Group B Assignment No. 11

Title: KVM Installation

Aim:- To install KVM and create a virtual machine**.**

Objective:- To study KVM and its installation process**.** To create a VM using VMM and to execute virsh commands.

Theory:-

A kernel-based virtual machine (KVM) is a virtualization infrastructure built for Linux OS and designed to operate on x86-based processor architecture.KVM is developed by Red Hat Corporation to provide a virtualization solution and services on the Linux operating system platform. KVM is designed over the primary Linux OS kernel.KVM is a type of hypervisor that enables, emulates and provides for the creation of virtual machines on operating systems. These machines are built on top of the Linux kernel, using operating systems such as Linux, Ubuntu and Fedora. KVM can be installed on all x86 processors and provide separate instruction set extensions for Intel and AMD processors.
KVM supports multiple different guest operating system images including Linux Kernel, Windows, BSD and Solaris. It also allocates separate virtualized computing resources for each virtual machine such as the processor, storage, memory, etc.

KVM Features:
Leverage HW virtualization support
– VT-x/AMD-V, EPT/NPT, VT-d/IOMMU
• CPU and memory overcommit
• High performance paravirtual i/o
• Hotplug (cpu, block, nic)
• SMP guests
• Live migration
• Power Management
• NUMA
• PCI Device Assignment and SR-IOV
• Page sharing
• SPICE
• KVM autotest

Installing KVM
KVM only works if your CPU has hardware virtualization support – either Intel VT-x or AMD-V. To determine whether your CPU includes these features, run the following command:

```
egrep -c ‘(svm|vmx)’ /proc/cpuinfo
```

A 0 indicates that your CPU doesn’t support hardware virtualization, while a 1 or more indicates that it does. You may still have to enable hardware virtualization support in your computer’s BIOS, even if this command returns a 1 or more.

Use the following command to install KVM and supporting packages. Virt-Manager is a graphical application for managing your virtual machines — you can use the kvm command directly, but libvirt and Virt-Manager simplify the process.

```
sudo apt-get install qemu-kvmlibvirt-bin bridge-utilsvirt-manager
```

After running the following command you should see an empty list of virtual machines. This indicates that everything is working correctly.

```
virsh -c qemu:///system list
```

Creating Virtual Machines

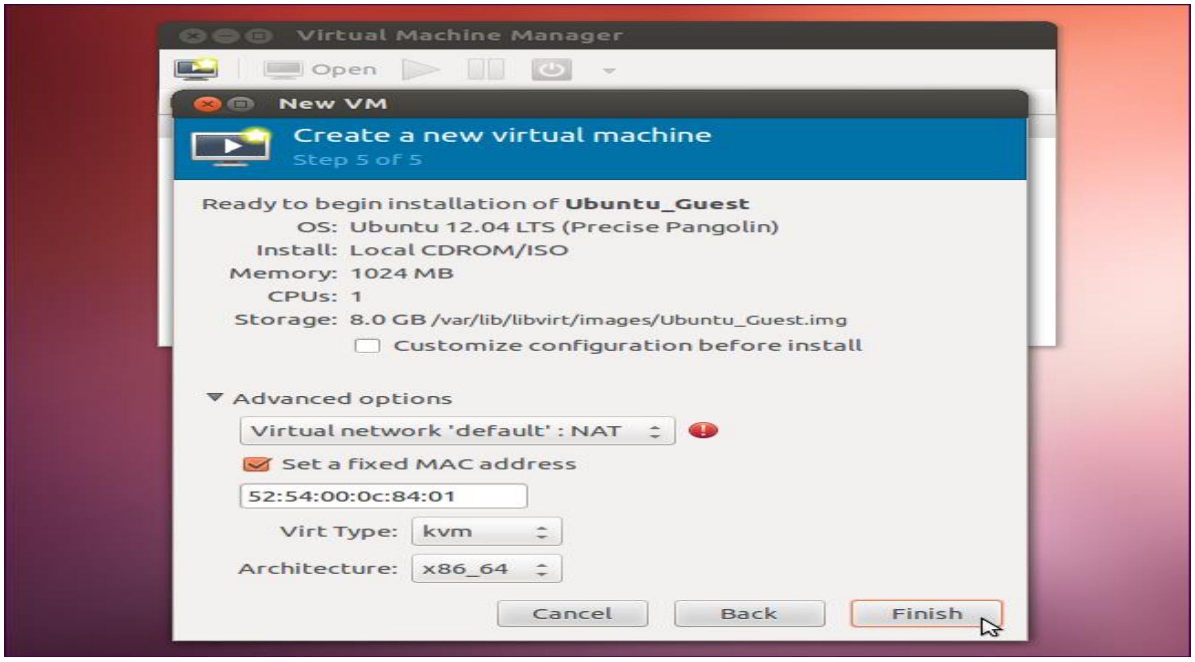
Once you’ve got KVM installed, the easiest way to use it is with the Virtual Machine Manager application. You’ll find it in your Dash.

