Introduction to Virtualization on Fedora with KVM

Charles Rose
charles_rose@dell.com

September 2012
Fedora Activity Day
RVCE, Bangalore

Agenda

- Jump Start
- The Hypervisor
  - Hardware-assisted virtualization
  - KVM
  - IO Virtualization
  - Qemu-kvm
- Managing virtualization
  - Libvirt

Jump Start

- Prepare System
  - Enable virtualization In BIOS
  - Install F17
  - Setup yum repositories
- Enable F17 virtualization
  - `# yum install @virtualization`
  - `# modprobe kvm-<intel|amd>`
  - `# ls -l /dev/kvm`

Create a Virtual Machine

- Create 8G disk image

```
# qemu-img create -f qcow2 vm.img 8G
```

- Install OS on vm.img from ISO

```
# qemu-kvm -cdrom Fedora-17-x86_64-DVD.iso
    -hda vm.img
```

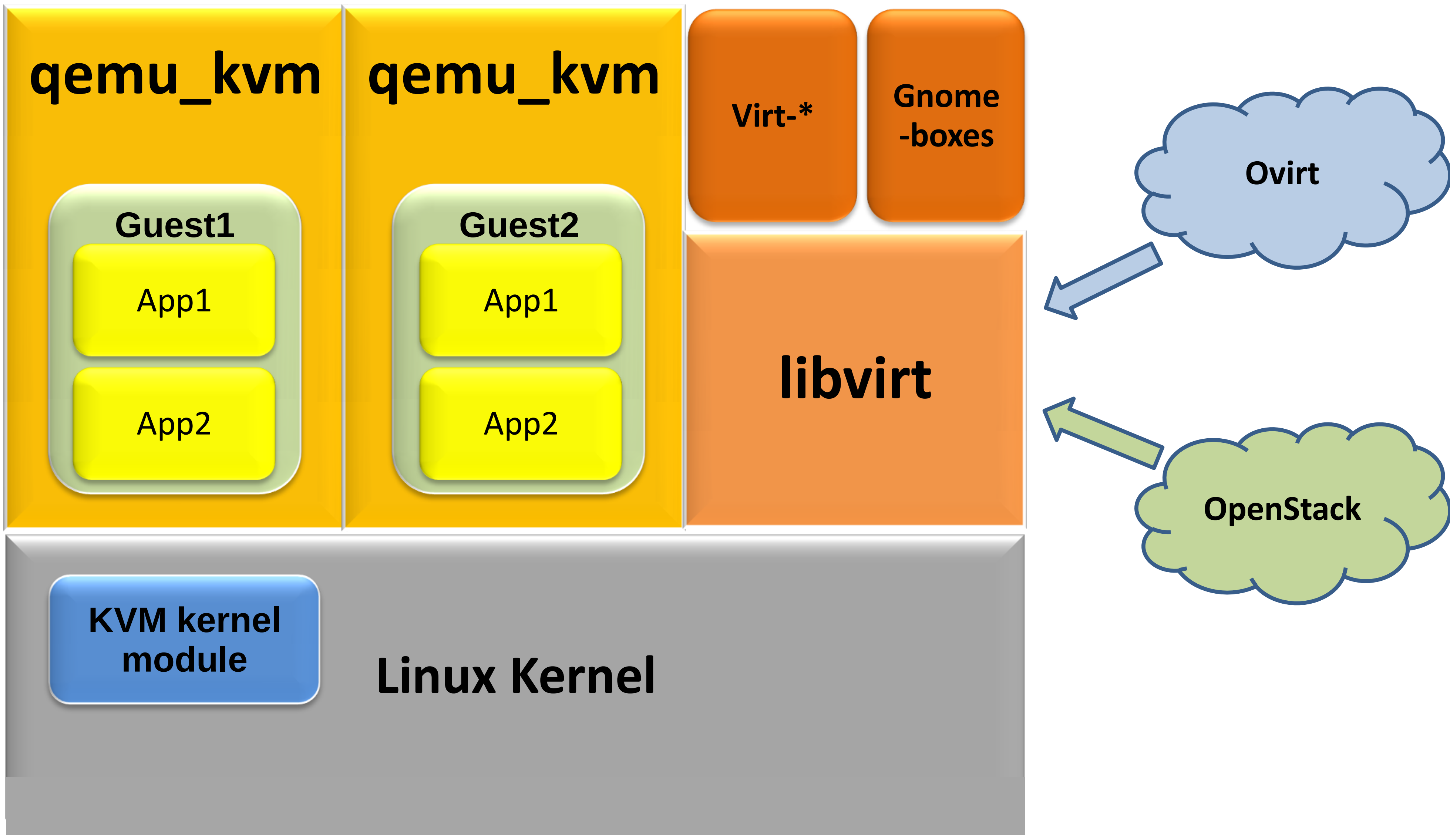
- Start the VM

```
# qemu-kvm vm.img
```

