these extensions make the CPU aware of VM boundaries

Another problem...

Until 2005, the Intel architecture did not meet Goldberg's requirement for the construction of a VMM

- 17 instructions were not virtualizable
- they do not trap, and they behave differently in supervisor vs. user mode
  - some leak processor mode (e.g., SMSW, or store machine status word)
  - some behave differently (e.g., CALL or JMP to addresses that reference the protection mode of the destination)

How to make Intel virtualizable

You have four choices...

1. Emulate: do not execute instructions directly, but instead interpret each
   - very slow (Virtual PC on the Mac)
2. Paravirtualize: modify the guest OS to avoid non-virtualizable instructions
   - very fast and safe, but not “pure” or backwards compatible (Denali, Xen)
3. Use binary translation instead of trap-and-emulate:
   - advanced technique; used by Mendel Rosenblum for VMware
4. Fix the CPUs:
   - In 2005/2006, Intel introduced “VT”, and AMD introduced “Pacifica”
     - re-implemented ideas from VM/370 virtualization support
     - basically added a new CPU mode to distinguish VMM from guest/app
   - now building a VMM is easy!
   - and VMware must make money some other way...

Outline

- Virtual machines
- Virtualization approaches
- Applications of virtualization

Applications of virtualization

Enterprise Consolidation Eliminate server sprawl by deploying systems into virtual machines that can run safely and move transparently across shared hardware

Business Continuity Reduce cost and complexity by encapsulating entire systems into single files that can be replicated and restored onto any target server

Enterprise Desktop Secure unmanaged PCs without compromising end-user autonomy by layering a security policy in software around desktop virtual machines

Test and Development Rapidly deploy test and development servers; store libraries of pre-configured test machines

Enterprise consolidation

- big companies usually have their own machine rooms or data centers
- operate many services: mail servers, file servers, Web servers, remote cycles
- want to run at most one service per machine (administrative best practices)
- leads to low utilization, lots of machines, high power bills, administrative problems
- instead, run one service per virtual machine
- and consolidate many VMs per physical machine
- leads to better utilization, easier management

Amazon EC2

Large-scale, hosted cloud computing

- Cloud provider buys tens of thousands of PC-like, rack-mounted servers to provide a data center
- You run your software in a VM on their computers and pay them rent
  - VM is a convenient container for uploading software and a safe "sandbox" that prevents you and other customers from harming each other

Acknowledgments and References

The material in these slides is taken from:

- a 2008 lecture by Steve Gribble of University of Washington on Virtual Machine Monitors: Implementation and Applications
- a 2007 lecture by Carl Waldspurger, Virtualization 101, hosted by VMware

References

- R. Goldberg. Survey of virtual machine research. IEEE Computer, 7(6):34-45, June 1974.
- G. Popek and R. Goldberg. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 17(7):412-421, July 1974.