Practical Applications of Virtualization

Mike Phillips <mpp@mit.edu>

IAP 2008

SIPB IAP Series

http://stuff.mit.edu/iap/

http://stuff.mit.edu/sipb/

Some Guy Rambling About Virtualization Stuff He's Read About and Played With Some And Wants to Share and Discuss With You

Mike Phillips

IAP 2008

SIPB IAP Series

http://stuff.mit.edu/iap/

http://stuff.mit.edu/sipb/

What's Virtualization?

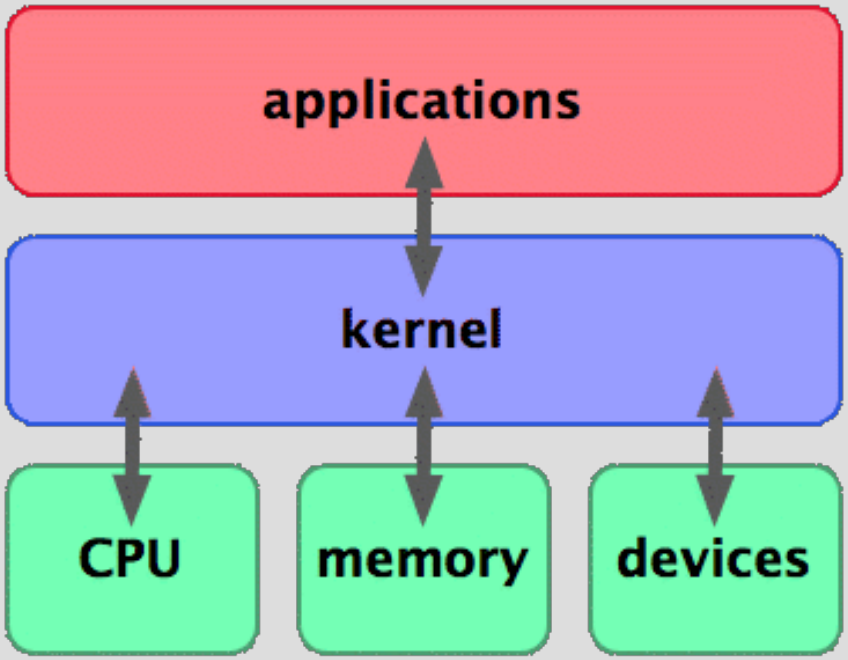
- Abstraction of computing resources
  - Hiding physical details and presenting a logical interface
- Typical resources to virtualize:
  - CPU
  - Memory
  - Storage
  - Network
- Resource virtualization vs. Platform virtualization

But first, some basics...

- What's a computer?
  - CPU
  - Memory
  - Storage
  - Other devices
- What's a CPU?
  - Runs programs
  - Instruction Set Architecture
    - Intel IA-32 (x86), AMD x86_64

But first, some basics...

- What's an Operating System?
  - Shares physical resources
  - Provides an interface to user programs
- Virtual memory
- Device drivers
- Userspace libraries



http://en.wikipedia.org/wiki/Image:Kernel.png

Platform Virtualization

- Virtualize the whole computer
  - Using another computer
  - Hey, they're all universal Turing machines
  - Virtual computer could be something new, or essentially the same architecture as the real hardware
- Emulation
  - No restriction on virtual computer architecture
  - Bochs: cross-platform IA-32 emulator
  - Qemu: Emulates several platforms, runs on x86 and PowerPC
  - Good academic uses

More commonly...

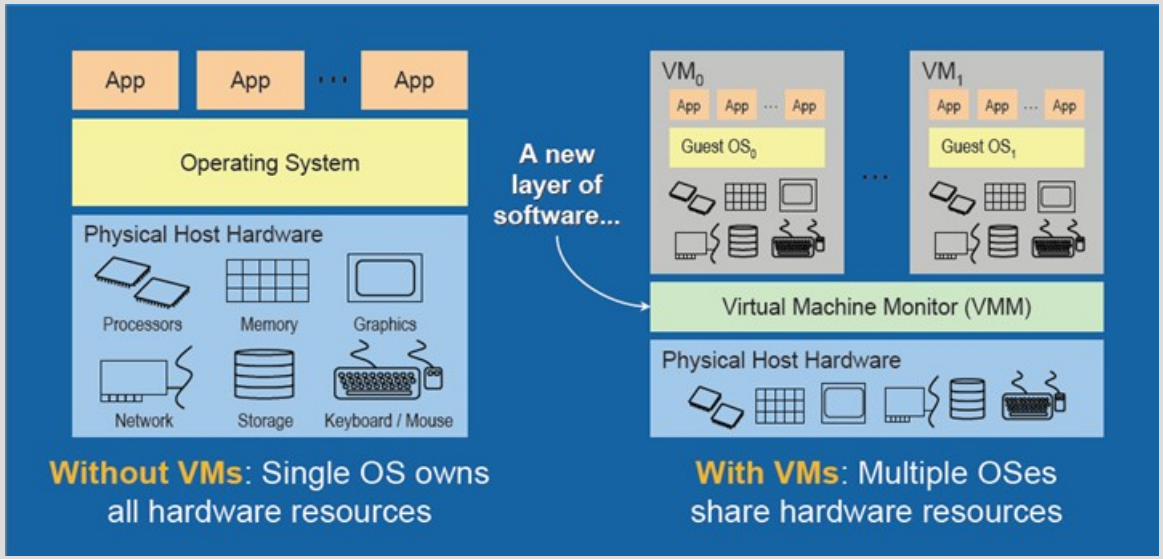
- Divide a physical computer up into virtual computers
  - Virtual machines have the same architecture as host
  - Maximize speed
    - Virtualization has overhead
    - Compare vs. native speed

