Deploy and test oVirt using nested virtualization environments

Mark Wu
wudxw@linux.vnet.ibm.com
Agenda

- Nested KVM
- Kickstart & Cobbler
- Kickstart files for VMs
- Install and clone oVirt VMs
- Integration test with Igor
- Q & A
Nested Virtualization

- Running multiple unmodified hypervisors with their associated unmodified VM’s

Why?

- Operating systems are already hypervisors (Windows 7 with XP mode, Linux/KVM)
- To be able to run other hypervisors in clouds
- Live migration of hypervisors and their vms
- Testing, demonstrating, debugging hypervisors and virtualization setups
Nested VMX

- Merged in kernel 3.1
- No hardware support
- Multiplex hardware
- Follows the “trap and emulate” model

Flow:
- L0 intercepts the 'vmlaunch' instruction which L1 execute to run L2
- L0 generates VMCS0-2 by merging VMCS1-2 and VMCS0-1 and then launches L2
How to enable nested KVM? (For VMX)

- Enable the nested switch of kvm_intel.ko
  - enable it at runtime
    - modprobe -r kvm_intel
    - modprobe kvm_intel nested=1
  - Verify
    - $cat /sys/module/kvm_intel/parameters/nested => Y
  - Persist the change
    - echo “options kvm-intel nested=1”>/etc/modprobe.d/kvm-intel.conf

- Qemu command line
  - qemu -cpu host
  - qemu -cpu qemu64,+vmx
How to enable nested KVM? (cont'd)

- **Libvirt XML**
  - Use host CPU model
    ```xml
    <cpu mode='host-model'/>
    ```
  - Specify a CPU model
    ```xml
    <cpu match='exact'>
    <model>core2duo</model>
    <feature policy='require' name='vmx'/>
    </cpu>
    ```

- **Verify in guest**
  - `cat /proc/cpuinfo | grep vmx`
  - `qemu-kvm` should not complain about no access to KVM kernel module
Kickstart & Cobbler

- **Kickstart**
  - Using 'answer file' to installer to do fully automatic installations
  - Used with PXE

- **Cobbler**
  - A provisioning (installation) and update server
  - Supports deployments via:
    - PXE (network booting)
    - Virtualization (Xen, QEMU/KVM, or Vmware) (by koan)
    - Re-installs of existing Linux systems (by koan)

- **Update server features**
  - yum mirroring
  - Integrate mirrors with kickstart
Cobbler concepts

- Distributions
  - contain information about what kernel and initrd are used, plus metadata
- Profiles
  - associate a Distribution with a kickstart file and optionally customize the metadata further.
- Systems
  - associate a MAC, IP, and other networking details with a profile
- Repositories
  - contain yum mirror information
Setup cobbler server for oVirt

```bash
mount -o loop Fedora-18-x86_64-DVD.iso /mnt
cobbler import --path=/mnt --name=fedora18 --arch=x86_64

cobbler repo add --name=ovirt-3.2
--mirror=http://resources.ovirt.org/releases/3.2/rpm/Fedora/18/

cobbler repo add --name=glustefs
--mirror=http://download.gluster.org/pub/gluster/glusterfs/qa-
releases/3.4.0alpha/Fedora/fedora-18/x86_64/

cobbler repo add --name=fedora18-everything
--mirror=http://mirrors.163.com/fedora/releases/18/Everything/x86_64/os --mirror-locally=N

cobbler repo add --name=fedora18-updates
--mirror=http://mirrors.163.com/fedora/updates/18/x86_64/ --mirror-locally=N

cobbler reposync

cobbler profile add --name=fedora18-engine --distro=fedora18-x86_64 --virt-ram=2048
--virt-type=qemu --virt-file-size=20 --virt-cpus=2 --virt-path=/var/lib/libvirt/images/
--virt-disk-driver=qcow2 --virt-bridge=virbr-ovirt --repos="ovirt-3.2 fedora18-everything
fedora18-updates" --kickstart=/var/lib/cobbler/kickstarts/engine.ks
...
```
## Profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Distro</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora18-engine</td>
<td>fedora18-x86_64</td>
<td>Edit  Copy  Rename  Delete  View Kickstart</td>
</tr>
<tr>
<td>fedora18-storage</td>
<td>fedora18-x86_64</td>
<td>Edit  Copy  Rename  Delete  View Kickstart</td>
</tr>
<tr>
<td>fedora18-vdsm</td>
<td>fedora18-x86_64</td>
<td>Edit  Copy  Rename  Delete  View Kickstart</td>
</tr>
<tr>
<td>fedora18-x86_64</td>
<td>fedora18-x86_64</td>
<td>Edit  Copy  Rename  Delete  View Kickstart</td>
</tr>
</tbody>
</table>

