Running KVM for Dynamic Infrastructure Creation

Jay Brenneman – rjbrenn@us.ibm.com
August 12, 2015
Agenda

KVM Review

Virtualizing Networks with KVM

Virtualizing Disk with KVM
KVM Review

▪ Hopefully you attended Mark Post's 'KVM for z Systems' session?

▪ QEMU Virtualizes the Processor and Memory resources of a Server using the KVM resources provided by a Linux Kernel assisted by whatever hardware instructions the processor supports

▪ QEMU Virtualizes the IO of a server using a mix of paravirtualization and good ole trap and translate.

▪ From an administrators point of view – a virtual machine is contained within the qemu process running on the KVM host
Agenda

KVM Review

Virtualizing Networks with KVM

Virtualizing Disk with KVM
Virtualizing Networks with KVM

- KVM offers several methods to virtualize the host's network connectivity, each with their own plusses and minuses

- Ethernet Routing with NAT
- Ethernet Routing without NAT
- Ethernet Bridging
- MacVTap
- Open vSwitch
KVM – Ethernet Routing with NAT

KVM Host performs NAT translation on outbound packets
KVM – Ethernet Routing with NAT

- Relatively straightforward to set up
- Does not require additional IP addresses from the network
- MAC addresses local to KVM host

- Inbound connections require additional setup for each mapped port on each guest
- KVM host is performing packet forwarding
- Otherwise, there is no inbound connectivity

- Provides a good starting point for gaining familiarity with KVM

```xml
<network>
  <name>default</name>
  <bridge name="bridge0" />
  <forward mode="nat"/>
  <ip address="192.168.1.1" Netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.1.2" end="192.168.1.254" />
    </dhcp>
  </ip>
</network>
```

```xml
<interface type='network'>
  <mac address='52:54:00:c0:d5:cf' />
  <source network='default' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0' />
</interface>
```

Host definition

Guest definition
KVM – Ethernet Routing without NAT

External Network requires a route to 192.168.1.0 for the return trip

10.0.0.0/24

192.168.1.0/24

eth0 – 10.0.0.100

bridge0 - 192.168.1.1

Linux Kernel

NIC

10.0.0.0/24

Guest Linux OS

QEMU

tap

Guest Linux OS

QEMU

tap

Guest Linux OS

QEMU

tap

Linux Kernel

External Network requires a route to 192.168.1.0 for the return trip
KVM – Ethernet Routing without NAT

- Relatively straightforward to set up
- Does not require additional IP addresses from the network
- MAC addresses local to KVM host

- Inbound connections require a route for the private network in the external network
- KVM host is performing packet forwarding

- Provides a good starting point for gaining familiarity with KVM

```xml
code
<network>
  <name>default</name>
  <bridge name="bridge0" />  
  <forward mode="route" dev="eth0" />
  <ip address="192.168.1.1"
       Netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.1.2"
              end="192.168.1.254" />
    </dhcp>
  </ip>
</network>
```

Host definition

```xml
code
<interface type='network'>
  <mac address='52:54:00:c0:d5:cf'/>
  <source network='default'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000'
           bus='0x00' slot='0x06'
           function='0x0'/>
</interface>
```

Guest definition
KVM – Ethernet Bridging

External Network IP addresses required for each bridge attached guest.
KVM – Ethernet Bridging

- +++
  - Relatively straightforward to set up
  - Very common KVM networking for production environments
  - No external routing changes required

- ---
  - KVM host is performing packet forwarding
  - MAC address management required since MACs travel to the external network

- More or less the defacto network standard for KVM until recently

```xml
<network>
  <name>host-bridge</name>
  <bridge name="bridge0" />
  <forward mode="bridge"/>
</network>
```

```xml
<interface type='network'>
  <mac address='52:54:00:c0:d5:cf'/>
  <source network='host-bridge'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</interface>
```

Host definition

Guest definition
External Network IP addresses required for each guest
KVM – MacVTap

- +++
  - Higher performance than Bridging or Routing or NAT
  - Compatible with the Network teams management tools and processes
    - Supports VEPA, 802.1Qbg
  - No external routing changes required

- ---
  - MAC address management required since MACs travel to the external network
  - New Technology is new

- Fast, but requires kernels 2.6.34 or newer ( not in RHEL 6 )

```xml
<network>
  <name>macvtap-passthru</name>
  <forward mode="bridge">
    <interface dev="eth1"/>
    <interface dev="eth2"/>
    <interface dev="eth3"/>
  </forward>
</network>

<interface type='network'>
  <mac address='52:54:00:c0:d5:cf'/>
  <source network='macvtap-passthru'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</interface>
```

Host definition

Guest definition
KVM – Open vSwitch

External Network IP addresses required for each bridge attached guest

10.0.0.0/24

Guest Linux OS
QEMU

Guest Linux OS
QEMU

Guest Linux OS
QEMU

ovsbr0

eth0 – 10.0.0.100

eth1

Uplink

NIC

NIC

Linux Kernel

10.0.0.0/24
KVM – Open vSwitch

- +++
  - Relatively straightforward to set up
  - Compatible with the Network teams management tools and processes
    - Supports VEPA, 802.1Qbg, VLAN, and more!
  - No external routing changes required

- ---
  - KVM host is performing packet forwarding
  - MAC address management required since MACs travel to the external network

- Requires Open vSwitch package compatible with running Kernel

```xml
<network>
  <name>ovs-bridge</name>
  <bridge name="ovsbr0" />
  <forward mode="bridge"/>
  <virtualport type="openvswitch"/>
</network>
```

```xml
<interface type='network'>
  <mac address='52:54:00:c0:d5:cf'/>
  <source network='ovs-bridge'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</interface>
```

