Introduction to Virtualization on Linux using KVM

Pat Barron
(pat@lectroid.com)
Western Pennsylvania Linux Users Group

Disclaimer

- “The views expressed in this presentation are those of the author, and do not necessarily reflect the opinions, positions, or strategies of WPLUG, the author's employer, or any other person or organization.”

Goals

- Define the problem we're trying to solve
- Brief overview of some general strategies for solving the problem
- Introduce KVM, a virtualization solution included with many Linux distributions
- Explain the basic capabilities of KVM
- Review what you need in order to use KVM
- Review some competing no-cost solutions (some open source, some not)
- Demonstrate the use of KVM on Fedora 16

Non-goals

- You'll learn about the basic use of KVM, but we won't get into too much detail about exactly how KVM does what it does
- We won't cover the use of KVM and associated tools on distributions other than Fedora 16 (specific instructions for your distro may vary)
- We won't cover configuration of bridged networking for KVM (you'll understand what that means after the demo...). It's not really hard, but it's a little involved, and beyond the scope of this introduction

The problem: Too Many Computers

- You may need or want separate systems to run different operating systems, different software packages, or to isolate environments from each other
  - You may want to test different Linux distros
  - You may want to set up temporary test environments
  - You probably don't want your web server running on the same system as your accounting system
  - Etc...
- You may not actually have enough computers for all of these separate purposes (or may just not want to have so many computers)

Why is this a problem?

- All those extra computers cost money
  - Money to buy them
  - Money to power them
  - Money to cool them
  - Money to maintain them
- They also take up an ever increasing amount of physical space

What makes it even more frustrating...

- Modern computers have an impressive amount of capacity
- More capacity than most applications will ever use, in fact
- Even if you use your computer to it's capacity limits occasionally, you probably won't do so all the time
- So you've bought and paid for a bunch of capacity you're not going to use
- So why do I have to spend money on another computer again?
- Every additional computer you buy increases your total wasted investment

Virtualization addresses this problem.

- Virtualization lets you create the illusion of one or more additional, separate systems, using only the resources you already have available
- “If you can see it, and it's there, it's real. If you can see it, and it's NOT there, it's virtual. And if you can't see it and it's not there, it's gone...”

Some general strategies for harnessing excess capacity

- Emulation
- Hardware virtualization
  - Hosted virtualization
  - Hypervisor virtualization
- Paravirtualization

Emulation

- Software models the behavior of the target system
  - Emulator models processor, physical memory, I/O devices, etc., in software code and data structures.
  - Emulator examines instruction stream (including I/O), decides what the instructions would do on the emulated hardware, and replicates that behavior on the host system.
- Pros:
  - Can emulate any hardware you have a functional spec for (not just the type of system you're running on).
  - Can (if done correctly) run unmodified operating systems and software.
  - Popular in the console and arcade gaming community.
- Cons:
  - Can be difficult to get right – must be “bug for bug” compatible.
  - SLOW!

Hardware virtualization

- Leverages the underlying hardware
  - Whenever possible, code executes directly on the underlying hardware
  - A special program (virtual machine monitor) may manage access to the hardware, emulate I/O devices, trap and emulate privileged operations
- Pros:
  - Much faster than emulation
  - Can run unmodified operating systems and software
- Cons:
  - Can only run software for the type of system being virtualized.
  - Some architectures (including x86...) are difficult or impossible to cleanly virtualize (hardware assists can help).

Hosted virtualization

- Virtual system and VMM run as standard user-mode processes (with O/S assist) on an otherwise ordinary operating system.
- Pros:
  - Less complicated to implement
  - Often easier to use
  - Can run unmodified operating systems and software
  - Minimal interference with host system, which can continue to operate as usual
- Cons:
  - Somewhat less efficient than using a hypervisor

Hypervisor virtualization

- VMM is a small, purpose-built operating system dedicated to this task; arbitrates access to resources at low level
- Virtual systems run under the hypervisor
- Pros:
  - Can be more efficient than hosted virtualization
  - Can run unmodified operating systems and software
- Cons:
  - Takes over the whole machine, which can then only be used for running virtual machines

Paravirtualization

- Hardware virtualization, with a twist...
- Virtual system behaves somewhat differently than “real” hardware, usually with the intent of efficiency
- Pros:
  - Somewhat more efficient than pure hardware virtualization
- Cons:
  - Can only run operating systems and software which are “aware” they may be running in virtualized environment – standard, unmodified software will not run