Why?

- Server consolidation
  - PCs are cheap, powerful.
- Isolation
  - Vs. “all on one box”
  - Security
- Experimentation
  - Try new things without impacting existing services
- Cool factor
  - Completely encapsulate a virtual machine and manipulate it (copy, move, checkpoint)

How?

- Physical hardware
- Virtual Machine Monitor
- Virtual machines
- Guest operating systems



http://softwarecommunity.intel.com/articles/eng/1408.htm

How?

- Virtual Machine Monitor
  - Also called a “hypervisor”
- Takes sole responsibility for hardware
  - Like the “OS” normally would
- Prevent the execution of sensitive instructions
  - Would interfere with VMM, host OS, or other VMs

The problem with x86

- Privileged mode
  - Privilege hierarchy: Rings 0 through 3
  - Call gate / system calls
- Privileged instructions cause a trap
  - VMM can emulate expected behavior
- Not all sensitive instructions are privileged
  - 17 instructions are problematic
  - Context-dependent
  - May fail silently
- Virtualization isn't new, but not in x86 design
  - IBM's CP-40 (1967), System/370 (1972)

Binary patching

- Analyze instruction stream ahead of time
  - Look for problematic situations
  - Set hardware breakpoints, or otherwise patch the code
  - Cache results
  - VMWARE approach (out of research at Stanford)
    - Supports unmodified guest operating systems

Paravirtualization

- Guest OS is modified to be aware of VMM
- The VM provided is like x86, but isn't actually x86
  - Missing the problematic bits
- Xen's approach
  - Xen hypervisor in Ring 0
  - Dom0 (host) – Modified Linux OS
  - DomU (guests) – Paravirtualized versions of:
    - Linux (integrated in the mainline kernel at 2.6.23)
    - OpenBSD
    - OpenSolaris
  - Device access via Dom0

Hardware extensions

- Intel VT-x / AMD-V
  - Available within last year
- Adds instructions for interacting with VMM
- Root mode vs. non-root mode
  - Both have rings 0-3, but only root mode (for VMM) has real hardware access
  - Traps happen when they should
- But what about I/O devices?
  - Need IOMMU to make giving devices to guests safe (DMA)
  - Intel VT-d coming out now, not quite available

Xen HVM

- Xen Hardware Virtual Machine
  - Abstraction over Intel and AMD extensions
- Support for unmodified guests
  - Yes, this means Windows
- Uses QEMU for I/O virtualization
  - Presents generic looking devices to guest
    - IDE controller
    - Network card
    - Vanilla VGA
- Not all CPUs have these extensions

A Real Use Case

- I have an older machine running MythTV
  - Ubuntu Edgy
  - Hauppauge PVR-350 MPEG2 encoder/decoder
- Want to add more services
  - Outward-facing websites
  - Media sharing
  - Debathena
- Isolation and security
- Start by making existing installation a guest

Installing a new OS

- Debian Etch (stable)
  - Has xen packages
  - Has openvz packages (sort of.. more later)
  - Has vserver packages
- Boot off the install CD
- Current disk use:
  - / - 10G
  - /mythtv – 500GB
- Moved /mythv to /mythtv/mythv
- Copied /* to /mythtv/ partition
- Installed new OS on /
- Can still boot old system

Xen on Etch

- Thank you, Internet
  - http://wiki.debian.org/Xen
  - http://www.howtoforge.com/debian_etch_xen_from_debian_repository
- dom0 (host)
  - `apt-get install xen-linux-system-2.6.18-4-xen-vserver-686 libc6-xen xen-tools`
  - Gets us the xen hypervisor and modified kernel for dom0

Xen on Etch

- domU:
  - `xen-create-image --debootstrap --hostname xen-etch --dhcp --dist=etch`
  - `xm create /etc/xen/xen-etch.cfg`
  - `xm list`
  - `xm console xen-etch`
- Tweak configuration for network and memory
- Manually add /dev/sda* device files in dom0
- Make another domU
  - Modify the disk image to contain a copy of the MythTV Ubuntu Edgy installation
  - Configure PCI-passthrough so this domU will own the PVR-350; hide from dom0

Drum roll, please..