**Filter**

- **on**

- **Add**
Kickstart file for oVirt engine

- Key packages:
  - ovirt-engine
  - firefox
  - spice-xpi

- Prepare an answer file in %post section

```
cat >/home/ovirtadm/engine/answer <<EOF
[general]
...
HOST_FQDN=ENGINE_FQDN  # Replace with hostname before running engine-setup
AUTH_PASS=ovirt
DC_TYPE=NFS
DB_REMOTE_INSTALL=local
DB_LOCAL_PASS=ovirt
NFS_MP=/var/lib/export/iso
...
EOF
```
Kickstart file for VDSM

- Key packages
  - vdsmd
  - vdsmd-cli
  - vdsmd-gluster
  - NetworkManager

- Inject ssh public key

```bash
cat >> .ssh/authorized_keys << END_AUTHORIZED_KEYS
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQmD6md ...
... (replace it with the ssh public key on your management host.
END_AUTHORIZED_KEYS
chmod 600 .ssh/authorized_keys

if -x /usr/sbin/selinuxenabled && /usr/sbin/selinuxenabled; then
  chcon -R -h -t home_ssh_t .ssh
fi
```
Kickstart file for storage server

- Key packages
  - nfs-utils
  - targetcli
- Create volume group for NFS

```
part pv.02 --size=83920
volgroup vg_iscsi pv.02
logvol /ovirt-iso --fstype ext4 --vgname=vg_nfs --name=lv1 --size=20480
logvol /ovirt-data --fstype ext4 --vgname=vg_nfs --name=lv2 --size=20480 --grow
```

- Enable NFS service

```
chown 36:36 /ovirt-data /ovirt-iso
echo "/ovirt-data  *(rw)" > /etc/exports
echo "/ovirt-iso *(rw)" >> /etc/exports
systemctl enable targetcli.service
systemctl start targetcli.service
```
Kickstart file for storage server (Cont'd)

- Create volume group for iscsi target
- Setup iscsi target via LIO
  - LIO is the standard and unified SCSI Target in Linux
  - Supports different fabrics as the frontend of the SCSI target by fabric modules:
    - Fibre Channel, FcoE, iSCSI, vHost, etc.
  - Backstores implement methods of accessing data on devices.
    - block, fileio, pscsi, ramdisk, etc
- Configured by targetcli.

```bash
targetcli "/backstores/block create name=block01 dev=/dev/iscsi/lv01"
targetcli "/iscsi create wwn=iqn.2013-04.org.ovirt.storage-server:t01"
targetcli "/iscsi/iqn.2013-04.org.ovirt.storage-server:t01/tpg1/luns create storage_object=/backstores/block/block01"
targetcli "/iscsi/iqn.2013-04.org.ovirt.storage-server:t01/tpg1/portals create"
... targetcli "/ saveconfig"
```
• Make use of libvirt's NATed virtual network.
• Set mac, ip, and hostname mapping for VMs

```xml
<network>
  <name>ovirt-test</name>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr-ovirt' stp='on' delay='0' />
  <mac address='52:54:00:BA:19:DF'/>
  <domain name='test.ovirt.org'/>
  <ip address='192.168.247.1' netmask='255.255.255.0'>
    <dhcp>
      <host mac='52:54:00:70:9e:33' name='engine1' ip='192.168.247.2' />  
      <host mac='52:54:00:9a:82:be' name='storage1' ip='192.168.247.3' />  
      <host mac='52:54:00:e1:dc:f4' name='host1' ip='192.168.247.4' />  
      <host mac='52:54:00:1b:3a:a2' name='host2' ip='192.168.247.5' />
    </dhcp>
  </ip>
</network>
```
Create oVirt VMs