- Thats it!

  But how does it all work?

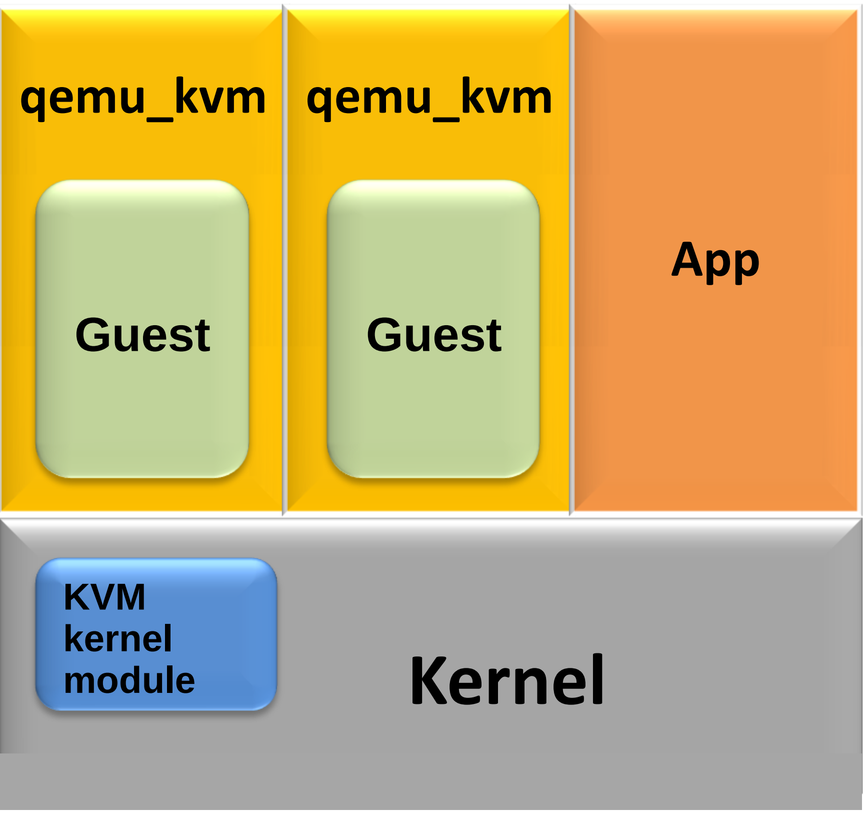
30,000' View


qemu_kvm
qemu_kvm
Guest1
App1
App2
Guest2
App1
App2
Virt-*
Gnome -boxes
libvirt
Ovirt
OpenStack
KVM kernel module
Linux Kernel