- Extra hardware modules
  - ivtv for PVR-350
  - lirc_modules for IR remote input
- Built fine (using Debian module-assistant)
- Kernel panic in ivtv
  - DMA; swiotlb issue
  - swiotlb-related switches suggested by the Internet did not help
  - Possibly try newer Linux kernel, ivtv, or Xen?
- Okay, what about using ivtv in dom0
  - Similar result

Maybe Xen isn't for this...

- Machine only has 512MB of memory
- Hard virtualization means I need to partition that among the domUs
- Xen is way cool, but it may be overkill for this application.
- Perhaps some sort of virtualization that gives me good isolation but isn't as strict?

Container Virtualization

- Also “Operating-system virtualization”
  - Single kernel instance
- Create isolated virtual environments
  - Also called Virtual Private Servers
  - Processes locked inside
- chroot jails
- Solaris zones/containers
- BSD jails
- Linux: vserver, OpenVZ

OpenVZ

- SWSoft's GPL'ed version of Virtuosso
- File system isolation
- Disk quotas
- I/O rate-limiting
- Memory Limits
- CPU quotas
- Network isolation
- Checkpointing / Live migration
- Defanged root user in VEs
- Limited hardware access

OpenVZ

- Linux Kernel patch
  - PID namespaces (2.6.24)
  - IPC namespaces (semaphores, shared memory)
  - Separate hostnames for VEs
  - Virtual CPU fair scheduler (2-level)
  - 2-level disk quotas
  - Hardware access/capability limits on VEs
- Distributed as patch or pre-built kernels
  - Even has a Debian repository
- Debian Etch has a patch package and userspace tools (not as shiny and new)

Installing OpenVZ

- http://www.howtoforge.com/debian_etch_openvz
- http://wiki.openvz.org/Installation_on_Debian
- First pass: used the openvz.org apt repository
  - But very few kernel features enabled
  - Certainly no PVR-350 support
- Built it myself, the Debian way:
  - Installed kernel sources, copied my current config, applied patch, and checked selected kernel features

Creating a VE

- vzctl create 101
- vzctl set 101 --onboot yes --save
- vzctl set 101 --hostname openvz-test --save
- vzctl set 101 --ipadd 10.1.0.5 --save
- vzctl set 101 --nameserver 10.0.0.1 --save
- vzctl start 101
- vzctl enter 101
  - Get a shell inside the VE

How VEs are stored

- /var/lib/vz/private/101 contains VE's files
- Some vzctl --set commands modify files here
- Mounted on /var/lib/vz/root/101/ at start
- Bind mounts are possible
  - mount --bind /some/path /var/lib/vz/root/101/foo
  - Done after VE started, can be automated
- VE0 can see into all VEs' filesystems

What VEs Cannot Do

- Get raw access to the network, nor reconfigure the network interface (by default)
- Insert kernel modules
- Exceed resource limits on CPU, memory, disk, etc
- Access devices via /dev without explicit permission
- Run their own operating system

User Bean Counters

- cat /proc/user_beancounters
- 20 resources that can be restricted
  - Process count
  - Socket count
  - Memory allocation guarantee
- Defaults are rather restrictive
  - Configuring is a bit of a pain, though decently documented
- Output includes a failure count for each
  - Watch this if you are having problems in VEs

Back to my use case

- Want to move MythTV to VE101
- Just copied files over /var/lib/vz/private/101/
- Added mount script to bind mount the video storage area
- However, kernel modules for the relevant hardware must be rebuilt for new kernel

Back to my use case

- Extra kernel modules
  - ivtv
    - Built and inserted just fine (in VE0)
  - lirc_modules
    - Needed to patch myself (find_task_by_pid())
  - openafs
    - Not tried yet, but have seen a patch on openvz.org
- X Window System inside VE101
  - MythTV front-end uses framebuffer device
  - Grant explicit access to /dev/fb0 with vzctl
  - MythTV back-end needs to read from MPEG encoder-- grant access to /dev/video0

More typical usage

- Virtual hosting
  - Divide physical hardware up for each customer
  - Isolation is nice
  - Can overcommit physical hardware resources slightly
    - Promise more physical memory than exists
    - Swap will cover usage bursts
  - “vzctl exec” for mass changes
  - Better than just chroot
  - Not as isolated as Xen
  - Worry about third-party nature of patch

Final thoughts

- Xen and OpenVZ aren't really competing
- Both are gaining mainstream kernel acceptance and are actively developed
- Xen is really going places
  - Compatibility layer for Windows Server 2008's hypervisor to allow running paravirtualized guests
- One day maybe I'll be able to dedicate my 3D graphics card to a VM
  - Maybe even securely... (IOMMU)
- Virtualization is hot
  - Complex software distributed as VMs?

Links, etc

- http://stuff.mit.edu/iap/virtual/