

Virtualization

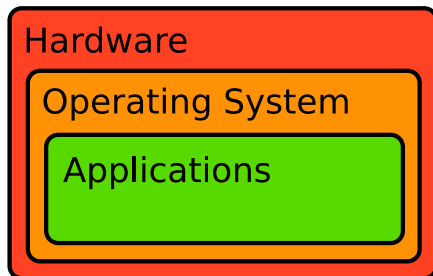
...or how adding another layer of abstraction is changing the world.

CIS 399: Unix Skills
University of Pennsylvania

April 6, 2009

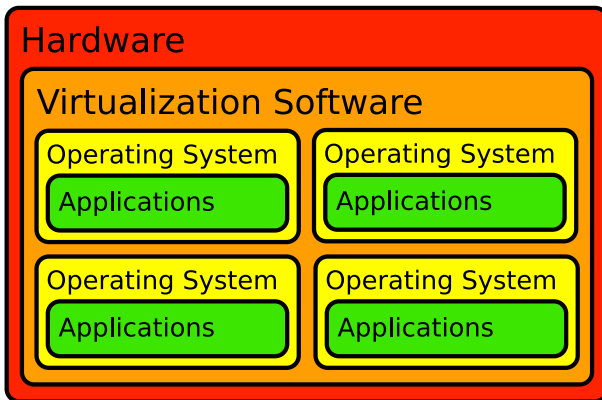
What is virtualization?

- Without virtualization:



What is virtualization?

- With virtualization:



Why virtualize?

Why virtualize?

- Operating system independence
- Hardware independence
- Resource utilization
- Security
- Flexibility

Virtualization for Users

- Parallels Desktop and VMware Fusion have brought virtualization to normal computer users.
- Mostly used for running Windows programs side-by-side with OS X programs.
- Desktop use has pushed support for:
 - ▶ USB devices
 - ▶ Better graphics performance (3d acceleration)
 - ▶ Integration between the guest and host operating system and applications.

Virtualization for Developers

- Build and test on multiple operating systems with a single computer.
- Use VM snapshots to provide a consistent testing environment.
- Run the debugger from outside the virtual machine.
 - ▶ Isolates the debugger and program from each other.
 - ▶ Allows easy kernel debugging.
 - ▶ Snapshotting and record/replay allow you to capture and analyze rare bugs.

Virtualization for Business

- Hardware independence - upgrade hardware without reinstalling software.
- Resource utilization - turn 10 hosts with 10% utilization into 1 host with 100% utilization. Big power and cooling savings!
- Migration - move a server to a different machine without shutting it down.
- High availability - automatically restart a server on a different host if there is a hardware failure.

Cloud Computing

- Cloud computing, a fancy term for exporting your computational needs to a computer somewhere else, over the internet.
- Amazon EC2 offers administrative access to your own virtual computer running on a host machine in one of their datacenters.
- Charged based on how much you use it.
- Don't need to maintain your own hardware.

How it works...

- All about building a computer in software.
- The CPU is simple: memory, instructions.
- Devices are complicated:
 - ▶ video
 - ▶ storage
 - ▶ network
 - ▶ sound
 - ▶ etc...
- In reality, devices make CPUs complicated too.

Enforcing Isolation

- Kernel space vs. userspace
 - ▶ Kernel code can interact with devices.
 - ▶ User code doesn't, it has to call kernel functions.
- The processor enforces this boundary.
- Since user code doesn't touch devices, it is easy to run in a virtual machine.
- Kernel code needs to be handled specially.

Virtual Devices

- Guest operating systems can't access real hardware directly (in most cases).
- Virtualization platform creates *virtual* devices.
 - ▶ Virtual disks backed by a file on the host system.
 - ▶ Virtual graphics card with graphics output to a window.
 - ▶ Virtual network card that shares the host's network connectivity.
 - ▶ etc...
- One advantage of virtualization is that these devices are always the same, only one driver needed no matter the host hardware.

Virtualization methods

- Emulation
- Binary translation
- Para-virtualization
- Hardware-assisted virtualization
- OS-level virtualization

The biggest difference between these methods is how they handle kernel code.

Emulation

- Build the whole system in software.
- Write a function to handle each processor instruction.
- Imitate device behavior.
- Really slow!
- Used by Boches, QEMU, and other x86 simulators.

Binary translation

- Rewrite the kernel code on the fly so that it accesses virtual devices instead of real devices.
- User code doesn't get modified at all.
- Much faster!
- Used by VMware.¹

¹Binary translation is also used to speed up emulators that run code from one hardware architecture on the other. In this case, both userspace and kernelspace code must be translated.

Para-virtualization

- Teach the kernel how to access virtual devices.
- No translation or emulation necessary anymore, but requires a special kernel in the guest operating system.
- Similar speed to binary translation, or slightly faster.
- Used by Xen.

Hardware-assisted virtualization

- Teach the processor about a new type of code with different privileges.
 - ▶ Hypervisor, kernel, userspace.
 - ▶ Implemented by Intel VT-x and AMD-V on modern processors.
- No modifications necessary to the guest operating system, like emulation and binary translation.
- Similar in performance to binary translation and paravirtualization.
- Used by KVM and HyperV (optional for Xen and VMware).

OS-level virtualization

- Operating system kernel splits itself into different “zones.”
- Lower overhead, since only one OS instance is running.
- Requires all instances to be running the same operating system, even kernel version.
- Used by Linux VServers, Solaris Zones and others.

Para-virtualized Device Drivers

- Systems other than paravirtualization must mimic real hardware so that unmodified operating systems can run.
- Instead you can install special drivers that know they are running in a virtual machine.
- Avoids the need for emulating the behavior of a real device.
- Provides better performance for graphics, networking, etc.
- e.g. VMware Guest Tools, VirtualBox Extras, etc.

Downsides

- I/O performance is never quite as good as native.
- Programs perform I/O using a “system call,” a special type of function call that executes code in the kernel, with kernel-level privileges.
 - Even if user programs run unmodified kernel code has to deal with the virtualization layer.
 - Especially evident in hosted virtualization (one OS on top of another), since I/O has to pass through the host OS too.

Downsides (contd.)

Extra disk and memory usage.

- Multiple systems means multiple virtual drives worth of data, mostly the same operating system.
- Each system needs it's own memory.
- If many copies of the same OS are running, lots of memory contains the same general information.
- VMware implements a special memory sharing system to avoid this.

Conclusion

- Virtualization adds a lot of new flexibility.
- In the enterprise, the market is really heating up:
 - ▶ VMware ESX
 - ▶ Citrix XenServer
 - ▶ Redhat KVM
 - ▶ Microsoft HyperV
- The future:
 - ▶ Mobile device virtualization?
 - ▶ Built-in hypervisors?