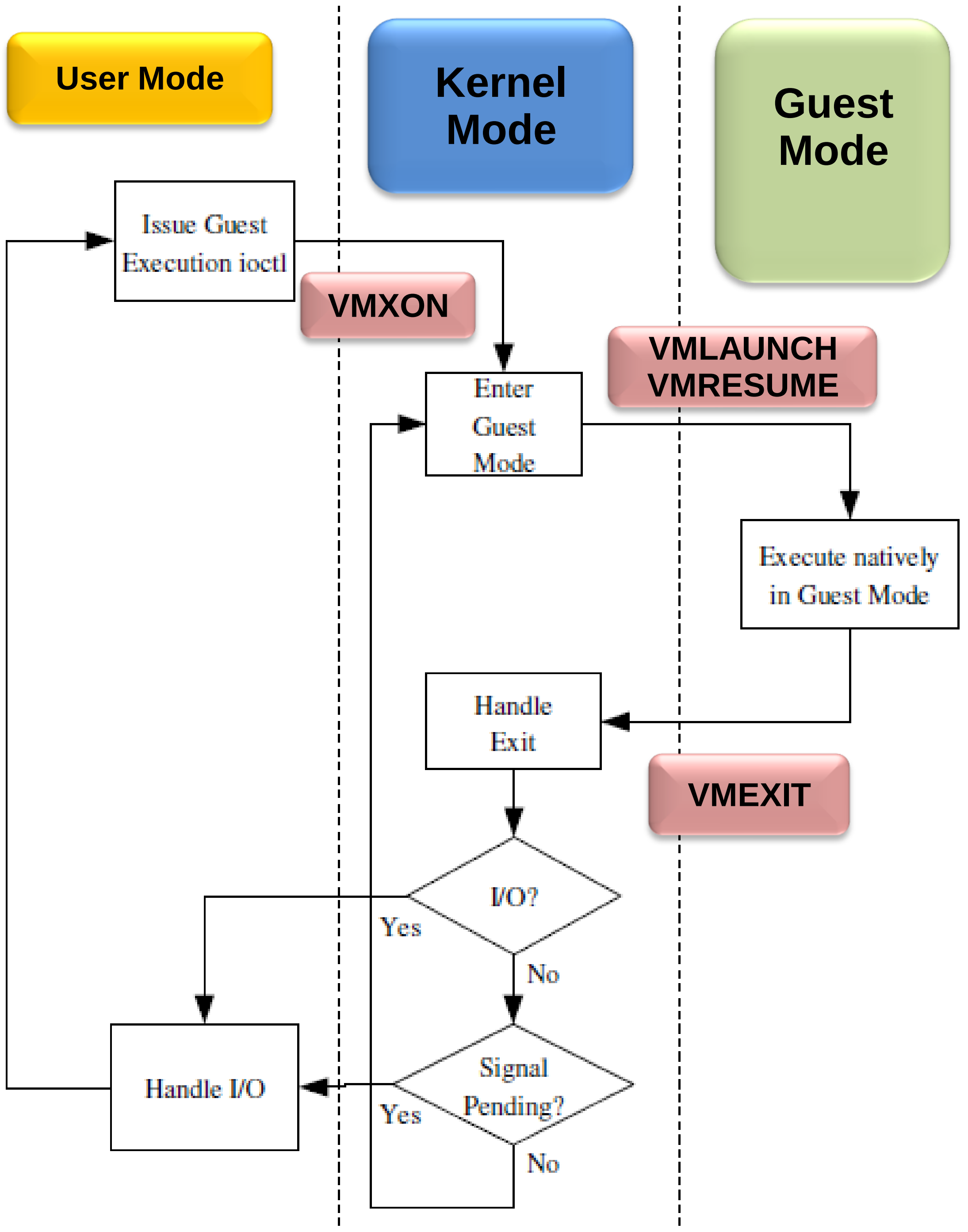
Approaches to Virtualization

- Binary Translation
  - Modify in-memory kernel/driver at runtime
- CPU Para-virtualization
  - Modify guest Kernel/Driver source
- Hardware Assisted (Full) - Intel VT-x, AMD-V
  - New operating modes: VMX root, VMX non-root
  - Transition b/w modes: VM entry, VM exit

