Getting To Know The Linux Kernel Virtual Machine (KVM)

Ryan Matteson
matty91@gmail.com
http://prefetch.net

Overview

- Tonight I am going to discuss KVM, and show how to configure a host to use KVM
- I plan to split my presentation into 3 parts:
  - Part 1 will provide an overview of the technology
  - Part 2 will show you how to use the technology
  - Part 3 will be a Q&A period

What is KVM?

- KVM (Kernel Virtual Machine) is a full virtualization solution that allows you to run unmodified guests (Linux, Solaris, Windows) on x86 and X64 systems that support hardware virtualization extensions
- KVM is implemented as two components:
  - A kernel module that manages the hardware resources (the hypervisor)
  - A userland process (a modified version of QEMU) that provides PC platform emulation
- When a new KVM guest is booted, it becomes a process (qemu-kvm on Fedora and CentOS hosts) of the underlying operating system and is scheduled like any other process*
- The qemu-kvm process is a modified version of QEMU, and communicates with the KVM hyper visor through the /dev/kvm character device

* There is one exception. The modified QEMU processes run in “guest” mode vs. user or kernel mode

Why would I want to use KVM?

- Allows you to run multiple operating system instances on a single system, which is great for environments that need a place to test things, or for companies that are consolidating hosts to better utilize existing hardware resources
- Supports live migration, which allows you to move running guests between systems
- Libvirt has been enhanced to manage KVM guests, allowing you to re-use provisioning and management infrastructure built on top of it
- KVM has been part of the mainline kernel source since 2.6.20 was released, so you don't need to apply a slew of kernel patches to get KVM up and operational!

Does my machine support KVM?

- KVM utilizes hardware virtualization extensions from Intel and AMD, which are available in modern CPUs
- You can check /proc/cpuinfo to see if you have the necessary virtualization extensions:

```
$  egrep '^flags.*(vmx|svm)' /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
 pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp lm 3dnowext
 3dnow rep_good nopl pni cx16 lahf_lm cmp_legacy svm extapic cr8_legacy 3dnow
```

- If you see one of the following flags, your golden:
  - vmx indicates the CPU support Intel's virtualization extensions
  - Svm indicates the CPU supports AMD's virtualization extensions

Managing KVM Guests

- KVM guests can be managed in one of two ways:
  - Directly through the modified QEMU executable
  - Indirectly through libvirt (this fires up the QEMU binary under the covers)
- The QEMU qemu-system-x86_64 /qemu-kvm binary can be used to directly manage KVM guests
- Libvirt provides the virsh (virtual shell) to manage KVM guests, and the virt-install utility to configuration and provision new guests
- The examples in this presentation will use libvirt

KVM Hardware Support

- KVM / QEMU provide device emulation for several types of devices:
  - IDE disks
  - SCSI disks
  - USB, Parallel, Serial
  - Virtio for accelerated network and disk performance
- KVM / QEMU also several forms of host networking:
  - User networking
  - Private bridge networking
  - Public bridge networking
  - Virtual distributed Ethernet

How do I create a KVM guest?

- Guests can be installed from DVDs and ISO images, or though your favorite network installation method
- To create a new guest, you can pass one or more parameters to the virt-install command:

```
$ virt-install --connect qemu:///system \
        --name puppet --ram 512 \
        --file /nfs/vms/puppet.img \
        --network=bridge:br0 \
        --accelerate -s 36 --pxe -d \
        --noautoconsole \
        --mac=54:52:00:53:20:00 \
        --nographics --nonsparse
```

Booting and accessing KVM guests

- Once a guest is created, you can manage it through the virtual shell (virsh)
- To boot a KVM guest, you can use the virsh “start” command:

  $ virsh start <guest>
- To destroy a running guest (i.e., pull the power plug), you can use the virsh “destroy” command:

  $ virsh destroy <guest>
- To access the hosts console, you can use the virsh “console” command (this requires the host to be configured to write to the serial console):