KVM – A virtualization system for Linux

- Developed by Qumranet, acquired by Red Hat in 2008
- Primarily consists of kernel modules that arbitrate access access to resources, facilitate I/O, and permit VMM (QEMU, by default) interface to virtual systems
- Hybrid strategy – hosted virtualization + paravirtualization
  - Optional “virtio” module paravirtualizes I/O for efficiency, for operating systems that can use it; need not be used
- Probably already present on your Linux system, or available in default repositories
- Supports desktop and data center virtualization
- Supports guest migration (moving a running guest from one KVM host to another “on the fly”)
- Default installation on Fedora 16 supports only NAT networking; bridged networking can be set up, and is required to enable virtual guests to have virtual network interfaces visible outside of the host

KVM – Supporting Tools

- QEMU – virtual machine monitor, allocates resources, emulates I/O devices
- SeaBIOS – BIOS code for virtual systems
- virt-manager – GUI to create, remove, and manage virtual systems

KVM – What you need to use it

- A fast machine (most modern PCs available include a processor fast enough to use KVM in a reasonable way)
- A processor with hardware virtualization support (Intel VT-x or AMD AMD-V [a/k/a “Pacifica”]) – KVM will not work without this, you will still be able to use virtual systems (slowly...) using QEMU emulation
  - Ensure virtualization feature is enabled in BIOS – some systems disable this by default!
- A 64-bit processor (if you want to run 64-bit virtual systems)
- As much memory as possible (my recommended minimum is 4 GB; 8 GB would be better) – memory is shared between host operating system and all virtual systems
- As much disk as possible (shared between host operating system and all virtual systems)

KVM – How to install it (on Fedora 16)

- At initial system installation using GUI install– select “Customize Now” during package selection; under “Base” category, select “Virtualization” group
- After initial system installation – from command line:
  - yum groupinstall “Virtualization”
- After initial system installation – from Add/Remove Programs GUI
  - Install packages from “Virtualization” group

KVM – How to start using it

- Start “Virtual Machine Manager” from GNOME desktop
  - You'll see more details in the demo

Other no-cost virtualization solutions

- KVM isn't the only game in town
- However, it might be the most convenient or sensible, since it already comes with your Linux system

QEMU (without virtualization)

- Strategy: Emulation
- QEMU alone can be used without KVM or processor virtualization feature
- Can run unmodified guest operating systems and software
- Can emulate other processor architectures besides x86, such as ARM (QEMU is used in Android SDK system emulator), PowerPC, SPARC, some others
- It's an emulator, so it's slow

Xen

- Strategy: hypervisor virtualization / paravirtualization (early versions supported only paravirtualization)
- Varying levels of support in major Linux distributions
- Processors without hardware virtualization features can support only paravirtualized guests; with hardware virtualization features, can run unmodified guest operating systems
- Fell out of fashion for a while because Xen kernel mods were not accepted into mainline Linux kernel – later kernels include Xen mods out of the box

Oracle VirtualBox

- Strategy: hosted virtualization
- Popular and well supported
- Must be managed/patched separately from operating system
- Some features (e.g., USB 2.0) available only with proprietary-licensed add-on pack, free only for personal, non-commercial use

Citrix XenServer [Free Edition]

- Strategy: hypervisor virtualization/paravirtualization
- Based on Xen and CentOS
- Supports advanced features such as guest migration (moving a running guest from one hypervisor to another)
- Takes over the entire machine, though can use GUI console if you want to (not recommended)
- Proprietary, not open source, requires license activation and annual renewal (even for Free Edition)
- Primarily intended for data center use, not desktop

VMware Server

- Strategy: hosted virtualization
- On older systems, very easy to just drop in and run (though must be managed/patched separately from the operating system)
- Can be very challenging to get running on modern Linux systems, due to changes in kernel data structures
- Must rebuild VMware Server driver modules every time you update the kernel
- Generally works well, and easy to administer – though works very poorly when virtual guests do heavy I/O
- No support for virtual guest migration
- Best for data center virtualization, though can be used on desktop
- Proprietary, not open source – must register with VMware for a (free) license key in order to use
- VMware Server is a dead product, will be out of support soon, not clear if new license keys will be made available

VMware vSphere Hypervisor (a/k/a VMware ESXi)

- Strategy: hypervisor virtualization
- Free version of VMware ESX
- Feature-limited (no guest migration, no centralized management, etc.)
- Takes over entire machine; display is made completely useless. All management except initial setup done via vCenter Management Client; all access to virtual guests over the network, or via vCenter Console Client. Intended for data center environment
- Performs very well; does not require processor virtualization feature
- Limitations on systems it can run on – ESXi 4.0 enforces max of two processor sockets and 256 GB RAM; ESXi 5.0 enforces max of 32 GB RAM and 8 GB RAM max per virtual guest. Most people won't care...
- Proprietary, not open-source; must register with VMware for a (free) license key in order to use

KVM Live Demo

- Stay tuned...