Host definition          Guest definition
Agenda

KVM Review
Virtualizing Networks with KVM

Virtualizing Disk with KVM
Virtualizing Disk with KVM

- KVM offers several methods to virtualize the host's disk, each with their own plusses and minuses
  - Files
    - Raw
    - QCOW2
  - Block Devices
    - Entire devices
    - Partitions
    - Logical Volumes
  - Directories & Network services
KVM – Virtualizing Disk with Raw Files

- Guest Linux OS
- Guest Linux OS
- Guest Linux OS

QEMU

/var/.../guest1.img
/var/.../guest2.img
/var/.../guest3.img

Linux Kernel

/var/lib/libvirt
/bin /lib /var
/
sda
KVM – Virtualizing Disk with Raw Files

- +++
  - Relatively straightforward to set up and work with
  - No need to ask the storage group to clone disks for you
  - Less overhead than QCOW2

- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - Not as fast as block devices

- Provides a good starting point for gaining familiarity with KVM

```xml
<disk type='file' device='disk'>
  <driver name='qemu' type='raw'/>
  <source file='/var/lib/libvirt/images/s12sp0.img'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition
KVM – Virtualizing Disk with QCOW2 Files

Guest Linux OS
QEMU
vda

/sda

Shared Backing Image

Internal File Snapshots

/sda

/var/.../guest1.qcow2
<1 preconfig>
<2 preSP4>

/var/.../guest2.qcow2
/var/.../guest3.qcow2
/var/.../guest_backing.qcow2 (ro)

/var/lib/libvirt

/bin /lib /var
KVM – Virtualizing Disk with QCOW2 Files

- +++
  - Thin Provisioning
  - Snapshots and Base Images
  - No need to ask the storage group to clone disks for you

- ---
  - Overhead
  - Base Images introduce the possibility for data loss if the base is written to
  - Not nearly as fast as block devices

- Provides extensive possibilities for cloning and extending OS images and moving them to production

```xml
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/var/lib/libvirt/images/sl12sp0.qcow2'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition
KVM – Virtualizing Disk with entire Block Devices

- QEMU
- Guest Linux OS
- vda

/vvar/lib/libvirt
/bin /lib /var

/sda
/sdb
/sdc
/sdd
KVM – Virtualizing Disk with entire Block Devices

- +++
  - Fast
  - Flashcopy et all do exactly what we want them to do
  - Compatible with existing DR processes
- ---
  - Thick provisioning ( lots of 0x0000 on disk )
  - Requires interaction with Storage admins

- Put Databases here

```xml
<disk type='block' device='disk'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/sdb'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition
KVM – Virtualizing Disk with Partitions

Linux Kernel

/QEMU

Guest Linux OS
vda

Guest Linux OS
vda

Guest Linux OS
vda

/bin
/lib
/var

/sda

/var/lib/libvirt

/sdb1

/sdb2

/sdb3
KVM – Virtualizing Disk with Partitions

- +++
  - Fast
  - Flashcopy et all do mostly what we want them to do
  - Compatible with existing DR processes

- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - Requires interaction with Storage admins
  - Partition table maintenance can be tricky and require a boot to refresh

- Make efficient use of very large storage volumes

```xml
<disk type='block' device='disk'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/sdb1'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition
KVM – Virtualizing Disk with Logical Volumes

Linux Kernel

VolGroup00

LogVol00
LogVol01
LogVol02

VolGroup00

sda

/bin /lib /var

/var/lib/libvirt

vda

Guest Linux OS
QEMU

Guest Linux OS
QEMU

Guest Linux OS
QEMU

KVM – Virtualizing Disk with Logical Volumes

VolGroup00

LogVol00
LogVol01
LogVol02

VolGroup00

sda

/bin /lib /var

/var/lib/libvirt

vda

Guest Linux OS
QEMU
KVM – Virtualizing Disk with Logical Volumes

- +++
  - Fast
  - Linux LVM has its own snapshot function
  - Very Flexible and Dynamic, manageable from within libvirt

- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - LVM snapshots are kinda slow and drive lots of host IO

- Most flexible and performant disk management approach

```
<disk type='block' device='disk'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/VolGroup00/LogVol00'/>
  <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition
KVM – Virtualizing Disk with Directories and Services

- QEMU with vda
- Guest Linux OS
- /bin /lib /var
- /var/lib/libvirt
- /var/lib/libvirt
- ../guest1
- NIC
- eth0 – 10.0.0.100
- sda
- NFS
- iSCSI
KVM – Virtualizing Disk with Directories and Services

- Flexible
- Diskless Guests

Guest definition

```xml
<disk type='network' device='lun'>
  <driver name='qemu' type='raw'/>
  <source protocol='iscsi' name='iqn.2013-07.com.example:iscsi-nopool/1'>
    <host name='example.com' port='3260'/>
  </source>
  <auth username='myuser'>
    <secret type='iscsi' usage='libvirtiscsi'/>
  </auth>
  <target dev='sdb' bus='scsi'/>
</disk>

<filesystem type='mount' accessmode='passthrough'>
  <driver type='path' wrpolicy='immediate'/>
  <source dir='/export/to/guest'/>
  <target dir='/import/from/host'/>
  <readonly/>
</filesystem>
```
Agenda

KVM Review
Virtualizing Networks with KVM
Virtualizing Disk with KVM

Questions?