- **Using virt-install**

  ```
  virt-install --name engine-base --vcpus 2 --ram 2048
  --disk path=/var/lib/libvirt/engine-base.qcow2,format=qcow2,bus=virtio,cache=none
  -w network=ovirt-test --accelerate --location $(INSTALL_TREE)
  --os-variant fedora18 --extra-args ks=http://cobbler-server/engine.ks --noreboot
  ```

- **Using koan**

  ```
  koan --server 192.168.247.1 --virt --profile=fedora18-engine --virt-type=kvm --qemu-disk-type=virtio --virt-name=engine-base
  ```

  ```
  koan --server 192.168.247.1 --virt --profile=fedora18-vdsm --virt-type=kvm --qemu-disk-type=virtio --virt-name=vdsm-base
  ```

- **Update CPU model to enable nested KVM for VMs running VDSM**
Clone oVirt VMs

- Clone images based on the base Vms
  
  ```
  qemu-img create -f qcow2 -b /var/lib/libvirt/image/engine-base /var/lib/libvirt/engine-foo.qcow2
  ```

- Append a new dhcp entry (mac, ip and hostname) to virtual network for the new vm.

- Clone new vm
  
  ```
  virt-clone --connect qemu:///system -o engine-base -n engine1 -f /var/lib/libvirt/image/engine-foo.qcow2 --preserve-data --mac MAC
  ```

- Update configurations inside guests
  - guest-mount
  - Run guest commands via ssh, like engine-setup
• Create VM network and base Vms

ovirt-setup.py setup
ovirt-setup.py create-base --type engine --name engine-base --profile fedora18-engine
ovirt-setup.py create-base --type vdsm --name vdsm-base --profile fedora18-vdsm
ovirt-setup.py create-base --type storage --name storage-base --profile fedora18-storage

• Clone a new oVirt setup

ovirt-setup.py clone-vm --base engine-base --name engine1
ovirt-setup.py clone-vm --base vdsm-base --name vdsm1
ovirt-setup.py clone-vm --base vdsm-base --name vdsm2
ovirt-setup.py clone-vm --base storage-base --name storage1
oVirt cluster level tests

- ovirt-engine-sdk
  - an auto-generated python API which uses REST-API to perform operations against ovirt-engine

Examples

- Create iSCSI Data Center
  ```python
  if api.datacenters.add(params.DataCenter(name=DC_NAME, storage_type='iscsi', version=VERSION)):
    print 'iSCSI Data Center was created successfully'
  ```

- Create Cluster
  ```python
  if api.clusters.add(params.Cluster(name=CLUSTER_NAME, cpu=params.CPU(id=CPU_TYPE), data_center=api.datacenters.get(DC_NAME), version=VERSION)):
    print 'Cluster was created successfully'
  ```
- Used for auto testing for oVirt Node/REHV-H
- Automate deployment, installation
  - Installation via PXE
  - Import ISO using livecd-to-pxeboot
  - Create profile and system in Cobbler.
  - Run installation or update
- Works for virtual guests and real hardware
  - Libvirt for virtual guests
- Testsuite life-cycle management
Proposed solution for oVirt functional test

- Expand Igor's test plan for oVirt functional tests
  - Allow specify the specs of test environment
  - Allow creating test vm based on a template
  - Allow creating vm template if it doesn't exist
  - Skipped system installation
- Ship igor-client in ovirt-engine VM
  - Associate test cases with host name
- Run test suites based on oVirt engine SDK
  - Run update rpm packages as a 'setup' test
Test flow with Igor using nested KVM

1. Set up cobbler server
2. Update repo with test packages
3. Submit test plan
4. Clone vm for test
5. Download testsuites and Run
6. Update oVirt packages

Test Environment

Physical Hosts

Installation medias

Kickstart Files

oVirt RPMS Repo

Test Environment

Storage Server

Fedora/RHEL

oVirt-Node/RHEV-H

Nested KVM

Nested KVM

Test Environment

oVirt engine

virt-install

koan

Cobbler

virt-clone

Libvirt
Resources

- The Turtles Project: Design and Implementation of Nested Virtualization
- http://fedoraproject.org/wiki/Anaconda/Kickstart
- http://www.cobblerd.org/
- http://www.ovirt.org/Testing/PythonApi
- Automated Testing of oVirt Node
- https://github.com/wudx05/ovirt-setup
Thanks for Listening!

Q & A

http://www.ovirt.org
wudxw@linux.vnet.ibm.com