KVM - Execution

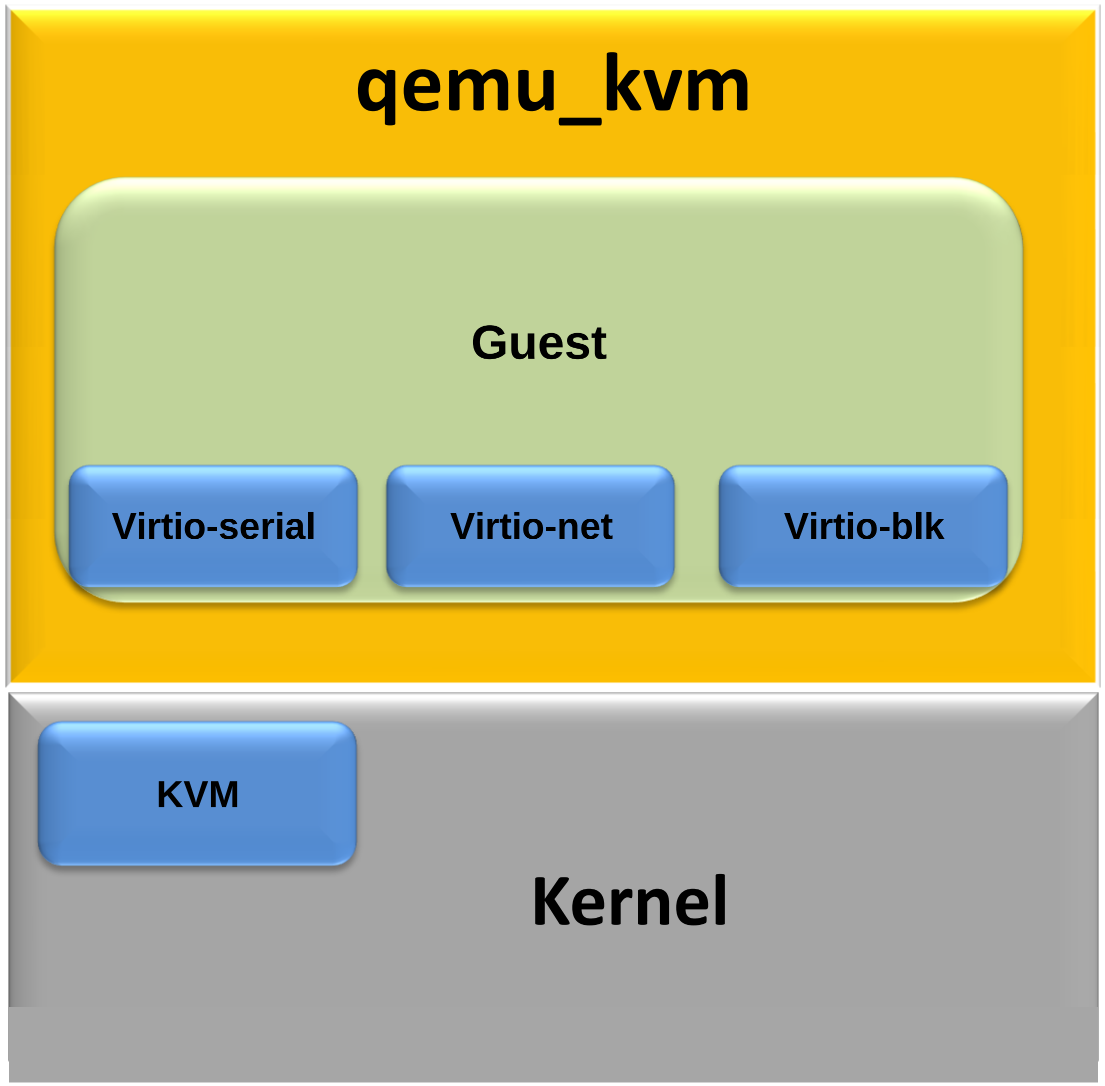


- ioctl to /dev/kvm
- qemu-kvm

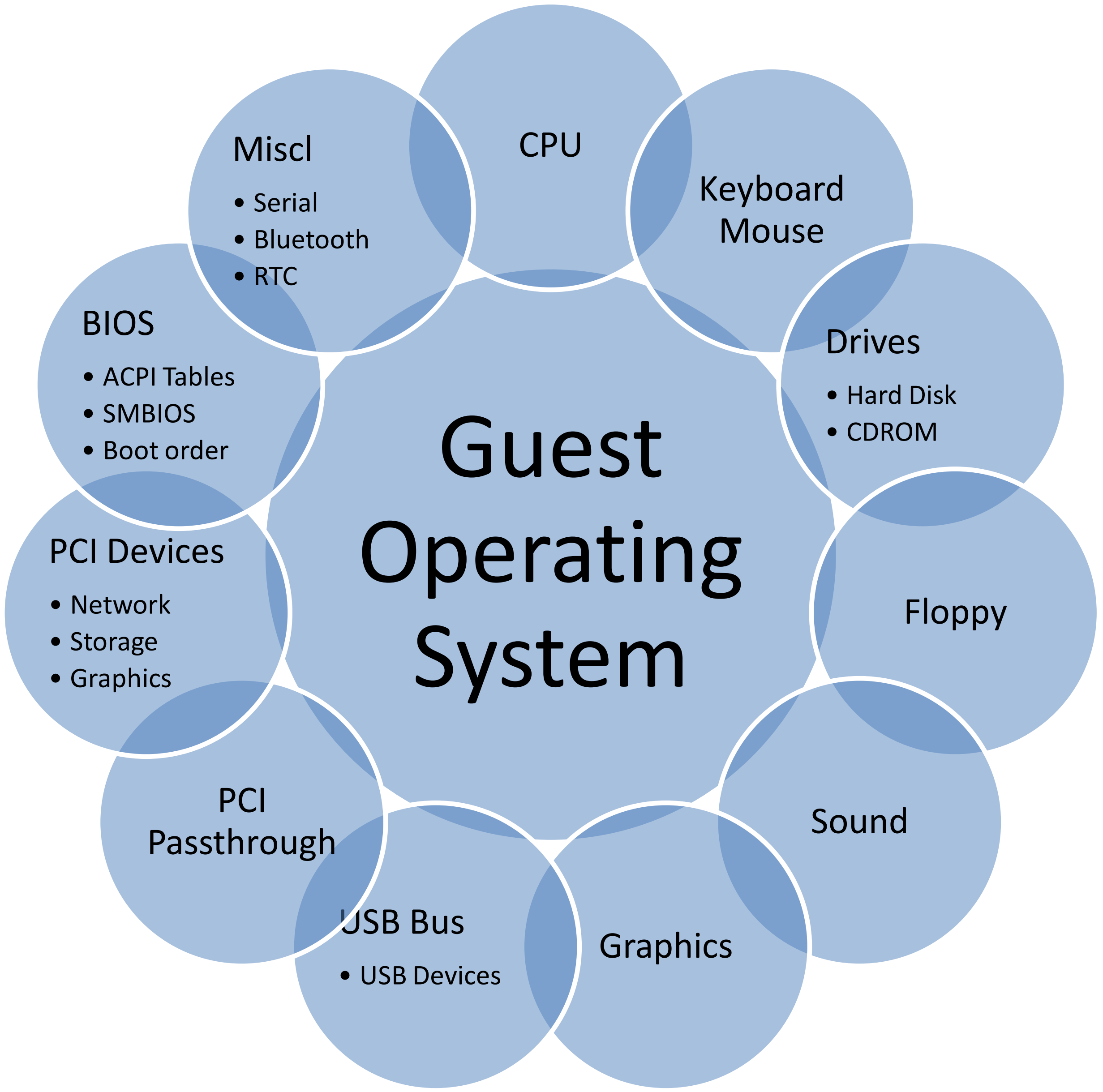


I/O Virtualization

- Para-virt I/O
- Virtio
  - Hypervisor Agnostic
  - Guest Drivers
    - Virtio-blk
    - Virtio-net
    - Virtio-baloon
    - Virtio-serial