Click the Create New Virtual Machine button on the toolbar and the Virtual Machine Manager will walk you through selecting an installation method, configuring your virtual machine’s virtual hardware, and installing your guest operating system of choice.

The process will by familiar if you’ve ever used VirtualBox, VMware, or another virtual machine application. You can install from a disc, ISO image, or even a network location.

To assign more than 2GB of memory to a virtual machine, you’ll need a 64-bit Linux kernel. Systems running 32-bit kernels can assign a maximum of 2 GB of RAM to a virtual machine.

By default, KVM gives you NAT-like bridged networking – your virtual machine won’t appear on the network as its own device, but it will have network access through the host operating system. If you’re running server software in your virtual machine and want it accessible from other devices on the network, you’ll have to tweak the networking settings.



Ready to begin installation of **Ubuntu_Guest**
OS: Ubuntu 12.04 LTS (Precise Pangolin)
Install: Local CDROM/ISO
Memory: 1024 MB
CPUs: 1
Storage: 8.0 GB /var/lib/libvirt/images/Ubuntu_Guest.img
☐ Customize configuration before install
Advanced options
Virtual network 'default' : NAT
☑ Set a fixed MAC address
52:54:00:0c:84:01
Virt Type: kvm
Architecture: x86_64

After selecting your installation method, Virt-Manager will boot the guest operating system in a window. Install the guest operating system as you would on a physical machine.

Managing Virtual Machines

The Virtual Machine Manager window displays a list of your installed virtual machines. Right-click virtual machines in the window to perform actions, including starting, shutting down, cloning, or migrating them.

You can view information about the virtual machine and configure its virtual hardware by clicking the i-shaped toolbar icon in the virtual machine’s window.

Example1: Get KVM version installed on the host machine

virsh version

Output:

Compiled against library: libvir 0.9.13
Using library: libvir 0.9.13
Using API: QEMU 0.9.13
Running hypervisor: QEMU 1.2.0

Example2: Get KVM Hypervisor(Host) Memory info

virsh nodememstats

Output:

total : 8027952 KiB
free : 2772452 KiB
buffers: 264476 KiB
cached : 1677176 KiB

Example3: Get KVM Hypervisor CPU info

virsh nodecpustats

Output:

```
user: 3916550000000
system: 1183160000000
idle: 21003220000000
iowait: 655350000000
```

Note: The above numbers are in nanoseconds of time available for user/system/idle etc.
If you want we can even get individual CPU, if you have more than 1 CPU. Suppose if we want to get CPU2 details use –cpu 1 for that

virsh nodecpustats –cpu 1

Output:

```
user: 1131610000000
system: 348770000000
idle: 5141870000000
iowait: 136620000000
```

How about getting the values in percentage of total CPU available?

virsh nodecpustats –percent

Output:

```
usage: 20.3%
user: 17.5%
system: 2.8%
idle: 79.7%
iowait: 0.0%
```

Example4: How to get number of Guest Virtual machines irrespective of state such as running, save, shutdown etc.

virsh list –all

Output:

```
Id Name State
————————————————————————-
1 BaseMachine running
– bt51 shut off
– centos-64 shut off
– Clusterbase shut off
– mint1 shut off
– node.linuxnix.com shut off
– node1.linuxnix.com shut off
– node2.linuxnix.com shut off
```

To get only running machines in KVM hypervisor

virsh list

Output:

```
Id Name State
```

```
————————————————————————————-
1 BaseMachine running
```

Example5: To get all the networks available for KVM hypervisor

virsh net-list

Output:

```
Name State Autostart
—————————————————————————
default active yes
net1 active yes
net2 active yes
NewNAT active yes
```

To get info of particular network use below command

virsh net-info default

Output:

```
Name default
UUID 456a45cf-dac8-17b2-c736-e7cf287283cc
Active: yes
Persistent: yes
Autostart: yes
Bridge: virbr0
```