  $ virsh console <guest>
- **TONS** of additional options, which are documented in virsh(1)

Adding NICs to KVM guests

- NICs can be added at guest creation time by appending several “--network” options to the virt-install command line:

  ```
  $ virt-install --network=bridge:br0 –mac=X \
                 --network=bridge:br0 –max=Y …
  ```

- For existing guests, the virsh “edit” command can be used to edit the guest configuration, and a stanza similar to the following can be added to create a new NIC:

  ```
  <interface type='bridge'>
      <mac address='54:52:00:53:20:00'/>
      <source bridge='br0'/>
   </interface>
  ```

Adding Disks to KVM guests

- Disks can be added at guest creation time by appending several "--file" options to the virt-install command line

  $ virt-install –file /nfs/vms/puppet/disk1.img –s 18 \

  --file /nfs/vms/puppet/disk2.img -s 18 ….

- To attach a disk to an existing host, you will first need to create a disk image with either dd or qemu-create:

  $ qemu-img create /nfs/vms/puppet/disk3.img 18G

- The virsh "edit" command can be used to edit the XML definition, and a stanza similar to the following can be added to add a new virtual disk to the guest:

```
<disk type='file' device='disk'>
   <source file='/nfs/vms/puppet/disk3.img'/>
   <target dev='hda' bus='ide'/>
</disk>
```

Configuring Console Access

- KVM guest consoles can be accessed through VNC, or via a virtual serial console
- To enable VNC access, you can add the "—vnc" and "—vncport" options to the virt-install command line
- To configure serial access, you can add the "-nographics" option to the virt-install command line (the guest operating systems need to be configured to send output to ttyS0)

KVM Migration

- KVM supports live migration, which allows you to move active KVM guests from one host to another
- Migration requires that the the files backing the guest are located on some type of share storage (e.g., GFS2 file system, OCFS2 file system, NFS, CIFS share, etc.)
- To migrate a guest from the current host to a machine named "disarm", you can run the virsh migrate command with the "—live" option, the name of the guest to migrate, and a connection string:

  virsh # **migrate –live kvmnode1 qemu+tls://disarm/system**
- The connection string listed above contains the the driver (qemu) and transport protocol (tls) to use, the machine (disarm) to connect to, and tells virsh to connect to the system process

Backing up and Restoring KVM guests

- To back up the configuration of a guest, you can use the virsh “dumpxml” command:

  $ virsh dumpxml <guest>

- To restore a guest from an XML file, you can use the virsh “define” command:

  $ virsh define <pathtoxmlfile>

KVM Logging

- If you encounter a failure, the first place to check is the system and guest logs:

  /var/log/messages

  /var/log/libvirt/<guest>.log

- If the logs don't contain sufficient data to debug an issue, you can increase the log level by adjusting the log_level and log_outputs in the libvirtd.conf configuration file

Gotchas

- KVM is a relatively new technology, and with any new technology comes some growing pains
- If you encounter an issue, check the Fedora, KVM and libvirt bug trackers and mailing lists
- To avoid major issues, make sure you are running a relatively current version of QEMU, KVM and libvirt (the examples shown in the presentation came from a Fedora 11 host running the latest packages from rawhide)

Conclusion

- While KVM is relatively new, it is shaking out to be one of the leaders in Linux virtualization
- If you are interested in playing with KVM, it is most likely a <insert your favorite package manager> install away
- All examples in this article should work, and were tested on a Fedora 11 host running a Linux 2.6.30 kernel with the latest virtualization updates

Questions?

References

- KVM website:

  http://www.linux-kvm.org/page/Main_Page
- Configuring Linux hosts to log to a serial console:

  http://prefetch.net/blog/index.php/2009/06/17/redirecting-the-centos-and-fedora-linux-console-to-a-serial-port-virsh-console-edition/
- Libvirt Website:

  http://libvirt.org/
- QEMU Website:

  http://www.nongnu.org/qemu/