qemu-kvm


Guest Operating System
CPU
Keyboard Mouse
Drives
• Hard Disk
• CDROM
Floppy
Sound
Graphics
USB Bus
• USB Devices
PCI Passthrough
PCI Devices
• Network
• Storage
• Graphics
BIOS
• ACPI Tables
• SMBIOS
• Boot order
Miscl
• Serial
• Bluetooth
• RTC

Qemu-kvm command line

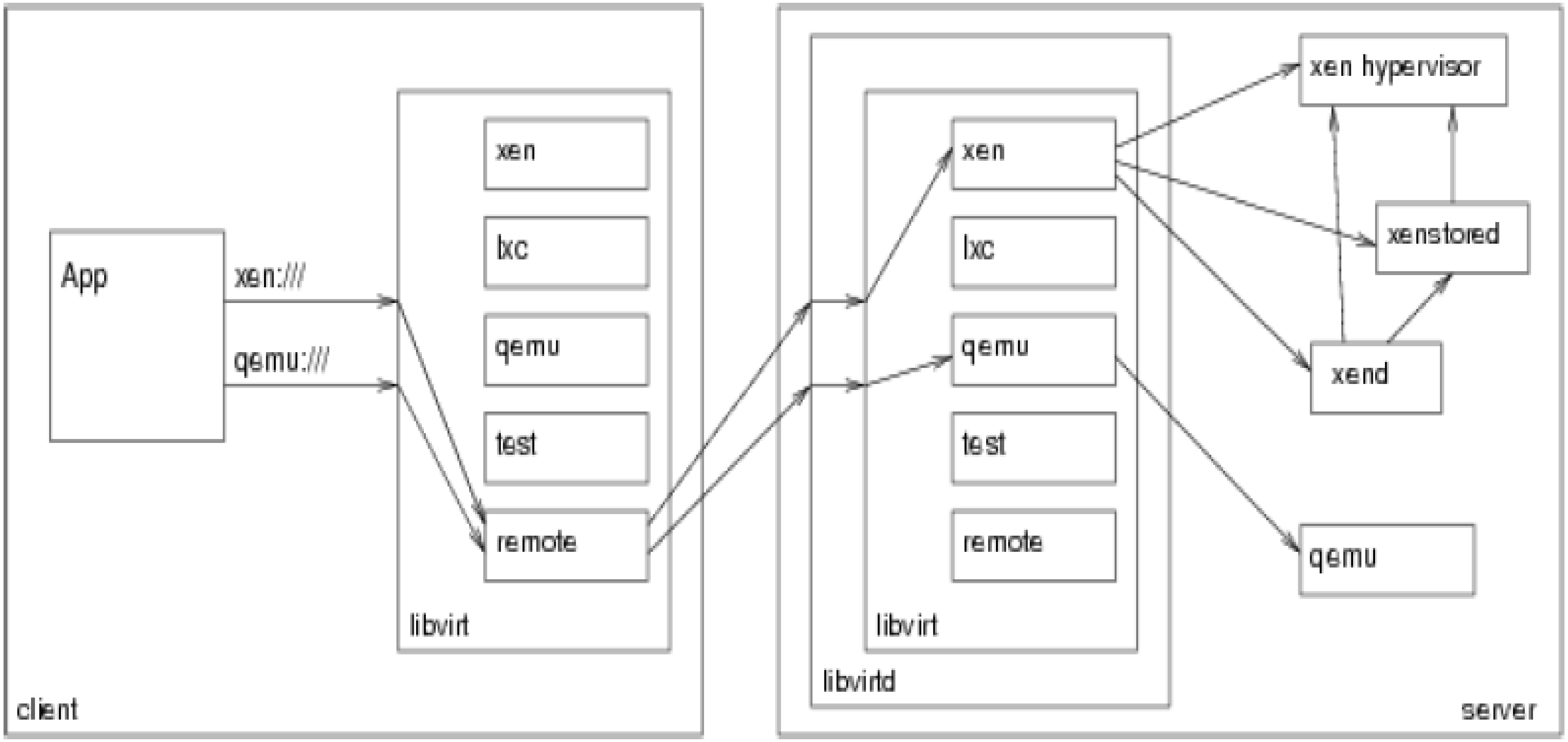
- /usr/bin/qemu-kvm -S -M pc-0.15 -enable-kvm -m 1024 -smp 1,sockets=1,cores=1,threads=1 -name devbox -uuid a073c70f-36bc-08c2-2a65-2cbdc30b95bc -nodefconfig -nodefaults -chardev socket,id=charmonitor,path=/var/lib/libvirt/qemu/devbox.monitor,server,nowait -mon chardev=charmonitor,id=monitor,mode=control -rtc base=utc -no-shutdown -device piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -device virtio-serial-pci,id=virtio-serial0,bus=pci.0,addr=0x5 -drive file=/var/lib/libvirt/images/devbox.img,if=none,id=drive-virtio-disk0,format=raw -device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x6,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 -netdev tap,fd=24,id=hostnet0,vhost=on,vhostfd=25 -device virtio-net-pci,netdev=hostnet0,id=net0,mac=52:54:00:3f:33:d3,bus=pci.0,addr=0x3 -chardev pty,id=charserial0 -device isa-serial,chardev=charserial0,id=serial0 -chardev spicevmc,id=charchannel0,name=vdagent -device virtserialport,bus=virtio-serial0.0,nr=1,chardev=charchannel0,id=channel0,name=com.redhat.spice.0 -device usb-tablet,id=input0 -spice port=5900,addr=127.0.0.1,disable-ticketing -vga qxl -global qxl-vga.vram_size=67108864 -device intel-hda,id=sound0,bus=pci.0,addr=0x4 -device hda-duplex,id=sound0-codec0,bus=sound0.0,cad=0 -device virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x7