Get KVM Guest machine details

Example6: Get Hardware information of a KVM guest machine

virsh dominfo BaseMachine

Output:

```
Id: 1
Name: BaseMachine
UUID: 78c48a29-7c2c-4b84-8968-198f8ed17db2
OS Type: hvm
State: running
CPU(s): 1
CPU time: 32.4s
Max memory: 1048576 KiB
Used memory: 1048576 KiB
Persistent: yes
Autostart: disable
Managed save: no
Security model: apparmor
Security DOI: 0
Security label: libvirt-78c48a29-7c2c-4b84-8968-198f8ed17db2 (enforcing)
```

Example7: Get guest machine CPU details like how many vCPU’s etc.

virsh vcpucount centos-64

Output:

```
maximum config 2
maximum live 2
current config 2
current live 2
```

Example8: Get guest machine RAM details

virsh dommemstat centos-64

Output:

```
actual 3223552
rss 1009044
```

Note: The above numbers in KB, so I assigned 3GB of RAM to this machine.

Example9: List all networks available for a VM

virsh domiflist centos-64

Output:

```
Interface Type Source Model MAC
———————————————————————————————-
vnet1 network default virtio 52:54:00:cf:99:0c
```

$ virsh --connect qemu:///system

```
Connecting to uri: qemu:///system
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # define /etc/libvirt/qemu/newvm.xml
Domain newvm defined from /etc/libvirt/qemu/newvm.xml
```

Note that to list newvm, you must use 'list --inactive' or 'list --all', since list without any options will only list currently running machines.

List your VMs

Virsh allows you to list the virtual machines available on the current host:

```
yhamon@paris:/etc/libvirt/qemu$ virsh --connect qemu:///system
Connecting to uri: qemu:///system
Welcome to virsh, the virtualization interactive terminal.

virsh # list
 Id Name                 State
----------------------------------
 15 mirror               running
 16 vm2                  running
```

```
virsh # list --all
 Id Name                 State
----------------------------------
 15 mirror               running
 16 vm2                  running
  - test5                shut off
```

Define, undefine, start, shutdown, destroy VMs

The VMs you see with list --all are VMs that have been "defined" from an XML file. Every VM is configured via a XML file in /etc/libvirt/qemu.

To be able to undefine a virtual machine, it needs to be shutdown first:

```
virsh # shutdown mirror
Domain mirror is being shutdown
```

This command asks for a nice shutdown (like running shutdown in command line).

Notice: Ubuntu 10.04 server doesn't have acpid installed by default. This package needs to be installed on the guest OS before it will listen to any requests from the host.

You can also use "destroy", the more brutal way of shutting down a VM, equivalent of taking the power cable off:

```
virsh # destroy mirror
Domain mirror destroyed
```

If you have made a change to the XML configuration file, you need to tell KVM to reload it before restarting the VM:

```
virsh # define /etc/libvirt/qemu/mirror.xml
Domain mirror defined from /etc/libvirt/qemu/mirror.xml
```

Then, to restart the VM:

```
virsh # start mirror
Domain mirror started
```

Suspend and resume a Virtual Machine

Virsh allows you to easily suspend and resume a virtual machine.

```
virsh # suspend mirror
Domain mirror suspended
virsh # resume mirror
Domain mirror resumed
```

Editing the attributes of a Virtual Machine

To view VM details ,

1. **virsh # dominfo vm-name**

2. **virsh # edit vm-name**

Adding CPUs

KVM allows you to create SMP guests. To allocate two CPUs to a VM, edit your xml using above command:

```
<domain type='kvm'>
  ...
  <vcpu>2</vcpu>
  ...
</domain>

Save the file
```

Now define the VM as above.

Adding Memory

To change the memory allocation in a VM, edit your xml to have:

```
<domain type='kvm'>
  ...
  <memory>262144</memory>
  <currentMemory>262144</currentMemory>
  ...
</domain>
Save the file
```

virsh # save vm-name anyname.xml

Now define the VM as above. Keep in mind that the memory allocation is in kilobytes, so to allocate 512MB of memory, use 512 * 1024, or 524288.

Input : Commands to install KVM and commands to view the VM status.

Output: KVM is installed.

Platform: Ubuntu 12.04

Conclusion : Thus KVM is installed successfully, Virtual Machine is created and virsh commands are executed to view the VM status.

FAQs:

- What are the minimum requirements of KVM?
- How many maximum Virtual Machines can be created in KVM?
- What is the difference between KVM and VMware?
- What is the difference between KVM and Xen?