Making it easy

- virsh
- virt-install
- virt-manager
- virt-viewer
- …
- Made possible through abstraction to qemu_kvm
  - libvirt

- # service start libvirtd
- # virt-install --name myvm --prompt
- # virsh shutdown myvm
- # virsh start myvm
- # virt-viewer myvm
- # virt-manager
- # gnome-boxes

Libvirt – Abstract it all!



- # service start libvirtd
- # virsh --connect qemu:///system start devbox

Domain Configuration
# virsh dump myvm

```
<domain type='kvm'>
  <name>devbox</name>
  <uuid>a073c70f-36bc-08c2-2a65-
      2cbdc30b95bc</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory
      unit='KiB'>1048576</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-
      0.15'>hvm</type>
    <boot dev='hd'/>
  </os>
<clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
 <devices>
    <emulator>/usr/bin/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source
        file='/var/lib/libvirt/images/devbox.img'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000'
        bus='0x00' slot='0x06' function='0x0'/>
    </disk>
<interface type='network'>
      <mac address='52:54:00:3f:33:d3'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
        bus='0x00' slot='0x03' function='0x0'/>
    </interface>
</devices>
</domain>
```

References

- www.linux-kvm.org
- wiki.qemu.org/KVM
- libvirt.org
- Native Linux KVM tool
  - github.com/penberg/linux-kvm/tree/master/tools/kvm
- Datacenter Management of KVM hypervisors
  - www.ovirt.org
- Openstack and libvirt
  - wiki.openstack.org/LibvirtAPI

Backup

Thin provisioning

- Create a thinly provisioned vmthin.img based on vm.img
- # qemu-img –b vm.img –f qcow2 vmthin.img
- # qemu-kvm vm.img
- # qemu-kvm vmthin.img